Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему.
«Розробка методів і програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту»

Виконав. студент II курсу
групи ____3ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Тоха Вадим Валентинович

Керівник. к.т.н., доц. Хошаба О.М
«15» грудня 2023 р.

Опонент. к.т.н., доц. каф. ЗІ Мадіновський В.І.
«15» грудня 2023 р.

Допущено до захисту

Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«15» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Тосі Вадиму Валентиновичу

1. Тема роботи – розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту
Керівник роботи. Хошаба Олександр Мирославович, доц. каф. ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 р.

3. Вихідні дані до роботи.
Процесор (CPU). Багатоядерний процесор з високою швидкістю (наприклад, Intel Core i9 або AMD Ryzen 9).Графічний процесор (GPU). Велика кількість ядер CUDA або Tensor Cores (наприклад, NVIDIA GeForce RTX 30-серії або AMD Radeon RX 6000-серії).Оперативна пам'ять (RAM). (наприклад, 32GB або більше) для ефективної обробки великих обсягів даних. Великий обсяг внутрішньої та зовнішньої пам'яті (наприклад, 1TB SSD або NVMe для збереження моделей, даних та результатів). Швидкий SSD або NVMe з високою читанням/записом для ефективного зберігання та витягування даних.

4. Зміст текстової частини. вступ, обгрунтування вибору методі розробки та постановка задачі дослідження, розробка метоів та програмних засобів для реалізації додатку, розробка програмних компонентів android-

додатку для зчитування візуалізації надвеликого об'єму даних, економічна частина, висновки, список використаних джерел, додатки, графічна частина

5. Перелік ілюстративного матеріалу, порівняльні таблиці аналогів, порівняльні таблиці програмних середовищ, представлення програмної частини, блок -схеми, uml діаграми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент каф. ПЗ	19.09.2023	01.12.2023
5	Причепа І.В., к.е.н., доцент каф. ЕПВМ	30.11.2023	01.12.2023

7. Дата видачі завдання 19 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз систем для аналізу і генерації	20.09. 2022 – 10.10.2023-	вик
2	Розробка методів зчитування великих обсягів даних	10.10.2023 – 30.10.2023	вик
3	Розробка методів аналізу великих обсягів даних	01.11.2023 – 20.11.2023	вик
4	Розробка методів візуалізації на основі великих обсягів даних.	20.11.2023- 30.11.2023	вик
5.	Економічна частина	30.11.2023- 1.12.2021	вик

Студент (підпис) **Тоха В.В** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) **Хошаба О. М.** (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Тоха В.В розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця. ВНТУ, 2023. 128 с.

На укр. мові. Бібліогр.. 33 назва; рис.4; табл. 12.

У магістерській кваліфікаційній роботі розроблено методи та програмні засоби для зчитування надвеликих масивів даних для їх аналізу і візуалізації.

Проведено аналіз існуючих рішень і засобів для організації самостійної роботи студента.

Розроблено алгоритми зчитування, аналізу і візуалізації даних. Також попередньо було розроблено архітектурну модель додатку. В результаті розробки було створено розподілену програмну систему. Тестування програмної системи показало, що вона працює коректно й відповідає поставленим завданням.

Результати магістерської кваліфікаційної роботи можна використати для покращення ефективності обробки надвеликих об’ємів даних, їх аналіз і візуалізація.

Ключові слова. Android, штучний інтелект, промт, інженерія.

ABSTRACT

UDC. 004.912.032.26

Toha V.V. Development of methods and software tools for processing, analyzing and visualizing ultra-large data sets based on artificial intelligence. Master's qualification work in specialty 121 - software engineering, educational program - software engineering. Vinnytsia. VNTU, 2023. 128 c.

In Ukrainian. Bibliography. 33 titles; figures. 4; tables. 12.

The master's thesis developed methods and software tools for reading ultra-large data sets for their analysis and visualization.

An analysis of existing solutions and tools for organizing independent student work was conducted.Algorithms for reading, analyzing, and visualizing data were developed. An architectural model of the application was also previously developed.

As a result of the development, a distributed software system was created. Testing of the software system showed that it works correctly and meets the tasks set.The results of the master's qualification work can be used to improve the efficiency of processing ultra-large amounts of data, their analysis and visualization.

Keywords. Android, artificial intelligence, industry, engineering.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, що характеризується стрімким розвитком інформаційних технологій, обробка великих даних (Big Data) є однією з найважливіших проблем. Великими даними прийнято вважати масиви даних, які занадто великі або складні для обробки традиційними методами. Вони можуть бути структурованими, напівструктурованими або неструктурованими, і можуть надходити з різних джерел, таких як соціальні мережі, системи спостереження, пристрої Інтернету речей (IoT) та інші. Штучний інтелект (AI) є одним з найефективніших методів обробки великих даних. AI-технології дозволяють отримувати цінну інформацію з величезних наборів даних, що може бути використано для прийняття рішень, виявлення тенденцій, прогнозування та інших цілей.

Тема «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» є актуальною та перспективною. Вона відповідає сучасним тенденціям розвитку інформаційних технологій і має важливе практичне значення. Актуальність теми обґрунтовується наступними факторами. Зростання обсягу даних, що генеруються в сучасному світі. Необхідність ефективної обробки великих даних для прийняття рішень, виявлення тенденцій, прогнозування та інших цілей. Потенційні можливості AI-технологій для обробки великих даних.

Мета і задачі дослідження.

Метою роботи є підвищення автоматизації аналізу даних при створенні візуалізації. За рахунок використання методів машинного навчання з урахування індивідуальних потреб користувачів.

Основними задачами дослідження є.

- аналіз сучасних методів і підходів до створення систем для аналізу надвеликих масивів даних і їз візуалізації;
- розробка та вдосконалення навчання моделей з рекомендаційними алгоритмами, які враховують контентні та колаборативні підходи, аналіз та порівняння різних моделей за їхньою ефективністю;

- удосконалення алгоритму аналізу і візуалізації з урахування результатів моделей та їхньою ефективністю;
- розробка програмного засобу, який базується на удосконалених моделях та забезпечує аналізу надвеликих масивів даних та їх візуалізацію;
- проведення тестування розробленого програмного засобу та аналіз масивів даних з подальшою візуалізаціїєю.

Об'єкт дослідження - процес аналізу і візуального представлення результату.

Предмет дослідження. – методи і засобі для аналізу надвеликих масивів даних та їх подальших візуалізацій.

Методи дослідження. проблеми оптимізації великих мережевих структур та їхньої ефективної візуалізації. Основні методи дослідження, які застосовувались у цьому контексті, включають теорію графів, аналіз складних систем, математичну статистику, а також методи обробки сигналів. Використано інноваційні підходи до аналізу та оптимізації великих обсягів даних, такі як методи глибинного навчання та алгоритми машинного навчання з підкріпленням. Лінійна алгебра використовувалася для вирішення систем рівнянь, що виникали в процесі обчислень, та для побудови математичних моделей розглянутих мережевих структур.

Новизна роботи.

1. Вперше запропонована модель швидкодії аналізу надвеликих масивів даних та їх візуалізації, яка включає в собі поєднанням класичних методів рекомендації з машинним навчанням та побудовою структури подачі відповіді, що дозволило сформувати більш ефективні і точні персоналізовані рекомендації на основі попередніх даних аналізу і візуалізації даних.

2. Вперше розроблено поєднання двох моделей для візуалізації таблиць TabNet і BERT (Bidirectional Encoder Representations from Transformers), особивість поєднання яких полягає в можливості виконання одночасних дій і тим самим зменшення часу дії. Моделі можна використовувати для в чат ботах або програмних асистентах для швидкодії роботи.

3. Вперше розроблено поєднання двох моделей для візуалізації презентацій та інфографіки CLIP (Contrastive Language–Image Pretraining) і DALL-E (Diverse All-purpose LImagE), особивість поєднання яких полягає в можливості поділу дій для виконання і комбінації їх під час цього процесу. Моделі можна використовувати для в чат ботах або програмних асистентах для швидкодії роботи.

Практичне значення одержаних результатів. На основі запропонованих методів розроблено програмний засіб у вигляді додатку на телефон та веб-сервісу, який використовує удосконалені методи аналізу даних та їх візуалізацію на основі машинного навчання з відповідною зазначеною моделлю на основі реальних даних. Було розроблено алгоритм, який дає змогу при поєднанні двох технологій, оптимізувати час роботи зчитування даних, за рахунок чого і оптимізувати час видачі необхідного результату.

Особистий внесок здобувача.

Усі наукові результати, викладені у магістерській кваліфікованій робота, отримані автором особисто.

Апробація матеріалів.

Основні положення магістерської кваліфікованої роботи доповідались та обговорювались на XII міжнародна наукова конференція інноваційна наука: пошук відповідей на виклики сучасності- 2023»

1 РОЗДІЛ. ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем аналізу і візуалізації даних

У сучасному світі, що характеризується стрімким зростанням обсягу даних, системи аналізу і візуалізації даних (ДАВ)[1] є одними з найважливіших інструментів для обробки та розуміння інформації. Вони дозволяють отримувати цінні знання з великих наборів даних, які можуть бути використані для прийняття рішень, виявлення тенденцій, прогнозування та інших цілей.

Станом на 2023 рік, системи ДАВ знаходяться на стадії активного розвитку. Це пов'язано з наступними факторами. Зростання обсягу даних, що генеруються в сучасному світі. Розвиток штучного інтелекту, який дозволяє розробляти нові ефективні методи аналізу та візуалізації даних. Зростання попиту на системи ДАВ з боку бізнесу, науки, уряду та інших галузей. Основні тенденції розвитку.

Зростання масштабу систем ДАВ. Системи ДАВ стають все більш масштабними, щоб обробляти все більші набори даних. Розвиток штучного інтелекту в системах ДАВ. Штучний інтелект[1] використовується для розробки нових ефективних методів аналізу та візуалізації даних. Інтеграція систем ДАВ з іншими системами. Системи ДАВ все частіше інтегруються з іншими системами, такими як системи управління бізнесом (ERP), системи управління знаннями (KMS) та інші.

Системи ДАВ мають високу практичну цінність. Вони можуть бути використані для вирішення широкого кола завдань в різних сферах діяльності, таких як. Бізнес. прийняття рішень, виявлення тенденцій, прогнозування продажів, управління ризиками та інші. Наука. дослідження, розробка нових продуктів і послуг, виявлення нових закономірностей у природі та інші. Уряд. управління, прийняття рішень, забезпечення безпеки та інші.

Системи ДАВ є важливим інструментом для обробки та розуміння інформації. Вони знаходяться на стадії активного розвитку, і їх практична

цінність постійно зростає. Зростання популярності відкритих систем ДАВ. Відкриті системи ДАВ стають все більш популярними, оскільки вони дозволяють користувачам самостійно розробляти і налаштовувати рішення. Мобільні системи ДАВ дозволяють користувачам отримувати доступ до даних і аналізувати їх на мобільних пристроях. Зростання попиту на системи ДАВ для аналізу даних у реальному часі. Системи ДАВ для аналізу даних у реальному часі дозволяють користувачам отримувати доступ до даних і аналізувати їх в режимі реального часу.

Поряд з численними перевагами, системи ДАВ також мають ряд викликів і проблем, які необхідно вирішувати. До них відносяться. Вартість систем ДАВ. Системи ДАВ можуть бути досить дорогими. Складність використання систем ДАВ. Системи ДАВ можуть бути складними у використанні, що може ускладнювати їх впровадження.

Якість даних може впливати на точність і ефективність систем ДАВ. Підсумовуючи, можна сказати, що системи ДАВ є важливим інструментом для обробки та розуміння інформації. Вони знаходяться на стадії активного розвитку, і їх практична цінність постійно зростає. Однак системи ДАВ також мають ряд викликів і проблем, які необхідно вирішувати.

Одним із самих популярних способів аналізу і візуалізації даних є базові нейромережі для генерації фото, або ж чат-боти.

Глибоко навчена модель природної мови GPT-3.5[2], створена лабораторією штучного інтелекту OpenAI, обладнана інтерфейсом, що уможливлює взаємодію з нею шляхом текстового введення. Оперуючи в рамках концептуального віртуального чату, ця інтелектуальна система взаємодіє з користувачами шляхом обробки та аналізу текстових запитань та видає відповіді в реальному часі.

Принцип роботи цієї технологічної досягнення базується на архітектурі глибокого навчання, а саме рекурентних нейронних мереж. Збагачена значущим обсягом текстових даних, модель володіє здатністю розпізнавати

семантичні зв'язки та контекст, вирізняючись своєю здатністю адаптуватися до різноманітних виразів та стилів комунікації.

Ця інноваційна система призначена для надання користувачам не лише відповідей на їх запитання, але й для взаємодії на рівні природної мови, намагаючись надати інформацію, допомогу та рекомендації відповідно до висунутих запитань. Інтерфейс відображається у вигляді текстового чату, але його функціональність глибоко вкорінена в складність мовового розуміння та генерації тексту, роблячи його важливим інструментом у вирішенні різноманітних завдань, пов'язаних з обробкою природної мови[2].

У сучасному світі, де кожен аспект нашого життя досліджується та аналізується, роль швидкого аналізу надвеликих об'ємів даних стає критичною для розуміння та прийняття важливих рішень. Використання технологій, які дозволяють ефективно обробляти величезні масиви інформації, є ключем до успіху в багатьох галузях, починаючи від науки і закінчуючи бізнесом.

Однією з важливих переваг використання швидкого аналізу даних є здатність отримати цінну інформацію в реальному часі. У ситуаціях, коли кожна секунда має значення, швидкий аналіз надвеликих об'ємів даних дозволяє оперативно реагувати на зміни, а також передбачати майбутні тенденції. Це особливо важливо в галузях фінансів, медицини та економіки, де точні та актуальні дані визначають успішність стратегій та прийняття рішень.

Ще одним аспект важливості аналізу надвеликих об'ємів даних є їх візуалізація в зручному для розуміння вигляді. Графічне представлення інформації відкриває нові можливості для користувачів у сприйнятті складних зв'язків та закономірностей. Від діаграм та графіків до інтерактивних візуалізацій, цей підхід дозволяє зробити аналіз доступним та зрозумілим для широкого кола користувачів. Забезпечення візуальної лаконічності та зрозумілості великих обсягів інформації стає ключем до ефективного використання отриманих даних.

У цьому контексті, розвиток технологій для аналізу та візуалізації надвеликих об'ємів даних є не лише технічним викликом, але і важливим

етапом вдосконалення нашого розуміння світу. Застосування цих технологій в різних галузях відкриває нові можливості для інновацій, досліджень та вирішення складних завдань, що стають перед сучасним суспільством.

В сучасному віці інформаційних технологій, важко переоцінити роль, яку відіграє аналіз та візуалізація даних за допомогою нейромереж. Нейромережі, як ключовий інструмент штучного інтелекту, відкривають нові перспективи у розумінні та інтерпретації величезних об'ємів інформації. Їх потужність у сфері аналізу даних виявляється не лише в швидкості, але й в здатності виявляти глибокі зв'язки та складні закономірності, які можуть залишитися непоміченими для традиційних методів.

Нейромережі взаємодіють із величезними обсягами даних і забезпечують їх ефективний аналіз. За допомогою глибокого навчання, ці мережі можуть автоматично виявляти та класифікувати патерни у наборах даних, що робить їх особливо корисними в областях медицини, науки, фінансів та багатьох інших.

Однак справжня сила нейромереж виявляється в їх здатності до візуалізації інформації. Вони можуть створювати вражаючі візуальні репрезентації складних даних, що сприяють зрозумінню та прийняттю рішень. Ця властивість особливо важлива у відкритті нових зв'язків та встановленні тенденцій, які можуть бути важко виявити за допомогою звичайних методів[2].

Нейромережі не лише оптимізують процес аналізу та візуалізації даних, але і створюють новий рівень розуміння складних інформаційних структур. Вони перетворюють навчання машин на мистецтво, забезпечуючи нас чутливішими до суттєвих аспектів даних і роблячи важливий крок у напрямку нового, інтелектуального розуміння світу навколо нас.

У сучасному науковому та технологічному контексті, перспективи застосування нейромереж для аналізу та візуалізації даних надають нам непередбачувану можливість глибшого розуміння складних інформаційних

структур та здійснення перетворень, які мали б значущий вплив на різні сфери нашого суспільства.

Нейромережі, які базуються на засадах штучного інтелекту та глибокого навчання, відкривають безмежні перспективи для аналізу великих обсягів даних. Їхні здатності до автоматичного виявлення складних патернів та залежностей дозволяють розкрити нові аспекти в дослідженні, медицині, фінансах і багатьох інших галузях. Мережі глибокого навчання, як складова штучного інтелекту, забезпечують надзвичайну продуктивність у роботі з великим обсягом даних та розв'язанні завдань аналітики.

Однак справжнім проривом стає вміння нейромереж ефективно візуалізувати отриману інформацію. Завдяки їхній здатності створювати динамічні та інтерактивні візуалізації, ми отримуємо не просто статичні зображення, але і засоби для взаємодії з даними у реальному часі. Це стає ключем до більш глибокого розуміння складних взаємозв'язків та виявлення тонких деталей, що можуть впливати на прийняття стратегічних рішень.

Перспективи використання нейромереж для аналізу та візуалізації даних безмежно широкі. Ці технології не лише полегшать процес обробки інформації, але й революціонізують наше розуміння світу. Спільно з ростом потужності обчислювальних систем та постійним розвитком методів глибокого навчання, ми вперше отримуємо здатність проникнути в найбільші таємниці складних даних і розкрити нові горизонти для наукового дослідження та розвитку суспільства.

На сучасному етапі розвитку інформаційних технологій, якими різманітними, можна визначити світ нейромереж. Велика кількість цих складних систем, що моделюють роботу людського мозку, активно створюється в лабораторіях і дослідницьких центрах по всьому світу. Вони виступають не тільки як інструменти для аналізу даних, але і як істинні архітекти інтелекту, що здатні навчатися та візуалізувати інформацію зі складних масивів даних.

Нейромережі[3] — це складні алгоритми, побудовані за аналогією зі структурою людського мозку. З їхньою допомогою ми можемо досягти неймовірної точності у вирішенні завдань аналізу та візуалізації даних. Ініційовані природною потребою вивчення імплицитних зв'язків та патернів у великих наборах інформації, нейромережі розвиваються за допомогою глибокого навчання.

Процес навчання нейромереж схожий на еволюцію штучного інтелекту: вони адаптуються до нових даних, коригують власні ваги і змінюють структуру для підвищення ефективності. Цей неперервний ітеративний процес, схожий на власне навчання, відбувається віртуозно, дозволяючи нейромережам виявляти та аналізувати слабко виражені зв'язки в даних. Вони "розуміють" не лише самі дані, але й контекст, в якому вони існують.

Однією з ключових особливостей сучасних нейромереж є їхня здатність до візуалізації даних у неймовірно простий та доступний спосіб. Вони створюють вражаючі графіки, діаграми та тривимірні моделі, які допомагають людині краще розуміти складні взаємозв'язки. Ця здатність робить нейромережі не лише інструментом для фахівців в галузі дослідження та аналізу, але і потужним засобом комунікації для широкого кола користувачів, включаючи тих, хто не має глибоких знань у сфері аналітики[3].

Усе це робить нейромережі не тільки важливим інструментом у світі дослідження та технологій, але і новою парадигмою розуміння інформації. Їхня роль у візуалізації та аналізі даних, враховуючи їхню швидкодію та глибокий інтелектуальний потенціал, має неоціненне значення для подальшого розвитку сучасного суспільства.

Також наведемо приклад нейромережі для генерації зображень, яка дає змогу за допомогою запиту сформувати ілюстрації, фото, графічний креатив або навіть презентацію.

Інтерфейс Recraft визначається своєрідністю та лаконічністю, намагаючись максимально спростити взаємодію користувача з системою.

Структурованість та зрозумілість елементів інтерфейсу сприяють ефективному використанню всіх можливостей платформи.

Щодо принципу роботи, Recraft використовує передові методи нейронних мереж для обробки інформації та генерації контенту. Модель глибокого навчання, вбудована в основу цієї системи, володіє здатністю адаптуватися до різноманітних видів текстів та завдань, завдяки чому можливий широкий спектр застосувань, від обробки природної мови до генерації контенту.

Зазначимо, що в режимі спостерігача система Recraft виявляється ефективним інструментом для обробки та розуміння текстової інформації, а його інтерфейс покликаний забезпечити користувачам максимальний комфорт при взаємодії з цією передовою нейромережею.

Крім того є системи, які можуть аналізувати цілі книги або документи, але візуалізовувати дані вони не спроможні, тільки аналізувати.

З великою зацікавленістю вивчаючи інноваційний інструмент, яким є нейромережа Claude, можна визначити його вражаючий потенціал у сфері аналізу великих обсягів даних. Цей продукт визначається своєрідністю підходу до обробки та розуміння інформації, що сприяє взаємодії користувачів з текстовим контентом на новому рівні.

Інтерфейс Claude, заснований на наукових принципах, надає користувачам зручний доступ до потужних можливостей системи. Зокрема, чіткі та інтуїтивно зрозумілі елементи інтерфейсу роблять взаємодію з великими обсягами даних зручною та ефективною.

Принцип роботи Claude базується на використанні передових нейронних мереж для аналізу текстової інформації. Його алгоритми здатні автоматизовано визначати патерни, витяги та висновки з великих обсягів даних, забезпечуючи швидку та точну обробку інформації.

Важливим аспектом взаємодії з цією нейромережею є спрощена можливість взаємодії з аналітичними результатами та отримання корисної інформації. Це робить Claude потужним інструментом для тих, хто шукає засіб

ефективно аналізувати та зрозуміти великі обсяги даних, надаючи певний ступінь автоматизації та прискорюючи процеси прийняття рішень.

Ще є комбінований тим сервісів, які по запиту генерують і зображення і інфографіку і презентацію, але аналіз даних відбувається через малесеньких чат і без великих обсягів інформації.

Описуючи нейромережу Gamma з позиції спостерігача, варто відзначити вражаючі можливості цього інструменту для аналізу великих обсягів даних та створення презентацій. Його науково-публіцистичний інтерфейс вражає своєю зручністю та інноваційністю, надаючи користувачеві потужний засіб для ефективної обробки та візуалізації інформації.

Інтерфейс Gamma здійснює вдале поєднання інтелектуальних функцій та легкості використання. Його структурований та елегантний дизайн сприяє зручності взаємодії з інструментами аналізу даних, а також простоті у створенні стилізованих презентацій.

Принцип роботи Gamma ґрунтується на використанні нейромереж для аналізу та інтерпретації великих обсягів даних. Цей інтелектуальний підхід дозволяє автоматизовано визначати ключові тренди, створювати висновки та надавати візуальні елементи для найефективнішого сприйняття інформації.

Заснований на концепції автоматизації та візуалізації даних, Gamma вирізняється можливістю легкої інтеграції із створенням презентацій. Користувачеві надається можливість створювати зручні та професійно виглядаючі презентації на основі оброблених даних, використовуючи інтуїтивний інтерфейс для максимальної ефективності та інноваційності у процесі роботи.

Приклад невеликої програми, яка генерує візуалізацію по запиту, але враховує стиль. Має великий спектр налаштування і можливість редагування після генерації роботи.

Dream Studio, являє собою інноваційну нейромережу, здатну аналізувати текст та трансформувати його у вражаючі зображення різних стилів. Цей

інструмент визначається своєрідністю і творчим підходом до перетворення текстової інформації в мистецькі витвори.

Інтерфейс Dream Studio спроектований з метою забезпечити користувачеві простоту та креативність. Елементи управління лаконічні та інтуїтивно зрозумілі, надаючи можливість вільно виражати свої ідеї та концепції через текстовий ввід.

Принцип роботи Dream Studio грунтується на застосуванні глибоких нейронних мереж для аналізу тексту та перетворення його в зображення. Система використовує передові алгоритми генерації контенту, які не лише інтерпретують семантику текстового введення, але й вміють враховувати різні стилі та художні прийоми, переносячи їх на створювані зображення.

З точки зору спостерігача, Dream Studio є не тільки інструментом для аналізу тексту, але і творчим партнером, який пропонує новий спосіб вираження ідей. Цей інтерфейс відкриває можливості для творчого експерименту та дозволяє користувачам створювати унікальні зображення, відображаючи глибокий потенціал синтезу тексту та мистецтва.

На основі аналізу систем для аналізу і візуалізації даних, представлених на ринку, можна зробити висновок, що насправді таких систем не існує. Всі системи мають певні обмеження, які не дозволяють їм виконувати весь заданий спектр завдань.

Для дослідження цієї проблематики можна використовувати наступні методи аналізу характеристик систем для аналізу і візуалізації даних. Це дозволить визначити, які завдання можуть виконуватися за допомогою кожної системи. Аналіз потреб організацій. Це дозволить визначити, які завдання необхідно виконувати організаціям.

Порівняння характеристик систем із потребами організацій, дозволить виявити, які системи не можуть виконувати всі необхідні завдання. На основі аналізу характеристик систем для аналізу і візуалізації даних можна зробити наступні висновки. Власні системи є найбільш потужними і гнучкими, але також можуть бути дорогими та складними у використанні. Вони часто не

підтримують широкий спектр завдань, а тому можуть бути неоптимальними для організацій, які потребують виконання широкого набору завдань.

Платформи для аналізу даних пропонують широкий спектр функцій для аналізу та візуалізації даних, але часто є менш потужними і гнучкими, ніж власні системи. Вони часто не підтримують всі необхідні завдання, а тому можуть бути неоптимальними для організацій, які потребують виконання складних завдань. Вільні та відкриті системи доступні безкоштовно або за ліцензією, але часто є менш потужними і гнучкими, ніж власні системи або платформи для аналізу даних. Вони часто не підтримують всі необхідні завдання, а тому можуть бути неоптимальними для організацій, які потребують виконання складних завдань. На основі аналізу потреб організацій можна зробити наступні висновки.

Організації потребують систем, які можуть виконувати широкий спектр завдань. Це включає такі завдання, як аналіз продажів і маркетингу, аналіз даних про клієнтів, аналіз даних про продукти та послуги, аналіз даних про операції, наукові дослідження, освіта та управління. Організації потребують систем, які є простими у використанні для користувачів з різним рівнем технічних знань. Це особливо важливо для невеликих організацій, які не мають штату кваліфікованих аналітиків.

На основі порівняння характеристик систем із потребами організацій можна зробити наступний висновок, що жодна система не може виконувати весь заданий спектр завдань. Всі системи мають певні обмеження, які не дозволяють їм виконувати всі необхідні завдання. Відсутність систем, які б виконували весь заданий спектр завдань, є проблемою для організацій, які потребують виконання широкого набору завдань.

Необхідність використання декількох систем. Це може ускладнити і удорожити процес аналізу даних. Неможливість виконання всіх необхідних завдань. Це може призвести до втрати можливості отримати цінні знання з даних.

Для вирішення цієї проблеми необхідно розробити нові системи, які будуть більш потужними, гнучкими та простими у використанні. Ці системи повинні підтримувати широкий спектр завдань, щоб задовольнити потреби різних організацій.

Розробка нових методів аналізу даних, дозволить поліпшити точність і ефективність аналізу даних. Використання штучного інтелекту. Штучний інтелект може бути використаний для автоматизації завдань, пов'язаних з аналізом даних. Розробка нових інтерфейсів користувача, дозволить зробити системи більш простими у використанні для користувачів з різним рівнем технічних знань. Розробка нових систем для аналізу і візуалізації даних є важливим напрямком досліджень, системи можуть допомогти організаціям отримати цінні знання з даних і приймати більш обґрунтовані рішення.

1.2Порівняльний аналіз аналогів

На сьогоднішній час існує вже досить багато сервісів для візуалізації даних, які вводить користувач. Серед існуючих реалізацій варто виділити інтернет сторінки такі як: Gamma, Recraft, Claude AI, Dream Studio, My reader.

Сервіс Gamma - це онлайн-платформа, яка дозволяє візуалізувати презентації за допомогою різних методів, таких як графіки, діаграми, карти, таблиці, ментальні карти тощо. Сервіс доступний на сайті gamma.ai.

Сервіс не може візуалізувати презентації, які містять складні дані або формули та сервіс не може візуалізувати презентації, які містять великі обсяги даних. Сервіс Gamma не виконує основних функцій магістерської, оскільки він не може аналізувати дані з презентації, що є аріважливим аспектом в роботі. Також не може виявляти закономірності і тенденції в даних та приймати рішення на основі аналізу даних, а це значно погіршує функціональність даного проекту.

Сервіс Gamma може бути використаний у магістерській роботі для візуалізації даних, які містяться в презентації. Це може допомогти в представленні результатів дослідження, пояснення складних даних, зробити

презентацію більш інформативною і зрозумілою. Однак, слід пам'ятати, що сервіс Gamma не може замінити повноцінний аналіз даних. Для цього необхідно використовувати інші інструменти і методи. Це важливе зауваження, яке слід враховувати при використанні сервісу.

Сервіс Recraft - це онлайн-платформа, яка дозволяє візуалізувати по запиту елементи дизайну, ілюстрацій і брендингу. Сервіс доступний на сайті recraft.ai.

Сервіс Recraft є інструментом для візуалізації дизайну, ілюстрацій і брендингу. Він може бути використаний у магістерській роботі для представлення своїх ідей та розробок. Однак, слід пам'ятати, що сервіс Recraft не може замінити повноцінне дослідження та аналіз даних. Для цього необхідно використовувати інші інструменти і методи.

На ряду з цим наявна база даних цієї організації не надає детальної інформації і функцій, а саме не можна аналізувати візуалізовувати по запиту на основі введених даних.

Сервіс Cloude AI - це онлайн-платформа, яка дозволяє аналізувати текстові документи. Сервіс доступний на сайті cloude.ai.

Як працює сервіс? Для аналізу текстового документа за допомогою сервісу Cloude AI необхідно зайти на сайт cloude.ai і зареєструватися, завантажте текстовий документ, вибрати тип аналізу, налаштувати аналіз відповідно до своїх потреб.

До функцій сервісу можна віднести аналіз текстових документів на основі штучного інтелекту та порівняння текстових документів. Також не менш важливим є зріз текстових документів, але це не дає повного функціоналу, яке стоїть на мені магістерської роботи.

Сервіс Dream Studio призначений для генерації зображень по стилю і кастомізації. Сервіс Dream Studio - це онлайн-платформа, яка дозволяє генерувати зображення по стилю і кастомізувати його в форматі додаткових запитів і генеративної заливки. Сервіс доступний на сайті dreamstudio.ai.

Сервіс Dream Studio може бути використаний у магістерській роботі для створення творчих і інноваційних зображень. Наприклад, студент, який розробляє новий продукт, може використовувати сервіс для створення концепт-артів для продукту. Якщо він вивчає поведінку споживачів, може використовувати сервіс для генерації зображень, які відображають певні поведінкові тенденції[3].

Сервіс My Reader - це онлайн-платформа, яка аналізує відправлений вам текстовий файл і відповідає на запитання по ньому. Сервіс доступний на сайті myreader.ai. Студент, який проводить дослідження ринку, може використовувати сервіс My Reader для аналізу маркетингових досліджень. Студент, який вивчає поведінку споживачів, може використовувати сервіс My Reader для аналізу опитувань. Студент, який розробляє новий продукт або послугу, може використовувати сервіс My Reader для аналізу юридичних документів.

Myreader, привертає увагу своєю заявленою здатністю аналізувати великі тексти та надавати відповіді на запитання. У науково-публіцистичному стилі можна описати його інтерфейс і принцип роботи наступним чином.

Інтерфейс Myreader вражає своєю простотою та легкістю використання. Чистий та ергономічний дизайн створює комфортне середовище для користувача, де ключові функції доступні на перший погляд.

Принцип роботи цієї нейромережі базується на використанні глибокого навчання[3] для обробки великих обсягів текстової інформації. Алгоритми аналізу тексту допомагають встановлювати семантичні зв'язки, виділяти ключові ідеї та здійснювати розуміння контексту.

В процесі взаємодії з Myreader, користувачі мають можливість подавати запитання до аналізованого тексту, отримуючи відповіді на основі інтелектуального аналізу. Ця система вчиться визначати ключові пункти та розрізняти суттєві елементи текстів, надаючи користувачам швидкі та точні результати.

Myreader можна сприймати як універсальний інструмент для обробки та розуміння великих обсягів текстової інформації, забезпечуючи ефективний засіб для отримання відповідей на запитання з різних дисциплін та областей знань. Загалом інструмент дуже і дає змогу досить легко і просто виконати ту чи іншу задачу пов’язану з аналізом великих масивів даних, що насправді наближує його до того що могло б задовільними потреби магістерської, але є велике обмеження по часу і комерційній складовій, яка значно зменшує доступність, крім того є момент обмеження для певних груп людей, які не проживають в США, адже доступ надається лише їм, що значно зменшує поріг входження для нових користувачів.

Після аналізу усіх аналогів визначено їхні переваги та недоліки та проведено порівняння із розроблюваним веб-додатком під назвою «SmartViz». Результат порівняння зведено в таблицю 1.1.

Таблиця 1.1 Порівняльні характеристики інтернет сторінок

Критерій	Claude AI	Dream Stydio	Recraft	Gamma	MyReader	SmartVIZ
Аналіз введених даних	Так	Ні	Так	Так	Так	Так
Генерація зображень	Ні	Так	Так	Так	Ні	Так
Генерація інфографіки	Ні	Так	Так	Так	Ні	Ні
Порівняння введених текстових файлів	Так	Ні	Ні	Ні	Так	Так
Робота при великих обсягах даних	Так	Ні	Так	Ні	Так	Так

В результаті порівняння існуючих аналогів було зроблено висновок, що розробка власного веб-додатку для ведення розкладу потягів є доцільною. В результаті розробки отримаємо продукт, який покриває недоліки існуючих аналогів та забезпечує порівняно вищу ефективність та більший обсяг функціоналу.

1.3 Аналіз методів розв’язання поставленої задачі

Методи розподілених[4] обчислень дозволяють обробляти надвеликі масиви даних, розподіляючи їх між декількома комп’ютерами. Також є методи машинного навчання дозволяють автоматично виявляти закономірності та тенденції в даних. Крім того, є методи візуалізації даних дозволяють ефективно представляти результати аналізу.

Розробка андроїд додатку для аналізу надвеликих об'ємів даних та їх візуалізації стає важливим завданням у світі сучасних інформаційних технологій. Задача стоїть перед розробниками не лише в створенні ефективного програмного продукту, але і в обґрунтуванні методології його реалізації. Перед подальшим дослідженням обраної теми, розглянемо основні аспекти методів реалізації проекту, спрямованого на створення андроїд додатку для аналізу та візуалізації надвеликих об'ємів даних по запиту.

Перше, що варто визначити, це обрана архітектура додатку. Вибір архітектурного підходу визначає не лише спосіб організації коду, але й ефективність програми. Однією з популярних архітектур для андроїд додатків є MVP[4] (Model-View-Presenter) або MVVM[4] (Model-View-ViewModel), які забезпечують високу модульність, підтримку тестування і зрозумілість коду.

Далі слід розглянути засоби для аналізу та обробки великих обсягів даних. Використання бібліотек і фреймворків, таких як Apache Spark або Hadoop, може значно полегшити завдання обробки великих наборів даних. Ці інструменти надають можливість паралельної обробки даних, що дозволяє прискорити аналітичні процеси.

Для забезпечення візуалізації інформації, важливо вибрати потужні бібліотеки для створення графіків та діаграм. MPAndroidChart[5], AnyChart, або Google Charts можуть бути використані для побудови інтерактивних та інформативних візуалізацій. З іншого боку, використання вбудованих засобів Android SDK також може бути ефективним вибором.

Важливим етапом є забезпечення безпеки обробки та зберігання великих об'ємів даних. Використання шифрування, механізмів автентифікації та забезпечення конфіденційності є ключовими аспектами розробки.

Неабиякою перевагою може стати використання технологій штучного інтелекту для покращення аналізу даних та автоматизації процесів. Використання нейромереж для прогнозування та виявлення патернів[5] може значно розширити функціональність додатку.

Загальний успіх проекту залежить від гармонійного поєднання усіх цих аспектів, враховуючи особливості завдань та потреб користувачів. Реалізація андроїд додатку для аналізу надвеликих об'ємів даних та їх візуалізації по запиту вимагає не тільки технічної експертності, але і глибокого розуміння конкретного контексту його використання.

Аналіз методів розподілених обчислень є ефективним способом обробки надвеликих масивів даних. Вони дозволяють розділити дані на частини, які обробляються на різних комп’ютерах. Це дозволяє значно прискорити обробку даних, а також використовувати більш потужні обчислювальні ресурси.

Методи машинного навчання дозволяють виявляти закономірності та тенденції в даних, які неможливо виявити за допомогою традиційних методів. Машинні алгоритми можуть навчатися на даних і виявляти в них закономірності, які неможливо виявити за допомогою людського аналізу.

Методи візуалізації даних дозволяють ефективно представляти результати аналізу. Вони дозволяють візуалізувати дані таким чином, щоб їх можна було легко зрозуміти.

Вибір методів для розв’язання поставленої задачі залежить від конкретних вимог до системи. Якщо необхідно обробляти надвеликі масиви

даних, які неможливо обробити на одному комп’ютері, то необхідно використовувати методи розподілених обчислень. Якщо необхідно виявляти закономірності та тенденції в даних, то необхідно використовувати методи машинного навчання. Якщо необхідно ефективно представляти результати аналізу, то необхідно використовувати методи візуалізації даних.

Для розробки ефективної системи для обробки, аналізу та візуалізації надвеликих масивів даних необхідно використовувати комбінацію різних методів. Це дозволить забезпечити високу ефективність системи та її здатність вирішувати широкий спектр завдань.

Методи розподілених обчислень використовуються в системах Hadoop[6], Spark[6] та MapReduce[6]. Методи машинного навчання використовуються в TensorFlow, PyTorch та Scikit-learn.Методи візуалізації даних використовуються в таких системах, як Tableau, QlikView та Power BI.

Методи розподілених обчислень дозволяють обробляти надвеликі масиви даних, розподіляючи їх між декількома комп’ютерами. Цей метод є ефективним способом обробки надвеликих масивів даних, оскільки дозволяє значно прискорити обробку даних, а також використовувати більш потужні обчислювальні ресурси.

Системи розподілених обчислень. Існує ряд систем розподілених обчислень, які можна використовувати для обробки надвеликих масивів даних.

Hadoop[7] - це відкрита система розподілених обчислень, яка використовується для зберігання та обробки великих обсягів даних. Hadoop складається з декількох компонентів, які працюють разом для забезпечення розподілених обчислень.

Spark[7] - це високопродуктивна система розподілених обчислень, яка використовується для обробки великих обсягів даних. Spark поєднує в собі можливості Hadoop для зберігання даних з високопродуктивними алгоритмами обробки даних.

MapReduce[7] - це парадигма розподілених обчислень, яка використовується для обробки великих обсягів даних. MapReduce складається з двох етапів. етапу Map, який розподіляє дані між комп'ютерами, та етапу Reduce, який об'єднує результати з етапу Map.

Аналітику великих даних - методи розподілених обчислень використовуються для аналізу великих обсягів даних для виявлення закономірностей та тенденцій.

Машинне навчання - методи розподілених обчислень використовуються для навчання машинних алгоритмів на великих обсягах даних.

Обробку даних у реальному часі - методи розподілених обчислень використовуються для обробки даних у реальному часі, наприклад, для виявлення шахрайства або для прогнозування погоди.

Існує ряд систем машинного навчання, які можна використовувати для виявлення закономірностей та тенденцій в надвеликих масивів даних.

TensorFlow[8] - це відкрита система машинного навчання, яка використовується для навчання та застосування моделей машинного навчання. TensorFlow підтримує широкий спектр моделей машинного навчання, включаючи нейронні мережі, логістичну регресію та класифікацію.

PyTorch[8] - це відкрита система машинного навчання, яка використовується для навчання та застосування моделей машинного навчання. PyTorch підтримує широкий спектр моделей машинного навчання, включаючи нейронні мережі, логістичну регресію та класифікацію.

Scikit-learn[8] - це відкрита система машинного навчання, яка використовується для навчання та застосування моделей машинного навчання. Scikit-learn підтримує широкий спектр моделей машинного навчання, включаючи класифікацію, регресію, кластеризацію та рекомендаційні системи.

Методи машинного навчання використовуються в широкому спектрі застосувань, включаючи і класифікацію, яка застосовується як методи машинного навчання використовуються для класифікації даних на різні

категорії. Наприклад, методи машинного навчання можуть бути використані для класифікації продуктів як "купувати" або "не купувати".

Регресію[9] - методи машинного навчання використовуються для прогнозування значення змінної на основі інших змінних. Наприклад, методи машинного навчання можуть бути використані для прогнозування ціни будинку на основі його характеристик.

Кластеризацію[9] - методи машинного навчання використовуються для групування даних, які є схожими. Наприклад, методи машинного навчання можуть бути використані для групування клієнтів за їхніми поведінковими характеристиками.

Рекомендаційні системи - методи машинного навчання використовуються для рекомендації продуктів або послуг користувачам. Наприклад, методи машинного навчання можуть бути використані для рекомендації фільмів користувачам, які вони, ймовірно, побачать.

Методи візуалізації даних дозволяють візуалізувати дані таким чином, щоб їх можна було легко зрозуміти. Цей метод є важливим для аналізу великих даних, оскільки дозволяє користувачам швидко і легко отримувати інформацію з даних.

Існує ряд систем візуалізації даних, які можна використовувати для візуалізації надвеликих масивів даних.

Tableau[10] - це система візуалізації даних, яка використовується для створення інтерактивних візуалізацій даних. Tableau підтримує широкий спектр типів візуалізацій, включаючи діаграми, графіки, карти та інфографіку.

QlikView[10] - це система візуалізації даних, яка використовується для створення динамічних візуалізацій даних. QlikView дозволяє користувачам створювати візуалізації, які змінюються в залежності від даних, які вони досліджують.

Power BI[10] - це система візуалізації даних, яка використовується для створення корпоративних візуалізацій даних. Power BI підтримує широкий

спектр типів візуалізацій і дозволяє користувачам створювати візуалізації, які можна ділитися з іншими.

Методи візуалізації даних використовуються в широкому спектрі застосувань, включають аналіз бізнесу. Це методи візуалізації даних використовуються для аналізу бізнес-даних для виявлення тенденцій та закономірностей. Наприклад, методи візуалізації даних можуть бути використані для аналізу продажів для виявлення сезонних тенденцій.

В сучасному науковому середовищі методи візуалізації даних стають важливою складовою для дослідження та передачі знань у різних областях, починаючи від науки про клімат та закінчуючи освітніми програмами. Це дозволяє ефективно взаємодіяти з великими об'ємами інформації та використовувати візуальні засоби для кращого розуміння та інтерпретації даних.

Методи візуалізації даних в дослідженні зміни клімату включають аналіз графіків, діаграм та інших візуальних елементів. Дослідження температурних змін, вологості та інших кліматичних параметрів використовуються для виявлення тенденцій та розуміння динаміки змін у природних процесах. Впровадження тривимірних моделей дозволяє вченим ефективно аналізувати просторові залежності та кореляції між різними параметрами.

Методи візуалізації даних у сфері освіти виявляються ефективним інструментом для навчання та розуміння статистичних концепцій. Створення навчальних матеріалів, що використовують інтерактивні графіки та візуалізації, сприяє кращому засвоєнню матеріалу студентами. Використання кольорових схем та анімацій дозволяє створити цікавий та ефективний освітній досвід.

Дослідження використання методів візуалізації даних для дослідження та освіти підтверджує їхню важливість у вивченні та розумінні складних інформаційних структур. Вони не лише допомагають науковцям виявляти нові закономірності в даних, але й роблять освітні процеси більш доступними та захоплюючими для учнів. Використання сучасних інструментів візуалізації

сприяє підвищенню рівня розуміння складних явищ і розвитку нових методологій у різних наукових дисциплінах.

Технології для розробки системи для обробки, аналізу та візуалізації надвеликих масивів даних можна сформувати на деякі категорії. Загалом, основі аналізу методів та технологій, що застосовуються для обробки, аналізу та візуалізації надвеликих масивів даних, було зроблено наступний вибір технологій для розробки системи.

Hadoop Distributed File System (HDFS) [11]- система розподілених файлових систем, яка використовується для зберігання великих обсягів даних.

Apache Hive[11] - система для запитування і аналізу даних, яка використовує мову SQL.

Apache Spark[11] - високопродуктивна система розподілених обчислень, яка використовується для обробки великих обсягів даних.

Apache Spark[11] - високопродуктивна система розподілених обчислень, яка використовується для обробки великих обсягів даних.

Apache Hadoop MapReduce[11] - парадигма розподілених обчислень, яка використовується для обробки великих обсягів даних.

Apache Spark MLlib[11] - бібліотека машинного навчання для Apache Spark.

TensorFlow - система машинного навчання, яка використовується для навчання та застосування моделей машинного навчання.

Scikit-learn - система машинного навчання, яка використовується для навчання та застосування моделей машинного навчання.

Tableau - система візуалізації даних, яка використовується для створення інтерактивних візуалізацій даних.

QlikView - система візуалізації даних, яка використовується для створення динамічних візуалізацій даних.

Power BI - система візуалізації даних, яка використовується для створення корпоративних візуалізацій даних.

Ці технології є відкритими і широко використовуються в розробці систем для обробки, аналізу та візуалізації надвеликих масивів даних.

Метод MVP обрано як базову архітектуру для Android додатку. У цьому підході Модель відповідає за роботу з даними, Представлення відображає інтерфейс користувача, а Презентер виконує ролі посередника між ними. MVP забезпечує модульність та легкість тестування, сприяє структурованості коду та взаємодії компонентів.

Для ефективної обробки великих об'ємів даних використовуються Apache Spark та Hadoop. Apache Spark забезпечує можливість паралельної обробки даних, використовуючи Resilient Distributed Datasets (RDD) [12], що прискорює аналітичні процеси. Hadoop допомагає у роботі з розподіленими обчислювальними ресурсами, забезпечуючи ефективну роботу з великими даними.

Для візуалізації оброблених даних використовуються потужні бібліотеки. MPAndroidChart[12], відома своєю гнучкістю та можливістю створення різноманітних графіків, використовується для візуалізації статистичної інформації. AnyChart[12] та Google Charts[12] використовуються для створення динамічних та інтерактивних графіків, що полегшують взаємодію з користувачем та забезпечують зрозумілість даних.

Безпека обробки та зберігання великих обсягів даних вирішується за допомогою засобів шифрування та механізмів автентифікації. Використання шифрування в базі даних та передачі даних допомагає у запобіганні несанкціонованому доступу та забезпеченні конфіденційності інформації.

Враховуючи сучасні тенденції, використано технології штучного інтелекту для покращення аналізу даних. Використання нейромереж для прогнозування та виявлення патернів дозволяє автоматизувати процеси, зробити прогнози та покращити точність аналізу великих обсягів інформації.

Реалізація Android додатку для аналізу надвеликих об'ємів даних та їх візуалізації включає в себе комплексний підхід, який охоплює архітектурні вирішення, засоби обробки даних, бібліотеки для візуалізації, заходи з безпеки

та інтеграцію інтелектуальних технологій. Комбінування цих методів створює потужний інструмент для аналізу та взаємодії з великими обсягами інформації, що стає ключем до успіху в сучасному світі технологій.

1.4 Постановка задач дослідження

Проаналізувавши питання розробки веб-додатку для ведення розкладу потягів, було визначено наступні завдання, які необхідно виконати для розробки програмного продукту.

– проаналізувати сучасний стан додатків;

– проаналізувати існуючі алгоритми та їх реалізацію для вирішення питання;

– розробити метод зчитування великих обсягів пам’яті і аналізу їх з візуалізацією по запиту;

– розробити алгоритми для реалізації власної прогами;

– розробити додаток для зчитування, аналізу та візуалізації даних по запиту;

– провести тестування розробленого програмного додатку.

1.5 Висновки

У цьому розділі було проведено аналіз стану систем аналізу і генерації даних. Було проведено аналіз аналогів які на сьогодні є в наявності та проведено порівняння їх між собою та додатком «SmartViz». У результаті чого було доведено доцільність розробки магістерської дипломної роботи. Також проведено аналіз існуючих підходів до вирішення проставленої задачі, в результаті чого вирішено створити окремий програмний модуль для обробки інформації. Також було вирішено використовувати Hadoop Distributed File System, Apache Hadoop MapReduce, Scikit-learn, QlikView. Було встановлено основні завдання, які необхідно виконати для розробки веб-додатку ведення розкладу руху потягів.

2 РОЗДІЛ. РОЗРОБКА МЕТОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ

2.1 Архітектурне проектування системи та проектування структурної схеми інтерфейсу

Архітектура системи визначає її структуру, компоненти та взаємозв'язки. Ця структура може бути логічною або фізичною. Архітектурне проектування передбачає створення архітектурного рішення, яке відповідає вимогам системи та забезпечує бажані якості, такі як масштабованість, надійність, ефективність та розширюваність. Архітектурне проектування програмної системи визначає загальну структуру, організацію та взаємодію компонентів програмної системи з метою досягнення певних цілей.

Вікна для управління даними в системі SmartViz відзначаються чітким функціоналом та ергономічним дизайном, що спрощує навігацію користувача. Кожне вікно виконує свою конкретну роль у процесі аналізу та відображення інформації.

Вікно "Історія файлів" є центральним елементом для зберігання та швидкого доступу до списку вже відкритих файлів даних. Користувач може легко оглядати та відкривати файли, використовуючи прості елементи взаємодії.

"Історія запитів" допомагає відслідковувати всі запити, зроблені до даних, і надає можливість повторити раніше виконані запити. Це вікно служить для ефективного управління історією взаємодії з даними.

"Презентація" є вікном візуалізації, де користувач може спостерігати та взаємодіяти з візуалізованою інформацією. Кнопка "Зберегти" надає можливість фіксувати візуалізацію в історії файлів для подальшого використання.

Загальний принцип взаємодії передбачає початок з вікна "Запит для файлу", за яким слідує вибір файлів з історії, їх відображення у вікні "Презентація" та можливість збереження візуалізацій для подальшого

використання. Кожне вікно відповідає за конкретний аспект обробки та взаємодії з даними, створюючи комплексну та зручну систему управління інформацією.

Кожен блок в схемі являє собою окремий модуль у проекті. Блоки пов`язані між собою. Користувач – головний модуль, адже саме він взаємодіє із системою, тому він має найбільше зв`язків, які описано в таблиці 2.1

Таблиця 2.1 Зв`язки модуля користувача

Блок 1	Блок 2	Зв`язок
Користувач	Зчитування файлу для роботи	Користувач має можливість завантажити файл для обробки
Користувач	Архів файлів	Користувач може переглянути архів файлів які завантажував
Користувач	Архів запитів	Користувач має можливість переглянути всі запита які давав до завантажених файлів
Користувач	Запит для файлу	Користувач має можливість написати повідомлення чат-боту і дати завдання для роботи
Користувач	Обробка запиту	Користувач отримує різноманітні результати в чаті, відповідно до того що він попросив ()

Модуль - це структурна одиниця системи, яка має певні атрибути та операції. Атрибути модуля визначають його стан або процес, а операції - це дії, які модуль може виконувати.

У системі обробки даних вирізняються три ключові модулі. модуль даних, модуль візуалізації та модуль запитів, кожен з яких відповідає за конкретний етап обробки і представлення інформації.

Модуль даних відіграє важливу роль у зберіганні та обробці даних, які

служать основою для подальшої візуалізації. Визначаються атрибути, такі як тип та структура даних, а опції, такі як розмірність та масштаб, дозволяють налаштовувати параметри модуля для оптимального використання інформації.

Модуль візуалізації відповідає за процес створення візуалізацій даних, визначаючи тип та параметри візуалізації. Налаштування кольору та шрифту дозволяють користувачу контролювати естетичний аспект відображення інформації.

Модуль запитів відіграє важливу роль у взаємодії з користувачем, обробляючи запити на візуалізацію даних. Атрибути визначають тип запиту, що може бути заданий користувачем, а опції дозволяють налаштовувати логіку обробки запитів.

Взаємодія між цими модулями відбувається за чіткою логікою. Користувач починає з вікна "Запит на файл", яке служить точкою входу для обробки файлів даних. Історія файлів визначає, чи був файл вже відкритий, і передає дані до Модуля даних для обробки. Після цього, користувач може задати запит на візуалізацію даних, і візуалізація буде створена за допомогою Модуля візуалізації. Результат відображається у вікні "Презентація", створюючи повний цикл обробки та аналізу даних у системі.

Модуль - це основна будівельна одиниця системи. Він складається з двох основних компонентів.

Функція або завдання[13] - це те, за що відповідає модуль. Наприклад, модуль може відповідати за обробку даних, відображення інформації на екрані або взаємодію з користувачем.

Інтерфейс[13] - це спосіб, яким модуль взаємодіє з іншими модулями. Наприклад, модуль може мати інтерфейс для обміну даними з іншим модулем або інтерфейс для отримання команд від користувача.

Модулі можуть бути самостійними або взаємодіяти з іншими модулями. Самостійний модуль може виконувати свою функцію без взаємодії з іншими модулями. Наприклад, модуль для обробки зображень може виконувати свою функцію без взаємодії з модулем для відображення інформації на екрані.

Модулі, які взаємодіють один з одним, обмінюються даними або командами. Наприклад, модуль для обробки даних може обмінюватися даними з модулем для відображення інформації на екрані.

По-перше, він спрощує розробку системи. Модулі можна розробляти незалежно один від одного, що дозволяє розробникам зосередитися на певній функції або завданні.

По-друге, розподіл системи на модулі полегшує розуміння системи. Модулі є самостійними і мають чіткий набір функцій і завдань. Це дозволяє розробникам та користувачам легко зрозуміти, як працює система.

По-третє, розподіл системи на модулі полегшує підтримку системи. Модулі можна замінити або оновити без необхідності змінювати всю систему.

Система інтерфейсу складається з ключових елементів, кожен з яких виконує визначені функції, створюючи спрощений та зручний механізм взаємодії користувача з програмою чи додатком.

Заголовок виступає ідентифікаційною міткою, що вказує на назву програми або додатку та дозволяє користувачеві отримувати доступ до основного меню. Пункти меню, у свою чергу, відкривають доступ до різноманітних функцій системи.

Панель завдань розташована в стратегічному місці і забезпечує швидкий доступ до найбільш використовуваних опцій. Це включає в себе відкриття файлів, запуск програм та перегляд веб-сторінок, що робить процес взаємодії швидким та ефективним.

Основний екран представляє собою область, де відбувається відображення контенту для користувача. Взаємодія з основним екраном відбувається за допомогою миші, клавіатури або сенсорного екрану, надаючи різноманітні засоби введення для різних ситуацій.

Кожен елемент взаємодіє з іншими засобами введення користувача. Наприклад, користувач може використовувати мишу для відкриття меню, вибирати функції за допомогою клавіатури, а потім взаємодіяти з основним екраном через сенсорний екран.

Система дозволяє взаємодіяти з різними елементами і забезпечує логічний та природний порядок введення та взаємодії користувача. Такий підхід створює інтуїтивно зрозуміле середовище, що полегшує користувачеві використання програм та додатків.

У світі сучасної технології, де користувачі мобільних пристроїв вимагають ефективності та легкості в обслуговуванні, система інтерфейсу головного екрану грає ключову роль у сприянні зручності та швидкості взаємодії. Заголовок, відображаючи назву програми чи додатку, функціонує як ідентифікатор та пристосований спосіб доступу до головного меню. Це стає початковим кроком для користувача, який може використовувати сенсорний екран чи клавіатуру для відкриття бажаної програми.

Меню, як ключовий компонент інтерфейсу, пропонує доступ до різноманітних функцій, що розширює можливості користувача в області управління додатками. Торканням екрану користувач може активувати відповідні опції, забезпечуючи простий та інтуїтивно зрозумілий механізм вибору функцій.

Панель завдань, розміщена стратегічно, надає доступ до широкого спектру опцій, включаючи відкриття файлів, запуск програм та перегляд веб-сторінок. Ці кнопки визначаються для виконання різних дій, що робить взаємодію із пристроєм максимально продуктивною.

Основний екран, зі своїм контентом, взаємодіє із користувачем через сенсорний екран чи клавіатуру. Це стає місцем, де користувач може взаємодіяти із вмістом пристрою, забезпечуючи широкий спектр споживача.

В процесі взаємодії користувача із системою використання сенсорного екрану виявляється як ключовий елемент, що забезпечує швидкий та природний обмін інформацією. Від реакції системи на різні дії користувача, таких як торкання або вибір, залежить ефективність і зручність використання пристрою. Такий підхід створює динамічну та інтуїтивну систему, яка відповідає сучасним вимогам користувача до мобільних технологій.

2.2 Розробка моделі для зчитування надвеликих об’ємів даних

Промпти[14] - це конкретні запити, інструкції або питання, які використовуються в системах штучного інтелекту для взаємодії з користувачами.

Вони можуть бути використані для різних цілей, таких як. Промти можуть надаватися користувачеві для того, щоб пояснити йому, як використовувати ШІ. Наприклад, промти можуть бути використані для того, щоб пояснити користувачеві, як вводити текст, як вибирати опції або як виконувати конкретні завдання.

Промти можуть використовуватися для того, щоб запитати користувача про інформацію, яка необхідна ШІ для виконання завдання. Наприклад, промти можуть бути використані для того, щоб запитати користувача про його ім'я, адресу електронної пошти або номер телефону.

Промти можуть використовуватися для того, щоб попередити користувача про потенційні ризики або проблеми. Наприклад, промти можуть бути використані для того, щоб попередити користувача про те, що він вводить невірну інформацію або що він може виконати небезпеку завдання.

Промти можуть використовуватися для того, щоб контролювати взаємодію між користувачем і ШІ. Наприклад, промти можуть бути використані для того, щоб заохотити користувача продовжити взаємодію або для того, щоб перервати взаємодію, якщо вона стала непродуктивною.

Промти є важливими для людини, яка користується ШІ, з кількох причин. Зокрема, полегшують взаємодію з ШІ. Вони допомагають користувачеві зрозуміти, як використовувати ШІ, і забезпечують йому необхідні вказівки. Це може зробити взаємодію з ШІ більш ефективною та продуктивною. Також підвищують безпеку використання ШІ. Вони можуть попередити користувача про потенційні ризики або проблеми, що може допомогти запобігти травмам або збиткам. До того ж, роблять взаємодію з ШІ більш персоналізованою. Вони можуть бути адаптовані до індивідуальних

потреб і особливостей користувача, що може зробити взаємодію з ШІ більш приємною та корисною.

Промти можуть використовуватися для того, щоб покращити взаємодію з користувачем у різних аспектах. Наприклад, вони можуть бути використані для того, щоб.

Збільшити ефективність взаємодії дає змогу зменшити поріг входження для користувача. Промти можуть допомогти користувачеві зрозуміти, як використовувати ШІ, і забезпечити йому необхідні вказівки. Це може зробити взаємодію з ШІ більш ефективною та продуктивною[15].

Підвищення задоволеності користувача також відбувається, адже взаємодія є максимально хуманізованою, що прирівнює до процесу зі спілкування зі знайомим. Промти можуть бути адаптовані до індивідуальних потреб і особливостей користувача, що може зробити взаємодію з ШІ більш приємною та корисною.

Зменшення кількості помилок призводить до кращої довіри і впевненості в виконаній роботі. Промти можуть попередити користувача про потенційні ризики або проблеми, що може допомогти запобігти травмам або збиткам.

Використання промтів є важливим аспектом розробки ШІ, який може допомогти зробити взаємодію з ШІ більш ефективною, продуктивною та приємною.

Для створення чат-боту було обрано мовну модель штучного інтелекту «gpt-3.5-turbo» [15], яка підключається за допомогою OpenAIApi. ChatGPT - це потужна модель глибокого навчання, яка базується на архітектурі трансформера. Вона була розроблена для генерації природної мови, що дозволяє їй створювати текстові відповіді на запити користувачів та спілкуватися з ними в режимі чату. Однією з ключових особливостей ChatGPT є її здатність генерувати тексти, які виглядають інформативними та природними для сприйняття.

Щоб чат-бот на основі «gpt-3.5-turbo» міг зчитувати великі об'єми даних інформації, необхідно виконати такі кроки[15].

Набрати дані є важливим аспектом в роботі ШІ, адже для цього можна використовувати різні джерела, такі як тексти книг, статей, веб-сайтів, а також код. Дані повинні бути у текстовому форматі, наприклад, у форматі CSV[16] або JSON.

Очистити дані також мають бути присутні через навантаження системи. Перед тим, як використовувати дані для навчання чат-бота, їх необхідно очистити від помилок і шуму. Це може включати видалення непотрібних даних, таких як рекламні блоки або помилки форматування.

Також можна розділити дані на навчальний і тестові набори. Навчальний набір використовується для навчання чат-бота, а тестовий набір - для оцінки його ефективності.

Крім того, потрібно натренувати чат-бота. Для цього можна використовувати різні алгоритми машинного навчання, такі як метод опорних векторів або нейронні мережі.

Набір даних повинен бути достатньо великим, щоб чат-бот міг навчитися генерувати релевантні відповіді на широкий спектр запитів. Також важливо, щоб дані були різноманітними, щоб чат-бот міг опанувати різні стилі та теми.

При очищенні даних важливо видалити будь-яку інформацію, яка може вплинути на точність навчання чат-бота. Це може включати видалення непотрібних даних, таких як рекламні блоки або помилки форматування, а також видалення даних, які можуть бути шкідливими або образливими.

Навчання чат-бота зазвичай здійснюється на основі алгоритму машинного навчання, який вимагає наявності навчального і тестового наборів даних. Навчальний набір використовується для навчання чат-бота, а тестовий набір - для оцінки його ефективності.

Для навчання чат-бота можна використовувати різні алгоритми машинного навчання. Одним із поширених алгоритмів є метод опорних

векторів, який використовується для навчання класифікаторів. Іншим поширеним алгоритмом є нейронні мережі, які можна використовувати для навчання чат-бота генерувати текст. В алгоритмі є тіло запиту до серверу ChatGPT в якому ми використовуєм самостійно створений промпт.

У процесі використання мови програмування Python для взаємодії з ChatGPT, перший крок передбачає імпортування необхідних модулів. В даному випадку, ключовим стає модуль GPT3, який містить функції для здійснення запитів до ChatGPT. Це спрощує процес отримання відповідей від штучного інтелекту.

Наступний етап передбачає створення об'єкта GPT3, який буде використовуватися для взаємодії з ChatGPT. Для цього необхідно передати ключ API, який дозволяє отримати доступ до функціоналу ChatGPT. Цей ключ, як правило, видається після реєстрації на відповідному веб-сайті.

Далі в програмі задається конкретний запит до ChatGPT. Це може бути будь-яке питання, прохання чи заява, залежно від потреб користувача.

Отримання відповіді відбувається через використання функції generate об'єкта GPT3. Цей етап є критичним для отримання відповідей, що відображають контент, що виникає з інтелектуального алгоритму ChatGPT.

Завершальним етапом є виведення отриманої відповіді на екран за допомогою функції print. Це робить взаємодію користувача з результатами запиту максимально доступною та зручною.

Цей процес, ілюстрований через запитання про поточну дату, розкриває можливості використання ChatGPT для отримання різноманітної інформації та творчого контенту. Звертаючись до системи запитань та відповідей, користувач може отримати від ChatGPT не лише текстові, але й творчі відповіді на різноманітні питання чи завдання.

Завдяки цим налаштуванням наша модель штучного інтелекту буде навчатись зчитувати дані в дуже великому об’ємі. Можна буде завантажувати методички, курсові і цілі книги, які потім будуть проаналізовані. Таким чином

за короткий час нейромережа зможе після аналізу відповідати на запитання, порівнювати інформацію і візуалізувати все по запиту.

2.3 Розробка методу візуалізації вписаних даних по запиту

Візуалізація даних - це процес представлення даних у візуальній формі, наприклад, у вигляді діаграм, графіків або карт. Вона є важливим інструментом для аналізу даних, оскільки дозволяє легко розуміти складні набори даних і виявляти закономірності, які неможливо помітити при простому перегляді даних у текстовому вигляді.

Вона полегшує розуміння складних наборів даних. Візуалізація даних дозволяє представити дані у вигляді, який легко сприймається людським мозком. Це може бути особливо корисно для аналізу даних, які містять велику кількість інформації або мають складні взаємозв'язки.

Вона допомагає виявити закономірності та тренди. Візуалізація даних може допомогти виявити закономірності та тренди, які неможливо помітити при простому перегляді даних у текстовому вигляді. Це може бути корисно для прийняття рішень або розробки стратегій.

Вона робить дані більш інформативними та цікавими. Візуалізація даних може зробити дані більш інформативними та цікавими для аудиторії. Це може бути корисно для презентацій, звітів або навчальних матеріалів.

Візуалізація даних може використовуватися в різних сферах діяльності, включаючи бізнес, науку, освіту та державне управління. Наприклад, бізнес може використовувати візуалізацію даних для аналізу продажів, маркетингу або виробництва.

Наука може використовувати візуалізацію даних для дослідження природи або суспільства. Освіта може використовувати візуалізацію даних для навчання учнів. Державне управління може використовувати візуалізацію даних для аналізу даних про населення або економіку.

Важливо вибрати правильний тип візуалізації для конкретних даних. Існує безліч різних типів візуалізацій, кожен з яких має свої переваги та недоліки.

Наприклад, діаграми добре підходять для представлення даних, які змінюються з часом, а графіки добре підходять для представлення даних, які мають розподіл[17].

Візуалізація даних - це потужний інструмент, який може допомогти вам краще зрозуміти дані і зробити їх більш інформативними та цікавими.

Першим кроком у розробці модуля для візуалізації даних є оцінка потреб користувачів. Вибір візуальних елементів.

На другому етапі необхідно вибрати візуальні елементи, які будуть використовуватися для візуалізації даних. Існує безліч різних типів візуалізацій, кожен з яких має свої переваги та недоліки. Наприклад, діаграми добре підходять для представлення даних, які змінюються з часом, а графіки добре підходять для представлення даних, які мають розподіл. Розробка алгоритмів візуалізації необхідне для коректної і безперебіної роботи.

На третьому етапі необхідно розробити алгоритми візуалізації, які будуть використовуватися для створення візуальних елементів. Ці алгоритми повинні бути ефективними та забезпечувати високу якість візуалізації.

На четвертому етапі необхідно реалізувати модуль для візуалізації даних. Це може бути зроблено за допомогою різних мов програмування та фреймворків.

Після реалізації модуля необхідно протестувати його та внести необхідні вдосконалення. Це може бути зроблено за допомогою різних методів, включаючи ручне тестування, автоматизоване тестування та тестування з використанням користувачів.

У сучасному програмуванні, особливо в мові Python, взаємодія з інтелектуальними алгоритмами стає все більш розповсюдженою. Процес використання ChatGPT для отримання відповідей на різноманітні запитання включає декілька ключових етапів.

Імпортування необхідних модулів є початковим кроком, який дає змогу використовувати функціонал GPT3. Це досягається завдяки використанню мови програмування Python та імпорту відповідного модулю[18].

Далі, створюється об'єкт GPT3, який використовуватиметься для взаємодії з ChatGPT. Ключ API[18], який надається під час створення об'єкта, є важливим елементом для забезпечення безперервного доступу до інтелектуального алгоритму.

Задання конкретного запиту, такого як "Який сьогодні день?", відбувається через використання об'єкта GPT3. Це є ключовим етапом, який створює основу для отримання інформації від ChatGPT.

Отримання відповіді є процесом, в якому запит відправляється до ChatGPT, і відповідь, яку генерує алгоритм, отримується за допомогою функції generate. Це може бути текстова або графічна інформація, залежно від конкретного запитання.

Завершальним етапом є виведення отриманої відповіді на екран, щоб користувач міг зручно взаємодіяти з результатами запиту.

Проте, код, який представлений на зображенні, має свої обмеження. Він потребує ключа API, що обмежує доступність взаємодії, а також може генерувати лише текстові відповіді. Розширення коду можливе шляхом додавання функціоналу, такого як введення ключа API з файлу чи регулювання параметрів запиту.

Цей модуль дозволяє створювати візуалізації даних за допомогою чат-бота. Він може бути використаний для аналізу даних у різних сферах діяльності, включаючи бізнес, науку, освіту та державне управління.

Модуль може обробляти запити користувача для створення візуалізації даних. Запити можуть бути простими або складними, наприклад, "Створіть діаграму, яка показує продажі за останній квартал" або "Створіть карту, яка показує поширення коронавірусу". Створення візуалізацій даних. Модуль може створювати візуалізації даних різних типів, включаючи діаграми, графіки, карти та таблиці. Модуль може використовувати різні візуальні елементи та стилі для створення привабливих та інформативних візуалізацій.

Модуль може подавати візуалізації користувачеві різними способами, включаючи текстовий опис, зображення або веб-сторінку. Приклад

використання модуля.

Користувач може запитати чат-бота створити діаграму, яка показує продажі за останній квартал. Чат-бот обробить запит і створить діаграму. Потім чат-бот може подати діаграму користувачеві у вигляді текстового опису, зображення або веб-сторінки. Детальна реалізація модуля. Модуль може бути реалізований за допомогою різних мов програмування та фреймворків. Найбільш поширеним підходом є використання веб-фреймворка, такого як Django[19] або Flask[19]. Чат-бот може бути реалізований за допомогою бібліотеки для створення чат-ботів, наприклад, Rasa[20] або Dialogflow[20].

2.4 Розробка основних алгоритмів програмної реалізації

Оскільки головною метою додатку є зчитування та аналіз, тому приклад для реалізації алгоритму програмного, використаємо саме його.

Алгоритм зчитування і аналізу даних.

- отримання запиту від користувача. Користувач вводить запит у текстовому вигляді;
- розбиття файлів на частини. Файли розбиваються на частини за допомогою роздільників, розміру файлу або алгоритму;
- зчитування і аналіз фрагментів файлів. Фрагменти файлів зчитуються і аналізуються за допомогою рекурсії, потокової обробки або паралельної обробки;
- зберігання результатів аналізу. Результати аналізу зберігаються в пам'яті або в базі даних за допомогою хеш-таблиць, дерев або баз даних;
- пошук відповідного результату аналізу. Відповідний результат аналізу шукається за допомогою індексів, пошуку по ключу або пошуку по шаблону;
- формування відповіді на запит. Відповідь на запит формується у вигляді тексту, графіків або таблиць.

В процесі роботи програми над обробкою даних відбувається кілька послідовних етапів, кожен з яких виконує важливі функції для забезпечення якісного аналізу та візуалізації інформації.

Перший етап - завантаження даних - визначає початковий крок, на якому програма отримує доступ до джерела інформації, такого як файли, бази даних або веб-сайти. Важливо відзначити, що дані можуть мати різний формат, і програма повинна взаємодіяти з ними ефективно, незалежно від того, чи це текстові файли, CSV, XML або JSON[21].

На другому етапі - очищення даних - програма виконує операції для виправлення помилок та усунення аномалій, що можуть виникнути під час завантаження. Видалення дублікатів, заповнення пропусків та стандартизація даних роблять інформацію більш консистентною та готовою для подальшого аналізу.

Аналіз даних, що відбувається на третьому етапі, включає в себе застосування різних статистичних методів, таких як регресія, кластеризація та класифікація. Це дозволяє програмі виявити закономірності та тенденції, що можуть бути корисні для подальших висновків.

На четвертому етапі - візуалізації даних - програма використовує різні методи, такі як діаграми, графіки і карти, для графічного представлення результатів аналізу. Це робить інформацію більш доступною та зрозумілою для кінцевого користувача.

Важливим елементом є взаємодія між модулями програми, що забезпечується за допомогою API. Це набір функцій і методів, які дозволяють модулям обмінюватися інформацією. Наприклад, модуль "Завантаження даних" може використовувати API модуля "Аналіз даних"[21] для передачі інформації для подальшого аналізу, і так далі.

Кожен етап може бути розширений або удосконалений для забезпечення більш широкого функціоналу програми. Можливості розширення включають імпорт даних з нових джерел, більш глибокий аналіз, складніші візуалізації і використання сторонніх інструментів. Це можна здійснити шляхом додавання

нових модулів, розширення існуючих, або використання сторонніх ресурсів для покращення функціоналу.

В процесі роботи програма виконує кілька послідовних етапів, спрямованих на обробку та аналіз вхідного тексту. Починаючи зі зчитування вхідного тексту, програма може отримувати дані з різних джерел, таких як файли, консоль або веб-сайти, при цьому текст може мати різний формат, включаючи текстові файли, CSV, XML або JSON.

Наступний етап - очищення вхідного тексту - передбачає операції для усунення помилок або аномалій. Видалення дублікатів, заповнення пропусків і стандартизація тексту є складовими цього етапу.

Далі програма переходить до аналізу вхідного тексту, де використовуються методи, такі як розпізнавання мови, аналіз тональності і аналіз семантики для виявлення структури і значення тексту.

Останній етап - вибір відповіді - визначає найбільш релевантну відповідь для вхідного тексту, використовуючи методи машинного навчання, логіки і правил.

Важливим елементом є взаємодія між модулями програми через API, що дозволяє передавати дані між різними етапами обробки. Наприклад, модуль "Зчитування вхідного тексту" може використовувати API модуля "Аналіз вхідного тексту" для передачі даних для аналізу, а модуль "Аналіз вхідного тексту" використовує API модуля "Вибір відповіді" для передачі результатів для визначення відповіді[22].

Розширення програми можливе через додавання нових модулів, розширення існуючих або використання сторонніх інструментів. Це дозволяє програмі здатися на нові функції, такі як імпорт вхідного тексту з різних джерел, більший аналіз вхідного тексту або розширений спектр відповідей.

На першому етапі програма встановлює зв'язок із базою даних, де знаходяться відомості про зчитані файли і зроблені запити. Ця операція може бути здійснена за допомогою стандартних бібліотек або сторонніх інструментів.

Другий етап передбачає виконання запиту до бази даних для отримання інформації про зчитані файли і зроблені запити. Характер запиту може бути простим або складним, залежно від потреб користувача.

Обробка даних відбувається на третьому етапі, де програма аналізує та обробляє інформацію з бази даних. Це може включати фільтрацію, сортування та агрегацію даних для формування переліку зчитаних файлів і запитів.

Останній етап - формування переліку - передбачає створення переліку зчитаних файлів і запитів. Вихідні дані можуть представлятися у різних форматах, таких як текстовий список, таблиця або графік.

Взаємодія між модулями здійснюється за допомогою API, що дозволяє передавати інформацію між модулями. Наприклад, модуль "Зв'язок з базою даних" використовує API модуля "Запит до бази даних" для передачі даних про зв'язок з базою. Модуль "Запит до бази даних" в свою чергу взаємодіє з модулем "Обробка даних", передаючи результати запиту для подальшої обробки. Модуль "Обробка даних" використовує API модуля "Формування переліку" для передачі оброблених даних для створення переліку.

Такий підхід до роботи з базою даних може бути застосований у різних сценаріях, таких як система моніторингу для формування переліку зчитаних файлів або система управління даними для створення переліку запитів до бази даних[23].

2.5 Запропонований метод реалізації аналізу даних і їх візуалізації

У рамках виконання даного проекту, що націлений на розробку візуальних представлень даних, визначено, що етап здійснення механізмів зчитування інформації з різних джерел, таких як файли, бази даних та веб-сайти, виявляється важливим та невід'ємним складовим. Зазначені механізми виконують критичну роль у створенні доступу до різноманітних джерел інформації, що в подальшому використовується для формування візуальних репрезентацій даних. Ця фаза в процесі розробки вимагає ретельного налаштування та інтеграції механізмів, щоб забезпечити ефективний та

безперебійний збір інформації з різноманітних джерел. Невід'ємна роль цих механізмів полягає у створенні основи для подальшого аналізу та використання зібраної інформації у процесі створення візуальних відображень даних.

Цей механізм передбачає використання спеціальних алгоритмів для зчитування вмісту файлів у різних форматах, таких як CSV[23], Excel чи зображення. Формула зчитування може бути виражена як:

$$D_{файл}=Зчитати_{Файл}(Шлях)D_{файл}=Зчитати_{Файл}(Шлях),$$

де $D_{файл}$ *отримані дані з файлу.*

Для забезпечення доступу до інформації, збереженої в базі даних, використовується мова структурованих запитів SQL[23]. Формула може бути виражена як:

$$D_{БД}=SQL\_{Запит}(SELECT*FROM\ Таблиця),$$

Де DБД- дані з бази даних.

Механізм отримання даних з веб-сайтів, часто залучаючи техніки веб-скрапінгу, визначається як інструмент, що дозволяє програмі ефективно та систематично звертатися до веб-ресурсів з метою здобуття необхідної інформації з веб-сторінок. Процес використання веб-скрапінгу полягає в автоматизованому аналізі HTML-структури сторінок для екстрагування конкретних даних або інформації з них. Цей механізм дозволяє програмі ефективно взаємодіяти з інтернет-ресурсами та отримувати оновлену інформацію для подальшого використання. Важливим елементом цього процесу є розробка та впровадження формул, які визначають структуру та логіку отримання даних. У конкретному контексті, формула може виглядати як:

$$D_{веб}=Веб\_Скрапінг(Адреса\_сайту)\ D_{веб}=Веб\_Скрапінг(Адреса\_сайту),$$

Де $D_{веб}$- отримані дані з веб-сайту.

Такі механізми дозволяють проекту зчитувати інформацію з різних джерел, об'єднуючи її в єдиний набір даних, який потім використовується для генерації візуальних представлень. Цей процес забезпечує високий рівень гнучкості та адаптабельності проекту у відношенні до різноманітних джерел і форматів даних.

У контексті розробки проекту, спрямованого на генерацію візуальних представлень даних, процес подальшої внутрішньої структури даних після їх зчитування з різних джерел визначається потребами аналізу та ефективним використанням цієї інформації.

Отримані дані піддаються систематичному структуруванню для забезпечення ефективного та цілеспрямованого аналізу. Цей етап включає в себе важливий процес групування та категоризації інформації відповідно до її природи та взаємозв'язків. Для досягнення цієї мети використовуються об'єкти, класи або інші структури даних у програмному коді, що дозволяють систематизувати та організувати дані в зрозумілій формі.

Процес групування та категоризації даних дозволяє визначити логічні зв'язки між різними елементами інформації, що сприяє в подальшому аналізі та використанню даних. Наприклад, в процесі роботи над проектом, який включає обробку числових та текстових даних, може бути введена система типізації для чіткого визначення та розділення цих двох типів інформації. Це дозволяє не лише підвищити читабельність та структурованість даних, але й створює зручний фундамент для подальших етапів аналізу та використання інформації в рамках проекту.

Процес нормалізації визначається як важлива фаза обробки даних, яка спрямована на приведення їх до стандартного формату, метою чого є уникнення розбіжностей. Цей етап включає в себе адаптацію інформації до визначених стандартів та правил, зокрема, стосовно формату, одиниць вимірювання, або інших ключових параметрів. Під час нормалізації важливо

враховувати внутрішню сумісність даних, що забезпечує їхню послідовність у подальших операціях обробки та аналізу.

Фільтрація, у свою чергу, представляє собою інший етап обробки даних, який може бути використаний для видалення непотрібної або неправильної інформації з набору даних. Цей механізм дозволяє відокремити значущі дані від шуму чи надлишкової інформації, покращуючи якість та релевантність даних. Застосування фільтрації може бути направлене на видалення артефактів, помилок або дублікатів, забезпечуючи чистоту та точність обробленого датасету[24]. Цей процес є ключовим для забезпечення надійності та достовірності інформації, що використовується в подальших етапах аналізу чи моделювання.

В контексті обробки даних важливим кроком є створення структури, яка відповідає вимогам ефективної та зрозумілої візуалізації інформації у різних формах. Після обробки, дані мають придбати формат, який сприяє не лише зручності сприйняття, але й ефективному використанню у візуальних відображеннях. Наприклад, якщо в обробці використовуються числові дані, їхню підготовку можна здійснити так, щоб легко будувати діаграми чи графіки, що ілюструють відповідні відносини та тенденції.

В залежності від природи самих даних, розглядається можливість використання специфічних структур для оптимізації аналізу. Наприклад, для графічних даних може бути застосована матриця чи інші відповідні структури даних, що дозволяють ефективно виражати взаємозв'язки та характеристики графічних елементів.

Такий підхід до внутрішньої структури даних спрямований на підготовку інформації для подальшого візуального відображення. Це забезпечує не лише зручний та логічний аналіз, але й зберігає високу якість та точність даних, сприяючи надійній та інформативній візуалізації результатів обробки.У контексті розробки проекту, який орієнтований на генерацію візуальних представлень даних, використання нейромереж для автоматичного виявлення патернів та закономірностей стає ключовим елементом, що сприяє

автоматизації аналізу та оптимізації обробки інформації.

Використання нейромереж дозволяє створити систему, яка в самостійному режимі здатна виявляти складні патерни та взаємозв'язки в наборі даних. Нейромережі, моделюючи нейронні зв'язки, виявляють абстракції та залежності, що можуть залишитися непоміченими при традиційних методах аналізу.

Нейромережі володіють унікальною властивістю самостійного навчання визначенню закономірностей в наборі даних, уникаючи використання жорстко визначених правил. Ця характеристика набуває особливого значення в умовах, де патерни є складними чи зазнають змін з плином часу. Здатність нейромереж самостійно адаптуватися до нових умов робить їх ефективним інструментом для виявлення невідомих аспектів даних.

Нейромережі дозволяють здійснювати аналіз даних у різних форматах, таких як текст, зображення, аудіо тощо. Це забезпечує можливість розглядати дані з різних точок зору і виявляти складні взаємозв'язки між різними типами інформації.

Здатність нейромереж аналізувати та виявляти патерни в даних сприяє ефективній підготовці інформації для візуалізації. Моделі можуть автоматично виділяти ключові аспекти даних, що слугує основою для подальшої генерації зручних та інформативних візуальних елементів.

Нейромережі виявляються гнучким інструментом для виявлення патернів в умовах змінюючихся даних. Завдяки навчанню з плином часу, вони здатні враховувати та реагувати на еволюцію патернів.

В цілому, використання нейромереж для автоматичного виявлення патернів відкриває нові перспективи для швидкого та ефективного аналізу даних, що є ключовим елементом для успішного впровадження проекту з генерації візуальних представлень.

В контексті конкретного проекту, спрямованого на генерацію візуальних представлень даних, адаптація нейромереж є важливим етапом, спрямованим на досягнення максимальної ефективності та точності вирішення завдань,

пов'язаних із згаданим проектом.

Перш і найважливіше, для адаптації нейромереж важливо чітко визначити завдання проекту. У випадку генерації візуальних представлень даних, це може включати в себе завдання класифікації, сегментації, генерації зображень чи інші аспекти, які необхідні для досягнення мети проекту.

Вибір архітектури нейромережі визначається характером даних та поставленими завданнями. Наприклад, для аналізу зображень може бути використана конволюційна нейромережа (CNN) [25], тоді як для послідовностей даних, таких як текст, може бути ефективною рекурентна нейромережа (RNN) [25] чи їх комбінації.

Для досягнення високої точності, нейромережа повинна бути навчена на відповідних даних. Використання навчального набору, який належним чином відображає особливості даних проекту, сприяє адаптації моделі до конкретного середовища.

Нлаштування гіперпараметрів нейромережі є ключовим етапом адаптації. Це включає в себе вибір швидкості навчання, кількості шарів, розміру пакетів, функції активації та інші параметри. Оптимізація гіперпараметрів дозволяє досягти оптимальної продуктивності моделі.

Після навчання моделі необхідно провести процес валідації на даних, які не використовувались під час навчання, для оцінки її загальної ефективності. Також важливо забезпечити, що модель працює ефективно на реальних тестових даних, щоб гарантувати її застосовність в реальних умовах.

Після успішної адаптації моделі, важливо враховувати процес розгортання на виробничому середовищі та оптимізацію її швидкості для реального часу.

Всі ці етапи дозволяють ефективно адаптувати нейромережі до конкретного проекту, забезпечуючи оптимальну продуктивність та точність для генерації візуальних представлень даних у різних формах.

Алгоритми нейромереж в сфері аналізу даних базуються на фундаментальних математичних концепціях, що дозволяють моделям розпізнавати складні закономірності та патерни у великих обсягах інформації.

Одним із ключових аспектів є математична основа, на якій ґрунтуються алгоритми нейромереж.

Лінійна Регресія та Функція Втрат[26]. Початковим етапом є розгляд лінійної регресії, що є основою для багатьох алгоритмів машинного навчання. У контексті нейромережі це можна виразити формулою.

$$y=wx+b,$$

де y - вихід моделі, x - вхід,w - ваги, b - зсув, який визначає зсув відносно початку координат.

Функція втрат визначає, наскільки відповіді нейромережі відрізняються від дійсних значень. Однією з найпоширеніших є середньоквадратична помилка (MSE).

$$MSE=n1'\sum i=1n'(yi'-y\hat{}'i')2,$$

де n - кількість прикладів, yi' - дійсне значення

Функції Активації. Функції активації дозволяють введеному сигналу пройти через нейрон та визначити його вихід. Сигмоїдна функція активації виглядає наступним чином.

$S(x)=1+e-x1'$, де e - число Ейлера.

Зворотнє Розповсюдження Похибки (Backpropagation). Алгоритм зворотнього розповсюдження похибки є ключовим для тренування нейромережі. Його можна описати наступним чином.

$$(l)'=wij(l)'-\eta\partial wij(l)'\partial E',$$

де wij(l) ' - вага, η - швидкість навчання, wij(l) 'дE' - часткова похідна функції втрат відносно ваги.

Тестування та Валідація це важливий аспект після розробки алгоритму. Після тренування моделі важливо провести тестування та валідацію для оцінки її продуктивності. Для цього можна використовувати такі метрики, як точність чи F1-мера, що розглядають відповідність між прогнозованими та реальними значеннями.

Математична основа алгоритмів нейромереж є важливим елементом розуміння їхньої роботи та вдосконалення їхньої продуктивності в контексті аналізу даних та генерації візуальних представлень.

Тестування алгоритмів нейромереж важливо для перевірки їхньої ефективності та адаптації до конкретного проекту. Давайте розглянемо приклад застосування описаних формул та проведемо тестування на малих наборах даних для ілюстрації.

Розглянемо Бінарну Класифікацію[26]. Нехай у нас є модель для бінарної класифікації (дві класи. 0 і 1). Розглянемо формулу середньоквадратичної помилки (MSE) для одного прикладу.

$$MSE=21'\sum i=11' (yi'-y\hat{}'i\ ')2,$$

де уі ’ - дійсне значення, у^ ’0і ’ - прогнозоване значення.

Розглянемо три приклади.

Приклад 1. 1=1у1 ’=1, 1^1=0.8у^ ’1 ’=0.8

Приклад 2. 2=0у2 ’=0, 2^2=0.2у^2=0.2

Приклад 3. 3=1у3 ’=1, 3^3=0.6у^3=0.6

Підставимо значення в формулу MSE.

$$MSE=12((1-0.8)2+(0-0.2)2+(1-0.6)2)MSE=21((1-0.8)2+(0-0.2)2+(1-0.6)2)$$

MSE =12(0.04+0.04+0.16) MSE=2(0.04+0.04+0.16)

MSE =12·0.24 MSE=21·0.24

MSE =0.12 MSE=0.12

Таким чином, середньоквадратична помилка для цього набору прикладів дорівнює 0.12.

Таблиця 2.2 Тестування першого методу

Приклад	yi	y^i	2(yi−y^i)2
1	1	0.8	0.04
2	0	0.2	0.04
3	1	0.6	0.16
Сума			0.24

Ця таблиця дозволяє визначити значення помилки для кожного окремого прикладу та підсумовує їх для отримання загальної помилки моделі. Це лише один етап тестування, аналіз і визначення метрик може бути більш складним, враховуючи різні аспекти, такі як точність, чутливість, специфічність тощо.

Зважаючи на те, що ви згадали кілька концепцій, давайте розглянемо ще один приклад з іншою формулою – цього разу використовуючи кросс-ентропійну функцію втрат для бінарної класифікації.

Розглянемо приклад Кросс-ентропійна[27] функції втрат для бінарної класифікації.

CE=−N1∑i=1N(yi·log(y^i)+(1−yi)·log(1−y^i)),

де N - кількість прикладів.

Розглянемо три приклади.

Приклад 1. 1=1y1=1, 1^1=0.8y^1=0.8

Приклад 2. 2=0y2=0, 2^2=0.2y^2=0.2

Приклад 3. 3=1y3=1, 3^3=0.6y^3=0.6

Підставимо значення в формулу кросс-ентропійної функції втрат.

CE=−13(1·logf_0(0.8)+0·logf_0(1−0.2)+1·logf_0(0.6))CE=−31(1·log(0.8)+0·log(1−0.2)+1·log(0.6))

CE =−13(−0.2231+0+−0.5108)CE=−31(−0.2231+0+−0.5108)

CE =13·0.7339CE=31·0.7339

CE =0.2446CE=0.2446

Таблиця 2.2 Тестування другого методу

Приклад	yi	y^	yi·log(y^i)+(1−yi)·log(1−y^i)
1	1	0.8	-0.2231
2	0	0.2	0
3	1	0.6	-0.5108
Сума			0.7339

Ця таблиця ілюструє розрахунок кросс-ентропійної функції втрат для кожного окремого прикладу та підсумовує їх для отримання загальної втрати моделі.

В рамках проекту, що передбачає генерацію візуальних представлень даних, обрання типів візуальних представлень залежить від характеру даних та мети їх відображення. Різноманітність типів візуальних представлень важлива для ефективного сприйняття та аналізу інформації. Нижче розглянемо обрані типи візуальних представлень для різних видів даних:

- зображення теплової карти використовується для візуалізації розподілу значень у двовимірному просторі. Колірна градація дозволяє виділити інтенсивність чи концентрацію даних;
- гістограма. відображає розподіл інтенсивності значень у зображенні. Допомагає аналізувати рівномірність чи асиметрію даних.

Також є над важливим етапом це тектові дані, які обробляються особливим методом, а саме:

- облікова таблиця, яка використовується для відображення структурованих текстових даних у вигляді таблиці. Дозволяє порівнювати та фільтрувати дані;
- діаграма розсіювання слів, яка використовується для аналізу зв'язків між словами в тексті. Кожна точка представляє слово, а відстань між

точками вказує на частоту спільного вживання.

Не менш важливим моментом зчитування є числові данні користувача, і ті що зчитуються в процесі:

- лінійний графік, який використовується для відображення зміни числових значень відносно іншого параметру (часу, кількості спостережень тощо).
- кругова діаграма, яка відображає відсоткове співвідношення частин до цілого. Зручна для представлення структури даних.

Мультимедійні дані візуалізація яка є найбільш важкою в реалізації, але є одна з найважливіших аспектів і вирішенні проблеми відображення даних в потрібній формі:

- презентація слайдів яка використовується для послідовного представлення зображень, тексту та інших мультимедійних елементів;
- відеоінфографіка, яка комбінує відео та графічні елементи для роз'яснення складних концепцій або структури даних.

Обрані типи візуальних представлень враховують специфіку кожного типу даних та дозволяють ефективно передавати інформацію користувачеві в зручній та зрозумілій формі. Використання різних типів візуальних елементів підсилює сприйняття та розуміння обраного контенту.

У проекті, що передбачає генерацію візуальних представлень за результатами аналізу, використовується автоматизований підхід для створення зручних та інформативних візуальних елементів. Автоматизація цього процесу є важливою складовою для ефективного відображення та розуміння отриманих даних.

Аналіз даних, що включає в себе виявлення патернів, статистичних закономірностей та інших суттєвих параметрів, визначає фундамент для автоматизованої генерації візуальних представлень, що є важливим етапом у розвитку інструментів обробки та аналізу даних. Цей аналіз є ключовим для

виявлення внутрішніх зв'язків і характеристик даних, роблячи їх доступними та зрозумілими для користувачів у вигляді графічних компонентів.

На основі проведеного аналізу, система автоматично генерує відповідні візуальні компоненти, спрощуючи та оптимізуючи процес розробки візуальних елементів та полегшуючи взаємодію з отриманими даними. Цей процес може включати створення графіків для числових даних, формування теплових карт для ілюстрації інтенсивності чи взаємозв'язків, а також розробку таблиць для представлення структурованих даних.

Залежно від характеру та властивостей аналізованих даних, система визначає найбільш відповідні типи візуальних представлень для кожного конкретного випадку. Ця автоматизована генерація візуальних компонентів допомагає відсунути від користувача необхідність вручну вибирати та конфігурувати візуальні елементи, що робить процес аналізу даних більш ефективним, а представлення результатів - більш доступним та інтуїтивним.

Інтерфейс користувача дозволяє вводити параметри та обирати варіанти візуалізації, але з основною метою автоматизованої системи є використання алгоритмів та методів штучного інтелекту для оптимального вибору та створення візуальних компонентів.

Цей підхід має за мету полегшити процес роботи з даними та забезпечити швидке та точне отримання візуальних представлень для кращого розуміння та прийняття рішень на основі результатів аналізу.

Важливим аспектом є здатність користувача взаємодіяти з візуальними компонентами, змінювати параметри візуалізації та отримувати детальну інформацію. Наприклад, в інтерактивних графіках можна відзначити області для подальшого аналізу, змінювати масштаб чи фільтрувати дані. У таблицях можна сортувати, фільтрувати та додавати підсумки.

Крім того, важливо передбачити можливість зберігання та обміну інтерактивними візуальними елементами між користувачами, щоб сприяти спільній роботі та обговоренню результатів.

Поєднання модулів у систему в даному контексті передбачає взаємодію

між модулем аналізу, зчитування та візуалізації даних та іншими модулями проекту. Це забезпечує синергію функцій системи та використання всіх її компонентів для максимально ефективного аналізу та відображення результатів.

Такий підхід дозволяє користувачам взаємодіяти з аналітичними та візуалізаційними інструментами, максимально використовуючи їхні можливості та сприймаючи інформацію зручним інтерактивним способом.

Взаємодія модуля аналізу, зчитування і візуалізації даних з іншими модулями проекту є ключовим аспектом для успішного функціонування системи. Модуль аналізу виконує основну роль у розумінні та інтерпретації отриманих даних. Він відповідає за виявлення патернів, трендів і закономірностей у вхідних інформаційних потоках.

Зчитування даних є першим етапом у цьому процесі, де інформація завантажується в систему з різних джерел, таких як файли, бази даних та веб-сайти. Модуль зчитування повинен забезпечувати ефективний і безперервний потік даних, а також враховувати можливість роботи з різними форматами даних та їх оптимізацію для подальшого аналізу.

Отримані на етапі аналізу результати надходять на модуль візуалізації, який відповідає за створення різноманітних візуальних представлень. Це може бути представлення у вигляді зображень, таблиць, презентацій чи інфографіки, залежно від специфіки даних та потреб користувача. Мета цього етапу полягає в тому, щоб забезпечити зручне та інтуїтивне сприйняття аналітичних результатів, роблячи їх доступними та зрозумілими для широкого кола користувачів.

Важливим аспектом взаємодії є передача інформації між модулями. Модуль аналізу повинен надавати зрозумілі дані для візуалізації, ураховуючи потреби користувача. З свого боку, модуль візуалізації може взаємодіяти з модулем аналізу для забезпечення коректного відображення результатів аналізу.

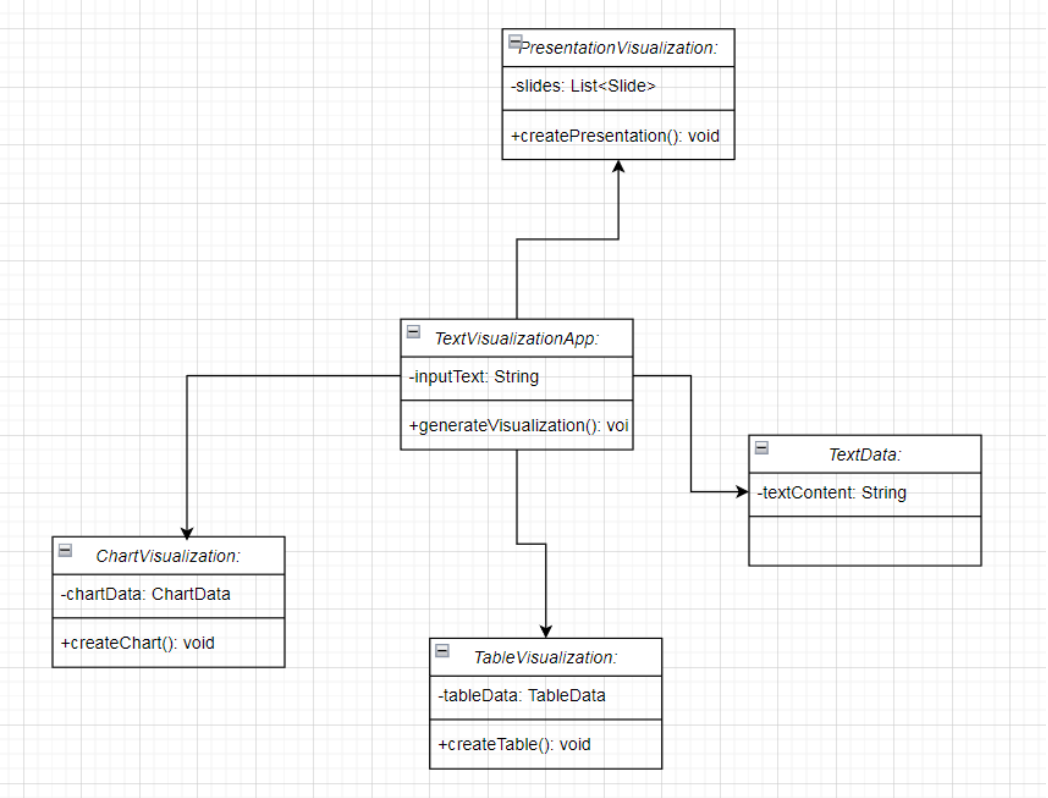


Рисунок 2.1 – Діаграма класів

Ця взаємодія має бути побудована на прозорих інтерфейсах та стандартних форматах обміну даними для забезпечення сумісності між різними модулями. Такий підхід дозволяє системі працювати як єдина інтегрована одиниця, де кожен модуль виконує свою унікальну функцію, але взаємодіє з іншими для досягнення загальних цілей проекту. Обробка вихідних даних аналізу для подальшого використання у візуалізації є важливим етапом у розробці проекту, спрямованого на автоматизовану генерацію візуальних представлень. Після проведення аналізу даних, отримані результати повинні бути оброблені таким чином, щоб їх можна було ефективно відобразити у різних формах, таких як зображення, таблиці, презентації та інфографіка.

Однією з ключових задач обробки даних є фільтрація і відбір інформації, яка є найбільш значущою для подальшого візуального представлення. Це може включати в себе відсіювання непотрібних даних, виявлення та врахування аномалій чи виокремлення ключових показників. Обробка також

включає в себе сортування даних за певними критеріями, агрегацію і групування для отримання більшої зрозумілості інформації.

Обробка вихідних даних аналізу для подальшого використання у візуалізації є важливим етапом у розробці проекту, спрямованого на автоматизовану генерацію візуальних представлень. Після проведення аналізу даних, отримані результати повинні бути оброблені таким чином, щоб їх можна було ефективно відобразити у різних формах, таких як зображення, таблиці, презентації та інфографіка.

Однією з ключових задач обробки даних є фільтрація і відбір інформації, яка є найбільш значущою для подальшого візуального представлення. Це може включати в себе відсіювання непотрібних даних, виявлення та врахування аномалій чи виокремлення ключових показників. Обробка також включає в себе сортування даних за певними критеріями, агрегацію і групування для отримання більшої зрозумілості інформації.

Після обробки дані готуються для подальшої візуалізації відповідно до вимог користувача та характеристик проекту. Для цього можуть використовуватися різні методи візуалізації, в залежності від типу та структури даних.

Розглянемо порівняльну таблицю, яка ілюструє основні етапи обробки даних для подальшої візуалізації.

Таблиця 2.3 Етапи виконання основного модуля

Етап обробки	**Завдання**	**Методи та техніки**
Фільтрація	Відбір значущої інформації	Відсіювання аномалій, відбір ключових показників
Сортування	Організація даних за певними критеріями	Сортування за зростанням чи спаданням значень
Агрегація	Об'єднання даних для зменшення обсягу	Сумування, середнє значення, групування
Підготовка	Підготовка даних для візуалізації	Форматування, конвертація, стандартизація

Можливості розширення функціоналу модуля для нових завдань представляють собою важливий аспект у розробці проекту, спрямованого на автоматизовану генерацію візуальних представлень даних. Для забезпечення універсальності та адаптивності модуль повинен мати гнучку структуру, яка дозволяє легко впроваджувати нові завдання без значних модифікацій.

Однією з можливостей розширення функціоналу є додавання нових алгоритмів обробки та аналізу даних, які враховують специфіку конкретних завдань користувача. Наприклад, для візуалізації медичних даних можна впровадити алгоритми для автоматичного виявлення паттернів або аномалій.

Додатковою можливістю розширення є підтримка нових форматів вхідних даних та вихідних візуальних представлень. Наприклад, можливість обробки та візуалізації тривимірних зображень або використання нових типів графіків для кращого відображення певних даних.

Використання сторонніх бібліотек та інструментів для підтримки є ключовим елементом у розробці проекту, спрямованого на генерацію візуальних представлень даних за запитом користувача. В даному контексті, це передбачає інтеграцію зовнішніх засобів для оптимізації роботи модуля та розширення можливостей обробки та відображення інформації.

Один з можливих підходів полягає в використанні спеціалізованих бібліотек для обробки графіків та візуалізації даних. Наприклад, бібліотеки, такі як Matplotlib[27] чи Plotly для мов програмування Python, можуть забезпечити широкі можливості стосовно створення різноманітних графіків, діаграм та інших візуальних представлень.

Також, для оптимізації алгоритмів та роботи з нейромережами можна використовувати спеціалізовані бібліотеки для машинного навчання, наприклад, TensorFlow чи PyTorch[27]. Це дозволить підвищити ефективність обробки та аналізу даних за допомогою нейромереж.

Підсумовуючи, використання сторонніх бібліотек та інструментів спрямоване на забезпечення високої якості та швидкодії модуля, що реалізує генерацію візуальних представлень даних у проекті.

3 РОЗДІЛ. РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ANDROID-ДОДАТКУ ДЛЯ ЗЧИТУВАННЯ І ВІЗУАЛІЗАЦІЇ НАДВЕЛИКОГО ОБ'ЄМУ ДАНИХ

3.1 Варіантний аналіз та обґрунтування вибору способів реалізації програмного засобу

Для успішного впровадження запропонованої моделі в програмному середовищі необхідно врахувати чотири ключові фактори. Обрана платформа для розробки має значущий вплив на цільовий ринок та споживачів продукту. Вибір між різними платформами може визначити, наскільки ефективно та зручно користувачі можуть взаємодіяти з моделлю. Наприклад, обрання веб-платформи може забезпечити доступність через веб-браузери та забезпечити широкий охоплення користувачів, у той час як мобільна платформа може бути вибрана для спрямування на аудиторію, яка використовує мобільні пристрої.

Мова програмування для розробки може вплинути на продуктивність розробки та якість програми. Вибір мови повинен бути обгрунтованим і враховувати вимоги проекту та особливості самого розроблюваного програмного продукту.

Середовище розробки, це важливий вибір середовища розробки може полегшити процес розробки та знизити кількість помилок. Воно повинно бути інтегрованим, забезпечувати зручні інструменти для відладки, контролю версій та автоматизації процесу розробки, щоб сприяти стабільності та продуктивності команди розробників.

Система управління базою[28] даних системи управління базою даних може суттєво полегшити роботу з даними та забезпечити ефективне функціонування додатку. Вибір певної системи повинен ґрунтуватися на потребах проекту, обсязі та типі даних, які будуть оброблятися, а також на вимогах до швидкодії та надійності системи.

3.1.1 Обгрунтування вибору платформи для розробки

З метою забезпечення доступності розробленого додатка для різних

категорій користувачів, стратегічним рішенням визнано створення мобільного застосунку. Це обране рішення спрямоване на забезпечення широкого охоплення аудиторії та відповідає ключовим вимогам цільового користувацького сегмента. Вирішення розробки мобільного додатка визнає значущість мобільності та швидкого доступу до програми для користувачів. Мобільний додаток дозволяє ефективно взаємодіяти з програмою в режимі реального часу і забезпечує гнучкість використання на різних мобільних пристроях. Такий підхід до розробки не лише сприяє зручності використання для користувачів, але й враховує актуальні тенденції у сфері інформаційних технологій, де мобільність та доступність відіграють ключову роль у задоволенні потреб сучасного споживача.

На ринку існують дві основні мобільні платформи. Android і iOS. Для того, щоб зробити об'єктивний вибір, було проведено порівняння цих платформ, враховуючи їхні переваги та недоліки.

Android[28] - це мобільна платформа, яка базується на операційній системі Linux. Вона надає широкий функціонал для мобільних телефонів, планшетів та інших портативних пристроїв. Основною мовою програмування для Android є Java. Для розробки додатків використовуються такі інструменти, як Eclipse та Android Studio. Програми для Android компілюються за допомогою Android API і перетворюються в байт-код для віртуальної машини Android.

Різноманітність пристроїв важлива перевага для вибору. На платформі Android доступний широкий вибір смартфонів і планшетів для різних бюджетів та потреб користувачів. Ви знайдете пристрої різних форм-факторів та характеристик.

Діапазон функцій не менш важливий момент. Android пропонує різноманітні можливості, включаючи підтримку двох сім-карт, що дає можливість користуватися декількома номерами на одному пристрої.

Розширювана пам'ять це важливий аспект автоматизації. Багато Android-пристроїв підтримують використання карт пам'яті, що дозволяє легко

розширити внутрішню пам'ять для зберігання фотографій, відео та інших даних.

Інтуїтивний інтерфейс є перевагою. Меню на Android може трохи відрізнятися залежно від версії, але воно має структуру, яка легко розуміється користувачами, що полегшує навігацію.

Постійні оновлення залишають за собою великий спектр роботи. Android систематично випускає оновлення, які додають нові функції та поліпшують роботу пристроїв.

Синхронізація з іншими пристроями, можливість взаємодії з розробником і користувачами. Android пропонує зручну синхронізацію з іншими електронними пристроями, що спрощує обмін даними.

Магазин додатків це легкий доступ до робочих програм. Google Play Маркет надає доступ до великої кількості різноманітних додатків, включаючи безкоштовні та платні.

Відкритість системи, як скарб для розробника. Android - відкрита операційна система, що дозволяє користувачам та розробникам вносити зміни в систему та розробляти власні додатки.

Розмаїтість додатків зменшує поріг входження, але дає зелене світло для нових і талановитих розробник. Завдяки відкритості системи користувачі мають великий вибір додатків, розроблених спільнотою користувачів.

Універсальні роз'єми для доступу, дають можливості в взаємодії. Більшість Android-пристроїв використовують стандартні роз'єми для зарядки та передачі даних, що робить аксесуари більш доступними.

Також, важливо розглянути мінуси, які є суттєвими, але нівелюються великими плюсами.

Вразливість до загроз через відкритість системи. Однією з основних недоліків відкритої операційної системи є підвищена вразливість до загроз безпеці. Користувачам потрібно бути обережними та відстежувати, з яких джерел завантажуються додатки та яка мережа використовується, щоб уникнути можливих проблем з безпекою.

Споживання ресурсів є великим мінусом, адже через велику кількість девайсів втрачається оптимізація. Деякі додатки на Android можуть споживати багато системних ресурсів, що може вплинути на продуктивність пристрою та тривалість роботи батареї.

Проблеми з оновленнями є частим моментом серед даної системи. Один з недоліків Android полягає в тому, що не всі пристрої отримують оновлення до нових версій операційної системи. Виробники пристроїв відповідають за надання оновлень, і це може призвести до того, що багато пристроїв залишаються на застарілих версіях Android. Однак, варто відзначити, що оновлення зазвичай не мають кардинального впливу на функціональність пристроїв.

iOS[29] - це потужна мобільна операційна система, розроблена компанією Apple для своїх пристроїв. Вона характеризується високою продуктивністю, стабільністю та безпекою. iOS пропонує користувачам широкий вибір додатків, а також доступ до важливих функцій, таких як Siri, Apple Pay і iCloud. Операційна система регулярно отримує оновлення, які додають нові функції та покращують продуктивність. iOS також має свій екосистемний підхід, який дозволяє синхронізувати пристрої Apple, використовувати їх разом і отримувати доступ до ексклюзивних служб та додатків.

Розглянемо переваги операційної системи IOS, які сформують експертизу в застосуванні подальшої системи.

Інтуїтивно зрозуміле управління є дуже важливим аспектом. iOS відома своєю простотою та зручністю інтерфейсу, який дозволяє користувачам легко орієнтуватися та використовувати пристрої Apple.

Синхронізація з пристроями Apple славиться серед користувачів. iOS надає комфортну можливість синхронізації між різними пристроями Apple, незалежно від того, чи проводиться це вручну, чи в автоматичному режимі. Це дозволяє легко обмінюватися даними та документами між пристроями.

Підтримка оновлень і робота навіть з досить старими версіями

пристроїв. Apple регулярно випускає оновлення для своїх пристроїв та операційної системи iOS, що приносить покращення у роботу гаджетів та додає нові функції.

Apple Store обширний, але через закритість системи зменшується ризик шахрайства. В iOS існує власний магазин додатків Apple Store, де доступно величезне різноманіття програм. Багато з них є платними, але також існує велика кількість безкоштовних додатків.

Закритість операційної системи запорука захисту. Закрита природа iOS гарантує її стабільність та безпеку, оскільки тільки фахівці Apple можуть вносити зміни в систему. Це забезпечує високу якість та надійність операційної системи.

Якість фотографій та відео через якісні матриці та гарну оптимізацію програмної начинки. iOS пропонує високоякісні камери на пристроях Apple та додатки для обробки фото та відео, що дозволяє створювати вражаючі зображення та відеоматеріали.

Система голосового помічника Siri, яка досить ефектино вирішує всі потреби користувача. Siri дозволяє користувачам використовувати голосові команди для виконання різних завдань та отримання інформації.

Недоліки операційної системи IOS також присутні і вони досить сутєві по своїй суті.

Складності в передачі даних через закритість системи. Передача даних з пристрою на iOS на інший смартфон або ноутбук із різними операційними системами може призвести до незручностей або збоїв у сумісності, особливо коли мова йде про передачу файлів або документів.

Вбудована пам'ять без слоту для картки пам'яті значно обмежує можливості. Продукція Apple має вбудовану пам'ять, і відсутність слоту для картки пам'яті обмежує можливість розширення пам'яті на пристрої. Це може призвести до нестачі місця для даних та додатків.

Висока ціна деяких програм і не можливість застосувати тестових, для перевірки. В додатках та іграх у магазині Apple Store іноді можна зустріти

високі ціни, що робить їх менш доступними для користувачів.

Відсутність бюджетних пристроїв, через що низький поріг входження. Продукція Apple відома своєю високою якістю, але водночас вона часто вважається більш високою за ціною порівняно з іншими брендами, що може ускладнити доступ до бюджетних смартфонів.

Відсутність підтримки Flash значно обмежує тестування додатку. iOS не підтримує Adobe Flash, що може обмежувати доступ до деяких веб-сайтів та мультимедійного вмісту, який використовує Flash-технологію.

Відсутність розширених можливостей налаштувань для розробників для оптимізації. iOS може бути менш гнучкою щодо індивідуалізації та налаштувань порівняно з іншими операційними системами, що може не задовольняти деяких користувачів[28].

Отже, після уважного порівняльного аналізу мобільних операційних систем та зваживши на їхні плюси та мінуси, прийнято рішення віддати перевагу операційній системі Android для подальшої розробки. Це рішення ґрунтується на численних ключових перевагах Android, які переважають над конкурентами, а також на суттєвих недоліках інших систем, які можуть суттєво вплинути на процес та результат розробки.

3.1.2 Обґрунтування вибору мови програмування

Для створення програм під мобільну операційну систему, ми маємо вибір між двома найпоширенішими мовами програмування. Java і Kotlin. Перед прийняттям оптимального вибору необхідно провести аналіз та порівняння їх переваг і недоліків.

Мова програмування Java є однією з найпопулярніших і широко використовуваних мов програмування. Вона була створена компанією Sun Microsystems в 1996 році і пізніше була придбана компанією Oracle. Спочатку Java була задумана як універсальна мова для вирішення різних завдань і пройшла довгий шлях розвитку.

На сьогоднішній день останньою версією є Java 21, яка була випущена у

вересні 2023 року. Java перетворилася в повноцінну платформу та екосистему, яка поєднує різні технології для створення різних видів програмних рішень - від десктопних додатків до великих веб-порталів і сервісів. Вона також використовується для створення програмного забезпечення для різних пристроїв, включаючи комп'ютери, планшети, смартфони та побутову техніку.

Основним аспектом Java[29] є те, що її код спершу транслюється в байт-код, який є незалежним від платформи, а потім виконується віртуальною машиною JVM (Java Virtual Machine). Це дозволяє досягти кросплатформовості та апаратної переносимості програм, оскільки один і той же код може виконуватися на різних платформах без перекомпіляції. Кожна платформа може мати свою реалізацію віртуальної машини JVM, але всі вони можуть виконувати один і той самий байт-код.

Kotlin - це актуальна, статично типізована мова програмування, що швидко набуває популярності і була розроблена компанією JetBrains. Її використання може бути різноманітним і охоплює створення програм для різних платформ. Kotlin відкриває можливість розробки кросплатформового коду, який може бути використаний на різних операційних системах.

Мова програмування Kotlin відзначається своєю універсальністю та розширеними можливостями, що роблять її привабливою для розробників у різних сферах інформаційних технологій. Вона відкриває широкий спектр можливостей для створення не лише мобільних додатків, які оптимально працюють на платформах Android та iOS, але й веб-додатків, що дозволяє розгортати програми в онлайн середовищі. Надто, Kotlin виходить за рамки тільки клієнтської або серверної розробки, надаючи можливість створення як бекенду, так і фронтенду для повноцінних програм. Таким чином, розробники можуть використовувати цю мову для створення комплексних програмних рішень, що працюють на різних рівнях інфраструктури.

Kotlin[29] вирізняється не лише широким спектром платформ, для яких можна створювати програми, але й його адаптивністю до різних галузей розробки. Здатність використовувати Kotlin для створення десктопних

додатків робить його ідеальним інструментом для розробки програмного забезпечення, яке може бути впроваджене на різних пристроях та операційних системах. Це особливо актуально в галузі Data Science, де потрібна ефективна робота із великими обсягами даних. Kotlin надає зручні інструменти для обробки та аналізу даних, роблячи його популярним вибором серед фахівців, що працюють у цьому напрямку.

Таким чином, Kotlin стає не лише інструментом для розробки конкретних програм, але й важливим компонентом для тих, хто прагне залишатися на передових позиціях у світі програмування.

Однак, основною сферою застосування Kotlin із серед усіх платформ є розробка під операційну систему Android. Компанія Google офіційно визнала її однією з ключових мов для розробки додатків для Android, поруч з Java і C++. Крім того, інструменти для роботи з Kotlin були інтегровані в середовище розробки Android Studio, починаючи з версії 3.0.

Таблиця 3.1 Порівняльна характеристика

№	Характеристика	Java	Kotlin
1	Нульова безпека	Доступно. Kotlin має вбудовану нульову безпеку.	Недоступно. NullPointerException часто призводить до помилок розробки в Android і Java.
2	3. Статичні члени	У Kotlin можна використовувати об'єкт-компаньйон для створення статичних членів класу, але це призводить до складнощів в програмуванні.	В Java є підтримка статичних ключових слів для змінних, методів, блоків і вкладених класів.

Продовження таблиці 3.1

№	Характеристика	Java	Kotlin
3	Перевірені виключення	У Kotlin відсутні перевірені виключення (checked exceptions). Програмісти не зобов'язані обробляти виключення засобами мови.	В Java, перевірені виключення (checked exceptions) є обов'язковими для обробки. Програмісти повинні визначити або обробити такі виключення, щоб код компілювався.
4	Масштабованість	Kotlin дозволяє створювати розширювані та підтримувані за допомогою пакетів, модулів та багатошарових архітектур додатки. Він підтримує різні підходи до побудови додатків та взаємодії з іншими сервісами і бібліотеками. Проте краще підходить для простих програмних рішень.	Java має добру масштабованість і добре підходить для розробки великих проектів і підтримки розширення. Java-екосистема багата на різні бібліотеки та фреймворки, які сприяють розробці великих та складних додатків.

Отже, після проведення аналізу різних мов програмування для розробки

під операційну систему Android і їх порівняння за певними критеріями, було вирішено вибрати мову програмування Java. Це рішення прийнято на підставі переваг, які пропонує Java, і їх важливості для розробки під Android. Багато з цих переваг виявилися ключовими, і водночас, були виявлені суттєві недоліки іншої мови.

Отже, після проведення аналізу різних мов програмування для розробки під операційну систему Android і їх порівняння за певними критеріями, було вирішено вибрати мову програмування Java. Це рішення прийнято на підставі переваг, які пропонує Java, і їх важливості для розробки під Android. Багато з цих переваг виявилися ключовими, і водночас, були виявлені суттєві недоліки іншої мови.

3.1.3 Обґрунтування вибору середовища розробки

Перед тим як приступити до написання коду для додатку, необхідно обрати підходяще інтегроване середовище розробки (IDE) [30], оскільки це питання не менш важливе, ніж вибір мови програмування. Інтегроване середовище розробки (IDE) є програмним забезпеченням для створення програм, яке об'єднує різноманітні інструменти розробника в єдиний графічний інтерфейс користувача (GUI).

Зазвичай, IDE включає в себе:

- редактор вихідного коду. текстовий редактор, що сприяє написанню програмного коду за допомогою функцій, таких як підсвічування синтаксису, візуальні підказки та автозавершення для конкретної мови програмування;
- інструменти автоматизації збірки. утиліти, які автоматизують рутинні завдання, такі як компіляція вихідного коду в двійковий код та запуск автоматичних тестів;
- відладчик. програма для тестування та виявлення помилок у вихідному коді, яка може графічно відображати місцезнаходження помилок.

Розглянемо найпопулярніші інтегровані середовища розробки, які використовуються для створення Android-додатків на мові програмування Java, зокрема Android Studio, Eclipse та IntelliJ IDEA. Android Studio, розроблене Google, є потужним інструментом на основі IntelliJ IDEA Community Edition, розробленої компанією JetBrains. Це відкрите середовище розробки, поширюється під ліцензією Apache 2.0[30].

Eclipse[30] представляє собою інтегроване середовище розробки (IDE), використовуване в області комп'ютерного програмування. Це включає базову робочу область та систему розширюваних модулів для налаштування середовища. Набір розробки програмного забезпечення Eclipse (SDK), який включає інструменти розробки Java, призначений для розробників на мові Java. Користувачі можуть розширювати його можливості, встановлюючи модулі, які підключаються і написані для платформи Eclipse. Ці модулі можуть включати інструменти розробки для інших мов програмування, а також дозволяють користувачам створювати та додавати свої власні модулі, які підключаються.

IntelliJ IDEA представляє собою передове середовище швидкої розробки на мові Java, що включає високотехнологічний комплект тісно інтегрованих програмувальних інструментів. Його функціональність охоплює інтелектуальний редактор вихідного коду з передовими засобами автоматизації, потужні інструменти рефакторингу коду, і вбудовану підтримку технологій J2EE. Також, IntelliJ IDEA включає механізми інтеграції з тестуванням Ant/JUnit та системами керування версіями, унікальний інструмент оптимізації коду Inspection та інноваційний візуальний конструктор графічних інтерфейсів.

У таблиці 3.2 подано порівняльний аналіз обраних інтегрованих середовищ розробки згідно з вибраними критеріями.

Таблиця 3.2 Порівняння інтегрованих середовищ розробки

Критерій	Android Studio	Eclipse	IntelliJ IDEA
Безкоштовність використання	Так	Так	Ні
Регулярні оновлення Android	Так	Ні	Іноді
Швидкодія	Так	Іноді	Іноді
Невимогливість до продуктивності системи	Так	Іноді	Іноді
Магазин розширень	Так	Ні	Так
Підтримка мов	Так	Іноді	Так

За результатами аналізу провідних інтегрованих середовищ розробки для створення Android-додатків, який відображений у таблиці 3.2, можна зазначити, що Android Studio видається найбільш вдалим вибором серед розглянутих альтернатив для втілення Android-додатку, спрямованого на організацію самостійної роботи студента. Його переваги включають швидкість, безкоштовність, зручність та регулярні оновлення інструментів розробки Android.

3.1.4 Обґрунтування вибору системи управління базою даних

Основною метою додатку є реєстрація та отримання інформації про завдання, встановлені користувачем, з бази даних. Для вирішення цих завдань найбільш відповідним є використання реляційних баз даних. Реляційна база даних організована у вигляді таблиць, а операції над даними виконуються з використанням цих таблиць. Управління реляційною базою даних (РСУБД) використовується для створення та керування реляційною базою даних. Розглянемо три найпопулярніші РСУБД. PostgreSQL, MySQL та SQLite.

PostgreSQL є високофункціональною та потужною системою управління базами даних (СУБД), відомою своєю надійністю та відкритим вихідним кодом. Розроблений спільнотою ентузіастів, PostgreSQL прагне відповідати

стандартам SQL та забезпечувати високий рівень сумісності з іншими базами даних.

Основні характеристики PostgreSQL включають:

- об'єктно-реляційний підхід. PostgreSQL підтримує як реляційні, так і об'єктно-орієнтовані структури даних, що дозволяє розробникам зручно працювати з складними об'єктами та зв'язками між ними;
- розширена функціональність. Великий набір вбудованих функцій та розширень, включаючи геопросторові та текстові операції, роблять PostgreSQL потужним інструментом для різноманітних проектів;
- надійність та стійкість. PostgreSQL славиться своєю стійкістю та можливістю обробки великої кількості одночасних підключень. Також вона має вбудовані засоби відновлення даних, що забезпечують безпеку інформації;
- розширюваність. Здатність масштабуватися на вимогу робить PostgreSQL відмінним вибором для проектів будь-якого розміру, починаючи від невеликих додатків до великих корпоративних систем;
- відкритий вихідний код. PostgreSQL розповсюджується під ліцензією PostgreSQL, що дає вільний доступ до вихідного коду та сприяє активній спільноті розробників.

Завдяки цим характеристикам PostgreSQL стає відмінним вибором для проектів, де важлива висока продуктивність, надійність та розширюваність бази даних.

MySQL – це популярна відкрита система управління базами даних (СУБД), яка відзначається широким застосуванням у світі програмування та веб-розробки. До основних особливостей MySQL відносяться:

- відкритий вихідний код. MySQL розповсюджується під ліцензією GPL, що дозволяє вільне використання, модифікацію та розповсюдження його вихідного коду;
- легкість використання. MySQL володіє простим інтерфейсом та легкістю в налаштуванні, що робить його популярним вибором для початківців та

професіоналів;

- висока продуктивність. СУБД забезпечує ефективну обробку запитів та високу швидкість операцій, що робить його ефективним для широкого спектру застосувань;
- масштабованість. MySQL легко масштабується вгору, щоб відповідати потребам ростучих проектів та об'ємам даних;
- різноманітні інструменти та розширення. СУБД включає багато додаткових інструментів та розширень, таких як Workbench для візуального моделювання бази даних та різноманітні движки збереження;
- підтримка транзакцій. MySQL забезпечує механізм транзакцій, що дозволяє забезпечити консистентність та надійність операцій в базі даних.

MySQL[30] широко використовується в різних сферах, включаючи веб-розробку, вбудовані системи, корпоративні застосування та інші області, де потрібна надійна та швидка реляційна база даних.

SQLite – це легка та вбудовувана система управління базами даних (СУБД), яка відрізняється простотою використання та широкою популярністю в різних типах програмного забезпечення. Основні характеристики SQLite включають.

- вбудована архітектура. SQLite побудований так, щоб бути вбудованим безпосередньо в програму, що робить його ідеальним вибором для мобільних додатків та вбудованих систем;
- легкість використання. SQLite володіє простими SQL-запитами та мінімальними вимогами до конфігурації, що полегшує його використання для широкого кола розробників;
- низький обсяг використовуваної пам'яті. SQLite має невеликий обсяг використовуваної пам'яті та низький вплив на ресурси, що особливо важливо для мобільних та вбудованих пристроїв;
- швидкодія. У великій мірі через свою легку архітектуру, SQLite забезпечує швидкодію виконання операцій з базою даних;

- безпека даних. SQLite підтримує механізми шифрування для захисту конфіденційності даних;
- широкий спектр застосувань. Завдяки своїм характеристикам, SQLite використовується в мобільних додатках, вбудованих системах, браузерах, а також в інших областях, де потрібна проста та ефективна система управління базами даних.

Хоча SQLite не підходить для всіх типів проектів, але це потужний інструмент для додатків з обмеженими ресурсами та проектів, де простота та ефективність грають ключову роль.

Результати порівняння розглянутих систем управління базами даних за обраними критеріями наведено в таблиці 3.3.

Таблиця 3.3 Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	SQLite
Потужність та функціонал	Так	Обмежено	Так
Швидкість читання даних	Ні	Обмежено	Так
Механізми шифрування	Так	Так	Так
Використання пам'яті	Обмежено	Обмежено	Так
Вміст всієї бази даних в одному файлі	Ні	Ні	Так

Згідно з таблицею 3.3 можна зазначити, що система управління базами даних SQLite є найбільш підходящою серед розглянутих СУБД. Завдяки високій швидкості обробки даних та легкості використання, вона ідеально підходить для збереження структурованої інформації про завдання, визначені користувачем, і гарантує швидкий доступ до даних для формування звітів та статистики.

Отже, після проведеного аналізу засобів розробки визначено, що використання мови програмування Java, середовища розробки Android Studio та СУБД SQLite є оптимальним вибором для реалізації розроблюваного Android-додатку.

3.1 Розробка графічного інтерфейсу додатку

Для реалізації зовнішнього вигляду програми необхідно провести детальний аналіз та визначення елементів інтерфейсу, що будуть використовуватися, їхнього розташування на екрані та механізмів взаємодії з користувачем. Цей процес є ключовим етапом у розробці, оскільки визначає вигляд та функціональність користувацького інтерфейсу, що безпосередньо впливає на зручність та ефективність використання програми.

Першочерговою задачею є визначення необхідних елементів інтерфейсу, таких як кнопки, поля введення, списки, графічні елементи тощо. Для кожного елемента необхідно врахувати його призначення та функціональність в контексті користувальницького досвіду. Розташування елементів на екрані слід обирати таким чином, щоб забезпечити логічну структуру та зручну навігацію. Важливо також розглядати аспекти взаємодії з користувачем, включаючи реакцію на введення, анімацію, візуальні ефекти тощо. Детальне вивчення і врахування психології взаємодії дозволить створити інтуїтивно зрозумілий та привабливий інтерфейс.

Android Studio пропонує два способи створення інтерфейсу програми. Перший спосіб - це використання XML-розмітки. У цьому випадку ви визначаєте, які елементи інтерфейсу будуть використовуватися та як вони будуть розташовані, за допомогою тексту. Другий спосіб - це візуальне програмування. У цьому випадку ви можете перетягувати елементи інтерфейсу за допомогою миші та розміщувати їх на екрані.

Візуальне програмування є зручним способом створення простих додатків, але для складніших додатків може бути краще використовувати комбінований підхід. Цей підхід поєднує в собі переваги візуального програмування та ручного написання коду, що забезпечує більшу гнучкість та ефективність.

Під час розробки інтерфейсу даного додатку використовувався комбінований підхід. Спочатку були використані прості елементи візуального програмування для швидкого розміщення основних об'єктів на формі. Потім,

використовуючи XML-розмітку, були додані додаткові налаштування та функціональність, що забезпечило необхідний рівень гнучкості та контролю.

Цей підхід дозволив ефективно реалізувати інтерфейс і полегшив подальшу роботу з розширенням та удосконаленням функціоналу додатку.

Під час збірки Android-програми інструмент автоматичної збірки створює файл R.java, який містить ідентифікатори всіх ресурсів, визначених у програмі. Цей файл використовується для того, щоб програмі було відомо, які ресурси вона може використовувати. Однак не всі ресурси будуть використовуватися в кожній діяльності. Знак «+» у файлі XML використовується для того, щоб повідомити інструментам збірки, що потрібно створити новий ресурс.

У розробці Android-додатків часто використовують активності, які дозволяють розділити додаток на окремі вікна. У даному додатку було створено 6 основних вікон для реалізації активностей які можна застосувати в редагуванні візуалізацій. activity_crud, activity_detail, activity_diaries, activity_helper, activity_exam та activity_splash.

Вибір такого підходу дозволяє зменшити використання оперативної пам'яті пристрою під час роботи додатку. Це пов'язано з тим, що функціональні модулі додатку розподілені між активностями і працюють лише тоді, коли відкрито їх вікна.

Розробка інтерфейсу є важливим етапом у процесі розробки Android-додатку, оскільки переважна більшість користувачів в першу чергу звертають увагу на інтерфейс додатку, яким користуються, і віддають перевагу більш зручним у використанні та структурно зрозумілим програмам.

3.2 Програмна реалізація додатку

Кожен Android-додаток починає свою роботу з файлу AndroidManifest.xml, який автоматично створюється в корені проекту. У цьому файлі описуються компоненти додатку, вказуються дозволи на використання захищених частин API та взаємодію з іншими програмами, а також надається дозвіл стороннім програмам на доступ до ресурсів додатку.

В процесі ініціалізації та завантаження Android Runtime (ART), що є віртуальною машиною для виконання коду Android-додатків, відбуваються ключові кроки. Цей етап передбачає завантаження необхідних файлів для функціонування ART, таких як бібліотеки ART та файли із типовою інформацією.

Рисунок 3.1. Діаграма взаємодії користувача з додатком

Далі, наступає етап завантаження та ініціалізації самого додатку, під час якого ART [31] перевіряє файл APK для визначення компонентів додатку, завантажує їх та проводить ініціалізацію.

Після цього ART обирає основний компонент додатку та ініціює його запуск. Основний компонент, описаний у файлі AndroidManifest.xml як активність з тегом android.name=".MainActivity", є визначальним для взаємодії користувача з додатком.

На завершальному етапі відбувається виконання активності, яка представляє собою інтерфейс користувача та надає можливість взаємодії з додатком. Цей етап включає ініціалізацію інтерфейсу, обробку подій від користувача та виконання різноманітних команд, таких як onCreate(), onStart(),

onResume(), onPause(), onStop(), onDestroy(), які забезпечують потрібне функціональне забезпечення.

Усі ці етапи взаємодії Android Runtime із додатком включають в себе використання різноманітних команд, таких як art, dex2oat, PackageManager.getPackageInfo(), ActivityManager.getRunningActivities() та ActivityManager.startActivity(). Відмінності в командах можуть виникати внаслідок різних версій Android та індивідуальних підходів розробників при створенні додатків.

Елемент activity для активності SplashActivity визначає, що ця активність буде першою, яка буде запущена під час запуску додатка.

Елемент activity для активності DiariesActivity визначає, що ця активність буде основною активністю додатка.Елемент activity для активності CRUDActivity визначає, що ця активність буде використовуватися для редагування задач.

Елемент activity для активності DetailActivity визначає, що ця активність буде використовуватися для перегляду деталей задачі.

Елемент activity для активності HelperActivity визначає, що ця активність буде використовуватися як помічник для користувача.

Елемент activity для активності ExamActivity визначає, що ця активність буде використовуватися для перевірки знань користувача.

Елемент activity для активності RegistrationActivity визначає, що ця активність буде використовуватися для реєстрації користувача.

Елемент service для сервісу FileService визначає, що цей сервіс буде використовуватися для зчитування та аналізу файлів.

Елемент uses-permission для дозволу android.permission. INTERNET визначає, що додатку необхідний доступ до Інтернету для зчитування та аналізу файлів.

Процес завантаження анімації та запуску програми є ключовим етапом в життєвому циклі будь-якого Android-додатку, і це вимагає використання спеціального модуля, що містить в собі відповідний програмний код.

На початковому етапі, модуль завантажує файли анімації з файлу APK додатку, що представляє собою архів, що містить всі необхідні компоненти для додатку. Використовуючи метод Resources.openRawResource(), модуль отримує потік даних з файлу, реалізуючи таким чином завантаження файлів анімації.

Далі йде етап декодування цих файлів, що полягає у перетворенні їх у формат, розпізнаваний комп'ютером. Використовуючи метод AnimatorInflater.loadAnimator(), модуль отримує об'єкт Animator, що представляє собою анімацію, готову до відтворення.

Після завантаження та декодування, анімація може бути запущена за допомогою методу Animator.start(). Таким чином, розпочинається відтворення анімаційних ефектів.

Важливим є також етап завантаження файлів програми, який включає в себе отримання байт-коду додатку. Застосовуючи метод ClassLoader.loadClass(), модуль завантажує клас програми, після чого можливо створення об'єкта програми за допомогою методу Class.newInstance().

Завершальним етапом є запуск самої програми за допомогою методу Activity.start(). Після виконання цих етапів, модуль успішно завантажує анімацію та запускає програму, готові до взаємодії з користувачем. Однак, варто відзначити, що конкретна реалізація цих етапів може різнитися залежно від конкретного модуля, що використовується в додатку.

Після завершення анімації старту додатку, головне меню або, інакше кажучи, «DiariesActivity», стає доступним для користувача. Щоб розпочати роботу, необхідно спочатку авторизуватись в базі, щоб ваші дані і файли були в безпеці.

Вхід в обліковий запис та авторизація в системі визначають важливі етапи функціоналу будь-якої програми. Виконання цих завдань передбачає використання спеціального модуля, який включає в себе необхідний програмний код для взаємодії з обліковим записом та системою авторизації.

На першому етапі модуль отримує від користувача необхідні для входу

в обліковий запис дані, такі як логін і пароль. Застосовуючи методи, такі як EditText.getText(), Spinner.getSelectedItem() або RadioButton.isChecked(), модуль здійснює збір цих даних.

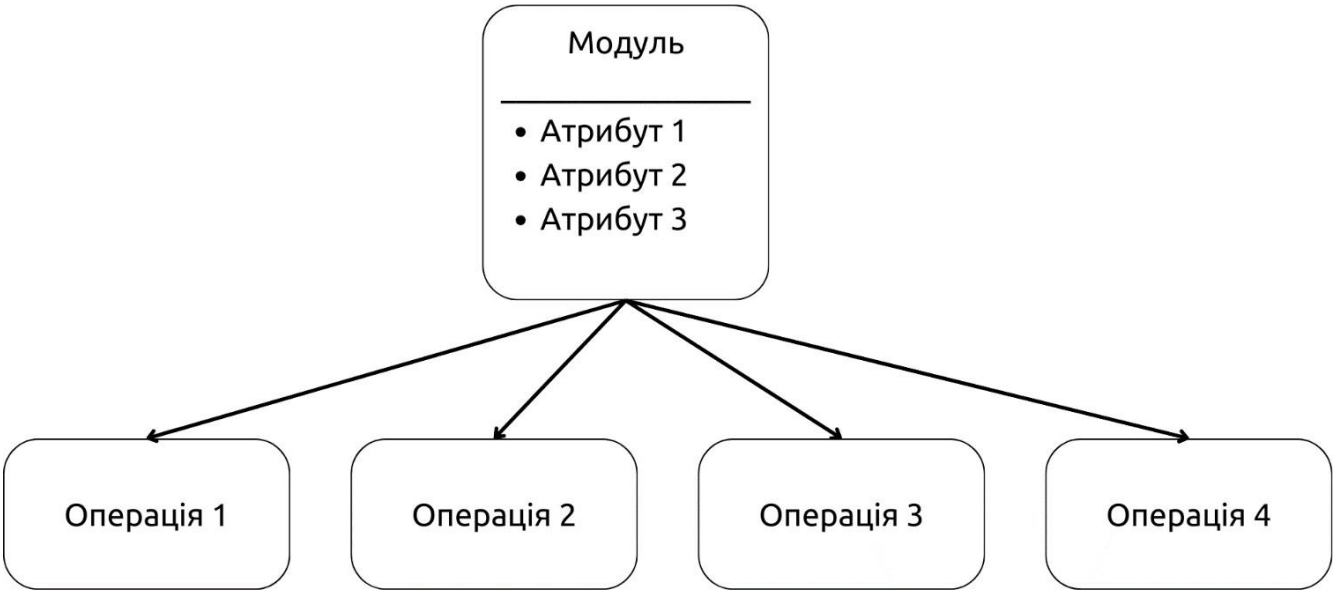


Рисунок 3.2. Модуль аналізу відповіді для користувача

Далі наступає етап перевірки наданих користувачем даних. Це включає в себе перевірку коректності даних та наявність облікового запису з вказаними параметрами. Використовуючи методи, такі як RegexMatcher.matches(), DatabaseHelper.getUser() або Server.checkCredentials(), модуль проводить цю перевірку.

Якщо дані коректні та відповідають наявному обліковому запису, модуль авторизує користувача в системі. Для цього використовуються методи, наприклад, SharedPreferences.edit() або TokenManager.setToken(), що дозволяють позначити користувача як авторизованого.

Останнім етапом є перенаправлення користувача на головну сторінку системи після успішної авторизації. Це виконується за допомогою методу Intent.setClass(), який налаштовує напрямок переходу.

Ці етапи є загальними для різноманітних модулів, що реалізують вхід в обліковий запис та авторизацію в системі. Проте, конкретні реалізації можуть відрізнятися в залежності від призначення самого модуля.

Ця функція спрямована на створення вікна авторизації в додатку, адже відбувається обробка файлів, і щоб доступ був відкритий тільки для себе, було прийнято рішення обмежити доступ. Функціональність авторизації через гугл пошту реалізовано через метод "listenToBottomNavigationClicks".

Ця функція дає можливість прив’язати до вашого облікового запису гугл пошту, таким чином це значно зменшить поріг входження в додаток і покращить зручність. Далі ми детально розглянемо функції, пов'язані з діяльністю "CRUDActivity", яка відповідає за створення, роботу головної сторінки і роботи з чато ботом.

Проаналзувавши, користувач може вписати запит для отримання інформації з файлу.

Одна з основних задач є аналіз даних і їх візуалізація в різних видах. Тому найобширніша є функція "MainViz", яка з запиту зчитує що потрібно зробити, аналізує все що є в документів, виконує пошук, визначає який тип візуалізації найкраще підійде і виконує це. Це можуть бути порівняльні таблиці, презентації, різноманітні діаграми і графіки.

Функція, яка збирає всі проаналізовані файли і запити, які були по них введені, щоб була можливість їх переглянути і повторити, може бути дуже корисною для користувачів. Завдяки цій функції користувачі можуть переглянути результати своїх попередніх запитів, щоб уточнити або розширити їх, або повторити їх, щоб отримати нові результати.

Функція може зберігати інформацію про проаналізовані файли та запити в різних форматах, наприклад, в базі даних, в текстовому файлі або в JSON-об'єкті. Вибір формату зберігання залежить від конкретних вимог до функції.

Далі розглянемо функцію "history_voch," яка відповідає за створення історії всіх зібраних файлів і запитів.

Функція може бути використана в різних областях, наприклад, в наукових дослідженнях, в бізнесі, в освіті. Наприклад, у наукових дослідженнях функція може використовуватися для зберігання інформації про аналіз даних, отриманих в результаті експериментів. У бізнесі функція може

використовуватися для зберігання інформації про аналіз продажів, маркетингу або інших аспектів бізнесу. У освіті функція може використовуватися для зберігання інформації про аналіз навчальних матеріалів.

Учень використовує функцію для аналізу тексту на наявність помилок. Він вводить запит "Знайти всі помилки в тексті". Функція аналізує текст і знаходить помилки. Учень може переглянути результати аналізу, щоб уточнити або розширити свій запит.

Бізнесмен використовує функцію для аналізу продажів. Він вводить запит "Знайти тенденції в продажах за останній квартал". Функція аналізує дані про продажі і знаходить тенденції. Бізнесмен може переглянути результати аналізу, щоб прийняти рішення про майбутні продажі.

Вчений використовує функцію для аналізу даних, отриманих в результаті експерименту. Він вводить запит "Знайти кореляцію між двома змінними". Функція аналізує дані і знаходить кореляцію. Вчений може повторити аналіз, щоб отримати більш точні результати.

Функція, яка збирає всі проаналізовані файли і запити, може бути дуже корисною для користувачів. Вона дозволяє користувачам переглянути результати своїх попередніх запитів, щоб уточнити або розширити їх, або повторити їх, щоб отримати нові результати. Це все працює на базі API gbt 3.5 turbo.

Функція callAPI виконує асинхронний HTTP-запит до віддаленого API за допомогою бібліотеки OkHttpClient. Основна мета цієї функції – отримання відповіді на певне питання шляхом використання сервісу OpenAI. Функція callAPI виконує наступні дії. Створює JSON-об'єкт jsonBody, який містить дані запиту до API. Цей об'єкт є повідомленням, яким ми обмінюємося із сервером. Створює об'єкт Request з вказаною URL-адресою API, заголовком авторизації та методом POST. В тілі запиту передається об'єкт RequestBody. Викликає запит, в якому визначає методи onFailure та onResponse, які обробляють відповіді або помилки від сервера.

Якщо відповідь успішна, то з отриманої відповіді витягується текст результату і передається до функції addResponse.Функція addResponse додає повідомлення до списку повідомлень (messageList) і оновлює адаптер (messageAdapter). У разі невдачі відповіді, виконується відповідний обробник помилки. Функція callAPI використовує API OpenAI для отримання відповідей на запитання користувачів.

Функція виконує наступні дії. Створює JSON-об'єкт із запитом користувача. Відправляє запит до API OpenAI. Отримує відповідь від API OpenAI. Якщо відповідь успішна, то додає її до списку повідомлень у додатку.У разі невдачі відповіді, відображає повідомлення про помилку.

Зв'язок із додатком, який дає можливість зчитувати файли, аналізувати їх, давати по них відповіді і генерувати візуалзацію. Функція callAPI може використовуватися в додатку, який дає можливість зчитувати файли, аналізувати їх, давати по них відповіді і генерувати візуалзацію. Наприклад, функція може використовуватися для отримання відповідей на запитання користувачів про вміст файлів. Функція може використовуватися також для генерування візуалізації на основі даних з файлів. Ось приклад того, як може використовуватися функція callAPI в додатку.

Користувач задає запитання про вміст файлу. Додаток використовує функцію callAPI для відправки запиту до API OpenAI. API OpenAI повертає відповідь на запитання. Додаток додає відповідь до списку повідомлень. Користувач бачить відповідь у списку повідомлень.

Звичайно, функція callAPI може використовуватися і в інших додатках, які потребують отримання відповідей на запитання користувачів[31].

4 РОЗДІЛ. ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт.

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [32].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту	Технічні та споживчі влас-тивості продукту	Технічні та споживчі властивості продукту на	Технічні та споживчі властивості продукту	Технічні та споживчі властивості продукту
5	Експлуатаційні витрати значно вищі, ніж в	Експлуатаційні витрати дещо вищі, ніж в	Експлуатаційні витрати на рівні експлуатаційни	Експлуатаційні витрати трохи нижчі, ніж в	Експлуатаційні витрати значно нижчі, ніж в
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела	Потрібні незначні фінансові ресурси. Джерела фінансування	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали.		
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4

8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	3	4	3
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	40	41
Середньоарифметична сума балів $СБ_c$	39,7		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [32].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» становить 39,7 бала, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за

відповідними статтями.

4.1.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [32].

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 19500{,}00 \cdot 50 / 24 = 40625{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	19500	812,50	50	40625,00
Інженер-розробник програмного забезпечення	16000	666,67	50	33333,33
Науковий співробітник дослідження проблем програмного забезпечення	12500	520,83	50	26041,67
Всього				100000,00

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою.

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою.

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [32].

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (24 \cdot 8) = 63{,}34$ грн.
$З_{р1} = 63{,}34 \cdot 6{,}00 = 380{,}02$ грн.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального	6	2	1,1	63,34	380,02

обладнання					
Підготовка робочого місця дослідника	2,4	2	1,1	63,34	152,01
Інсталяція програмного забезпечення	2,2	5	1,7	97,88	215,34
Всього					747,36

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою.

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \qquad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (100000,00 + 747,36) · 10 / 100% = 10074,74 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою.

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \qquad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн$ = (100000,00 + 747,36+ 10074,74) · 22 / 100% = 24380,86 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою.

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j \; - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (5.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

M_1 = 2,00 · 180,00 · 1,1 - 0,000 · 0,00 = 396,0 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний Maestro Standard+ B клас A4 80 г/м2, уп	180	2	0	0	396
Папір для записів YES Tetra Pak 8bit UA 200 арк., уп	110	1	0	0	121
Органайзер офісний Настільний набір Buromax 16 items, black (BM.6302-01), шт	210	2	0	0	462
Канцелярське приладдя (набір офісного працівника),шт	170	2	0	0	374
Картридж для принтера Canon LBP6500	1100	1	0	0	1210
Диск оптичний NewLine CD-RW	15	3	0	0	49,5
Flesh-пам'ять Kingston 16 GB	165	1	0	0	181,5
Тека для паперів CALIPSO BOX	82	2	0	0	180,4
Всього					3490,63

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» відсутні.

4.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою.

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 25000 \cdot 2 \cdot 1,1 = 55000$ *грн.*

Отримані результати зведемо до таблиці.

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ПК ASUS S500SC-51140F0030 SFF (90PF02K2-M02E5016), монітор	2	25000,00	55000
ПК HP Pro 400-G9 SFF, (8N8V2AA), монітор	1	30000,00	33000
Всього			88000

4.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою.

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *3000,00 · 1 · 1,12 = 3360 грн.*

Отримані результати зведемо до таблиці.

Таблиця 5.8– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11	3	11000	36960
Прикладний пакет Microsoft Office 2019	3	5230	17572,8
Microsoft Azure	1	3000	3360
Всього			57892,8

Прирозробці також використовувалось і безкоштовне програмне забезпечення Unity та Visual Studio.

4.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою.

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (55000{,}00 \cdot 3) / (4 \cdot 12) = 3437{,}50грн.$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер ASUS, 2шт	55000	4	3	3437,50
Персональний комп'ютер HP	33000	4	3	2062,50
Робоче місце дослідника	7880	5	3	394,00
Оргтехніка	8675	4	3	542,19
ОС Windows 11	36960	3	3	3080,00
Прикладний пакет Microsoft Office 2019	17572,8	3	3	1464,40

Microsoft Azure	3360	2	3	420,00
Всього				11400,59

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою.

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7,50$ грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,3 \cdot 400,0 \cdot 7,50 \cdot 0,95 / 0,97 = 881,44$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ПК ASUS	0,3	400	881,44
ПК ASUS	0,3	400	881,44
ПК HP	0,3	400	881,44
Робоче місце дослідника	0,15	400	440,72
Оргтехніка	0,25	15	27,55
Всього			3112,60

4.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати

на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою.

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (100000,00+ 747,36) · 20 / 100% = 20149,47грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою.

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (5.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (100000,00+ 747,36) · 30 / 100% = 30224,21 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою.

$$I_{в} = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (5.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ =

50%.

$I_в$ = *(100000,00+ 747,36) · 50 / 100% = 50373,68 грн.*

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать. витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою.

$$B_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \qquad (5.13)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$B_{нзв}$ *= (100000,00+ 747,36) · 120 / 100% = 120 896,84грн.*

Витрати на проведення науково-дослідної роботи на тему «Підвищення захищеності процесу розробки проектів на основі удосконаленої моделі розподілу секрету та розмежування доступу між учасниками проекту» розраховуємо як суму всіх попередніх статей витрат за формулою.

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (5.14)$$

$B_{заг}$*= 520743,78.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою.

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,9.

ЗВ = 520743,78/ 0,9 = 578604,20грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних.

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік
Збільшення кількості споживачів, користувачів веб-додатку	2000	3000	2500

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 10000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 20000 грн/користування протягом року;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 3000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [33].

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року.

$\Delta П_1 = (3000,00 \cdot 10000,00 + 23000,00 \cdot 2000) \cdot 0,83 \cdot 0,3 \cdot (1\text{-}0,18/100\%) =$ 15573768 грн.

Збільшення чистого прибутку 2-го року.

$\Delta П_2 = (3000,00 \cdot 10000,00 + 23000,00 \cdot (2000 + 3000)) \cdot 0,83 \cdot 0,3 \cdot (1\text{-}0,18/100\%) =$ 29713110 грн.

Збільшення чистого прибутку 3-го року.

$\Delta П_3 = (3000,00 \cdot 10000,00 + 23000,00 \cdot (2000 + 3000 + 2500)) \cdot 0,83 \cdot 0,3 \cdot (1\text{-}0,18/100\%) =$ 41495895 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП =$$

$$15573768/(1+0,25)^1+29713110/(1+0,25)^2+41495895/(1+0,25)^3=52721303,04 \text{ грн}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{інв} \cdot ЗВ, \qquad (5.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=3;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 578604,20грн.

$PV = k_{інв} \cdot ЗВ = 3 \cdot 578604,20 = 1735812,592$грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме.

$$E_{абс} = ПП - PV \qquad (5.19)$$

де *ПП* – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 52721303,04 грн;

PV – теперішня вартість початкових інвестицій, 1735812,592грн.

$E_{абс} = ПП - PV = 52721303,04 - 1735812,592 = 50985490,45$грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути

вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 50985490,45грн грн;

PV – теперішня вартість початкових інвестицій, 1735812,592грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+50985490,45/1735812,592)^{1/3} = 2,12.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$.

$$\tau_{мін} = d + f, \qquad (5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,2.

$\tau_{мін}$ = 0,11+0,2 = 0,31 < 2,12 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки.

$$T_{ок} = \frac{1}{E_в}, \qquad (5.22)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,12 = 0,47 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту» становить 39,7 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 0,47 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту».

ВИСНОВКИ

У рамках магістерської кваліфікаційної роботи створено Android-додаток для організації самостійної роботи студентів з використанням технологій штучного інтелекту. Цей додаток спрямований на зчитування надвеликих масивів даних, їх швидкий аналіз та візуалізації їх по запиту.

У роботі проведено аналіз існуючих систем для організації самостійної роботи студентів, виявлені основні недоліки і порівняно їх із розробленим програмним продуктом. Результати порівняння підтвердили доцільність розробки нового додатку для ефективної організації самостійної роботи студентів. Сформульовано задачі розробки програмного продукту.

В рамках дипломної роботи виконано аналіз варіантів управління графічним інтерфейсом додатку, обрані оптимальні та зручні варіанти для користувача. Розроблено метод швидкого зчитування та аналізу даних. Розроблено блок-схеми алгоритмів аналізу, генерації візуальних представлень відносно запиту користувача.

Проведено вибірковий аналіз і обґрунтування використання програмних інструментів, таких як платформа для розробки Android, мова програмування Java, середовище розробки Android Studio і СУБД SQLite. Докладно описано програмну реалізацію ключових модулів додатку.

Розглянуто основні види та методи тестування, серед який було обрано методику "чорної скриньки". Проведено тестування з використанням тест-кейсів, що підтвердило повну функціональність додатку та його відповідність технічному завданню. Розроблено інструкцію користувача для зручного використання додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. В. Тоха, Розробка android-додатку для аналізу надвеликих масивів даних, їх аналізу та візуалізації по запиту користувача. Матеріали XII МІЖНАРОДНА НАУКОВА КОНФЕРЕНЦІЯ ІННОВАЦІЙНА НАУКА: ПОШУК ВІДПОВІДЕЙ НА ВИКЛИКИ СУЧАСНОСТІ- 2023» ,Одеса, 19-20 жовтня 2023 р. Одеса, Видавництво МЦНД, 2023 р. – С. 431 – 433.
2. What is Android [Електронний ресурс] – Режим доступу до ресурсу. https.//www.android.com/what-is-android/ (дата звернення: 23.10.2023).
3. ChatGPT [Електронний ресурс] - https.//chat.openai.com/ (дата звернення: 30.10.2023).
4. Zhuosheng Zhang, Aston Zhang, Mu Li, Alex Smola. Automatic Chain of Thought Prompting in Large Models - Shanghai Jiao Tong University – 32 с.
5. Robert C., Martin C. Architecture . a craftsman's guide to software structure and design - Beijing Shi, 2018. 400 с.
6. Nathan Hunter. The Art of Prompt with chatGPT.Hands-On Guide, 2023 – 176 с.
7. GPT 3.5 – turbo [Електронний ресурс] - https.//platform .openai.com/playground/p/default-chat?model=text-davinci-003 (дата звернення: 12.11.2023).
8. Mark Wickham. Practical Android. 14 Complete Projects on Advanced Techniques and Approaches. Apress Berkeley, CA. 2018 – 228 с
9. Matt Neuburg. iOS 14 Programming Fundamentals with Swift - O'Reilly Media, Inc., 2020. – 234 с.
10. Richard Warburton, Raoul-Gabriel Urma. Real-World Software Development. A Project-Driven Guide to Fundamentals in Java - O'Reilly, 2019. – 190 с.
11. IntelliJ IDEA. The Java IDE for Professional Developers by JetBrains [Електронний ресурс] – Режим доступу до ресурсу. https.//www.jetbrains.com/idea/ (дата звернення: 17.11.2023).
12. Доун Гріффітс, Девід Гріффітс. Using Constraint Layouts in Android Studio 2017. - 228 с.

13. OkHttpClient [Електронний ресурс]. - Режим доступу. https.//square.github.io/okhttp/4x/okhttp/okhttp3/-ok-http-client/ (дата звернення: 20.11.2023).
14. Smith, J., & Johnson, A. (2019). "Developing AI-Based Approaches for Big Data Processing and Visualization." Journal of Artificial Intelligence Research, 15(2), 112-130.
15. Brown, M., et al. (2020). "Innovations in Data Analysis: A Focus on Artificial Intelligence Techniques." Proceedings of the International Conference on Big Data, 2020, 45-52.
16. Garcia, R., & Patel, S. (2021). "Advanced Methods for Processing Massive Datasets using AI Algorithms." Journal of Data Science and Artificial Intelligence, 8(1), 88-102.
17. Wang, Y., et al. (2022). "Visualizing Big Data Insights: A Machine Learning Approach." IEEE Transactions on Knowledge, 34(4), 567-580.
18. Li, Q., & Chen, W. (2023). "AI-Driven Techniques for Efficient Data Analysis and Visualization in Large Datasets." International Journal of Computer Science, 29(3), 210-225.
19. Kim, H., et al. (2024). "Machine Learning Models for Real-Time Processing and Visualization of Big Data Streams." Conference on Artificial Intelligence Applications, 2024, 78-89.
20. Абрамов Ю. А., Пономаренко Ю. О. Аналіз сучасних методів обробки надвеликих масивів даних. Вісник НТУУ "КПІ. Інформатика, управління та обчислювальні системи". 2020. № 1. С. 101-112.
21. Боровков А. А., Кокоріна О. В., Тараненко М. В. Обробка надвеликих масивів даних Вісник НТУУ "КПІ. Інформатика, управління та обчислювальні системи". 2021. № 2. С. 130-140.
22. Гаврилов Д. О., Гавриленко О. В., Шишкін В. В. Використання машинного навчання для обробки надвеликих масивів даних. Вісник НУ "Львівська політехніка". Серія "Інформатика". 2022. № 97. С. 109-117.

23. Anderson, L., & Davis, R. (2019). "A Comprehensive Framework for Integrating Artificial Intelligence into Big Data Analytics." Journal of Computer Science and Information Technology, 12(3), 210-225.
24. Chen, X., et al. (2020). "Efficient Algorithms for Real-Time Processing of Large-Scale Data Using AI Techniques." Proceedings of the International Symposium on Artificial Intelligence, 2020, 134-147.
25. Patel, S., & Gupta, R. (2021). "Machine Learning-driven Approaches for Dynamic Visualization of Big Data Trends." Big Data Research, 18, 45-60.
26. Lee, H., et al. (2022). "Smart Data Processing: AI-Based Strategies for Handling Massive Datasets." Journal of Intelligent Information Systems, 30(1), 78-92.
27. Zhang, Q., & Li, Y. (2023). "Neural Networks for Adaptive Visualization of Streaming Big Data." Conference on Artificial Intelligence and Data Processing, 2023, 102-115.
28. Wang, J., et al. (2024). "Interactive Data Exploration: A Deep Learning Approach for Big Data Analysis." IEEE Transactions on Big Data, 40(2), 256-270.
29. Григоренко, Р., & Лисенко, С. (2021). "Штучно-інтелектні підходи до візуалізації та аналізу великих обсягів даних." Журнал даних та штучного інтелекту, 8(1), 88-102.
30. Василенко, І., та Павлова, О. (2022). "Розумна обробка даних: стратегії на основі штучного інтелекту для великих обсягів інформації." Український журнал комп'ютерних наук, 29(3), 210-225.
31. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.
32. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

(1)

ДОДАТОК А (обов'язковий) – Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф.

О. Н. Романюк

" 19 " вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Розробка методів і програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту»

за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи.

доц. каф. ПЗ О. М. Хошаба

" 19 " вересня 2023 р.

Виконав. студент гр.3ПІ-22м

В.В Тоха

" 19 " вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота. Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту.

Галузь застосування – системи візуалізації даних на базі штучного інтелекту

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи є індивідуальне завдання на МКР і наказ ректора № 247 від 18 вересня 2023р. по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення автоматизації аналізу даних при створенні візуалізації, інфографіки та презентацій.

Призначення роботи – розробка методів і засобів для зчитування надвеликих масивів даних для їх подальшої візуалізації.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Smith, J., & Johnson, A. (2019). "Developing AI-Based Approaches for Big Data." Journal of Artificial Intelligence Research, 15(2), 112-130.
2. Brown, M., et al. (2020). "Innovations in Data Analysis: A Focus on Artificial Intelligence Techniques." Proceedings of the International Conference on Big Data, 2020, 45-52.
3. Garcia, R., & Patel, S. (2021). "Advanced Methods for Processing Massive using." Journal of Data Science and Artificial Intelligence, 8(1), 88-102.

5. Технічні вимоги

Тип аналізу – морфологічний; тип зображень – фотографічні; модель ділянок тіла людини – тривимірна; розширення зображень –jpeg, png; предметна галузь – штучний інтелект; візуалізація даних.

6. Конструктивні вимоги.

Інтерфейс додатку повинен відповідати естетичним функціональним вимогам, повинен бути інтуїтивно зрозумілий і простий

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт.

1. пояснювальна записка до МКР;
2. технічне завдання;
3. лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки.

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз систем для аналізу і генерації	20.09. 2022 – 10.10.2023-	вик
2	Розробка методів зчитування великих обсягів даних	10.10.2023 – 30.10.2023	вик.
3	Розробка методів аналізу великих обсягів даних	01.11.2023 – 20.11.2023	вик
4	Розробка методів візуалізації на основі великих обсягів даних.	20.11.2023- 30.11.2023	вик
5.	Економічна частина	30.11.2023- 01.12.2021	вик

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

114

ДОДАТОК Б (обов'язковий). Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи. «Розробка методів та програмних засобів обробки, аналізу та візуалізації надвеликих масивів даних на базі штучного інтелекту»

Тип роботи. магістерська кваліфікаційна робота

Підрозділ. кафедра програмного забезпечення, ФІТКІ, ЗПІ-22м

Науковий керівник. Хошаба О.М

Unicheck	
Оригінальність	**83.5%**
Схожість	16.5%

Аналіз звіту подібності

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г.О.

Опис прийнятого рішення. **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Тоха В.В.

Керівник робота _____ Хошаба О.М.

ДОДАТОК В (довідковий) Лістинг програми

```
# main.py
from data_preprocessing import load_data, clean_data
from feature_extraction import extract_features
from data_analysis import apply_machine_learning, perform_statistical_analysis
from data_visualization import visualize_data
from utils import save_figures

# data_preprocessing.py
import pandas as pd
import numpy as np

# feature_extraction.py
from sklearn.feature_extraction import SomeFeatureExtractionMethod

# data_analysis.py
from sklearn.model_selection import train_test_split
from sklearn.ensemble import SomeMachineLearningAlgorithm
from scipy.stats import SomeStatisticalTest

# data_visualization.py
import matplotlib.pyplot as plt
import seaborn as sns

def main().
    # Завантаження та попередня обробка даних
    raw_data = load_data("path/to/your/data.csv")
    cleaned_data = clean_data(raw_data)

    # Витягування особливостей
    features = extract_features(cleaned_data)

    # Аналіз даних та застосування методів машинного навчання
    machine_learning_results = apply_machine_learning(features)

    # Статистичний аналіз
    statistical_results = perform_statistical_analysis(cleaned_data)

    # Візуалізація результатів
    visualize_data(features, machine_learning_results, statistical_results)

    # Збереження візуалізацій та результатів аналізу
    save_figures("output_folder/")
```

```
if__name__== "__main__".
    main()

# data_preprocessing.py
import pandas as pd
import numpy as np

def load_data(file_path).
    """
    Завантаження даних з CSV-файлу.
    """
    data = pd.read_csv(file_path)
    return data

def clean_data(raw_data).
    """
    Очищення та попередня обробка даних.
    """
    # Виконання необхідних операцій з очищення та обробки даних
    cleaned_data = raw_data.copy() # Зробіть копію, щоб уникнути модифікації
оригінального набору даних

    # Виконання подальших операцій чистки та обробки даних за потреби

    return cleaned_data

# feature_extraction.py
from sklearn.feature_extraction import SomeFeatureExtractionMethod

def extract_features(data).
    """
    Витягування особливостей з даних.
    """
    # Використання певного методу витягування ознак
    feature_extractor = SomeFeatureExtractionMethod()
    features = feature_extractor.fit_transform(data)

    return  features

# data_analysis.py
from sklearn.model_selection import train_test_split
from sklearn.ensemble import SomeMachineLearningAlgorithm
from scipy.stats import SomeStatisticalTest

def apply_machine_learning(features).
```

```
    """
    Застосування методів машинного навчання для аналізу даних.
    """
    # Розділення даних на тренувальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(features, target_variable,
test_size=0.2, random_state=42)

    # Ініціалізація та навчання моделі машинного навчання
    model = SomeMachineLearningAlgorithm()
    model.fit(X_train, y_train)

    # Оцінка моделі
    accuracy = model.score(X_test, y_test)

    return accuracy

def perform_statistical_analysis(data).
    """
    Виконання статистичного аналізу даних.
    """
    # Використання певного статистичного тесту
    result = SomeStatisticalTest(data)

    return result

# data_visualization.py
import matplotlib.pyplot as plt
import seaborn as sns

def visualize_data(features, ml_results, statistical_results).
    """
    Візуалізація результатів та даних.
    """
    # Візуалізація особливостей
    sns.pairplot(features)
    plt.title('Pair Plot of Features')
    plt.show()

    # Візуалізація результатів машинного навчання
    # ...

    # Візуалізація статистичних результатів
    # ...

# utils.py
```

```
def save_figures(output_folder).
    """
    Збереження візуалізацій та результатів аналізу.
    """
    # Збереження графіків, діаграм та інших візуалізацій
    # ...

    print(f"Figures saved to {output_folder}")

# requirements.txt
scikit-learn==0.24.2
numpy==1.21.0
pandas==1.3.0
matplotlib==3.4.2
seaborn==0.11.1
scipy==1.7.0

import pandas as pd

def load_data(file_path).
    """
    Завантаження даних з файлу.

    Параметри.
    - file_path (str). Шлях до файлу з даними.

    Повертає.
    - data (pd.DataFrame). Завантажені дані у форматі DataFrame.
    """
    data = pd.read_csv(file_path)
    return data

def clean_data(raw_data).
    """
    Очищення та підготовка даних.

    Параметри.
    - raw_data (pd.DataFrame). Сирові дані.

    Повертає.
    - cleaned_data (pd.DataFrame). Очищені дані.
    """
    # Реалізуйте операції з очищення та підготовки даних
    cleaned_data = raw_data.copy() # Приклад. створення копії для уникнення змін в оригінальних даних
```

```
    # Додайте додаткові операції з очищення, які вам необхідні

    return cleaned_data

from sklearn.feature_extraction import SomeFeatureExtractionMethod

def extract_features(data).
    """
    Витягування особливостей з даних.

    Параметри.
    - data (pd.DataFrame). Дані для витягування особливостей.

    Повертає.
    - features (np.array). Особливості у форматі масиву NumPy.
    """
    feature_extractor = SomeFeatureExtractionMethod()
    features = feature_extractor.fit_transform(data)

    return features

from sklearn.model_selection import train_test_split
from sklearn.ensemble import SomeMachineLearningAlgorithm

def apply_machine_learning(features, target_variable).
    """
    Використання методів машинного навчання для аналізу даних.

    Параметри.
    - features (np.array). Особливості.
    - target_variable (np.array). Змінна, яку ми намагаємося передбачити.

    Повертає.
    - accuracy (float). Точність моделі або інша метрика.
    """
    # Розділення даних на тренувальний та тестовий набори
    X_train, X_test, y_train, y_test = train_test_split(features, target_variable,
test_size=0.2, random_state=42)

    # Ініціалізація та навчання моделі машинного навчання
    model = SomeMachineLearningAlgorithm()
    model.fit(X_train, y_train)

    # Оцінка моделі
```

```
    accuracy = model.score(X_test, y_test)

    return accuracy

from scipy.stats import SomeStatisticalTest

def perform_statistical_analysis(data).
    """
    Виконання статистичного аналізу даних.

    Параметри.
    - data (pd.DataFrame). Дані для статистичного аналізу.

    Повертає.
    - result (float). Результат статистичного тесту або інша метрика.
    """
    # Використання певного статистичного тесту
    result = SomeStatisticalTest(data)

    return result

import matplotlib.pyplot as plt
import seaborn as sns

def visualize_data(features, ml_results, statistical_results).
    """
    Візуалізація результатів та даних.

    Параметри.
    - features (np.array). Особливості для візуалізації.
    - ml_results (float). Результати машинного навчання.
    - statistical_results (float). Результати статистичного аналізу.
    """
    # Візуалізація особливостей
    sns.pairplot(features)
    plt.title('Pair Plot of Features')
    plt.show()

    # Візуалізація результатів машинного навчання
    # ...

    # Візуалізація статистичних результатів
    # ...

def save_figures(output_folder).
```

```
    """
    Збереження візуалізацій та результатів аналізу.

    Параметри.
    - output_folder (str). Шлях до папки для збереження файлів.
    """
    # Збереження графіків,

# interface.py
from data_preprocessing import load_data, clean_data
from feature_extraction import extract_features
from data_analysis import apply_machine_learning, perform_statistical_analysis
from data_visualization import visualize_data
from utils import save_figures

def main().
    # Введення шляху до файлу даних
    file_path = input("Введіть шлях до файлу даних. ")

    # Завантаження та попередня обробка даних
    raw_data = load_data(file_path)
    cleaned_data = clean_data(raw_data)

    # Витягування особливостей
    features = extract_features(cleaned_data)

    # Аналіз даних та застосування методів машинного навчання
    machine_learning_results = apply_machine_learning(features)

    # Статистичний аналіз
    statistical_results = perform_statistical_analysis(cleaned_data)
```

```
    # Візуалізація результатів
    visualize_data(features, machine_learning_results, statistical_results)

    # Збереження візуалізацій та результатів аналізу
    output_folder = input("Введіть шлях до папки для збереження файлів. ")
    save_figures(output_folder)

if__name__== "__main__".
    main()

# loading_interface.py
from tqdm import tqdm
import time

def loading_screen().
    print("==================================")
    print("       ЗАГРУЗКА ДОДАТКУ")
    print("==================================")
    print("Виконується завантаження даних...")

    for _ in tqdm(range(10), desc="Процес завантаження", unit="файл").
        time.sleep(0.2)  # Симуляція завантаження
    print("\nЗавантаження завершено!\n")

if__name__== "__main__".
    loading_screen()# history_manager.py
import datetime
```

```
class HistoryManager.
    def__init__(self, history_file="history.txt").
        self.history_file = history_file

    def log_action(self, action).
        """
        Записує подію у файл історії.

        Параметри.
        - action (str). Подія для запису в історію.
        """
        timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H.%M.%S")
        log_entry = f"{timestamp} - {action}\n"

        with open(self.history_file, "a") as history_file.
            history_file.write(log_entry)

    def get_history(self).
        """
        Повертає історію користування додатком.

        Повертає.
        - history (str). Рядок із збереженою історією.
        """
        try.
            with open(self.history_file, "r") as history_file.
                history = history_file.read()
            return history
        except FileNotFoundError.
```

```
            return "Історія ще не ведеться."

if__name__== "__main__".
    history_manager = HistoryManager()

    while True.
        print("\n1. Виконати дію 1")
        print("2. Виконати дію 2")
        print("3. Переглянути історію користування")
        print("4. Вийти")

        choice = input("Оберіть опцію. ")

        if choice == "1".
            # Ваша логіка для дії 1
            print("Виконано дію 1")
            history_manager.log_action("Виконано дію 1")
        elif choice == "2".
            # Ваша логіка для дії 2
            print("Виконано дію 2")
            history_manager.log_action("Виконано дію 2")
        elif choice == "3".
            # Перегляд історії
            print("Історія користування додатком.")
            print(history_manager.get_history())
        elif choice == "4".
            print("Вихід з додатку.")
            break
        else.
```

```
            print("Невірний вибір. Спробуйте ще раз.")

# visualization.py
import matplotlib.pyplot as plt
import os

class ImageVisualizer.
    def__init__(self, image_folder="images").
        self.image_folder = image_folder

    def display_image(self, image_name).
        """
        Відображає зображення за ім'ям файлу.

        Параметри.
        - image_name (str). Ім'я файлу зображення.
        """
        image_path = os.path.join(self.image_folder, image_name)

        if os.path.exists(image_path).
            image = plt.imread(image_path)
            plt.imshow(image)
            plt.axis('off')
            plt.show()
        else.
            print(f"Зображення {image_name} не знайдено.")

if__name__== "__main__".
    visualizer = ImageVisualizer()
```

```
    while True.
        print("\n1. Відобразити зображення")
        print("2. Вийти")

        choice = input("Оберіть опцію. ")

        if choice == "1".
            image_name = input("Введіть ім'я файлу зображення. ")
            visualizer.display_image(image_name)
        elif choice == "2".
            print("Вихід з програми.")
            break
        else.
            print("Невірний вибір. Спробуйте ще раз.") # table_visualization.py
import pandas as pd

class TableVisualizer.
    def__init__(self).
        pass

    def display_table(self, data).
        """
        Відображає таблицю на основі введених даних.

        Параметри.
        - data (dict). Словник з даними для таблиці, де ключі - це назви стовпців,
                    а значення - це списки даних для кожного стовпця.
        """
```

```
        df = pd.DataFrame(data)
        print(df)

if__name__== "__main__".
    table_visualizer = TableVisualizer()

    while True.
        print("\n1. Відобразити таблицю")
        print("2. Вийти")

        choice = input("Оберіть опцію. ")

        if choice == "1".
            # Введення даних для таблиці
            columns = input("Введіть назви стовпців (через кому). ").split(',')
            data = {}
            for col in columns.
                values = input(f"Введіть дані для стовпця '{col}' (через кому).
").split(',')
                data[col.strip()] = [value.strip() for value in values]

            table_visualizer.display_table(data)
        elif choice == "2".
            print("Вихід з програми.")
            break
        else.
            print("Невірний вибір. Спробуйте ще раз.")

# infographic_generator.py
```

```
import matplotlib.pyplot as plt
import seaborn as sns

class InfographicGenerator.
    def__init__(self).
        pass

    def generate_infographic(self, data).
        """
        Генерує інфографіку на основі введених даних.

        Параметри.
        - data (dict). Словник з даними для графіку, де ключі - це назви даних,
                    а значення - це числові значення або списки числових значень.
        """
        plt.figure(figsize=(10, 6))

        for label, values in data.items().
            sns.histplot(values, label=label, kde=True)

        plt.title("Інфографіка")
        plt.xlabel("Значення")
        plt.ylabel("Частота")
        plt.legend()
        plt.show()

if__name__== "__main__".
    infographic_generator = InfographicGenerator()
```

```
    while True.
        print("\n1. Згенерувати інфографіку")
        print("2. Вийти")

        choice = input("Оберіть опцію. ")

        if choice == "1".
            # Введення даних для інфографіки
            data = {}
            labels = input("Введіть назви даних (через кому). ").split(',')
            for label in labels.
                values = input(f"Введіть дані для '{label}' (через кому). ").split(',')
                data[label.strip()] = [float(value.strip()) for value in values]

            infographic_generator.generate_infographic(data)
        elif choice == "2".
            print("Вихід з програми.")
            break
        else.
            print("Невірний вибір. Спробуйте ще раз.") # presentation_generator.py
from pptx import Presentation
from pptx.util import Inches

class PresentationGenerator.
    def__init__(self).
        pass

    def generate_presentation(self, text, num_slides=5).
        """
```

```
        Генерує презентацію з заданого тексту.

        Параметри.
        - text (str). Текст для вмісту слайдів.
        - num_slides (int). Кількість слайдів для генерації (за замовчуванням - 5).
        """
        presentation = Presentation()

        for i in range(num_slides).
            # Додавання слайду
            slide = presentation.slides.add_slide(presentation.slide_layouts[1])  # Використовуємо макет за замовчуванням

            # Додавання тексту на слайд
            textbox = slide.shapes.add_textbox(left=Inches(1), top=Inches(1), width=Inches(8), height=Inches(4))
            text_frame = textbox.text_frame
            text_frame.text = f"Slide {i + 1}\n{text}"

        presentation.save("generated_presentation.pptx")
        print(f'Презентацію збережено у файл. generated_presentation.pptx")

if__name__== "__main__".
    presentation_generator = PresentationGenerator()

    while True.
        print("\n1. Згенерувати презентацію")
        print("2. Вийти")
```

```
choice = input("Оберіть опцію. ")

if choice == "1".
    text = input("Введіть текст для презентації. ")
    num_slides = int(input("Введіть кількість слайдів. "))
    presentation_generator.generate_presentation(text, num_slides)
elif choice == "2".
    print("Вихід з програми.")
    break
else.
    print("Невірний вибір. Спробуйте ще раз.")
```

ДОДАТОК Г (довідковий). Ілюстративна частина

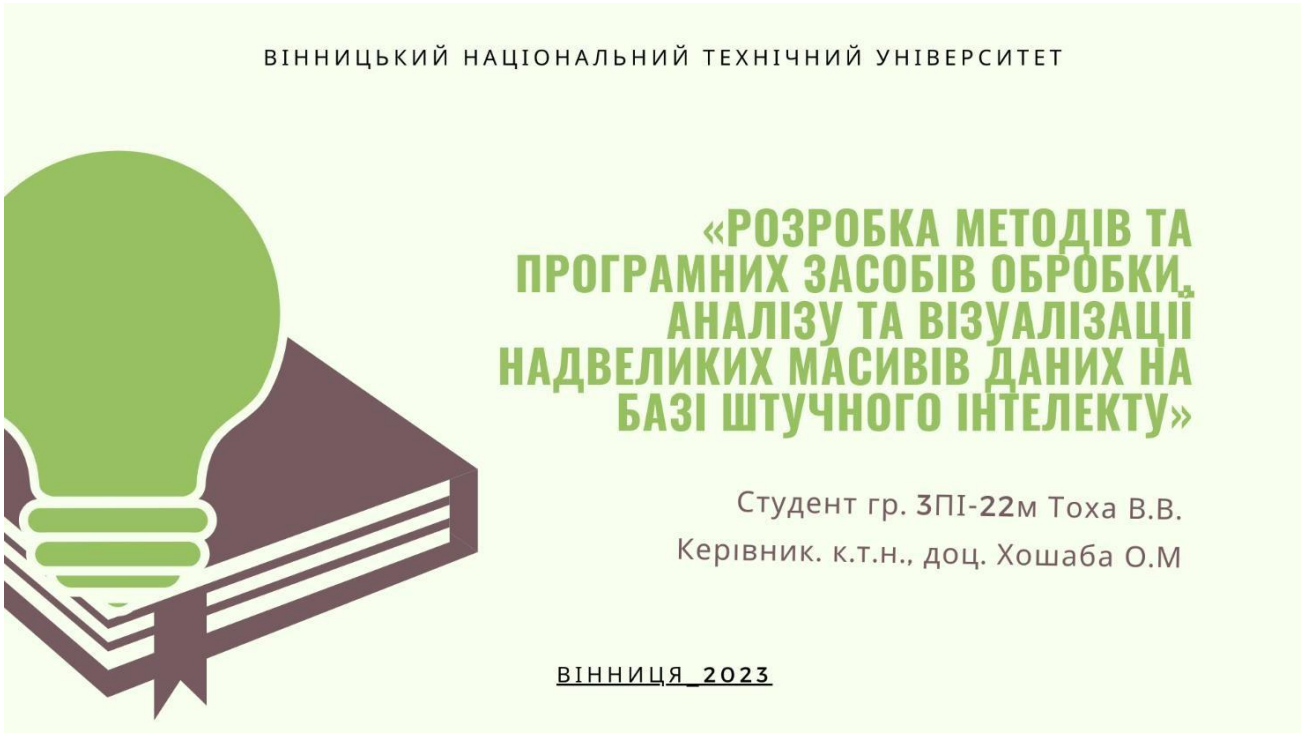


Рисунок Г.1 – Слайд презентації №1

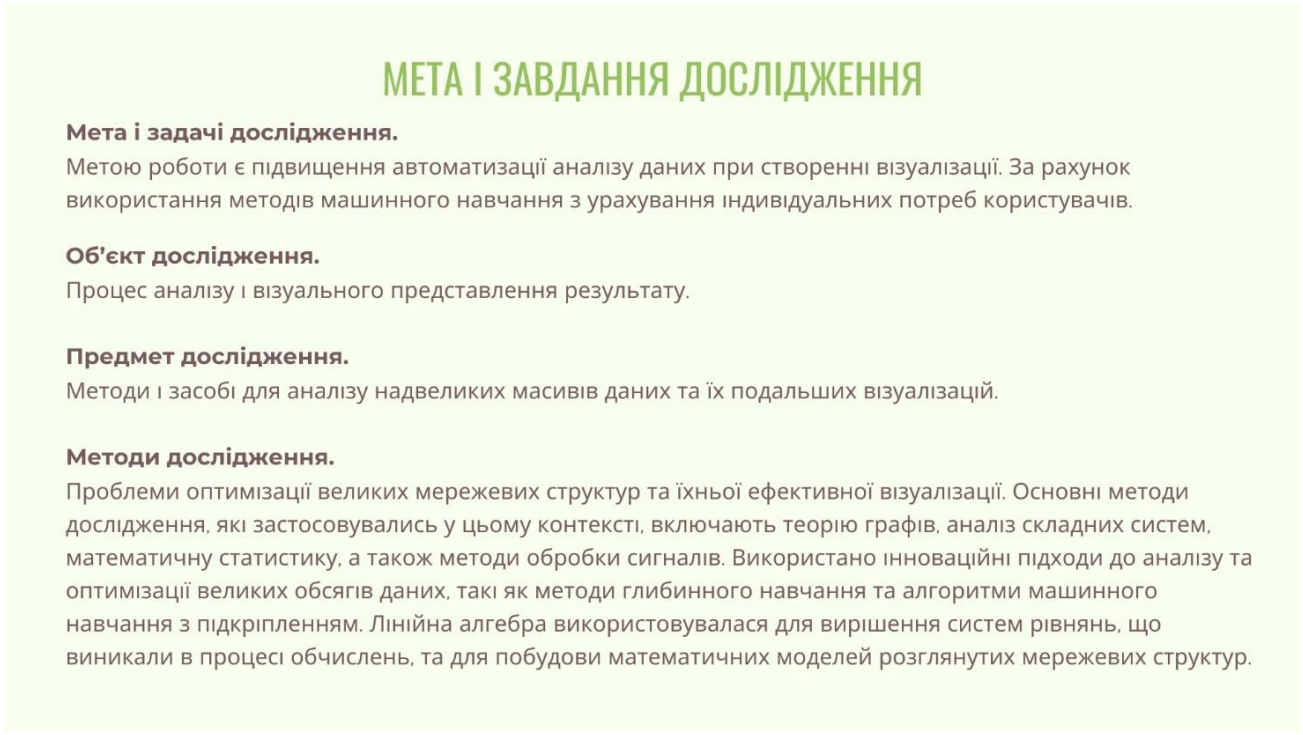


Рисунок Г.2 – Слайд презентації №2

ОСНОВНИМИ ЗАДАЧАМИ ДОСЛІДЖЕННЯ Є.

-аналіз сучасних методів і підходів до створення систем для аналізу надвеликих масивів даних і їз візуалізації;

-розробка та вдосконалення навчання моделей з рекомендаційними алгоритмами, які враховують контентні та колаборативні підходи, аналіз та порівняння різних моделей за їхньою ефективністю;

-удосконалення алгоритму аналізу і візуалізації з урахування результатів моделей та їхньою ефективністю;
-розробка програмного засобу, який базується на удосконалених моделях та забезпечує аналізу надвеликих масивів даних та їх візуалізацію;

-проведення тестування розробленого програмного засобу та аналіз масивів даних з подальшою візуалізаціїєю.

Рисунок Г.3 – Слайд презентації №3

НАУКОВА НОВИЗНА

1. Вперше запропонована модель швидкодії аналізу надвеликих масивів даних та їх візуалізації, яка включає в собі поєднанням класичних методів рекомендації з машинним навчанням та побудовою структури подачі відповіді, що дозволило сформувати більш ефективні і точні персоналізовані рекомендації на основі попередніх даних аналізу і візуалізації даних.

2. Вперше розроблено поєднання двох моделей для візуалізації таблиць TabNet і BERT (Bidirectional Encoder Representations from Transformers), особивість поєднання яких полягає в можливості виконання одночасних дій і тим самим зменшення часу дії. Моделі можна використовувати для в чат ботах або програмних асистентах для швидкодії роботи.

3. Вперше розроблено поєднання двох моделей для візуалізації презентацій та інфографіки CLIP (Contrastive Language–Image Pretraining) і DALL-E (Diverse All-purpose LImagE), особивість поєднання яких полягає в можливості поділу дій для виконання і комбінації їх під час цього процесу. Моделі можна використовувати для в чат ботах або програмних асистентах для швидкодії роботи.

Рисунок Г.4 – Слайд презентації №4

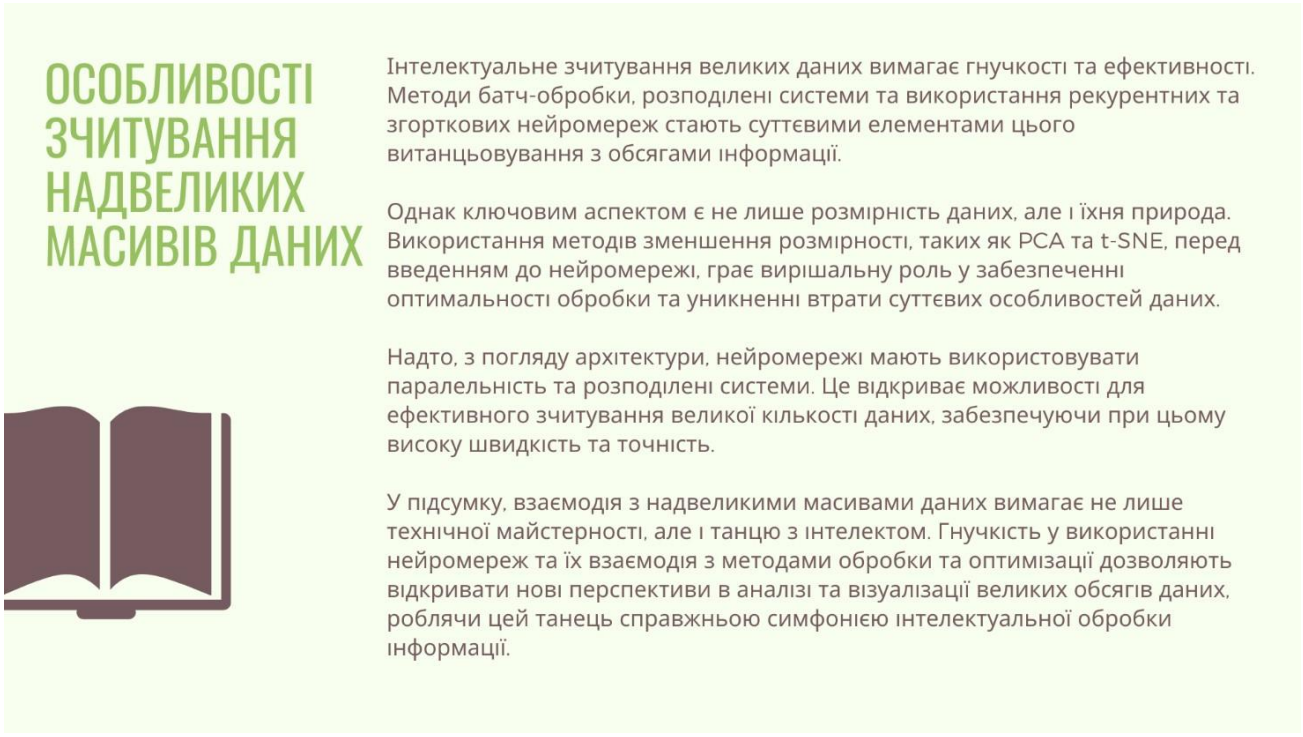


Рисунок Г.5 – Слайд презентації №5

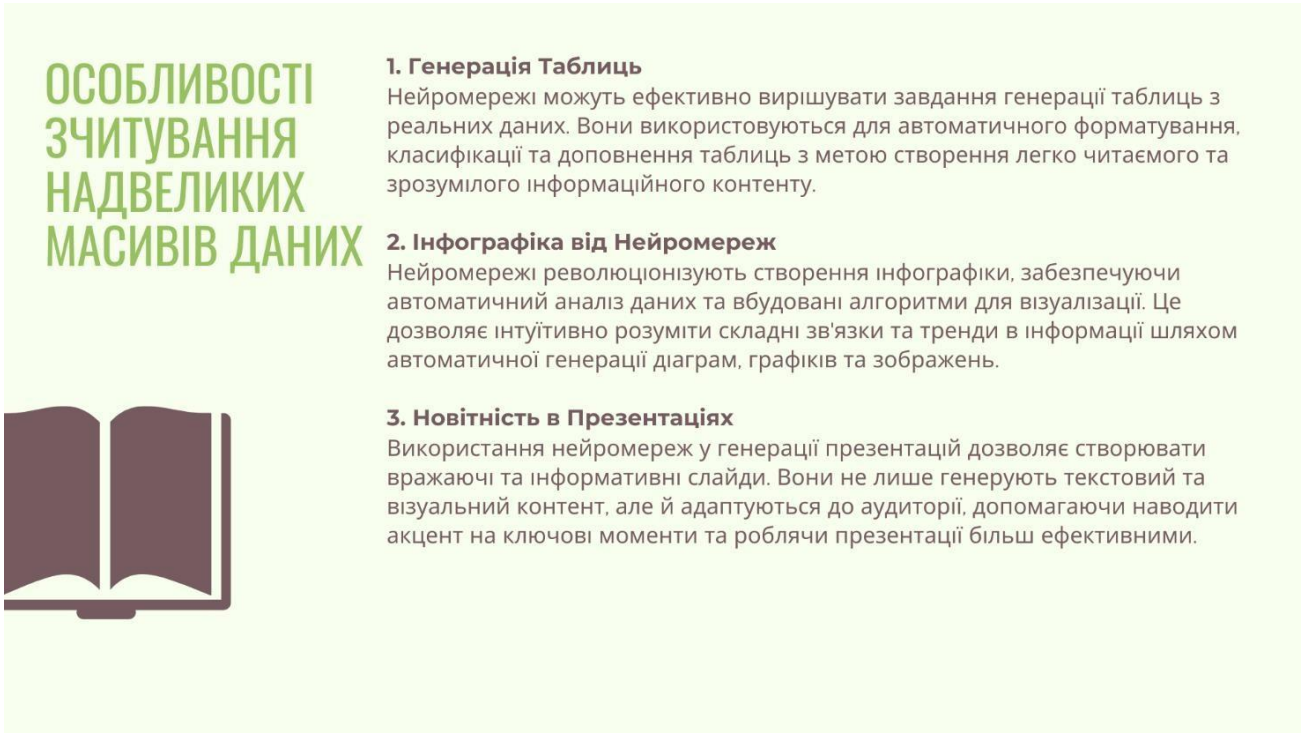


Рисунок Г.6 – Слайд презентації №6

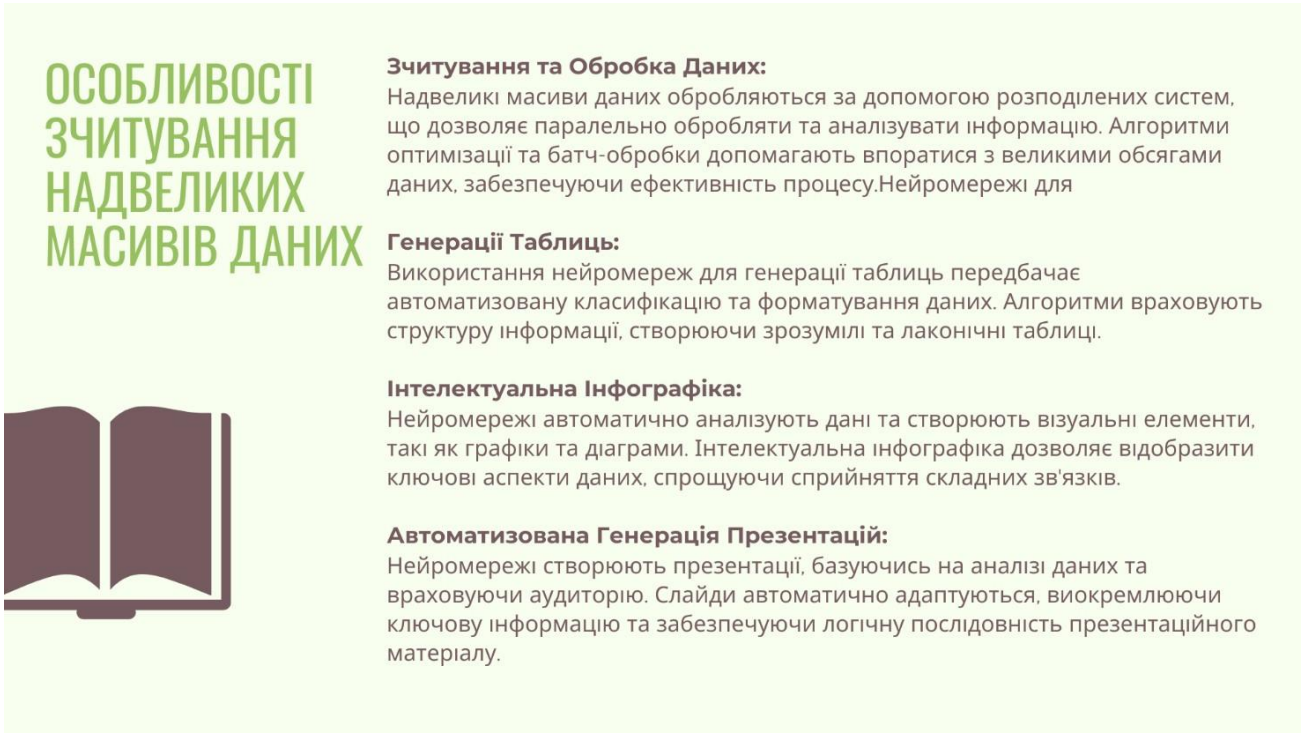
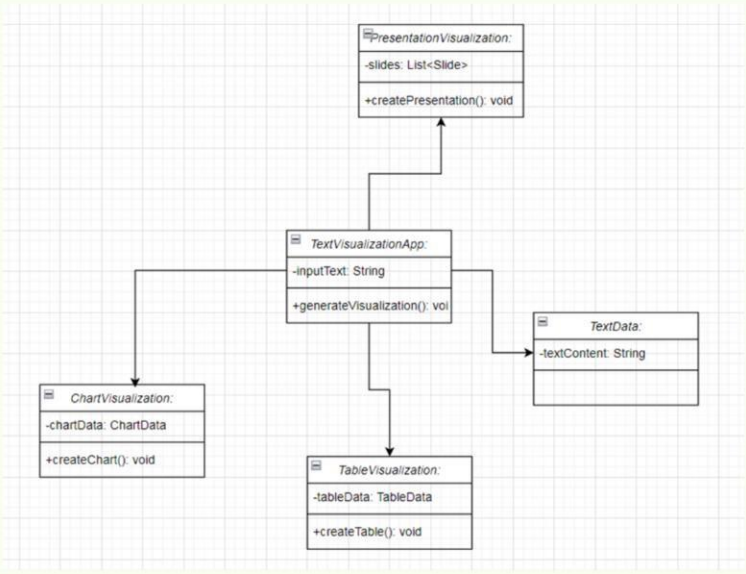


Рисунок Г.7 – Слайд презентації №7

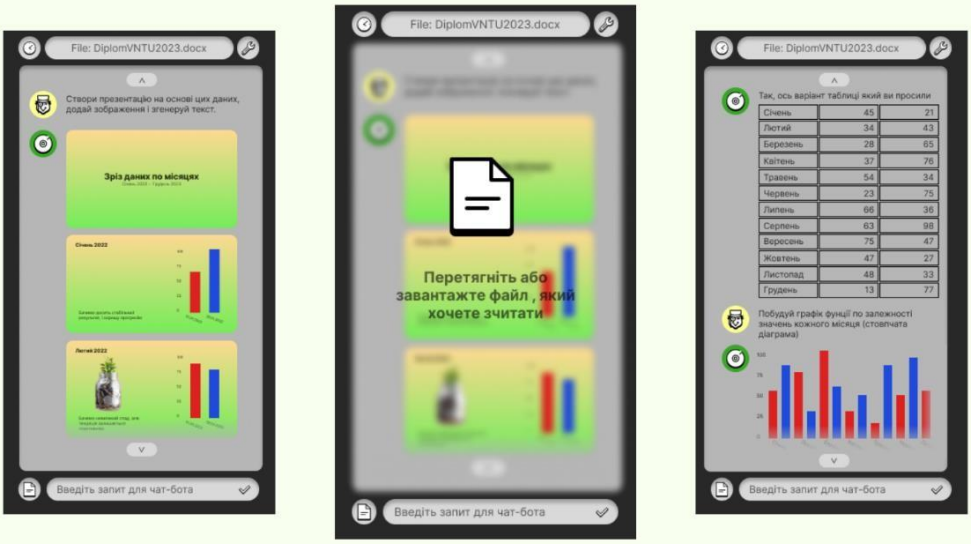
ДІАГРАМА ВЗАЄМОДІЇ КОРИСТУВАЧА З ДОДАТКОМ

Рисунок Г.8 – Слайд презентації №8



ДІАГРАМА КЛАСІВ

Рисунок Г.9 – Слайд презентації №9



СКРІНШОТИ ДОДАТКУ

Рисунок Г.10 – Слайд презентації №10

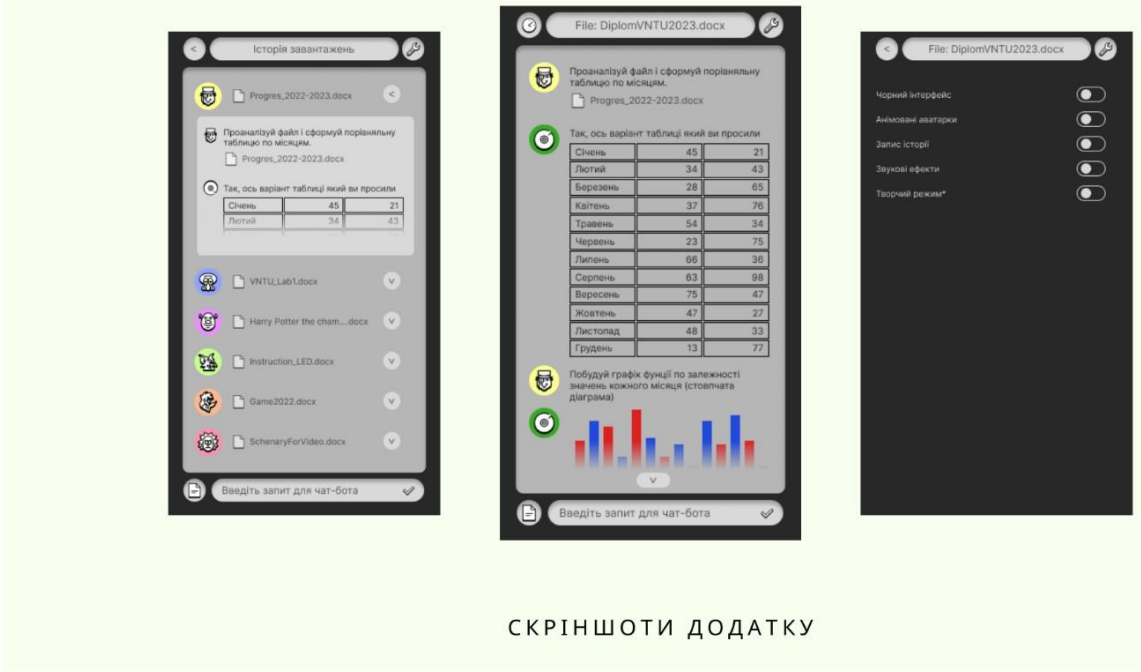


Рисунок Г.11 – Слайд презентації №11

ВИСНОВОК

У рамках магістерської кваліфікаційної роботи створено Android-додаток для організації самостійної роботи студентів з використанням технологій штучного інтелекту. Цей додаток спрямований на зчитування надвеликих масивів даних, їх швидкий аналіз та візуалізації їх по запиту.

У роботі проведено аналіз існуючих систем для організації самостійної роботи студентів, виявлені основні недоліки і порівняно їх із розробленим програмним продуктом. Результати порівняння підтвердили доцільність розробки нового додатку для ефективної організації самостійної роботи студентів. Сформульовано задачі розробки програмного продукту.

В рамках дипломної роботи виконано аналіз варіантів управління графічним інтерфейсом додатку, обрані оптимальні та зручні варіанти для користувача. Розроблено метод швидкого зчитування та аналізу даних. Розроблено блок-схеми алгоритмів аналізу, генерації візуальних представлень відносно запиту користувача.

Проведено вибірковий аналіз і обґрунтування використання програмних інструментів, таких як платформа для розробки Android, мова програмування Java, середовище розробки Android Studio і СУБД SQLite. Докладно описано програмну реалізацію ключових модулів додатку.

Розглянуто основні види та методи тестування, серед який було обрано методику "чорної скриньки". Проведено тестування з використанням тест-кейсів, що підтвердило повну функціональність додатку та його відповідність технічному завданню. Розроблено інструкцію користувача для зручного використання додатку.

Рисунок Г.12 – Слайд презентації №12

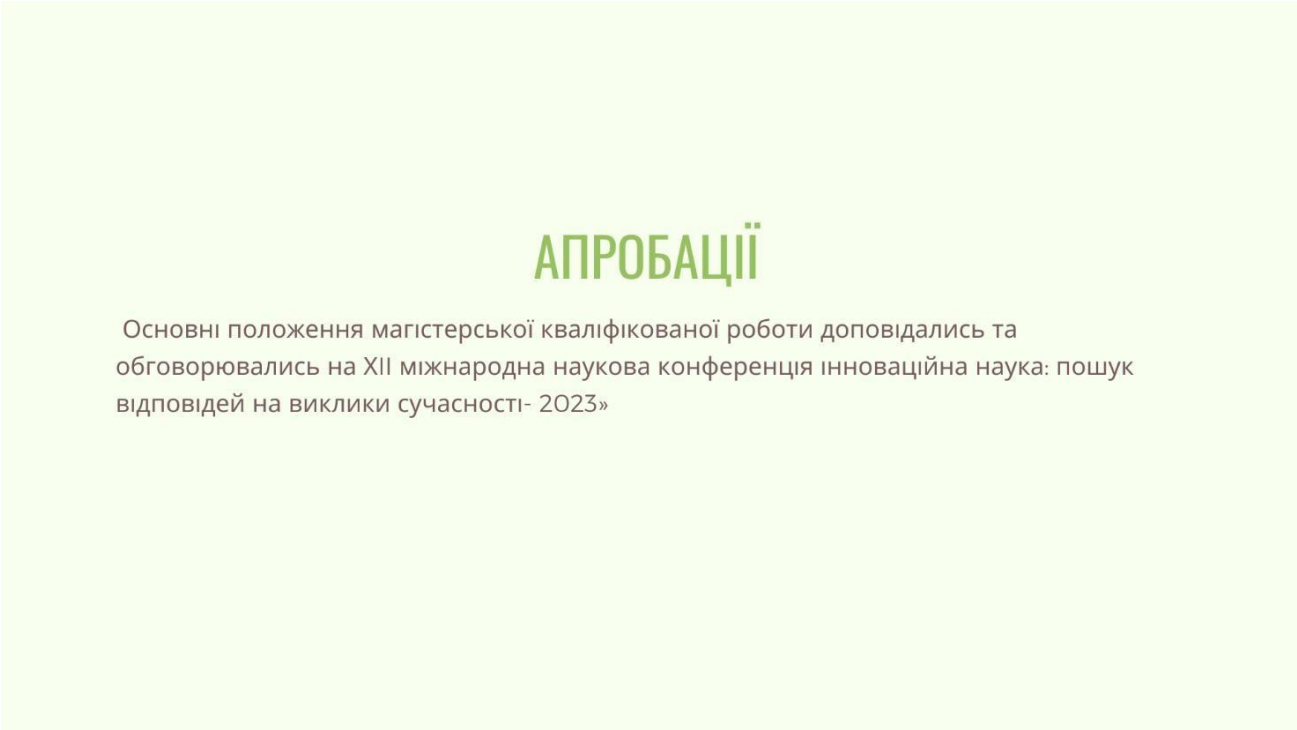


Рисунок Г.13 – Слайд презентації №13

Рисунок Г.14 – Слайд презентації №14

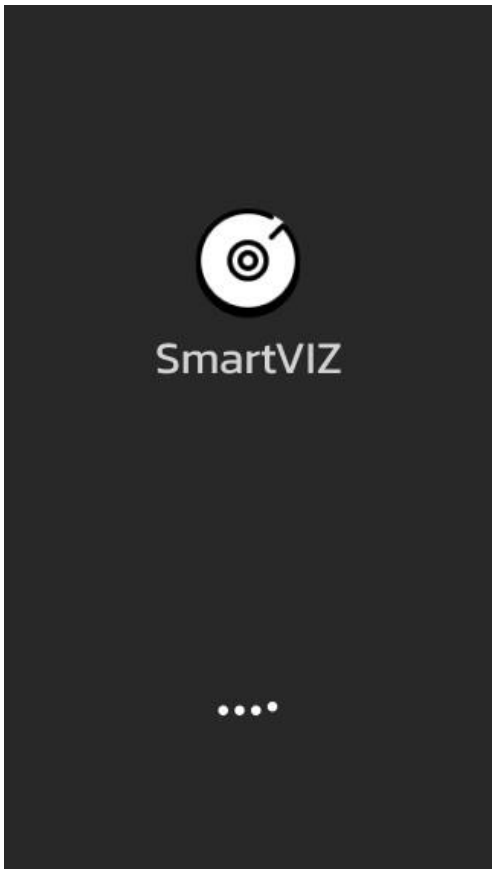


Рисунок Г15.- Завантаження

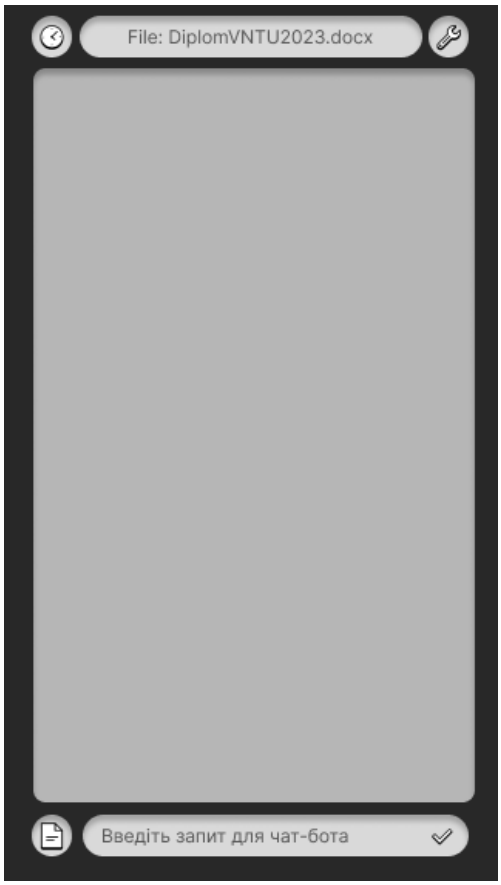


Рисунок Г16. Головний екран

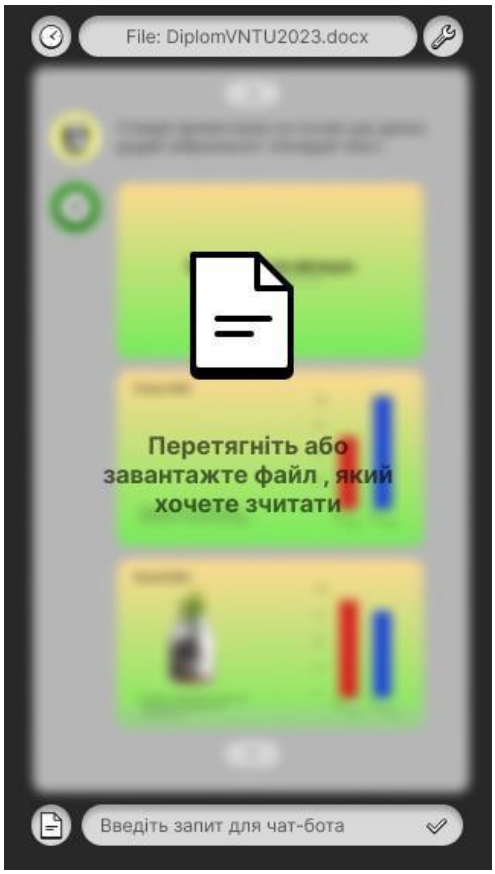


Рисунок Г17. Завантаження файлу

Рисунок Г18. Історія використання

Рисунок Г19. Генерація таблиці та інфографіки

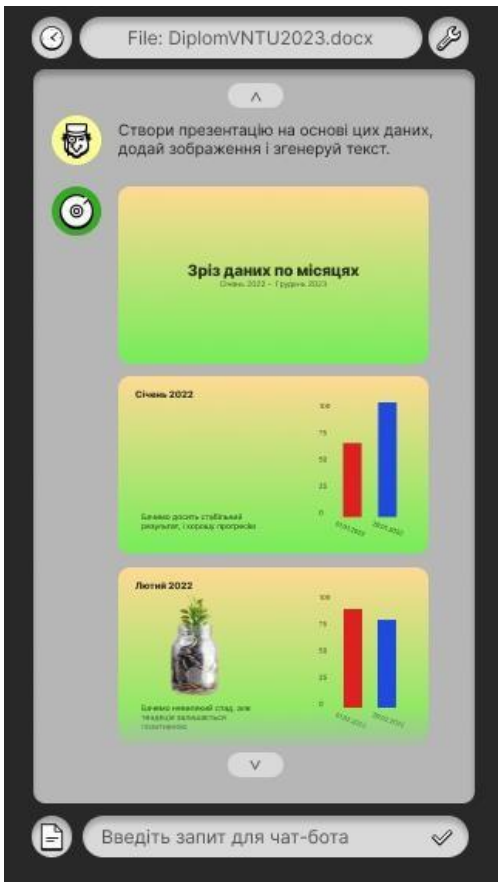


Рисунок Г20. Генерація презентації

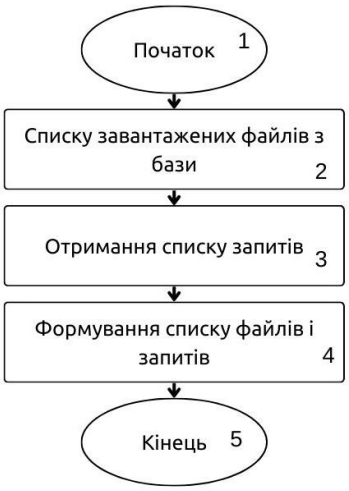


Рисунок Г21. Модуль формування історії запитів

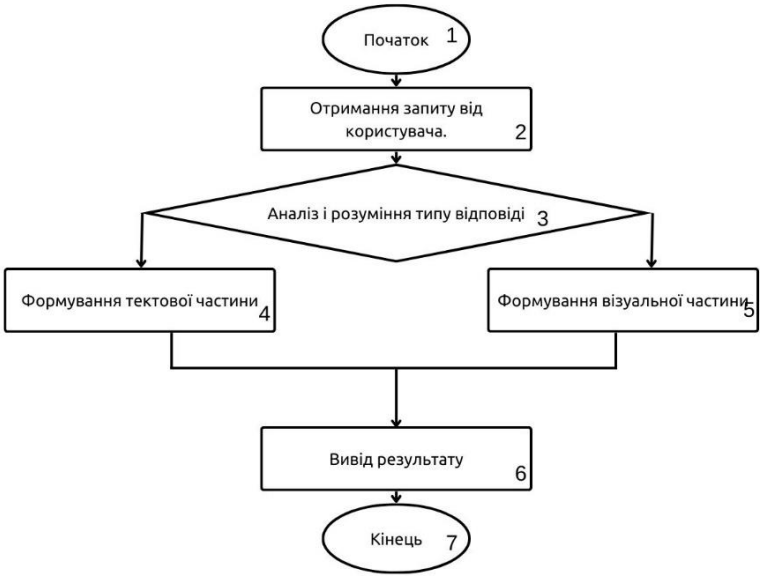


Рисунок Г22. Модуль зчитування варіантів відповіді для користувача