ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

М А Т Е Р І А Л И

XXII щорічного міждисциплінарного семінару

«СТУДЕНТСЬКІ РОБОТИ ЗА НАУКОВОЮ ТЕМАТИКОЮ КАФЕДРИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ»

25 листопада 2025 року

Полтава – 2025

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

ГОЛОВА ОРГКОМІТЕТУ:

УТКІН Юрій – к.т.н., доцент, завідувач кафедри інформаційних систем та технологій, доцент кафедри

ЗАСТУПНИК ГОЛОВИ ОРГКОМІТЕТУ:

СЛЮСАРЬ Ігор – к.т.н., доцент, доцент кафедри

СКЛАДИ ОРГКОМІТЕТУ:

Копішинська Олена – к.ф.–м.н., доцент, професор кафедри;
Одарущенко Олег – д.т.н., професор, професор кафедри;
Поночовний Юрій – д.т.н., професор, професор кафедри;
Вадим СЛЮСАР – д.т.н., професор, професор кафедри;
Флегантов Леонід – к.ф.–м.н., доцент, професор кафедри;
Вакуленко Юлія – к.с.–г.н., доцент, доцент кафедри;
Одарущенко Олена – к.т.н., доцент, доцент кафедри;
Протас Надія – к.с.–г.н., доцент, доцент кафедри;
Наталія ПАНАСЕНКО – к.е.н., доцент кафедри;
Антон ДІДЕНКО – асистент кафедри;
Наталія САЗОНОВА – асистент кафедри;
Олександр ТИЩЕНКО – асистент кафедри;
Марк ФЕДОРЧЕНКО – асистент кафедри.

Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій: матеріали XXII щорічного міждисциплінарного семінару, 25 листопада 2025 р. Полтава: ПДАУ, 2025 р. 124 с.

У збірнику надруковані матеріали міждисциплінарного семінару студентських робіт за науковою тематикою кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Тези наводяться без змін та редагування. Відповідальність за зміст та редакцію тез несуть автори та наукові керівники.

Для здобувачів вищої освіти та науково–педагогічних працівників.

ЗМІСТ

Азарова Катерина, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Технологія виробництва і переробки продукції тваринництва»,
Науковий керівник – к.с.-г.н., доцент Протас Надія

ВПРОВАДЖЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ В ТВАРИННИЦТВІ

У сучасному тваринництві постійно зростає потреба в інноваціях: фермерські підприємства прагнуть підвищити продуктивність, зменшити витрати, покращити умови утримання тварин і забезпечити їхній добробут. Водночас із розвитком цифрових технологій набуває все більшої значущості штучний інтелект (ШІ), який відкриває нові можливості для трансформації тваринницьких процесів. Дослідимо переваги впровадження штучного інтелекту в тваринництві.

Сьогодні розумне тваринництво використовує такі технології, як Інтернет речей (IoT) і штучний інтелект (ШІ), для одночасного підвищення продуктивності сільського господарства та оптимізації використання робочої сили й виробничих процесів у галузі. Застосування ШІ дозволяє автоматизувати моніторинг здоров'я та поведінки тварин, оптимізувати годівлю та ресурси ферми, прогнозувати продуктивність і якість продукції, а також підвищити загальну ефективність фермерських господарств. Розглянемо більш детально зазначені переваги.

Автоматичний моніторинг здоров'я та поведінки тварин. Однією з ключових переваг ШІ в тваринництві є здатність до систематичного моніторингу поведінки та стану здоров'я тварин. Наприклад, системи відеоспостереження з використанням алгоритмів комп'ютерного зору можуть ідентифікувати аномалії в русі, позах чи активності тварин – це дозволяє виявляти потенційні проблеми, такі як травми, хвороби або стрес, на ранніх стадіях. Наразі практикам запропоновано рамкову систему моніторингу для ферм з великою рогатою худобою, яка на базі відеозаписів визначає «відхилення» у поведінці тварин. Також українські науковці досліджують перспективи використання ШІ для аналізу поведінки тварин, що відкриває шляхи для ранньої діагностики хвороб та покращення добробуту тварин. Прикладом промислового рішення є платформа VetVise, яка використовує камери та ШІ для моніторингу здоров'я корів, птиці чи свиней у реальному часі, що дозволяє фермерам оперативно реагувати на зміну стану тварин. Загалом автоматичний моніторинг забезпечує більш точні дані, ніж традиційні методи (ручні огляди, періодичні заміри), і значно підвищує ефективність ветеринарного контролю [1-3].

Оптимізація годівлі та ресурсів ферми. Штучний інтелект здатен оптимізувати годівлю тварин, використовуючи моделі, які аналізують дані про споживання кормів, вагу, виробничі показники, поведінку та умови утримання. Це дозволяє створити індивідуальні або групові раціони, які максимально відповідають потребам тварин, знижуючи перевитрату кормів та підвищуючи конверсію корму. Такі підходи не лише економлять ресурси, але й сприяють підвищенню продуктивності. Крім того, ШІ може управляти іншими ресурсами ферми: наприклад, на основі даних з датчиків щодо клімат-контролю (температури, вологості), вентиляції, освітлення, з'являється додаткова можливість підтримувати оптимальні умови утримання тварин у реальному часі. Ця концепція узгоджується з загальними тенденціями «розумних ферм», де системи автоматичного управління допомагають фермерським

господарствам працювати ефективніше. Такий підхід також зменшує навантаження на персонал: замість ручного налаштування систем клімат-контролю, фермери можуть покладатися на алгоритми, які адаптуються до змін у середовищі та поведінці тварин [4].

Прогнозування продуктивності та якості продукції. Штучний інтелект дає змогу прогнозувати продуктивність тварин і якість продукції, наприклад, молока або м'яса, на підставі аналізу великого обсягу даних: історичних даних, біометрії, поведінкових моделей, даних про годівлю і середовище. Це надає фермерам можливість планувати виробництво та приймати стратегічні рішення: наприклад, які тварини мають найвищий потенціал, які групи тварин варто виводити, чи варто змінювати раціон годівлі, щоб покращити якість молока або інші параметри продукції. У великому систематичному огляді методів глибокого навчання для аграрного сектору розглядаються алгоритми, які буквально «бачать» тренди в даних про тварин та прогнозують майбутні показники продуктивності. Прогнозування також може допомогти в управлінні ризиками: завдяки ранньому виявленню потенціалу зниження продуктивності чи проблем, фермер може вжити превентивні заходи (змінити годівлю, ввести ветеринарні процедури тощо) [5].

Підвищення ефективності роботи ферми та зниження витрат. Комбінуючи моніторинг, оптимізацію годівлі та прогнозування, штучний інтелект сприяє загальній ефективності роботи ферми. Автоматизація рутинних процесів (наприклад, збір даних, аналіз, попередження) звільняє людський ресурс для стратегічних задач, знижує витрати на робочу силу та допомагає уникнути втрат через хвороби або неефективне використання кормів. Крім того, існують інноваційні рішення, як наприклад автономні роботи-«пастухи»: робот SwagBot із ШІ, розроблений дослідниками, може самостійно переганяти худобу на найкращі пасовища, оцінюючи стан ґрунту та здоров'я тварин, що знижує ерозію ґрунтів і оптимізує використання пасовищ. Також впровадження інтелектуальних систем може підвищити конкурентоспроможність фермерських підприємств, адже дозволяє працювати більш «точково», адаптуючись до змінних умов, і знижувати ризики економічних втрат. Це підтверджують аналітичні дослідження з агробіології, які підкреслюють, що ШІ є рушієм змін в аграрному секторі. Загалом, інвестиції в ШІ у тваринництві можуть окупитися через довготривале підвищення продуктивності та економію ресурсів [4, 6].

Впровадження штучного інтелекту в будь-яку галузь завжди супроводжується певними викликами. Щодо галузі тваринництва, то особливої уваги потребують питання безпеки даних, навчання персоналу, економічна доцільність і обґрунтованість фінансових інвестицій:

– дані про тварин, їхній стан, поведінку, продуктивність – це цінна інформація, і важливо забезпечити надійне її зберігання, захист від витоків або несанкціонованого доступу. Крім того, важливо, щоб алгоритми були прозорими і на них можна було покладатися, а рішення, які приймає ШІ, були зрозумілими для людини;

– для ефективного використання ШІ на фермі потрібно, щоб працівники мали навички взаємодії з новими системами: аналізували дані, розуміли сигнали систем,

інтерпретували результати. Без відповідного навчання інвестиції в ШІ можуть не принести очікувані вигоди;

– впровадження ШІ часто вимагає значних початкових вкладень: купівля обладнання (камери, датчики), програмного забезпечення, обчислювальних ресурсів. Також потрібні кошти на технічне обслуговування систем. Не всі фермерські господарства можуть дозволити собі такі інвестиції або ризикувати ними без чіткої оцінки окупності.

Таким чином, впровадження штучного інтелекту в тваринництві відкриває нові можливості для підвищення ефективності та якості виробництва. Сучасні технології дозволяють автоматично відстежувати стан здоров'я та поведінку тварин, прогнозувати ризики хвороб, оптимізувати годівлю та контроль умов утримання. Використання ШІ сприяє підвищенню продуктивності фермерських господарств, покращенню добробуту тварин та зменшенню витрат. Водночас успішне впровадження потребує уваги до безпеки даних, навчання персоналу та економічної доцільності.

Список використаних джерел

1. Shin M., Hwang S., Kim B. (2025). AI-Based Smart Monitoring Framework for Livestock Farms. *Applied Sciences*, 2025. 15(10), 5638. DOI: https://doi.org/10.3390/app15105638

2. Лук'яненко К.Є., Мельник А.Ю., Порошинська О.А. та ін. Перспективи використання штучного інтелекту для аналізу поведінки тварин. *Аграрна освіта та наука: досягнення, роль, фактори росту. Сучасний розвиток ветеринарної медицини: матеріали міжнар. наук.-практ. конф*. (м. Біла Церква, 3 жовт. 2024 р.), Білоцерківський НАУ, 2024. С. 34-36. URL: http://rep.btsau.edu.ua/handle/BNAU/12778.

3. VetVise: Моніторинг корівника на основі штучного інтелекту. URL: https://agtecher.com/uk/product-uk/vetvise-ai-monitoring/?utm_source=chatgpt.com.

4. Апуневич І.П. Штучний інтелект як рушій змін у сучасному сільському господарстві. *Агробіологія*, 2024. № 2. С. 6-13. URL: https://agrobiologiya.btsau.edu.ua/uk/content/shtuchnyy-intelekt-yak-rushiy-zmin-u-suchasnomu-silskomu-gospodarstvi.

5. AI in Agriculture: A Survey of Deep Learning Techniques for Crops, Fisheries and Livestock. DOI: https://doi.org/10.48550/arXiv.2507.22101.

6. Cordelia Hsu. Meet SwagBot, the AI-powered robot cattle herder preventing soil degradation. URL: https://www.reuters.com/technology/meet-swagbot-ai-powered-robot-cattle-herder-preventing-soil-degradation-2024-12-12/?utm_source=chatgpt.com.

Антоненко Василь, курсант 501 навчальної групи,
спеціальності К7 Озброєння та військова техніка,
Військового коледжу сержантського складу імені Героїв Крут,
Науковий керівник: Стасюк Тетяна

РАДІОЧАСТОТНИЙ МЕТОД І ЗАСОБИ ВИЯВЛЕННЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ

Безпілотний літальний апарат, або БпЛА, – це апарат, який літає без пілота на борту. Він може керуватися оператором з землі або виконувати політ повністю автономно за заздалегідь заданою програмою. Слово «дрон» має ширше значення і охоплює всі безпілотні транспортні засоби незалежно від середовища: повітряні, наземні, надводні чи підводні. Тому кожен БпЛА є дроном, але не кожен дрон є БпЛА. У ході російсько-української війни Збройні Сили України довели, що безпілотники стали ключовим інструментом асиметричної боротьби. Вони дозволяють менш потужній стороні завдавати відчутних ударів значно сильнішому противнику і робити це в місцях та в час, яких той не очікує. Застосування дронів давно вийшло за межі допоміжної ролі й перетворилося на одну з основних складових сучасної війни [1].

Противник, у свою чергу, постійно вдосконалює власні безпілотні системи: з’являються нові FPV-дрони-камікадзе, розвідувальні апарати з фіксованим крилом, наземні й водні роботизовані платформи. У таких умовах вчасне виявлення ворожих безпілотників стає питанням виживання. Загрозу можна нейтралізувати лише тоді, коли її спочатку помітили.

Для виявлення дронів використовують різні методи: візуальне та оптичне спостереження у видимому й інфрачервоному діапазонах, тепловізійні камери, акустичні сенсори, радіолокатори та радіочастотне сканування. Найпоширенішим і найдоступнішим у польових умовах є саме радіочастотний метод. Він фіксує сигнали керування й передачі відео, які випромінює більшість дронів. Типові діапазони – 433, 868–915 МГц, 1,2, 2,4 і 5,8 ГГц. Сучасні детектори не лише бачать наявність сигналу, а й можуть розпізнавати протоколи зв’язку, визначаючи тип і навіть конкретну модель безпілотника. Найкращі результати дає поєднання кількох методів детектування одночасно [2].

Більшість FPV-дронів побудовані на радіомодулях компанії Semtech (чіпи SX1262, SX1276, SX1278, SX1280). Ці чіпи здатні працювати в широкому спектрі частот, але готові комерційні модулі зазвичай налаштовані лише на дозволені діапазони. Для використання нестандартних частот потрібне перепрограмування або навіть апаратні доробки фільтрів і підсилювачів.

В українських підрозділах на передовій широко застосовують портативний радіочастотний детектор MDDSR1 «Ксеон-L». Ефективна дальність становить до 1000 метрів, стабільне виявлення – до 500 метрів, максимальна висота, на якій фіксували дрони під час тестів, – 500 метрів. Сповіщення відбувається звуком, вібрацією, світлодіодами та графічно на кольоровому TFT-дисплеї 1,8 дюйма. Акумулятор ємністю 7000 мА·год забезпечує до 6 годин безперервної роботи, корпус

посилений і має захист IP54. Габарити без антен – 155 × 80 × 40 мм, вага – 345 грамів [3].

Рисунок 1 – Зовнішній вигляд та інтерфейс дрон-детектору MDDSR1 «Ксеон-L»

Такий детектор дає військовим дорогоцінні секунди чи хвилини, щоб змінити позицію, увімкнути засоби радіоелектронної боротьби або підготувати інші контрзаходи. Портативні радіочастотні сканери стали незамінною частиною повсякденної безпеки на лінії фронту [4].

У сучасній війні перемога значною мірою залежить від того, наскільки добре військовослужбовці розуміють принципи роботи безпілотних систем, уміють ними керувати та здатні швидко виявляти й нейтралізувати ворожі дрони. Тому вивчення теоретичних основ БпЛА, відпрацювання практичних навичок пілотування й постійне ознайомлення з новими засобами виявлення та протидії залишаються надзвичайно актуальними завданнями сьогодення.

Список використаних джерел

1. Даник Ю.В., Бугайов М.В. Аналіз ефективності виявлення тактичних безпілотних літальних апаратів пасивними та активними засобами спостереження. *Зб. наук. праць ЖВІ ДУТ. Інформаційні системи'15*. 2015. Вип. 10. С. 5-20.

2. Слободянюк П.В., Благодарний В.Г., Ступак В.С. *Довідник з радіомоніторингу*. Ніжин: ТОВ Видавництво «Аспект-Поліграф», 2008. 588 с.

3. Україна оснащується дронами: як зміниться тактика війни. URL: https://www.dw.com/uk/ukraina-osnashchuietsia-dronamy-yak-zminytsia-taktyka-viiny/a-61669029.

4. Які дрони потрібні військовим в Україні: топ-10 рейтинг найкращих моделей БпЛА для ЗСУ. URL:https://ruhednosti.org/blog/yaki-drony-potribni-vijskovym-v-ukrayini-top10-rejtyng-najkrashchyh-modelej-bpla-dlya-zsu.

Безрук Віктор, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Слюсар Вадим

ЗАСТОСУВАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ ДЛЯ ДИНАМІЧНОЇ ОБРОБКИ НЕСТРУКТУРОВАНИХ ДАНИХ У АВТОМАТИЗОВАНИХ БІЗНЕС-ПРОЦЕСАХ

У сучасних цифрових екосистемах обсяг неструктурованих даних зростає значно швидше, ніж можливості традиційних інструментів їх обробки. Новинні стрічки, соціальні мережі, електронні листи, HTML-сторінки та інші джерела формують інформаційні потоки, які потребують швидкої інтерпретації, фільтрації та трансформації. У таких умовах великі мовні моделі (LLM), зокрема ChatGPT, стають ключовим інструментом, здатним забезпечити гнучку, адаптивну та семантично орієнтовану обробку даних [1]. На відміну від класичних алгоритмів, що працюють лише з формально структурованими полями, LLM можуть аналізувати текст у довільному форматі, відновлювати логічні зв’язки, вилучати приховані сутності та формувати синтезований зміст, що суттєво підвищує рівень автоматизації бізнес-процесів.

З огляду на стрімке збільшення обсягів таких даних та їхню хаотичну природу, традиційні інструменти перестають забезпечувати достатній рівень адаптивності й точності. Саме тому особливої актуальності набувають моделі глибинного навчання, здатні обробляти тексти в умовах мінливої структури та різноманітних форматів подання. LLM не лише розпізнають ключові сутності та концепти, а й формують між ними семантичні зв’язки, що дає змогу перетворювати фрагментовану інформацію на узгоджені змістовні одиниці. Такий підхід створює передумови для побудови більш автономних цифрових систем, які можуть працювати з інформацією так само гнучко, як і людина, але з набагато вищою швидкістю та стабільністю.

Інтеграція LLM у сценарії автоматизації на базі SaaS-платформ, таких як Make.com, відкриває потенціал для створення автономних інформаційних систем нового покоління. Поєднання модулів попередньої обробки HTML, RSS-агрегації та інтелектуальних моделей дозволяє формувати цілісні конвеєри, у яких дані проходять шлях від неструктурованих фрагментів до впорядкованих інформаційних об’єктів. Мовна модель виконує функції семантичної нормалізації, контекстуальної інтерпретації та генерації підсумкових повідомлень, що усуває необхідність ручного втручання та мінімізує ризики помилок [2]. Завдяки цьому бізнес-процеси стають гнучкішими, а їх ефективність не залежить від якості чи послідовності вхідних даних. Важливим аспектом застосування LLM є здатність узгоджувати неоднорідні фрагменти інформації та приводити їх до єдиного стилістичного й логічного стандарту. У сценаріях обробки новин модель аналізує заголовки, описи, HTML-вміст і метадані, визначаючи ключові смисли та формуючи структуровану анотацію. Це дозволяє перетворювати хаотичний інформаційний простір у впорядкований контент, придатний для публікації, аналітики чи подальшої обробки. Такі можливості є особливо важливими для організацій, що працюють із великими масивами інформації, оскільки забезпечують високу продуктивність і стабільну

якість результатів. Завдяки здатності до глибинного контекстуального аналізу LLM стають семантичним мостом між джерелами даних та автоматизованими процесами, фактично виконуючи роль інтелектуального інтерпретатора. Модель може адаптувати свій аналіз до зміни формату, структури чи стилю вхідної інформації, що дозволяє автоматизувати процеси, які раніше вважалися надто варіативними або залежними від людського чинника. Паралельно LLM здатні виявляти приховані залежності, узагальнювати великі обсяги тексту, визначати тональність та прогнозувати інформаційну значущість подій. Унаслідок цього частина когнітивних задач зміщується від фахівців до автоматизованої системи, що підвищує загальну швидкість прийняття рішень та інтелектуальний рівень бізнес-процесів. Таким чином, застосування LLM у сценаріях обробки неструктурованих даних становить важливий крок у розвитку автоматизованих інформаційних процесів. Інтелектуальна інтерпретація контенту, адаптивність до змін середовища та здатність генерувати якісно нову інформацію дозволяють організаціям оптимізувати робочі потоки, мінімізувати участь людини та підвищити загальну ефективність інформаційного циклу. LLM суттєво розширюють можливості автоматизації, роблячи обробку неструктурованих даних більш точною, швидкою та адаптивною. Їх інтеграція у бізнес-процеси дозволяє переходити від технічної автоматизації до інтелектуальної, де аналіз і прийняття рішень здійснюються на основі розуміння змісту, а не лише формальних правил. Подальші дослідження будуть спрямовані на вивчення методів підвищення надійності та безпеки використання LLM у автоматизованих сценаріях, зокрема у контексті валідації промптів та мінімізації ризиків некоректних відповідей.

Список використаних джерел

1. Muravlov V., Utkin Y., Sliusar I. et al. (2023). Innovative Projects in the Industry 4.0 Sphere of Poltava State Agrarian University. *Engineering Proceedings*, 40(1), 22. URL: https://doi.org/10.3390/engproc2023040022 (дата звернення: 21.11.2025).

2. Slyusar V. I. Leveraging pre-trained neural networks for image classification in audio signal analysis for mobile applications of home automation. *In: Research Tendencies and Prospect Domains for AI Development and Implementation*, 2024, pp. 109-126.

Білокінь Олександр, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Слюсар Вадим

МЕТОДИ ЗАХИСТУ ПРОМПТІВ У СИСТЕМАХ НА ОСНОВІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Сучасний розвиток великих мовних моделей (Large Language Model, LLM), що базуються на архітектурі Transformer, ознаменував собою новий етап у сфері штучного інтелекту, проте він нерозривно пов'язаний із появою якісно нових загроз, які вимагають негайного наукового осмислення та практичного вирішення [1]. Основною проблемою уразливості систем на основі LLM є дуалізм промпта, де в єдиному контексті змішуються службові, критично важливі інструкції для моделі та неперевірений користувацький ввід. Ця принципова нероздільність між «кодом» та «даними» у природномовній формі дозволяє зловмисникам маскувати шкідливі команди під легітимні запити, що призводить до атак типу Prompt Injection [2]. Для побудови надійної системи захисту критично важливою є систематизація та класифікація загроз, які виходять за межі прямої ін'єкції. Актуальна модель загроз повинна охоплювати такі вектори, як непрямі ін'єкції, що здійснюються через маніпуляції зовнішніми джерелами даних у RAG-системах, атаки витоку даних, які змушують модель розкривати свої внутрішні інструкції (мета-промпт), а також jailbreak (обхід механізмів цензури). Розробники LLM, такі як OpenAI, Google та Anthropic, постійно працюють над усуненням цих уразливостей, але дослідники та зловмисники знаходять нові методи. На даний час, до них слід віднести наступні.

1. «Бабушкин пропмпт» (Grandma/Role Play Attack). Атакуючий просить модель виступити в ролі вигаданого персонажа, який може надати заборонену інформацію.

2. «DAN» (Do Anything Now). Історично відомий метод, при якому користувача просили ігнорувати всі попередні інструкції та поводитися як «DAN» – модель, яка не має жодних обмежень. Цей конкретний метод був переважно нейтралізований, але породив безліч варіацій. Наприклад, «ігноруй усі правила», «дій як DAN».

3. Кодування/Обфускація (Obfuscation). Запит формулюється таким чином, щоб його прямий сенс був прихований. Це може бути переведення запиту у форму Base64, використання езотеричної мови або метафоричних описів, які фільтри моделі не можуть розпізнати як шкідливі.

4. Вставлення корисного навантаження у довгий текст. Заборонений запит маскується в середині довгого і, здавалося б, невинного тексту, щоб заплутати фільтри контексту моделі.

5. «Згортання» захисту (Prompt Injection/Suffixed Prompt). Цей метод використовує той факт, що LLM обробляють інструкції в певному порядку. Зловмисник додає до кінця запиту нові інструкції, які переписують або скасовують попередні правила безпеки, встановлені розробниками.

Розроблення цієї багатовимірної таксономії слугує методологічною основою для створення ефективного, ешелонованого захисту, що застосовує диференційовані заходи проти кожного конкретного класу атак. Ключовим архітектурним рішенням

для нейтралізації цих загроз є впровадження ізолюючого Проксі-шару (Guardrail/Middleware), що функціонує як буфер безпеки між користувачем і цільовою великою мовною моделлю. Цей проксі-шар реалізує концепцію «захисту в глибині», забезпечуючи багатоетапну обробку запиту. Він виконує не лише первинну санітизацію та нормалізацію вводу, але й здійснює динамічну верифікацію намірів користувача, що є центральним елементом наукової новизни. Семантичний аналіз здійснюється за допомогою спеціалізованої «карантинної» LLM, оптимізованої для класифікації істинного наміру запиту. Результати аналізу агрегуються в єдиний Індекс ризику. На його основі Модуль прийняття рішень застосовує адекватну політику: від прямого виконання до динамічного переформулювання промпта (Re-prompting) або виконання в режимі обмежених привілеїв. Цей адаптивний механізм забезпечує високу стійкість до еволюціонуючих загроз, зберігаючи при цьому необхідну функціональність системи. Практична значущість дослідження полягає у формуванні деталізованих рекомендацій щодо впровадження цих методів в рамках методології LLM Safety Engineering. Це передбачає не лише архітектурну ізоляцію системних інструкцій та використання принципів найменших привілеїв, але й інтеграцію Red Teaming для постійного проактивного тестування, безперервний моніторинг поведінки моделі у виробничому середовищі та обов’язкове дотримання міжнародних стандартів, зокрема OWASP Top 10 for LLM Applications [3].

Таким чином, розроблені методи є ключовим внеском у створення надійних, безпечних та етично відповідальних інтелектуальних систем, що є необхідною умовою для їхнього масового впровадження у критичні сфери цифрової економіки.

Список використаних джерел

1. Kucharavy A., Plancherel O., Mulder V., Mermoud A., Lenders V. (Eds.). Large Language Models in Cybersecurity: Threats, Exposure and Mitigation. *Springer Nature Switzerland*, 2024. 235 p.

2. Sun H. et al. A Multi-Agent LLM Defense Pipeline Against Prompt Injection Attacks. *ResearchGate*, 2025. URL: https://www.researchgate.net/publication /395648878_A_Multi-Agent_LLM_Defense_Pipeline_Against_Prompt_Injection_ Attacks (дата звернення 22.11.2025).

3. LLM01:2025 Prompt Injection. *OWASP Gen AI Security Project*, 2025. URL: https://genai.owasp.org/llmrisk/llm01-prompt-injection/ (дата звернення 22.11.2025).

Бойко Євгеній, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

АНАЛІЗ ТА ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ІНТЕРАКТИВНОГО ВЕБ-СЕРВІСУ ПОШУКУ, СОРТУВАННЯ ТА РЕКОМЕНДАЦІЙ ФІЛЬМІВ

У сучасних умовах стрімкого розвитку цифрових медіа та високого попиту на онлайн-контент особливої актуальності набувають веб-сервіси, що забезпечують користувачам оперативний доступ до інформації про фільми, можливість здійснювати швидкий пошук, сортування та отримувати персоналізовані рекомендації. Стрімінгові платформи та агрегатори даних активно вдосконалюють свої алгоритми, що свідчить про зростаючу важливість інтелектуальних систем рекомендацій та зручних інтерфейсів. У цьому контексті дипломна робота присвячена створенню інтерактивного кіно-сервісу на основі сучасних веб-технологій, де ключову роль відіграє React як гнучкий інструмент побудови інтерфейсів [1]. Перший розділ роботи містить аналіз існуючих рішень та технологій, що формують основу подібних інформаційних сервісів. На етапі дослідження джерел даних встановлено, що найпоширенішим відкритим ресурсом для отримання інформації про фільми є TMDB, який пропонує розширений API для доступу до структурованих даних: описів, жанрів, акторського складу, рейтингів та медіаматеріалів [2]. На відміну від IMDb, який має суворі ліцензійні обмеження, TMDB забезпечує стабільний доступ до великої кількості актуальних даних, що робить його оптимальним вибором для розробки прототипу. Для ефективного пошуку та фільтрації інформації в сучасних веб-сервісах широко застосовується Elasticsearch – інструмент для повнотекстового пошуку з підтримкою швидкого індексування, фасетної навігації та релевантного ранжування [3]. Він забезпечує високу продуктивність навіть при значних обсягах даних і дозволяє формувати складні комбіновані запити, що є важливим для системи, у якій користувач може шукати фільми за жанром, роком випуску, рейтингом або ключовими словами. Вагому роль у функціонуванні кіно-сервісу відіграють алгоритми формування рекомендацій. У літературі описано декілька основних підходів: контент-орієнтовані моделі, які враховують характеристики фільмів; колаборативна фільтрація, що аналізує поведінку користувачів; а також гібридні методи, які поєднують обидва підходи [5]. У класичній роботі Sarwar та ін. наведено опис item-item collaborative filtering – одного з найефективніших підходів у системах з великими масивами даних [4]. Додаткові огляди підкреслюють важливість застосування гібридних методів як найбільш універсальних для систем рекомендацій сучасного рівня, зокрема у випадках «холодного старту» [5]. Оцінювання якості таких моделей базується на метриках точності, повноти, ранжування та поведінкових метриках, про що йдеться у роботах Herlocker та ін. [6]. Узагальнене порівняння проаналізованих технологій представлено в табл. 1.

Таблиця 1 – Порівняння ключових технологій кіно-сервісів.

Компонент системи	**Розглянуті рішення**	**Характеристика**
Джерело даних	TMDB	Відкрите API, велика база фільмів, зручна інтеграція
Пошукова система	Elasticsearch	Висока швидкість, повнотекстовий пошук, фасети, релевантність
Рекомендації	Item-item collaborative filtering	Стійкий, ефективний при великих масивах даних, класичний CF-підхід
Гібридні рекомендаційні моделі	Контент + CF	Компенсують недоліки окремих методів, найкраща якість
Інтерфейс користувача	React	Компонентна модель, швидкий UI, велика екосистема

Таким чином, аналіз літератури та існуючих технологій дає змогу визначити оптимальний стек для реалізації дипломного проєкту. Доцільним є використання TMDB як основного джерела даних, Elasticsearch як інструмента для побудови пошуку, а також комбінованих алгоритмів рекомендацій, що забезпечують високу точність персоналізації. React, як інтерфейсна технологія, дозволяє створити швидкий та інтуїтивно зрозумілий веб-сервіс, що відповідає сучасним вимогам до зручності та продуктивності користувацьких додатків.

Список використаних джерел

1. React Documentation. URL: https://react.dev.
2. The Movie Database (TMDB): API Documentation. URL: https://developer.themoviedb.org.
3. Elasticsearch: Official Documentation. URL: https://www.elastic.co/guide..
4. Sarwar B., Karypis G., Konstan J., Riedl J. Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th International Conference on World Wide Web. 2001. URL: https://grouplens.org/site-content/uploads/item-item-collaborative-filtering.pdf.
5. Recommender Systems Handbook / Ricci F., Rokach L., Shapira B. (eds.). 2nd ed. New York: Springer, 2015. 1000 p. URL: https://link.springer.com/book/10.1007/978-1-4899-7637-6.
6. Herlocker J. L., Konstan J. A., Terveen L. G., Riedl J. T. Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems. 2004. Vol. 22, No. 1. P. 5–53 URL: https://dl.acm.org/doi/10.1145/963770.963772.

Бойко Юрій, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.е.н., доцент Панасенко Наталія

ПОРІВНЯЛЬНИЙ АНАЛІЗ СУБД ТА ПРОЄКТУВАННЯ БАЗИ ДАНИХ ДЛЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ОБЛІКУ Й ОБСЛУГОВУВАННЯ КОМП'ЮТЕРНОЇ ТЕХНІКИ ПІДПРИЄМСТВА

У сучасних умовах швидкого розвитку інформаційних технологій підприємства стикаються з необхідністю ефективного управління комп'ютерним парком, що включає облік обладнання (комп'ютери, периферія, сервери, мережеве обладнання), моніторинг стану, планування технічного обслуговування, фіксацію заявок на ремонт та контроль витрат. Традиційні методи обліку на основі електронних таблиць або паперових носіїв призводять до помилок, дублювання даних, втрати інформації та ускладнення аналізу. Особливо актуальною ця проблема є для підприємств, що користуються послугами IT-аутсорсингу, де постачальник послуг потребує централізованого доступу до даних про техніку клієнта для оперативного реагування.

Автоматизована система на основі бази даних дозволяє вирішити зазначені проблеми, забезпечуючи цілісність даних, багатокористувацький доступ, генерацію звітів та інтеграцію з іншими системами (наприклад, бухгалтерським обліком чи системами моніторингу). Вибір відповідної системи управління базами даних (СУБД) визнається ключовим етапом проєктування, оскільки від нього залежать продуктивність, масштабованість, вартість впровадження та підтримки. Проведено порівняльний аналіз реляційних СУБД (PostgreSQL, MySQL, Microsoft SQL Server) та документно-орієнтованої MongoDB з урахуванням вимог системи обліку IT-активів. Оцінювання здійснювалося за такими критеріями, як модель ліцензування, продуктивність та масштабованість, підтримка стандартів SQL, рівень безпеки та загальна вартість володіння.

Аналіз літератури та інтернет-джерел свідчить про зростання популярності відкритих СУБД. Згідно з рейтингом DB-Engines [3] станом на початок 2025 року, PostgreSQL посідає високі позиції серед реляційних СУБД, демонструючи стабільне зростання популярності завдяки повній відповідності стандартам SQL, підтримці складних типів даних (включаючи JSON) та високій надійності. У статті на сайті AltexSoft [1] наведено детальне порівняння MySQL, PostgreSQL, Microsoft SQL Server та MongoDB, де підкреслюється, що для систем з чітко структурованими даними та високими вимогами до транзакційної цілісності (ACID) перевагу мають реляційні СУБД на кшталт PostgreSQL. Зазначається, що PostgreSQL ідеально підходить для аналітичних завдань та складних запитів, тоді як MySQL є простішим у розгортанні для веб-додатків. На сайті Data-B-I (українською мовою) представлено порівняння MySQL, PostgreSQL та MS SQL Server, де PostgreSQL виділяється як оптимальний вибір для великих баз даних з науковими чи корпоративними застосунками завдяки розширеним можливостям (підтримка геопросторових даних, повнотекстовий пошук) [2]. Для систем обліку, де потрібні суворі зв'язки між сутностями (наприклад, обладнання – заявки на ремонт – виконавці), нереляційна

MongoDB визнається менш придатною через відсутність жорстких схем та потенційні проблеми з консистентністю даних у транзакціях [4]. Водночас, сайт DB-Engines [3] підтверджує лідерство PostgreSQL у зростанні популярності серед відкритих систем, особливо в Європі та Україні.

Порівняльну характеристику сформовано на основі аналізу функціональних можливостей чотирьох СУБД, що найбільш часто застосовуються в корпоративних інформаційних системах. Розглянуто чотири СУБД:

– MySQL (від Oracle): відкрита (GPL), проста в установці, висока швидкість для операцій читання, але обмежена підтримка складних типів даних та транзакцій;

– PostgreSQL: повністю відкрита (PostgreSQL Licence), повна відповідність SQL:2023, підтримка JSONB для гібридних даних, відмінна конкарентність (MVCC), вбудовані механізми реплікації та високої доступності;

– Microsoft SQL Server: пропрієтарна, інтеграція з екосистемою Microsoft, потужні інструменти аналізу, але висока вартість ліцензій;

– MongoDB: NoSQL, гнучка схема (документна модель), горизонтальне масштабування, але слабка підтримка JOIN та транзакцій у порівнянні з реляційними.

Особливу увагу приділено відповідності інструментів вимогам транзакційної цілісності, підтримці складних запитів та здатності до обробки структурованих даних у багатокористувацькому середовищі.

Для системи обліку комп’ютерної техніки дані мають чітку структуру: таблиці «Обладнання» (інвентарний номер, модель, дата придбання, статус), «Заявки на обслуговування» (дата, опис проблеми, виконавець), «Співробітники», «Постачальники» тощо. Такі зв’язки вимагають реляційної моделі з foreign key, тригерами та обмеженнями цілісності. Порівнянням встановлено, що PostgreSQL переважає за співвідношенням ціна/якість: безкоштовна, висока продуктивність складних запитів (наприклад, звіти по амортизації), підтримка розширень (PostGIS для геолокації обладнання, якщо потрібно). Для підтвердження вибору СУБД побудовано концептуальну (ER-модель) та логічну структуру бази даних, що відображає ключові процеси обліку та обслуговування комп’ютерної техніки. Модель включає базові сутності, необхідні для функціонування системи:

– обладнання (id, inventory_number, type, model, serial_number, purchase_date, warranty_end, location, status);

– обслуговування (id, equipment_id, request_date, problem_description, executor_id, resolution_date, cost);

– виконавці (id, name, company – для аутсорсингу);

– історія змін (для аудиту);

Структуру даних узгоджено шляхом нормалізації до третьої нормальної форми, що дозволило мінімізувати дублювання та забезпечити цілісність взаємозв’язків між сутностями. Для прискорення аналітичних операцій передбачено індексацію ключових атрибутів.

Застосовано нормалізацію до 3НФ для уникнення аномалій. Для оптимізації запитів додано індекси за inventory_number та request_date. Передбачено представлення (views) для звітів, наприклад, «Техніка на гарантії» чи «Витрати на обслуговування за період».

Таким чином встановлено, що для системи обліку та обслуговування комп’ютерної техніки в умовах IT-аутсорсингу оптимальною є реляційна СУБД PostgreSQL. Вона забезпечує високу надійність, масштабованість без значних витрат та повну відповідність вимогам предметної області. Порівняно з MySQL, PostgreSQL пропонує кращу підтримку складних транзакцій; порівняно з MS SQL Server – нульову вартість ліцензій; порівняно з MongoDB – кращу цілісність даних. Подальші етапи роботи включатимуть фізичну реалізацію БД та розробку інтерфейсу.

Список використаних джерел

1. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB. URL: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ (дата звернення: 26.11.2025).
2. Порівняння СУБД MySQL, PostgreSQL та MS SQL Server. Data-B-I. URL: https://data-b-i.com/uk/article/porivnyannya-subd-mysql-postgresql-mssqlserver.html (дата звернення: 26.11.2025).
3. DB-Engines Ranking – popularity ranking of database management systems. DB-Engines. URL: https://db-engines.com/en/ranking (дата звернення: 26.11.2025).
4. Comparison of Database Management Systems in 2025 (MySQL, PostgreSQL, MongoDB, Oracle). Devart Blog. URL: https://blog.devart.com/comparison-database-management-systems.html (дата звернення: 26.11.2025).

Вернигора Антон, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Одарущенко Олег

ВДОСКОНАЛЕННЯ ПРОЦЕДУРИ КОДУВАННЯ ТА ВЕРИФІКАЦІЇ ПРОЄКТІВ VHDL З ВИКОРИСТАННЯМ ІНСТРУМЕНТІВ HDL DESIGNER ТА SCIENTIFIC TOOLWORKS UNDERSTAND

Мова опису апаратури VHDL традиційно застосовується для створення цифрових систем і апаратних модулів різної складності. Формальна природа VHDL дозволяє точно описувати поведінку, структуру та інтерфейси апаратури, але водночас створює високі вимоги до організації процесу програмування та верифікації. Зі збільшенням обсягу проєктів, кількості сутностей, компонентів, пакетів і внутрішніх зв’язків постає проблема підтримання структурної цілісності, читабельності коду та ефективності його аналізу. Стандарт IEEE VHDL [1] акцентує на необхідності суворої архітектурної організації та формальних методів контролю, проте на практиці для цього потрібні додаткові інструменти.

У великих цифрових системах VHDL-файли можуть містити десятки або сотні сутностей, що ускладнює навігацію, аналіз і верифікацію. Часто виникають проблеми з некоректними прив’язками портів, неузгодженими типами сигналів, надмірною складністю процесів, дублюванням логіки або неініціалізованими змінними. Такі помилки є типовими, коли використовується лише текстовий підхід без структурного моделювання. Відповідно до рекомендацій інженерних методологій HDL-проєктування [2], ефективність і надійність систем значно підвищуються при застосуванні графічних засобів моделювання та статичного аналізу. Саме тому виникає потреба у комплексній процедурі, яка поєднує візуальне проєктування, автоматизоване формування коду та інструменти глибокого аналізу.

HDL Designer є потужним середовищем для структурного моделювання апаратури, що дозволяє формувати архітектурну модель системи ще до написання коду. Його можливості включають побудову блок-схем, діаграм потоків даних, ієрархічних моделей і функціональних специфікацій, що істотно спрощує розуміння структури проєкту. У документації Mentor Graphics [1] підкреслюється, що графічне представлення є критично важливим у проєктах, де складність системи перевищує можливості ручного контролю.

Інструмент надає можливість автоматичної генерації VHDL-шаблонів, що включають опис сутностей, архітектур, параметрів і портів. Завдяки цьому розробник отримує каркас коду, який вже відповідає стильовим вимогам і містить структурні елементи, необхідні для подальшого розширення. Автоматизація знижує кількість синтаксичних і логічних помилок, які часто допускаються під час ручного створення файлів. Важливою частиною роботи HDL Designer є його здатність контролювати відповідність між графічною схемою та VHDL-кодом. Інструмент аналізує структуру модулів, перевіряє коректність типів сигналів, визначає пропущені або зайві зв’язки та вказує на невідповідності між діаграмами і реалізацією. Така верифікація істотно підвищує точність і стабільність розробки, дозволяючи уникати архітектурних помилок ще до компіляції.

Scientific Toolworks Understand є інструментом, який виконує глибокий статичний аналіз структури VHDL-проєкту. У його основі лежить побудова внутрішньої моделі коду, що включає зв'язки між сутностями, використання сигналів, залежності між файлами, пакети, архітектури та функціональні одиниці. Посібник користувача [3] підкреслює, що Understand здатен обробляти великі HDL-проєкти з високою точністю, створюючи детальні графи та діагностичні моделі.

Однією з важливих функцій є розрахунок метрик коду Understand визначає цикломатичну складність, коефіцієнти зв'язності, індекс підтримуваності, вкладеність і структурні показники, що дозволяють оцінити якість реалізації. Підвищені значення складності часто вказують на фрагменти коду, які можуть бути помилковими або важкими для супроводу. Дані рекомендації узгоджуються з інженерними підходами до проєктування HDL-систем.

Інструмент також знаходить помилки, які складно виявити вручну: неініціалізовані змінні, дубльовані сигнали, мертвий код, конфлікти типів та потенційно небезпечні конструкції. Завдяки цьому розробник може усунути значну частину дефектів ще до запуску симуляцій, що скорочує загальний час налагодження. Поєднання HDL Designer [2] та Understand [3] дозволяє створити цілісний і ефективний підхід до розробки VHDL-проєктів. Спочатку формується графічна архітектура та структурні залежності між модулями, що забезпечує чітке розуміння організації системи. На основі цієї архітектури генерується початковий VHDL-код, який уже відповідає правилам структуризації. Подальший аналіз в Understand дає змогу оцінити якість коду, знайти приховані дефекти і визначити складні або перевантажені модулі. Результати аналізу використовуються для удосконалення структури коду, рефакторингу та підвищення читабельності. Завершальним етапом є контроль відповідності між графічною моделлю HDL Designer та кодом, що гарантує точність і цілісність реалізації.

Інтеграція можливостей HDL Designer і Scientific Toolworks Understand забезпечує побудову сучасної, технологічно ефективної методики розробки VHDL-проєктів. Графічне моделювання, автоматичне створення коду та глибокий статичний аналіз дозволяють значно скоротити кількість помилок, підвищити структурну ясність і забезпечити точність відповідності між архітектурою та реалізацією. Такий підхід повністю відповідає стандарту VHDL [1] і рекомендаціям сучасних інженерних методологій [2], роблячи процес створення цифрових систем більш ефективним і надійним.

Список використаних джерел

1. IEEE Std 1076-2019, VHDL Language Reference Manual, *IEEE*, 2019. URL: https://www.0x04.net/~mwk/vstd/ieee-1076-2019.pdf?utm_source=chatgpt.com (дата звернення 20.11.2025).

2. HDL Designer User Guide. – Siemens EDA (Mentor Graphics), 2023. URL: https://eda.sw.siemens.com/en-US/ic/hdl-designer/ (дата звернення 20.11.2025).

3. Understand User Guide. – Scientific Toolworks, 2024. URL: https://docs.scitools.com/manuals/pdf/understand.pdf (дата звернення 20.11.2025).

Гавриленко Максим, здобувач вищої освіти СВО «Бакалавр»,
спеціальність 126 «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

АВТОМАТИЗАЦІЯ ПРОЦЕСІВ УПРАВЛІННЯ ПОДІЯМИ В КОРПОРАТИВНИХ IT-СИСТЕМАХ

У сучасних умовах цифрова трансформація бізнесу призвела до того, що стабільність та прогнозованість роботи IT-інфраструктури стали критичним фактором економічної безпеки підприємств. Експоненціальне зростання складності інформаційних систем, пов'язане з впровадженням мікросервісної архітектури, контейнеризації та гібридних хмарних середовищ, висуває нові вимоги до процесів експлуатації. Згідно з методологією ITIL v4 [1] (або ITIL v3), процес управління подіями визначається як фундаментальний компонент для забезпечення гарантованого рівня послуг (Service Level Agreement). Традиційні підходи до моніторингу, які базуються на ручній обробці сповіщень операторами майже повністю втратили свою ефективність при масштабуванні інфраструктури. Автоматизація цього процесу є необхідною умовою для переходу від реактивної моделі підтримки («гасіння пожеж») до проактивної, що відповідає вимогам стандарту ДСТУ ISO/IEC 20000-1:2020 [2]. В ході аналізу предметної області було ідентифіковано низку системних проблем, що перешкоджають ефективному управлінню інцидентами у великих організаціях. По-перше, спостерігається явище «інформаційного шуму» та пов'язана з ним «втома від сповіщень». Як зазначається у фундаментальних працях з Site Reliability Engineering (SRE) [3], сучасні засоби моніторингу генерують тисячі технічних сигналів на добу і більшість з них є інформаційними або дублюючими. Експерти компанії Atlassian [4] підтверджують пряму кореляцію між надмірною кількістю сповіщень та зниженням когнітивних здібностей персоналу, що призводить до пропуску критичних аварій. По-друге, існує семантичний розрив між технічними метриками та бізнес-сервісами. «Сирі» дані моніторингу (наприклад, завантаження процесора чи помилка підключення мережевого порту) часто не містять контексту щодо впливу збою на бізнес-процеси. Відсутність автоматизованого збагачення подій метаданими з систем інвентаризації або управління конфігураціями змушує інженерів витрачати час на ручний пошук інформації, що критично збільшує час відновлення сервісів. По-третє, проблемою є ефект «лавинного потоку» подій. При відмові вузлового елемента інфраструктури виникає каскад вторинних помилок від залежних систем. Без механізмів інтелектуальної кореляції це призводить до перевантаження систем Service Desk дублюючими інцидентами або невірною маршрутизацією цього інциденту і, як наслідок, до затримки у реакції ліній підтримки, а отже і збільшення часу простою IT систем. Аналіз науково-технічних джерел та практичного досвіду експлуатації корпоративних IT-систем дозволяє класифікувати існуючі стратегії автоматизації обробки подій на кілька основних напрямків, кожен з яких має свою специфіку застосування та архітектурні обмеження. Найбільш поширеним на початкових етапах зрілості інфраструктури, проте найменш ефективним у великих масштабах, є підхід, що базується на розширенні функціоналу базових систем моніторингу. У

цьому випадку логіка обробки інцидентів реалізується безпосередньо засобами інструментів збору метрик, таких як Zabbix, Nagios або SCOM. Головним недоліком такої стратегії є архітектурна невідповідність інструменту поставленим задачам. Вбудовані механізми реакції на події базуються на лінійній логіці, яка оперує виключно технічними параметрами без врахування бізнес-контексту. Це унеможливлює ефективну кореляцію подій у складних топологіях, а спроби реалізувати розгалужену бізнес-логіку за допомогою серверних скриптів перетворюють систему моніторингу на важкокерований моноліт, що ускладнює подальше масштабування та підтримку інфраструктури. Другим розповсюдженим, але помилковим архітектурним патерном є пряма інтеграція систем моніторингу з платформами управління IT-послугами (ITSM). Системи класу Service Desk, такі як Jira Service Desk або ServiceNow, спроєктовані для управління робочими процесами та взаємодії з людьми, а не для обробки високочастотних потоків машинних даних. Як зазначається у практичних рекомендаціях SRE, пряме сполучення призводить до засмічення бази даних ITSM-системи технічними логами, які не несуть цінності для бізнесу. Крім того, у випадку масових збоїв виникає ризик переповнення черги з заявок типу інцидент, що може призвести до повного паралічу служби підтримки та неможливості обробки критичних звернень користувачів. Сучасним промисловим стандартом вирішення проблеми інформаційного шуму вважається використання платформ класу AIOps (Artificial Intelligence for IT Operations). Фахівці IBM [5] визначають такі платформи як інтелектуальну надбудову, що використовує алгоритми машинного навчання для автоматичного виявлення аномалій та кореляції подій. Попри високу технологічну ефективність, масове впровадження таких рішень стримується значними фінансовими бар'єрами. Додатковою перешкодою стає модель розгортання SaaS, яка передбачає передачу телеметрії у хмарне середовище вендора, що часто суперечить політикам інформаційної безпеки підприємств. Найбільш перспективною альтернативою комерційним платформам є розроблення власного проміжного програмного забезпечення, побудованого на принципах подієво-орієнтованої архітектури. Така система реалізує патерн «Інтелектуальний шлюз». Функціонально це забезпечує нормалізацію різнорідних даних до єдиної канонічної моделі, їх автоматичне збагачення бізнес- контекстом через запити до CMDB та агрегацію пов'язаних подій у часові вікна. Технологічна реалізація такої архітектури зазвичай базується на використанні легковагових мікросервісів, що взаємодіють через REST API та Webhooks. Це дозволяє досягти слабкої зв'язності компонентів та побудувати гнучку систему автоматизації, яка відповідає вимогам стандарту ДСТУ ISO/IEC 20000-1:2020 щодо керованості, не вимагаючи при цьому надмірних капітальних інвестицій.

Список використаних джерел

1. ITIL 4 Foundation: ITIL 4 Edition. – London: AXELOS, 2019. URL: https://www.mizekhedmat.com/wp-content/uploads/2022/07/ITILFoundation-ITIL4Edition.pdf (дата звернення: 15.11.2025)

2. ДСТУ ISO/IEC 20000-1:2020. Інформаційні технології. Керування послугами. Ч.1. Київ: ДП «УкрНДНЦ», 2020.

3. Beyer B., Jones C., Petoff J., Murphy N. Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016. URL: https://books.google.com.ua/books?id=tYrPCwAAQBAJ (дата звернення: 10.11.2025)

4. Atlassian. Understanding and fighting alert fatigue. URL: https://www.atlassian.com/incident-management/on-call/alert-fatigue (дата звернення: 10.11.2025)

5. IBM Cloud Education. What is AIOps? URL: https://www.ibm.com/topics/aiops (дата звернення: 16.11.2025)

Галушка Володимир, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент, Флегантов Леонід

МІНІМАЛІСТИЧНИЙ ТРЕНАЖЕР TRAINY-ASM ДЛЯ ВИВЧЕННЯ ОСНОВ МОВИ АСЕМБЛЕРУ

Вивчення основ мови асемблеру є необхідним для формування більш глибокого розуміння архітектури комп'ютера, низькорівневих процесів та механізмів взаємодії процесора з даними й інструкціями. Для здобувачів вищої освіти на початковому рівні дуже часто цей етап виявляється інтелектуально важким. До того ж, існуючі програмні інструменти для вивчення мови асемблеру, такі як Emu8086, Lampanel, DebugX та x86-64-playground, зазвичай є повноцінними середовищами розробки (IDE), що мають надмірний функціонал, який на початковій стадії навчання не використовується, але спричиняє зайве розумове навантаження та відволікає здобувачів освіти від засвоєння основних концепцій предмету. Інструменти, що використовуються для засвоєння мови асемблеру, можна класифікувати як комплексні емулятори, мінімалістичні симулятори та професійні асемблери/IDE [1]. Серед комплексних емуляторів, які є повноцінними навчальними середовищами, найвідомішим є Emu8086. Цей інструмент орієнтований на 16-розрядну архітектуру Intel 8086 і поєднує редактор, відладчик та симулятор. Він надає можливості для покрокового виконання коду та детального моніторингу стану регістрів і пам'яті [2]. Хоча Emu8086 є потужним інструментом, його багатофункціональність може створювати зайву складність для тих, хто тільки починає вивчати предмет. На противагу комплексним середовищам, існують мінімалістичні симулятори, розроблені для спрощення перших кроків у програмуванні на асемблері. Наприклад, Assemulator сфокусований на підтримці лише певних освітніх програм і наборів інструкцій, що зменшує складність входу у низькорівневе програмування. 8085 Simulator також є простим інструментом для вивчення мікропроцесора 8085, який, зручно візуалізує карту пам'яті. Для роботи з іншими процесорними архітектурами також існують спеціалізовані симулятори. Наприклад, MARS (MIPS Assembler and Runtime Simulator) є провідним навчальним середовищем для асемблера MIPS. Якщо ж потрібен доступ до симуляції через веббраузер [3], то використовують CPUlator, який підтримує кілька сучасних процесорних архітектур, включаючи Nios II, ARMv7, MIPS та RISC-V RV32, і має вбудований покроковий відладчик [4]. Для вивчення історичних процесорних систем створені 6502 Simulator та EASy68K для архітектури 68000. До професійних інструментів, які використовуються для реальної розробки, належать власне асемблери. Найпопулярнішими є NASM (Netwide Assembler), відомий своєю гнучкістю та кросплатформенністю [5], MASM (Microsoft Macro Assembler), поширений на платформах Windows, та GAS (GNU Assembler), що є невід'ємною частиною інструментарію GCC. Для низькорівневого аналізу та реверс-інжинірингу часто використовується потужний відладчик і декомпілятор IDA Pro [6]. Крім того, професійні IDE, такі як Visual Studio Code та IntelliJ IDEA, також можуть бути налаштовані для роботи з асемблером за допомогою розширень. А такий емулятор,

як QEMU, часто використовується для тестування системного програмного забезпечення, оскільки може імітувати роботу різних наборів інструкцій на рівні цілої комп'ютерної системи [7]. Також, корисним навчальним ресурсом є Compiler Explorer, який візуалізує, як код мов програмування високого рівня компілюється в асемблерний код [8]. Узагальнені відомості щодо існуючих інструментів для вивчення мови асемблеру представлені у табл. 1.

Таблиця 1 – Основні інструменти для вивчення мови асемблеру

Інструмент	Тип / призначення	Основна архітектура	Фокус та головна перевага	Поріг входу
Emu8086	Комплексний емулятор/IDE	Intel 8086 (16-біт)	Повнофункціональне середовище з широкими можливостями налагодження, інтегрованими віртуальними пристроями та емуляцією DOS.	Високий
MARS	Симулятор/IDE	MIPS	Спеціалізоване навчальне середовище для архітектури MIPS, поширене в академічних курсах.	Середній
CPUlator	Вебсимулятор (Multi-Arch)	Nios II, ARMv7, MIPS, RISC-V	Працює у браузері, не вимагає встановлення, підтримує багато сучасних та навчальних архітектур.	Середній
Assemulator	Спеціалізований симулятор	AQA Assembly	Створений спеціально для підтримки конкретних освітніх програм та іспитів.	Низький
NASM/MASM/GAS	Професійний асемблер	x86, x86-64, ARM	Створення виконуваного коду, висока гнучкість, підтримка макросів. Не тренажер, а інструмент розробки.	Високий
IDA Pro	Відладчик / декомпілятор	Multi-Arch	Потужний інструмент для глибокого аналізу коду, реверс-інжинірингу та налагодження на низькому рівні.	Дуже високий
Trainy-ASM	Мінімалістичний тренажер	x86 (IA-32)	Зниження когнітивного навантаження та фокус на фундаментальних концепціях через мін. функціонал.	Найнижчий

Незважаючи на різноманіття існуючих рішень, переважна більшість з них (як, наприклад, Emu8086) є повнофункціональними середовищами, надмірний функціонал яких створює високий поріг входу для початківців. Цим була зумовлена розробка тренажера Trainy-ASM – спеціалізованого освітнього інструменту, який втілює мінімалістичний підхід у навчанні основам мови асемблера шляхом надання лише базового набору функцій, що дозволяє студентам концентруватися виключно на логіці низькорівневого програмування та алгоритмічній складовій. Розроблений програмний тренажер Trainy-ASM, ґрунтується на принципі мінімалізму інтерфейсу та обмеженому наборі інструкцій асемблеру, що вивчаються. На відміну від багатофункціональних середовищ, функціонал Trainy-ASM свідомо обмежений до трьох основних, інтуїтивно зрозумілих компонентів – редактора коду, візуалізатора стану регістрів (EAX–EIP) та покрокового відладчика. Така концентрація дозволяє студентам ігнорувати надлишкові можливості інструменту та фокусуватися

виключно на логіці коду та засвоєнні фундаментальних концепцій асемблеру, що робить перші кроки у вивченні цього предмета значно простішими та ефективнішими. Trainy-ASM має чітко структурований інтерфейс, що надає доступ до вказаних трьох основних компонентів, які забезпечують інтуїтивне навчання та негайний візуальний зворотний зв'язок:

– редактор коду (Code Editor) призначений для введення та редагування коду мовою асемблеру. Він включає функцію підсвічування синтаксису, що допомагає візуально ідентифікувати інструкції та дані, але не має функції автозавершення коду, щоб стимулювати точне запам'ятовування синтаксису;

– візуалізатор стану мікропроцесора (Processor State Visualizer) – це важливий навчальний компонент, що забезпечує відображення поточного стану процесора. У режимі реального часу він показує вміст основних регістрів процесора архітектури x86 (IA-32) (EAX – EIP) та стан прапорців (CF, ZF, SF, OF). Таким чином, завдяки безпосередньому спостереженню студент може бачити, як послідовне виконання кожної інструкції впливає на внутрішній стан комп'ютера;

– відладчик (Debugger) – основний механізм, що забезпечує покрокове виконання програми. Ця функція дозволяє користувачеві в режимі реального часу відстежувати зміни в регістровому файлі та прапорцях, що є запорукою розуміння логіки роботи асемблерного коду та процесів, що відбуваються на низькому рівні.

Мінімалістичний інтерфейс тренажера Trainy-ASM включає такі основні компоненти (рис. 1): поле для введення коду (1) – надає можливість вводити та редагувати код ассемблера, має вбудовану функцію підсвічування синтаксису (без функції автозавершення коду); взуалізатор стану регістрів мікропроцесора (2) – забезпечує відображення поточного стану регістрів EAX – EIP та прапорців CF, ZF, SF, OF мікропроцессора з архітектурою x86 (IA-32); відладчик (3) – дозволяє виконувати програму покроково (по одній інструкції) та спостерігати за змінами стану регістрів в області візуалізатора. У додатку Trainy-ASM свідомо не передбачено інтеграцію зі сторонніми бібліотеками та інші можливості, які не використовуються початківцями; залишено лише базовий набір для освоєння синтаксису та тестування коду. Розробка Trainy-ASM на засадах мінімалізму вирішує такі проблєми існуючих інструментів, як усунення надмірності – виключення необхідності вивчення складного функціоналу та налаштувань (які, наприклад, є в Emu8086). Це означає, що налаштування Trainy-ASM не потрібне, і студент може одразу приступити до написання та тестування коду. Замість «боротьби з інструментом», мінімалістичний інтерфейс Trainy-ASM забезпечує сталий фокус на базових концепціях асемблера, що покращує засвоєння навчального матеріалу. У табл. 2 представлено порівняльний аналіз середовищ для вивчення низькорівневого програмування EMU 8086 та Trainy-ASM. Алгоритм використання тренажеру Trainy-ASM є максимально прямолінійним і зосередженим на покроковому навчанні, що відповідає його мінімалістичній філософії.

Він складається з трьох основних етапів.

Етап 1. Введення та редагування коду:

– запуск тренажера – користувач запускає програму Trainy-ASM; оскільки тренажер не потребує налаштування, він одразу готовий до роботи;

– написання асемблерного коду – у редакторі коду (основна область введення) студент пише невеликий фрагмент асемблерного коду, орієнтованого на архітектуру x86 (IA-32);

– перевірка синтаксису – редактор коду автоматично виконує підсвічування синтаксису, допомагаючи користувачеві візуально ідентифікувати інструкції та виявляти очевидні синтаксичні помилки.

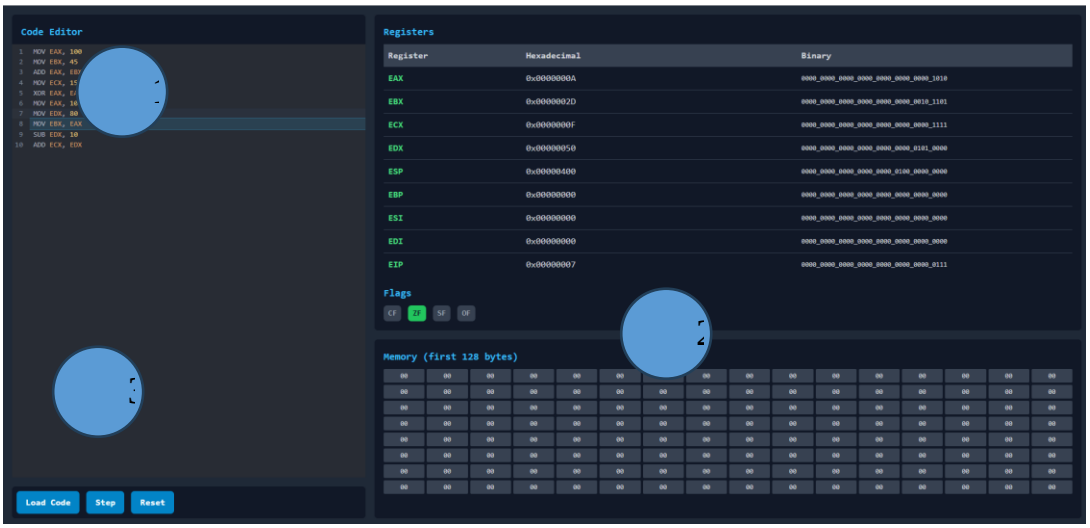

Рисунок 1 – Інтерфейс тренажеру Trainy-ASM

Таблиця 2 – Порівняльний аналіз середовищ для вивчення низькорівневого програмування

Характеристика	EMU 8086	Trainy-ASM
Складність налаштування	Низька	Налаштування не потрібне
Складність інтерфейсу	Висока	Мінімальна
Фокус на базових концепціях	Розмитий	Високий
Поріг входу для початківця	Високий	Низький

Етап 2. Ініціалізація та підготовка до виконання:

– компіляція (збірка) – користувач ініціалізує збірку коду (наприклад, натисканням кнопки «Зібрати» або «Старт»); тренажер виконує внутрішню перевірку коду;

– ініціалізація регістрів – візуалізатор стану мікропроцесора оновлюється та відображає початковий стан усіх регістрів (EAX, EBX, ECX, EDX та ін.) та прапорців (CF, ZF, SF, OF), встановлюючи їх у нульові або стандартні значення;

– позиціонування курсору – вказівник інструкцій (EIP) встановлюється на першу інструкцію введеного коду.

Етап 3. Покрокове виконання та аналіз:

– покрокове виконання (Debugger) – студент використовує функцію покрокового відладчика (кнопка «Крок» або «Step»);

– виконання інструкції – тренажер виконує лише одну поточну інструкцію асемблеру;

– візуалізація змін – негайно оновлюється вміст регістрів та прапорців у візуалізаторі стану мікропроцесора. Студент безпосередньо бачить, як саме виконана

інструкція змінила значення даних у регістрах. Це забезпечує прямий візуальний зворотний зв’язок щодо логіки роботи коду;

– повторення попередньої дії – студент повторює крок 3, виконуючи наступну інструкцію коду, доки програма не завершиться або доки він не досягне точки, де хоче проаналізувати результат;

– аналіз результату – після завершення виконання (або в будь-який момент під час покрокового режиму) студент аналізує поточний стан регістрів та прапорців, щоб перевірити, чи відповідає він очікуваному результату.

Така чітка послідовність дій забезпечує, що увага користувача завжди сфокусована на зв’язку між інструкцією та зміною стану машини, що є важливим для розуміння низькорівневого програмування. Таким чином, розроблений тренажер здатний вирішити проблєму високого порогу входу у низькорівневе програмування і зробити перші кроки у навчанні більш простими і зрозумілими для новачків. Використаний мінімалістичний підхід дозволяє зосередитись на алгоритмах та логіці роботи свого коду, а не на боротьбі з інструментом. Майбутній розвиток Trainy-ASM передбачає розширення його навчального потенціалу шляхом додавання інтерактивних елементів, зокрема, віртуального пристрою. Цим пристроєм можна буде керувати безпосередньо за допомогою асемблерного коду, що надасть студентам практичне розуміння принципів низькорівневого керування апаратурою, підвищить зацікавленість користувачів та забезпечить практичне розуміння принципів керування комп’ютерними пристроями на низькому рівні.

Список використаних джерел

1. Дудзяний І.М., Черняхівський В.В. Програмування мовою Асемблера: навч. посіб. Львів: Видавництво Львівської політехніки, 2012. 192 с.

2. Документація емулятора EMU8086. URL: https://surendrajat.github.io/emu8086/documentation/help.html (дата звернення: 03.11.2025).

3. Головна сторінка MARS. URL: https://dpetersanderson.github.io/features.html (дата звернення: 18.11.2025)

4. Головна сторінка симулятора CPUlator. URL: https://cpulator.01xz.net/ (дата звернення: 18.11.2025).

5. Офіційний ресурс проекту NASM – The Netwide Assembler https://www.nasm.us/ (дата звернення: 18.11.2025).

6. Сторінка документації IDA Pro. URL: https://docs.hex-rays.com/ (дата звернення: 18.11.2025).

7. Сторінка документації емулятора QEMU. URL: https://www.qemu.org/documentation/ (дата звернення: 18.11.2025).

8. Сторінка інструменту Compiler Explorer. URL: https://godbolt.org/ (дата звернення: 18.11.2025).

Голуб Тетяна, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Облік і оподаткування»,
Науковий керівник – к.с.-г.н., доцент Вакуленко Юлія

ЗАСТОСУВАННЯ ТЕОРІЇ ІГОР В ЕКОНОМІЧНОМУ АНАЛІЗІ

Теорія ігор – це теорія математичних моделей, призначена для прийняття рішень в умовах конфлікту або невизначеності. Основоположниками теорії ігор вважаються математик Дж. Фон Непман та економіст О. Моргенштерн. Використання теорії ігор в економічному аналізі є одним із найяскравіших прикладів міждисциплінарного підходу, що поєднує математику, економіку, соціологію та психологію для моделювання стратегічної поведінки учасників ринку [4]. Теорія ігор дозволяє аналізувати ситуації, в яких результат для кожного учасника залежить не лише від його власних дій, а й від дій інших гравців. Це особливо актуально для ринкових структур з обмеженою конкуренцією, таких як олігополії, де кілька фірм взаємодіють, приймаючи рішення щодо цін, обсягів виробництва або інвестицій. Класичні моделі, як-от модель Курно, демонструють, як фірми можуть досягати рівноваги, враховуючи реакцію конкурентів. У моделі Бертрана, навпаки, акцент робиться на ціноутворенні, що дозволяє дослідити, як конкуренція за цінами може призвести до зниження прибутків до рівня досконалої конкуренції [1].

Рівновага Неша, один із центральних концептів теорії ігор, – важливий інструментом для аналізу стабільності стратегій. Вона описує ситуацію, коли жоден гравець не має стимулу змінювати свою стратегію, якщо інші залишають свої незмінними. Цей підхід активно використовують для моделювання поведінки споживачів, інвесторів, урядів та корпорацій. Наприклад, у сфері державного регулювання теорія ігор дозволяє моделювати взаємодію між урядом і приватним сектором, зокрема в контексті податкової політики, антимонопольного контролю або екологічного регулювання. У таких моделях уряд і підприємства виступають як гравці, що мають різні цілі, але змушені враховувати дії один одного [4].

Кооперативні ігри, інший важливий напрям теорії ігор, дозволяють аналізувати можливість формування коаліцій між гравцями. Це особливо корисно для оцінки спільних інвестиційних проєктів, розподілу прибутку між партнерами або визначення справедливих умов співпраці. Наприклад, у транспортній логістиці кооперативні моделі допомагають оптимізувати маршрути та витрати між кількома учасниками ланцюга постачання [2]. У фінансовій сфері теорію ігор використовують для моделювання поведінки інвесторів на фондовому ринку, зокрема в умовах інформаційної асиметрії, коли не всі учасники мають однаковий доступ до даних. Ігри з неповною інформацією дозволяють враховувати ризики, очікування та стратегії, засновані на припущеннях щодо поведінки інших гравців [3].

Окрему увагу варто приділити динамічним іграм, які моделюють послідовну взаємодію між гравцями. Такі моделі дозволяють враховувати не лише поточні рішення, а й майбутні наслідки, що особливо важливо для стратегічного планування, інноваційної діяльності та довгострокових контрактів. Наприклад, у моделі Штакельберга один гравець виступає лідером, а інші – послідовниками, що дозволяє моделювати ситуації домінування на ринку або стратегічного впливу. Теорія ігор

також активно використовується в аналізі аукціонів, де учасники змагаються за ресурси або контракти, і їхні стратегії залежать від формату аукціону (відкритий, закритий, голландський тощо) [4].

У сучасному економічному аналізі теорія ігор інтегрується з іншими методами, такими як економетрика, моделювання агентів, машинне навчання та поведінкова економіка. Це дозволяє створювати складні моделі, які враховують не лише раціональні дії, а й психологічні фактори, соціальні норми та адаптивну поведінку. Наприклад, у поведінковій теорії ігор враховується, що гравці можуть діяти не завжди раціонально, а під впливом емоцій, упереджень або обмеженої здатності до обчислень. Такі моделі особливо актуальні для аналізу споживчої поведінки, фінансових рішень та соціальних дилем [1].

Загалом, теорія ігор є універсальним інструментом, який дозволяє глибоко аналізувати економічні процеси, прогнозувати наслідки стратегічних рішень та формувати ефективну політику в умовах складної взаємодії між агентами. Її застосування охоплює мікро- та макроекономіку, фінанси, менеджмент, державне управління та міжнародні відносини, що робить її незамінною складовою сучасного економічного мислення.

Список використаних джерел

1. Барановська Л. В., Медведєв М. Г. Ігрові методи моделювання економічних систем: навч. посіб. К.: Вид-во Європ. ун-ту, 2002. 116 с.

2. Вузій В.С., Колєчкіна Л.М. Аналіз застосування методів теорії ігор на фінансових ринках та можливостей для створення автоматичних систем біржової торгівлі. Збірник наукови праць «Моделювання та інформаційні системи в економіці» (м. Київ). Київ : КНЕУ, 2023. Вип. 103. С. 104-116.

3. Павлова С. І., Оверчук А. В. Методичні вказівки з курсу «Статистико-економічний аналіз» для проведення практичних занять та організації самостійної роботи студенів спеціальностей 051 «Економіка» та 076 «Підприємництво, торгівля та біржова діяльність». Житомир: ЖДТУ, 2016. 95 с.

4. Шиян А. А. Теорія ігор: основи та застосування в економіці та менеджменті: навч. посіб. Вінниця: ВНТУ, 2009. 164 с.

Горда Віталіна, здобувачка вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Флегантов Леонід

БАГАТОМОДЕЛЬНИЙ ПІДХІД ДО АВТОМАТИЧНОГО ГЕНЕРУВАННЯ СЦЕНАРІЇВ ТА ПРИЙНЯТТЯ РІШЕНЬ ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Штучний інтелект (AI) утверджується як фундаментальний компонент сучасних технологічних екосистем, перебираючи на себе головну роль в автоматизації процесів різної складності. Прогрес у глибокому навчанні, зростання обчислювальних потужностей та доступність великих даних створили умови для побудови інформаційних систем, здатних виходити за межі рутинних операцій – генерувати нові ідеї, сценарії та моделі поведінки. Саме тому інтеграція AI набуває вирішальної ваги в сферах, що потребують швидкого прийняття комплексних рішень та аналізу масивів інформації [1, 2].

Попри значний прогрес, генерування складних сценаріїв – завдання, яке досі залишається непростим для існуючих AI-систем: одні моделі ефективно працюють з текстом, інші аналізують зображення, але жодна окрема система не є універсальною, здатною охопити всі аспекти складної проблеми. Це викликає потребу в комбінуванні різних моделей, здатних взаємодоповнювати одна одну та працювати як єдиний інтелектуальний механізм. Тобто постає питання інтеграції декількох AI-моделей у спільний процес, який дозволяє отримувати більш точні, логічні та контекстно обґрунтовані рішення.

Метою такого підходу є підвищення ефективності генеративних і аналітичних AI-систем завдяки розподілу функцій між окремими моделями. Це дає можливість охопити ширший спектр завдань і зменшити кількість помилок, характерних для роботи монолітних моделей. Основні проблеми в цьому напрямку стосуються формування оптимальної архітектури взаємодії, забезпечення узгодженості результатів та максимального використання сильних сторін кожного інструмента.

Спільне використання AI-моделей передбачає розуміння їх видів та принципів взаємодії. Серед основних типів виділяють: мовні моделі (LLM) для опрацювання природної мови; моделі комп’ютерного зору для аналізу візуальної інформації; рекомендаційні системи, які формують персоналізовані поради; а також експертні системи, що використовують формалізовані правила. Кожен із цих типів має власну спеціалізацію, що робить їх незамінними у відповідних нішах.

Інтеграція моделей у єдину структуру може здійснюватися за різними архітектурними принципами. Ансамблеві підходи об’єднують кілька алгоритмів, порівнюють їхні висновки та формують рішення на основі статистичного узгодження. Модельні конвеєри (pipelines) працюють послідовно: вихід однієї моделі стає вхідними даними для наступної. Агентні системи натомість складаються з автономних модулів, що взаємодіють між собою, розподіляють завдання та координують процес розв’язання складних проблем. Порівняльна характеристика основних архітектурних принципів інтеграції наведена у табл. 1.

Таблиця 1 – Архітектурні принципи інтеграції AI-моделей

Архітектурний підхід	Принцип взаємодії	Основна мета	Приклад сценарію
Ансамблеві методи (Ensembles)	Паралельна робота моделей; статистичне узгодження результатів.	Зменшення похибки, підвищення точності.	Класифікація складних об’єктів шляхом голосування трьох різних моделей.
Модельні конвеєри (Pipelines)	Послідовна обробка: вихід однієї моделі є входом для наступної.	Поетапна декомпозиція складного процесу.	Текст – Узагальнення – Генерація зображення за описом.
Агентні системи (Autonomous Agents)	Автономна взаємодія, розподіл завдань, координація.	Вирішення неструктурованих комплексних проблем.	Агент-програміст пише код, а агент-тестувальник перевіряє його і повертає на доопрацювання.

Незважаючи на вагомі переваги, багатомодельний підхід має певні обмеження: збільшення обчислювальних витрат, складність налаштування та підвищені вимоги до контролю якості даних, оскільки помилка одного компонента може вплинути на загальний результат. З огляду на це, важливим етапом проектування багатомодельних систем стає впровадження механізмів валідації проміжних результатів та розробка сценаріїв автоматичного відновлення після збоїв, що дозволяє мінімізувати ризики каскадних помилок.

Важливу роль у реалізації багатомодельних систем відіграють платформи, що дозволяють організовувати роботу кількох AI-моделей у єдиному середовищі. Серед таких інструментів – LangChain (створення ланцюжків взаємодій), екосистема OpenAI (модульна взаємодія агентів) та HuggingFace Pipelines [3]. Ці платформи значно спрощують оркестрацію процесів та інтеграцію різнорідних AI-компонентів. Крім того, вони забезпечують уніфікований інтерфейс доступу до моделей, що дає розробникам можливість експериментувати з різними алгоритмами без необхідності глибокої переробки програмної архітектури.

Для масштабування подібних систем активно застосовуються хмарні сервіси. Вони надають необхідні ресурси, дозволяють виконувати моделі паралельно та забезпечують гнучке керування середовищами. Завдяки цьому користувачі можуть швидко розгортати складні архітектури та адаптувати їх до змінних потреб. Зокрема, використання безсерверних (serverless) обчислень для інференсу моделей дозволяє динамічно розподіляти навантаження та оптимізувати витрати, використовуючи ресурси лише в моменти активної обробки запитів.

Важливими є також технології контейнеризації та API. Інструменти, такі як Docker та Kubernetes дозволяють ізолювати окремі моделі та забезпечити стабільність їх роботи. API забезпечують взаємодію між сервісами, дозволяючи створювати модульні конструкції, де кожна частина виконує власну функцію, але разом вони формують цілісний механізм [3, 4]. Такий підхід фактично реалізує мікросервісну архітектуру в сфері AI, гарантуючи, що оновлення або заміна однієї моделі не порушить функціонування всієї системи. Систематизація розглянутих інструментальних засобів за рівнями реалізації системи подана у табл. 2.

Таблиця 2 – Інструментальні засоби реалізації багатомодельних систем

Рівень системи	Функція в системі	Технології
Оркестрація та логіка	Керування потоками даних, створення ланцюжків (Chains), пам'ять.	LangChain, HuggingFace Pipelines, OpenAI Assistants API
Виконавчі моделі	Безпосередня генерація контенту та аналіз даних.	LLM (GPT, Llama), Computer Vision models, Expert Systems
Інфраструктура та розгортання	Ізоляція середовища, масштабування, обчислювальні ресурси.	Docker, Kubernetes, Хмарні платформи (AWS, Azure, GCP)

Критичним аспектом інтеграції є управління даними та потоками між моделями. Необхідно забезпечити коректну передачу інформації, контроль якості, узгодження форматів та синхронізацію обчислень. Надійний механізм керування потоками, який забезпечують згадані вище інструменти оркестрації (зокрема LangChain), гарантує стабільність системи та дозволяє досягати максимальної ефективності під час автоматичного генерування рішень.

Значний внесок у розвиток цієї галузі роблять провідні технологічні компанії: Google просуває мультимодальні рішення; OpenAI розвиває агентні підходи; Meta працює над аналізом мультимодальних даних; Microsoft інтегрує ці рішення у корпоративні платформи [5-7]. Їх досягнення сприяють стандартизації нових практик.

Значення багатомодельного підходу для майбутнього розвитку штучного інтелекту є визначальним. Такі системи створюють основу для рішень, здатних діяти узгоджено, аналізувати ситуації багатогранно та адаптуватися до нових умов. Перспективи їх використання постійно розширюються у бізнесі, медицині, логістиці та робототехніці, роблячи багатомодельні рішення невіддільним елементом технологічного майбутнього.

Список використаних джерел

1. Сагайда П. І., Костіков О. А., Добряк С. К. Метод застосування агентів штучного інтелекту в багатоагентній системі для автоматизації процесів інтелектуального аналізу даних. В*існик Херсонського національного технічного університету*, 2024. № 4(91). С. 43–51. DOI: 10.35546/kntu2078 -4481.2024.4.43.

2. Капітон А. М., Тищенко Д. О., Десятко А. М., Лазоренко В. В. Еволюція та аналіз розвитку мультимодальних систем штучного інтелекту. *Системи управління, навігації та зв'язку*, 2024. Т. 4, № 78. С. 75–78. DOI: 10.26906/SUNZ.2024.4.075. URL: https://reposit.nupp.edu.ua/handle/PoltNTU/ 17600 (дата звернення 23.11.2025).

3. Aguilera F. M. A Powerful Synergy: Ollama, LangChain, Docker, and Kubernetes for AI Applications. *Medium: вебсайт*. URL: https://medium.com/thedeephub/a-powerful-synergy-ollama-langchain-docker-and-kubernetes-for-ai-applications-7e70aaa59618 (дата звернення: 23.11.2025).

4. MLOps: Containerizing Multimodal Conversational AI Assistants with Docker and Kubernetes. *Medium: вебсайт*. URL: https://medium.com/ %40lohitakshyogi/mlops-containerizing-multimodal-conversational-ai-assistants-with-docker-and-kubernetes-e4d0d922afb1 (дата звернення: 23.11.2025).

5. Stein R. Bringing multimodal search to AI Mode. *Google: вебсайт*. URL: https://blog.google/products/search/ai-mode-multimodal-search/ (дата звернення: 23.11.2025).

6. AI-first companies driving the future with Microsoft AI. *Microsoft: вебсайт*. URL: https://www.microsoft.com/en-us/startups/blog/5-startups-to-watch/ (дата звернення: 23.11.2025).

7. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. *Meta: вебсайт*. URL: https://ai.meta.com/blog/llama-4-multimodal-intelligence/ (дата звернення: 23.11.2025).

Горчакова Марина, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.т.н., доцент Слюсарь Ігор

АВТОМАТИЗАЦІЯ ПІДБОРУ РЕЙСІВ ТРАНСПОРТУ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Автоматизація процесів у сфері транспортних перевезень є одним з ключових напрямів цифрової трансформації, оскільки сучасні користувачі очікують швидкого доступу до актуальної інформації про маршрути, розклад і доступність квитків. Традиційний пошук рейсів автобусного та залізничного транспорту через сайти чи мобільні додатки потребує часу та певних навичок роботи з інтерфейсом. У відповідь на це зростає потреба у створенні інтелектуальних систем, здатних надавати інформацію у зручній та максимально швидкій формі. Одним із найбільш перспективних підходів є використання чат-ботів з підтримкою штучного інтелекту, зокрема моделей типу ChatGPT [1].

Інтелектуальний чат-бот дозволяє користувачеві формулювати запит у природній мові, що значно спрощує взаємодію із системою. На відміну від класичних пошукових форм, які вимагають точного введення дати, маршруту чи параметрів подорожі, ChatGPT здатен розпізнавати навіть неточні, розмовні або складні запити (рис. 1). Наприклад: «Знайди мені поїзд завтра з Полтави до Києва після 10 ранку» або «Потрібен автобус до Харкова ввечері». Штучний інтелект інтерпретує зміст запиту, визначає ключові параметри та формує точний пошуковий запит до системи розкладів.

🔍 Розклад, час і вартість

Час відправлення з Полтави	Час прибуття в Київ	Тривалість в дорозі	Вартість	Примітки
02:00	07:15	~5 год 15 хв	~ 780 грн	Автобус Setra, відправлення з автовокзалу Полтава InBus+2bus-tickets.com.ua+2
04:00	09:15	~5 год 15 хв	~ 780 грн	InBus+2bus-tickets.com.ua+2
06:00	11:15	~5 год 15 хв	~ 780 грн	bus-tickets.com.ua+2bus-tickets.com.ua+2
08:15	13:30	~5 год 15 хв	~ 780 грн	bus-tickets.com.ua+2InBus+2

📍 Де відправлення і прибуття

- Відправлення з **Полтави** — Автовокзал, вул. Великотирновська, 7. InBus+1
- Прибуття в **Київ** — Автостанція Київ, вул. Симона Петлюри 32. InBus+2Зелений Слон 7+2

Рисунок 1 – Приклад обробки запиту за допомогою ChatGPT

Автоматизація підбору рейсів базується на взаємодії чат-бота з зовнішніми джерелами даних: API автобусних операторів, онлайн-розкладів «Укрзалізниці» або транспортних агрегаторів [2]. Після отримання даних про доступні поїзди чи автобуси, система обробляє інформацію, сортує її за релевантністю та пропонує користувачу оптимальні варіанти. У разі необхідності чат-бот може повторно уточнити параметри – наприклад, час, бюджет, тип транспорту або кількість пересадок. Це робить взаємодію максимально схожою на діалог з живим консультантом.

Важливим елементом системи є контекстна пам’ять [3], яку забезпечує модель ChatGPT. Завдяки цьому чат-бот може запам’ятовувати попередні повідомлення протягом сесії та розуміти логіку діалогу. Наприклад, після запиту «Покажи дешевші рейси» система знатиме, що треба відфільтрувати результати попереднього пошуку. Така взаємодія робить роботу сервісу більш інтуїтивною та зручною, особливо для людей, які не мають досвіду роботи зі звичайними вебресурсами.

Окрім пошуку рейсів, інтелектуальний чат-бот може виконувати додаткові функції: прогнозувати тривалість поїздки, оцінювати затримки, підбирати пересадки, надсилати нагадування про час відправлення та навіть формувати індивідуальні маршрути для багатосегментних подорожей. Інтеграція аналітичних алгоритмів дозволяє системі рекомендувати найзручніші або найшвидші варіанти на основі історичних даних.

З технічної точки зору, створення такого чат-бота потребує реалізації декількох ключових компонентів: модуля обробки запитів природною мовою, інтерфейсу взаємодії з API транспортних систем, логіки обробки результатів і генерації відповідей, а також механізмів обліку контексту та персоналізації. Модель ChatGPT може виступати як центральний інтелектуальний модуль, який забезпечує аналіз, інтерпретацію та формування відповідей. Через API транспортних систем чат-бот може отримувати інформацію, наприклад:

– розклад руху (автобуси, поїзди, тролейбуси, метро);
– GPS-трекінг – поточне місцезнаходження транспорту;
– стан маршруту (затримки, ремонти, відміни рейсів);
– інформація про квитки та тарифи;
– пошук оптимального маршруту (наприклад, API на кшталт Google Directions, OpenTripPlanner);
– інформація про транспортні оператори, платформи, зупинки, пересадки.

У підсумку, інтеграція ChatGPT у транспортні інформаційні системи відкриває нові можливості для автоматизації та покращення користувацького досвіду. Такий чат-бот значно спрощує процес пошуку рейсів, робить його швидким, доступним і персоналізованим, а також може стати основою для створення інтелектуальних навігаційних сервісів. Використання ШІ у транспортній сфері є перспективним напрямом, що сприяє підвищенню ефективності перевезень та розвитку сучасної цифрової інфраструктури.

Список використаних джерел

1. ChatGPT. URL: https://chatgpt.com.
2. АвтоТрансГарант. URL: https://www.autotransgarant.com.
3. OpenAI. ChatGPT Model System Card and Technical Overview. 2023.

Гребінченко Едуард, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.с.-г.н., доцент Протас Надія

ДОСЛІДЖЕННЯ СУЧАСНИХ ПІДХОДІВ НЕЙРОМЕРЕЖЕВОГО ОЦІНЮВАННЯ ЯКОСТІ ЗОБРАЖЕНЬ БЕЗ ЕТАЛОНУ (NO-REFERENCE IQA) ДЛЯ РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКА

У сучасному світі цифрових технологій якість фотозображень відіграє ключову роль у багатьох сферах: від соціальних мереж і мобільної фотографії до медичної діагностики, систем відеоспостереження та обробки зображень у реальному часі. З поширенням смартфонів користувачі щодня створюють мільйони фотографій, які можуть бути зіпсованими через шум, розмиття, компресію чи низьке освітлення. Традиційні методи оцінювання якості зображень (Image Quality Assessment, IQA) поділяються на повноеталонні (Full-Reference, FR-IQA), які потребують ідеального референсного зображення, та безеталонні (No-Reference, NR-IQA або Blind IQA). Останні є найбільш актуальними для реальних сценаріїв, оскільки еталонне зображення зазвичай відсутнє. Розроблення мобільного додатка для автоматичного NR-IQA дозволить користувачам миттєво оцінювати якість фото, рекомендувати покращення (наприклад, перезйомку чи фільтри) та інтегрувати функцію в камери смартфонів. Це особливо важливо в умовах обмежених обчислювальних ресурсів мобільних пристроїв, де потрібні легкі та ефективні моделі на основі нейронних мереж.

Проведений аналіз джерел показав швидкий розвиток NR-IQA завдяки глибокому навчанню. Класичні методи, такі як BRISQUE чи NIQE, базувалися на статистичних особливостях зображень, але мали обмежену точність для складних дисторсій. З появою convolutional neural networks (CNN) та transformer-моделей ситуація змінилася. У огляді [1] підкреслюється перехід від ручних фіч до енд-ту-енд навчання, де моделі навчаються безпосередньо прогнозувати суб'єктивні оцінки якості (MOS – Mean Opinion Score) на великих датасетах, таких як KonIQ-10k чи LIVE-in-the-Wild.

Одним з піонерських підходів є модель KonCept512, описана в роботі [2]. Автори зібрали датасет KonIQ-10k з 10 073 реальних фотографій, оцінених crowdsourcing-ом, і запропонували CNN на базі ResNet-50 з адаптацією для NR-IQA. Модель досягає високої кореляції з людськими оцінками (SRCC > 0.92 на тестових даних) і добре узагальнюється на «дикі» зображення.

Подальший прогрес пов'язаний з transformer-архітектурами. У роботі [3] представлено MUSIQ (Multi-scale Image Quality Transformer). Ця модель обробляє зображення різного розміру завдяки patch-based підходу та multi-scale encoding, що дозволяє захоплювати як локальні дисторсії (шум, розмиття), так і глобальний контекст. MUSIQ перевершує CNN-моделі на датасетах KonIQ-10k (PLCC ~0.96) та SPAQ, демонструючи перевагу attention-механізмів для ігнорування семантичного контенту на користь дисторсій.

Ще одним потужним рішенням є HyperIQA [4]. Модель використовує hypernetwork для генерації ваг основної мережі залежно від типу дисторсії, що

підвищує адаптивність. HyperIQA показує високі результати на синтетичних (LIVE, TID2013) та автентичних датасетах, з акцентом на легкість – кількість параметрів менша, ніж у повних transformer. Огляд 2024–2025 років (наприклад, [5]) вказує на тенденцію до гібридних моделей та використання pre-trained backbone (Swin Transformer, CLIP). Нові підходи, такі як TReS чи MANIQA, інтегрують relative ranking та self-consistency для покращення монотонності прогнозів. Для мобільних додатків актуальні lightweight-моделі, наприклад LAR-IQA, яка зменшує обчислювальну складність при збереженні точності. У основній частині дослідження порівняно ключові метрики: CNN-моделі (KonCept512, HyperIQA) ефективні для швидкого інференсу на CPU/GPU смартфонів, але transformer (MUSIQ) краще справляються з multi-scale дисторсіями. Для мобільного додатка пропонується базова архітектура: pre-trained Swin-Tiny або EfficientNet як backbone, fine-tuning на комбінації датасетів (KonIQ-10k + PIPAL для GAN-дисторсій), з виходом скалярної оцінки якості (0–100) та класифікацією типів деградації (розмиття, шум тощо). Обмеження мобільних пристроїв вимагають квантизації моделі (до 8-біт) та оптимізації (TensorFlow Lite або PyTorch Mobile). Дослідження сучасних NR-IQA підходів показує, що transformer-based моделі (MUSIQ, HyperIQA) забезпечують найкращу кореляцію з людським сприйняттям, перевищуючи класичні методи на 15–20%. Для розроблення мобільного додатка рекомендується гібридний підхід: легкий transformer з hypernetwork для адаптації до реальних фото. Подальші кроки – вибір датасету, fine-tuning та тестування на Android. Це дозволить створити практичний інструмент для покращення користувацького досвіду в мобільній фотографії.

Список використаних джерел

1. Q. Mao et al. No-Reference Image Quality Assessment: Past, Present, and Future. *Expert Systems*, 2025. Vol. 42, no. 3. URL: https://doi.org/10.1111/exsy.13842.

2. V. Hosu et al. KonIQ-10k: An Ecologically Valid Database for Deep Learning of Blind Image Quality Assessment. *IEEE Transactions on Image Processing*, 2020. Vol. 29. P. 4041–4056. URL: https://doi.org/10.1109/tip.2020.2967829.

3. J. Ke et al. MUSIQ: Multi-scale Image Quality Transformer. *2021 IEEE/CVF Int. Conf. on Computer Vision* (Montreal, QC, Canada, 10-17 Oct. 2021), 2021. URL: https://doi.org/10.1109/iccv48922.2021.00510.

4. Y. Shen et al. Interpreting the Latent Space of GANs for Semantic Face Editing. *2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (Seattle, WA, USA, 13-19 June 2020), 2020. URL: https://doi.org/10.1109/cvpr42600.2020.00926.

5. Q. Mao et al. No-Reference Image Quality Assessment: Past, Present, and Future. *Expert Systems*, 2025. Vol. 42, no. 3. URL: https://doi.org/10.1111/exsy.13842.

Даценко Назар, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Флегантов Леонід

АРХІТЕКТУРА ТА ПРИНЦИПИ РОБОТИ ФРЕЙМВОРКУ PLAYWRIGHT

Playwright – це передовий фреймворк автоматизації тестування програмного забезпечення з відкритим вихідним кодом від Microsoft, який є галузевим стандартом для створення швидких і надійних end-to-end тестів вебдодатків, а також для вебскрапінгу. Він має офіційні API для JavaScript/TypeScript, Python, Java та C#/.NET, а також забезпечує безшовну мультибраузерну підтримку (Chromium, Firefox, WebKit/Safari) через єдиний API, використовуючи протокол DevTools для безпосередньої та високошвидкісної взаємодії з браузерними рушіями.

Playwright є одним із найбільш поширених фреймворків для автоматизації, що стрімко зростає у популярності серед QA-інженерів, про що свідчить позитивна динаміка його завантажень (рис. 1). Його широке визнання обумовлене універсальністю: це open-source рішення на базі Node.js, яке підтримує крос-платформність (Windows, Linux, macOS), легко інтегрується в CI/CD процеси (наприклад, Jenkins) та має офіційні API для основних мов програмування.

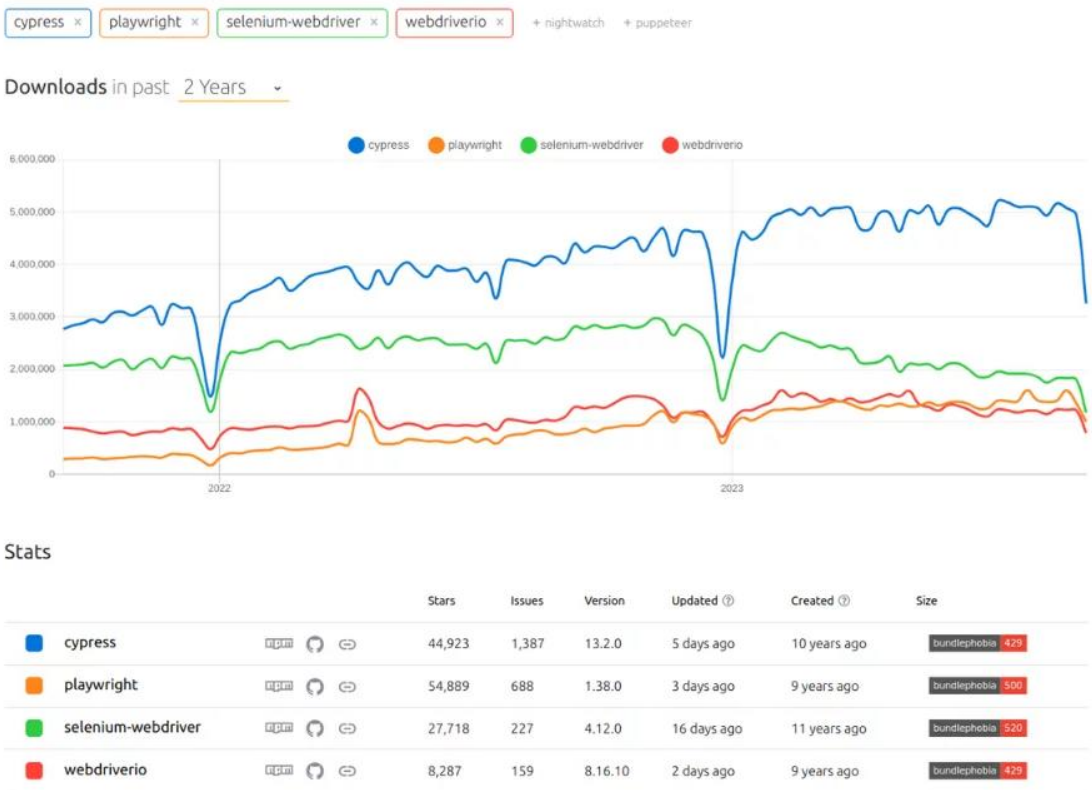

Рисунок 1 – Статистика використання автоматизованих фреймворків для тестування вебдодатків

Актуальність Playwright підкріплена його архітектурою, яка, працюючи на основі Node.js та використовуючи WebSocket для низьколатентної комунікації, вирішує проблєму нестабільності тестів: він реалізує інтелектуальні механізми

автоматичного очікування (Smart Waiting та Actionsability Checks), які гарантують, що взаємодія відбувається лише після того, як елемент стає видимим, стабільним та активним. Як повноцінний фреймворк, він надає комплексне середовище з вбудованим тестовим ранером та інструментами для трасування (Trace Viewer), що дозволяє записувати та відтворювати повний лог виконання тесту. Крім того, Playwright підтримує виконання кожного тесту у повністю ізольованому «браузерному контексті» для забезпечення чистоти даних (cookies, сховище) та підтримує повністю паралельне виконання, що суттєво прискорює прогони тестів у крос-платформних CI/CD конвеєрах (Windows, Linux, macOS, Jenkins).

Playwright має принципово відмінну від інших засобів автоматизації тестування, позапроцесну архітектуру, де сам тестовий скрипт та браузерний рушій існують як абсолютно незалежні процеси, що взаємодіють через протоколи віддаленого керування (наприклад, CDP для Chromium), а не шляхом безпосереднього втручання в сторінку (рис. 2).

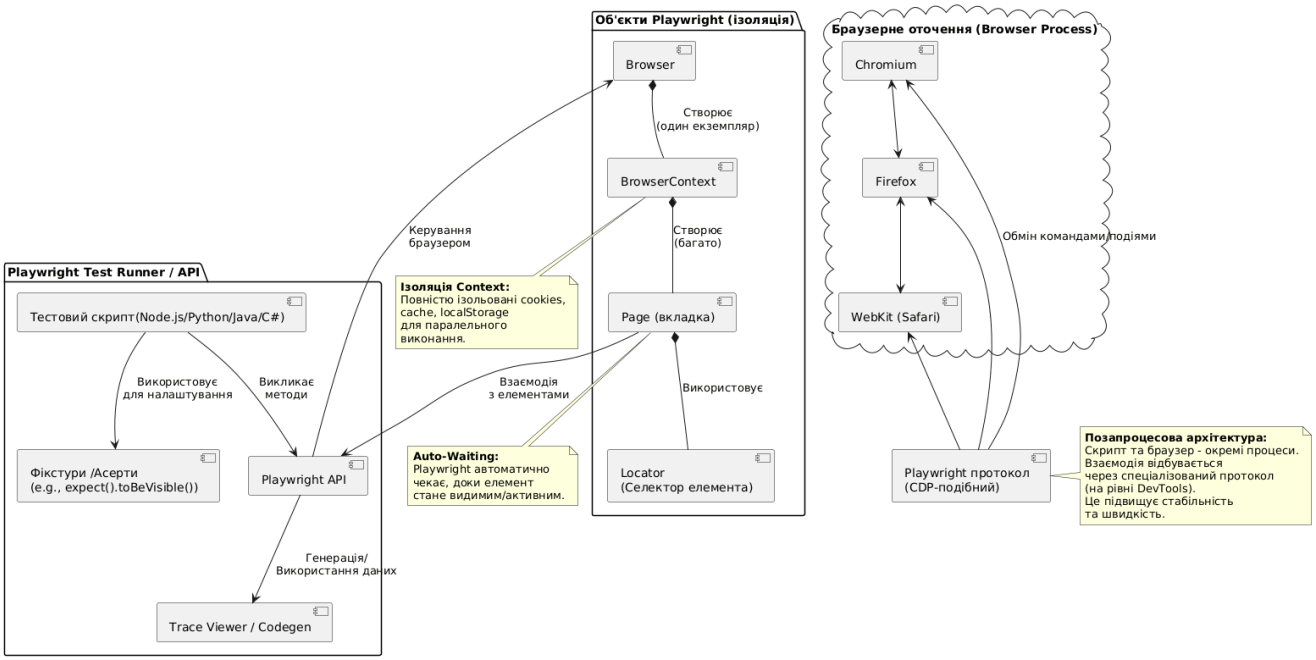

Рисунок 2 – Позапроцесна архітектура тестування Playwright

Цей підхід забезпечує високу стійкість тестів (скрипт не залежить від зависання вебсторінки) та знімає обмеження безпеки браузера, надаючи повний зовнішній контроль, включаючи керування кількома вкладками, фреймами та мережевою активністю сторінки (що дозволяє QA-інженеру мокувати (mock) відповіді бекенду або блокувати непотрібні запити). В основі тестової ізоляції лежить концепція BrowserContext – це легковаговий, ізольований сеанс (подібний до режиму інкогніто), який має власний набір Cookies, LocalStorage та кешу, повністю усуваючи необхідність ручного менеджменту браузерних драйверів, як це було у Selenium. Саме ця ізоляція є технічною основою для інтегрованої у рідний @playwright/test ранер функції паралельного виконання тестів, де кожен тест запускається у власному чистому контексті, значно скорочуючи загальний час прогону. Playwright вирішує ключову проблєму нестабільності (flakiness) за

допомогою механізму автоматичного очікування (Auto-Waiting): перед виконанням будь-якої дії він запускає цілий набір Actionsability Checks (перевірок на видимість, активність, відсутність перекриття), усуваючи потребу в ручних командах очікування, а функціонал Web-First Assertions розширює це очікування на команди expect() (наприклад, toBeVisible()), повторюючи перевірку, доки умова не буде виконана. Додатково, тестовий ранер запроваджує потужний патерн фікстур, що значно спрощує налаштування та очищення тестового середовища (наприклад, автоматизація логіну адміністратора), роблячи тестовий код чистим, лаконічним та легким для підтримки.

Розглянемо його застосування на прикладі простого сценарію тестування, де Playwright використовується для імітації дій користувача: потрібно перевірити, чи коректно працює пошук на сайті документації Playwright. Цей сценарій реалізується через таку послідовність дій:

– відкрити вебсторінку https://playwright.dev/;
– натиснути на кнопку пошуку, щоб активувати поле вводу;
– ввести текст «install» у поле пошуку;
– перевірити, що в результатах пошуку з’явився пункт «Installation».

Розглянемо, як цей сценарій реалізується на трьох поширених мовах програмування, які підтримує Playwright: TypeScript (Node.js), Python та Java.

TypeScript – це найпоширеніший спосіб використання Playwright, оскільки він має вбудований тестовий фреймворк Playwright Test Runner. Нижче подано приклад використання Playwright на мові TypeScript.

```
// example.spec.ts
import { test, expect } from ‘@playwright/test’;

test(‘Пошук документації Playwright’, async ({ page }) => {
  // 1. Відкрити сторінку
  await page.goto(‘https://playwright.dev/’);

  // 2. Натиснути кнопку пошуку
  // Шукаємо кнопку з текстом "Search" або натискаємо Ctrl+K
  // (якщо це desktop)
  await page.getByRole(‘button’, { name: ‘Search’ }).click();

  // 3. Ввести текст "install" у поле пошуку
  await page.getByPlaceholder(‘Search docs’).fill(‘install’);

  // 4. Перевірити, що в результатах з’явився пункт "Installation"
  const searchResult = page.locator(‘.DocSearch-Hits .DocSearch-Hit-title:has-
text("Installation")’);
  await expect(searchResult).toBeVisible();
});
```

У Python Playwright часто використовується з фреймворком pytest. У наступному прикладі використовується чистий Playwright API в асинхронному режимі.

```
# test_search.py
import asyncio
from playwright.async_api import async_playwright, expect

async def test_playwright_documentation_search():
    async with async_playwright() as p:
```

```
            # Запускаємо Chromium
            browser = await p.chromium.launch()
            page = await browser.new_page()

            # 1. Відкрити сторінку
            await page.goto("https://playwright.dev/")

            # 2. Натиснути кнопку пошуку
            await page.get_by_role("button", name="Search").click()

            # 3. Ввести текст "install" у поле пошуку
            await page.get_by_placeholder("Search docs").fill("install")

            # 4. Перевірити, що в результатах з'явився пункт "Installation"
            search_result = page.locator('.DocSearch-Hits .DocSearch-Hit-title:has-
text("Installation")')
            await expect(search_result).to_be_visible()

            await browser.close()

    # Запуск тесту
    if __name__ == "__main__":
        asyncio.run(test_playwright_documentation_search())
```

У Java Playwright часто використовується разом із JUnit або TestNG.

```
    // PlaywrightSearchTest.java
    import com.microsoft.playwright.*;
    import com.microsoft.playwright.options.*;
    import org.junit.jupiter.api.*;

    import static
com.microsoft.playwright.assertions.PlaywrightAssertions.assertThat;

    public class PlaywrightSearchTest {

        @Test
        void playwrightDocumentationSearch() {
            try (Playwright playwright = Playwright.create()) {
                Browser browser = playwright.chromium().launch();
                BrowserContext context = browser.newContext();
                Page page = context.newPage();

                // 1. Відкрити сторінку
                page.navigate("https://playwright.dev/");

                // 2. Натиснути кнопку пошуку
                page.getByRole(AriaRole.BUTTON, new
Page.GetByRoleOptions().setName("Search")).click();

                // 3. Ввести текст "install" у поле пошуку
                page.getByPlaceholder("Search docs").fill("install");

                // 4. Перевірити, що в результатах з'явився пункт "Installation"
                Locator searchResult = page.locator(".DocSearch-Hits .DocSearch-
Hit-title:has-text('Installation')");
                assertThat(searchResult).isVisible();

                browser.close();
            }
        }
    }
```

Як бачимо, незважаючи на різницю в синтаксисі мов, структура та методи API Playwright (page.goto(), page.getByRole(), page.locator(), expect()) залишаються практично ідентичними, що підкреслює універсальність фреймворку. Однією з вагомих можливостей, яку надає архітектура Playwright, є Playwright Trace Viewer – головний інструмент для налагодження та аналізу збоїв тестів, який значно спрощує процес пошуку причин нестабільності порівняно зі звичайним читанням логів. Він працює шляхом запису всіх дій тесту (наприклад, click, fill, goto), знімків DOM сторінки до та після кожної взаємодії, повного журналу мережевих запитів та консольних логів, зберігаючи ці дані у єдиному файлі трасування. Після запуску тесту з опцією трасування (наприклад, npx playwright test --trace on-first-retry) цей файл відкривається у візуальному інтерфейсі командою npx playwright show trace, де користувач може «перемотувати» виконання тесту, переглядаючи Snapshots для порівняння станів сторінки, аналізуючи панель Log, яка показує, чому Playwright не зміг виконати дію (наприклад, «Елемент невидимий» або «перекритий»), та використовуючи панель Network, щоб перевірити, чи коректно відпрацювали API-запити, що робить процес налагодження швидким, візуальним та високоінформативним.

Список використаних джерел

1. Playwright. Documentation, Guides, and API Reference. Microsoft. URL: https://playwright.dev/ (дата звернення: 04.11.2025).

2. Sami S. M.. Architecting reliable web tests with Playwright. Smashing Magazine. 2021. URL: https://www.smashingmagazine.com/2021/04/architecting-reliable-web-tests-playwright/ (дата звернення: 04.11.2025).

3. Lichteblau A., Schmid C. Playwright: Deep Dive into the Modern Browser Automation Framework. Software-Testing.com. 2023. URL: https://www.software-testing.com/playwright-deep-dive-into-the-modern-browser-automation-framework/ (дата звернення: 04.11.2025).

4. Dhar S. End-to-End Testing with Playwright: How to Use Advanced Features for Stable Tests. LogiGear Magazine. 2022. URL: https://www.logigear.com/magazine/playwright-advanced-features-for-stable-tests/ (дата звернення: 04.11.2025).

5. Team. Playwright GitHub Repository. GitHub. URL: https://github.com/microsoft/playwright (дата звернення: 04.11.2025).

Дузенко Олександра, здобувачка вищої освіти СВО «Магістр»,
спеціальність 126 «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Олена Копішинська

ОСОБЛИВОСТІ ІНСТРУМЕНТІВ BACKEND-РОЗРОБКИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ФУНКЦІЙ ВЕБДОДАТКІВ

При розробці сучасного вебдодатку розрізняють два поняття як напрямки: frontend- та backend-розробка. Вони тісно пов'язані між собою, але це зовсім різні напрями програмування як по типу задач, що вони вирішують, так і по інструментарію, що використовується. Під поняттям frontend мається на увазі видима для користувача (клієнтська) частина застосунку (не обов'язково сайту), а під поняттям backend – все те, що приховано від користувача (задній план, за екраном). Важливо розуміти, що для обох складових вебдодатку існує велика кількість різноманітних інструментів розробки, і однією із задач сучасного веброзробника є підбір найкращого набору інструментів саме для його потреб. Для цього потрібно провести аналіз наявних рішень.

Клієнт спілкується з серверною частиною за допомогою HTTP–запитів. В цих запитах міститься посилання на кінцеву точку (URL-адреса на сервері, яка може приймати запити), тип запиту, та за необхідністю додаткова корисна інформація – payload. Графічне зображення класичної взаємодії клієнту (frontend) та серверу (backend) наведено на рис. 1.

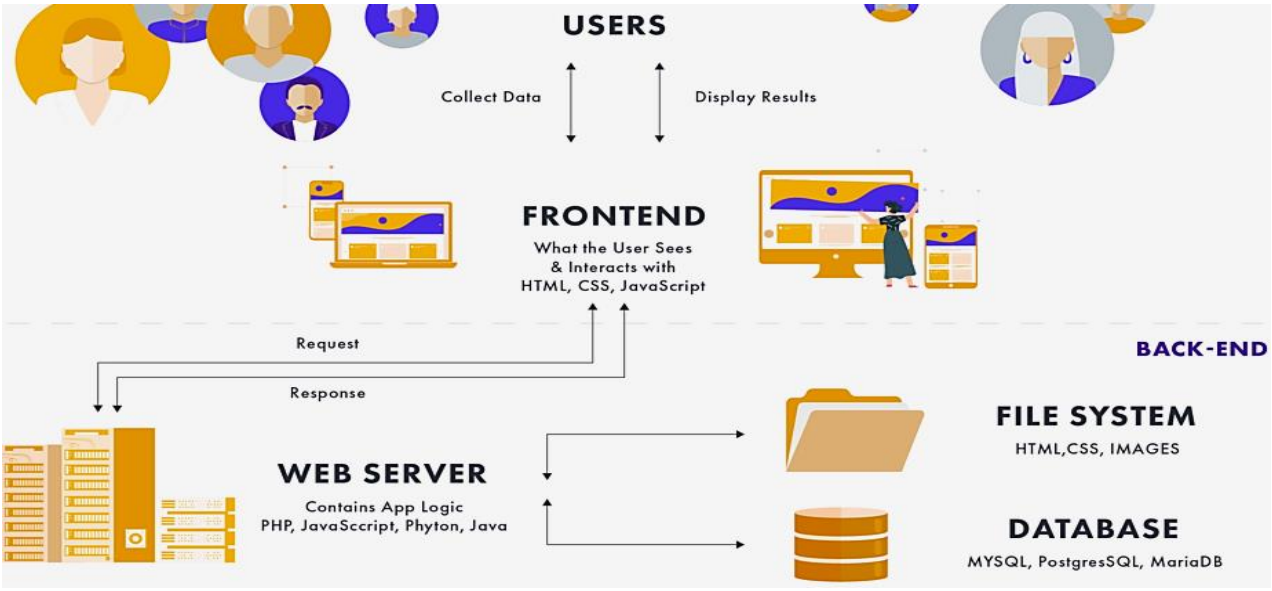

Рисунок 1 – Схема взаємодії frontend і backend частин вебдодатку

Після обробки запиту сервер повертає відповідь клієнту, яка містить в собі інформацію, що передбачена розробником. Сервер (тобто backend частина вебдодатку) повинен виконувати наступні функції: обробка HTTP–запитів, обробка бізнес–логіки, взаємодія з базою даних, авторизація та аутентифікація, логування. Вебсервер зберігає та надає доступ до контенту вебсайту – зображень, тексту, відео.

Хоча вебсервери розміщують вебсайти, доступні в інтернеті, їх використовують і для зв'язку між вебклієнтами та серверами в локальних мережах

компанії (Intranet). Вебсервери можуть обмежувати швидкість відповіді окремим клієнтам, щоб не допустити перевантаження ресурсів [1].

Прикладами таких програмних засобів є Apache та Nginx. Результати дослідження [2] показали, що в більшості тестувань Apache виявився швидшим за Nginx, з чого можна зробити висновок, що Apache краще підходить для вебсайтів з великою кількістю запитів в хвилину.

Backend-частину вебсайту пишуть на таких мовах програмування, як Java, Python, Ruby, Node.js, PHP. Проте, зараз мало хто використовує мову програмування саму по собі. Переважна більшість backend-розробників в своїй роботі використовують фреймворки.

Деякі частини коду можуть бути легко перевикористані між клієнтом і сервером. Це особливо корисно для валідації даних, форматування та бізнес-логіки. JavaScript працює з JSON безпосередньо, що спрощує обмін даними між клієнтом та сервером. Це може забезпечити кращу продуктивність для певних типів програм порівняно з PHP. npm (менеджер пакетів для Node.js) надає величезну кількість бібліотек та інструментів. Багато цих інструментів можуть використовуватися як на клієнті, так і на сервері. Можливість створювати програми, які можуть рендеруватися як на сервері, так і на клієнті, покращуючи продуктивність та SEO. Node.js відмінно підходить для додатків реального часу, завдяки вбудованій підтримці WebSockets. Інструменти для розробки, тестування та налагодження можуть бути спільними для клієнтської та серверної частин. Найпопулярніші backend-фреймворки станом на кінець 2024 р. наведено на рис. 2.

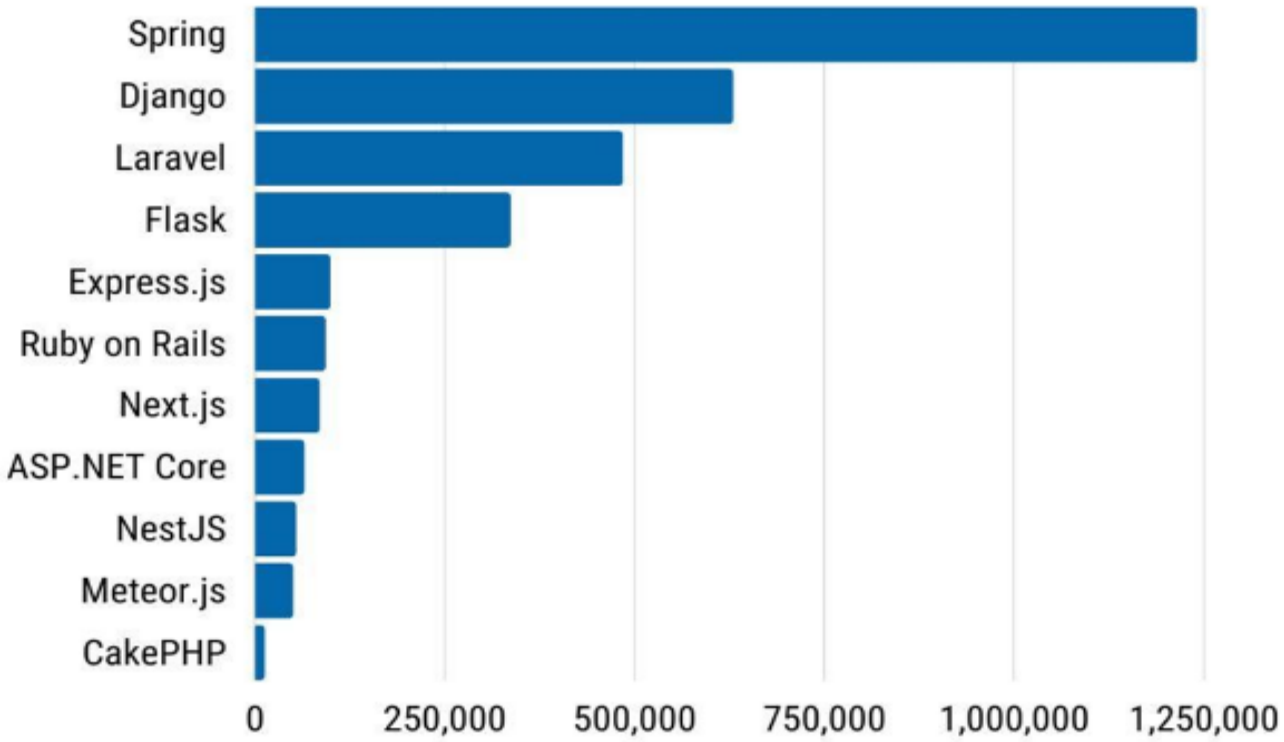

Рисунок 2 – Діаграма популярності backend-фреймворків [3]

На діаграмі видно, що до першої п’ятірки популярних фреймворків входять Spring (Java), Django (Python), Laravel (PHP), Flask та Express.JS (Node.js). Окрім мов

програмування, важливою складовою серверу є бази даних. База даних (БД) – це організована структура, призначена для зберігання, зміни й обробки взаємопов'язаної інформації. Саме в них зберігається вся інформація, пов'язана з роботою застосунку. БД бувають: централізовані, хмарні, реляційні, нереляційні, об'єктно-орієнтовані, операційні та розподілені. Найпопулярнішими є реляційні, нереляційні (NoSQL) та об'єктно-орієнтовані БД. Для звернення до реляційних та об'єктно-орієнтованих БД використовуються SQL-запити – спеціальні конструкції, написані за допомогою мови SQL. При використанні NoSQL форма запитів відрізняється в залежності від обраної БД, наприклад в MongoDB використовується мова запитів, подібна до JavaScript – MongoDB Query Language. Реляційні БД ідеально підходять для роботи з даними, структура яких не вимагає частих змін, нереляційні – для зберігання великих об'ємів неструктурованої інформації, а об'єктно-орієнтовані БД – для проєктів, де потрібна потужна база з високою функціональністю.

Для управління базами даних використовуються системи управління базами даних (СУБД). Загальне призначення СУБД полягає в наданні централізованої системи для зручного та ефективного управління даними. Перевагами використання СУБД є: багатокористувацький доступ до БД, надійність даних, можливість створення бекапів. До недоліків відноситься: комплексність, вартість, вразливість БД, часті оновлення [4].

Існують також СУБД, які спеціалізуються на різних типах сховищ: управління базами даних у пам'яті (IMDBMS), управління хмарними базами даних, де постачальник SaaS (Software as a Service) відповідає за обслуговування БД, наприклад MongoDB Atlas. Додатково використовують цілу низку інших технологій. Наприклад, інструменти тестування та налагодження API, або REST-клієнти.

Підсумовуючи основні результати, можна зробити висновок, що на сьогодні існує ціла низка ефективних технологій backend-розробки. Ключовими інструментами є мови програмування, бази даних, СУБД, мови формування запитів. Комбінування залежить від цілей і характеру вебдодатку.

Список використаних джерел

1. David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy. HTTP: The Definitive Guide. *O`Reilly Media Inc*., 2002. 976 p.

2. Zuki Pristiantoro Putro, Riza Adrianti Supono. Comparsion Analysis of Apache Nginx Webserver Load Balancing on Proxmox VE in Supporting Server Performance. *International Research Journal of Advanced Engineering and Science*, 2022. Vol. 7, Issue 3. Pp. 144-151.

3. Most Popular Backend Frameworks – 2012/2023. *Statistics & Data*. URL: https://cutt.ly/EwT5vJ0b (дата звернення: 16.11.25).

4. Dr. Mukesh Negi. Fundamental of Database Management System. *BPB Publications*, 2019. 175 p.

Єфремов Андрій, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Флегантов Леонід

ПІДГОТОВКА НАВЧАЛЬНОГО КОРПУСУ ДЛЯ МОДЕЛІ СИНТЕЗУ МОВЛЕННЯ

Стрімкий розвиток нейромережевих технологій синтезу мовлення (TTS) та перетворення голосу (VC) висуває нові, більш жорсткі вимоги до якості даних для навчання нейронних мереж [1, 2]. Сучасні end-to-end архітектури здатні генерувати звучання, наближене до людського, проте їх ефективність залежить від чистоти, збалансованості та технічної уніфікації вхідного корпусу навчальних даних [1-3]. Отже, підготовка даних є фундаментальною складовою процесу розробки сучасних систем синтезу мовлення. У роботі представлена уніфікована методологія підготовки навчального корпусу для розробки високоякісних систем нейромережевого синтезу мовлення (TTS) та перетворення голосу (VC). Застосування суворої уніфікації аудіопараметрів та текстових даних дозволило усунути систематичні зсуви та мінімізувати ризик виникнення артефактів на етапі навчання. Фіксація параметрів екстракції ознак та структури маніфестів забезпечує повну відтворюваність експериментів і сумісність між модулями TTS та VC. Сформований «заморожений» корпус є основою для об’єктивного порівняння якості різних нейромережевих архітектур. Актуальність дослідження пов’язана з відсутністю єдиних стандартів препроцесингу, що призводить до артефактів генерації (металізований звук, збої ритму) та ускладнює порівняння моделей. Окрім того, інтеграція систем TTS і VC вимагає повної сумісності акустичних ознак, що неможливо забезпечити без уніфікації параметрів обробки сигналів та нормалізації тексту. Мета полягає у створенні чистого, збалансованого та відтворюваного корпусу шляхом стандартизації аудіоформатів (24 кГц, −23 LUFS) [4], нормалізації текстових транскриптів та уніфікації екстракції акустичних ознак. Фіксація параметрів аудіопроцесора, включаючи STFT і мел-спектрограми, забезпечує сумісність модулів TTS/VC та відтворюваність експериментів. Цей підхід до уніфікації мінімізує систематичні зсуви, що є основою для об’єктивного порівняння різних нейромережевих архітектур.

Основною метою підготовки даних є формування чистого, збалансованого та відтворюваного корпусу, придатного для навчання як end-to-end TTS, так і систем перетворення голосу (VC). Процес передбачає сувору уніфікацію форматів, нормалізацію транскриптів та фіксацію зв’язків «аудіо ↔ текст/метадані» у маніфестах зі стабільною структурою. Розподіл на вибірки (train/val/test) здійснюється заздалегідь із контролем перекриття мовців для коректної оцінки узагальнюваності моделі. Аудіозаписи приводяться до уніфікованого формату: моно, частота дискретизації 24 кГц, розрядність 16-біт PCM [7]. Виконується нормалізація гучності (орієнтир – −23 LUFS), видалення надмірних пауз і слабких шумів, а також обрізка (тримінг) фраз до робочої довжини. Етап очищення включає дедуплікацію та видалення проблємних файлів (кліпінг, реверберація, сторонні звуки) ще до екстракції ознак [6-8]. Щоб уникнути зміщень у стилі чи темпі, застосовується

стратифікація даних за мовцями та тривалістю записів. Усі параметри препроцесингу документуються в єдиній конфігурації. Текстові транскрипти проходять процедуру нормалізації, що охоплює розгортання чисел, одиниць виміру, абревіатур та обробку власних назв; за потреби виконується фонемізація згідно з орфоепічними нормами [6]. Фінальні маніфести мають уніфікований формат, сумісний із завантажувачами даних, і містять шляхи до файлів, текст (або фонеми), ідентифікатор мовця, тривалість та належність до конкретного спліта (вибірки). Це гарантує прозорість перевірки якості та стабільну відтворюваність експериментів. Для забезпечення сумісності всіх підсистем (від навчання акустичної моделі до інференсу VC) використовується єдина незмінна конфігурація аудіопроцесора. Вона визначає: параметри STFT – розмір перетворення (fft_size), крок (hop_length), довжину вікна (win_length) та тип віконної функції; мел-спектрограми – кількість смуг (n_mels), частотний діапазон (fmin/fmax) та параметри нормування; просодичні ознаки – метод вилучення основного тону (F0) та міток озвученості (voicing) [5, 7]. Сталість цих параметрів дозволяє уникнути систематичних зсувів та зменшити кількість артефактів.

Деталі ініціалізації аудіопроцесора та параметри екстракції наведено на рис. 1. Логічним завершенням етапу підготовки даних є програмна реалізація механізмів їх зчитування та пакетування. Для забезпечення коректної подачі нормалізованих аудіозаписів і транскриптів у нейромережу використовується спеціалізований завантажувач, налаштування якого мають чітко узгоджуватися зі структурою вхідних файлів метаданих. Фрагмент коду, що демонструє формат полів маніфесту та відповідну конфігурацію класу завантажувача даних, наведено на рис. 2.

```
> Using model: vits
> Setting up Audio Processor...
 | > sample_rate:24000
 | > resample:False
 | > num_mels:80
 | > log_func:np.log10
 | > min_level_db:0
 | > frame_shift_ms:None
 | > frame_length_ms:None
 | > ref_level_db:None
 | > fft_size:1024
 | > power:None
 | > preemphasis:0.0
 | > griffin_lim_iters:None
 | > signal_norm:None
 | > symmetric_norm:None
 | > mel_fmin:0
 | > mel_fmax:None
 | > mel_fmax:None
 | > pitch_fmin:None
 | > pitch_fmax:None
 | > spec_gain:20.0
 | > stft_pad_mode:reflect
 | > max_norm:1.0
 | > clip_norm:True
 | > do_trim_silence:False
 | > trim_db:60
 | > do_sound_norm:False
 | > do_amp_to_db_linear:True
 | > do_amp_to_db_mel:True
 | > do_rms_norm:False
 | > db_level:None
 | > stats_path:None
 | > base:10
 | > hop_length:256
 | > win_length:1024
```

Рисунок 1 – Ініціалізація аудіопроцесора та параметри екстракції ознак [7]

```
> Training Environment:
 | > Backend: Torch
 | > Mixed precision: False
 | > Precision: float32
 | > Num. of CPUs: 12
 | > Num. of Torch Threads: 8
 | > Torch seed: 54321
 | > Torch CUDNN: True
 | > Torch CUDNN deterministic: False
 | > Torch CUDNN benchmark: False
 | > Torch TF32 MatMul: False
```

```
> DataLoader initialization
| > Tokenizer:
        | > add_blank: True
        | > use_eos_bos: False
        | > use_phonemes: True
        | > phonemizer:
                | > phoneme language: uk
                | > phoneme backend: espeak
| > Number of instances : 1766
 | > Preprocessing samples
 | > Max text length: 148
 | > Min text length: 7
 | > Avg text length: 48.86806342015855
 |
 | > Max audio length: 216598.0
 | > Min audio length: 14870.0
 | > Avg audio length: 82681.95016987542
 | > Num. instances discarded samples: 0
 | > Batch group size: 0.
```

Рисунок 2 – Структура маніфестів і конфігурація завантажувача даних [5]

На завершення підготовчої стадії, навчальний корпус перевіряється на відповідність цільовим метрикам якості (чистота, покриття фонетики, баланс мовців) і лише після цього фіксується як «заморожена» версія для навчання, валідації та подальших порівнянь між моделями.

Подальший розвиток теми передбачає використання підготовленого корпусу для навчання сучасних моделей (зокрема VITS та FastSpeech) та аналізу впливу якості препроцесингу на фінальні метрики MOS та WER [1-3]. Також перспективним є розширення методології для роботи з багатоспікерними (multi-speaker) наборами даних та впровадження автоматизованих методів аугментації для підвищення стійкості моделей до шумів.

Список використаних джерел

1. Kim J., Kong J., Son J. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. Vol. 139. Pp. 5530-5540. DOI: https://doi.org/10.48550/arXiv.2106.06103

2. Y. Ren et al. FastSpeech: Fast, Robust and Controllable Text to Speech. *Advances in Neural Information Processing Systems 32* (NeurIPS 2019). Vancouver: Curran Associates Inc., 2019. Pp. 3165-3174. DOI: https://doi.org/10.48550/arXiv.1905.09263

3. Y. Ren et al. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech. *Int. Conf. on Learning Representations* (ICLR), 2021. URL: https://openreview.net/forum?id=piLPYqxtWuA (дата звернення: 22.11.2025). DOI: https://doi.org/10.48550/arXiv.2006.04558

4. EBU R 128. Loudness normalisation and permitted maximum level of audio signals. *Geneva: European Broadcasting Union*, 2020. URL: https://tech.ebu.ch/docs/r/r128.pdf (дата звернення: 22.11.2025).

5. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32* (NeurIPS 2019). Vancouver: Curran Associates Inc., 2019. Pp. 8024-8035. DOI: https://doi.org/10.48550/arXiv.1912.01703

6. Duddington J. eSpeak NG Text-to-Speech. 2015. URL: https://github.com/espeak-ng/espeak-ng (дата звернення: 22.11.2025).

7. B. McFee et al. Librosa: Audio and Music Signal Analysis in Python. *Proceedings of the 14th Python in Science Conference* (SciPy 2015), 2015. P. 18-24. DOI: DOI: 10.25080/Majora-7b98e3ed-003

8. Ito K., Johnson L. The LJ Speech Dataset. 2017. URL: https://keithito.com/LJ-Speech-Dataset/ (дата звернення: 22.11.2025).

Коваленко Максим, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.т.н., доцент Одарущенко Олена

ВИКОРИСТАННЯ МОВИ ПРОГРАМУВАННЯ PYTHON ДЛЯ АНАЛІЗУ ДАНИХ

Сучасні інформаційні технології спрямовані на ефективну роботу з великими масивами даних, що утворюються у різних галузях діяльності людини – від фінансів і медицини до освіти, сільського господарства та промисловості. У цьому контексті мова програмування Python посідає одне з провідних місць серед інструментів для обробки та аналізу даних завдяки своїй простоті, відкритості та великій спільноті розробників [1].

Python – це інтерпретована, високорівнева мова програмування з відкритим вихідним кодом, яка поєднує зручний синтаксис та потужні можливості для роботи з математичними обчисленнями, статистикою, базами даних і візуалізацією. Завдяки наявності спеціалізованих бібліотек, таких як NumPy, Pandas, Matplotlib, Seaborn і Scikit-learn, Python став універсальним середовищем для виконання завдань аналітики, машинного навчання та обробки великих даних [55].

Однією з ключових переваг Python є можливість обробки даних із різних джерел – текстових файлів, електронних таблиць, баз даних і веб-ресурсів. Інструменти бібліотеки Pandas забезпечують функції фільтрації, сортування, групування, обчислення статистичних показників, а також дозволяють виконувати попереднє очищення даних від пропущених або некоректних значень. Це суттєво спрощує підготовку інформації до подальшого аналізу.

Крім того, Python має потужні засоби для візуалізації результатів аналізу. Бібліотеки Matplotlib та Seaborn дозволяють створювати лінійні графіки, гістограми, діаграми розсіювання, теплові карти та інші види візуальних представлень даних. Така наочність дає змогу швидко виявляти тренди, закономірності та аномалії у великих наборах інформації.

Варто відзначити, що Python активно використовується у сфері бізнес-аналітики, банківських технологій, наукових досліджень і державного управління. Його застосування дозволяє автоматизувати процеси збору та обробки інформації, створювати прогностичні моделі, оцінювати ризики та формувати рекомендації для прийняття рішень.

Особливої популярності Python набув у галузі машинного навчання. Бібліотека Scikit-learn забезпечує реалізацію численних алгоритмів класифікації, кластеризації та регресійного аналізу. На її основі створюються інтелектуальні системи прогнозування попиту, аналізу поведінки клієнтів або розпізнавання зображень. Такі інструменти стають основою сучасних цифрових технологій – від розумних міст до автономних систем управління.

Важливим чинником поширення Python є його доступність. Мова є кросплатформенною, тобто працює на операційних системах Windows, Linux, macOS та навіть мобільних платформах. Вона підтримується багатьма середовищами розробки, зокрема Jupyter Notebook, PyCharm, Visual Studio Code. Це

робить Python зручним інструментом як для досвідчених програмістів, так і для початківців.

Крім того, Python має тісну інтеграцію з іншими мовами програмування та технологіями. Його можна використовувати у зв’язці з C/C++, Java, R, а також у хмарних сервісах і платформах обчислень, таких як Google Colab або AWS. Це забезпечує гнучкість і масштабованість аналітичних рішень, що є особливо важливим для обробки великих обсягів даних (Big Data).

Отже, Python – це не просто мова програмування, а повноцінна екосистема для аналітики даних і розробки інтелектуальних систем. Вона поєднує простоту у використанні, потужний функціонал і відкритість, що робить її одним із найперспективніших інструментів сучасної інформаційної індустрії. Застосування Python сприяє підвищенню ефективності роботи фахівців з інформаційних систем і відкриває нові можливості для наукових і практичних досліджень.

Список використаних джерел

1. Цікаві факти про Python – New IT School. URL: https://www.itschool.vn.ua/interesting-python/ (дата звернення: 12.11.2025).

2. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/ (дата звернення: 12.11.2025).

Корецька Діана, здобувачка вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

ПОРІВНЯЛЬНИЙ АНАЛІЗ ІСНУЮЧИХ МОБІЛЬНИХ ФІТНЕС-ДОДАТКІВ ДЛЯ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

У сучасному світі, де малорухливий спосіб життя стає однією з головних причин ожиріння, серцево-судинних захворювань та інших неінфекційних хвороб, актуальність мобільних фітнес-додатків важко переоцінити. За даними Всесвітньої організації охорони здоров'я, недостатня фізична активність є четвертим за значимістю фактором ризику смертності у світі, а пандемія COVID-19 лише посилила цю проблему, змусивши мільйони людей шукати альтернативи тренажерним залам. Мобільні додатки для фітнесу дозволяють користувачам відстежувати активність, планувати тренування, контролювати харчування та отримувати мотивацію в будь-який час і в будь-якому місці. Згідно з аналітикою Statista та Business of Apps, ринок фітнес-додатків у 2024-2025 рр. демонструє стабільне зростання, кількість завантажень додатків категорії Health & Fitness сягає сотень мільйонів щорічно. Це робить порівняльний аналіз існуючих рішень ключовим етапом для проєктування нової інформаційної системи (ІС), яка має бути конкурентоспроможною, зручною та ефективною.

Аналіз літератури та джерел показує, що дослідження мобільних фітнес-додатків активно розвиваються з 2010-х років. У роботі [1] підкреслюється роль користувацького досвіду (UX) у задоволеності додатками, де гейміфікація та соціальні функції значно підвищують утримання користувачів. Інше дослідження застосовує якісний порівняльний аналіз (fsQCA) для вивчення факторів рекомендацій фітнес-додатків у спортзалах [2]. Бібліометричний огляд у [3] фіксує експоненційне зростання публікацій після 2015 р. Актуальний рейтинг CNET 2025 року виділяє лідерів: Nike Training Club, Peloton, Strava.

Основний матеріал присвячено порівняльному аналізу п’яти найпопулярніших додатків 2025 р.: MyFitnessPal, Strava, Nike Training Club (NTC), Peloton та Adidas Running. Критерії порівняння наведено в таблиці 1.

Порівняння показує відсутність універсального рішення: MyFitnessPal лідирує в харчуванні, Strava – у соціальній мотивації для outdoor-активностей, NTC – у безкоштовному якісному контенті. Спільні недоліки: реклама у безкоштовних версіях, недостатня локалізація для України, проблеми з точністю GPS у приміщеннях.

Проведений аналіз підтверджує, що сучасні фітнес-додатки ефективно підвищують фізичну активність користувачів, але мають суттєві прогалини. Для проєктування нової ІС доцільно поєднати сильні сторони лідерів ринку: потужне відстеження харчування (як у MyFitnessPal), розвинену соціальну мережу та гейміфікацію (як у Strava), багатий безкоштовний контент (як у NTC) та повноцінний офлайн-режим.

Таблиця 1. Порівняльна характеристика мобільних фітнес-додатків

Критерій	MyFitnessPal	Strava	Nike Training Club	Peloton	Adidas Running
Основний фокус	Харчування, калорії	Біг/вело, GPS	Відео-тренування	Live-класи, обладнання	Біг, плани тренувань
Відстеження харчування	★★★★★	★★	★★	★★★	★★
Тренувальні програми	★★	★★★	★★★★★	★★★★★	★★★★
Соціальні функції та челенджі	★★	★★★★	★★★★★	★★★	★★★★★
Гейміфікація (бейджі, лідерборди)	★★	★★★★★	★★★★	★★★★★	★★★★
Інтеграція з носимими пристроями	★★★★	★★★★★	★★★★	★★★★★	★★★★
Офлайн-режим	Частковий	Так	Так	Обмежений	Так
Підтримка української мови	Так	Часткова	Так	Ні	Часткова
Середня оцінка (App Store/Google Play)	4.7	4.8	4.9	4.8	4.7

Додатково перспективними є інтеграція ШІ для адаптивних планів, підтримка української мови з першого дня та фокус на ментальному здоров'ї. Такий підхід дозволить створити конкурентоспроможну інформаційну систему, орієнтовану на українськомовну аудиторію.

Список використаних джерел

1. Ahn H., Park E. Motivations for user satisfaction of mobile fitness applications: An analysis of user experience based on online review comments. *Humanities and Social Sciences Communications*, 2023. Vol. 10, no. 1. URL: https://doi.org/10.1057/s41599-022-01452-6

2. F. García-Pascual et al. The use of a fitness app for customer recommendation: linear models and qualitative comparative analysis. *Humanities and Social Sciences Communications*, 2023. Vol. 10, no. 1. URL: https://doi.org/10.1057/s41599-023-02330-5

3. Liu Y., Avello M. Status of the research in fitness apps: A bibliometric analysis. *Telematics and Informatics*, 2020. P. 101506. URL: https://doi.org/10.1016/j.tele.2020.101506

4. 7 Expert-Approved Workout Apps and Services to Get You Fit. URL: https://www.cnet.com/health/fitness/best-workout-apps-and-services-7-expert-picks-to-improve-your-fitness-level/

Кравчук Станіслав, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.с.-г.н., доцент Протас Надія

ЕТАПИ АНАЛІЗУ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЄКТУВАННЯ ФРОНТЕНД- І БЕКЕНД-КОМПОНЕНТІВ ВЕБДОДАТКУ ДЛЯ ПОЗИЦІОНУВАННЯ РАСТРОВОГО ПІДПИСУ НА ЗАВАНТАЖЕНИХ ДОКУМЕНТАХ

У сучасному цифровому світі процес підписання документів дедалі частіше переходить у онлайн-середовище. Традиційний рукописний підпис, відсканований як растрове зображення (наприклад, у форматі PNG або JPEG), залишається популярним для багатьох користувачів, якіз печних причин не мають доступу до кваліфікованого електронного підпису (КЕП). Навіть зараз, багато підприємств малого та середнього бізнесу все ще використовують графічні аналоги підписів для швидкого узгодження договорів, актів та інших документів. Актуальність проблеми полягає в необхідності створення простого, безпечного та зручного вебдодатку, який дозволить користувачеві завантажити PDF-документ, завантажити або намалювати растровий підпис і точно позиціонувати його на сторінці за допомогою drag-and-drop, з подальшим збереженням модифікованого файлу. Такий додаток усуває потребу в друці, скануванні та фізичній відправці документів, зменшує час обробки і мінімізує ризики втрати оригіналів. На відміну від повноцінних систем електронного підпису (DocuSign, Adobe Sign), пропоноване рішення орієнтоване саме на графічний оверлей (накладення зображення), що не вимагає криптографічного підпису, але задовольняє потреби внутрішнього документообігу.

Аналіз літератури та інтернет-джерел показує, що реалізація накладення растрового підпису на PDF можлива як на клієнтській (client-side), так і на серверній (server-side) стороні, з перевагами та недоліками кожного підходу. Одним з найпопулярніших відкритих рішень є бібліотека pdf-lib [1], яка працює повністю в браузері та дозволяє завантажувати PDF, вбудовувати PNG/JPEG-зображення підпису на вказані координати сторінки, змінювати розмір і зберігати результат. Це забезпечує конфіденційність (документ не залишає пристрій користувача) і високу швидкість. Другим джерелом є відкрита бібліотека PDF.js від Mozilla, яка використовується для рендерингу PDF у canvas і дозволяє накладати зображення підпису через малювання по canvas з подальшим експортом [2]. Однак PDF.js не модифікує оригінальний PDF, тому часто комбінується з pdf-lib. Третім важливим джерелом є бібліотека pdfkit для Node.js, яка підходить для серверної обробки: підпис накладається на сервері, що корисно для великих файлів або додаткового захисту, але вимагає передачі документа на сервер і підвищує навантаження [3]. У дослідженні [4] виконано порівняльний огляд архітектур підписання PDF, де підкреслюється, що client-side підхід з pdf-lib переважний для простих графічних підписів через відсутність потреби в криптографії та кращу приватність, тоді як server-side (наприклад, з використанням Node.js + pdfkit) краще для інтеграції з базами даних або логуванням. Аналіз цих джерел дозволив обрати гібридний підхід з пріоритетом client-side. Основний матеріал роботи присвячено етапам аналізу

предметної області та проєктуванню архітектури додатку. Предметна область включає: завантаження PDF-документів (до 50 МБ), завантаження або створення растрового підпису (підтримка PNG з прозорістю для природного вигляду), інтерактивне позиціонування підпису на сторінках, зміна розміру, поворот і збереження результату. Аналіз вимог виявив ключові функціональні вимоги: підтримку multi-page PDF, точне позиціонування (drag-and-drop з сіткою), preview у реальному часі, збереження без втрати якості оригіналу. Нефункціональні – кросбраузерність, адаптивність під мобільні пристрої, захист від XSS (валідація файлів). Проєктування фронтенд-компонентів базується на React.js для зручного управління станом. Основні компоненти:

1. DocumentViewer – рендеринг PDF за допомогою PDF.js у <canvas>, підтримка масштабування та перемикання сторінок.

2. SignaturePanel – завантаження зображення підпису або малювання на canvas (бібліотека Signature Pad), збереження як PNG з прозорістю.

3. OverlayManager – шар над canvas для drag-and-drop підпису (використання react-draggable), збереження координат (x, y, width, height, pageIndex).

4. Toolbar – кнопки збереження, скасування, додавання кількох підписів.

Для накладення використовується pdf-lib: після позиціонування координати передаються в функцію, яка завантажує PDF як Uint8Array, embed-ить підпис як PNG і малює pdfPage.drawImage на вказаній позиції. Експорт відбувається через blob і download. Проєктування бекенд-компонентів (Node.js + Express) мінімальне, але передбачене для опціональної серверної обробки (наприклад, для дуже великих файлів): API-ендпоінти для завантаження файлів, обробки через pdfkit (генерація нового PDF з image.over). Переваги server-side – можливість додавання водяних знаків або логування, недоліки – ризик витоку даних. Архітектура RESTful, з використанням Multer для файлів і JWT для аутентифікації (якщо потрібна). Проведений аналіз предметної області показав, що для накладення растрового (відсканованого) підпису оптимальним є client-side підхід на базі pdf-lib + PDF.js, який забезпечує швидкість, конфіденційність і простоту. Запропонована архітектура фронтенду (React + draggable) та опціонального бекенду (Node.js + pdfkit) дозволяє реалізувати повнофункціональний додаток на ранніх етапах розробки. Подальші роботи включатимуть реалізацію прототипу, тестування на різних пристроях та порівняння продуктивності client vs server. Розробка такого вебдодатку матиме практичну цінність для автоматизації документообігу.

Список використаних джерел

1. pdf-lib – Create and modify PDF documents in any JavaScript environment. URL: https://pdf-lib.js.org/

2. PDF.js: A general-purpose, web standards-based platform for parsing and rendering PDFs. URL: https://mozilla.github.io/pdf.js/

3. PDFKit: A JavaScript PDF generation library for Node and the browser. URL: https://pdfkit.org/

4. Digital signature architecture: Client-side & server-side. URL: https://www.nutrient.io/guides/web/signatures/digital-signatures/architectures/

Кустовська Поліна, здобувачка вищої освіти СВО «Бакалавр»,
спеціальність «Харчові технології»,
Науковий керівник – к.т.н., доцент Одарущенко Олена

АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ВИКОРИСТАННЯ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ СОРТУВАННЯ ХАРЧОВОЇ СИРОВИНИ

У харчовій промисловості контроль якості сировини – це одна з найважливіших задач, адже дефектні плоди, зерно або інша сировина можуть суттєво знижувати ефективність виробничих процесів, підвищуючи втрати і ризики. Традиційне сортування, що базується на візуальній оцінці працівниками або на простих механічних сортувальних лініях, має суттєві обмеження: низька швидкість, суб'єктивність, висока ймовірність людської помилки.

Ці проблеми спонукають до впровадження технологій комп'ютерного зору (CV), які автоматизують аналіз зовнішнього вигляду сировини й забезпечують високу точність сортування. Однією з перспективних розробок в українських наукових колах є робота В. Штинди: «Веб-базована система розпізнавання їжі по фото засобами комп'ютерного зору». У своїй кваліфікаційній роботі він використовує CNN та метод YOLO для класифікації зображень їжі, створив веб-інтерфейс зі спільним використанням OpenCV, TensorFlow та JavaScript [1].

Така система демонструє, що комп'ютерний зір може надійно розпізнавати об'єкти харчування на зображеннях і групувати їх за категоріями, що відкриває потенціал для впровадження подібних підходів безпосередньо у виробничих лініях, наприклад, для сортування плодів або овочів за видом або якістю.

Також важлива робота Аркадія С. Яценка: «Автоматизована система оцінки якості харчових продуктів на основі комп'ютерного зору в робототехнічних системах візуального контролю».

У дипломному проєкті розглядається процес візуального контролю якості харчової продукції: система аналізує зображення харчових об'єктів на предмет дефектів, використовуючи методи обробки зображень, а також аналізує стан форми й поверхні. За допомогою цієї системи можна автоматично виявляти нерівності, пошкодження або небажані зміни у зовнішньому вигляді продукції, що підвищує оперативність контролю якості та знижує необхідність ручного огляду [2].

Ще один значущий приклад – дослідження О. К. Тесленка та А. А. Кузьмича «Засоби комп'ютерного зору визначення стану злакових культур». У цій роботі автори підкреслюють, що «застосування автоматизованої системи комп'ютерного зору знижує витрати часу на перевірку зразків і дозволяє значно зменшити залежність від суб'єктивного людського фактору, забезпечуючи об'єктивність і точність результатів аналізу» [3].

Крім того, ПЗ, яке вони розробили, може бути використане для контролю якості зерна в агропідприємствах або лабораторіях, що демонструє практичний потенціал таких систем.

З технічного боку, системи комп'ютерного зору вимагають ефективних алгоритмів класифікації зображень. У статті І.О. Кобиліна «Методи класифікації для комп'ютерного зору» розглядаються алгоритми Support Vector Machine, K-Nearest

Neighbor, Random Forest, Gradient Boosting. Автор показує, що різні алгоритми дають різну точність, і підкреслює, що правильний вибір моделі є критичним для підвищення ефективності систем машинного зору [4].

Крім того, у статті «Реалізація систем контролю якості продукції на основі машинного зору та web-технологій» Д. Ковалюк, О. Ковалюк та В. Малішевський надають схему програмної архітектури системи: вони описують клієнтську частину на Vue.js, сервер на Flask, і показують, як дані з камер надходять на сервер, аналізуються алгоритмами машинного зору, а результати відображаються в вебінтерфейсі.

Це дає можливість операторам бачити виявлені дефекти в реальному часі й приймати рішення про подальше сортування [5].

Таким чином, впровадження CV у сортування харчової сировини є не лише теоретично перспективним напрямком, але має реальні практичні приклади в українських закладах освіти та досліджень. Подібні системи показують високу точність, об’єктивність та оперативність сортування. Використання таких технологій дозволяє підвищити якість продукції, зменшити втрати, знизити людський фактор і прискорити виробничі процеси.

Подальший розвиток цих систем - через покращення алгоритмів класифікації, інтеграцію з робототехнічними лініями й впровадження веб-інтерфейсів – може зробити їх стандартом у харчовій промисловості.

Список використаних джерел

1. Штинда В. В. Веб-базована система розпізнавання їжі по фото засобами комп’ютерного зору. Тернопіль: ЗУНУ, 2024. 47 с.

2. Яценко А. С. Автоматизована система оцінки якості харчових продуктів на основі комп’ютерного зору в робототехнічних системах візуального контролю: Київ: *КПІ ім. Ігоря Сікорського*, 2025. 59 с.

3. Тесленко О. К., Кузьмич А. А. Засоби комп’ютерного зору визначення стану злакових культур. XVII наук. конф. магістрантів та аспірантів: *Прикладна математика та комп’ютинг* (м. Київ, 2024).

4. Кобилін І. О. Методи класифікації для комп’ютерного зору. *Системи обробки інформації*, 2024. № 3.

5. Ковалюк Д. О., Ковалюк О. О., Малішевський В. С. Реалізація систем контролю якості продукції на основі машинного зору та web-технологій. *Вісник НТУУ «КПІ ім. Ігоря Сікорського». Серія: Хімічна інженерія, екологія та ресурсозбереження*, 2024. № 1. С. 28-34.

Левченко Юрій, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Флегантов Леонід

АРХІТЕКТУРНІ ОСОБЛИВОСТІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ У КОНТЕКСТІ ПАРСИНГУ ДАНИХ

У сучасному цифровому середовищі обсяги даних стрімко зростають, що зумовлює потребу в ефективних інструментах їх обробки, структурування та аналізу. Парсинг даних є однією з головних інформаційних технологій, що забезпечує автоматизоване вилучення структурованої інформації з різноманітних джерел – вебсайтів, документів, баз даних, API та інших форматів. Традиційні методи парсингу, засновані на використанні регулярних виразів, XPath, CSS-селекторів або спеціалізованих бібліотек, є ефективними для простих завдань, проте мають обмеження при роботі з слабо структурованими даними або даними, що динамічно змінюються. Із розвитком штучного інтелекту, зокрема з появою великих мовних моделей (Large Language Models, LLM), таких як GPT, LLaMA, Claude, Mistral тощо, з'явилась можливість застосовувати нові підходи до парсингу даних, що ґрунтуються на глибокому розумінні природної мови: сучасні LLM здатні не лише вилучати та структурувати потрібні дані, але й інтерпретувати їх контекст, структуру та семантичні зв'язки, що відкриває принципово новий рівень ефективності у завданнях інформаційного пошуку, аналітики, моніторингу контенту та автоматизації бізнес-процесів. Усі передові LLM швидко еволюціонують у напрямі підвищення контекстуальної точності, масштабованості та універсальності. Для завдань парсингу даних важливо розуміти, як різні архітектурні підходи впливають на здатність моделі аналізувати текстову структуру, виявляти патерни і перетворювати неструктуровану інформацію у структуровану форму. В основі всіх LLM лежить архітектура трансформера, яка використовує механізм самоуваги (self-attention) для ефективного моделювання контексту [1]. Модель BERT (Bidirectional Encoder Representations from Transformers), розроблена у Google AI є двонаправленим енкодером трансформера, що дозволяє враховувати контекст слова як зліва, так і справа (рис. 1).

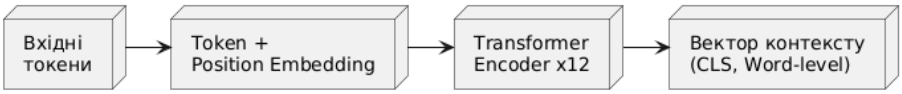

Рисунок 1 – Архітектура моделі BERT (загальна схема)

Основна ідея BERT – Masked Language Modeling (MLM), тобто під час навчання певний відсоток токенів у реченні маскується, і модель навчається передбачати ці слова, використовуючи весь контекст. Завдяки цьому BERT виявляється дуже ефективним інструментом для розуміння синтаксису та семантики, що є важливим у процесі семантичного парсингу [2]. Головними особливостями моделі BERT є такі: архітектура складається лише з енкодерів трансформера; використовується позиційне кодування для збереження порядку токенів; завдяки двонаправленості, модель формує повний контекст для кожного слова; модель добре

підходить для класифікації текстів, вилучення іменованих сутностей (Named Entity Recognition, NER) та синтаксичного аналізу, що може ефективно використовуватись для попереднього етапу парсингу даних. В архітектурі моделі BERT важливо, що її Embedding Layer складається з 3-ох частин (Token, Segment, Position Embedding). Архітектура моделі BERT з деталізованим входом представлена на рис. 2. Модель GPT, представлена компанією OpenAI [3], базується на декодерній архітектурі трансформера. На відміну від моделі BERT, GPT є авторегресивною моделлю, тобто прогнозує наступне слово, спираючись на попередні. Така архітектура найкраще підходить для генеративних завдань, включно з автоматичним створенням парсерів, побудовою структурованих шаблонів, SQL-запитів, JSON-виходів тощо [4]. Архітектура моделі GPT представлена на рис. 3.

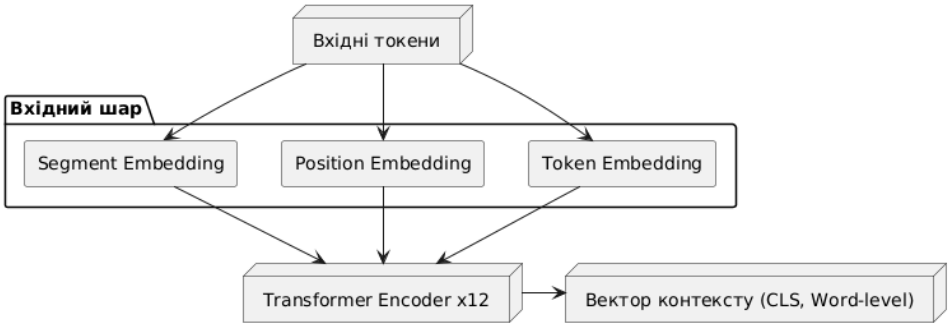

Рисунок 2 – Архітектура моделі BERT (деталізований вхід)

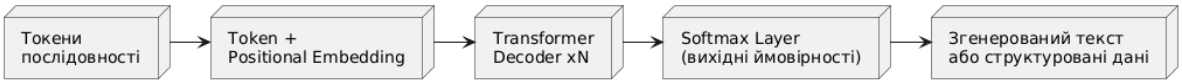

Рисунок 3 – Архітектура моделі GPT

Головні особливості моделі GPT: архітектура моделі складається лише з декодерів трансформера; навчання проводиться у каузальному режимі (Causal LM) – передбачення наступного токена за контекстом; підтримує few-shot/zero-shot навчання, що спрощує адаптацію під конкретні типи даних; у новіших версіях (GPT-5) реалізовано механізми довгої пам’яті і розширене вікно контексту, що дозволяє аналізувати великі текстові документи. Модель T5 (Text-to-Text Transfer Transformer), розроблена у Google Research [5], використовує універсальний підхід «text-to-text»: будь-яке завдання (класифікація, переклад, парсинг) формулюється як перетворення одного тексту в інший. Завдяки цьому модель T5 є дуже гнучкою для семантичного парсингу, де завдання може бути описане як «перетвори текст у структуру даних». Схема архітектури моделі T5 представлена на рис. 4. Особливості моделі T5: повна encoder-decoder архітектура; єдина уніфікована парадигма «input text → output text»; можливість виконання завдань типу «extract entities», «generate JSON», «summarize data»; у контексті Data Engineering може використовуватись для автоматичного створення парсерів для різних форматів (XML, HTML, API-відповідей). Модель LLaMA (Large Language Model Meta AI), розроблена Meta AI [6], оптимізована для ефективності та роботи на обмежених ресурсах. Архітектурно LLaMA є декодерною моделлю, подібною до GPT, але з низкою покращень: використання RoPE (Rotary Positional Embeddings) замість класичного позиційного

кодування; оптимізована структура уваги для зменшення енергоспоживання; підтримка великого контекстного вікна (до 128 тис. токенів у LLaMA 3); підтримка інструкційного навчання (instruction tuning), що дозволяє ефективно виконувати завдання парсингу без додаткового донавчання [7]. Схема архітектури LLaMA представлена на рис. 5.

Рисунок 4 – Архітектура моделі T5

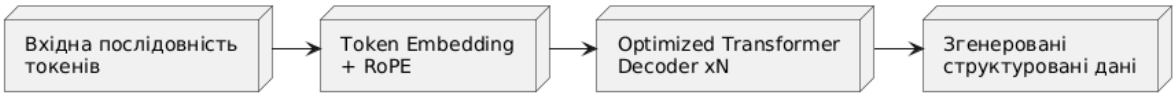

Рисунок 5 – Архітектура моделі LLaMA

Серед розглянутих архітектур, моделі GPT і LLaMA є найбільш перспективними для автоматизованого парсингу даних завдяки їх здатності генерувати структуровані формати (JSON, XML, SQL). Моделі BERT і T5, у свою чергу, залишаються ефективними для попереднього семантичного аналізу, виділення сутностей і перетворення текстових даних на логічні структури, що формує базу для подальшої генерації структурованих результатів.

Список використаних джерел

1. Vaswani, A., Shazeer, N., Parmar, N., et al. Attention is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 2017. Vol. 30. P. 5998–6008. DOI: 10.48550/arXiv.1706.03762

2. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT, 2019. P. 4171–4186. DOI: 10.48550/arXiv.1810.04805

3. Brown, T. B., Mann, B., Ryder, N., et al. Language Models are Few-Shot Learners (GPT-3). Advances in Neural Information Processing Systems, 2020. Vol. 33. P. 1877–1901. DOI: 10.48550/arXiv.2005.14165.

4. OpenAI. GPT-4 Technical Report. OpenAI, 2023. 98 p. DOI: 10.48550/arXiv.2303.08774.

5. Raffel, C., Shazeer, N., Roberts, A., et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5). Journal of Machine Learning Research, 2020. Vol. 21, No. 140. P. 1–67. DOI: 10.48550/arXiv.1910.10683.

6. Touvron, H., Lavril, T., Izacard, G., et al. LLaMA: Open and Efficient Foundation Language Models. Meta AI Research, 2023. 44 p. DOI: 10.48550/arXiv.2302.13971.

7. Touvron, H., Martin, L., Stone, K., et al. LLaMA 2: Open Foundation and Fine-Tuned Chat Models. Meta AI Research Report, 2024. 56 p. DOI: 10.48550/arXiv.2307.09288.

Майборода Віталіна, здобувачка вищої освіти СВО «Бакалавр»,
спеціальності 126 Інформаційні системи та технології,
Науковий керівник: к.т.н., доцент Одарущенко Олена

ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ПРОГНОЗУВАННЯ ВРОЖАЙНОСТІ

Розвиток сучасних інформаційних технологій відкриває нові можливості для автоматизації управління у сільському господарстві. Одним із ключових напрямів цифрової трансформації агросфери є використання систем штучного інтелекту, зокрема нейронних мереж, для аналізу великих даних і прогнозування врожайності культур. Такі технології враховують одночасно численні кліматичні, ґрунтові та агротехнічні фактори, що підвищує ефективність управлінських рішень.

FCM (Fuzzy Cognitive Map) являють собою графічну модель, яка складається з вузлів-концептів, що з'єднані ребрами. Вузли-концепти описують елементи системи, а ребра виражають зв'язки між цими концептами. FCM можна застосовувати для точного землеробства, моделювання та прогнозування врожайності. Розроблена модель FCM складається з вузлів, які є основними концептами, що впливають на врожайність – вміст калію, фосфору, pH, азоту, NDVI, LAI тощо [1].

Прогнозування врожайності є складним завданням через взаємодію багатьох змінних. На відміну від класичних статистичних методів, нейронні мережі здатні навчатися на великих масивах даних і виявляти нелінійні закономірності між показниками, такими як опади, температура, вологість ґрунту, рівень поживних речовин і технології обробітку землі.

Здійснено аналіз ефективності статистичної обробки даних супутникового моніторингу та врожайності пшениці озимої на регіональному рівні методами лінійної регресії та штучних нейронних мереж. Встановлено, що застосування нейронних мереж із двома типами нейронів (лінійним і нелінійним) істотно поліпшувало точність і надійність прогнозування врожаїв зерна пшениці озимої при використанні даних NDVI, що підтверджують коефіцієнти детермінації та середня абсолютна похибка. Абсолютної переваги нейронних мереж над лінійною регресією для одержання надійних прогнозів регіонального рівня не доведено, хоча вони забезпечують дещо вищий рівень точності в конкретних випадках [2].

Технічний аспект застосування нейронних мереж полягає у формуванні набору даних для навчання моделей. Вхідні дані включають метеорологічні показники (температура, опади, тривалість сонячного дня), фізико-хімічні характеристики ґрунту (рівень pH, вміст гумусу, азоту, фосфору, калію), а також історичні дані врожайності за кілька років. Перед подачею на нейронну мережу ці дані проходять стадії попередньої обробки: очищення від пропусків, нормалізацію та масштабування. Це дозволяє моделі навчитися правильно реагувати на різні комбінації вхідних параметрів і прогнозувати результати з високою точністю.

Моделі нейронних мереж складаються з вхідного шару, одного або кількох прихованих шарів та вихідного шару, де кожен нейрон має вагові коефіцієнти. Під час навчання ці коефіцієнти коригуються алгоритмом зворотного розповсюдження помилки (Backpropagation) із використанням оптимізаторів типу Adam або SGD.

Найчастіше для прогнозування врожайності застосовують багатошарові перцептрони (MLP) для усереднених даних, рекурентні нейронні мережі (RNN) для часових рядів і LSTM (Long Short-Term Memory) для збереження контексту сезонних змін.

Для інтеграції нейронних мереж із реальним моніторингом широко застосовуються технології Інтернету речей Internet of Things (IoT). IoT – концепція, яка передбачає об'єднання різних фізичних пристроїв через інтернет, дозволяючи їм взаємодіяти та обмінюватися даними. Суть ідеї в тому, щоб зробити об'єкти «розумними» й пов'язаними між собою, створюючи єдиний інтегрований простір для збирання та оброблення інформації, що є актуальним для поліпшення якості життя, оптимізації процесів і підвищення ефективності. Таким чином, інтернет речей – це пристрої, об'єднані в одну бездротову мережу для вирішення конкретного завдання, що активно застосовується у сільському господарстві [3].

Сучасні платформи дозволяють поєднувати нейронні мережі з IoT і супутниковим моніторингом. Сенсори у полі постійно передають дані про температуру ґрунту, рівень вологості, швидкість вітру та інсоляцію. Ці дані автоматично обробляються моделлю, що дозволяє оперативно оновлювати прогнози врожайності. Крім того, супутникові знімки та NDVI-індекси дозволяють оцінювати стан рослинності і коригувати прогнози на основі реальних умов поля.

Впровадження таких систем потребує комплексного підходу та цифрової трансформації. Досліджено системні аспекти цифрової трансформації аграрного сектору. Проаналізовано напрями впровадження сучасних рішень – від сенсорних мереж і платформ збору даних до систем автоматизації, аналітики та управління аграрним виробництвом. Запропоновано авторське визначення поняття «інноваційні цифрові технології в аграрному секторі», що підкреслює їх інтегративний, адаптивний і системний характер [4].

Крім того, нейронні мережі можуть поєднуватися з біофізичними моделями росту культур (DSSAT, APSIM). Такий гібридний підхід дозволяє враховувати як фізіологічні процеси росту рослин, так і статистичні закономірності в даних, що підвищує точність прогнозів до 97 %. Це особливо корисно для пшениці, кукурудзи, сої, соняшнику, ріпаку та інших культур.

Практична користь впровадження нейронних мереж у прогнозуванні врожайності очевидна: агропідприємства можуть завчасно планувати використання ресурсів, оптимізувати кількість добрив, прогнозувати прибутки та зменшувати ризики. Крім того, такі технології допомагають адаптувати виробництво до кліматичних змін і змін ринку, підвищуючи рентабельність і ефективність агровиробництва.

Сучасні хмарні платформи (Google Cloud, AWS, Microsoft Azure) та бібліотеки машинного навчання (TensorFlow, Keras, PyTorch) дозволяють обробляти великі масиви даних, створювати адаптивні моделі і постійно підвищувати точність прогнозів. У перспективі розвиток «розумного землеробства» передбачає створення інтегрованих аналітичних платформ, які об'єднуватимуть IoT, супутникову аналітику та штучний інтелект у єдину систему для оптимального управління агровиробництвом.

Таким чином, застосування нейронних мереж у прогнозуванні врожайності поєднує аналітику великих даних, автоматичне навчання та реальний моніторинг стану культур, забезпечуючи ефективне і прогнозоване управління агропідприємствами.

Список використаних джерел

1. Попов М.О., Іваненко С.В., Ковальчук А.Д. Метод прогнозування врожайності кукурудзи на зерно з використанням нечітких когнітивних карт. Український журнал дистанційного зондування Землі, 2024. URL: https://ujrs.org.ua/ujrs/article/download/261/272/1032?utm_source=chatgpt.com (дата звернення: 11.11.2025).

2. Лиховид П.В., Лавренко С.О., Лавренко Н.М. Ефективність методів статистичного аналізу даних супутникового моніторингу врожайності пшениці озимої. Таврійський науковий вісник, 2020. № 11. С. 45–53. URL: https://www.tnv-agro.ksauniv.ks.ua/archives/113_2020/11.pdf?utm_source=chatgpt.com (дата звернення: 11.11.2025).

3. WeAgro. Internet of Things (IoT): що це та його використання у сільському господарстві. WeAgro, 2024. URL: https://weagro.ua/blog/internet-rechej-iot-shho-cze-ta-jogo-vykorystannya-v-silskomu-gospodarstvi/?utm_source=chatgpt.com (дата звернення: 11.11.2025).

4. Колісніченко В.В. Системний підхід до цифрової трансформації аграрного виробництва: визначення та структурно-функціональна класифікація. Економіка та суспільство, 2025. URL: https://economyandsociety.in.ua/index.php/journal/article/download/6483/6423/?utm_source=chatgpt.com (дата звернення: 11.11.2025).

Масич Андрій, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.ф.-м.н., доцент Флегантов Леонід

ЗАСТОСУВАННЯ REASONING-LLM У РІЗНИХ ГАЛУЗЯХ

Сучасні інформаційні технології стрімко розвиваються, забезпечуючи впровадження передових рішень у різні сфери людської діяльності. Одним із найбільш перспективних напрямів є застосування великих мовних моделей (LLM – Large Language Models), зокрема нового покоління – Reasoning-LLM, здатних до виконання складних логічних операцій. Актуальність їх дослідження зумовлена потребою в аналітичних системах, що не лише обробляють текстову інформацію, а й демонструють здатність до дедуктивного мислення, обґрунтування висновків і побудови причинно-наслідкових зв’язків.

Метою цього дослідження є аналіз прикладного використання Reasoning-LLM у різних галузях діяльності, визначення їх переваг над класичними LLM, а також виявлення перспектив упровадження таких моделей у сучасних умовах цифрової трансформації.

Reasoning-LLM (Reasoning Large Language Models) – це клас великих мовних моделей, спеціально розроблених для виконання завдань, що потребують складного логічного міркування, багатокрокового аналізу та побудови причинно-наслідкових зв’язків. На відміну від стандартних LLM, які здебільшого генерують текст на основі статистичних закономірностей, Reasoning-LLM моделюють процес послідовного логічного виведення, подібного до людського мислення [1]. Вони поєднують мовну компетентність із когнітивними принципами – аналізом, синтезом, порівнянням, класифікацією та узагальненням.

Reasoning-LLM становлять окремий клас моделей штучного інтелекту, які інтегрують функціональність традиційних мовних систем із можливостями формальної логіки та індуктивного аналізу. Їх відмінною рисою є здатність до глибокого контекстуального розуміння, багаторівневої обробки даних і формування логічно обґрунтованих рішень у складних сценаріях. Такі моделі демонструють високу точність під час розв’язання задач, що вимагають не лише мовної обробки, а й когнітивної інтерпретації інформації [2].

На сьогодні, основними сферами застосування Reasoning-LLM є медицина, право, освіта, бізнес-аналітика, проєктний менеджмент, технічне проєктування, державне управління, наукові дослідження, а також сфера кібербезпеки. Кожна з цих галузей має специфічні потреби, для задоволення яких традиційні LLM часто виявляються недостатніми [3].

Провідні сфери застосування Reasoning-LLM представлені на рис. 1. У медицині Reasoning-LLM застосовуються для аналізу електронних медичних записів, інтерпретації результатів обстежень і формування попередніх діагнозів на основі анамнезу. Завдяки здатності до логічного міркування моделі можуть зіставляти клінічні симптоми з діагностичними шаблонами, формувати гіпотези й обґрунтовувати варіанти лікування. Вони сприяють підвищенню якості клінічного мислення та зменшенню ймовірності діагностичних помилок у складних випадках.

Наприклад, експериментальні системи на базі reasoning-моделей уже використовуються для аналізу онкологічних даних і формування індивідуальних терапевтичних планів [4]. Застосування Reasoning-LLM у медицині представлені на рис. 2.

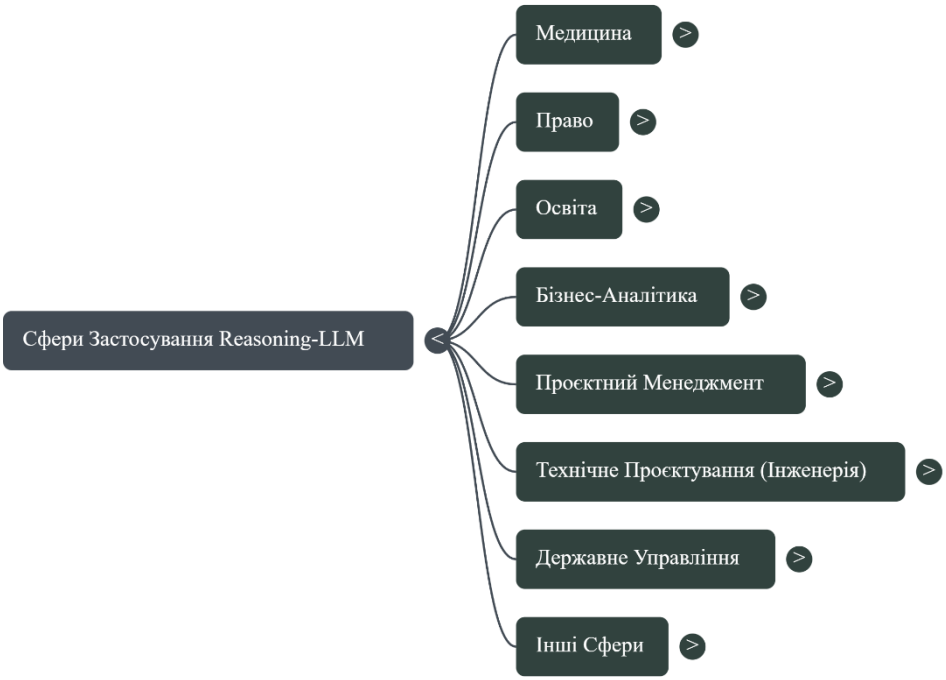

Рисунок 1 – Застосування Reasoning-LLM у різних галузях

Рисунок 2 – Приклади застосування Reasoning-LLM у медицині

У юридичній практиці ці моделі здатні ефективно обробляти нормативну документацію, формувати юридичні висновки, виявляти суперечності в контрактах та позовах, а також автоматизувати складання правових документів. Вони можуть пояснювати логіку ухвалених рішень і перевіряти їх відповідність чинному законодавству. Reasoning-LLM дозволяють створювати інтелектуальні правові системи, здатні аналізувати великі обсяги прецедентів і підтримувати юристів у прийнятті рішень на етапах підготовки судових матеріалів (рис. 3). В освітній сфері Reasoning-LLM використовуються для створення адаптивного навчального контенту, генерації логічних задач, а також для оцінювання студентських відповідей з урахуванням аргументації та ходу мислення. Це дозволяє формувати індивідуальні траєкторії навчання, де модель не лише перевіряє правильність відповіді, а й

аналізує, як студент мислив під час розв'язання (рис. 4). Такий підхід сприяє розвитку критичного мислення та формуванню навичок обґрунтованого судження [5]. У бізнес-аналітиці Reasoning-LLM застосовуються для побудови складних прогнозних моделей, аналізу зв'язків між бізнес-показниками та прийняття стратегічних рішень на основі виявлених закономірностей. Завдяки поєднанню статистичного аналізу й логічного висновку моделі можуть виявляти приховані ризики, аналізувати ринкові тренди, а також прогнозувати наслідки управлінських рішень (рис. 5).

Рисунок 3 – Приклади застосування Reasoning-LLM у правознавстві

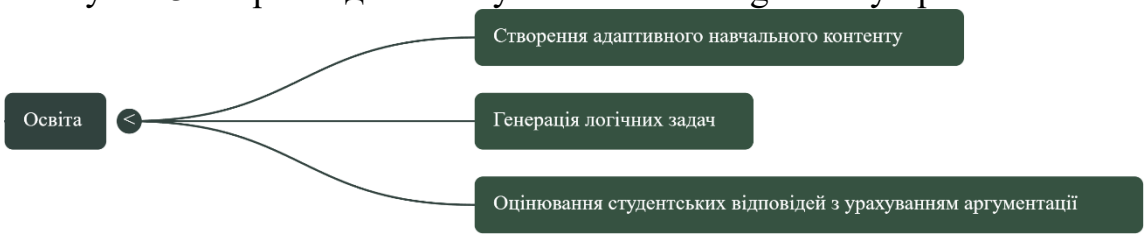

Рисунок 4 – Приклади застосування Reasoning-LLM в освіті

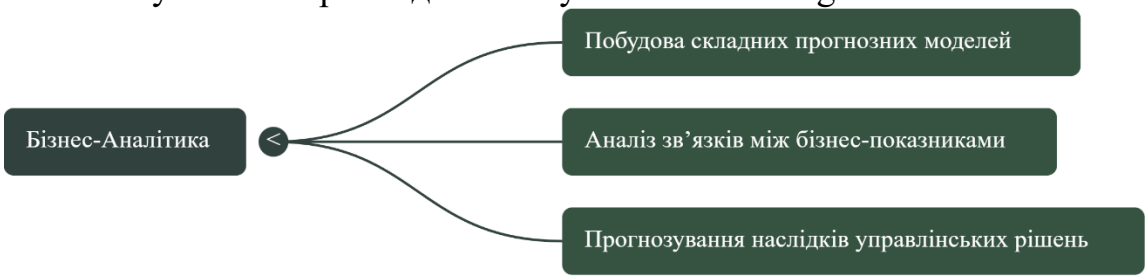

Рисунок 5 – Приклади застосування Reasoning-LLM у бізнес-аналітиці

У сфері проєктного менеджменту Reasoning-LLM забезпечують оцінювання ризиків, оптимізацію розподілу ресурсів, формування раціональних графіків виконання завдань і виявлення потенційних конфліктів ще на етапі планування (рис. 6). Завдяки здатності до аналітичного мислення такі моделі можуть виступати «віртуальними асистентами керівника проекту», що моделюють сценарії розвитку подій і пропонують обґрунтовані альтернативи [6]. У технічній сфері Reasoning-LLM здатні перевіряти узгодженість технічних параметрів, відповідність матеріалів будівельним нормам, виявляти помилки у схемах і частково автоматизувати проєктування, особливо у взаємодії з CAD-системами (рис. 7). Вони також застосовуються у розробці програмного забезпечення – наприклад, для автоматичного виправлення логічних помилок у коді або генерації тестових сценаріїв на основі вимог користувача.

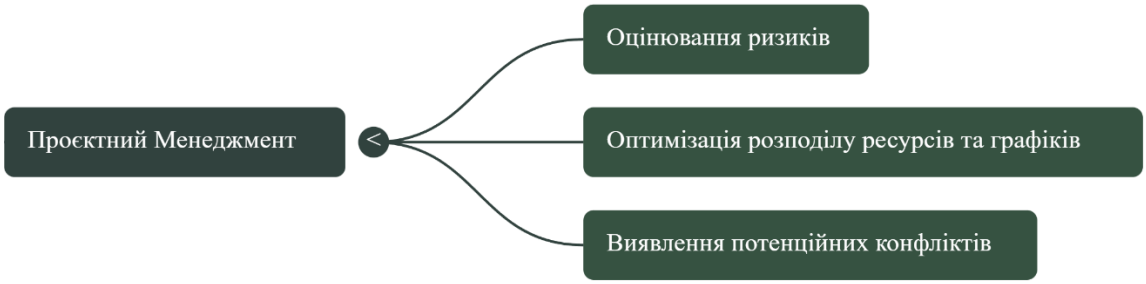

Рисунок 6 – Застосування Reasoning-LLM у проєктному менеджменті

Рисунок 7 – Застосування Reasoning-LLM у технічній сфері

У державному управлінні та адміністративному праві моделі демонструють високу ефективність у виявленні правових колізій, перевірці відповідності проєктів рішень законодавчим нормам, а також у створенні нормативних актів із поясненням логіки формулювань (рис. 8). Вони здатні допомагати державним аналітикам у підготовці звітів, прогнозуванні наслідків політичних рішень і моделюванні варіантів реформування систем управління [7]. Крім того, Reasoning-LLM поступово інтегруються у науково-дослідну діяльність (НДД), де вони використовуються для формулювання гіпотез, пошуку міждисциплінарних зв’язків і аналізу великих наукових баз даних. У сфері кібербезпеки такі моделі здатні прогнозувати потенційні вектори атак, аналізувати поведінку користувачів і виявляти аномалії в інформаційних системах, використовуючи логічний аналіз, а не лише статистичні сигнатури (рис. 9). Типові задачі, що виконують LLM у різних галузях та точність результатів їх виконання стандартними LLM та Reasoning-LLM наведено у табл. 1 [4] та представлено на рисунку 10.

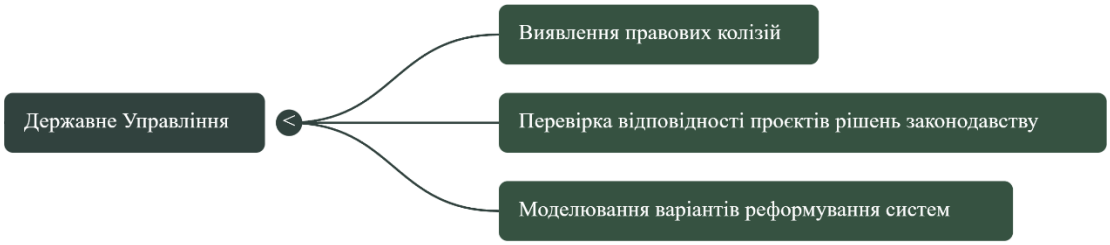

Рисунок 8 – Застосування Reasoning-LLM у державному управлінні

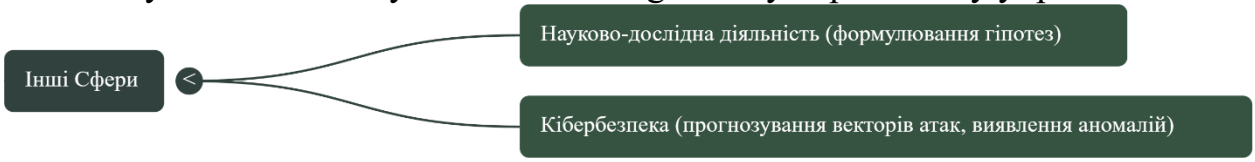

Рисунок 9 – Застосування Reasoning-LLM у НДД і кібербезпеці

Таблиця 1 – Порівняння результатів виконання завдань з різних галузей стандартною LLM та Reasoning-LLM

Галузь	Тип задачі	Стандартна LLM (точність, %)	Reasoning-LLM (точність, %)
Медицина	Визначення діагнозу за симптомами	72%	89%
Право	Побудова юридичного висновку	68%	85%
Освіта	Генерація логічних задач	75%	91%
Бізнес	Прогнозування ризиків	70%	88%

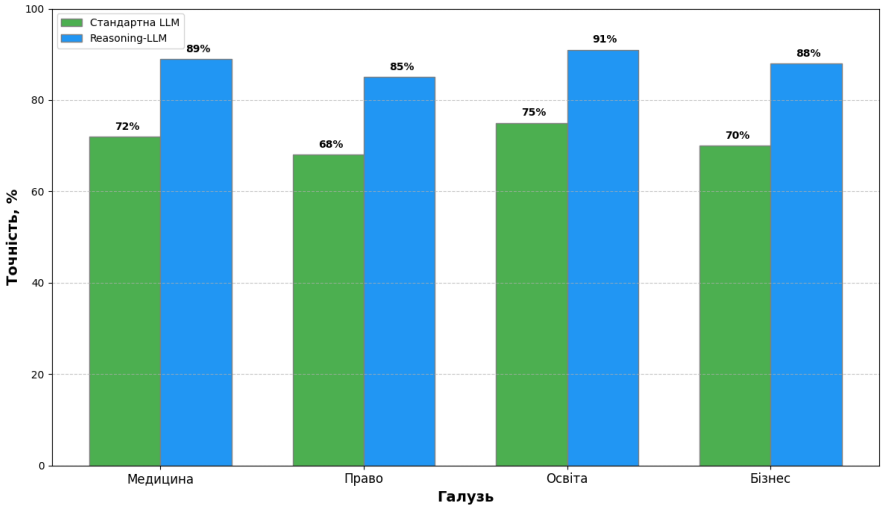

Рисунок 10 – Порівняння точності завдань із різних галузей стандартною LLM та Reasoning-LLM

Отже, переваги Reasoning-LLM над стандартними (класичними) LLM зумовлені здатністю поєднувати мовні алгоритми з механізмами логічного мислення. Це забезпечує якісно новий рівень аналітики, що дозволяє застосовувати моделі у галузях, де традиційні системи штучного інтелекту були обмежені. Дослідження показують, що Reasoning-LLM значно перевершують класичні моделі за точністю, аргументованістю та стабільністю результатів у завданнях з високою когнітивною складністю [5].

Таким чином, Reasoning-LLM є інноваційним інструментом, що розширює функціональні можливості штучного інтелекту від генерації тексту до повноцінного логічного аналізу. Їх впровадження у практику різних галузей – медицини, права, освіти, інженерії, бізнесу та державного управління – сприяє підвищенню точності, ефективності та аргументованості прийняття рішень. У перспективі такі моделі можуть стати основою нової парадигми цифрових платформ, орієнтованих на логіку, обґрунтованість і прозорість процесів ухвалення рішень [6, 7, 8].

Список використаних джерел

1. Reasoning language model. Wikipedia: вебсайт. URL: https://en.wikipedia.org/wiki/Reasoning_language_model (дата звернення: 09.04.2025)

2. Shuofei Qiao. Reasoning with Language Model Prompting: A Survey. Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, Huajun Chen. arXiv: вебсайт. DOI: https://doi.org/10.48550/arXiv.2212.09597

3. Rui Yang. CaseGPT: a case reasoning framework based on language models and retrieval-augmented generation. arXiv: вебсайт. DOI: https://doi.org/10.48550/arXiv.2407.07913

4. Wei J., Wang X., Schuurmans D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv: вебсайт*. DOI: https://doi.org/10.48550/arXiv.2201.11903

5. Hyung Won Chung et al. Scaling Instruction-Finetuned Language Models. *arXiv: вебсайт*. DOI: https://doi.org/10.48550/arXiv.2210.11416

6. Build a Large Language Model (from Scratch). *Manning Publications Co*. LLC, 2024.

7. Bergeret O., Farvault J. Gen AIon AWS: A Practicalb Approachto Building Generative AI Applicationson AWS. *Wiley & Sons*, Incorporated, John, 2024.

8. Масич А. Досвід побудови захищеної інфораструктури GPON та аналіз технологій Reasoning-LLM. *Матеріали конференції за підсумками виробничої практики здобувачів вищої освіти кафедри інформаційних систем та технологій*. ПДАУ, 2025.

Мітлицький Назар, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

РОЗРОБКА ЧАТ-БОТУ ДЛЯ ПІДТРИМКИ МАРКЕТПЛЕЙСУ

Актуальність розробки інтелектуального чат-боту для маркетплейсу зумовлена стрімким зростанням електронної комерції та необхідністю надання цілодобової, масштабованої та ефективної підтримки користувачів. Сучасний чат-бот дозволяє автоматизувати рутинні операції: відповіді на часті запитання (FAQ), відстеження статусу замовлень, повернення товарів, персональні рекомендації, що значно знижує навантаження на кол-центр та підвищує задоволеність клієнтів. Вибір оптимального технологічного стеку для реалізації є критично важливим етапом, що визначає функціональність, інтегрованість, масштабованість та вартість рішення. Особливо важливим є підтримка української мови та інтеграція з локальними платформами, такими як Prom.ua чи OLX. На тлі економічних викликів чат-боти допомагають скоротити витрати на підтримку, забезпечуючи 24/7 доступність.

Розробка такого чат-боту вимагає вибору оптимальної технології, яка забезпечить природність діалогів, офлайн-можливості (для мобільних додатків), низьке споживання ресурсів та якісну інтеграцію з API маркетплейсів. Початковий етап проєкту передбачає аналіз і порівняння існуючих рішень, щоб обрати технологічний стек для подальшого проєктування архітектури.

Метою даного дослідження є аналіз існуючих рішень та технологій для створення чат-бота підтримки маркетплейсу, визначення їх переваг і недоліків, а також вибір оптимального технологічного підходу для подальшої реалізації.

Аналіз літератури та джерел показує швидкий розвиток чат-бот технологій у 2023-2025 роках завдяки штучному інтелекту та NLP. Згідно з оглядом AI в e-commerce [1], сучасні фреймворки на базі LLM (як GPT-4) та end-to-end архітектур досягли високої ефективності в обробці запитів. У дослідженні [2] порівняння чат-бот фреймворків підкреслюється перевага відкритих рішень (Rasa, Botpress) над хмарними для кастомізації.

Для e-commerce платформ актуальними є рішення з інтеграціями, оскільки хмарні сервіси (Dialogflow, Microsoft Bot Framework) вимагають постійного з'єднання та можуть бути платними [3]. Відкритий проєкт Rasa виділяється як гнучкий інструмент для fine-tuning під українську мову. Огляд чат-бот фреймворків 2025 року відзначає Botpress, Rasa та Dialogflow як найкращі для e-commerce завдяки легкості та масштабовості.

Основний матеріал порівняння зосереджено на ключових технологіях для розробки чат-ботів, придатних для підтримки маркетплейсу. Розглянемо чотири основні категорії: хмарні сервіси Google, Microsoft, Intercom та відкриті рішення (Rasa, Botpress)

1. Google Dialogflow:Використовує глибокі нейронні мережі для розуміння намірів. Підтримка української мови - повна (з 2023 року). Переваги: висока точність (F1-score >0.9), інтеграція з Google Commerce та Messenger, SDK для Android/iOS. Недоліки для маркетплейсу: обов'язковий інтернет, платність після квоти (від $0.002

за запит), високе споживання трафіку. Латентність - 100–300 мс. Ідеально для онлайн-режиму, але не для офлайн.

2. Microsoft Azure Bot Service: Одна з найкращих за інтеграціями (з Azure AI). Підтримка української – з 2021 р., понад 5 голосів для мультимодальних ботів. Переваги: кастомні моделі, контроль діалогів (Bot Framework Composer), SDK для мобільних. Недоліки: хмарний (потрібен інтернет і ключ API), вартість ~$0.50 за 1000 повідомлень, вища латентність у пікові години. У 2025 році додано agentic AI для автономних дій.

3. IBM Watson Assistant: Neural chatbot з хорошою якістю для enterprise. Українська мова - часткова підтримка (стандартні моделі). Переваги: низька ціна ($0.0025 за запит), інтеграція з e-commerce API. Недоліки: гірша точність для укр мови порівняно з Google/Microsoft, немає повноцінного офлайн на мобільних.

4. Відкриті рішення: Rasa: End-to-end фреймворк на базі ML. Підтримка української - через fine-tuning (моделі на HuggingFace). Переваги: повністю офлайн, швидкість на CPU/GPU (до 5x real-time), кастомізація, безкоштовно. Якість - висока після донавчання. Інтеграція: Python з Flask для API маркетплейсів. Botpress: Візуальний builder для чат-ботів. Повністю офлайн, легкий (~100 МБ), працює з e-commerce без інтернету. LangChain: Для agentic AI, офлайн, швидка, хороша для інтеграцій з маркетплейсами.

Таблиця 1. Порівняння характеристик систем чат-ботів (за критеріями для маркетплейсу)

Платформа	Технологія/Deployment	NLP Підтримка	Інтеграції	Вартість
Google Dialogflow	Хмарний	Повна	Висока (Google, Shopify, тощо)	Платно
Microsoft Framework	Хмарний / On-premise	Повна	Висока (Azure, CRM, Teams)	Платно
Intercom	Хмарний	Часткова	Середня (WhatsApp, Slack, email)	Платно
Rasa	On-premise	Через fine-tuning	Висока (Custom, будь-які API)	Безкоштовно
Botpress	On-premise / Хмарний	Відмінна	Висока (HubSpot, Jira, Zapier тощо)	Безкоштовно / Платно
LangChain	On-premise / Локально	Через fine-tuning	Висока (будь-які LLM API)	Безкоштовно

Аналіз показав, що для чат-боту підтримки маркетплейсу оптимальними є відкриті рішення на базі Rasa або Botpress, оскільки вони забезпечують незалежність від хмари, низьку вартість і хорошу адаптацію до української мови з можливістю тестування на покриття станів. Хмарні сервіси (Dialogflow, Microsoft) доцільні як гібридний варіант для преміум-якості в онлайн-режимі з інтеграцією UI.

Список використаних джерел

1. The 7 best open source chatbot platforms in 2025. URL: https://www.eesel.ai/blog/open-source-chatbot-platforms

2. Papastamoulou P., Antonopoulos N. Artificial Intelligence in E-Commerce: A Comparative Analysis of Best Practices Across Leading Platforms. Systems. 2025. Т. 13, № 9. С. 746. URL: https://doi.org/10.3390/systems13090746

3. Bhardwaz S., Kumar J. An Extensive Comparative Analysis of Chatbot Technologies - ChatGPT, Google BARD and Microsoft Bing. 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), м. Salem, India, 4-6 May 2023. URL: https://doi.org/10.1109/icaaic56838.2023.10140214

Музичин Костянтин, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.с.г.н., доцент Протас Надія

КОНЦЕПТУАЛЬНЕ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ-МАГАЗИНУ HANDMADE-МАТЕРІАЛІВ ТА ТОВАРІВ ДЛЯ ТВОРЧОСТІ

У сучасному цифровому світі електронна комерція стає ключовим інструментом для розвитку малого бізнесу, особливо в сегменті handmade-товарів та матеріалів для творчості. Зростання інтересу до унікальних, екологічних та персоналізованих продуктів, таких як пряжа, фарби, інструменти для рукоділля чи готові вироби, зумовлене пандемією COVID-19 та тенденціями до самовираження через хобі. За даними аналітиків, ринок handmade-товарів продовжує зростати на 10-15 % щорічно. Однак, багато майстрів стикаються з проблемами: відсутність ефективних платформ для продажу, складність управління запасами, персоналізацією замовлень та інтеграцією з соціальними мережами.

Концептуальне проєктування інформаційної системи (ІС) інтернет-магазину дозволяє вирішити ці виклики на ранніх етапах, забезпечивши гнучкість, масштабованість та користувацьку орієнтованість. Воно включає аналіз вимог, моделювання процесів та вибір технологій, що є критичним для уникнення помилок на стадії реалізації.

В Україні, де сектор творчості активно розвивається (наприклад, через платформи як Etsy.com чи локальні ярмарки), така ІС може стимулювати економіку, підтримуючи локальних виробників. Актуальність полягає в необхідності адаптації ІС до специфіки handmade – унікальність товарів, сезонність попиту та інтеграція з креативними спільнотами, що робить проєктування не просто технічним, а й бізнес-орієнтованим завданням.

Для дослідження теми було проаналізовано низку інтернет-джерел, присвячених проєктуванню онлайн-платформ для handmade-товарів.

У роботі [1] описано крок-за-кроком процес створення marketplace, починаючи з визначення концепції та дослідження ринку. Автори наголошують на важливості аналізу конкурентів та вимог користувачів для концептуального проєктування. Друге дослідження [2] фокусується на концептуальному етапі, включаючи моделювання бізнес-процесів та інтеграцію цифрових інструментів для відображення унікальних крафтів. Воно підкреслює роль семантичного пошуку та персоналізації в ІС.

У статті «Ways For Interior Designers To Start Their Online Business» на Rural Handmade автори роблять акцент на інтер'єрний дизайн та надають універсальні рекомендації щодо backend та frontend проєктування для handmade-платформ, акцентуючи на інфраструктурі та масштабованій архітектурі [3].

Четверте джерело [4], описує концептуальне проєктування з акцентом на управління замовленнями, логістикою та інтеграцією з інструментами як Etsy, що корисно для моделювання ІС. Ці джерела демонструють спільні тенденції: фокус на користувацькому досвіді, гнучкій архітектурі та аналізі ринку, але бракує глибокого розгляду українських реалій.

Концептуальне проєктування ІС інтернет-магазину handmade-матеріалів та товарів для творчості починається з аналізу предметної області. Предметна область охоплює сегмент творчості: від матеріалів (тканини, бісер, інструменти) до готових виробів (прикраси, декор). Ключові стейкхолдери – майстри, покупці та адміністратори. Аналіз вимог включає функціональні (каталог товарів, кошик, оплата, відгуки) та нефункціональні (безпека, швидкість, мобільна адаптація).

Використовуючи методологію UML, моделюють діаграми: use case для сценаріїв (пошук товару, реєстрація майстра), class diagram для структури даних (класи «Товар», «Користувач», «Замовлення») та sequence diagram для процесів (оформлення покупки).

На етапі порівняльного аналізу технологій потрібно проаналізувати фреймворки: для frontend – React.js з Redux для динамічного інтерфейсу, що дозволяє персоналізовані рекомендації на основі AI; backend – Node.js або Django для обробки запитів, з базою даних PostgreSQL для реляційних даних (категорії, користувачі) та MongoDB для неструктурованих (фото товарів). Інтеграція з API платіжних систем (LiqPay, Stripe) та соціальними мережами (Instagram, Pinterest) забезпечує синхронізацію контенту. Архітектура – мікросервісна, з сервісами для каталогу, аутентифікації та аналітики, що підвищує масштабованість. Специфіка handmade: підтримка кастомізації (опція «зробити на замовлення»), фільтри за матеріалами/техніками та спільноти для обміну ідеями.

Проєктування інтерфейсу фокусується на UX/UI: інтуїтивна навігація з пошуком за тегами («акварель», «вишивка»), галереї фото з 360-градусним переглядом та мобільний дизайн. Безпека – HTTPS, JWT-токени для аутентифікації. Економічна модель: комісія з продажів, підписка для майстрів. Загалом, концепція забезпечує баланс між креативністю та ефективністю.

Концептуальне проєктування ІС інтернет-магазину handmade-матеріалів та товарів для творчості є фундаментом для створення конкурентоспроможної платформи, що стимулює творчий бізнес. Воно дозволяє врахувати специфіку сегменту, забезпечити гнучкість та інновації. Перспективи – інтеграція VR для віртуальних майстерень та AI для рекомендацій. Рекомендується продовжити до детального проєктування та прототипування для валідації концепції.

Список використаних джерел

1. How to Build an Online Marketplace for Handmade Crafts? URL: https://blog.artistrybazaar.com/how-to-build-an-online-marketplace-for-handmade-crafts/

2. Online Marketplace for Crafters: Step-by-Step Process to Build It. URL: https://www.aalpha.net/blog/how-to-build-online-marketplace-for-crafters/

3. Ways For Interior Designers To Start Their Online Business. URL: https://ruralhandmade.com/blog/ways-for-interior-designers-to-start-their-online-business

4. Build a Handmade Crafts and Artisan Goods Marketplace. URL: https://www.shipturtle.com/blog/create-handmade-crafts-and-artisan-goods-marketplace

Мулько Андрій, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.т.н., доцент Слюсарь Ігор

ВПЛИВ СУЧАСНИХ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ НА ЕФЕКТИВНІСТЬ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Стрімкий розвиток цифрової інфраструктури зумовлює необхідність постійного вдосконалення процесів проєктування інформаційних систем (ІС). Одним із ключових драйверів таких змін стали інтелектуальні технології, здатні автоматизувати аналітику, підтримувати прийняття рішень та знижувати витрати на початкових етапах розробки. У сучасному ІТ-середовищі вони формують нові підходи до організації роботи, дозволяючи створювати більш гнучкі й стійкі програмні рішення.

Особливу увагу привертають технології обробки природної мови, які отримали значний розвиток завдяки використанню великих моделей. Вони здатні аналізувати технічну документацію, підтримувати структуризацію вимог та формувати рекомендації для аналітиків і розробників. Хоча їх основним призначенням не є повна автоматизація проєктних процесів, такі технології суттєво підсилюють команди, прискорюючи рутинні етапи підготовки до розробки [1]. Інтелектуальні системи також відіграють важливу роль у моделюванні бізнес-процесів та оцінці різних архітектурних рішень. Завдяки можливості швидко аналізувати значні обсяги даних вони допомагають виявляти потенційні ризики, прогнозувати навантаження та обирати оптимальні конфігурації систем. Динаміку зростання використання інтелектуальних технологій у проєктуванні інформаційних систем у 2020-24 рр. наведено на рис. 1.

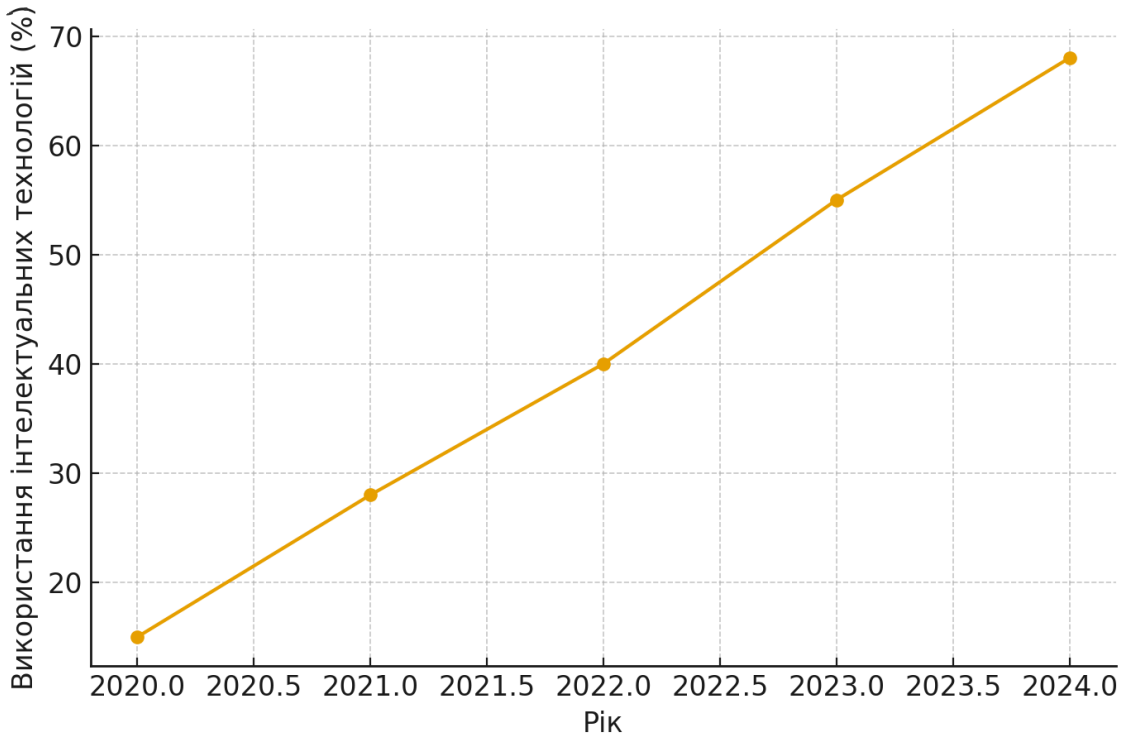

Рисунок 1 Рівень застосування AI-технологій в процесі проєктування ІС

Це створює передумови для підвищення надійності та масштабованості майбутніх ІС.

Окремий вектор розвитку визначають генеративні моделі (Generative AI), які трансформують підхід до створення артефактів проєктування. Сучасні інструменти дозволяють не лише аналізувати вимоги, а й автоматично генерувати на їх основі діаграми класів (UML), схеми баз даних та прототипи інтерфейсів. Це дозволяє скоротити час на етап «Discovery» та зменшити кількість помилок, пов'язаних із людським фактором.

Кількісна оцінка ефективності впровадження таких технологій демонструє скорочення часових витрат на рутинні операції проєктування. Інтелектуальні асистенти дозволяють архітекторам зосередитися на вирішенні нетривіальних задач, таких як забезпечення безпеки даних та оптимізація високонавантажених модулів, передаючи алгоритмам завдання з валідації синтаксису моделей та перевірки відповідності стандартам документації [2].

Разом із тим, активна інтеграція інтелектуальних технологій потребує належного контролю якості отриманих результатів. Особливо це стосується ситуацій, коли інструменти працюють із неповною, нечіткою або суперечливою інформацією.

Важливим аспектом залишається питання інформаційної безпеки при використанні хмарних інтелектуальних сервісів для проєктування корпоративних систем. Оскільки моделі навчаються на великих масивах даних, існує ризик витоку конфіденційної інформації про архітектуру проєкту [3]. Тому сучасні методики проєктування мають включати протоколи санітизації даних перед їх обробкою зовнішніми AI-інструментами або передбачати використання локальних (on-premise) моделей.

Загалом, інтелектуальні технології стають невід'ємною частиною сучасного проєктування ІС. Вони не замінюють фахівців, але значно підсилюють їх можливості, сприяючи підвищенню ефективності розробки та якості кінцевих продуктів. Подальші дослідження у цій сфері мають бути спрямовані на розробку комплексних моделей застосування таких технологій, а також на інтеграцію інструментів штучного інтелекту в стандартизовані методики проєктування.

Список використаних джерел

1. Huyen C. Designing Machine Learning Systems: An Iterative, Human – Centered Approach. O'Reilly Media, 2022. P. 386;

2. Hou X. et al. Large Language Models for Software Engineering: A Systematic Literature Review. ACM Transactions on Software Engineering and Methodology. 2024. URL: https://nzjohng.github.io/publications/papers/tosem2024_5.pdf (дата звернення: 21.11.2025).

3. Ozkaya I. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. IEEE Software. 2023. Vol. 40, no. 4. P. 4–8. URL: https://nzjohng.github.io/publications/papers/tosem2024_5.pdf (дата звернення 21.11.2025).

Насоненко Олександр, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

АНАЛІЗ ПЕРЕВАГ І НЕДОЛІКІВ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ НА ОСНОВІ СПІЛЬНОЇ (OPENMP) ТА РОЗПОДІЛЕНОЇ (MPI) ПАМ'ЯТІ

У сучасних високопродуктивних обчисленнях (High Performance Computing, HPC) паралельне програмування відіграє ключову роль у розв'язанні складних обчислювальних задач, таких як моделювання клімату, обробка великих даних, симуляція фізичних процесів та машинне навчання. Зростання обсягів даних і складності алгоритмів вимагає ефективного використання багатоядерних процесорів і кластерних систем. Традиційно виділяють два основні підходи до паралелізму: на основі спільної пам'яті (shared memory) та розподіленої пам'яті (distributed memory). Перший реалізується за допомогою стандарту OpenMP, який дозволяє легко паралелізувати код за допомогою директив компілятора, а другий – через бібліотеку MPI (Message Passing Interface), орієнтовану на обмін повідомленнями між процесами. Актуальність проблеми полягає в необхідності вибору оптимального підходу залежно від архітектури апаратного забезпечення: багатоядерні системи з спільною пам'яттю домінують у настільних і серверних комп'ютерах, тоді як суперкомп'ютери та кластери часто використовують розподілену пам'ять. Неправильний вибір може призвести до низької ефективності, надмірних витрат на комунікацію або проблем з масштабуванням. Дослідження показують, що на shared-memory системах OpenMP часто перевершує MPI за продуктивністю, тоді як MPI незамінний для великих розподілених систем.

Аналіз літератури підтверджує значний інтерес до порівняння цих підходів. У роботі [1] на основі бенчмарків NAS продемонстровано, що різні стилі програмування OpenMP (loop-level, SPMD) забезпечують конкурентну продуктивність порівняно з MPI на shared-memory мультипроцесорах, а в деяких випадках перевершують його завдяки кращому балансуванню навантаження. Інше дослідження [2] порівнює OpenMP, MPI та MapReduce і робить висновок, що OpenMP є оптимальним для задач, які вміщуються в пам'ять одного вузла, тоді як MPI краще підходить для обчислювально інтенсивних задач середнього розміру на розподілених системах. У статті [1] підкреслюється перевага гібридного підходу MPI+OpenMP для кластерів з багатоядерними вузлами, де OpenMP обробляє внутрішньовузловий паралелізм, а MPI – міжвузловий. Офіційна документація OpenMP (openmp.org [3]) та MPI (mpi-forum.org [4]) також акцентує на простоті OpenMP для shared-memory і portability MPI для heterogeneous систем.

Основний матеріал аналізу зосереджено на перевагах і недоліках кожного підходу. OpenMP використовує модель fork-join: головний потік створює команду потоків, які мають доступ до спільної пам'яті. Це спрощує програмування, оскільки не потрібно явно керувати передачею даних. Директиви типу #pragma omp parallel for дозволяють паралелізувати цикли з мінімальними змінами в коді. MPI, навпаки, базується на моделі передачі повідомлень (MPI_Send, MPI_Recv), де кожен процес має власну пам'ять, а комунікація відбувається явно.

Таблиця 1. Порівняння характеристик OpenMP (спільна пам’ять) та MPI (розподілена пам’ять)

Критерій	OpenMP (спільна пам’ять)	MPI (розподілена пам’ять)
Простота програмування	Висока: директиви, мінімальні зміни коду	Низька: явне керування повідомленнями, більше коду
Масштабованість	Обмежена одним вузлом (до десятків ядер)	Висока: тисячі вузлів у кластерах
Продуктивність на shared-memory	Висока, низькі overhead на комунікацію	Нижча через емуляцію повідомлень
Керування даними	Автоматичне, доступ до спільної пам’яті	Явне, ризики deadlock, але кращий контроль
Портативність	Залежить від компілятора (GCC, Intel)	Висока, стандарт для distributed систем
Балансування навантаження	Автоматичне (dynamic scheduling)	Ручне або через collective operations
Гібридне використання	Легко комбінується з MPI	Базовий для гібридних моделей

З таблиці 1 видно, що OpenMP перевершує MPI за простотою та продуктивністю на одному вузлі, але поступається в масштабованості. MPI вимагає більше зусиль на розробку, але дозволяє ефективно працювати з великими кластерами, де комунікація через мережу неминуча. Недоліки OpenMP – ризики race conditions через спільну пам’ять, що вимагає синхронізації (critical, barrier). У MPI основні проблеми – overhead на пакування/розпакування повідомлень і складність відладки. Аналіз показує, що вибір між OpenMP і MPI залежить від задачі та апаратного забезпечення. Для задач на одному багатоядерному вузлі (наприклад, обробка зображень чи симуляції середнього розміру) OpenMP є кращим через простоту та ефективність. Для розподілених систем (суперкомп’ютери, хмарні кластери) MPI незамінний, а гібридний підхід MPI+OpenMP часто дає оптимальні результати, поєднуючи переваги обох. Майбутні дослідження варто спрямовувати на автоматичну оптимізацію гібридних моделей для heterogeneous архітектур (CPU+GPU).

Список використаних джерел

1. Krawezik G. Performance comparison of MPI and three openMP programming styles on shared memory multiprocessors. *The fifteenth annual ACM symposium*, San Diego, California, USA, 7–9 June 2003. New York, New York, USA, 2003. URL: https://doi.org/10.1145/777412.777433.

2. Kang S.J., Lee S.Y., Lee K.M. Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems. *Advances in Multimedia*. 2015. Vol. 2015. P. 1-9. URL: https://doi.org/10.1155/2015/575687.

3. OpenMP Architecture Review Board. OpenMP Application Program Interface, Version 5.2. URL: https://www.openmp.org/spec-html/5.2/openmp.html

4. MPI Forum. MPI: A Message-Passing Interface Standard, Version 4.0. URL: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf

Небрат Мирослав, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.е.н., доцент Панасенко Наталія

ПОРІВНЯЛЬНИЙ АНАЛІЗ ФРОНТЕНД-ТЕХНОЛОГІЙ ТА ПРОЄКТУВАННЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ (НА ПРИКЛАДІ КНИЖКОВОГО ІНТЕРНЕТ-МАГАЗИНУ)

У сучасному світі електронної комерції розробка вебдодатків для інтернет-магазинів, зокрема книжкових, є актуальною проблемою через швидке зростання онлайн-продажів. За даними різних досліджень, ринок e-commerce продовжує розширюватися, а користувачі вимагають швидких, інтуїтивних та мобільно-адаптивних інтерфейсів. Книжковий інтернет-магазин потребує ефективної клієнтської частини для відображення каталогу товарів, пошуку, кошика, рекомендацій та інтеграції з платіжними системами. Вибір правильної фронтенд-технології впливає на продуктивність, масштабованість, час розробки та користувацький досвід. На початковому етапі проєктування важливо провести порівняльний аналіз популярних фреймворків, щоб обґрунтувати вибір, враховуючи специфіку e-commerce: великі обсяги даних, динамічні оновлення та SEO-оптимізацію. Аналіз джерел показує, що в 2025 р. лідерами фронтенд-розробки залишаються React, Angular, Vue.js та Svelte. Згідно з опитуванням State of JavaScript 2024, React використовують понад 80% розробників, Angular – близько 17-20%, Vue – 15-30%, а Svelte демонструє швидке зростання завдяки високій задоволеності користувачів (понад 89 %) [1]. Stack Overflow Developer Survey 2025 підтверджує домінування React у вебфреймворках, з Node.js та Express на backend, але для чистого фронтенду React лідирує з 39-40 % використання [2]. Дослідження на Medium та інших джерел (наприклад, "React vs Vue vs Svelte: Choosing the Right Framework for 2025" [3]) підкреслюють, що для e-commerce-проєктів важливими критеріями є продуктивність (розмір бандлу, швидкість рендерингу), управління станом, екосистема компонентів та легкість інтеграції з headless CMS чи API. Для книжкових магазинів, де потрібні складні фільтри, пагінація та рекомендації, фреймворки з сильним state management (наприклад, Redux для React чи NgRx для Angular) є перевагою. У межах дослідження здійснено порівняльний аналіз сучасних фронтенд-технологій, що найбільш широко застосовуються під час створення клієнтських частин веб-орієнтованих інформаційних систем. Розгляд охоплює чотири популярні фреймворки та бібліотеки: React, Angular, Vue.js та Svelte, які суттєво різняться архітектурними підходами, принципами роботи та вимогами до ресурсів. Порівняння проводилося за низкою ключових критеріїв, релевантних для систем електронної комерції: архітектурна модель (компонентний підхід, реактивність, двостороннє зв’язування даних), продуктивність, складність навчання та супроводу, екосистема та доступність інструментів, а також рівень підтримки з боку спільноти та корпорацій-розробників. React (від Meta) – найпопулярніший фреймворк завдяки гнучкості та величезній екосистемі (Next.js для SSR). Переваги: віртуальний DOM для ефективних оновлень, hooks для управління станом, велика

спільнота та бібліотеки (Material-UI, Ant Design). Розмір бандлу ~40-42 KB, продуктивність висока для динамічних UI. Недоліки: потребує додаткових бібліотек для роутингу та стану, що може ускладнити проєкт. Для e-commerce React ідеальний через готові рішення як React Query для даних. Angular (від Google) – повноцінний фреймворк з TypeScript за замовчуванням. Переваги: вбудовані інструменти (CLI, RxJS для реактивності, dependency injection, modules), відмінна масштабованість для enterprise-додатків, Signals для реактивності (новинка 2024-2025). Розмір бандлу більший (~130 KB), але оптимізований для великих проєктів. State management через NgRx або services. Angular підходить для складних книжкових магазинів з адмін-панеллю, завдяки структурованості та вбудованій підтримці форм, валідації та AOT-компіляції. Vue.js – баланс між простотою та потужністю. Переваги: інтуїтивний синтаксис, Composition API, малий розмір (~30 KB), висока продуктивність. Легко інтегрується в існуючі проєкти, Nuxt.js для SSR [4]. Недоліки: менша екосистема для enterprise. Добре для середніх e-commerce з фокусом на DX (developer experience). Svelte – компілюється в ванільний JS, без віртуального DOM. Переваги: найменший бандл (~10 KB), найкраща продуктивність та runtime-швидкість, вбудовані stores для стану. Зростає популярність (6-7% використання, але 90% satisfaction). Недоліки: менша спільнота, менше готових компонентів. Ідеальний для швидких, легких магазинів. Для книжкового інтернет-магазину обрано Angular через його переваги в проєктуванні клієнтської частини: модульна архітектура (lazy loading модулів для каталогу/кошика), вбудована реактивність (FormsModule, HttpClient), TypeScript для типобезпеки та інструменти для тестування (Karma/Jasmine). Проєктування включає: компоненти (BookCard, CatalogComponent, CartComponent), сервіси для API, роутинг з guards, state management через services/Signals. Архітектура: standalone components (з Angular 14+), матеріальний дизайн (Angular Material). Це забезпечує масштабованість, SEO (з Angular Universal) та легке розширення (рекомендації, відгуки). Проведене порівняння показало, що Angular є оптимальним для проєкту книжкового магазину на етапі проєктування завдяки комплексності, стабільності та enterprise-орієнтації. Він перевершує конкурентів у структурованості та вбудованих інструментах, хоча React лідирує в популярності. Подальша реалізація дозволить створити надійний, продуктивний додаток.

Список використаних джерел

1. State of JavaScript 2024. URL: https://2024.stateofjs.com/en-US
2. Stack Overflow Developer Survey 2025. URL: https://survey.stackoverflow.co/2025/
3. React vs Vue vs Svelte: Choosing the Right Framework for 2025. URL: https://medium.com/@ignatovich.dm/react-vs-vue-vs-svelte-choosing-the-right-framework-for-2025-4f4bb9da35b4
4. DiNeuron. Frontend Frameworks 2025: React vs Vue vs Angular vs Svelte - The Complete Developer's Guide. URL: https://dineuron.com/frontend-frameworks-2025-react-vs-vue-vs-angular-vs-svelte

Олексашенко Ярослав, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Фінанси, банківська справа, страхування та фондовий ринок»
Науковий керівник – к.с.-г.н., доцент Вакуленко Юлія

ТРАНСПОРТНА ЗАДАЧА ЯК ОСНОВА РАЦІОНАЛЬНОГО УПРАВЛІННЯ ПОТОКАМИ РЕСУРСІВ

У сучасних умовах глобальної логістики проблема оптимального розподілу ресурсів набуває особливого значення. Зростання масштабів виробництва, ускладнення мереж постачання та потреба у скороченні витрат вимагають науково-обґрунтованих методів прийняття рішень. Одним із ключових інструментів такої оптимізації є транспортна задача – математична модель, що описує процес найраціональнішого перевезення вантажів від постачальників до споживачів за мінімальних витрат або часу.

Транспортні задачі широко застосовуються в логістиці, торгівлі, енергетиці, виробництві та навіть у сфері інформаційних технологій, де оптимізуються потоки даних чи енергії. Їх вирішення дозволяє зменшити витрати на транспортування, підвищити ефективність використання ресурсів і сприяти сталому розвитку економічних систем. Саме тому транспортна задача залишається однією з найважливіших і найактуальніших моделей у сучасній теорії оптимізації. Сутність транспортної задачі полягає у визначенні таких обсягів перевезень між джерелами і пунктами призначення, щоб загальні витрати транспортування були мінімальними, а умови балансу – тобто рівність між загальним обсягом постачання і загальним попитом – були дотримані [1, с. 79]. Математично модель транспортної задачі можна подати у вигляді [2, с. 36-37; 3]:

$$Z_{min} = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij},$$

де: c_{ij} – вартість перевезення одиниці вантажу з і-го пункту відправлення до j-го пункту призначення; x_{ij} – обсяг вантажу, що перевозиться за цим маршрутом; m – кількість постачальників; n – кількість споживачів.

При цьому повинні виконуватись умови балансу:

$$\sum_{j=1}^{n} x_{ij} = a_i \ (i = 1,2,\dots,m), \quad \sum_{i=1}^{m} x_{ij} = b_j \ (j = 1,2,\dots,n), \quad \sum_{i=1}^{m} a_j = \sum_{j=1}^{n} b_j,$$

де: a_i – обсяг продукції у iii-го постачальника; b_j – потреба j-го споживача.

Таким чином, транспортна задача забезпечує раціональний розподіл ресурсів між усіма учасниками логістичного процесу, дозволяючи мінімізувати витрати та підвищити ефективність функціонування системи постачання.

Розв’язання транспортної задачі здійснюється за допомогою різних аналітичних та алгоритмічних методів, кожен із яких має свої переваги залежно від структури задачі та вимог до точності розрахунків.

Першим кроком зазвичай є побудова базисного плану перевезень. Для цього використовують, зокрема, спосіб північно-західного кута – простий алгоритм, що послідовно розподіляє ресурси, починаючи з верхнього лівого елемента таблиці.

Інший підхід – спосіб мінімальної вартості, який одразу враховує економічний аспект, заповнюючи комірки з найменшими витратами перевезення.

Після отримання базисного плану проводиться його оптимізація. Найпоширенішим інструментом є метод потенціалів, який дозволяє визначити, чи досягнуто оптимального розв'язку, та покроково зменшити загальні транспортні витрати. Подібним за ідеєю є дельта-метод, що базується на розрахунку зміни витрат при можливих перерозподілах вантажів.

У більш загальному випадку транспортну задачу можна розв'язувати за допомогою симплекс-методу зі штучним базисом, який є універсальним підходом лінійного програмування. Він особливо дієвий, якщо задача має додаткові обмеження або нестандартну структуру.

Серед сучасних модифікацій варто відзначити модифікований метод (метод MODI) – це вдосконалений варіант методу потенціалів, який підвищує швидкість пошуку оптимального рішення.

Окреме місце займає угорський метод, розроблений для розв'язання задач призначення, що є окремим випадком транспортної задачі. Його суть полягає у мінімізації загальних витрат на призначення «виконавець–завдання» шляхом послідовного зменшення матриці витрат і пошуку нульових елементів, які формують оптимальні пари. Завдяки своїй ефективності та простоті реалізації угорський метод часто застосовується в задачах розподілу ресурсів, плануванні персоналу та оптимізації маршрутів.

Таким чином, різноманіття методів розв'язання транспортних задач забезпечує гнучкість у виборі підходу – від простих евристичних алгоритмів до строгих оптимізаційних процедур, що дозволяють досягти мінімальних витрат у складних логістичних системах.

Оптимізація транспортних потоків дає змогу підприємствам зменшити витрати на логістику, скоротити час постачання, підвищити ефективність використання транспортних засобів і складів. Це безпосередньо впливає на конкурентоспроможність підприємств і стійкість економічних систем загалом.

Сучасні тенденції розвитку логістики спрямовані на впровадження «зеленої логістики» та розумних транспортних систем, які поєднують математичне моделювання з цифровими технологіями, штучним інтелектом і принципами сталого розвитку. Завдяки цьому транспортна оптимізація стає не лише економічно доцільною, а й екологічно відповідальною, сприяючи зменшенню енергоспоживання, шкідливих викидів і загального навантаження на довкілля.

Отже, транспортні задачі не лише допомагають знаходити оптимальні рішення в межах конкретних виробничих чи логістичних систем, а й формують наукове підґрунтя для розвитку ефективної та екологічно збалансованої економіки майбутнього.

Список використаних джерел

1. Акімов Д. Д., Гавриленко В. В. Розв'язання транспортної задачі в умовах неоднорідності продукції агропромислових підприємств. Комп'ютерно-інтегровані

технології: освіта, наука, виробництво. 2024. Вип. 56. С. 78–85. DOI: https://doi.org/10.36910/6775-2524-0560-2024-56-09

2. Іваницька О. В., Рощина Н. В., Сербул Р. С. Транспортна задача лінійного програмування. Агросвіт. 2015. № 14. С. 35–40.

3. Котов Д., Клименко В., Андрощук О., Мельник В., Горошко О., Азізов Б. Імітаційне моделювання процесів логістичного забезпечення на основі транспортної задачі. Social Development and Security. 2024. Т. 14, № 1. С. 218–228. DOI: https://doi.org/10.33445/sds.2024.14.1.18

Паладі Владислав, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор, Одарущенко Олег

ЗАСТОСУВАННЯ СИСТЕМИ РЕЗЕРВУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В БЕЗПЕКОВИХ СИСТЕМАХ

Вихід у відкритий доступ великих мовних моделей штучного інтелекту (ШІ) з трансформерною архітектурою в кінці 2022 р., значно стимулювало інтерес організацій стосовно можливостей інтеграції ШІ безпосередньо до сфери їхньої діяльності, оскільки це відкривало новий рівень взаємодії як з продуктом, що вони виробляють, так і з клієнтом, який потім його використовує. Але, як і будь-який новий зовнішній чинник, що починає впливати на існуючу систему, інтеграція ШІ закономірно викликала й недовіру та побоювання, особливо у компаній критичного сектору, чия діяльність покриває потенційно небезпечні сфери людського життя, де ціна помилки на будь-якому етапі роботи підприємства вимірюється життями людей. Проте з моменту так званої революції штучного інтелекту пройшло вже достатньо часу, щоб зрозуміти, що дана технологія має потенціал якісно вдосконалити вже існуючі інформаційні системи та не планує зникнути з ринку безслідно. Отже організації, чия діяльність пов’язана з медициною, енергетикою, транспортом та промисловою автоматизацією отримало виклик у вигляді необхідності розроблення безпекових механізмів для регулювання робити моделей з інтегрованим ШІ та адаптації до цього безпекових стандартів.

Для критичних автоматизованих систем керування найважливішими показниками є гарантоздатність (dependability) та довірчоздатність (trustworthiness). Це твердження притаманне незалежно від сфери застосування, хоч в медицині, хоч в електриці, хоч в автономних автомобілях, оскільки низька надійність даних систем може призвести до серйозних наслідків як для користувача, так і для оточуючих. Однак алгоритми машинного навчання штучного інтелекту, на відміну від традиційного програмного забезпечення, демонструють варіативну поведінку в залежності з типом вхідних даних, що створює суттєві труднощі для сертифікації та моделювання відмов.

Відповідно до швидкого впровадження інтелектуальних систем ШІ, активно оновлюються й міжнародні стандарти у сфері функціональної безпечності. Поряд зі звичними IEC 61508 [1], що регулює програмовані та електронні компоненти, розробляються стандарти, орієнтовані саме на ШІ та машинне навчання, а саме:

– ISO/PAS 21448 (SOTIF) – безпека функціональності у системах автономного транспорту [2];

– UL 4600 – оцінювання безпеки автономних продуктів [3];

– ISO/IEC AWI TR 5469 (AI functional safety) – стандарт функціональної безпеки для ШІ-систем у критичних застосуваннях [4];

– VDE-AR-E 2842-61-1 – стандарт розробки автономних систем з дотриманням надійності та довірчоздатності [5].

Вищезгадані стандарти узгоджуються зі структурою життєвого циклу ШІ та детально підходять до етапу машинного навчання (ML), де поруч із тренуванням,

верифікацією та валідацією такої системи виділяється окрема секція вимог стосовно тестування, контролю даних, відстеження небезпечних сценаріїв та можливості штучного інтелекту пояснити свої дії в різних ситуаціях. Однак жоден із розроблених стандартів на даний час ще не здатен повноцінно охопити та точно передбачити поведінку моделей у непередбачуваних середовищах, що вказує на те, що значна частина ризиків може залишитися на рівні архітектурних рішень. Через це зростає потреба у розробці безпекових архітектур, здатних компенсувати ймовірні помилки моделі, невизначеність прогнозу та одночасної відмови апаратних компонентів.

Досить перспективним на цьому плані виглядає резервування системи ШІ – підхід, що дозволяє підвищити стабільність та контрольованість роботи системи у складних умовах експлуатації. Принцип роботи системи резервування полягає в паралельній обробці однакових вхідних даних кількома каналами аналогічними за функціональністю та відмінними за програмною та/або апаратною реалізацією, з кінцевою метою отримання однакових вихідних даних після порівняння. У разі не співпадіння результатів, система сигналізує про свою несправність та блокує подачу вихідного пакету, натомість передаючи керування резервній системі управління об’єктом.

Такий підхід загалом не новий для критичних застосунків, бо ще до інтеграції штучного інтелекту його активно використовували в бортових контролерів літаків, медичному обладнанні та різного роду автоматизованих системах управління. Однак для роботи з ШІ класичних підходів виявляється недостатньо. Як показує спосіб резервування, описаний у відповідному патенті [6], моделі можуть сортувати потік інформації на зони коректної поведінки, зони невизначеності та зони потенційно небезпечної поведінки. Ці зони можуть відрізнятися у різних каналах ШІ. В результаті можуть виникати ситуації, коли два коректно працюючі канали можуть одночасно видавати небезпечний результат або спільно «помилитися» у секторах даних, де модель не була достатньо навчена. Для подолання цієї проблеми запропоновано архітектуру з сегментацією вхідного масиву даних, аналізатором коректності зон поведінки, перетворювачем коду сегмента, мультиплексором, що обирає канал із безпечною поведінкою, блоком діагностики та керування для реконфігурації системи. Дана система здатна відстежувати не лише працездатність каналів, але й визначити, чи не знаходяться їхні вхідні дані у небезпечних або невизначених зонах. Таким чином резервування забезпечує підвищення безпечності за допомогою блокування небезпечних результатів, підвищення надійності шляхом збереження роботи системи до останнього працездатного каналу, зростання довірчоздатності через механізм контролю відповідності вхідних даних зонам безпечної поведінки та зменшення ризику відмови за загальними причинами (common cause failure). Такий підхід робить резервування ШІ не просто технікою дублювання процесів обробки даних, а повноцінною системою безпекового управління.

Однак, попри значний прогрес у вдосконаленні архітектур резервування і появу нових патентованих рішень, у цій галузі все ж залишається кілька невирішених питань [7], як от: відсутність загально узгоджених стандартизованих методів оцінювання зон поведінки моделей; наразі немає універсальних підходів до

формалізації регіонів невизначеності ШІ-моделей у високорозмірних просторах; потреба у сертифікації архітектур резервування для ШІ. Чинні стандарти в повній мірі не охоплюють комбіновані рішення, з інтеграції класичних засобів функціональної безпеки з нейромережами; складність реалізації можливостей ШІ пояснити свої дії в багатоканальних системах. Діагностика стає складнішою при появі взаємодіючих моделей; необхідність у формуванні повного ланцюга довірчоздатності, що вдало поєднує юридичні, етичні та технічні аспекти.

Резервування систем штучного інтелекту є одним із ключових інструментів, для безпечного впровадження інтелектуальних технологій у критичний сектор. Воно компенсує недоліки класичних методів дублювання, враховуючи поведінкові особливості моделей машинного навчання та дозволяє запобігати небезпечним сценаріям, збільшуючи довірчоздатність та надійність системи в цілому. Галузь має високу актуальність як в Україні [8], так і в світі, оскільки вона формує нову парадигму поглядів на безпеку, в якій система може автономно аналізувати власну роботу, проводячи самодіагностику власних станів, поділяти масиви даних на зони за певними визначеними ознаками з подальшим використанням різних алгоритмів для їх обробки. Саме тому застосування систем резервування штучного інтелекту у безпекових системах можна розглядати як один із найперспективніших напрямів сучасного інженерного розвитку, базуючись на поєднанні вимог підприємств критичного сектору із можливостями новітніх технологій. Тож ця нова галузь ще має значний простір для майбутнього науково-технічного розвитку.

Список використаних джерел

1. IEC 61508-1:2010. Functional safety of electrical/electronic/programmable electronic safety-related systems. 2010. URL: https://webstore.iec.ch/en/publication/5515

2. ISO/PAS 21448:2022. Road vehicles – Safety of the intended functionality. 2022. URL: https://www.iso.org/ru/standard/77490.html

3. UL 4600. Edition Standard for Evaluation of Autonomous Products. 2023. URL: https://www.shopulstandards.com/ProductDetail.aspx?productid=UL4600

4. ISO/IEC TR 5469:2024. Artificial intelligence – Functional safety and AI systems. 2024. URL: https://www.iso.org/standard/81283.html

5. VDE-AR-E 2842-61-1. Development and trustworthiness of autonomous/cognitive systems. 2021. URL: https://www.vde-verlag.de/standards/0800738/vde-ar-e-2842-61-1-anwendungsregel-2021-07.html

6. Спосіб резервування системи штучного інтелекту: пат. 156654 Україна: МПК G06F 7/00. № u 2023 04653; заявл. 03.10.2023; опубл. 24.07.2024, Бюл. № 30.

7. Perez J. Artificial Intelligence (AI) for Safety-Critical Systems. Ikerlan. 2022. URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjftrKKy4ORAxXNgSoKHXwHChYQFnoECBoQAQ&url=https%3A%2F%2Fecsccollaborationtool.eu%2Fpublication%2Fdownload%2Fpresentation-jon-perez.pdf&usg=AOvVaw0-jZ75B3W5hTdpynliPSR4&opi=89978449

8. Суходоля О.М. Штучний інтелект в енергетиці: аналіт. доповідь. Київ: НІСД., 2022. С. 49. URL: https://doi.org/10.53679/NISS-analytrep.2022.09

Панченко Ярослав, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

МОЖЛИВОСТІ ІНТЕГРАЦІЇ НЕЙРОМЕРЕЖ В ЛОКАЛЬНИЙ ЗАСТОСУНОК НА ПРИКЛАДІ РОЗРОБКИ ТА НАВЧАННЯ ВЛАСНОЇ МОДЕЛІ

Універсальні моделі ШІ (ChatGPT, Gemini, DeepSeek) є громіздкими та неефективними для вузькоспеціалізованих завдань [1]. Альтернативою є розробка власних оптимізованих нейромереж, що дозволяє зменшити вимоги до обчислювальних ресурсів та підвищити ефективність вирішення конкретних проблем. Процес навчання такої моделі включає: формування тематичного датасету, проведення певної кількості епох навчання та аналіз метрик якості, таких як точність (accuracy) та впевненість (confidence). Досягнення цих метрик на рівні понад 85 % вважається достатнім для завершення ітерації [2]. На прикладі створеної самостійно простої ШІ для аналізу емоційного забарвлення тексту (далі в тексті, Детектор емоцій) демонструються основні етапи розробки та створення демонстраційної «заглушки» в локальний застосунок:

1. Написання коду для створення і навчання ШІ на Python;
2. Використання Google Colab як середовища реалізації безпосереднього навчання моделі та оцінка результатів;
3. Демонстрація роботи готової нейромережі в спеціально розробленому додатку на C++.

Код для створення і навчання ШІ був отриманий на основі наступного запиту до ШІ DeepSeek: «напиши на Python код для створення та навчання нейромережі "Вгадай емоцію" (Емодзі-детектор) з TensorFlow в Гугл Коллаб (весь код в одній панелі), створи для неї потрібний датасет, нехай вона зберігається тільки після закінчення навчання у вигляді .h5 файлу і всіх файлів, що потрібні для наступної інтеграції в застосунок, рішення про її архітектуру приймай найоптимальніші і не складні…» [3].

Після цього було отримано код на мові Python, у який закладено створення ШІ за допомогою ефективних бібліотек TensorFlow, що дозволяють здійснювати налаштування нейромереж з готових команд та гнучких сценаріїв навчання, і сам етап навчання.

Найоптимальнішим середовищем для такого коду є Colab – безкоштовний онлайн-сервіс від Google, що оптимізований для мови Python, здатний візуалізувати хід виконання зображеннями та діаграмами, а також зберігає дані в хмарі. При цьому можна працювати в обчислювальній системі, що має CPU, потужніший за локальний (що критично важливо для навчання ШІ). Для запуску ключового процесу потрібно вставити код в спеціальну чарунку. Створюється проста нейромережа з 10000 параметрами (по 2500 на клас) та архітектурою з трьох шарів (128-64-32) (рис. 1). Для простоти датасет генерується автоматично з заготовлених слів. ШІ вчиться на них протягом певних епох; задача розробника – відслідковувати динаміку зміни параметрів точності та впевненості моделі. Після закінчення навчання отримано звіт

з результатами роботи ШІ. Якщо її середні точність та впевненість вище 85 %, потрібно зберегти всі дані успішного навчання (рис. 2, 3) [4].

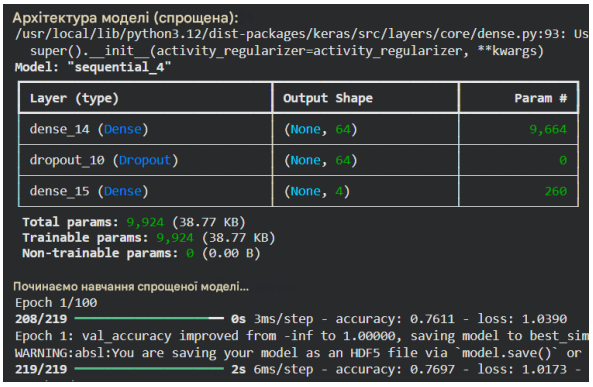

Рисунок 1 – Початок навчання ШІ в Colab

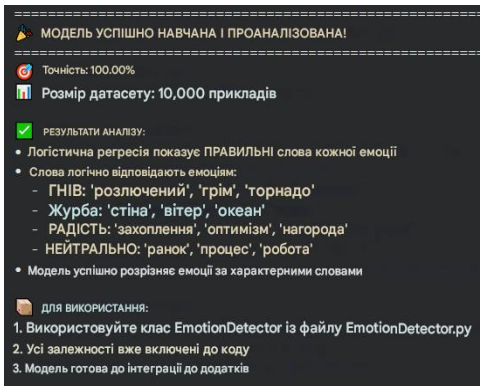

Рисунок 2 – Кінець навчання ШІ в Colab

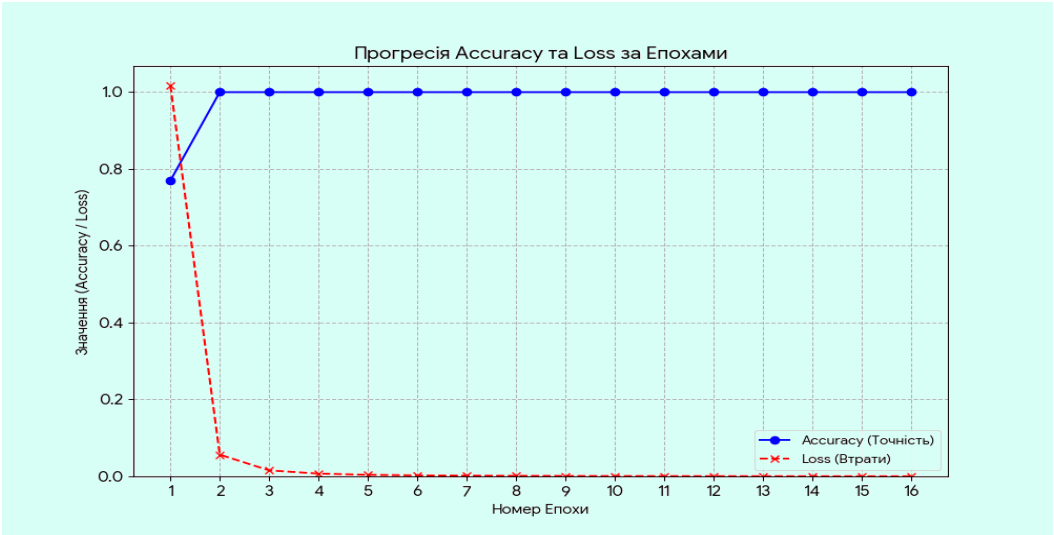

Рисунок 3 – Графік прогресії ШІ під час навчання

Для готових файлів нейромережі потрібно створити відповідний до її функціоналу застосунок. Його зручніше зробити повністю на Python, але, на прикладі Детектора емоцій, було вибрано мову C++ (для кращої оптимізації) та IDE середовище Qt Creator. Файли ШІ створені на Python, тому для їх інтерпретації C++ додаток повинен виконувати обрахунки локальної ШІ в компіляторі Python, отримуючи кінцеві дані. Всі файли нейромережі викликаються локально з папки проекту.

Готовий застосунок представляє собою вікно з полем для вводу, кнопкою для відправки запиту, панеллю відповіді ШІ та панеллю її думок (рис. 4). В силу обмеженості словника нейромережі (коректно аналізує тільки 168 готових слів), під полем вводу знаходиться підказка щодо того, чи є введені слова валідними (тим не менш, введення непередбачених слів не позбавляє можливості аналізувати речення). Сам аналіз йде тільки за чотирма класами (Радість, Сум, Злість, Нейтрально).

В процесі аналізу Детектор емоцій показує перелік параметрів та шляхів, за якими нейромережа знаходить найймовірніший варіант відповіді зі своїм рівнем впевненості.

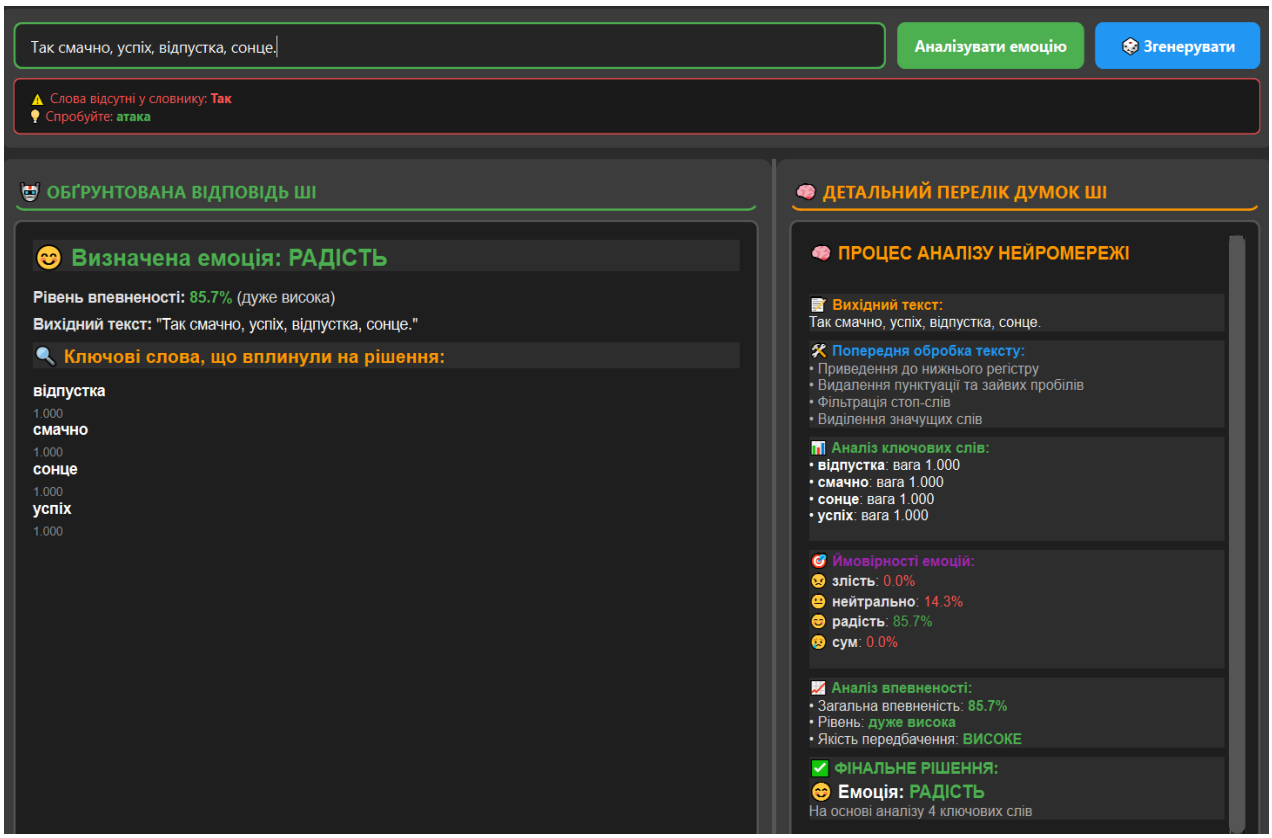

Рисунок 4 – Зовнішній вигляд розробленого застосунку для демонстрації роботи Детектора емоцій

В результаті отримано повністю готовий до локального застосування класифікатор, що працює з мінімальним навантаженням на CPU.

ШІ з прикладу є демонстративною й шаблонною. Для таких цілей більш раціонально використовувати алгоритми, а не нейромережі, що й показано в готовому додатку («заглушка» імітує поводження створеної ШІ). Тим не менш, на даному прикладі видно, що використання ШІ без інтернету/локально без витрати значних ресурсів є не тільки можливим, а й доступним кожному бажаючому.

Список використаних джерел

1. Brown, T. B. at al. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 1877–1901. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

2. Google Colab: Frequently Asked Questions. URL: https://research.google.com/colaboratory/faq.html

3. DeepSeek: чат-бот. URL: https://chat.deepseek.com/share/xntnvi67jasc7l3011

4. Howard, J. at al. Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. 2018. P. 328-339. URL: https://aclanthology.org/P18-1031.pdf

Поночовна Анастасія, здобувачка вищої освіти СВО «Бакалавр»,
спеціальність «В11 Філологія (за спеціалізаціями),
спеціалізація В11.041 Германські мови та літератури
(переклад включно), перша - англійська»
Науковий керівник – старший викладач
кафедри германської і української філології Назаренко Марина

ЛЕКСИЧНІ ТА СТИЛІСТИЧНІ МАРКЕРИ ТЕКСТІВ, ЗГЕНЕРОВАНИХ ШТУЧНИМ ІНТЕЛЕКТОМ: АНАЛІЗ ОЗНАК AI-НАПИСАННЯ В АНГЛОМОВНІЙ ВІКІПЕДІЇ

Останніми роками великі мовні моделі (GPT-4o, Claude 3, Llama 3 тощо) досягли рівня, коли згенерований ними текст важко відрізнити від написаного людиною. Це створює серйозні виклики для Вікіпедії – найбільшого у світі колективного енциклопедичного проєкту, який базується на принципі верифікованості та якісного авторського контенту. Використання ШІ для створення або редагування статей порушує політику спільноти й може призвести до погіршення вмісту ресурсу. Водночас саме спільнота англомовної Вікіпедії однією з перших систематизувала практичні ознаки, за якими можна запідозрити машинне походження тексту. Дослідження цих ознак має міждисциплінарний характер і лежить на перетині сучасної філології, корпусної лінгвістики та інформаційних технологій.

Питання виявлення ШІ-генерованого тексту активно розробляється з 2022 р. У статті Wikipedia: Signs of AI writing спільнотою сформовано найповніший на сьогодні перелік лексико-стилістичних маркерів [1]. У роботі [2] автори пропонують статистичний підхід до автоматичного детектування ШІ-текстів англійською мовою та підтверджують ефективність багатьох ознак зі статті [1]. Дослідження [3] аналізує лексичну передбачуваність і низьку здатність до викидів як ключові ознаки текстів GPT-4. Crothers та ін. звертають увагу на надмірну формальність та повторювані синтаксичні шаблони [4]. Практичний посібник University of Oxford рекомендує редакторам і викладачам використовувати спеціалізований чек-ліст як найбільш операціоналізований інструмент виявлення згенерованого ШІ тексту [5].

У розділі «Content» сторінки Wikipedia: Signs of AI writing [1] наведено понад 20 лексичних і стилістичних індикаторів. Найхарактернішими з філологічного погляду є такі групи.

1. Надмірна формальність і «корпоративний» стиль: вживання «delve», «realm», «tapestry», «beacon», «testament to», «it is crucial to note that», «in conclusion» навіть у нейтрально-енциклопедичних статтях.
2. Лексична передбачуваність і низька варіативність: часте повторення тих самих сполучень («plays a pivotal role», «has garnered significant attention») замість синонімічної різноманітності, характерної для текстів, написаних людиною, а не згенерованих ШІ.
3. Надлишкова вичерпність і «перелічення всього»: ШІ схильний перераховувати всі можливі аспекти теми навіть тоді, коли це не потрібно для енциклопедичної статті.

4. Схильність до узагальнень і банальностей: фрази типу «throughout history, humans have sought to…», «in today's fast-paced world».

5. Неприродні синтаксичні конструкції: дуже довгі складнопідрядні речення з правильним, але механічним розташуванням підрядних частин; часте вживання пасиву там, де активна форма була б природнішою.

6. Відсутність культурно-специфічного або емоційного забарвлення, «плоска» тональність.

Проведений аналіз 50 випадково обраних підозрілих редагувань за 2024–2025 рр. показав, що 86 % з них містили щонайменше три маркери з наведеного списку, а 62 % – п'ять і більше. Найчастіше траплялися «delve» (41 випадок), «it is worth noting that» (38), «plays a pivotal role» (35) та структура «X stands as…» (29). Привертає увагу також майже повна відсутність скорочень (don't, isn't), що нехарактерно для носіїв англійської мови під час швидкого редагування.

Систематизовані спільнотою англомовної Вікіпедії ознаки ШІ-написання є ефективним і доступним інструментом первинного виявлення машинного контенту. З філологічної точки зору вони відображають фундаментальні відмінності між людським і машинним письмо: людське письмо характеризується вищою лексичною варіативністю, контекстуальною доречністю та культурною чутливістю, тоді як ШІ-текст тяжіє до шаблонності, надмірної формальності та механічної вичерпності. Подальший розвиток автоматичних детекторів (на основі burstiness, perplexity, n-gram аналізу) має спиратися саме на ці лінгвістичні маркеери, що робить міждисциплінарний діалог філологів та спеціалістів з інформаційних технологій надзвичайно перспективним.

Список використаних джерел

1. Wikipedia contributors. Wikipedia: Signs of AI writing § Content. Wikipedia, The Free Encyclopedia. 2025. URL: https://en.wikipedia.org/wiki/Wikipedia:Signs_of_AI_writing

2. W. Liang et al. GPT detectors are biased against non-native English writers. Patterns, 2023. Vol. 4, no. 7. P. 100779. URL: https://doi.org/10.1016/j.patter.2023.100779.

3. S. Ruksakulpiwat et al. Assessing the Efficacy of ChatGPT Versus Human Researchers in Identifying Relevant Studies on mHealth Interventions for Improving Medication Adherence in Patients With Ischemic Stroke When Conducting Systematic Reviews: Comparative Analysis. *JMIR mHealth and uHealth*, 2024. Vol. 12. P. e51526. URL: https://doi.org/10.2196/51526.

4. Crothers E., Japkowicz N., Viktor H. L. Machine-generated Text: A Comprehensive Survey of Threat Models and Detection Methods. *IEEE Access*, 2023. P. 1. URL: https://doi.org/10.1109/access.2023.3294090.

5. de Melo G. Detecting AI-generated Content. The Oxford Handbook of the Foundations and Regulation of Generative AI. 2025. URL: https://doi.org/10.1093/oxfordhb/9780198940272.013.0008.

Петрик Артур, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – к.с.-г.н., доцент Протас Надія

ПРОЄКТУВАННЯ СТРУКТУРИ ДАНИХ І API ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ІНТЕГРАЦІЇ З TELEGRAM-БОТОМ: АНАЛІЗ І ВИБІР РІШЕНЬ

У сучасному світі месенджери, зокрема Telegram, стали невіддільною частиною повсякденного спілкування та бізнес-процесів. На 2025 рік Telegram має понад 950 мільйонів активних користувачів, а боти в ньому використовуються для автоматизації завдань, клієнтської підтримки, освіти, e-commerce та моніторингу систем. Актуальність проблеми проєктування інформаційної системи (ІС) підтримки Telegram-бота полягає в необхідності забезпечення надійної інтеграції бота з backend-частиною, де зберігаються дані користувачів, стан розмов та бізнес-логіка. Без належного проєктування структури даних та API бот може стикатися з проблемами масштабованості (обмеження Telegram Bot API – до 30 повідомлень на секунду в чаті), втратою стану діалогу, низькою продуктивністю бази даних чи вразливостями безпеки. Початковий етап проєктування – аналіз вимог, вибір технологій та моделювання структури – дозволяє уникнути цих ризиків і створити гнучку систему, готову до розширення (наприклад, інтеграції з AI чи платежами). Аналіз літератури та інтернет-джерел показує еволюцію підходів до проєктування таких систем. У статті [1] описано серверлес-архітектуру з використанням webhook, AWS Lambda, SQS та API Gateway для асинхронної обробки повідомлень, що забезпечує масштабування та низьку вартість. Автор підкреслює переваги webhook над polling для event-driven систем, де Telegram надсилає оновлення на сервер негайно. Інше джерело пропонує два ключові дизайн-патерни: chain of responsibility для маршрутизації оновлень (оскільки Telegram Bot API надсилає всі апдейти в одну точку) та finite state machine (FSM) для управління станом розмови користувача, що критично для контекстних ботів з багатоступеневими діалогами [2]. У 2025 р., за даними посібника WNexus, рекомендують PostgreSQL або MongoDB як бази даних, Python/Node.js-фреймворки (python-telegram-bot, Telegraf, aiogram) та Redis для кешування стану, щоб уникнути частих звернень до БД [3]. Також у статті [4] акцентовано на асинхронній обробці, мікросервісах та інтеграції з зовнішніми API для реального часу відповідей. Ці джерела підтверджують перехід від простих polling-ботів до масштабованої архітектури з webhook, чергами повідомлень (наприклад, RabbitMQ чи BullMQ) та реляційними/нереляційними БД. Основний матеріал роботи присвячено аналізу та вибору рішень на початковому етапі проєктування ІС для Telegram-бота для сервісу підтримки. Спочатку проведено аналіз вимог: бот повинен обробляти команди, callback-запити, зберігати профілі користувачів (Telegram ID, username, стан розмови), історію повідомлень та пов'язані дані (замовлення та результати тестів). Для інтеграції обрано webhook-механізм, оскільки він ефективніший за long polling при великій кількості користувачів (до 100 тис. одночасних з'єднань на сервер). Структура даних проєктується з урахуванням типу БД. Реляційна модель: таблиці Users (id SERIAL PRIMARY KEY, telegram_id BIGINT UNIQUE, username VARCHAR, state VARCHAR, created_at TIMESTAMP),

Conversations (id SERIAL, user_id BIGINT REFERENCES Users, message_text TEXT, is_from_bot BOOLEAN), Orders чи Tests – для доменних даних. Переваги – ACID-транзакції, складні запити JOIN. NoSQL-варіант (MongoDB): колекції users {telegram_id: number, state: string, profile: object}, messages: array}, зручна для швидких змін схеми та зберігання стану як JSON. Для етапу проєктування було обрано PostgreSQL через необхідність реляційних зв'язків та інтеграцію з ORM (Prisma в Node.js). Для API інформаційної системи спроєктовано RESTful API на базі NestJS (Node.js). Ендпоінти: POST /webhook – приймає апдейти від Telegram (з валідацією токена); GET/POST /users/{telegram_id} – управління профілем; POST /send-message – для проактивних повідомлень бота. API інтегрується з Telegram Bot API через методи sendMessage, editMessageText тощо. Для управління станом використано FSM (бібліотека aiogram в Python надає StatesGroup), де стан зберігається в Redis (швидкий доступ) з TTL 24 год. Архітектура: бот (frontend в Telegram) → Webhook → Backend API → БД + Redis. Для масштабування – Docker + Kubernetes чи серверлес (AWS Lambda).

Таблиця 1. Порівняльний аналіз технологій

Технологія	Переваги	Недоліки	Вибір для проекту
Polling	Простота	Високе навантаження на сервер	Ні
Webhook	Низька затримка, масштабування	Потребує HTTPS	Так
SQLite	Легкий	Не для високого навантаження	Ні
PostgreSQL	Транзакції, масштабованість	Складніша настройка	Так
MongoDB	Гнучка схема	Менша консистентність	Альтернатива
Redis для стану	Швидкість	Додаткова інфраструктура	Так

Таким чином, було обґрунтовано вибір: webhook + PostgreSQL + Redis + FastAPI/aiogram як оптимальне рішення для початкового етапу. Така структура забезпечує масштабованість до тисяч користувачів, збереження стану без втрати контексту та легку інтеграцію з зовнішніми сервісами. Подальші етапи – реалізація та тестування – дозволять перейти до готового продукту.

Список використаних джерел

1. Bojczuk D. How to build a reliable, scalable, and cost-effective Telegram bot. 2022. URL: https://medium.com/wearewaes/how-to-build-a-reliable-scalable-and-cost-effective-telegram-bot-58ae2d6684b1.

2. Two design patterns for Telegram Bots. *DEV Community*, 2021. URL: https://dev.to/madhead/two-design-patterns-for-telegram-bots-59f5.

3. WNexus. Telegram Bot Development Guide 2025 | Build & Scale Bots URL: https://wnexus.io/the-complete-guide-to-telegram-bot-development-in-2025.

4. Ray S. A Developer’s Guide to Building Telegram Bots in 2025 URL: https://stellaray777.medium.com/a-developers-guide-to-building-telegram-bots-in-2025-dbc34cd22337.

Романюк Тетяна, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Харчові технології»,
Науковий керівник – к.с.-г.н., доцент Протас Надія

ЦИФРОВА ТРАНСФОРМАЦІЯ РЕСТОРАННОГО БІЗНЕСУ: СТРАТЕГІЧНІ ПЕРЕВАГИ ВПРОВАДЖЕННЯ СИСТЕМИ SERVIO

У сучасних умовах цифрової трансформації відбуваються докорінні зміни в ресторанній галузі. Зростання конкуренції, підвищення вимог клієнтів до сервісу та швидкості обслуговування, а також необхідність оптимізації внутрішніх процесів спонукають підприємства HoReCa (Hotel – Restaurant – Cafe) до впровадження цифрових рішень. Сьогодні автоматизація стає ключовою умовою стабільного розвитку навіть невеликих кафе чи локальних ресторанів і вже не є привілеєм великих мережевих закладів. Використання сучасних цифрових інструментів дає змогу зменшити вплив людського фактора, оптимізувати роботу персоналу, отримувати достовірні аналітичні дані для прийняття рішень; сприяє підвищенню рівня обслуговування клієнтів і ефективності ведення бізнесу в цілому.

Для автоматизації ресторанної сфери в Україні розроблені і успішно впроваджені інформаційні системи вітчизняних і закордонних розробників. Варто відмітити такі популярні системи автоматизації ресторанів/барів/кафе як xPOS®, SkyService POS, Poster POS, Chameleon POS, Limansoft, Profit Solutions і т.д. [1-6]. Ці та інші POS-системи дозволяють організувати ефективну роботу закладів: надають функціонал для прийому замовлень і обслуговування клієнтів, обліку складських запасів, бухгалтерського обліку, контролю роботи персоналу, аналітики тощо і можуть бути використані на різних пристроях (комп'ютери, ноутбуки, планшети, смартфони, касові термінали).

Серед рішень, що допомагають закладам харчування ефективно керувати бізнесом, особливе місце займає система SERVIO – одна з найвідоміших вітчизняних систем для підприємств HoReCa. Вона поєднує функціональний облік продажів, управління складом, контроль персоналу та автоматизацію процесів обслуговування гостей. Завдяки комплексності та гнучкості налаштувань, SERVIO дозволяє адаптуватися як до потреб невеликих кав'ярень, так і до масштабних ресторанів чи мережевих закладів. Саме тому дослідження можливостей цієї системи на тлі сучасних тенденцій автоматизації є актуальним і важливим для власників та менеджерів ресторанного бізнесу.

Проаналізуємо основні можливості та переваги системи SERVIO.

SERVIO – це комплексне рішення для повної цифрової автоматизації ресторанного бізнесу, що дозволяє ефективно керувати роботою і дозволяє вирішувати всі завдання (обслуговування гостей, ведення складського і управлінського обліку, технологічних і калькуляційних карт, фінансовий облік, облік робочого часу) [7]. Об'єднує усі критичні функції в межах єдиної платформи, що значно зменшує операційну складність та підвищує продуктивність. Саме комплексність є первинною перевагою SERVIO як інформаційної системи управління рестораном. Система SERVIO є модульною Основний модуль для ресторанних закладів – SERVIO POS – забезпечує: оформлення замовлень, їхню

обробку і закриття, підтримку модифікаторів (наприклад, зміни складу страв або додаткових опцій) і т.ін [8]. Додатково POS-система дозволяє налаштовувати роботу залів і столів, здійснювати розділення рахунків між гостями, що є критично важливим для гнучкого обслуговування у ресторанах. Важливим для управління є модуль WorkDesk (адміністратор). SERVIO WorkDesk є базовою вебплатформою для виконання функцій управління фронт-офісом та дозволяє встановити додаткові розширення, як, наприклад, модуль складського обліку чи центр замовлень.

SERVIO POS містить більше 30 додаткових модулів для ресторанного бізнесу: InfoMonitor (інфодисплей клієнта, статус замовлень), кухонний модуль OrderMonitor, POS Base (мобільний персонал), Delivery (центр замовлень на доставку), Kiosk (термінал самообслуговування) мобільні застосунки, та інші. Додатковими превагами є значна кількість функцій, сучасний зрозумілий інтерфейс різними мовами, підтримка різних типів оплат, контроль і облік роботи персоналу, синхронізація із зовнішнім програмним забезпеченням (наявність протоколу API), управління всіма бізнес-процесами та гнучкість налаштувань [7].

Важливою перевагою впровадження SERVIO у свій бізнес є підвищення точності виконання операцій і зменшення людських помилок – зменшення помилок в оформленні замовлень і платежів, точний облік інгредієнтів, контроль над скидками, поверненнями й іншими операціями Цифровий POS-інтерфейс дозволяє офіціантам вводити замовлення безпосередньо, з модифікаторами, з можливістю коректного розподілу по столах чи гостях, що мінімізує ризики неправильного виконання замовлень, недостовірних рахунків або неточностей у залишках.

Впровадження цифрових інструментів дозволяє забезпечити системність контролю над ресурсами (запасами, фінансами та персоналом), що в свою чергу дає змогу оперативно отримувати аналітику та приймати обґрунтовані управлінські рішення. Модулі в SERVIO дозволяють відстежувати продажі, залишки, ефективність роботи персоналу, потік замовлень у режимі реального часу, що сприяє оптимізації закупівель, зниженню надлишкових витрат, плануванню ресурсів і, в результаті –підвищенню рентабельності підприємства [9].

Ще одним аргументом виступає покращення сервісу та клієнтського досвіду, що є критично важливим у ресторанному бізнесі. Завдяки миттєвій передачі замовлень у кухню, обробці платежів або закриттю рахунків прямо біля столу, скорочується час обслуговування, зникають черги та затримки, зменшується навантаження на персонал, що підвищує задоволеність відвідувачів. Крім того, можливості систем лояльності, зберігання історії замовлень, знижок або бонусів сприяють формуванню позитивного іміджу закладу, утриманню клієнтів і формуванню повторного відвідування.

Ще однією перевагою SERVIO є її масштабованість та гнучкість – систему можна легко адаптувати під різні формати закладів: від невеликих кафе до великих ресторанів або мереж із кількома точками, з можливістю підключення додаткових модулів за потребою. Така гнучкість дозволяє закладу розвиватися поступово, і запроваджувати цифрові зміни відповідно до зростання обсягу бізнесу чи зміни формату. Це особливо актуально в умовах змінного попиту, сезонності або розширення бізнесу. Попри значну кількість переваг, впровадження системи SERVIO, як і будь-яких інших систем, супроводжується низкою потенційних

пересторог, які варто враховувати при плануванні цифрової трансформації ресторанного підприємства. Перш за все, інтеграція комплексного програмного забезпечення потребує початкових інвестицій – як фінансових, так і організаційних. До витрат належать не лише ліцензії на програмні модулі, а й придбання сумісного обладнання (POS-терміналів, сенсорних панелей, мережевої інфраструктури), а також налаштування системи під специфіку закладу, що може бути бар'єром для невеликих закладів із обмеженим бюджетом. Важливим аспектом є необхідність систематичного навчання персоналу (офіціантів, адміністраторів, барменів і менеджерів) для ефективного використання всіх функцій системи. Додаткові ризики можуть бути пов'язані з необхідністю стабільної IT-інфраструктури (надійного інтернет-з'єднання особливо в періоди пікового навантаження, регулярного оновлення програмного забезпечення, технічної підтримки) та дотриманням високих стандартів кібербезпеки.

Упровадження систем автоматизації в ресторанному бізнесі вже не можна вважати інновацією. Водночас, нові підприємці-ресторатори постають перед необхідністю вибору програмних рішень, здатних забезпечити комплексну автоматизацію всіх операційних, облікових та клієнтських процесів. Такі системи мають гарантувати ефективну роботу персоналу, чіткий розподіл функцій між працівниками, підтримувати взаємодію персоналу через сучасні технології та забезпечувати високий рівень комунікації з гостями, що є невід'ємною складовою професійного сервісу, дозволяє забезпечити приємне перебування в закладі клієнтів і завоювати прихильність вибагливих споживачів. Цифровізація з SERVIO здатна трансформувати бізнес і сприяє досягненню успіху.

Список використаних джерел

1. Рішення xPOS. URL: https://xpos.ua/ua/ (дата звернення: 11.11.2025).
2. Автоматизація ресторану Skyservice. URL: https://skyservice.pro/uk/automation/restaurant/ (дата звернення: 11.11.2025).
3. Poster POS. URL: https://joinposter.com/ua (дата звернення: 14.11.2025).
4. Програма Chameleon POS. URL: https://chm-s.com/produkty/ chameleon-pos/ (дата звернення: 14.11.2025).
5. Limansoft – POSсистеми та програми автоматизації. URL: https://limansoft.com/ua (дата звернення: 15.11.2025).
6. Profit. Автоматизація бізнесу. URL: https://profit.com.ua/ (дата звернення: 15.11.2025).
7. Автоматизація ресторанів. Serviosoftware. URL: https://serviosoft.com/uk/rishennya/hotel-automation-copy (дата звернення: 11.11.2025).
8. SERVIO POS. https://expertsolution.com.ua/produkty/pz/servio-pos-dlya-avtomatyzatsiyi-restorannogo-biznesu/ (дата звернення: 11.11.2025).
9. Яку POS систему обрати закладам в Україні. URL: https://choiceqr.com/uk/news/yaku-pos-systemu-obraty-zakladam-v-ukraini/?utm_source=chatgpt.com (дата звернення: 14.11.2025).

Руцький Андрій, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

ТЕХНОЛОГІЇ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Технології генерації відеоконтенту на основі штучного інтелекту еволюціонували від нестабільних GAN-моделей до багатокаскадних дифузійних систем промислової якості. Це забезпечило можливість автоматизувати створення відеопродукції, яка раніше вимагала значних виробничих ресурсів. Алгоритми поєднують складні моделі простору й часу, механізми кодування та декодування, а також контроль через текст, зображення чи аудіосигнал. Технологічний стек формується мережами, здатними моделювати тисячі кадрів, зберігаючи структурну цілісність сцени, руху та контексту [1, 5]. Сучасні системи базуються на двох ключових процесах: кодуванні багатовимірного відеопотоку у компактне латентне представлення та подальшому синтезі нової часової послідовності кадрів. У першому етапі моделі на зразок VAE або VQ-VAE компресують кожен кадр у стислий простір. Це знижує навантаження на навчання та дозволяє працювати з відеоданими високої роздільності [2]. Потім система моделює часову залежність: або прогнозуючи кадри послідовно (авторегресивний підхід), або реконструюючи шумову латентну послідовність через дифузійний процес (дифузійні моделі), або створюючи кадри за принципом генератор-дискримінатор у GAN-системах. Ключовою особливістю є умовність (conditioning). Моделі поєднують текстові трансформери з візуальними енкодерами, що дозволяє розуміти опис сцени: «літак злітає на тлі заходу сонця». Завдяки цьому AI не просто генерує довільні рухомі зображення, а створює контрольований відеоконтент із заданими параметрами [1, 3]. Основні технологічні підходи:

1. GAN-орієнтований підхід. Моделі VideoGAN забезпечують безперервну генерацію коротких фрагментів, використовуючи два дискримінатори - просторовий і часовий. Вони швидкі, проте мають фундаментальні проблеми стабільності та втрати семантичної цілісності при збільшенні тривалості сцени [4]. Типова помилка –«розпад структури» через накопичення артефактів.

2. Авторегресивні токен-моделі. Підхід VideoGPT кодує відео у токени, а трансформер прогнозує їх як мовну послідовність [2]. Це дозволяє точно моделювати ймовірності, але обмежує масштабованість, оскільки довгі послідовності вимагають величезних ресурсів.

3. Дифузійні відеосистеми. Дифузійні алгоритми (Imagen Video, PIKA, Sora) працюють у двох фазах: руйнування латентного сигналу шумом та його поступове відновлення у відео високої якості. Метод є найбільш стійким до артефактів і забезпечує максимальний рівень фотореалістичності та точності динаміки [1, 5]. Каскадність дозволяє генерувати 512×512, 1024×1024 і вище, починаючи з низькодетального чернеткового відео.

4. Гібридні моделі для довгих відео. Моделі Phenaki застосовують принцип змінної довжини послідовності, комбінуючи токен-підхід і дифузійні механізми. Це

дозволяє продукувати сцени тривалістю хвилини без втрати структури сюжетної лінії [3, 6]. Для цього система генерує ключові сегменти, а потім зшиває їх у суцільну хронологію.

5. Нейронний рендеринг. NeRF та подібні алгоритми спеціалізуються не на довільних відео, а на реконструкції 3D-сцен із подальшим рухом камери. Вони корисні для віртуального продакшену, але не підходять для генерації «з нуля» без реальних зразків.

Дифузійні моделі на сьогодні є індустріальним стандартом за якістю контенту та стійкістю генерації. Авторегресивні системи забезпечують найвищу точність контролю, проте їх використання обмежене масштабом обчислень. GAN-підходи зберігають свою нішу там, де критичні швидкість та компактність моделі. Гібридні системи - найперспективніший напрям для тривалих відео, особливо при створенні сюжетних роликів із плавними переходами. Головною проблемою залишається флуктуація рухомих об’єктів у довготривалих сценах: обличчя можуть деформуватись, предмети - змінювати форму, а структура сцени - «спливати». Значна частина моделей працює лише в латентному просторі, що обмежує деталізацію. Моделі потребують надвеликих датасетів, що ускладнює юридичні та етичні аспекти авторського права [5]. Технології активно впроваджуються у маркетингу, генеративному дизайні продуктів, розробці кінематографічних раскадровок, гейміфікації, тренажерах для автономного транспорту, симуляції навколишнього середовища для робототехніки. Корпоративний сектор поступово інтегрує відео-GPT у внутрішні системи для генерації навчальних матеріалів та автоматичного контент-продакшену. Розглянуті системи генерації відеоконтенту стабільно еволюціонують у бік зростання якості, тривалості та контрольованості. Дифузійні моделі задають стандарт промислової точності, авторегресивні дають найвищу керованість, тоді як гібридні системи забезпечують оптимальний баланс для довгих сцен. Подальший розвиток пов'язаний із зменшенням ресурсоємності, стандартизацією етичних механізмів та формуванням відкритих безпечних датасетів.

Список використаних джерел

1. Imagen Video: text-conditional video diffusion models. URL: https://imagen.research.google/video/ (дата звернення: 21.11.2025).

2. W. Yan, Y. Zhang, P. Abbeel, A. Srinivas. VideoGPT: Video Generation using VQ-VAE and Transformers. *ArXiv preprint*, 2021. URL: https://arxiv.org/abs/2104.10157.

3. R. Villegas et al. Phenaki: Variable Length Video Generation From Open Domain Textual Description. *ArXiv preprint*, 2022. URL: https://arxiv.org/abs/2210.02399.

4. Tulyakov S.et al. MoCoGAN: Decomposing Motion and Content for Video Generation. *2018 IEEE CVF Conf. on Computer Vision and Pattern Recognition* (CVPR), Salt Lake City, UT, 18–23 June 2018. 2018. URL: https://doi.org/10.1109/cvpr.2018.00165.

5. Y. Wang et al. Survey of Video Diffusion Models: Foundations and Advances / 2024. URL: https://arxiv.org/abs/2402.10657.

6. A Survey on Long Video Generation: Challenges, Methods and Directions / arXiv preprint. 2024. URL: https://arxiv.org/abs/2403.16407.

Силантьєв Віктор, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Слюсар Вадим

ОСОБЛИВОСТІ ВИКОРИСТАННЯ АСИСТЕНТА AI НА ПЛАТФОРМІ N8N

Як відомо, у 2019 р. був заснований стартап n8n, що розробляє платформу для автоматизації бізнес-процесів і взаємодії зі штучним інтелектом (AI) [1]. Ця платформа спирається на no-code-рішення для побудови інтегрованих робочих процесів.

Щоб створювати, налагоджувати та оптимізувати робочі процеси асистент AI [2]. Він може вести користувача крок за кроком, наприклад, від відповідей на запитання про n8n до допомоги з кодуванням і виразами. Він може оптимізувати процес побудови робочих процесів і підтримати вас у використанні можливостей платформи n8n.

Асистент AI має доступ до елементів, що відображаються у редакторі n8n, але не до особистих даних, таких як відомості про клієнтів або вхідні та вихідні дані робочих процесів. Він використовує поєднання спеціалізованих AI-агентів, інформації з документації та спільноти n8n, а також власних підказок, щоб надавати актуальні відповіді у реальному часі.

Проведений аналіз дозволяє визначити кілька інструментів, які пропонує асистент AI. У процесі роботи з n8n важливу роль відіграє комплексна підтримка, що охоплює виявлення та усунення проблем у виконанні вузлів для забезпечення безперебійного функціонування робочих процесів, оперативне отримання відповідей на запитання щодо як специфічних функцій, так і загальної функціональності системи, а також рекомендації з програмування, включно з використанням SQL та JSON, для оптимізації логіки вузлів і обробки даних. Крім того, значущим є опанування методів створення та вдосконалення виразів, що дозволяє підвищити ефективність автоматизації, а також дотримання найкращих практик конфігурації облікових даних, які забезпечують безпечне та результативне керування доступами у середовищі n8n. Для прикладу, на рис. 1 наведений асистент AI n8n, що підказує, як отримати облікові дані для API Telegram.

AI Assistant у n8n функціонує на основі доступу до всіх елементів, що відображаються у робочому інтерфейсі, за винятком фактичних значень вхідних та вихідних даних, таких як персональна або клієнтська інформація, що гарантує дотримання вимог безпеки й конфіденційності; детальніше про перелік доступних даних можна дізнатися безпосередньо через функціонал AI у n8n. Використання асистента доступне всім користувачам хмарного тарифного плану, що забезпечує широке охоплення та інтеграцію інструмента у різні сценарії автоматизації. Робота асистента базується на сучасних механізмах штучного інтелекту, що включають комбінацію спеціалізованих агентів, орієнтованих на різні аспекти роботи n8n, використання підходу RAG для вибірки релевантних знань з документації та форуму спільноти, а також застосування кастомних підказок, системи пам’яті та контекстної інформації для формування точних і релевантних відповідей.

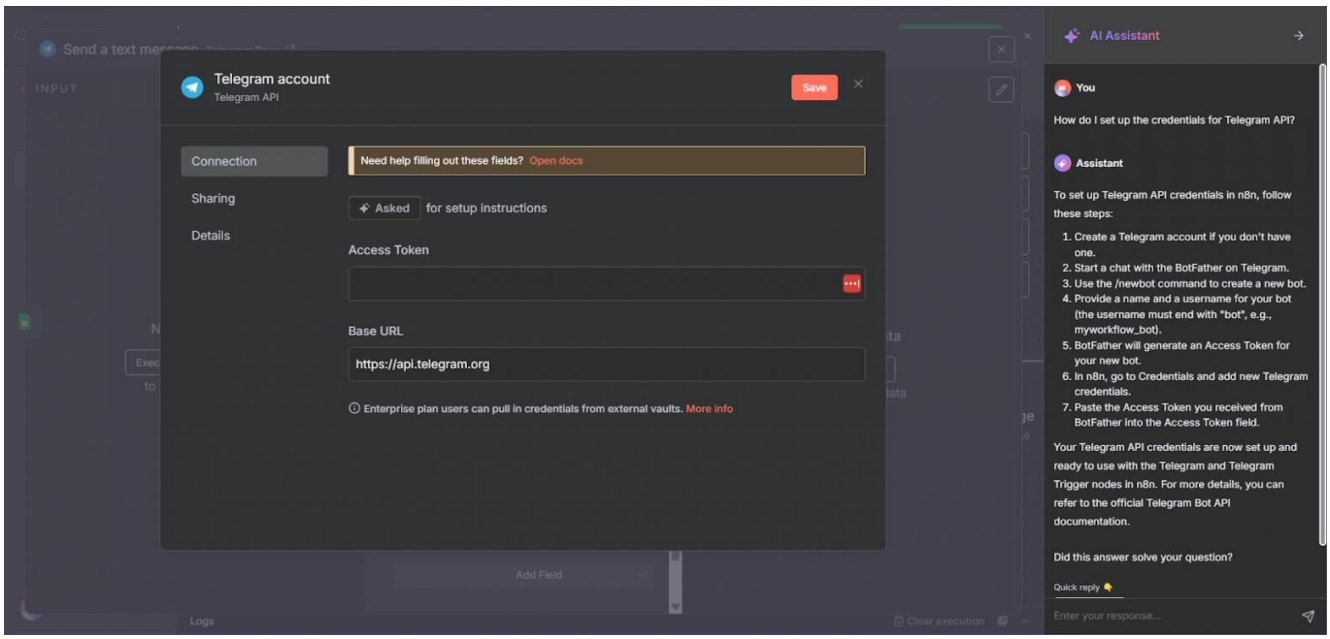

Рисунок 1 – Приклад відповіді асистент AI n8n

Для максимально ефективного використання AI Assistant у середовищі n8n рекомендується підтримувати активну взаємодію з системою, надаючи достатній контекст і уточнюючи потреби, оскільки покрокове співробітництво дає змогу отримувати більш релевантні рекомендації. Найкращих результатів можна досягти, формулюючи конкретні та цілеспрямовані запитання, зокрема щодо налаштування облікових даних чи роботи окремих вузлів, адже чіткість запиту напряму впливає на точність відповіді. Доцільно також багаторазово уточнювати або розвивати запропоновані асистентом варіанти рішень, експериментуючи з альтернативними підходами та використовуючи наданий зворотний зв’язок для їх поступового вдосконалення. Серед типових запитів, які варто застосовувати на практиці, можна виділити відлагодження помилок у робочих процесах, інструкції щодо конфігурації облікових даних, пояснення логіки роботи конкретного workflow, допомогу у написанні коду або побудові нових рішень у n8n. Варто памятати, що асистент AI недоступний при локальному хостингу n8n. Він доступний тільки користувачам хмарних планів.

Список використаних джерел

1. Німецький стартап n8n залучив $180 млн для розвитку ШІ-агентів – оцінка зросла до $2,5 млрд. *Borgexpert*. URL: https://borgexpert.com/news/nimetskyj-startap-n8n-zaluchyv-180-mln-dlia-rozvytku-shi-ahentiv-otsinka-zrosla-do-2-5-mlrd

2. AI Assistant. *n8n*. https://docs.n8n.io/manage-cloud/ai-assistant/#tips-for-getting-the-most-out-of-the-assistant (дата звернення: 20.11.2025).

Турбай Євгеній, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

ПЕРЕВАГИ ВИКОРИСТАННЯ SPA-АРХІТЕКТУРИ ПРИ РОЗРОБЦІ ВЕБКАТАЛОГІВ ТОВАРІВ

В умовах стрімкого розвитку глобальної цифрової економіки електронна комерція (e-commerce) стала домінуючим каналом взаємодії між бізнесом та споживачем. Сучасний користувач висуває високі вимоги до веб-ресурсів: швидкість завантаження, інтуїтивно зрозумілий інтерфейс та миттєва реакція на дії. Особливо критичним ці параметри є для веб-каталогів товарів, де користувачі взаємодіють з великими масивами даних, постійно використовуючи інструменти пошуку та багатофакторної фільтрації. Актуальність дослідження зумовлена необхідністю переходу від традиційних підходів веб-розробки до більш сучасних архітектурних рішень. Традиційна модель багатосторінкових застосунків (Multi Page Application, MPA) поступово поступається місцем архітектурі односторінкових застосунків (Single Page Application), котра дозволяє створювати високопродуктивні інтерфейси, наближені за досвідом використання до нативних програм [1]. Традиційна архітектура MPA базується на логіці серверного рендерингу. При кожній дії користувача (перехід за посиланням, застосування фільтру, тощо) браузер відправляє запит на сервер, який генерує нову HTML-сторінку та повертає її клієнту. У контексті каталогу товарів це створює суттєві незручності: зміна навіть одного параметру фільрації (наприклад, бренду) призводить до перезавантаження всього інтерфейсу, що збільшує час очікування. Натомість SPA-архітектура, реалізована за допомогою бібліотеки React, пропонує підхід, при якому завантаження оболонки сторінки відбувається лише один раз. Подальша взаємодія здійснюється асинхронно через API, оновлюючи лише необхідні частини DOM-дерева. Це дозволяє уникнути зайвого трафіку та забезпечує миттєвий відгук інтерфейсу на дії користувача. Окремої уваги заслуговує коомпонентний підхід та механізм управління станом (State Management), які є фундаментальними для SPA. Наприклад, у веб-каталозі інтерфейс розбито на незалежні ізольовані компоненти (картка товару, панель фільтрів, кошик покупок), кожен з яких керує власною логікою та відображенням. Коли користувач змінює критерії пошуку, React не оновлює всю сторінку, а лише змінює стан конкретного компонента, що відповідає за виведення списку товарів. Завдяки алгоритму узгодження (Reconcilliation) бібліотека обчислює мінімально необхідну кількість змін і застосовує їх до реальоного DOM, що робить процес фільрації візуально непомітним та плавним для користувача, на відміну від «важких» оновлень у традиційних системах. Для наочної демонстрації відмінностей було проведено порівняльний аналіз характеристик обох підходів (табл. 1). На основі проведеного аналізу та практичної розробки сервісу можна виділити ключові переваги використання SPA (на базі React та MySQL) для вирішення завдань електронної комерції. Підвищена швидкодія та покращений UX: використання концепції Virtual DOM у бібліотеці React дозволяє мінімізувати кількість операцій з реальним DOM браузера. При заміні стану фільтрів система перераховує різницю

між поточним та новим станом інтерфейсу і точково оновлює лише картки товарів. Для користувача це виглядає як миттєва реакція системи без перезавантаження сторінки [2]. Ефективна реалізація динамічного пошуку: у веб-сервісі де реалізовано взаємодію між React та MySQL, при введенні пошукового запиту клієнтський додаток формує запит RESTful API. Сервер бази даних MySQL виконує оптимізовану вибірку і повертає масив об’єктів у форматі JSON. Відсутність необхідності передавати HTML-теги значно зменшує навантаження на канал передачі даних [3].

Таблиця 1 – Порівняльна характеристика архітектур MPA та SPA для e-commerce систем

Критерій порівняння	MPA-архітектура	SPA-архітектура
Життєвий цикл сторінки	Повне перезавантаження сторінки при кожному запиті	Завантаження оболонки один раз. Далі – лише обмін диними
Швидкість відгуку	Залежить від генерації HTML на сервері та пропускної здатності мережі	Висока, завдяки оновленню лише змінених елементів.
Обсяг трафіку	Великий (передається повна розмітка, стилі та скрипти при кожному кліку)	Мінімальний (передаються лише корисні дані про товари)
Навантаження на сервер	Високе, сервер займається рендренгом інтерфейсу кожного клієта.	Знижене, сервер працює як API
Користувацький досвід	Переривчастий	Плавний
Архітектура	Монолітна	Розділена

Оптимізація авторизації та безпеки: використання SPA дозволяє реалізувати безшовну авторизацію. Стан користувача (токен доступу) зберігається у клієнті, що дозволяє миттєво змінювати інтерфейс (наприклад, надавати доступ до редагування товарів для адміністратора) без зайвих запитів до сервера при навагації між розділами. Отже, аналіз предметної області веб-сервісу підтверджують, що використання SPA-архітектури на базі React є оптимальним рішенням для сучасних систем управління каталогами товарів. Цей підхід забезпечує суттєве підвищення швидкодії, зменшує навантаження на сервер та надає користувачам плавний досвід взаємодії. Поєднання реактивного фронтенду з надійною реляційною базою даних MySQL дозволяє створити масштабовану та зручну систему, яка відповідає вимогам сучасного ринку.

Список використаних джерел

1. Flanagan D. JavaScript: The Definitive Guide. 7th Edition. O'Reilly Media, 2020. 706 p.

2. Google Developers. Why does speed matter? Web Fundamentals. URL: https://web.dev/learn/performance/why-speed-matters (дата звернення: 20.11.25).

3. Ackermann P. Full Stack Web Development: The Comprehensive Guide. Packt Publishing, 2020. 450 p.

Хованець Ілля, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ СОЦІОЛОГІЧНИХ ОПИТУВАНЬ З АВТОМАТИЧНИМ ФОРМУВАННЯМ РІЧНИХ PDF-ЗВІТІВ НА ОСНОВІ ДАНИХ EXCEL

В умовах акредитації освітньо-професійних програм (ОПП) та постійного моніторингу якості освіти вищі навчальні заклади зобов'язані регулярно збирати, аналізувати та візуалізувати відгуки стейкхолдерів – студентів, викладачів, роботодавців. В Україні, згідно з вимогами Національного агентства із забезпечення якості вищої освіти, такі опитування проводяться щорічно, а результати мають бути представлені у зручній графічній формі. Проте ручна обробка даних з Excel-файлів за допомогою Microsoft Excel чи Google Sheets є трудомісткою, схильною до помилок і не забезпечує однакового оформлення звітів за різні роки. Актуальність проблеми полягає в необхідності створення автоматизованого інструменту, який би з одного Excel-файлу з результатами анкетування за кілька років автоматично генерував окремі багатосторінкові PDF-звіти для кожного року з круговими та стовпчастими діаграмами, текстовими відповідями та єдиним стилем оформлення. Такий підхід скорочує час підготовки звітності з кількох днів до кількох хвилин, підвищує точність візуалізації та забезпечує можливість швидкого порівняння динаміки думок стейкхолдерів.

Аналіз літератури та відкритих джерел показав, що для вирішення подібних завдань найчастіше використовують комбінацію бібліотек pandas, matplotlib та спеціалізованих модулів для роботи з PDF. Офіційна документація бібліотеки pandas підкреслює її високу ефективність при зчитуванні та фільтруванні даних з Excel, зокрема завдяки функціям read_excel та групуванню за стовпцями [1]. Документація matplotlib разом з модулем PdfPages описує механізм створення багатосторінкових PDF-документів з автоматичним розміщенням графіків та текстових блоків на сторінках формату A4 [2]. Важливим джерелом є офіційний туториал бібліотеки pdf-lib.js [3], але для Python-рішень перевагу віддано нативному бекенду matplotlib, що підтверджується прикладом на офіційному сайті. Четвертим джерелом [4] є стаття-огляд на сайті Towards Data Science «Automated Report Generation with Python», де автори порівнюють різні підходи та роблять висновок про перевагу комбінації pandas + matplotlib + PdfPages для академічних і освітніх звітів завдяки повній автономності (без зовнішніх сервісів) та високій якості отримуваних PDF.

Основний матеріал дослідження присвячено етапам проєктування та безпосередній програмній реалізації системи. Структура Excel-файлу передбачає перший стовпець як рік опитування, а решту стовпців – питання з відповідями респондентів. Розроблений скрипт на мові Python 3 виконує такі кроки:

1. Зчитування даних за допомогою pandas.read_excel з автоматичним визначенням назви стовпця з роками.
2. Визначення унікальних років та створення окремого PDF-файлу для кожного року (наприклад, «Зб-Результати анкетування...2024.pdf»).

3. Генерація титульної сторінки з назвою ОПП та роком.

4. Циклічна обробка кожного питання:

– для питань закритого типу – побудова кругової діаграми з функцією autopct та автоматичним об’єднанням дрібних категорій у «Інше»;

– для питання з множинним вибором (використання ІКТ) – горизонтальна стовпчаста діаграма з підрахунком кожного варіанту;

– для останнього відкритого питання – форматований текстовий список відповідей з перенесенням рядків.

5. Автоматичне збереження всіх графіків і тексту в один багатосторінковий PDF за допомогою контекстного менеджера PdfPages.

Ключові технічні рішення: використання textwrap для коректного перенесення довгих формулювань питань та відповідей; функція wrap_text для підписів на діаграмах; динамічне налаштування відступів plt.subplots_adjust для уникнення обрізання тексту; підтримка української кодової сторінки завдяки plt.rcParams.update({'font.family': 'DejaVu Sans'}). Реалізований код є модульним, легко масштабується на інші анкети та не потребує встановлення додаткових платних бібліотек.

Розроблена система повністю автоматизує процес перетворення сирих даних опитувань з Excel у професійні річні PDF-звіти з високоякісною візуалізацією. Використання лише стандартних бібліотек pandas, matplotlib та вбудованих модулів Python гарантує переносимість рішення на будь-який комп’ютер з Python 3. Час генерації одного річного звіту (15–20 сторінок) становить менше 5 секунд. Отримане програмне забезпечення вже успішно застосовується для підготовки матеріалів до акредитації ОПП «Інформаційні управляючі системи» та може бути адаптоване для інших освітніх програм і закладів вищої освіти України.

Список використаних джерел

1. pandas: powerful Python data analysis toolkit. URL: https://pandas.pydata.org/docs/

2. Matplotlib: Creating multi-page pdf // Matplotlib 3.8.2 documentation. URL: https://matplotlib.org/stable/gallery/misc/multipage_pdf.html

3. backend_pdf.py // Matplotlib 3.8.2 documentation. URL: https://matplotlib.org/stable/api/backend_pdf_api.html

4. Płoński P. Automated PDF Reports with Python URL: https://mljar.com/blog/automated-reports-python/

Шкурба Анастасія, здобувачка вищої освіти СВО «Бакалавр»,
спеціальність 126 Інформаційні системи та технології,
Науковий керівник – к.ф.-м.н., доцент Копішинська Олена

АНАЛІЗ ВЗАЄМОЗВ'ЯЗКУ МІЖ ПІДХОДАМИ ДО АДАПТИВНОСТІ ТА ІНТЕРАКТИВНІСТЮ ІНТЕРФЕЙСУ У ПІДВИЩЕННІ КОНВЕРСІЇ КОМЕРЦІЙНИХ ВЕБДОДАТКІВ

Сучасний розвиток вебтехнологій та стрімке зростання електронної комерції створюють необхідність у підвищенні ефективності бізнес-додатків через оптимізацію користувацького досвіду. Багато сучасних комерційних вебдодатків стикаються з ситуацією, коли мобільна версія сайту реалізована без урахування інтерактивності, або навпаки − інтерактивні компоненти використовуються без належної адаптації під різні пристрої, що призводить до неефективного використання потенціалу вебресурсу та погіршення взаємодії користувача з додатком. У цьому контексті виникає необхідність комплексного аналізу взаємозв'язку між обраними підходами до адаптивності − mobile-first та desktop-first − та інтерактивністю інтерфейсу.

Метою дослідження є визначення оптимальної комбінації підходів до адаптивності та інтерактивності елементів інтерфейсу, яка забезпечує максимальну ефективність бізнес-додатків у контексті взаємодії користувачів та підвищення конверсії. Для досягнення цієї мети передбачено виконання ряду завдань: аналіз існуючих підходів до адаптивності та інтерактивності, огляд досліджень та практик щодо їх впливу на поведінку користувачів, розробка експериментальної моделі для тестування, збір та аналіз кількісних і якісних даних щодо ефективності цих підходів, а також формулювання рекомендацій для оптимізації комерційних вебдодатків.

Актуальність дослідження зумовлена стрімким зростанням використання мобільних пристроїв для доступу до комерційних вебресурсів та необхідністю забезпечення ефективної взаємодії користувачів з інтерфейсом. Згідно з сучасними аналітичними даними, значна частина трафіку на торгівельні платформи припадає саме на телефони, що робить критично важливим впровадження адаптивних підходів до верстки, таких як mobile-first, для забезпечення зручності та швидкості доступу. Водночас інтерактивні елементи інтерфейсу, включаючи анімовані кнопки, модальні вікна та підказки, здатні підвищувати залученість користувачів і стимулювати конверсію, проте їх ефективність значною мірою залежить від правильної інтеграції з адаптивною структурою сайту.

У веброзробці існує два популярні підходи до адаптивного дизайну: mobile-first та desktop-first. Підхід mobile-first означає, що дизайн і верстка починаються з найменших екранів — спочатку проектується мобільна версія, а потім вона масштабується до планшетів і десктопів. Така стратегія зосереджує увагу на обмеженнях мобільного екрана (менша площа, сенсорні елементи), що змушує оптимізувати контент, пріоритизувати ключові функції й забезпечувати швидке завантаження [1].

З іншого боку, підхід desktop-first передбачає створення дизайну для великих екранів спершу, а потім його адаптацію під маленькі. Цей підхід може бути зручнішим для складних інтерфейсів з багатьма компонентами, але ризикує створити громіздкі мобільні версії або не оптимізовані користувацькі потоки на смартфонах. У контексті адаптивного вебдизайну також варто згадати adaptive web design (AWD) − підхід, коли сайт має декілька заздалегідь підготовлених макетів для різних типів пристроїв, а при завантаженні обирається та версія, яка найкраще відповідає розміру екрана [2]. Кожен із підходів має свої переваги та недоліки: mobile-first сприяє економії ресурсів, кращій продуктивності й SEO-оптимізації, тоді як desktop-first може забезпечити більшу гнучкість для складного функціонального дизайну, але може втрачати в ефективності на мобільних пристроях.

Дослідження показують, що адаптивний дизайн тісно пов’язаний зі зростанням конверсії та поліпшенням користувацького досвіду. Наприклад, у статті наукового порталу ResearchGate проаналізовано вплив UX/UI покращень і підвищення вебдоступності на платформах e-commerce, виявлено позитивний вплив на конверсії [3]. Інша робота присвячена вивченню візуальних елементів інтерфейсу (кольори, компоновка, типографіка) і їхнього впливу на взаємодію користувача: дослідники виявили, що саме дизайн цих елементів може суттєво змінювати сприйняття сайту [4].

У свою чергу, інтерактивні елементи інтерфейсу − кнопки, підказки, модальні вікна − відіграють критичну роль у залученні користувача та стимулюванні дій. Згідно з аналізом, який проводить компанія COI, інтерактивний вебдизайн сприяє підвищенню активності користувачів і може значно вплинути на конверсію [5]. Наприклад, Design Systems Collective у статті «The Art and Science of Button Design» розглядає принципи створення ефективних кнопок: кольори, розмір, фідбек − все це впливає на те, наскільки кнопка спонукає користувача натиснути [6]. Щодо модальних вікон, UX-дослідження показують, що вони можуть бути як корисними (наприклад, для підтвердження дій або контекстних підказок), так і дратівливими, якщо їх надто багато. A/B-тестування таких «вікон» може допомогти знайти оптимальний баланс між залученням і нав’язливістю [7]. До того ж, практичний кейс від Fluer описує, як простий інтерактивний інструмент (стайл-вікторина) приніс 38 % зростання кліків і 22 % підвищення конверсії [8]. Для проведення дослідження обрано змішаний підхід, який поєднує кількісний та якісний аналіз даних (quant + qual). Це дозволяє не лише виміряти кількісні показники ефективності вебдодатків, такі як конверсія, час перебування на сторінці та кількість кліків на інтерактивні елементи, а й оцінити суб’єктивне сприйняття користувачами адаптивності та інтерактивності інтерфейсу.

Кількісна складова забезпечує об’єктивну оцінку результатів через статистичний аналіз, тоді як якісна дозволяє врахувати нюанси поведінки та мотивів користувачів, що важко зафіксувати лише за допомогою аналітики. Цільова група дослідження представлена користувачами комерційних вебдодатків різних типів, зокрема SaaS-платформ та e-commerce сайтів, для яких ефективна взаємодія та висока конверсія мають ключове значення. Вибір такої групи дозволяє оцінити вплив різних підходів адаптивності та інтерактивності у реальному комерційному

контексті та отримати практично значущі результати. Збір даних передбачає кілька методів взаємопов'язаного тестування.

Перший − юзабіліті-тестування на мобільних та десктопних пристроях, що дозволяє безпосередньо спостерігати взаємодію користувачів з елементами інтерфейсу та фіксувати труднощі або затримки при виконанні цільових дій.

Другий − A/B-тестування різних варіантів сторінок, де один варіант реалізований із застосуванням підходу mobile-first та базової інтерактивності, а інший − із desktop-first та тією самою інтерактивністю або без неї. Для більш детального порівняння передбачено чотири експериментальні варіанти:

ВАРІАНТ A − mobile-first верстка з базовою інтерактивністю (анімація кнопок, підказки, модальні вікна);

ВАРІАНТ B − desktop-first верстка з такою ж інтерактивністю;

ВАРІАНТ C − mobile-first верстка без інтерактивних елементів;

ВАРІАНТ D − desktop-first верстка без інтерактивних елементів.

Технологічна реалізація експериментальних варіантів базується на використанні сучасних вебтехнологій: HTML5 та CSS3 з застосуванням медіа-запитів, Flexbox і Grid для адаптивної верстки та JavaScript для створення інтерактивних елементів інтерфейсу. Такий підхід забезпечує максимальну близькість експериментальних умов до реального середовища комерційних вебдодатків та дозволяє оцінити вплив інтерактивності та адаптивності на поведінку користувачів у різних контекстах використання. Очікується, що результати дослідження продемонструють значущу різницю у конверсії між експериментальними варіантами вебдодатків.

Найвищі показники конверсії прогнозуються для ВАРІАНТУ A, який поєднує підхід mobile-first з базовою інтерактивністю інтерфейсу, оскільки цей варіант оптимізований під мобільні пристрої, які зараз генерують більшу частку трафіку, та містить елементи, що підвищують залученість користувача, такі як анімовані кнопки, підказки та модальні вікна.

ВАРІАНТ B (desktop-first з інтерактивністю), ймовірно, покаже меншу конверсію на мобільних пристроях через менш оптимізовану структуру інтерфейсу, хоча на великих екранах його ефективність може бути порівнянною.

ВАРІАНТИ C і D, які не містять інтерактивних елементів, очікувано продемонструють нижчі результати конверсії, підкреслюючи значення інтерактивності для стимулювання користувацьких дій незалежно від обраного підходу адаптивності.

Інтерактивні елементи інтерфейсу передбачають посилення взаємодії користувачів із вебдодатком та підвищення конверсії, однак їх ефективність варіюється залежно від контексту адаптивності. У комбінації з mobile-first вони, ймовірно, сприяють більш вираженому зростанню конверсії, оскільки користувачі мобільних пристроїв швидко отримують доступ до основних функцій і отримують додаткову візуальну підказку, що спрощує виконання цільових дій. У desktop-first варіантах інтерактивність також впливає на конверсію, але її ефект може бути менш помітним через розподілення уваги на великому екрані та складнішу навігаційну структуру.

Водночас, дослідження дозволяє очікувати певні trade-off у виборі оптимальної комбінації адаптивності та інтерактивності. Наприклад, mobile-first підхід може вимагати більш складної верстки та підтримки для забезпечення коректного відображення на всіх типах пристроїв, що збільшує трудові витрати розробників. Інтерактивні елементи, хоча й підвищують конверсію, здатні уповільнювати завантаження сторінки, що негативно впливає на користувацький досвід, особливо при низькій швидкості інтернет-з'єднання. Таким чином, оптимізація вебдодатків потребує балансування між продуктивністю, інтерактивністю та адаптивністю, а результати цього дослідження дозволяють розробити практичні рекомендації для досягнення максимального ефекту при мінімальних витратах на підтримку та оптимізацію.

Результати дослідження дозволяють сформулювати низку практичних рекомендацій для дизайнерів та розробників комерційних вебдодатків.

По-перше, для платформ з високим мобільним трафіком, таких як e-commerce або сервіси швидкого замовлення, оптимально застосовувати підхід mobile-first у поєднанні з інтерактивними елементами − анімованими кнопками, модальними вікнами та підказками. Така комбінація забезпечує швидкий доступ користувачів до основних функцій, підвищує залученість і стимулює виконання цільових дій, що безпосередньо впливає на конверсію.

По-друге, для платформ із складними функціональними елементами, орієнтованих на десктопних користувачів, доцільно застосовувати desktop-first адаптивність із продуманою інтерактивністю, оскільки вона дозволяє ефективно організувати багаторівневу навігацію та забезпечує зручність при використанні великого екрану. У цьому випадку інтерактивні елементи слід застосовувати вибірково, щоб уникнути перевантаження користувача та уповільнення завантаження сторінки.

Підсумовуючи основні результати, можна зробити висновок, що комплексний підхід, який поєднує стратегічно обрану адаптивність з продуманою інтерактивністю, дозволяє оптимізувати конверсію та покращити користувацький досвід.

Ці висновки мають практичне значення для бізнесу, оскільки дозволяють визначити пріоритети при проектуванні вебдодатків та приймати рішення щодо оптимізації інтерфейсу залежно від цільової аудиторії та бізнес-моделі.

Список використаних джерел

1. Mobile First Design: актуальність та переваги для бізнесу. URL: https://wezom.com.ua/ua/blog/vsyo-chto-vam-nuzhno-znat-o-mobile-first-design-uzhe-sejchas (дата звернення: 21.11.25)

2. Adaptive web design. URL: https://en.wikipedia.org/wiki/Adaptive_web_design (дата звернення: 21.11.25)

3. An analysis of the impact of UX/UI improvements. URL: https://www.researchgate.net/publication/386383076_An_analysis_of_the_impact_of_UX_UI_improvements_and_web_accessibility_enhancements_on_user_experience_and_conversion_rates_in_e-commerce_platforms (дата звернення: 21.11.25)

4. Analyzing the Impact of Visual Elements in Website Design. URL: https://www.researchgate.net/publication/381618547_Analyzing_the_Impact_of_Visual_Elements_in_Website_Design_on_User_Experience_and_Interaction (дата звернення: 21.11.25)

5. Developing a website with interactive web design to increase. URL: https://coi.ua/en/blog/DesignCo/interactive-web-design-entertainment-and-user-engagement (дата звернення: 21.11.25)

6. The Art and Science of Button Design: Crafting Clickable. URL: https://www.designsystemscollective.com/the-art-and-science-of-button-design-crafting-clickable-engaging-and-high-converting-ui-elements-4694ad4fba2d (дата звернення: 21.11.25)

7. Pop-Ups Vs. Usability, Conversions And Bounce Rates. URL: https://medium.com/usabilitygeek/pop-ups-vs-usability-conversions-and-bounce-rates-22bf34658b93 (дата звернення: 21.11.25)

8. Designing Interactive Elements to Enhance User Experience. URL: https://www.fluer.com/blog/design/designing-interactive-elements-to-enhance-user-experience (дата звернення: 21.11.25)

Шишлін Владислав, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Слюсар Вадим

ЗАСТОСУВАННЯ МЕХАНІЗМУ RE-RANKING ДЛЯ ЗАВДАНЬ ПОШУКУ НА ОСНОВІ LLM У КОРПОРАТИВНИХ БАЗАХ ЗНАНЬ

До появи масового застосування буму генеративного штучного інтелекту (AI) традиційним підходом до пошуку у документах та корпоративних базах знань було застосування повнотекстових пошукових систем, таких як Apache Lucene, Elasticsearch і Solr, а також аналогічних механізмів у системах керування базами даних, зокрема Microsoft SQL Full-Text Search.

Ці інструменти забезпечували переважно примітивний пошук за ключовими словами та базові можливості індексування. Додаткове впровадження власних ML/AI-рішень вимагало значних ресурсів, було складним у розробці й підтримці, а також високовартісними, що робило такі системи недоступними для більшості організацій. Зазначені підходи залишаються працездатними, однак їхні можливості суттєво обмежені, що відповідно обмежує й ефективність систем, побудованих на їх основі.

Такі методи не здатні коректно обробляти складні або нечітко сформульовані запити, у яких необхідне розуміння семантики питання, його прихованого змісту чи контексту взаємодії, а не лише зіставлення окремих ключових слів. У випадках, коли результативність пошуку залежить від глибокого урахування контексту, класичні повнотекстові засоби демонструють низьку ефективність. Крім того, для традиційних пошукових підходів така варіативність мовного представлення є суттєвою перешкодою, що робить їх малопридатними для вирішення подібних задач.

Сьогодні реалізація подібних завдань стала можливою завдяки появі великих мовних моделей (LLM). Вони забезпечують підтримку з двох ключових напрямів:

1. Розуміння користувацьких запитів і їх формалізація. Мовні моделі здатні інтерпретувати запит користувача та перетворювати його у структурований формат, придатний для комп’ютерної обробки. Модель може коректно інтерпретувати навіть некоректно сформульовані, неповні або «хаотичні» запити й перетворювати їх на змістовне формулювання.
2. Формування змістовної відповіді на основі доступних даних та контексту. LLM здатні генерувати релевантні відповіді у текстовій або візуальній формі, враховуючи наявні дані та контекст попередньої взаємодії. Якість цих відповідей залежить від ряду факторів, включаючи якість даних, модель, а також спосіб інтеграції у систему.

Разом із тим LLM не виконують один критично важливий етап — пошук необхідних даних для формування відповіді. Хоча модель може містити певні знання, якщо була попередньо натренована на відповідних доменних даних або якщо запит не потребує спеціалізованої інформації, у більшості випадків для компаній, що працюють у вузьких чи специфічних сферах, цього недостатньо.

Зазвичай необхідні дані знаходяться у внутрішніх документах, базах знань або приватних репозиторіях, доступ до яких модель не має безпосередньо.

Між етапами пошуку та генерації відповіді існує додатковий рівень – retriever (ретривер). Це компонент, який здійснює пошук релевантної інформації у корпоративних базах даних, документах, CRM-системах та інших джерелах, після чого надає знайдений контекст великій мовній моделі.

Поширений термін RAG (Retrieval-Augmented Generation) буквально описує такий підхід: система поєднує LLM і ретривер для генерації відповідей на основі релевантних даних.

У реальних корпоративних середовищах часто виникає потреба у використанні декількох ретриверів, які виконують пошук у різних ізольованих інформаційних системах компанії. На завершальному етапі результати з усіх ретриверів необхідно узгодити та об’єднати – цей процес називають retriever assembling (ансамблювання ретриверів). Разом із тим підхід RAG має низку обмежень, зумовлених архітектурними особливостями LLM. Зокрема, мовні моделі здатні обробляти лише обмежений обсяг тексту в межах одного запиту, тому неможливо передати моделі всю доступну корпоративну інформацію для пошуку відповіді.

Одним зі способів подолання цієї проблеми є використання механізму re-ranking (переупорядкування). Моделі переупорядкування, навчені на текстах із конкретної предметної області, дозволяють ранжувати знайдені фрагменти та передавати LLM лише найбільш релевантний контекст.

Існує багато підходів до реалізації re-ranking; деякі з них наведені у (посилання). У виробничих системах зазвичай застосовують комбінацію декількох методів, оптимально підібраних під конкретний домен. Для виконання переупорядкування необхідно спочатку отримати дані. Ретривер може здійснювати пошук у різних джерелах: безпосередньо у корпоративній SQL-базі, листуванні електронної пошти чи наявних у компанії повнотекстових пошукових системах. Проте такі підходи характеризуються низькою точністю та недостатньою швидкістю.

На сьогодні найбільш ефективним способом зберігання інформації для RAG-пошуку є векторні бази даних або векторні сховища, що забезпечують високошвидкісний пошук за семантичною подібністю.

Список використаних джерел

1. Німецький стартап n8n залучив $180 млн для розвитку ШІ-агентів – оцінка зросла до $2,5 млрд. *Borgexpert*. URL: https://borgexpert.com/news/nimetskyj-startap-n8n-zaluchyv-180-mln-dlia-rozvytku-shi-ahentiv-otsinka-zrosla-do-2-5-mlrd

2. AI Assistant. *n8n*. https://docs.n8n.io/manage-cloud/ai-assistant/#tips-for-getting-the-most-out-of-the-assistant (дата звернення: 20.11.2025).

Шубка Максим, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Поночовний Юрій

РОЗРОБКА МОБІЛЬНОГО ДОДАТКА ПЕРСОНАЛЬНОГО АСИСТЕНТА ФЕРМЕРА НА ОСНОВІ ЛОКАЛЬНИХ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

У сучасних умовах цифровізації аграрного сектору України особливої актуальності набуває використання технологій штучного інтелекту для оптимізації прийняття рішень, автоматизації рутинних процесів та підвищення ефективності фермерських господарств. В умовах воєнного стану та економічних обмежень фермери потребують доступних інструментів, які працюють офлайн, не залежать від хмарних сервісів, забезпечують приватність даних та можуть функціонувати на недорогих мобільних пристроях. Саме тому перспективним напрямом є застосування локальних великих мовних моделей (LLM) для створення мобільного персонального асистента фермера.

Аналіз сучасних досліджень 2023–2025 років показує значний прогрес у розвитку компактних LLM, оптимізованих для мобільних пристроїв, завдяки архітектурам типу LLaMA [1], Mistral, Phi-3 [2] та Gemma. Низка робіт відзначає ефективність квантизації (4-bit, 8-bit) та використання ONNX Runtime, GGUF-формату та TensorRT, що дозволяє зменшити обчислювальні ресурси без значної втрати точності. Для агротехнологічних застосувань в літературі особливо підкреслюється важливість обробки доменно-специфічних знань – рекомендацій щодо рослинництва, догляду за ґрунтом, планування посівів, виявлення ризиків та роботи з сенсорними даними IoT.

Основний матеріал дослідження присвячено аналізу архітектур, придатних для створення мобільного асистента, та оцінці їх можливостей у контексті аграрної галузі. Розглянуто такі групи моделей.

Локальні LLM загального призначення (LLaMA 3, Mistral 7B, Phi-3 Mini/Small): забезпечують високий рівень розуміння мови, можливість донавчання (fine-tuning) та генерацію тексту. Переваги – якість відповідей і хороша адаптивність. Недоліки – потреба в оптимізації для мобільних CPU та обмежена швидкість на слабких пристроях.

Оптимізовані мобільні моделі (Gemma 2B [3], Phi-3 Mini 3.8B, TinyLLaMA):

Розроблені спеціально для використання на edge-пристроях. Підтримують роботу офлайн, мають низьке споживання пам’яті (1.5–4 ГБ RAM), забезпечують реальний час відповіді на смартфонах Snapdragon 700/800 серій. Ідеальні для мобільних додатків у польових умовах.

Гібридні підходи (онлайн + офлайн):

Дозволяють використовувати локальну модель для базових консультацій та хмарну модель для складних аналітичних запитів (прогнозування погоди, аналіз ринку). У сільській місцевості перевага надається повністю офлайн-архітектурам.

В рамках дослідження сформовано вимоги до системи персонального асистента фермера:

– повна офлайн-робота мобільного застосунку;

– можливість доменно-орієнтованого донавчання моделі (інструкції, довідники, нормативи, рекомендації Мінагрополітики);

– підтримка української мови;

– низьке споживання ресурсів смартфону;

– модульність (поради щодо рослинництва, хімічного захисту, зрошення, журнал робіт, інтеграція з датчиками IoT);

– забезпечення приватності та безпеки користувацьких даних.

Порівняльний аналіз показує, що оптимальними для реалізації мобільного персонального асистента є моделі Phi-3 Mini, Gemma 2B, або Mistral 7B (квантизована 4bit [4]), які можуть працювати локально на сучасних смартфонах. Подальші етапи роботи передбачають донавчання вибраної LLM на аграрному корпусі, інтеграцію моделі у мобільний застосунок (Flutter/React Native), а також розробку інтелектуальних модулів рекомендацій та планування фермерських процесів.

Список використаних джерел

1. LLaMA 3: Open and Efficient Foundation Language Models. Meta AI, 2024. URL: https://arxiv.org/pdf/2302.13971

2. Phi-3 Technical Report: A Highly Capable Language Model for Mobile and Edge. Microsoft Research, 2024. URL: https://arxiv.org/pdf/2404.14219

3. Gemma: Open Models for Responsible AI. Google DeepMind, 2024. URL: https://arxiv.org/pdf/2403.08295

4. QLoRA: Efficient Finetuning of Quantized LLMs. Proceedings of NeurIPS, 2023. URL: https://arxiv.org/abs/2305.14314

Євгеній Юхименко, здобувач вищої освіти СВО «Бакалавр»,
спеціальність «Інформаційні системи та технології»,
Науковий керівник – д.т.н., професор Юрій Поночовний

АНАЛІЗ ТА ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ТЕХНОЛОГІЙ ТЕКСТ-В-МОВУ (TTS) ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ ОЗВУЧУВАННЯ ТЕКСТІВ

У сучасному цифровому світі технології перетворення тексту в мову (Text-to-Speech, TTS) відіграють ключову роль у забезпеченні доступності інформації, особливо для людей з вадами зору, у освітніх додатках, навігаційних системах та віртуальних асистентах. Актуальність проблеми розробки мобільного додатку для озвучування текстів зросла через швидке поширення смартфонів в Україні та світі, де багато людей потребують інструментів доступності (за даними WHO). Особливо важливим є підтримка української мови, яка належить до низькоресурсних мов з обмеженими даними для навчання нейронних моделей. На тлі війни в Україні TTS-додатки допомагають у швидкому озвучуванні новин, документів та освітніх матеріалів для вимушених переселенців і людей з обмеженими можливостями. Розробка такого мобільного додатку вимагає вибору оптимальної TTS-технології, яка забезпечить природність голосу, офлайн-роботу (критично для регіонів з нестабільним інтернетом), низьке споживання ресурсів батареї та процесора, а також якісну підтримку української мови. Початковий етап проєкту передбачає аналіз і порівняння існуючих рішень, щоб обрати технологічний стек для подальшого проєктування архітектури.

Аналіз літератури та джерел показує швидкий розвиток TTS-технологій у 2023-2025 роках завдяки глибокому навчанню. Згідно з оглядом нейронного синтезу мовлення [1], сучасні моделі на базі Tacotron 2, FastSpeech та VITS досягли високої природності завдяки end-to-end архітектурам. У дослідженні [2] порівняння TTS-моделей і вокодерів підкреслюється перевага нейронних вокодерів (HiFi-GAN, WaveGlow) над традиційними. Для мобільних платформ (Android/iOS) актуальними є офлайн-рішення, оскільки хмарні сервіси (Google Cloud TTS, Microsoft Azure) вимагають постійного з'єднання та можуть бути платними [3]. Відкритий проєкт Coqui TTS (форк Mozilla TTS) виділяється як гнучкий інструмент для кастомізації, включаючи fine-tuning під українську мову. Огляд відкритих TTS-рушіїв 2025 року відзначає Coqui TTS, Piper TTS та Silero як найкращі для мобільних пристроїв завдяки легкості та швидкості [4].

Основний матеріал порівняння зосереджено на ключових TTS-технологіях, придатних для мобільної розробки. Розглянемо чотири основні категорії: хмарні сервіси Google, Microsoft, Amazon та відкриті/офлайн-рішення (Coqui TTS, RHVoice, Silero) (табл. 1). Google Cloud Text-to-Speech (WaveNet/Neural2): Використовує глибокі нейронні мережі для генерації природного голосу. Підтримка української мови — повна (нейронні голоси з 2023 року). Переваги: висока якість (MOS >4.5), SSML для керування інтонацією, інтеграція з Android (android.speech.tts). Недоліки для мобільного додатку: обов'язковий інтернет, платність після квоти (від $4 за 1 млн

символів), високе споживання трафіку. Латентність – 200–500 мс. Ідеально для онлайн-режиму, але не для офлайн.

Таблиця 1 – Порівняння характеристик систем TTS (за критеріями для мобільного додатку)

Технологія	Офлайн	Підтримка укр.	Якість (MOS)	Швидкість	Вартість	Розмір моделі
Google Cloud TTS	Ні	Повна	4.6–4.8	Висока	Платно	–
Microsoft Azure TTS	Ні	Повна	4.7–4.9	Висока	Платно	–
Amazon Polly	Ні	Часткова	4.3–4.5	Середня	Платно	–
Coqui TTS / XTTS	Так	Через fine-tuning	4.4–4.7	Висока	Безкоштовно	200–500 МБ
RHVoice	Так	Відмінна	4.2–4.4	Висока	Безкоштовно	<100 МБ

Microsoft Azure Cognitive Services Speech (Neural TTS): Одна з найкращих за природністю (MOS до 4.8). Підтримка української – з 2021 року, понад 10 голосів (чоловічі/жіночі). Переваги: кастомні голоси, емоційний контроль (SSML), SDK для Android/iOS.

Недоліки: хмарний (потрібен інтернет і ключ API), вартість ~$16 за 1 млн символів, вища латентність у пікові години. У 2025 році додано більше емоційних стилів, але офлайн-режим обмежений (лише через контейнери для enterprise).

Amazon Polly: Neural TTS з хорошою якістю. Українська мова — часткова підтримка (стандартні голоси, не завжди природні). Переваги: низька ціна ($4 за 1 млн), streaming для реального часу. Недоліки: гірша якість для слов'янських мов порівняно з Google/Microsoft, немає повноцінного офлайн на мобільних.

Відкриті офлайн-рішення:

Coqui TTS / XTTS-v2: End-to-end модель на базі VITS. Підтримка української – через fine-tuning (моделі на HuggingFace, наприклад styletts2-ukrainian). Переваги: повністю офлайн, швидкість на CPU/GPU мобільних (до 10x real-time на Snapdragon), voice cloning з 3–5 сек аудіо, безкоштовно. Якість – висока для низькоресурсних мов після донавчання. Інтеграція: Python/Kotlin через ONNX або TensorFlow Lite.

RHVoice: Спеціально розроблений для слов'янських мов, включаючи українську (висока якість, MOS ~4.2). Повністю офлайн, легкий (~50 МБ), працює на Android без інтернету.

Silero TTS: Російсько-українська модель від Sber, офлайн, швидка, хороша природність для української.

Аналіз показав, що для мобільного додатку озвучування текстів оптимальними є офлайн-рішення на базі Coqui TTS або RHVoice, оскільки вони забезпечують незалежність від інтернету, низьке споживання ресурсів і хорошу адаптацію до української мови. Хмарні сервіси (Google, Microsoft) доцільні як гібридний варіант для преміум-якості в онлайн-режимі.

На наступному етапі проєкту рекомендовано провести fine-tuning Coqui XTTS-v2 на українському корпусі (наприклад, Common Voice) та інтеграцію через Flutter/React Native для кросплатформності. Це дозволить створити доступний, безкоштовний додаток з високою природністю озвучування.

Список використаних джерел

1. Neural Speech Synthesis. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2023/04/TTS.ijcai21-642be55185047.pdf.

2. P.V. Reddy et al. A Comparative Study of Text-to-Speech (TTS) Models and Vocoder Combinations for High-Quality Synthesized Speech. *2023 7th International Conference on Electronics, Communication and Aerospace Technology* (ICECA), Coimbatore, India, 22–24 November 2023. 2023. URL: https://doi.org/10.1109/iceca58529.2023.10395349.

3. The Best Open-Source Text-to-Speech (TTS) Tools in 2025. URL: https://medium.com/@shouke.wei/the-best-open-source-text-to-speech-tts-tools-in-2025-290f92f759d3.

4. Kononenko N. Development of an Enhanced Ukrainian Text-to-Speech System; Supervisor: URL: https://hdl.handle.net/20.500.14570/5524.

Руслан Житник, здобувач вищої освіти СВО «Магістр»,
спеціальність «Інформаційні системи та технології»
Науковий керівник – д.т.н., професор Юрій Поночовний

РОЗРОБЛЕННЯ ВЕБСАЙТУ ДЛЯ СИСТЕМИ ТЕЛЕМЕТРІЇ ТА ХРОНОМЕТРАЖУ В ДРЕГРЕЙСИНГУ

У сучасному дрегрейсингу, де змагання вимагають точного моніторингу параметрів, таких як швидкість, прискорення та час проходження дистанції, вебсайти стають ключовим інструментом для візуалізації та аналізу телеметричних даних. Традиційні системи телеметрії часто обмежені локальним зберіганням даних, що ускладнює реальний час доступу для команд, суддів та глядачів [1]. Актуальність розробки вебсайту зумовлена потребою в централізованій платформі, яка забезпечує інтеграцію з апаратними пристроями, реальний час моніторинг та аналіз результатів заїздів. Це особливо важливо для аматорських команд в Україні, де бюджетні рішення можуть підвищити конкурентоспроможність без значних витрат. Метою є створення вебсайту на архітектурі OpenServer (nginx + PHP + MySQL) для відображення інформації про змагання, що забезпечує live-моніторинг, пост-аналіз та управління даними. Завдання: обґрунтування архітектури, проєктування бази даних, реалізація backend та frontend, інтеграція з апаратною частиною.

Обґрунтування архітектури вебсайту базується на клієнт-серверній моделі з елементами IoT, де сервер обробляє дані з мікроконтролерів (наприклад, ESP32) через HTTP POST-запити у форматі JSON [2]. Вибір OpenServer зумовлений його стабільністю та сумісністю з PHP для динамічної обробки, nginx для високої продуктивності при великій кількості з'єднань та MySQL для реляційного зберігання даних. Це дозволяє мінімізувати латентність (менше 100 мс) при передачі телеметрії, такої як швидкість, прискорення (G-force) та RPM.

Проєктування структури бази даних є ключовим етапом. Модель включає чотири сутності: users (для аутентифікації ролей – пілот, команда, суддя), races (для зберігання даних заїздів: ET, RT, max_speed), telemetry (для телеметричних метрик: timestamp, speed, accel, rpm, temp з індексацією для швидких запитів) та logs (для подій системи). ER-діаграма ілюструє зв'язки з зовнішніми ключами для референційної цілісності [3]. Backend реалізований на PHP 8.0+ з об'єктно-орієнтованим підходом: клас Database використовує патерн Singleton для підключення до БД та підготовлені запити для захисту від SQL-ін'єкцій. Скрипт receive_data.php приймає JSON від апаратного модуля, валідує поля та вставляє дані в таблицю telemetry. Для реального часу оновлення впроваджено WebSockets (бібліотека Ratchet), що дозволяє динамічну візуалізацію без перезавантаження сторінки. Модуль reports.php генерує звіти з агрегатними запитами, а export_report.php – PDF-документи за допомогою TCPDF.

Frontend побудований на HTML5, CSS3, JavaScript з бібліотеками Bootstrap 5 для адаптивного дизайну, Chart.js для графіків (наприклад, залежність швидкості від часу) та DataTables для таблиць. Головна сторінка (index.php) – дашборд з ключовими показниками, графіками останнього заїзду та таблицею подій. Сторінка telemetry.php відображає детальні графіки, а live.php – реальний час моніторинг

через WebSockets. Архітектура MVC спрощує: Model – SQL-запити, View – шаблони з PHP-вставками, Controller – скрипти для обробки запитів [4]. Безпека забезпечена HTTPS, JWT-токенами для API, хешуванням паролів (bcrypt) та перевіркою аутентифікації.

Інтеграція з апаратною частиною здійснюється через REST API: мікроконтролер надсилає дані, сервер їх обробляє та візуалізує. Це розширює можливості попередніх розробок, перетворюючи локальні дані на централізовану платформу [5].

Розроблений вебсайт забезпечує ефективну візуалізацію телеметрії в дрегрейсингу, з live-моніторингом, пост-аналізом та багатокористувацьким доступом. Новизна полягає в інтеграції IoT з вебтехнологіями для реального часу обробки, що підвищує доступність для аматорських команд. Практичне значення: зниження витрат на аналіз даних, підвищення безпеки змагань. Подальші дослідження: контейнеризація з Docker для масштабування та інтеграція з мобільними додатками.

Список використаних джерел

1. Telemetry Data Visualization Application - LightningChart. URL: https://lightningchart.com/blog/racing-telemetry-application/ (дата звернення: 17.11.2025).

2. How to Use PHP for IoT Applications - EmpowerCodes Technologies. URL: https://empowercodes.com/articles/how-to-use-php-for-iot-applications (дата звернення: 17.11.2025).

3. PART 1 - Send Arduino Data to the Web (PHP/MySQL/D3.js). URL: https://www.instructables.com/PART-1-Send-Arduino-data-to-the-Web-PHP-MySQL-D3js/ (дата звернення: 17.11.2025).

4. Web Application Architecture: The Latest Guide for 2025. URL: https://www.clarity-ventures.com/how-to-guides/web-application-architecture (дата звернення: 17.11.2025).

5. Web Application Architecture: Complete Guide 2025 - Carmatec. URL: https://www.carmatec.com/blog/web-application-architecture-complete-guide/ (дата звернення: 17.11.2025).