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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій штучного інтелекту»

Виконав: студент II курсу
групи 3ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Пархоменко Роман Михайлович

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
«11» грудня 2023 р.
Опонент: к.т.н., доц. каф. ОТ Войцеховська О.В.
«11» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«11» 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

Пархоменку Роману Михайловичу

1. Тема роботи – розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій штучного інтелекту.
Керівник роботи: Ракитянська Ганна Борисівна, к.т.н., 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: онтологічна модель предметної області; понятійний апарат даної галузі науки; координати вершин графу статистики, база даних задач SQLite, мовна модель штучного інтелекту gpt-3.5-turbo.

4. Зміст текстової частини: вступ; аналіз та постановка задачі; аналіз аналогів; розробка методу роботи чат-боту екзаменатора, методу вирахування коефіцієнта навантаження, архітектури та алгоритмів системи; розробка графічного інтерфейсу додатку; розробка програмних компонент системи; тестування додатку; інструкція користувача; економічна частина; висновки; перелік посилань; додатки.

5. Перелік ілюстративного матеріалу: актуальність розробки, мета та завдання дослідження, порівняльний аналіз аналогів, концептуальна схема додатку, вирахування коефіцієнта навантаженості студента, блок-схеми алгоритмів, тестування додатку, апробація результатів, висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г. Б., к.т.н., доц. каф. ПЗ	19.09.2023	19.09.2023
5	Причепа І. В., к.е.н., доц. каф. ЕПВМ	19.09.2023	19.09.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання організації самостійної роботи студента	20.09. 2023 – 05.10.2023	Вик.
2	Розробка методів та програмних засобів для реалізації додатку	06.10.2023- 14.10.2023	Вик.
3	Розробка програмних компонент Android-додатку для організації самостійної роботи студента	15.10.2023- 01.11.2023	Вик.
4	Тестування програмного додатку	01.11.2023- 16.11.2023	Вик.
5.	Економічна частина	17.11.2023- 1.12.2023	Вик.

Студент _____ **Пархоменко Р. М.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ракитянська Г. Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 519.7:004.89

Пархоменко Р.М. розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій 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.: 30 назв; рис.: 33; табл. 14.

У магістерській кваліфікаційній роботі розроблено методи та програмні засоби для організації самостійної роботи студента з використанням технологій штучного інтелекту. Сформульовано мету досліджень – підвищення ефективності освітнього процесу студента шляхом організації самостійної роботи за допомогою функціоналу мобільного додатку.

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

Розроблено алгоритми сортування задач за категоріями, алгоритм виводу задач на вказану дату та алгоритм генерування графічного звіту статистики навантеженості днів задачами у вигляді стовпчикових діаграм. 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 та якості самостійної роботи студентів під час навчання в університеті.

Ключові слова: Android, штучний інтелект, промпт інженерія.

ABSTRACT

In the master's qualification work, methods and software tools have been developed for organizing students' independent work using artificial intelligence technologies. The research goal is formulated as improving the efficiency of the student educational process by organizing independent work through the functionality of a mobile application.

An analysis of existing solutions and tools for organizing students' independent work has been conducted.

Algorithms for sorting tasks by categories, displaying tasks on a specified date, and generating graphical reports on the workload of days with tasks in the form of bar charts have been developed. Additionally, an architectural model of the application was preliminarily developed. As a result of the development, a distributed software system was created. Testing of the software system demonstrated its correct operation and compliance with the set tasks.

The results of the master's qualification work can be used to enhance the efficiency and quality of student learning during their university studies.

Keywords: Android, artificial intelligence, prompt engineering.

ЗМІСТ

ВСТУП

Актуальність роботи. Освіта є ключовим аспектом розвитку кожної людини та суспільства в цілому. Здобуття знань, навичок та умінь є невід'ємною частиною життя кожного індивіда [1]. Сучасні технології та зростаючий обсяг інформації ставлять перед освітою нові виклики та завдання. Самостійна робота студентів стає дедалі важливішою, інформаційний простір стає все більш насиченим. Проблема "двох сигм", виявлена Бенджаміном Блумом в 1984 році, показала, що студенти, які навчаються самостійно з допомогою менторів, досягають значно кращих результатів у навчанні, завдяки персоналізованому підходу [2]. Це свідчить про необхідність покращення навчального процесу та надання студентам інструментів для ефективної самостійної роботи.

Розробка Android-додатку для організації самостійної роботи студентів стала актуальною, оскільки технологічний розвиток та поширення мобільних пристроїв відкривають нові можливості для покращення навчального процесу. Мобільні пристрої Android є широко поширеними та доступними, що робить їх ідеальною платформою для розробки додатків для освіти [3]. Важливою перевагою такого додатку є можливість використання його в офлайн-режимі, що дозволяє студентам мати доступ до навчальних матеріалів будь де та будь коли. Насичений ринок навчальних додатків для Android має свої недоліки, такі як відсутність персоналізації, незручність використання та наявність реклами. Тому створення додатку, який забезпечує зручний інтерфейс, персоналізований підхід до навчання та відсутність зовнішніх рекламних відволікань, стає важливим завданням.

Розробка Android-додатку для організації самостійної роботи студентів стає ще більш актуальною завдяки використанню інструментів штучного інтелекту. Один із ключових елементів цього додатку - це чат-бот, який використовує інтелектуальну модель chatGPT [4].

Чат-бот, опрацьовуючи запитання та вводячи студента в діалог, може надати корисні поради, пояснити складні матеріали, надати рекомендації щодо планування навчального процесу та навіть створити індивідуальні програми навчання. Модель chatGPT, завдяки своєму штучному інтелекту, може розпізнавати особисті особливості студента та адаптувати вміст та рекомендації під його потреби. Цей інноваційний підхід до навчання дозволить студентам отримувати персоналізовану та ефективну підтримку під час самостійної роботи, сприяючи підвищенню їхнього академічного успіху та розвитку навичок самостійного навчання. Такий додаток стає важливим інструментом в підтримці студентів та полегшує доступ до освіти.

Паралельно із чат-ботом-помічником, розроблений додаток включатиме чат-бота-екзаменатора, який також використовуватиме потужну модель chatGPT з вбудованим промптом. Такий чат-бот буде слугувати студентам інструментом для підготовки до іспитів та оцінювання їхнього рівня знань. Чат-бот-екзаменатор здатний створити індивідуальні тести та завдання для студентів, а також проводити оцінку їхніх відповідей. Завдяки вбудованому промпту, він може адаптувати запитання та завдання під конкретний матеріал, що вивчається, допомагаючи студентам систематизувати та закріпити знання. Цей чат-бот сприятиме покращенню підготовки до іспитів та допоможе студентам ефективно визначати свій рівень засвоєння матеріалу. Такий інтелектуальний інструмент допоможе студентам не лише навчатися ефективніше, але й більш впевнено впроваджувати отримані знання в практику.

Таким чином, розробка Android-додатку для організації самостійної роботи студентів з використанням технологій штучного інтелекту є актуальною та важливою задачею, яка сприяє покращенню навчального процесу та підвищенню якості освіти.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою роботи є підвищення ефективності освітнього процесу студента шляхом організації самостійної роботи за допомогою функціоналу мобільного додатку.

Основними задачами дослідження є:

- визначити функціонал навчальної системи «Live To Learn», методи і засоби її реалізації;
- розробити алгоритми генерування рекомендацій для створення розкладу самостійної роботи студента з врахуванням коефіцієнта навантаженості студента;
- розробити алгоритми генерування графічних звітів із статистикою навантаження задачами;
- розробити зручний та легкий для розуміння інтерфейс;
- розробити методи та засоби роботи чат-боту помічника використовуючи технології штучного інтелекту;
- розробити програмні компоненти Android-додатку;
- розробити інтерфейс програмного продукту;
- провести тестування програмного продукту.

Об’єкт дослідження – процеси створення навчальної системи, що включає в себе процеси збереження, сортування та аналізу інформації про постановку задач навчання користувача, генерування графічних звітів статистики по навантаженню та відповідей моделі штучного інтелекту.

Предмет дослідження – методи та засоби реалізації Android-додатку для організації самостійної роботи студента з використанням технологій штучного інтелекту.

Методи дослідження. У процесі дослідження використовувались: теорії чисел і математичної статистики та методи аналітичної геометрії для генерування звітів; методи алгоритмізації процесів для розробки алгоритмів роботи модулів системи з урахуванням їх функціонального призначення; методи проектування програмного забезпечення для розробки архітектурної

моделі програмної системи, методи тестування для перевірки роботи створеного програмного додатку.

Наукова новизна отриманих результатів.

1. Вперше запропоновано програмну модель екзаменатора на основі моделі штучного інтелекту gpt-3.5-turbo з вбудованим промптом в якості генератора питань, суть якого в адаптації запитань та завдань до конкретного матеріалу, що дозволяє чат-боту моделювати процес підготовки студентів до іспитів та оцінювати рівень їх знань, сприяючи систематизації та закріпленню знань.

2. Подальшого розвитку отримав метод генерування рекомендацій для складання розкладу СРС на основі розрахунку коефіцієнта навантаженості, у якому, на відміну від існуючих методів, запропонована нова багатофакторна модель, яка враховує не лише кількість завдань, але їх складність, що дає можливість спрямовано використовувати часові ресурси та забезпечити визначення оптимальної тривалості самостійної роботи студента.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмні засоби для підвищення ефективності організації самостійної роботи студента, які можна використати в навчальному процесі закладів освіти.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботах написаних у співавторстві автору належить: [7] – моделі, методи та програмні засоби для реалізації Android-додатку для організації самостійної роботи студента з використанням технологій штучного інтелекту; [11] – розробка чат-боту екзаменатора за допомогою промпт інженерії.

Апробація матеріалів. Основні положення й результати досліджень представлені на «LII науково-технічній конференції підрозділів ВНТУ – 2023»

та міжнародній науково-практичній конференції «Електронні інформаційні ресурси: створення, використання, доступ - 2023».

Публікації. Основні результати досліджень опубліковано в наукових працях:

- тези доповіді на «LII науково-технічній конференції підрозділів ВНТУ – 2023» [7].
- тези доповіді на міжнародній науково-практичній конференції «Електронні інформаційні ресурси: створення, використання, доступ - 2023» [11].

Структура та обсяг роботи. Магістерська кваліфікаційна робота містить вступ, п’ять розділів, висновки, список літератури, що містить 30 найменувань, та 4 додатки. Робота містить 33 рисунків та 14 таблиць. Ілюстративний матеріал до роботи подано у додатку Г.

1 АНАЛІЗ СТАНУ ПИТАННЯ ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ СТУДЕНТА

1.1 Аналіз стану питання організації самостійної роботи студента та роль штучного інтелекту в ньому

Розвиток суспільства та науково-технічний процес глибоко проник в життя кожної людини, в тому числі і студента. Важливим напрямком розвитку освітнього процесу стало впровадження засобів програмного забезпечення в систему навчання. Організація самостійної роботи студента за допомогою сучасних технологій забезпечує оптимальний курс для кожного окремого студента, покращує швидкість та якість вивчення матеріалу, дає можливості для корегування навчального плану, розвиває навички аналітичної та дослідницької діяльності, пропонує можливість самостійної роботи, контроль якості здобутих знань і вмінь та економію часу студента [5].

Зародження освітньої діяльності почалось ще в часи первісних людей, коли діти повторювали жести, вигуки, навички своїх батьків, переймаючи їх досвід. Однак з розвитком суспільства навчання стає все більш академічним, освіта все менш пов'язана з повсякденним життям, а форми та завдання освітнього процесу зазнають змін [6]. Кожен студент має свої індивідуальні потреби та рівень знань. Відсутність персоналізованих підходів у завданнях для самостійної роботи може призводити до низької мотивації та неефективного вивчення матеріалу. Вирішенням даної задачі є використання спеціалізованих програмних додатків, націлених на персоналізацію навчального процесу, для наповнення та розподілу власно обраних дисциплін за категоріями, постановкою часу на вивчення матеріалів та визначення приорітетності цілей.

Використання технологій штучного інтелекту в освіті може суттєво покращити ефективність навчання та надати персоналізований підхід до кожного студента. Такі технології можуть допомогти в удосконаленні процесів оцінювання, забезпеченні доступу до навчальних матеріалів,

забезпеченні більш точної діагностики та індивідуальної підтримки для учнів із різними потребами. Проте на разі існує ряд проблем з використанням технологій штучного інтелекту в навчальному процесі, такі як конфіденційність даних, вирішення етичних проблем та персоналізація генерації контенту під потреби студента.

Штучний інтелект (ШІ) - це галузь інформатики, що вивчає розробку програм та алгоритмів, що дозволяють комп'ютеру навчитися вирішувати завдання, які зазвичай вимагають людської інтелектуальної діяльності.

Головним завданням роботи є створення мобільного додатку для самостійної роботи студента з використанням технологій штучного інтелекту. Метою такого програмного продукту є підвищення ефективності організації самостійної роботи студента шляхом збереження інформації про його завдання і їх сортування, створення графічних звітів зі статистикою навантаження та інтеграції чат-боту помічника та чат-боту екзаменатора [7].

Робота із моделями машинного навчання тісно пов`язана з поняттям «promt engeneering» [8]. Promt Engeneering – це інженерія створення та оптимізації запитів для моделей машинного навчання, з метою отримання бажаного результату. Вміючи правильно використовувати запити, можна створити багато корисних інструментів для навчання студента. Перевагами такої взаємодії із моделями штучного інтелекту є:

- Індивідуалізація навчання – за допомогою промптів можна структуризувати завдання та інструкції для кожного студента, враховуючи його поточний рівень знань;
- Покращення швидкості відповідей – зазвичай мовні моделі машинного навчання намагаються відповідати на питання дуже відкрито, використовуючи промпти можна зменшити кількість тексту для отримання відповіді, що дозволяє негайно отримувати зворотню інформацію;
- Функціонал перевірки знань – задавши потрібні налаштування, можна перетворити модель штучного інтелекту на екзаменатора, який буде займатись перевіркою знань та давати поради щодо вивчення предмету.

Проте, важливо пам’ятати, що використання промптів пов’язане зі своїми труднощами й обмеженнями, тому важливо правильно формулювати свої запити, щоб отримати потрібну інформацію. Слід також враховувати точність і достовірність відповідей.

Так як розвиток мовних моделей штучного інтелекту набрав великих обертів не так давно, на разі існує не так багато додатків де впроваджуються інструменти штучного інтелекту для покращення навчання студентів. Тому створення такого роду Android-додатку є нелегкою задачею, адже доведеться використовувати технології API для підключення моделі штучного інтелекту та створити авторський промпт, який ще не використовувався раніше.

1.2 Порівняльний аналіз аналогів

Освіта пережила значні зміни через пандемію COVID-19. Для запобігання поширенню вірусу, уряди багатьох країн почали вводити обмеження, у тому числі закриття навчальних закладів. Через це освітнім закладам потрібно було переходити на дистанційний формат, що призвело до суттєвих змін у процесі навчання та наукової діяльності. Великої популярності набули застосунки для організації самостійного навчання учнів та студентів. Розглянемо найбільш популярні з них:

Smart Study – безкоштовний додаток створений студентами для студентів. Офіційна назва розробників – OP Design. Додаток дозволяє керувати та відслідковувати завдання, легко та швидко створювати індивідуальний розклад навчання, надавати вказівки під час виконання розкладу та надавати статистику навчання, яка дозволяє учням зрозуміти їхні звички у навчанні, ефективність і розподіл часу. Інтерфейс додатку зображено на рисунку 1.1.

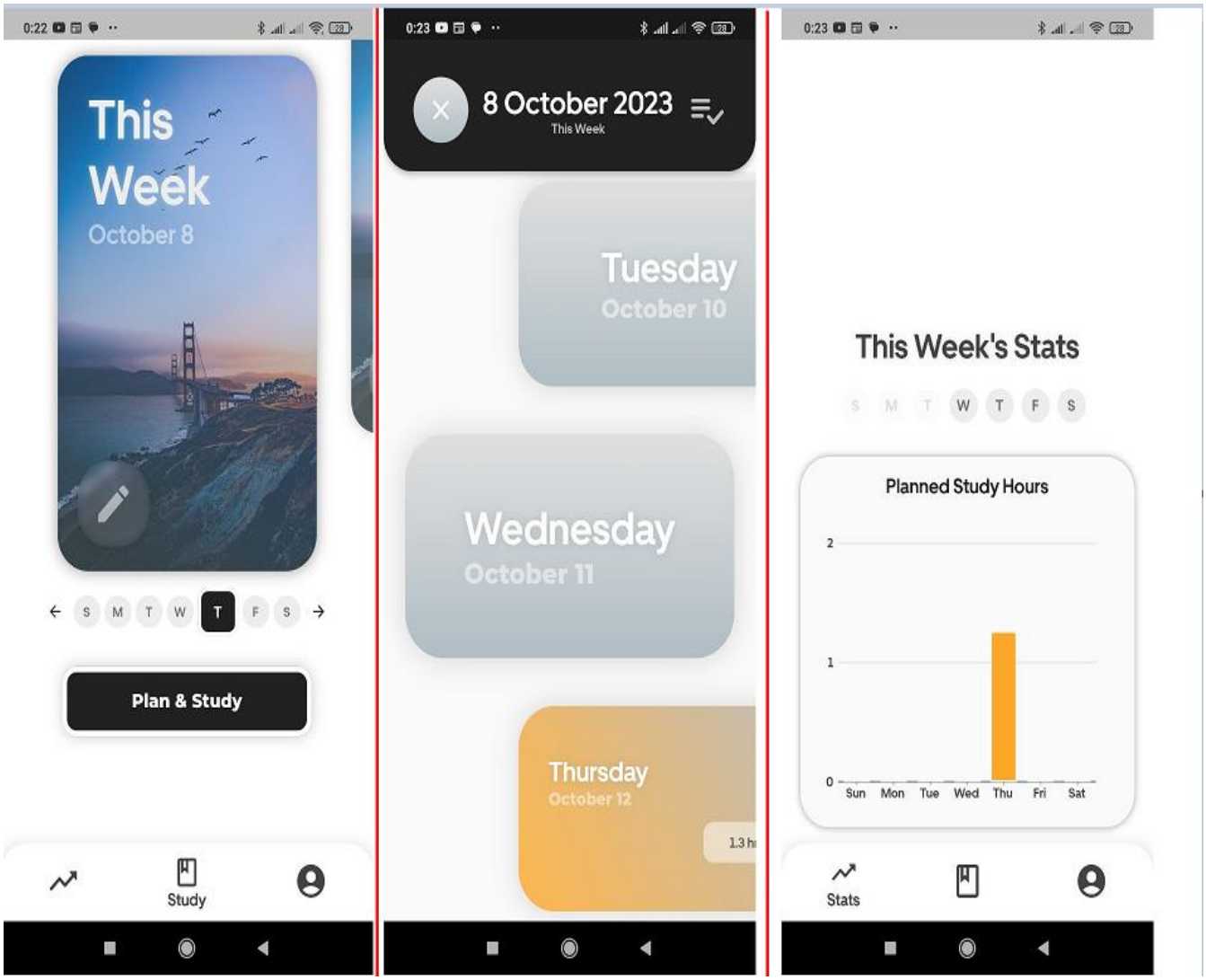


Рисунок 1.1 – Інтерфейс додатку Smart Study

Vaia – додаток для підготовки учнів та студентів до екзаменів. Створений компанією Vaia. Функціонал додатку передбачає створення флеш-карток з короткими відомостями, для завчання перед екзаменом, також пропонує велику кількість курсів для покращення знань та тести для перевірки вивченого матеріалу. Серед переваг додатку є функція пояснювання термінів за допомогою штучного інтелекту, проте велика частина функціоналу є платною. На рисунку 1.2 зображено інтерфейс додатку.

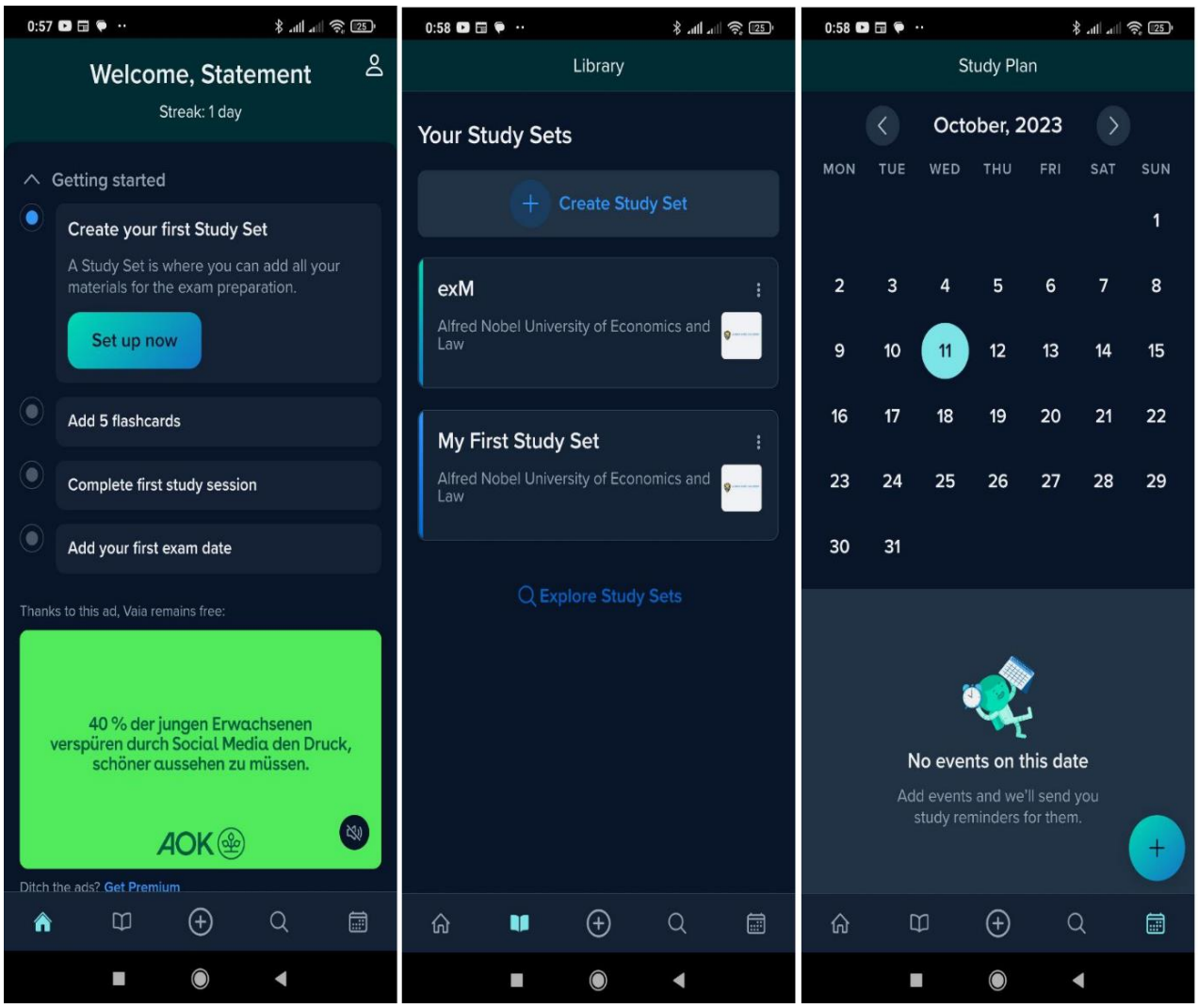


Рисунок 1.2 – Інтерфейс додатку Vaia

Focus To-Do – додаток для управління завданнями, подіями, нагадуваннями, що допомагає зосереджуватись на роботі та навчанні. До переваг даного додатку можна віднести функцію синхронізації даних зі всіх пристроїв та зручний віджет для головного екрану смартфону. Основні недоліки – велика частина функціоналу платна, відсутність сортування задач за категоріями. Існує для операційних систе м Android, IOS, Windows та MacOS. Розробник – «SuperElement Soft». На рисунку 1.3 зображено інтерфейс додатку.

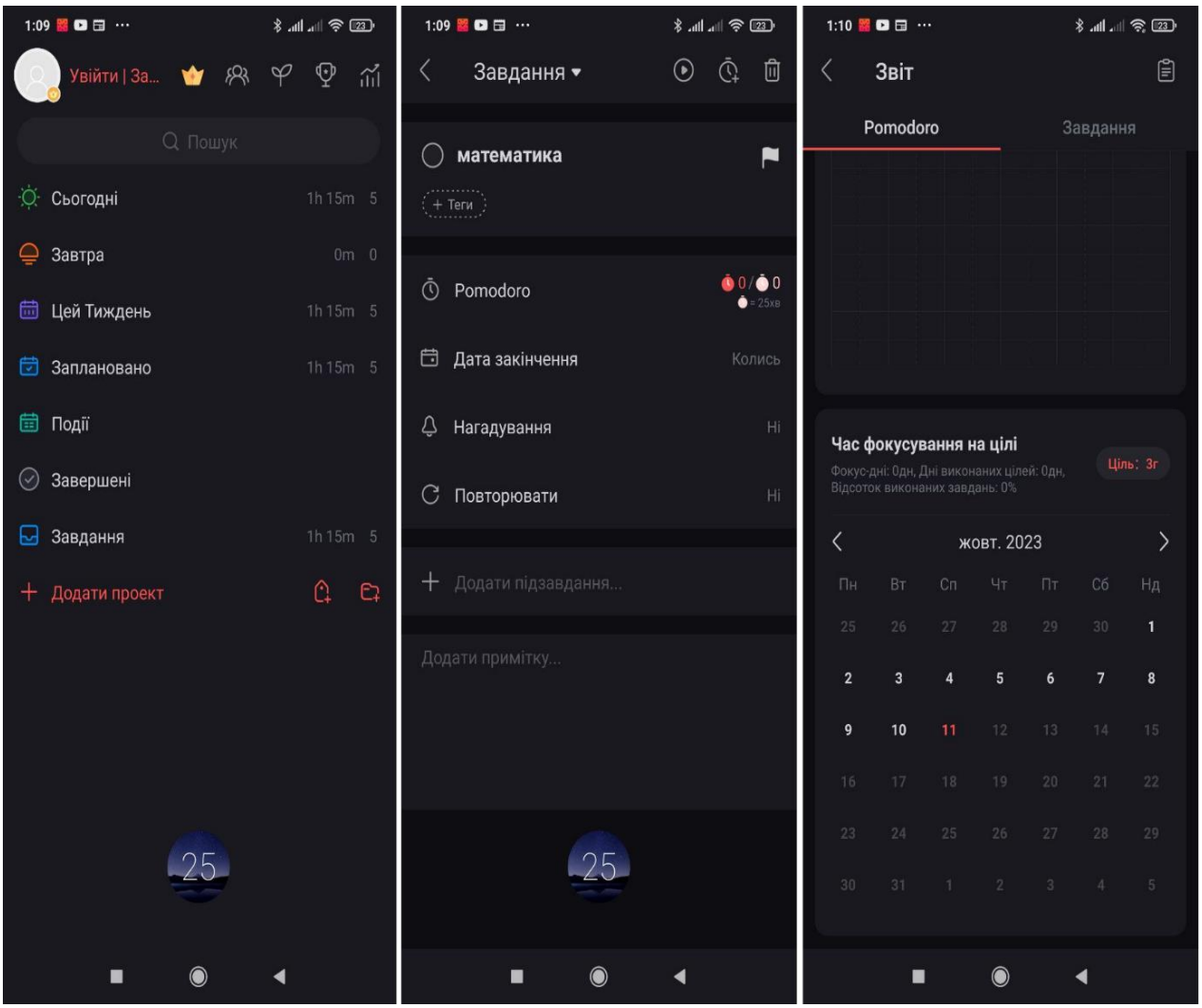


Рисунок 1.3 – Інтерфейс додатку Focus To-Do

JetIQ – це інтегрована клієнт-серверна навчальна система, в якій реалізовані функції дистанційного та змішаного навчання і управління закладом. Головними перевагами додатку є моніторинг результатів навчання, доступ до бази електронних посібників та комунікації з викладачами. До основних недоліків можна віднести відсутність персоналізації навчальних задач, нестабільна робота додатку та обмеженість у використанні лише студентами ВНТУ. Розробниками є колектив викладачів та студентів університету, які систематично вдосконалюють роботу і функціонал додатку. На рисунку 1.4 наведено користувацький інтерфейс додатку.

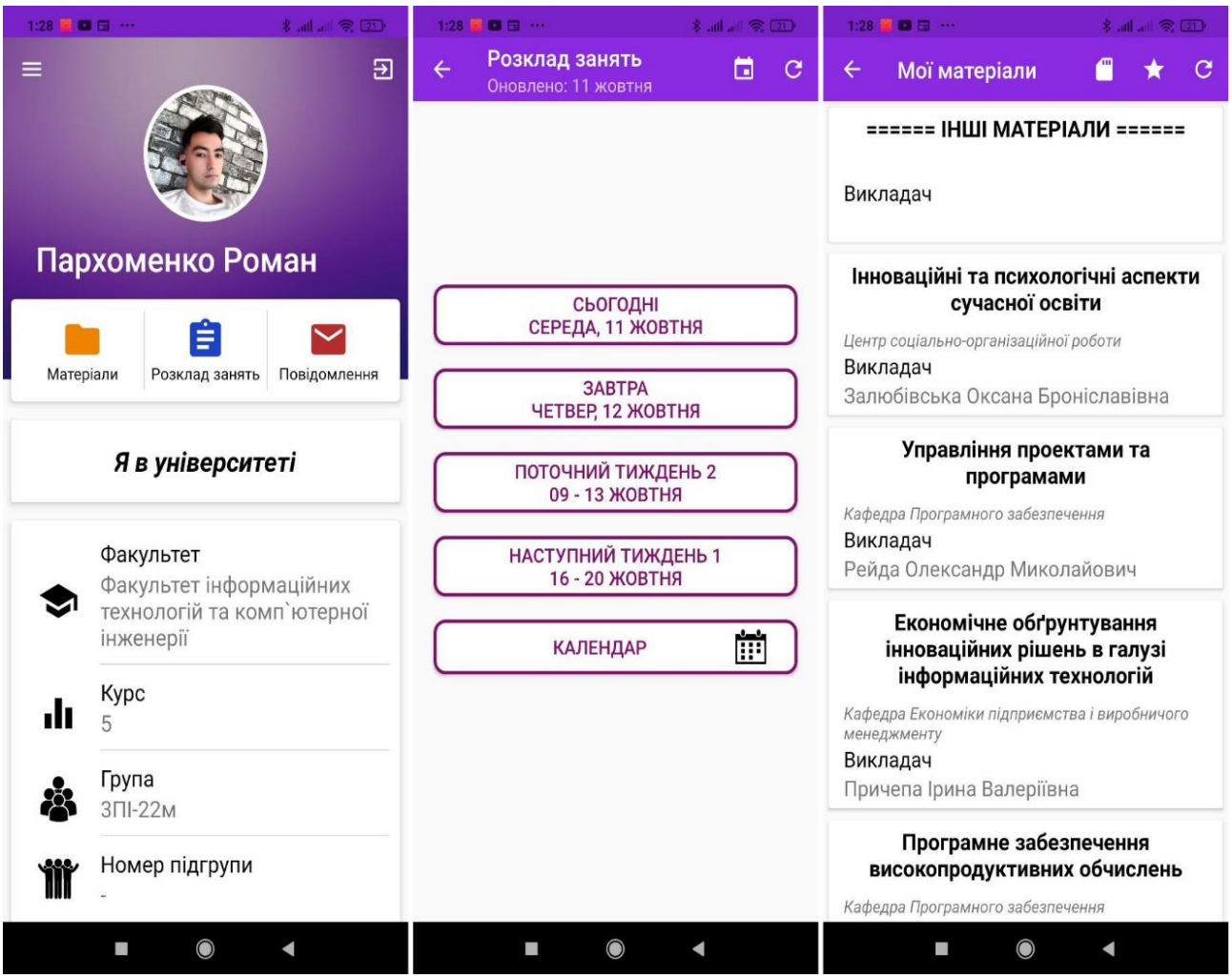


Рисунок 1.4 – Інтерфейс додатку JetIQ

Визначивши можливості, переваги та недоліки додатків було створено таблицю (табл. 1.1) для порівняння аналогів та розроблювального додатку «Live To Learn».

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	Smart Study	Vaia	Focus To-Do	JetIQ	Live To Learn
Створення власних категорій задач	+	-	-	-	+
Безкоштовність повного функціоналу	+	-	-	+	+
Використання технологій штучного інтелекту	-	+	-	-	+

Продовження таблиці 1.1

Автономна робота без мережі Internet	+	-	+	-	+
Кросплатформність	+	+	+	+	-
Можливість перевірки знань за допомогою тестів	-	+	-	-	+
Підсумковий результат	4	3	2	2	5

Отже, проаналізувавши конкурентний ринок і порівнявши наш додаток з провідними рішеннями, було визначено переваги та недоліки наших конкурентів. Виходячи з цього можна зробити висновок, що розробка нашого додатку обіцяє більше можливостей, ніж найпопулярніші аналоги на даний момент, що підтверджує доцільність розробки цього програмного продукту.

1.3 Постановка задач розробки

Після аналізу стану досліджуваної теми, пов'язаної із організацією самостійної роботи студента та використання технологій штучного інтелекту для покращення ефективності навчання, а також порівняння існуючих рішень з використанням визначених критеріїв, було ідентифіковано конкретні завдання, які потрібно виконати для подальшого дослідження та розробки мобільного застосунку «Live To Learn»:

- розробити алгоритм взаємодії моделі штучного інтелекту з користувачем, для надання інформації та відповідей на запитання користувача;
- розробити алгоритм поведінки моделі штучного інтелекту в якості екзаменатора, використовуючи технології промпт-інженерії;
- розробити алгоритми генерування графічних звітів статистики навантаження;
- розробити інтерфейс програмного продукту, який буде зручним та зрозумілим для користувачів;
- визначити найбільш ефективний метод зберігання інформації;
- провести тестування програмного продукту.

1.4 Висновки

У першому розділі був проведений аналіз поточного стану питання щодо організації самостійної роботи студента, а також розглянуто можливості використання інструментів штучного інтелекту для покращення навичок студентів, набутих під час навчання. Під час аналізу були ідентифіковані труднощі, пов'язані з використанням технологій штучного інтелекту. Також було досліджено можливі шляхи вирішення цих проблем та технічні особливості реалізації програмного продукту. Крім того, було проаналізовано аналоги систем для організації самостійної роботи студента та порівняно їх із розроблюваним продуктом за різними критеріями. Основними аналогами були такі системи, як Smart Study, Vaia, Focus To-Do і JetIQ. В результаті нашого порівняння була виокремлена доцільність розробки програмного продукту магістерської кваліфікаційної роботи. З огляду на недоліки аналогів були сформульовані основні завдання, які потрібно вирішити в процесі розробки. Ці завдання включають розробку алгоритму взаємодії моделі штучного інтелекту з користувачем, для надання інформації та відповідей на запитання користувача, розробку алгоритму поведінки моделі штучного інтелекту в якості екзаменатора, використовуючи технології промпт-інженерії, розробку алгоритмів генерування графічних звітів статистики навантаження, проектування зручного та зрозумілого інтерфейсу, аналіз найбільш ефективного методу зберігання інформації, а також проведення кінцевого тестування розробленого програмного продукту.

2 РОЗРОБКА МЕТОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ

2.1 Розробка методу роботи чат-боту екзаменатора із використанням інструментів промпт-інженерії

Персоналізація навчання - це підхід до освіти, що покликаний враховувати унікальні потреби, інтереси, темп та рівень знань кожного учня. Персоналізація освітнього процесу в часи дистанційного навчання може бути досягнута за допомогою інтерактивних онлайн-ресурсів, персональних завдань, адаптивних платформ, та використанням інших технології [9]. Цей підхід допомагає забезпечити ефективніше навчання для кожного студента, зберігаючи якість освіти в дистанційних умовах. Один із способів досягнення цієї мети в сучасних освітніх системах полягає в використанні промпт інженерії.

Промпти - це конкретні запити, інструкції або питання, які використовуються в системах штучного інтелекту для взаємодії з користувачами. Промпти в контексті навчання представляють собою спеціально створені запити, інструкції або питання, які спрямовані на активізацію відповідей та дій студентів в системах з використанням штучного інтелекту [10]. Аналіз ефективності промптів у підтримці процесу навчання спрямований на визначення того, наскільки ці інструкції або запити допомагають студентам навчатися більш ефективно. Ось ключові аспекти цього аналізу:

- Покращення знань і навичок: ефективні промпти повинні сприяти покращенню рівня знань і навичок студентів. Аналіз полягає в оцінці того, наскільки добре студенти відповідають на промпти та які конкретні питання чи інструкції найбільше сприяють набуттю знань.
- Залучення студентів: Промпти повинні бути цікавими та мотивуючими для студентів. Аналіз включає в себе оцінку того, наскільки студенти активно взаємодіють і відповідають на пропоновані промпти.

- Підвищення рівня самостійності: добре розроблені промпти сприяють розвитку самостійності у студентів. Аналіз полягає в оцінці того, наскільки студенти в змозі вирішувати завдання та проблеми, які вони зустрічають за допомогою промптів.
- Оцінка процесу навчання: процес навчання за допомогою промптів може бути відстежений і оцінений за допомогою даних, які системи штучного інтелекту збирають під час взаємодії студентів. Аналіз полягає в оцінці прогресу студентів та ідентифікації можливих слабких місць в навчанні.
- Підвищення ефективності навчального процесу: аналіз ефективності промптів також дозволяє визначити, чи варто змінювати або оптимізувати інструкції та питання, щоб досягти кращих результатів в навчанні.

Для розробки чат-боту було обрано мовну модель штучного інтелекту «gpt-3.5-turbo», яка підключається за допомогою OpenAIApi. ChatGPT - це модель глибокого навчання, яка базується на архітектурі трансформера і була розроблена для генерації природної мови. Вона може 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 [11].

Для того щоб модель працювала в якості екзаменатора, який повинен ставити питання по вказаній користувачем темі, та давати оцінку його відповіді на це запитання, потрібно вказати промпт перед початком роботи чат-боту. Для реалізації цього промпту ми в процесі написання коду задаєм налаштування для роботи моделі і ховаєм від користувача цей промпт, таким чином, що йому потрібно лише ввести назву теми по якій він хоче пройти опитування. Також важливим аспектом розробки такого чат-боту є те, що стандартна робота цього API не передбачає запам`ятовування попередніх запитів та відповідей, тому нам кожного разу потрібно надсилати попередні запити та відповіді до серверу і приховувати їх в самому чаті для збереження коректного виводу інформації.

Алгоритм складається з таких кроків:

1. Створення JSON-об`єкту для взаємодії з API
2. Створення JSON-об`єкту для створення запиту.
3. Створення запиту за допомогою бібліотеки OkhtttpClient.
4. Виклик асинхронного запиту до сервера.
5. Обробка помилки при невдалому запиті.
6. Обробка успішної відповіді та оновлення інтерфейсу.
7. Обробка невдалої відповіді.

На рисунку 2.1 зображено тіло запиту до серверу ChatGPT в якому ми використовуєм самостійно створений промпт.

```
JSONObject jsonBody = new JSONObject();
JSONObject obj = new JSONObject();
try {
    jsonBody.put( name: "model",  value: "gpt-3.5-turbo");
    obj.put( name: "role",  value: "user");
    if (i<1) {
        obj.put( name: "content",  value: "You are an examiner. Please ask me some questions for the topic: " + question +
                "\\nFor example:\n" +
                "1. You asking\n" +
                "2. I answer\n" +
                "3. you're saying if my answer was correct. If I was incorrect say \"Incorrect!\" " +
                "and critique my response to each question and provide example answers then ask another question. " +
                "If I was correct say \"Correct!\" and ask another question.\n" +
                "Begin by asking the first question and then waiting for my response. ");
        i++;
    } else {
        obj.put( name: "content", question);
    }
    messageArray.put(obj);
    jsonBody.put( name: "messages", messageArray);
```

Рисунок 2.1 – Тіло запиту до серверу chatgpt

Завдяки цим налаштуванням наша модель штучного інтелекту буде поводити себе як екзаменатор, і продовжуватиме ставити питання після відповіді користувача. Якщо користувач дасть правильну відповідь, екзаменатор повинен написати, що відповідь була правильною і поставити наступне питання, а якщо відповідь була неправильна екзаменатор повинен пояснити, чому відповідь була неправильною і показати приклад правильної відповіді, після чого перейти до наступного питання.

2.2 Розробка методу генерації рекомендацій для створення розкладу самостійної роботи з врахуванням коефіцієнта навантаженості студента

У сучасному освітньому середовищі студенти часто стикаються з необхідністю виконувати самостійну роботу і вирішувати завдання в позааудиторний час. Оптимальне розподілення часу та завдань важливо для забезпечення успішного навчання та зниження ризику стресу та перенавантаження.

Потрібно дослідити поняття коефіцієнта навантаженості студента і розглянути, як цей показник може бути використаний для оцінки навчального процесу. В дослідженні американських викладачів було визначено, що в середньому студенти витрачають до 4 годин в день на самостійне опрацювання матеріалів, що є оптимальною кількістю часу, перед тим як процес навчання починає втрачати свою ефективність (рисунок 2.2) [12].

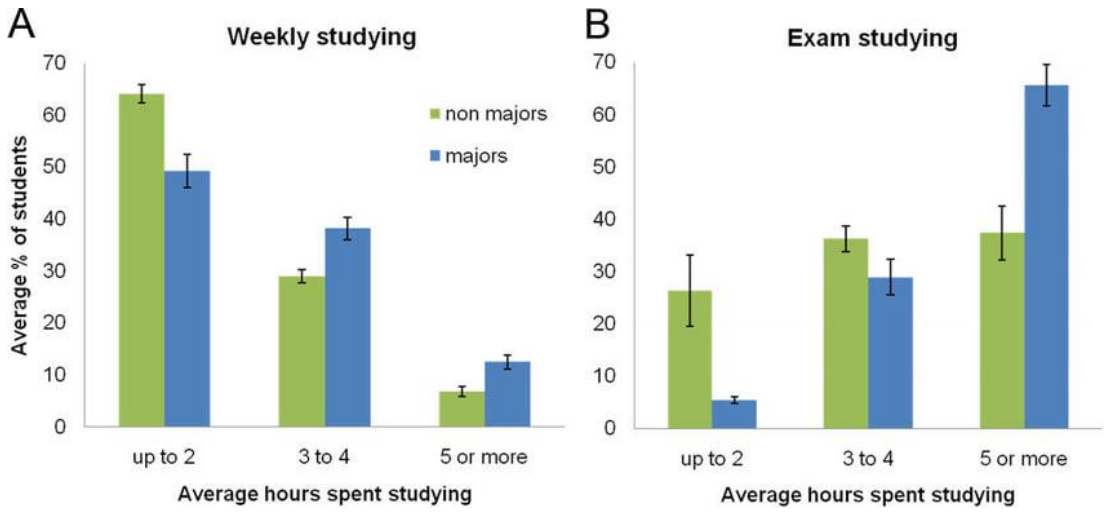


Рисунок 2.2 – Статистика використання часу студентів для самостійного навчання

Враховуючи ці показники було створено формулу для обрахування коефіцієнта навантаженості студента, яке буде супроводжуватися рекомендаціями, щодо використання часу.

Коефіцієнт навантаженості обчислюється за наступною формулою (2.1):

$$K = 240 - (x * 30) \quad (2.1)$$

де:

K - коефіцієнт навантаженості студента.

240 - кількість хвилин в 4 годинах, що відповідає оптимальній кількості часу на самостійну роботу в день.

x - сума рівнів складності задач, де 1 рівень дорівнює 30 хвилин на виконання, 2 рівень - 60 хвилин, а 3 рівень - 90 хвилин.

Коефіцієнт K може бути використаний для визначення того, наскільки студент завантажений у певний день. Якщо результат обчислення K менший за 0, це свідчить про перенавантаження студента, що може вплинути на якість його навчання та фізичний стан. У такому випадку, студент повинен розглянути можливості для розвантаження свого графіку, оптимізації роботи, або навіть розділити завдання на декілька днів. І навпаки, якщо коефіцієнт K більший за 0, це означає, що студент має достатньо часу для виконання своїх завдань, і його навантаженість є оптимальною. Проте, якщо кількість задач на вказаний день дорівнює нулю, обрахування коефіцієнта не відбувається і користувач отримує рекомендацію використати цей день для того, щоб розвантажити інші, більш завантажені дні. Блок схема алгоритму генерування рекомендацій зображена на рисунку 2.3.

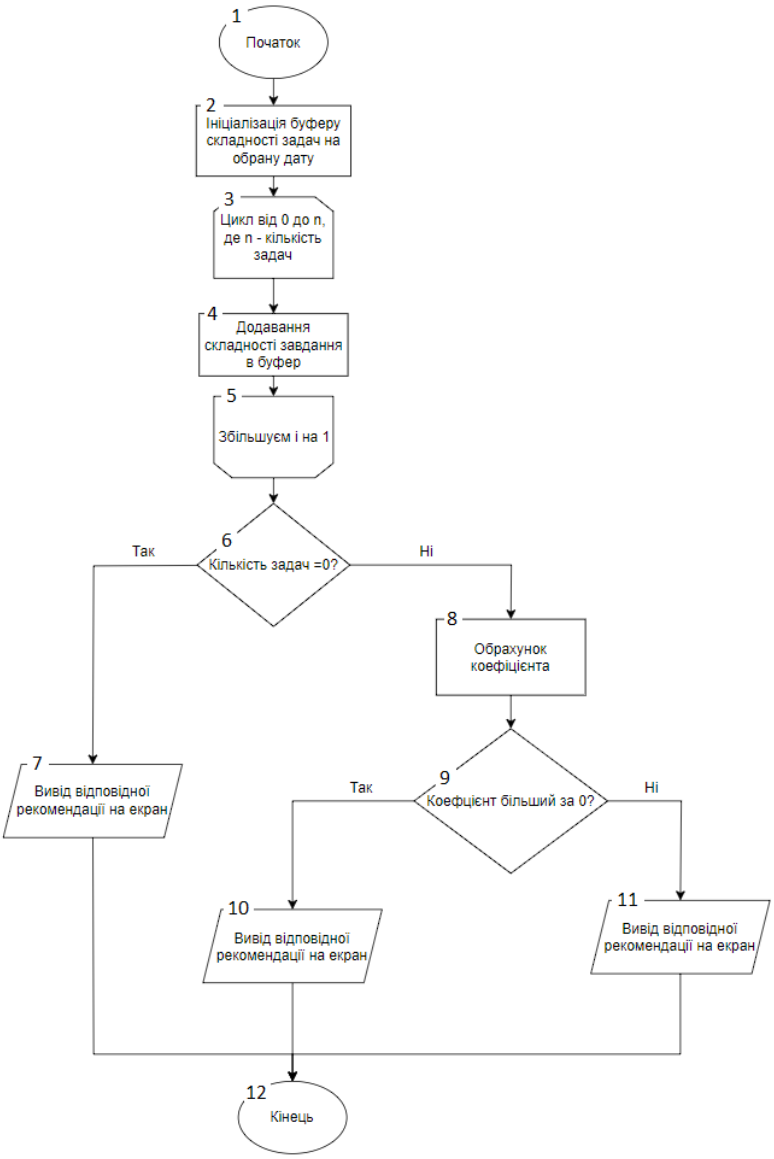


Рисунок 2.3 – Блок-схема алгоритму

Використання коефіцієнта навантаженості може бути корисним інструментом для студентів, вчителів і освітніх закладів. Студенти можуть використовувати цей показник для планування свого навчального режиму та визначення оптимального співвідношення між кількістю завдань і їх складністю. Вчителі можуть рекомендувати студентам способи оптимізації їхньої роботи, щоб досягти кращих результатів. Освітні заклади можуть використовувати дані про коефіцієнт навантаженості для вдосконалення навчальних програм та розподілу завдань.

Отже, коефіцієнт навантаженості студента є важливим інструментом для оцінки та планування навчального процесу. Його використання допомагає

студентам уникнути перенавантаження та забезпечувати оптимальну робочу обстановку для досягнення успіху у навчанні. Вивчення та розуміння цього показника сприяє покращенню навчального процесу і досягненню кращих результатів студентами.

2.3 Архітектурне проектування системи та проектування структурної схеми інтерфейсу

Архітектурне проектування визначає структуру, компоненти та взаємозв'язки системи, зокрема логічні та фізичні компоненти, шари, модулі, патерни, архітектурні стилі тощо. Його ціллю є 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 [13]. Архітектура Android-додатку для організації самостійної роботи студента складається з окремих модулів. На рисунку 2.4 зображено концептуальну схему системи.

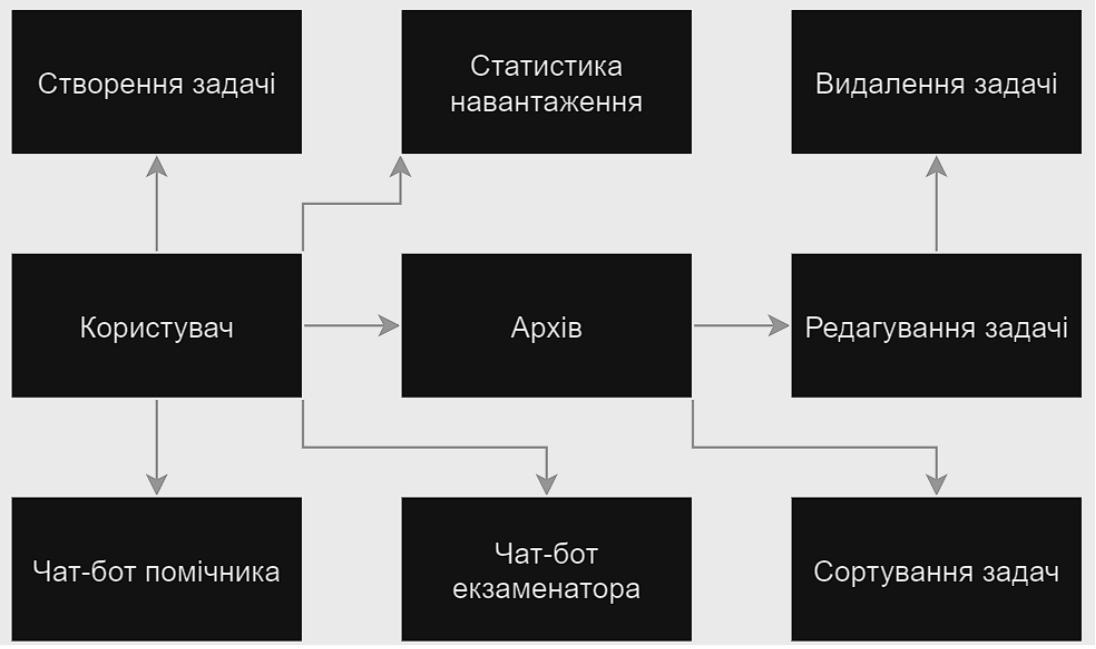


Рисунок 2.4 – Концептуальна схема системи

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	Створення задачі	Користувач має можливість створити нову задачу
Користувач	Архів	Користувач може переглядати раніше створені задачі в архіві
Користувач	Статистика навантаження	Користувач має можливість Переглянути графічний звіт зі статистикою навантаження
Користувач	Чат-бот помічника	Користувач має можливість звернутись до помічника
Користувач	Чат-бот екзаменатора	Користувач має можливість пройти тестування в чат-боті екзаменатора

Кожен модуль має чітку структуру та можливість мати атрибути та операції (рисунок 2.5). Атрибути модулю потрібні для ідентифікації того чи іншого стану чи процесу, а операції виконують якусь функцію з атрибутами модулів чи без них.

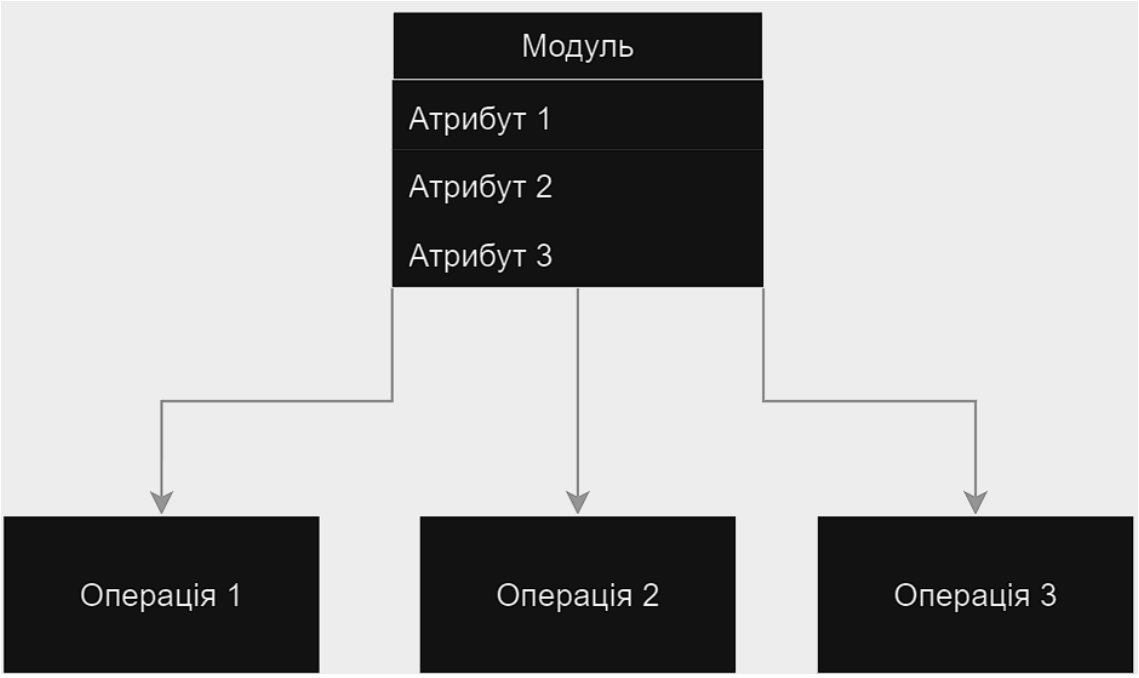


Рисунок 2.5 – Структура схеми модуля

У контексті архітектурного проектування, модуль - це окрема функціональна частина системи, яка виконує певну роль або виконує певні завдання. Модуль може бути самодостатнім компонентом системи, який працюватиме незалежно або буде взаємодіяти з іншими модулями.

Модулі допомагають розділити систему на логічні частини, що спрощує розробку, розуміння та підтримку системи. Кожен модуль може мати свою внутрішню структуру та інтерфейси для взаємодії з іншими модулями. Розподіл функціоналу великої системи на окремі модулі та визначення взаємозв'язків між ними дозволяє групувати завдання та цілі з метою спрощення процесу розробки.

Проектування структурної схеми графічного інтерфейсу для Android додатка - це важливий етап розробки, що вимагає детального підходу та уважного розгляду. У цьому процесі визначається, як компоненти і елементи інтерфейсу взаємодіятимуть один з одним, як користувач буде керувати додатком та як виглядатиме зовнішній вигляд інтерфейсу [14].

Спочатку потрібно визначити головні цілі та функціональність додатку, що буде вхідною точкою для подальшого проектування. Важливо визначити,

які екрани та функції повинні бути включені в інтерфейс. Під час цього процесу потрібно враховувати важливі аспекти, такі як інтуїтивність, доступність, ефективність та приємний дизайн. Треба обрати компоненти, які найкраще підходять для представлення функціональності системи, і розташовувати їх так, щоб користувачу було легко взаємодіяти з ними.

На рисунках 2.6 і 2.7 зображені структурні схеми інтерфейсів вікон системи.

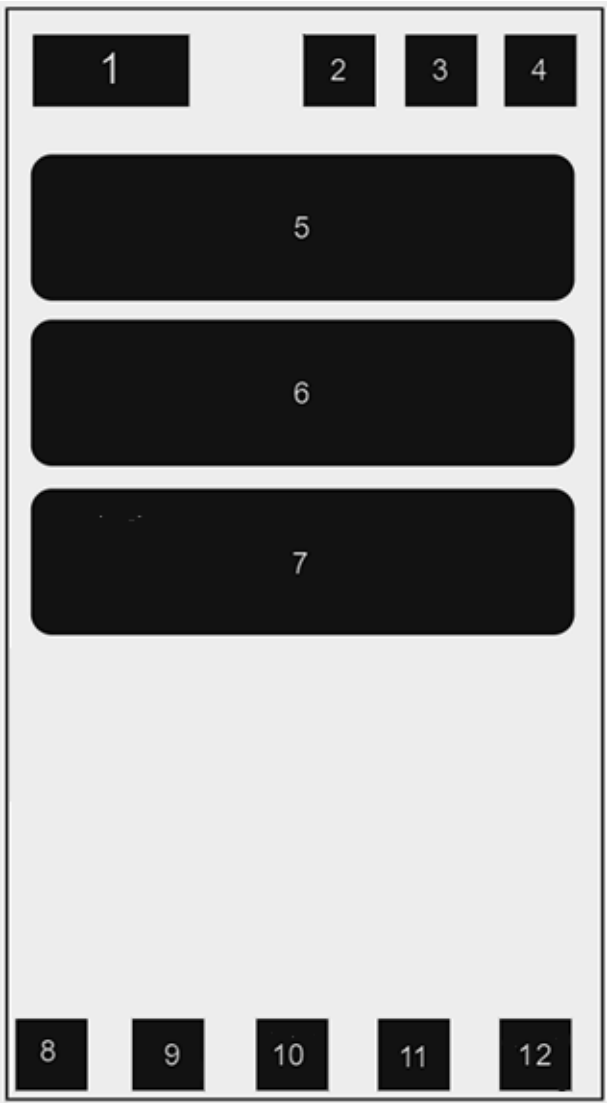


Рисунок 2.6 – Структурна схема головного вікна додатку

Основні елементи інтерфейсу головного вікна додатку:

1. Назва додатку.
2. Кнопка «Додати задачу».
3. Кнопка переходу до вікна чат-боту екзаменатора.

4. Кнопка переходу до вікна «Архів».
5. Календар для вибору дати.
6. Результат обчислення коефіцієнту навантаженості на обрану дату.
7. Випадаючий список задач на обрану дату розділений по категоріям.
8. Кнопка «Home».
9. Кнопка переходу до вікна «Статистика».
10. Кнопка переходу до вікна «Архів».
11. Кнопка переходу до вікна чат-боту помічника.
12. Кнопка «Вихід».

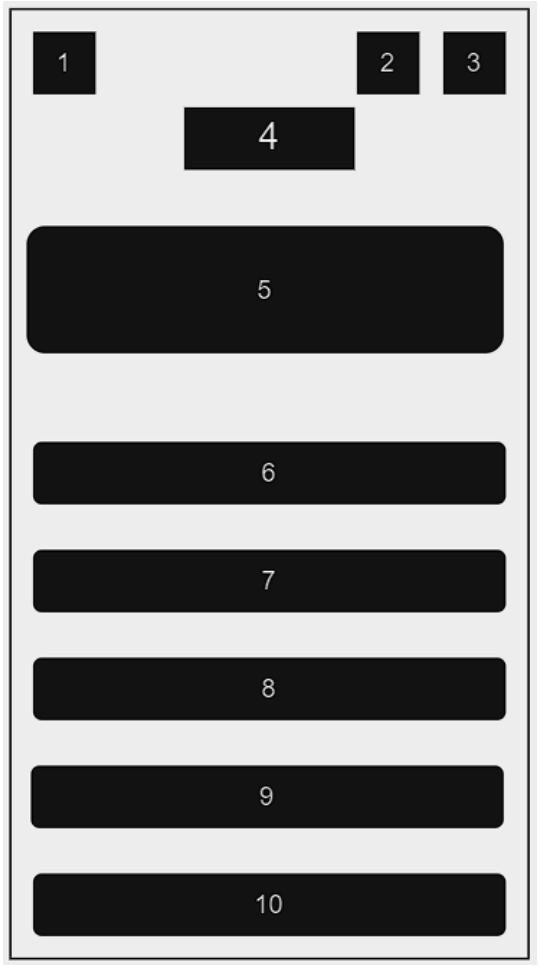


Рисунок 2.7 – Структурна схема вікна створення задачі

Основні елементи інтерфейсу вікна створення задачі:

1. Кнопка відміни створення задачі «Назад».

2. Кнопка «Зберегти» .
3. Кнопка певернення до вікна «Архів».
4. Назва вікна «Створення задачі».
5. Інструкція.
6. Поле введення назви задачі.
7. Поле введення наповнення задачі.
8. Поле введення категорії задачі.
9. Поле введення складності задачі.
10. Поле введення дати виконання задачі.

2.4 Розробка основних алгоритмів програмної реалізації

Оскільки головною метою додатку є організація навчального процесу, основним алгоритмом для його програмної реалізації є сортування завдань за категоріями (див. рисунок 2.8).

Алгоритм сортування завдань за категоріями включає такі кроки:

1. Початок.
2. Ініціалізація масиву буферів для категорій.
3. Початок циклу.
4. Отримання назви категорії для поточного завдання.
5. Перевірка наявності цієї категорії в масиві буферів для категорій завдань. Якщо категорія існує, перейти до кроку 7, в іншому випадку перейти до кроку 6.
6. Створення нового буфера для категорії завдань.
7. Додавання поточного завдання до буфера категорії.
8. Перехід до наступного завдання. Якщо ще є завдання, повернення до кроку 4, в іншому випадку завершення циклу і перехід до кроку 9.
9. Повернення відсортованих завдань за категоріями.
10. Кінець.

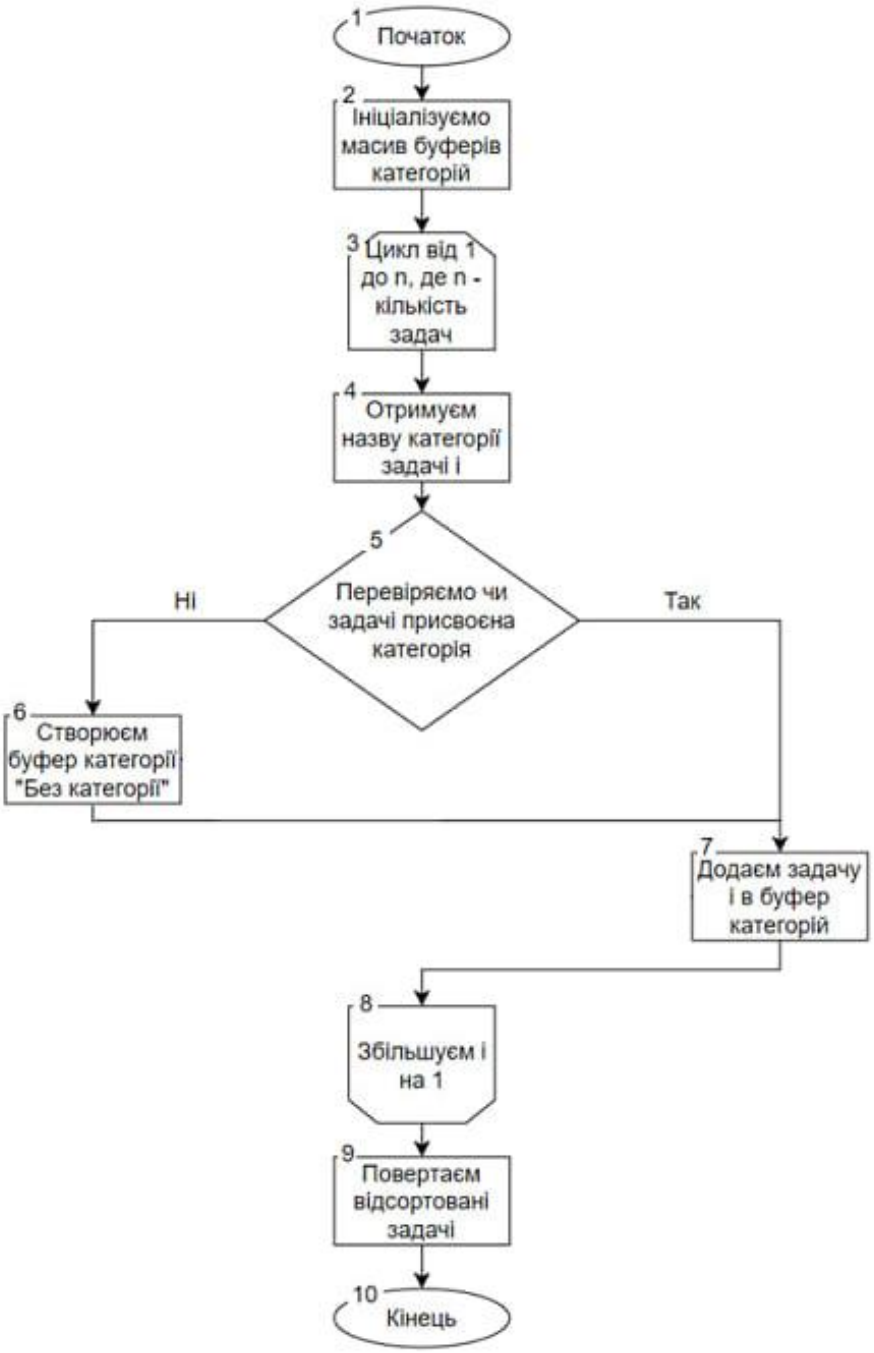


Рисунок 2.8 – Блок-схема роботи алгоритму сортування задач за категоріями

Далі було розроблено блок-схему алгоритму генерування графічного звіту статистики навантаження у вигляді стовпчикової діаграми, де вісь X відповідає датам, а вісь Y показує кількість завдань для кожної дати (рисунок 2.9).

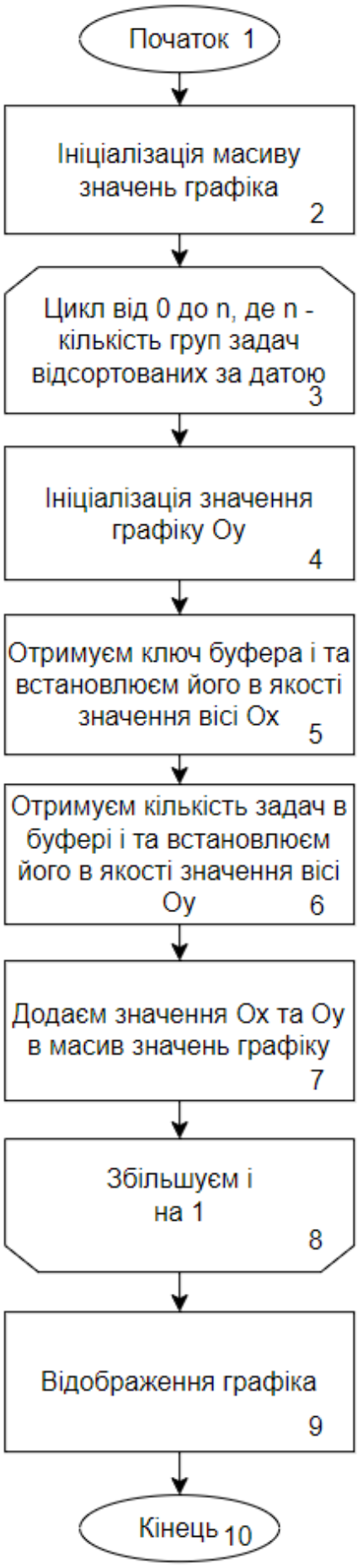


Рисунок 2.9 – Блок-схема алгоритму генерування стовбикового звіту

Алгоритм включає в себе такі кроки:

1. Початок.
2. Ініціалізація масиву значень графіка.
3. Початок циклу.
4. Ініціалізація значення графіку Оу.
5. Отримання ключа буфера і та встановлення його в якості значення вісі Ох.
6. Отримання кількості задач в буфері і та встановлення його його в якості значення вісі Оу.
7. Додавання значень Ох та Оу в масив значень графіку
8. Перехід до наступної дати.
9. Генерація графіка.
10. Кінець

Далі було розроблено алгоритм виводу задач відповідно до обраної дати, що допомагає користувачу ефективно планувати свій розпорядок дня та вчасно виконувати поставлені завдання. Вибір дати реалізовано в вигляді віджету календаря, що забезпечує не тільки приємний вигляд інтерфейсу а й зручність використання. Блок схему алгоритму зображено на рисунку 2.10.

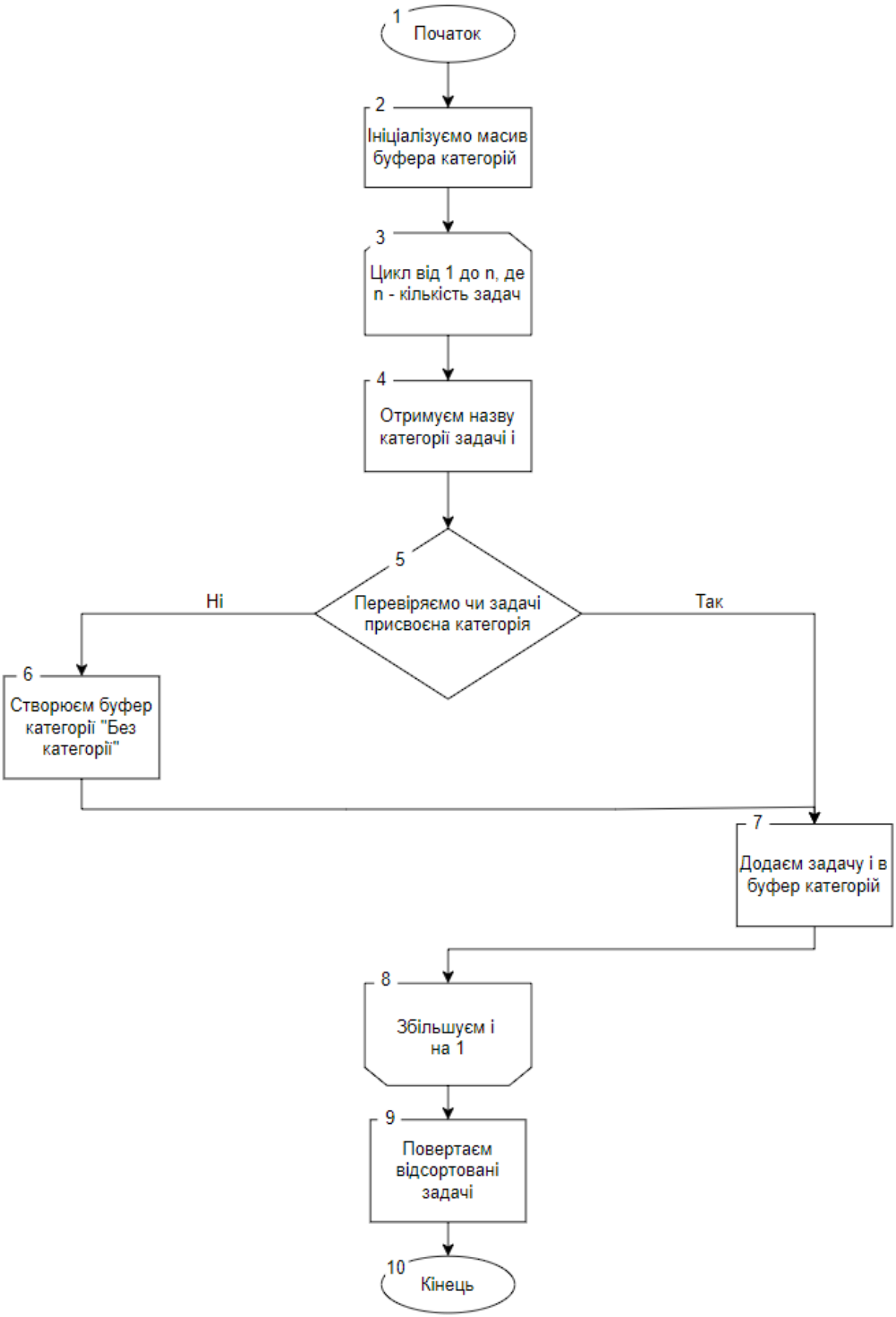


Рисунок 2.10 – Блок-схема алгоритму виводу задач на конкретну дату

Алгоритм складається з 12 кроків:

1. Початок.
2. Вибір дати за допомогою віджета календаря.
3. Отримання списку задач по шуканій даті.
4. Сортування задач з отриманого списку по категоріям.
5. Початок циклу.
6. Перевірка наявності категорії в задачі, якщо така є, то переходим до кроку 8, якщо немає – до кроку 7.
7. Присвоєння задачі категорії «Без категорії»
8. Створення списку задач для цієї категорії.
9. Додавання задачі до списку.
10. Додаєм список задач з категорією до масиву елементів UI для подальшого відображення на екрані.
11. Перехід до наступної задачі.
12. Кінець.

2.5 Висновки

У другому розділі було розроблено метод роботи чат-боту екзаменатора з використанням технологій промпт інженерії. В якості мовної моделі штучного інтелекту було обрано модель gpt-3.5-turbo.

Розроблено метод генерації рекомендацій для створення розкладу самостійної роботи з врахуванням коефіцієнта навантаженості студента.

Проведено архітектурне проектування системи та проектування структурної схеми інтерфейсу.

Розроблено основні алгоритми роботи додатку.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТ ANDROID-ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ СТУДЕНТА

3.1 Варіантний аналіз та обґрунтування вибору способів реалізації програмного засобу

Для успішної реалізації запропонованої моделі в програмному середовищі, потрібно врахувати декілька ключових аспектів. Перш за все, критичним аспектом є вибір правильної платформи для розробки. Ця важлива вибіркова дія визначатиме не лише цільову аудиторію розроблюваного продукту, але й можливості, доступні для реалізації.

Наступним етапом є вибір мови програмування, яка буде використовуватися для розробки. Потрібно враховувати, що для різних платформ може бути необхідно використовувати різні мови програмування. Вибір мови може вплинути на продуктивність розробки та якість програми, тож важливо ретельно розглянути всі аспекти перед прийняттям рішення.

Далі потрібно обрати середовище розробки. Обидві мобільні операційні системи надають різні варіанти середовищ розробки, і правильний вибір може значно полегшити процес розробки і знизити кількість помилок. Тому важливо провести ретельний аналіз доступних варіантів та вибрати той, який найкраще відповідає потребам.

Завершальним кроком є вибір системи управління базою даних. Різні бази даних пропонують різні можливості та характеристики, і правильний вибір може суттєво полегшити роботу з даними та забезпечити ефективне функціонування додатку. Тому важливо провести докладний аналіз доступних варіантів та вибрати той, який на найкращий спосіб відповідає вимогам проекту і забезпечить надійне управління базою даних.

3.1.1 Обгрунтування вибору платформи для розробки

З огляду на галузь використання розроблюваного додатку, важливим фактором розробки є забезпечення доступності для всіх категорій

користувачів, тому було прийнято рішення розробити мобільний додаток. Це дозволить забезпечити мобільність та швидкий доступ до програми, що є ключовими вимогами для нашої цільової аудиторії.

На сучасному ринку існують дві основні мобільні платформи: Android і IOS. Тому для більш об'єктивного вибору було проведено загальне порівняння цих операційних систем, аналізуючи їх переваги та недоліки.

Android - це мобільна платформа, яка базується на операційній системі Linux і надає мобільним телефонам, планшетам та іншим кишеньковим пристроям широкий функціонал. Основною мовою програмування для Android є Java, і для розробки додатків використовуються такі інструменти, як Eclipse та Android Studio. Програми для Android компілюються за допомогою AndroidAPI і перетворюються в байт-код для віртуальної машини Android [15].

Переваги Android-девайсів:

- Різноманітність пристроїв: 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: Android пропонує різноманітні можливості, включаючи підтримку двох сім-карт, що дає можливість користуватися декількома номерами на одному пристрої.
- Розширювана пам'ять: Багато Android-пристроїв підтримують використання карт пам'яті, що дозволяє легко розширити внутрішню пам'ять для зберігання фотографій, відео та інших даних.
- Інтуїтивний інтерфейс: Меню на Android може трохи відрізнятися залежно від версії, але воно має структуру, яка легко розуміється користувачами, що полегшує навігацію.
- Постійні оновлення: Android систематично випускає оновлення, які додають нові функції та поліпшують роботу пристроїв.
- Синхронізація з іншими пристроями: Android пропонує зручну синхронізацію з іншими електронними пристроями, що спрощує обмін даними.

- Магазин додатків: Google Play Маркет надає доступ до великої кількості різноманітних додатків, включаючи безкоштовні та платні.
- Відкритість системи: Android - відкрита операційна система, що дозволяє користувачам та розробникам вносити зміни в систему та розробляти власні додатки.
- Розмаїтість додатків: Завдяки відкритості системи користувачі мають великий вибір додатків, розроблених спільнотою користувачів.
- Універсальні роз'єми: Більшість Android-пристроїв використовують стандартні роз'єми для зарядки та передачі даних, що робить аксесуари більш доступними.

Недоліки Android-девайсів:

- Вразливість до загроз: 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: Деякі додатки на Android можуть споживати багато системних ресурсів, що може вплинути на продуктивність пристрою та тривалість роботи батареї.
- Проблеми з оновленнями: 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, створена компанією Apple, є мобільною операційною системою високої продуктивності, яка включає iPhone, iPad і iPod touch. Вона відома своєю оптимізацією та інтеграцією всіх аспектів пристроїв та сервісів Apple. iOS пропонує користувачам безперервний доступ до великої кількості додатків через App Store та надає можливість використовувати важливі

функції, такі як Siri, Apple Pay і iCloud [16].

Операційна система характеризується стабільністю та високою безпекою, що робить її популярною серед користувачів, які цінують захист своїх даних. Операційна система регулярно отримує оновлення, які принесли нові функції та покращення продуктивності. У iOS також є свій екосистемний підхід, що включає в себе синхронізацію між пристроями, можливість використовувати їх разом, та доступ до ексклюзивних служб та додатків, що робить цю операційну систему привабливою для користувачів, які користуються пристроями Apple.

Переваги операційної системи IOS:

- Інтуїтивно зрозуміле управління: iOS відома своєю простотою та зручністю інтерфейсу, який дозволяє користувачам легко орієнтуватися та використовувати пристрої Apple.
- Синхронізація з пристроями Apple: 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: Apple регулярно випускає оновлення для своїх пристроїв та операційної системи iOS, що приносить покращення у роботу гаджетів та додає нові функції.
- Apple Store: 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: 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: 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: Siri дозволяє користувачам

використовувати голосові команди для виконання різних завдань та отримання інформації.

Недоліки операційної системи IOS:

- Складності в передачі даних: 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: 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: В додатках та іграх у магазині Apple Store іноді можна зустріти високі ціни, що робить їх менш доступними для користувачів.
- Відсутність бюджетних пристроїв: 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: iOS не підтримує Adobe Flash, що може обмежувати доступ до деяких веб-сайтів та мультимедійного вмісту, який використовує Flash-технологію.
- Відсутність розширених можливостей налаштувань: iOS може бути менш гнучкою щодо індивідуалізації та налаштувань порівняно з іншими операційними системами, що може не задовольняти деяких користувачів.

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 входить до списку найпопулярніших і широко використовуваних мов програмування. Вона була створена компанією Sun Microsystens 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[17].

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.

Основний аспект Java полягає в тому, 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 кросплатформності та апаратної переносимості програм, оскільки один і той же код може запускатись на різних системах 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[18].

Ця мова програмування дозволяє створювати не лише мобільні додатки, такі як програми для Android та iOS, але і веб-додатки. Крім того, Kotlin можна

використовувати для розробки як серверної частини програм, яка відпрацьовує на стороні сервера (бекенд), так і клієнтських додатків, які працюють у браузері (фронтенд). Використання Kotlin можливе також для створення десктопних додатків, в галузі Data Science та багатьох інших напрямків.

Отже, Kotlin надає широкі набір інструментів для розробників програм на різних платформах, включаючи Windows, Linux, Mac OS, iOS та Android. Однак, основною сферою застосування Kotlin із серед усіх платформ є розробка під операційну систему Android. Мова Kotlin вже набула такої популярності в цій області, що компанія Google офіційно визнала її однією з ключових мов для розробки застосунків 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 Java і Kotlin

№	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.

3.1.3 Обґрунтування вибору середовища розробки

Перед тим як приступити до написання коду для додатку, необхідно обрати підходяще інтегроване середовище розробки (IDE), 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) [19].

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 представляє собою інтегроване середовище розробки (IDE) для платформи Android, розроблене компанією JetBrains та підтримуване Google. Вона надає розробникам усі необхідні інструменти для ефективної розробки мобільних додатків, включаючи потужний графічний редактор інтерфейсу, емулятор Android для тестування на різних пристроях,

систему збірки Gradle, підтримку мов програмування, відлагодження та інші інструменти для спрощення процесу розробки на платформі Android. Це стандартне середовище для багатьох розробників, що працюють над створенням додатків для мобільних пристроїв на базі Android[20]. На рисунку 3.1 зображено приклад інтерфейсу цього середовища.

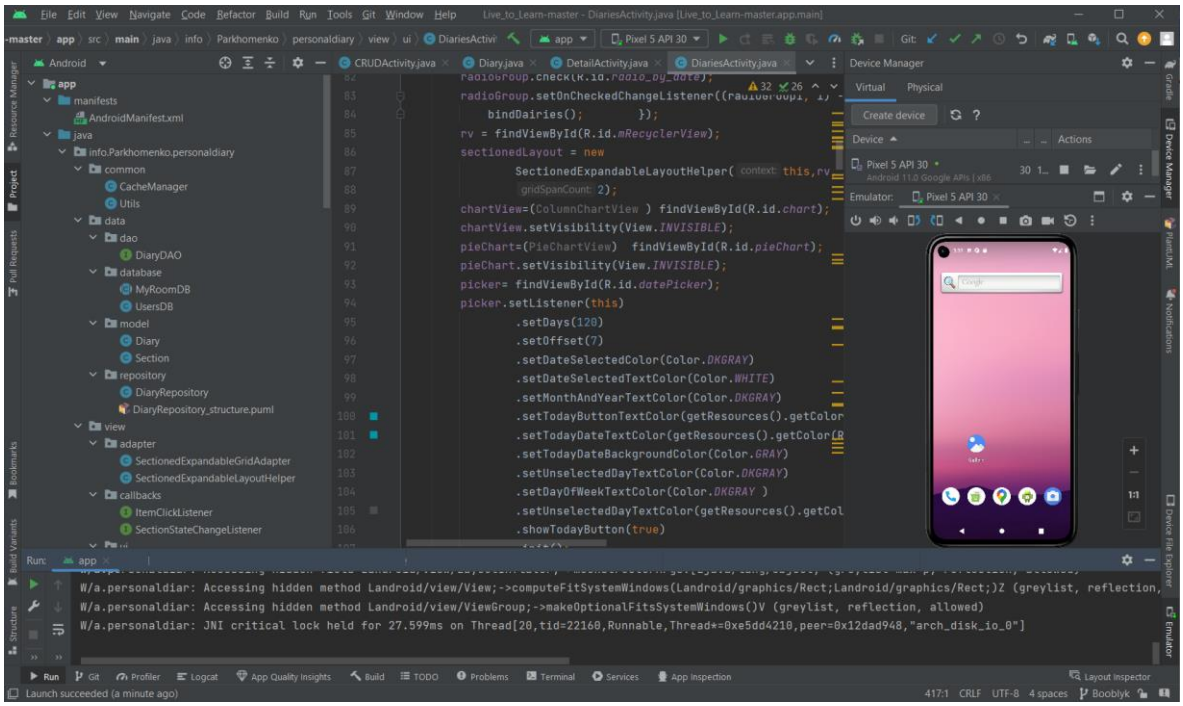


Рисунок 3.1 - Приклад інтерфейсу Android Studio

Eclipse представляє собою інтегроване середовище розробки, використовуване в області комп'ютерного програмування. Це включає базову робочу область та систему розширюваних модулів для розширення можливостей 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 [21]. Приклад інтерфейсу продемонстровано на рисунку 3.2.

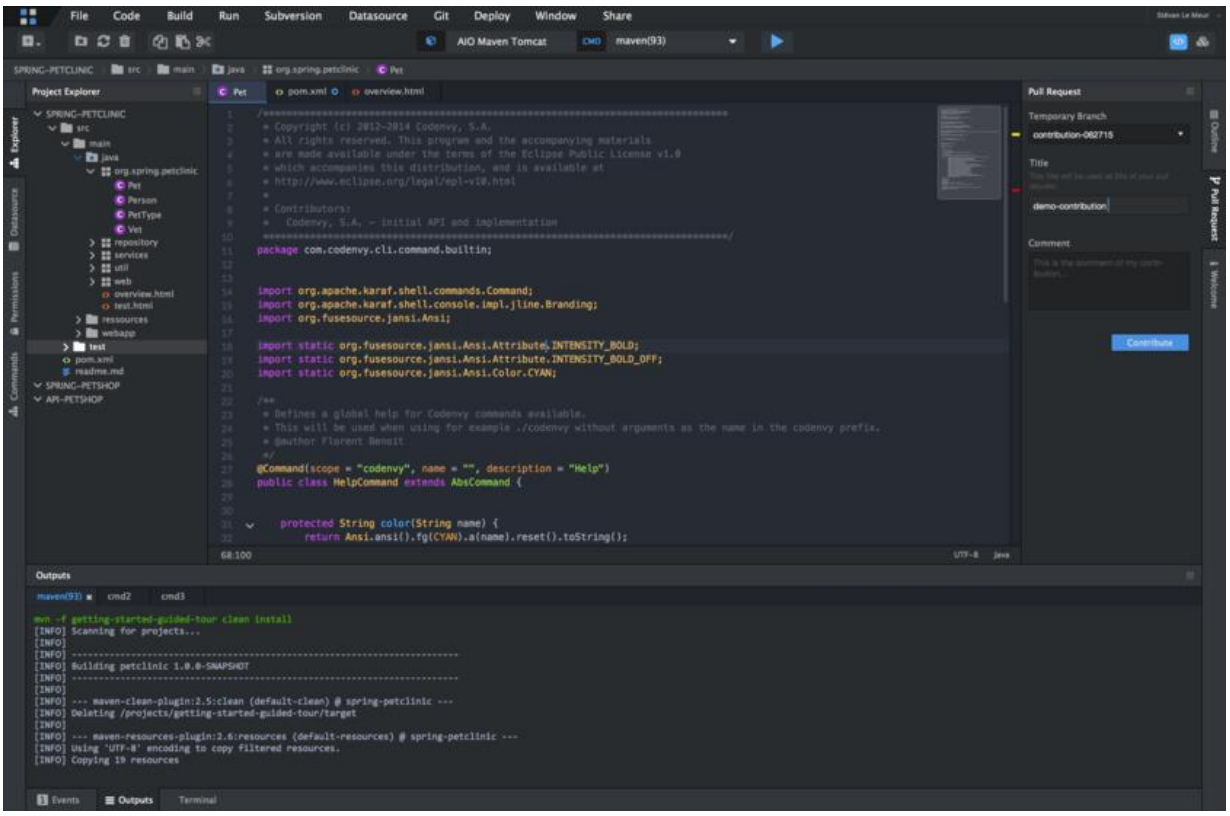


Рисунок 3.2 - Приклад інтерфейсу Eclipse

IntelliJ IDEA – це передове середовище для швидкої розробки на мові Java, об'єднане високотехнологічним комплектом інтегрованих програмних інструментів. У його функціоналі включений інтелектуальний редактор вихідного коду з передовими засобами автоматизації, потужні інструменти рефакторингу коду та вбудована підтримка технологій J2EE. Крім того, IntelliJ IDEA має засоби інтеграції з тестуванням Ant/JUnit і системами керування версіями, унікальний інструмент для оптимізації коду Inspection та інноваційний візуальний конструктор графічних інтерфейсів [22]. Приклад інтерфейсу можна переглянути на рисунку 3.3.

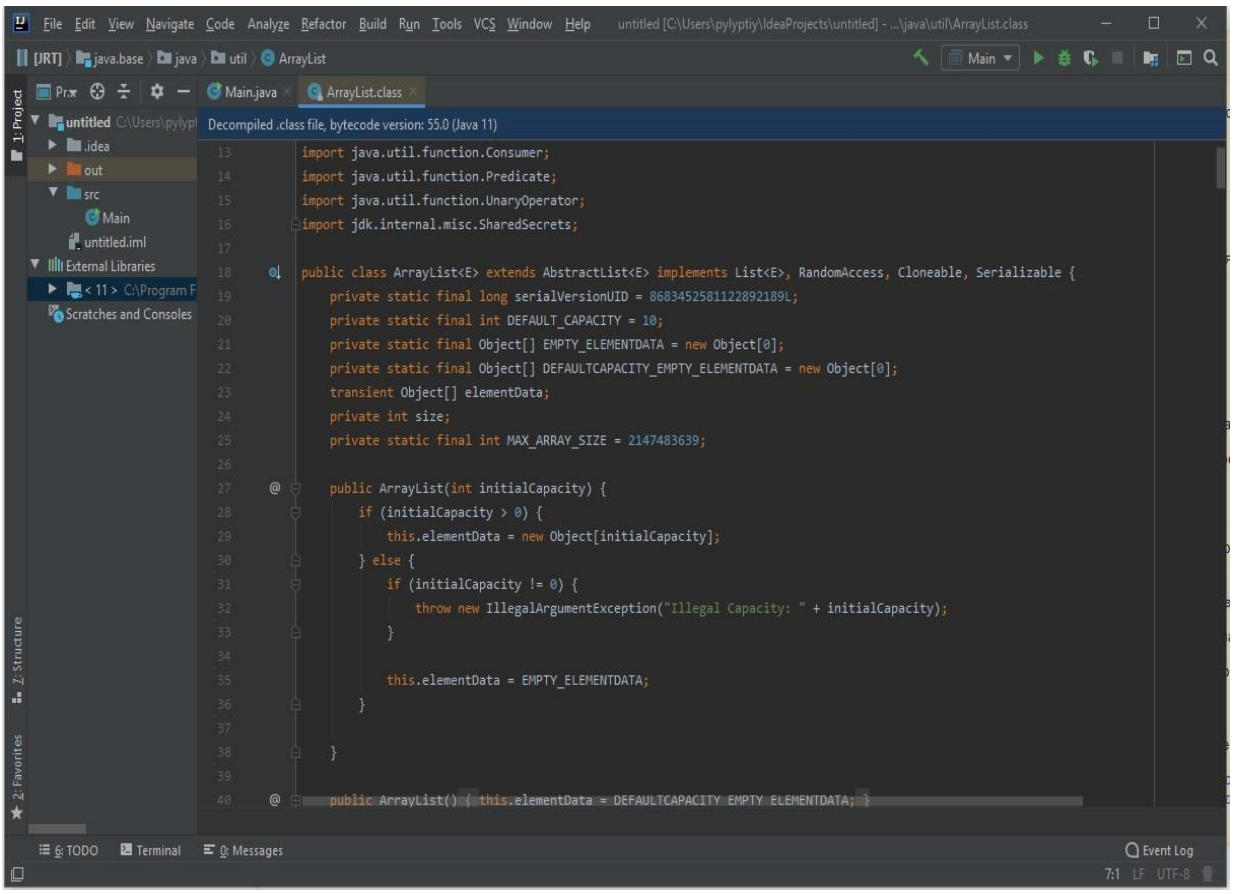


Рисунок 3.3 - Приклад інтерфейсу IntelliJ IDEA

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	1	1	0
Регулярні оновлення Android	1	0	0,5
Швидкодія	1	0,5	0,5
Невимогливість до продуктивності системи	1	0,5	0,5
Магазин розширень	0	0	1
Підтримка мов	1	0,5	1
Підсумковий результат	5	2,5	3,5

На основі аналізу провідних інтегрованих середовищ розробки для створення Android-додатків, який представлений 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 збереження та вивід 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 [23]. Основні завдання СУБД включають забезпечення безпеки даних, ефективного доступу до них, а також забезпечення цілісності та надійності інформації. 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[24].

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[25]. 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 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[26]. 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.

Порівняльний аналіз систем управління базами даних за вибраними критеріями представлено у таблиці 3.3.

Таблиця 3.3 – Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	SQLite
Потужність та функціонал	1	0,5	0
Швидкість читання даних	0	0,5	1
Механізми шифрування	1	1	1
Використання пам'яті	0,5	0,5	1
Вміст всієї бази даних в одному файлі	0	0	1
Підсумковий результат	2,5	2,5	4

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, в результаті аналізу засобів розробки визначено, що оптимальним вибором для реалізації розроблюваного Android-додатку є використання мови програмування Java, середовища розробки Android Studio та системи управління базами даних SQLite.

3.2 Розробка графічного інтерфейсу додатку

Для створення зовнішнього вигляду програми необхідно визначити, які елементи графічного інтерфейсу будуть використовуватися, як вони будуть розташовані на формі та як взаємодіятимуть з користувачем.

Android Studio надає два основних підходи до формування інтерфейсу програми. Перший з них ґрунтується на використанні XML-розмітки, де ви визначаєте, які елементи інтерфейсу будуть використовуватися та як вони будуть розташовані. Другий підхід, пов'язаний з візуальним програмуванням, дозволяє вам перетягувати об'єкти інтерфейсу за допомогою миші та розміщувати їх на формі.

Візуальний метод часто використовується для створення більш простих додатків, але розробники з досвідом можуть вибрати ручне написання коду. У більш складних випадках, як у розробці даного додатку, застосовується комбінований підхід. Це дозволяє використовувати переваги обох методів, поєднуючи зручність візуального програмування з можливістю ручного втручання для досягнення більшої гнучкості та ефективності в розробці інтерфейсу [27].

Під час розробки інтерфейсу додатку, комбінований підхід виявився оптимальним, оскільки дозволив використовувати прості елементи візуального програмування для швидкого розміщення основних об'єктів на формі. Тоді, використовуючи XML-розмітку, були додані додаткові налаштування та функціональність, що забезпечило необхідний рівень гнучкості та контролю. Цей підхід сприяв ефективній реалізації інтерфейсу, а також полегшив подальшу роботу з розширенням та удосконаленням функціоналу додатку.

На рисунках 3.1 та 3.2 зображено інструменти розробки інтерфейсів вікон програмного додатку.

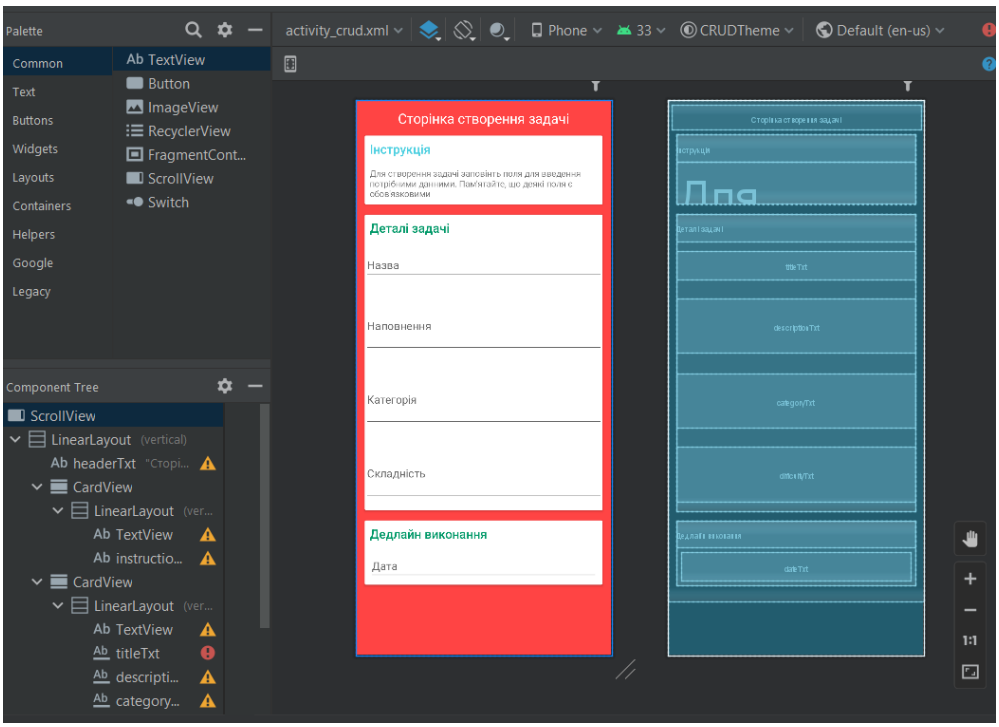


Рисунок 3.1 – Розробка інтерфейсу вікна створення задачі

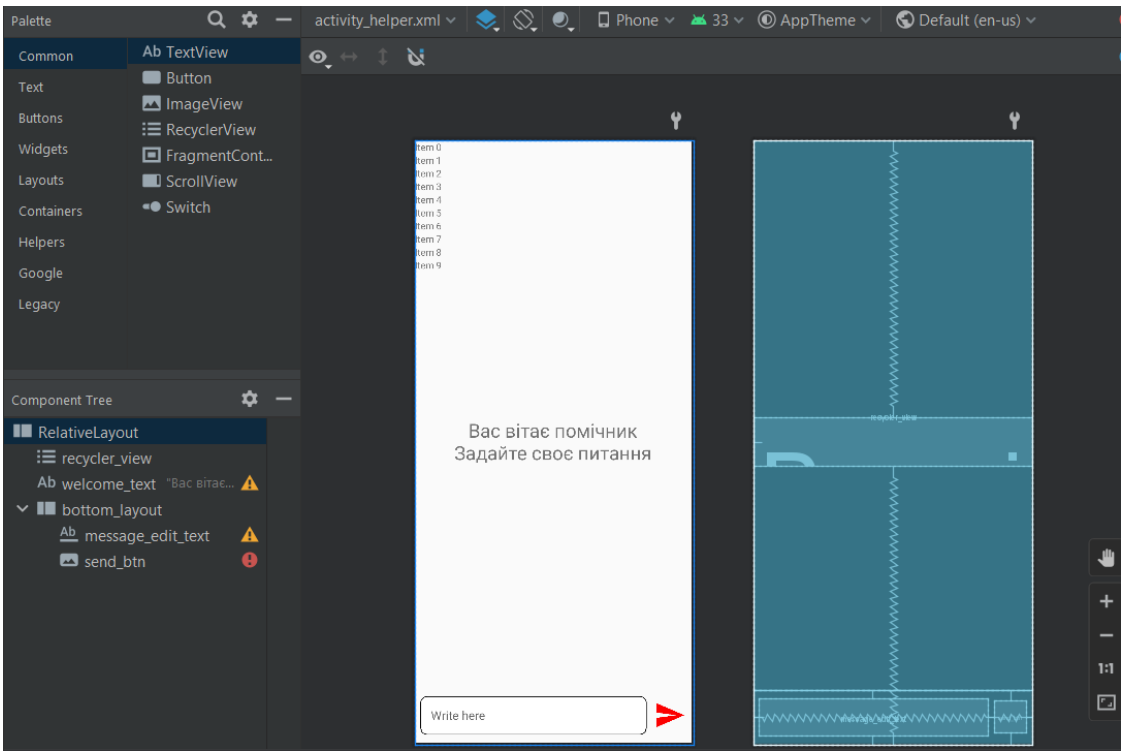


Рисунок 3.2 – Розробка інтерфейсу вікна чат-боту помічника

Інструмент автоматичної збірки Android створює файл під назвою R.java, що включає в себе ідентифікатори всіх ресурсів у каталозі res/ нашої програми. Цей файл містить усі ідентифікатори, визначені у програмі, проте не всі ресурси будуть задіяні в певній діяльності. Знак «+» повідомляє інструментам збірки Android про потребу створення нового ресурсу, яким є R.java [28]. На рисунку 3.3 зображено XML-розмітку, яка відповідає за створення кнопки відправки повідомлення для вікна чат-боту.

```
<ImageButton
    android:layout_width="48dp"
    android:layout_height="48dp"
    android:id="@+id/send_btn"
    android:layout_alignParentEnd="true"
    android:layout_centerInParent="true"
    android:layout_marginStart="10dp"
    android:padding="8dp"
    android:src="@drawable/ic_baseline_send_24"
    android:background="?attr/selectableItemBackgroundBorderless"
    android:layout_marginLeft="10dp"
    android:layout_alignParentRight="true" />
```

Рисунок 3.3 – XML-розмітка кнопки відправлення повідомлення

Зазвичай при розробці Android-додатків використовують поняття активностей для розподілення додатку на окремі вікна діяльностей. В ході розробки додатку було створено 6 головних вікон для реалізації цих активностей, такі як activity_crud, activity_detail, activity_diaries, activity_helper, activity_exam та activity_splash (рис. 3.4). Вибір такого підходу при створенні графічного інтерфейсу значно зменшує використання операційної памяті девайсу під час роботи додатку, таким чином функціональні модулі додатку розподілені між активностями і працюють лише тоді коли відкрито їх вікна.

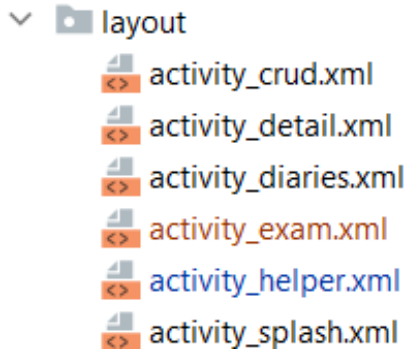


Рисунок 3.4 – Створені вікна активностей

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.

3.3 Програмна реалізація додатку

Робота будь-якого Android-додатку починається з файлу AndroidManifest.xml, який автоматично створюється в корені проекту. За допомогою цього файлу описуються компоненти додатку, вказуються необхідні дозволи використання захищених частин API та взаємодія з іншими програмами, надається дозвіл доступу до ресурсів стороннім програмам, які потрібні для взаємодії з додатком. В створеному додатку файл AndroidManifest.xml має наступний вигляд (рисунок 3.5). Елемент

<application> визначає інформацію про додаток, включаючи назву, анімацію запуску та налаштування пов'язані з резервним копіюванням та підтримкою RTL макетів. Зокрема, цей елемент дозволяє налаштовувати дозволи для резервного копіювання та інші атрибути додатку.

Елемент <activity> є ключовим компонентом додатку, оскільки кожне вікно, яке відкривається користувачем, представляє собою окрему <activity>. У нашому додатку передбачено використання семи основних вікон для взаємодії з користувачем: «SplashActivity», «DiariesActivity», «CRUDActivity», «DetailActivity», «HelperActivity», «ExamActivity», «RegistrationActivity» та «LoginActivity». Проте варто відзначити, що між ними є певні відмінності у спадкуванні. Конкретно, «SplashActivity», «DiariesActivity» «LoginActivity» та «RegistrationActivity» є незалежними активностями, тоді як «CRUDActivity», «DetailActivity», «HelperActivity» та «ExamActivity» успадковані від батьківської активності «DiariesActivity».

Розглянемо основні завдання, що виконуються цими активностями. «SplashActivity» призначена для запуску анімації старту додатку, використовуючи функцію «showSplashAnimation» (рисунок 3.5).

```
private void showSplashAnimation() {
    Animation bottomToTop = AnimationUtils.loadAnimation(this, R.anim.top_to_bottom);
    mLogo.startAnimation(bottomToTop);

    Animation fadeIn = AnimationUtils.loadAnimation(this, R.anim.fade_in);
    mainTitle.startAnimation(fadeIn);
    subTitle.startAnimation(fadeIn);

    this.preloadDiaries();
}
```

Рисунок 3.5 – Лістинг функції showSplashAnimation

Завдяки функції «preloadDiaries» відбувається завантаження задач із бази даних, які будуть використовуватися в подальшій роботі додатку (рисунок 3.6).

```
private void preloadDiaries() {
    DiaryViewModel diaryViewModel = ViewModelProviders.of(this).get(DiaryViewModel.class);
    diaryViewModel.getDiariesLiveData().observe(this, diaries -> {
        if (diaries != null && diaries.size() > 0) {
            CacheManager.ALL_DIARIES_MEMORY_CACHE = diaries;
            CacheManager.DIARIES_DIRTY = false;
        }
        proceed();
    });
}
```

Рисунок 3.6 – Лістинг функції preloadDiaries

Після завершення анімації старту додатку, головне меню або, інакше кажучи, «DiariesActivity», стає доступним для користувача. Щоб розпочати роботу, необхідно спочатку ініціалізувати використані віджети, використовуючи функцію «initializeViews», яка ініціалізує різні елементи інтерфейсу користувача (Views) для взаємодії з користувачем у додатку.

Основні елементи інтерфейсу, які ініціалізуються в цьому методі:

– RadioGroup (radioGroup): Група радіокнопок. Встановлюється невидимою (View.INVISIBLE), за замовчуванням вибрано один із радіокнопок (за допомогою radioGroup.check(R.id.radio_by_date)), і встановлюється слухач подій на зміну вибору радіокнопки.
– TextView (textView): Текстове поле.
– RecyclerView (rv): Відображення списку елементів.
– SectionedExpandableLayoutHelper (sectionedLayout): Допоміжний клас для роботи з розгортанням і згортанням секцій у списку.
– ColumnChartView (chartView): Відображення графіка стовпчиків.
– PieChartView (pieChart): Відображення кругової діаграми.
– DatePicker (picker): Віджет вибору дати.
– BottomNavigationView (bottomNavigationView): Нижнє меню навігації.

Ця функція спрямована на запуск календарних віджетів, які дозволяють вибирати потрібну дату, активувати випадаючі списки завдань у відповідності до графіку виконання і користувальницької навігаційної панелі. Останню функціональність реалізовано через метод "listenToBottomNavigationClicks". Цей метод служить для налаштування слухача подій на натискання пунктів

нижнього меню навігації (Bottom Navigation). У зазначеному коді використовується лямбда-вираз (лямбда-функція) для реалізації інтерфейсу OnNavigationItemSelectedListener.

Основні дії, які виконуються при натисканні на різні пункти нижнього меню:

- "Home" (R.id.action_home): Встановлюється значення defaultPage в true і викликається метод bindDairiesCat().
- "Time Manager" (R.id.time_manager): Встановлюється значення defaultPage в false і викликається метод bindTimeManager().
- "All Dates" (R.id.action_all_dates): Встановлюється значення defaultPage в false і викликається метод bindDairies().
- "Helper" (R.id.action_helper): Встановлюється значення defaultPage в false і викликається метод Utils.openActivity(this, HelperActivity.class), що відкриває нову активність HelperActivity.
- "Exit" (R.id.action_exit): Завершує поточну активність (закриває її).

Отже, цей метод дозволяє визначити реакцію на натискання певних пунктів нижнього меню навігації та викликати відповідні методи чи активності в залежності від вибору користувача.

Далі ми детально розглянемо функції, пов'язані з діяльністю "CRUDActivity", яка відповідає за створення, перегляд, редагування та видалення задач. Введення даних під час створення задачі обробляється функцією "insertData".

Ця функція, призначена для вставки даних щодо нового запису завдання в базу даних. Основні етапи цього методу:

- Оголошення змінних: title, description, date, category - це змінні для збереження даних, які будуть введені користувачем. difficulty - це змінна для рівня складності завдання.
- Валідація введених даних: Здійснюється перевірка введених даних за допомогою методу Utils.validate(). Цей метод перевіряє, чи всі необхідні поля заповнені коректно. Якщо валідація пройшла успішно (true),

продовжується виконання коду; в іншому випадку виводиться повідомлення про помилку.

- Отримання значень полів введення: Зчитування введених користувачем даних з відповідних полів введення.
- Перевірка дати: Перевіряється, чи була заповнена дата виконання задачі (dateTxt.getDate() != null). Якщо дата вказана, вона форматується за допомогою методу dateTxt.getFormat().format(dateTxt.getDate()). Якщо дата не вказана, виводиться помилка та викликається dateTxt.setError("Invalid Date"), після чого фокус переміщується на це поле.
- Виклик методу вставки даних: Якщо всі перевірки пройдені успішно, викликається метод insertDiary() з передачею в нього отриманих даних для здійснення вставки цих даних відповідним чином в базу даних.

Після введення коректних значень для створення задачі її потрібно зберегти, за збереження даних відповідає функція «insertDiary». Важливо, що функція забезпечує безпечне зберігання без ризику пошкодження цілісності бази даних. Після введення та збереження даних з’являється можливість редагування та видалення створеної задачі за допомогою функцій «updateData» та «deleteData».

Так як одним із основних завдань додатку є створення архіву, з можливістю сортування задач, розглянемо спосіб, яким вікно "Архів" реалізує цю можливість завдяки функції "bindDairies". Оскільки сортування задач в додатку включає в себе не лише групування за категоріями, але й за датою виконання, ми впроваджуємо функціональність перемикача "radioGroup" для переключення між різними методами сортування.

Для сортування задач за датою, ми створюємо масиви списків задач, відсортованих за датою, та подаємо їх до віджету випадаючого списку для відображення на екрані. У випадку, коли користувач переходить до архіву задач, застосовується метод сортування за категоріями.

Ми створюємо колекцію множин задач, впорядкованих за різними категоріями. Після цього ми перевіряємо, чи задача була призначена до певної категорії, та створюємо проміжну категорію "Без категорії" для задач, які не мають призначеної категорії. Після завершення цього процесу ми передаємо всі списки задач у віджет випадаючого списку для їх відображення на екрані.

Далі розглянемо функцію "bindTimeManager," яка відповідає за створення графічного звіту із статистикою. Мета цієї функції полягає в створенні графічного звіту, який відображає кількість задач, які були поставлені на різні дати. Це дозволяє користувачу відстежувати навантаженість в різні дні та розподіляти завдання рівномірно у часі.

Розглянемо основні кроки цієї функції:

– Створення об'єкту SectionedExpandableLayoutHelper, який служить для роботи з розділеними секціями у RecyclerView (rv). Цей об'єкт ініціалізується з посиланням на поточний контекст (this), самим RecyclerView і зовнішнім обробником подій (вказаним також як this).
– Отримання списку всіх завдань з пам'яті за допомогою CacheManager.ALL_DIARIES_MEMORY_CACHE.
– Виклик Utils.getAllDiariesGroupedByDates(diaries), щоб отримати список завдань, згрупованих за датами.
– Реверсування порядку списку згрупованих завдань (Collections.reverse(diariesLists)), щоб вони відображалися у правильному хронологічному порядку.

– Створення списку стовпчиків (об'єктів Column) для кожної секції завдань. Кожен стовпчик має значення, що відображається висотою стовпчика, яке дорівнює кількості завдань у цій секції.
– Створення значень для осі X (axisValues), де кожен позначений пункт відповідає даті виконання завдань.
– Створення об'єкту ColumnChartData і встановлення його властивостей, таких як осі X та Y, а також набір стовпчиків.

– Налаштування властивостей chartView, таких як видимість, активація можливості вибору значень та тип зуму.
– Встановлення готових даних для графіку за допомогою chartView.setColumnChartData(columnChartData).
– Встановлення chartView видимим для користувача з використанням chartView.setVisibility(View.VISIBLE).

Важливим модулем додатку є чат-бот помічника, який використовує мовну модель штучного інтелекту gpt-3.5-turbo для взаємодії з користувачем. За роботу цього модуля відповідає клас «HelperActivity», який наслідує клас "AppCompatActivity" і виконує роль активності у Android-додатку. Він відповідає за функціонал допоміжного вікна для виклику чат-боту помічника. У цьому класі є різні компоненти, такі як RecyclerView, TextView, EditText та ImageButton, які використовуються для відображення повідомлень та взаємодії з користувачем. Клас також містить список повідомлень (messageList) та адаптер (messageAdapter), які використовуються для керування вмістом (RecyclerView). Крім того, в класі є методи для додавання повідомлень до чату (addToChat), додавання відповіді від бота (addResponse) та виклику API для отримання відповіді на запитання користувача (callAPI). У цьому класі використовується бібліотека OkHttpClient для взаємодії з API OpenAI. Бібліотека OkHttpClient [29] є однією з популярних бібліотек для роботи з мережевими запитами в Android-додатках. Вона надає зручний і простий спосіб виконання HTTP-запитів, включаючи GET, POST, PUT і DELETE запити. OkHttpClient забезпечує підтримку таких функціональностей, як встановлення з'єднання, обробка таймаутів, керування перенаправленнями, автентифікація, кешування, підтримка проксі і багато іншого. Вона також дозволяє використовувати асинхронний підхід для виконання запитів, що робить її ефективною і потужною бібліотекою для мережевої комунікації у Android додатках.

Функція callAPI виконує асинхронний HTTP-запит до віддаленого API задопомогою бібліотеки OkHttpClient. Основна мета цієї функції – отримання

відповіді на певне питання шляхом використання сервісу OpenAI. У функції спочатку створюється JSON-об'єкт jsonBody, який містить дані запиту до API, цей об`єкт власне і є повідомленням яким ми обмінюємось із сервером. Далі створюється об'єкт Request з вказаною URL-адресою API, заголовком авторизації та методом POST. В тілі запиту передається об'єкт RequestBody.

Далі йде виклик запиту, в якому визначаються 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, яка додає повідомлення до списку повідомлень (messageList) і оновлює адаптер (messageAdapter). У разі невдачі відповіді, виконується відповідний обробник помилки. Отже, функція callAPI виконує запит до API з питанням і отримує відповідь, яку потім додає до списку повідомлень для подальшого відображення у додатку.

3.4 Висновки

У третьому розділі проведено варіативний аналіз та обгрунтування методів реалізації програмного продукту. В ході цього аналізу було визначено використання платформи Android, мови програмування Java, середовища розробки Android Studio та системи управління базами даних SQLite.

Було проведено розробку графічного інтерфейсу додатку та описано програмну реалізацію основних функцій та методів розроблюваного додатку.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Аналіз методів та засобів тестування

Тестування програмного забезпечення (software testing) - це процес, спрямований на аналіз та експертизу програмного забезпечення з метою виявлення можливих дефектів [30]. Зазвичай цей етап в розробці програмного продукту є найтривалішим і вимагає найбільше ресурсів. Тестування може бути класифіковане на два основні напрями: статичне та динамічне.

Статичне тестування - це метод, за яким тестування не передбачає виконання коду програми, а включає, наприклад, перевірку якості документації або вимог до програмного продукту. Воно часто використовується для попередньої оцінки якості програмного продукту на ранніх стадіях розробки.

Динамічне тестування - це процес, який включає в себе тестування програми в умовах безпосереднього виконання коду додатку. Воно фактично випробовує програму на відповідність заявленому функціоналу. Під час динамічного тестування можуть враховуватися різні аспекти, такі як використання ресурсів, швидкість та продуктивність.

Залежно від рівня знання внутрішньої будови системи, яка тестується, виділяють такі види динамічного тестування:

- Тестування "чорної скриньки" - це метод, який включає в себе відсутність знань про внутрішню структуру програмного додатку, його алгоритми та принципи виконання функцій. Під час цього виду тестування перевіряються зовнішні інтерфейси додатку.
- Тестування "білої скриньки" - це метод, що передбачає повне розуміння принципів роботи програмного додатку, його алгоритмів та принципів виконання функцій.
- Тестування "сірої скриньки" - це поєднання технік чорної та білої скриньок, що передбачає наявність часткового розуміння внутрішньої структури додатку та його функціонування.

Під час тестування "чорної скриньки" увага зосереджується на взаємодії з користувачем, перевірці правильності введення та виведення даних, а також на забезпеченні правильної реакції програми на різні сценарії використання. Зокрема, ми перевіряємо, як додаток взаємодіє з обладнанням пристрою, як реагує на введення даних від користувача та як відображає результати.

З урахуванням особливостей кожного з розглянутих методів тестування програмного забезпечення, було прийнято рішення використовувати методику "чорної скриньки" для тестування Android-додатку. Це дозволить виявити помилки в логіці роботи додатку та можливі несправності в роботі його основних алгоритмів, що не завжди можна досягти застосуванням інших методів динамічного тестування.

4.2 Тестування розробленого програмного продукту

Тестування Android-додатку методом "чорної скриньки" включає в себе перевірку коректності функціонування додатку під час виконання основних алгоритмів його використання та порівняння фактичних результатів з очікуваними. Ці алгоритми використання програмного додатку називаються тест-кейсами. Для тестування розробленого Android-додатку ми розробили наступні тест-кейси:

Тест-кейс №1 - Створення нової задачі:

1. Відкрити додаток.
2. Натиснути кнопку «Додати задачу».
3. Заповнити деталі задачі в відповідних полях.
4. Натиснути кнопку «Зберегти».
5. Перевірити результат у вікні «Архів».

Очікуваним результатом цього тест-кейсу є створення нової задачі, після чого вона повинна з'явитися у вікні «Архів». Для додавання задачі заповнюємо поля «Назва», «Наповнення», «Категорія», «Складність» та «Дедлайн виконання». Наприклад, введемо задачу «Лабораторна робота №3» із

категорією «Проектування інструментального програмного забезпечення». Результат виконання тест-кейсу показаний на рисунку 4.1.

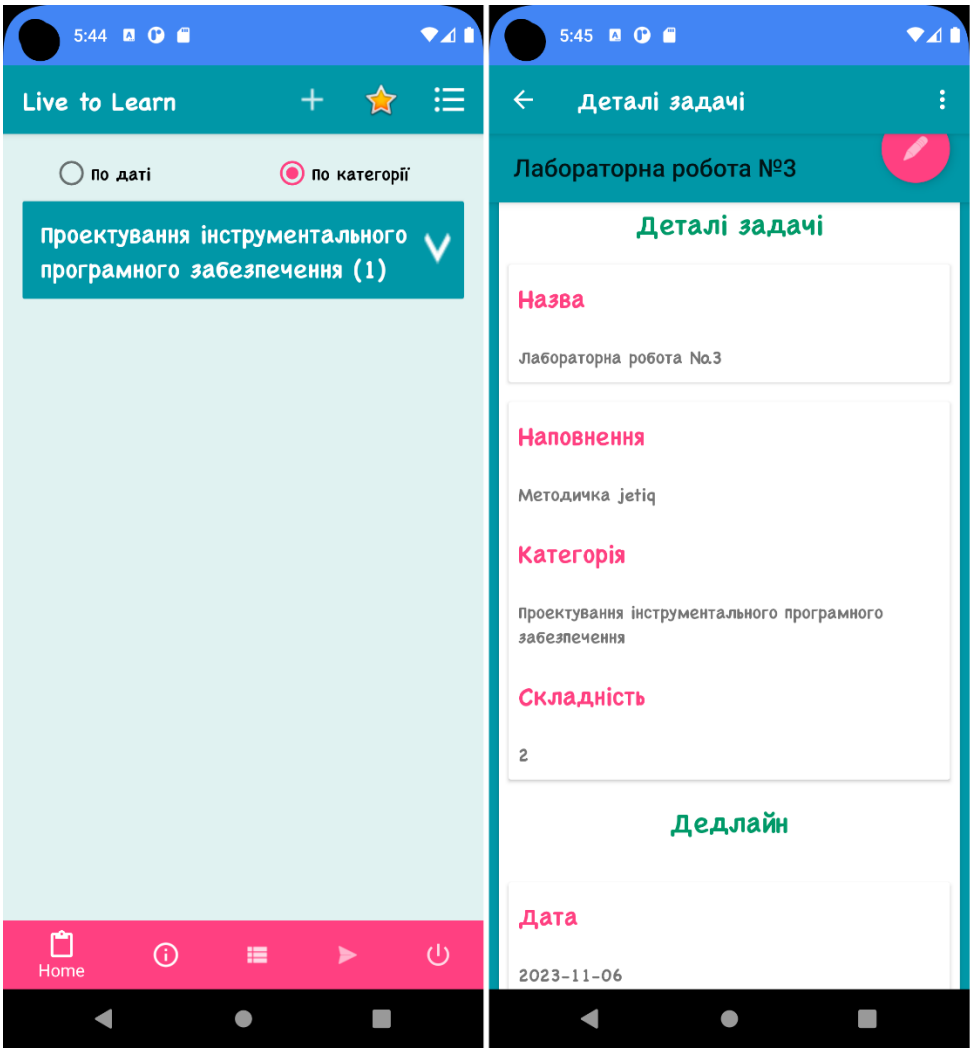


Рисунок 4.1 – Результат виконання тест-кейсу №1

Фактичний результат відповідає очікуваному, тому тест-кейс №1 успішно пройдено.

Тест-кейс №2 – Зміна методу сортування задач:

1. Відкрити додаток.
2. Перейти до вікна "Архів".
3. Натиснути на неактивний чекбокс вибору метода сортування.
4. Перевірити результат зміни вікна "Архів".

У цьому випадку очікуваним результатом є відкрите вікно "Архів", яке показує архів задач відсортованих по категорії. Перед переглядом вікна було

створено кілька задач за алгоритмом першого тест-кейсу. Результат виконання тест-кейсу представлено на рисунку 4.2.

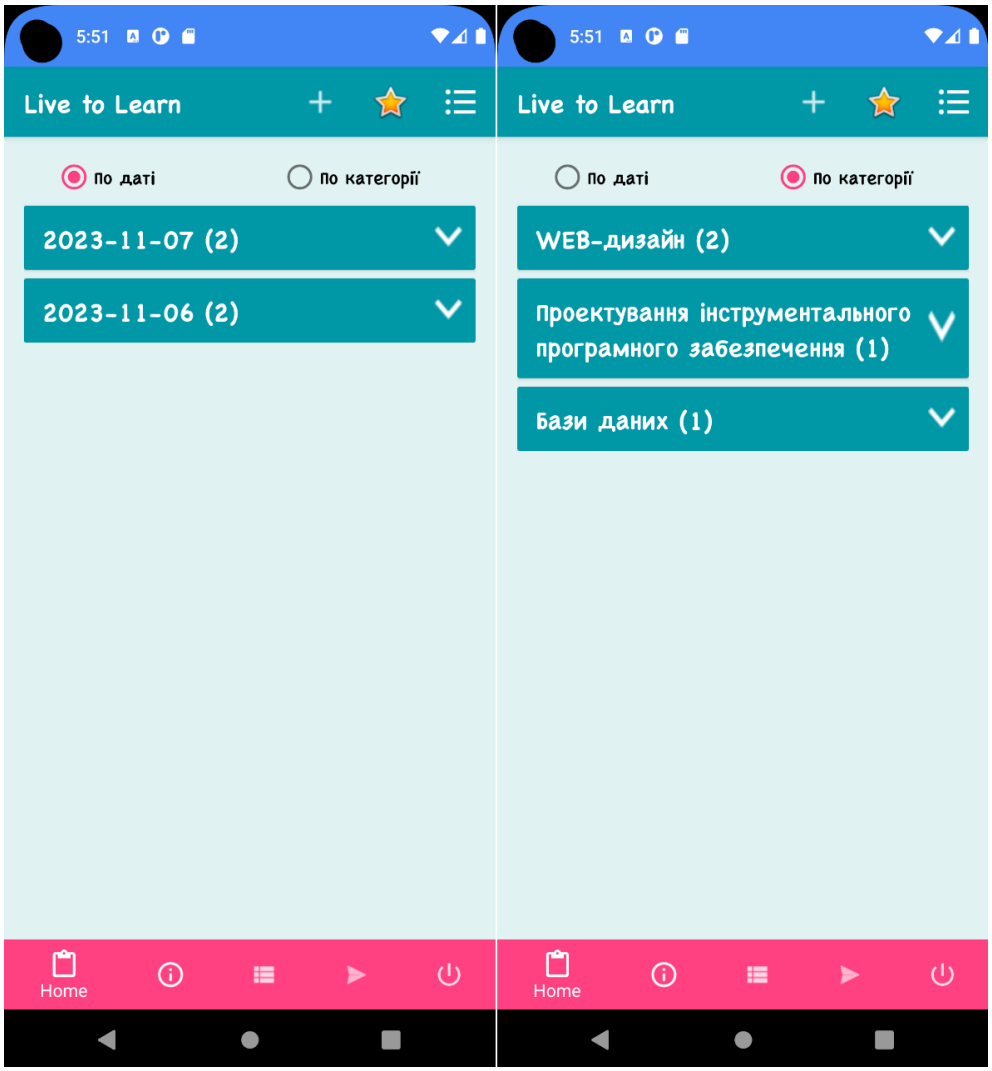


Рисунок 4.2 – Результат виконання тест-кейсу №2

Змінився метод сортування задач – тепер вони відсортовані за категоріями. Тому фактичний результат відповідає очікуваному. Отже, тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Перегляд вікна статистики навантаження:

1. Відкрити додаток
2. Перейти до вікна «Статистика»
3. Перевірити результат

У даному тест-кейсі проводиться перевірка генерації двох звітів: одного

за категоріями та іншого за датою виконання. Для цієї перевірки було створено різну кількість задач на три дні, відповідно до алгоритму, описаного в першому тест-кейсі. Очікуваним результатом є виведення двох графічних звітів, де задачі відсортовані за категоріями та за датою виконання. Перший звіт повинен мати вигляд колонок (column chart) і зберігати послідовність днів, другий звіт створено за принципом пирога (pie chart). Результат виконання цього тест-кейсу можна переглянути на рисунку 4.3.

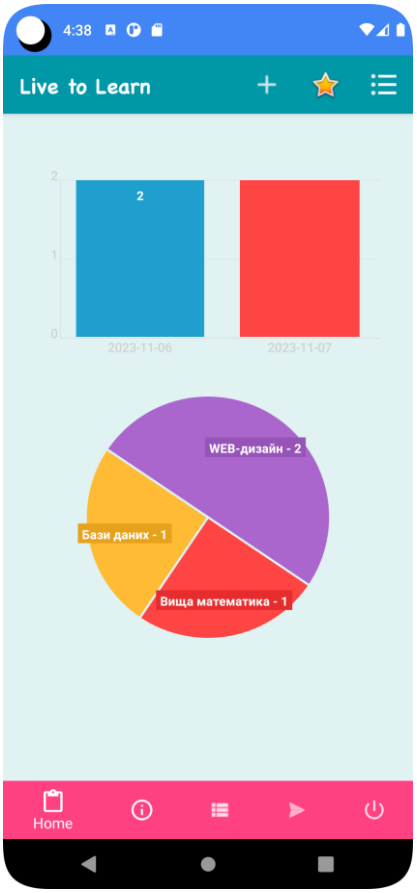


Рисунок 4.3 – Результат виконання тест-кейсу №3

В результаті виконання цього тест-кейсу було побудовано 2 графічних звіта з статистикою навантаження, отже тест-кейс №3 успішно виконано.

Тест-кейс №4 - Процедура видалення завдання, створеного раніше:

1. Відкрити додаток.
2. Перейти до вікна "Архів".

3. Вибрати раніше створене завдання з випадаючого списку.
4. Клацнути на кнопку "Змінити/Видалити", що призведе до відкриття вікна "Редагування завдання".
5. Натиснути кнопку "Видалити".
6. Повернутися до вікна "Архів".
7. Перевірити результат виконання.

Очікуваним результатом в даному випадку є відкрите вікно "Архів", яке не міститиме інформації про раніше створене завдання. Перед видаленням завдання це завдання було створено за алгоритмом першого тест-кейсу. Результат виконання тест-кейсу зображений на рисунку 4.4.

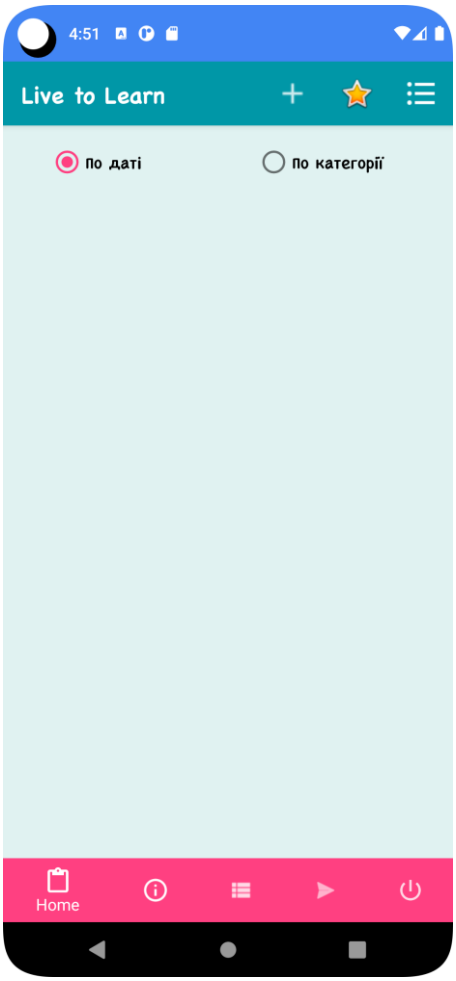


Рисунок 4.4 – Результат виконання тест-кейсу №4

Результат виконання цього тест-кейсу повністю відповідає очікуванням, що свідчить про успішне проходження тест-кейсу №4.

Тест-кейс №5 – Перевірка роботи чат-боту екзаменатора:

1. Відкрити додаток.
2. Перейти до вікна "Екзаменатор".
3. Написати тему опитування.
4. Отримати питання від чат-боту.
5. Дати правильну відповідь.
6. Отримати підтвердження правильності відповіді та наступне питання.
7. Дати неправильну відповідь.
8. Перевірити результат.

Очікуваним результатом для цього тест-кейсу є здатність моделі діяти як екзаменатор, встановлюючи діалог з користувачем, оцінюючи правильність відповідей користувача, і, у разі правильної відповіді, переходячи до наступного питання. У випадку неправильної відповіді, модель має надати пояснення, чому відповідь була неправильною, і надати приклад правильної відповіді перед переходом до наступного питання. Цей тест-кейс перевіряє, чи відповідає модель заданим критеріям для ролі екзаменатора. Результат виконання тест-кейсу зображено на рисунку 4.5.

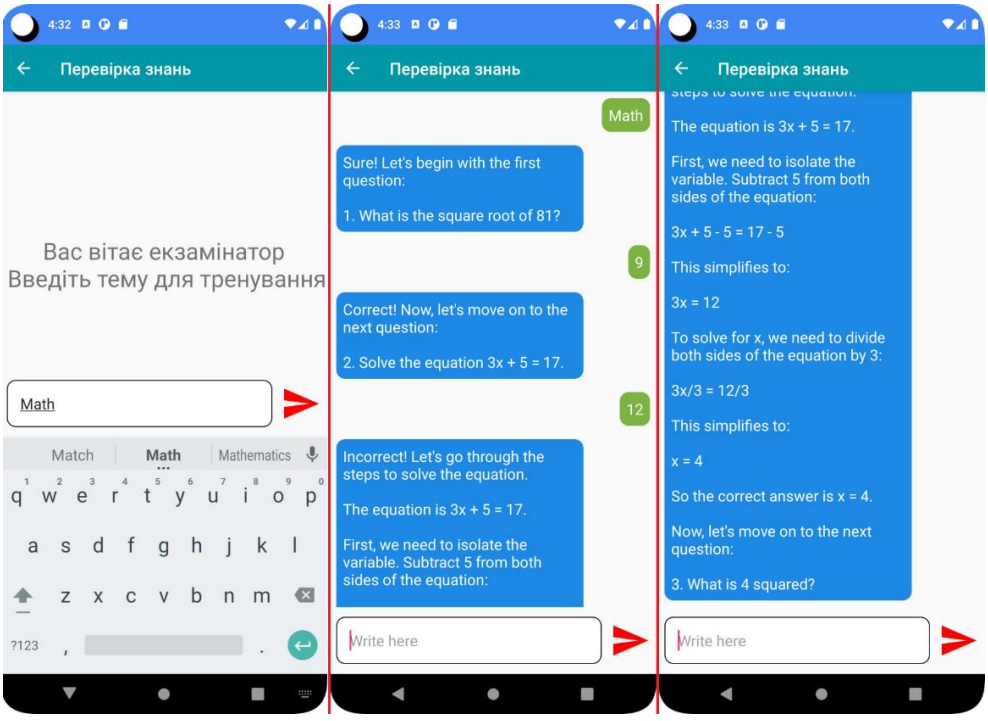


Рисунок 4.5 – Результат виконання тест-кейсу №5

В результаті виконання тест-кейсу чат-бот відпрацював як і було задумано, питання ставились відповідно до вказаної теми, правильні відповіді було прийнтято, неправильні відповіді було пояснено.

Під час тестування програмного додатку було отримано повну відповідність фактичних результатів очікуваним. Не виявлено жодних помилок або дефектів у роботі програми, і додаток працював відповідно до передбаченого функціоналу. Цей процес підтвердив повну працездатність програмного додатку і його відповідність усім вимогам, які були встановлені в технічному завданні.

4.3 Розробка інструкції користувача

У цьому розділі надано інструкцію для користувача додатку за допомогою ілюстрацій (знімків екрану). Після запуску додатку користувача вітає головне меню, де представлений відсортований за категоріями список задач, що повинні бути виконані у поточний день, рекомендації щодо навантаження, які було вираховано відповідно до коефіцєнта навантаження, а також віджет для вибору необхідної дати (див. рисунок 4.6).

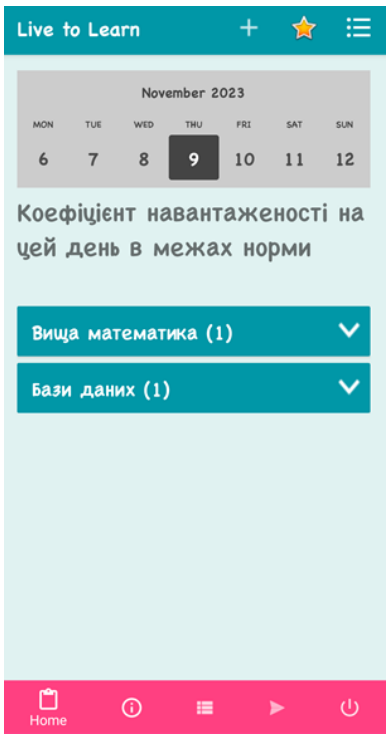


Рисунок 4.6 – Головне меню додатку

Щоб переглянути всі раніше визначені завдання, слід перейти до розділу «Архів» за допомогою іконки «  » (рисунок 4.7).

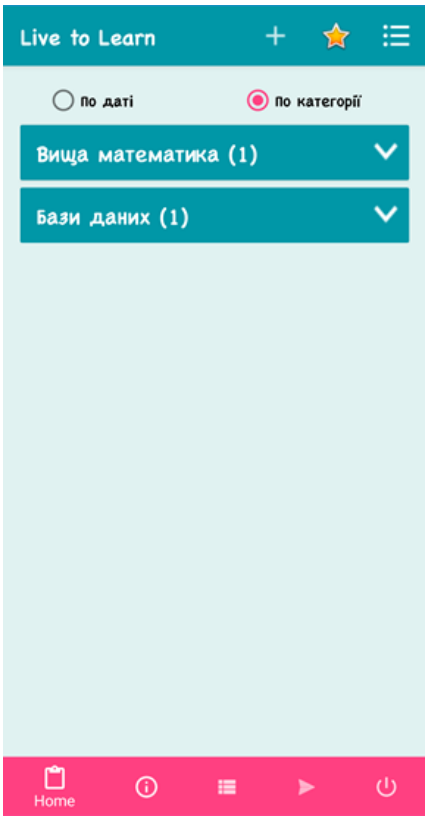


Рисунок 4.7 – Вікно «Архів»

В цьому вікні можна змінити режим сортування з задач обравши потрібний натиснувши на відповідний чек-бокс. Клацнувши на відповідне завдання, відкриється сторінка «Деталі задачі», де користувач матиме змогу переглянути назву, вміст, категорію, складність та дату виконання завдання (див. рисунок 4.8).

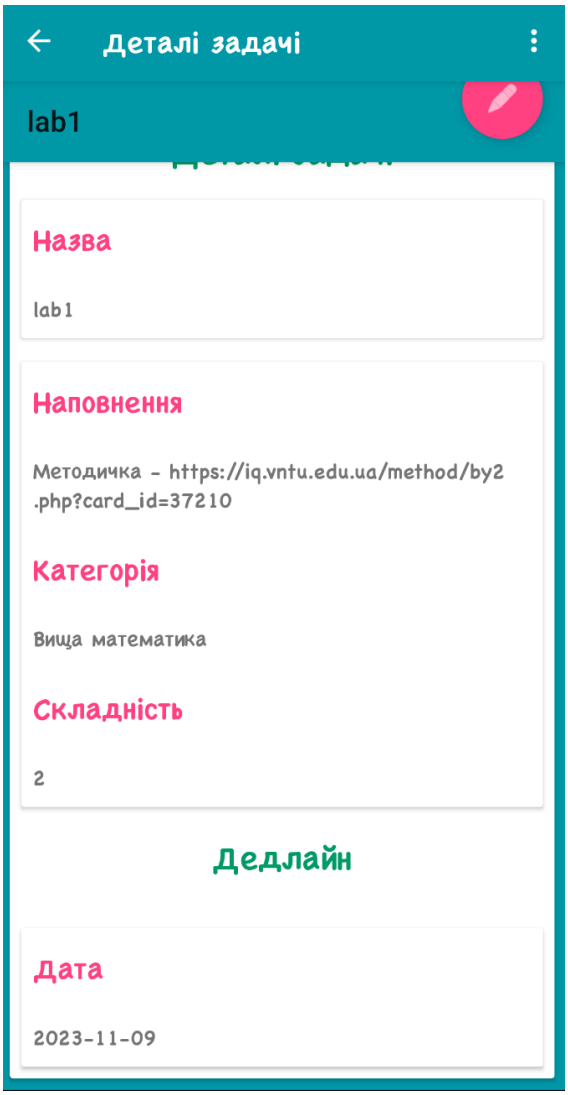


Рисунок 4.8 – Сторінка перегляду деталей задачі

На сторінці «Деталі задачі» користувач має можливість редагувати раніше створене завдання, за допомогою піктограми « ». Крім редагування існуючої задачі, функціонал цієї сторінки дозволяє повністю видалити непотрібне завдання за допомогою іконки « ». Приклад виклику сторінки редагування задачі показано на рисунку 4.10.

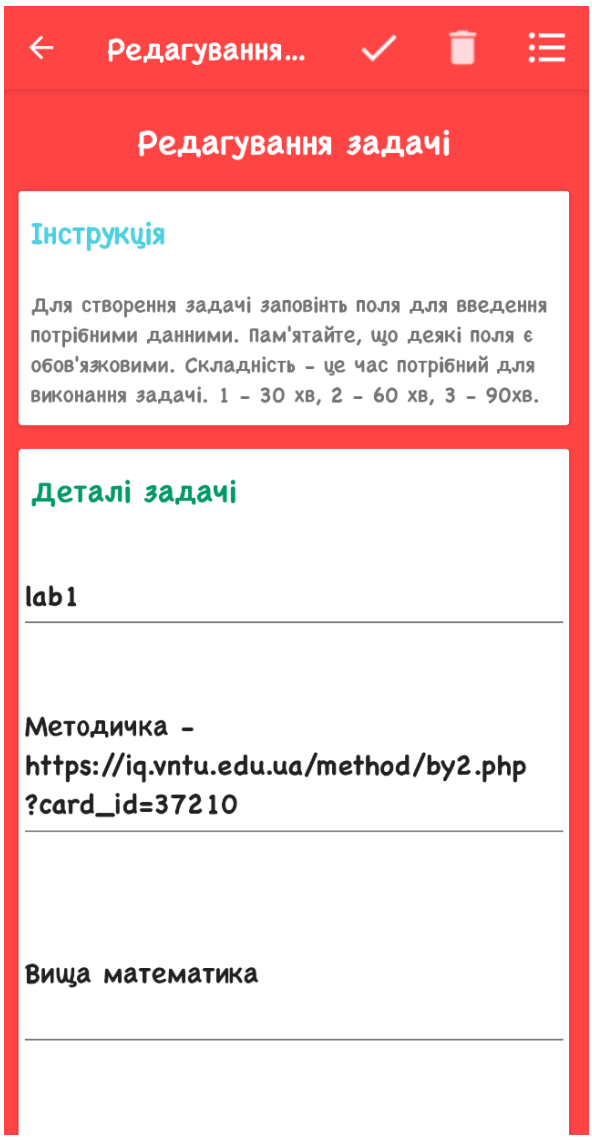


Рисунок 4.9 – Сторінка редагування задачі

Завдяки іконкам « » та « » можна перейти до вікон чат-боту помічника та чат-боту екзаменатора. Також користувач має змогу переглянути статистику своїх задач у вікні «Статистика», яке можна викликати натиснуваши на піктограму « ». Приклад виклику даного вікна наведено на рисунку 4.10.

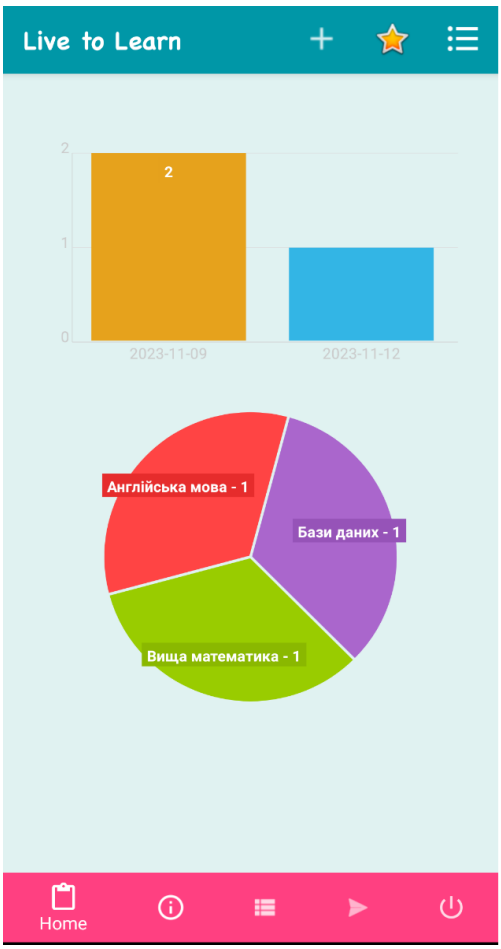


Рисунок 4.10 – Вікно «Статистика»

Після завершення роботи з додатком необхідно вийти, використовуючи піктограму « » .

Таким чином, створено інструкцію для користувача, в якій описані кроки для початку використання Android-додатку та детально розглянуті різні можливості взаємодії користувача з його функціоналом.

4.4 Висновки

В четвертому розділі було проведено аналіз методів та засобів тестування, в результаті чого було обрано метод чорної скриньки.

Проведено тестування, що підтвердило повну працездатність програмного додатку і його відповідність поставленим вимогам у технічному завданні.

Розроблено інструкцію користувача по використанню програмного засобу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Вибір методики оцінювання ефективності науково-дослідної роботи

Освіта є ключовим аспектом розвитку кожної людини та суспільства в цілому. Здобуття знань, навичок та умінь є невід'ємною частиною життя кожного індивіда [1]. Сучасні технології та зростаючий обсяг інформації ставлять перед освітою нові виклики та завдання. Самостійна робота студентів стає дедалі важливішою, інформаційний простір.

Розробка Android-додатку для організації самостійної роботи студентів стає ще більш актуальною завдяки використанню інструментів штучного інтелекту. Один із ключових елементів цього додатку - це чат-бот, який використовує інтелектуальну модель chatGPT [4].

Наукова новизна отриманих результатів:

1. Подальшого розвитку отримав метод розрахунку коефіцієнта навантаженості для ефективного визначення оптимальної тривалості самостійної роботи студента. На відміну від існуючих методів, запропонована модель використовує багатофакторний підхід, де враховуються не лише кількість завдань, але й їхня складність, що робить його більш точним та гнучким інструментом для планування самостійної роботи студентів. Такий підхід може сприяти оптимізації часових ресурсів та забезпечити більш ефективну організацію навчального процесу.

2. Вперше запропоновано використання моделі chatgpt з вбудованим промптом в якості чат-боту екзаменатора. Цей чат-бот слугує інструментом для студентів у підготовці до іспитів та оцінюванні їхнього рівня знань. Використання моделі chatGPT дозволяє чат-боту генерувати індивідуальні тести та завдання, а також проводити оцінку відповідей студентів. Із вбудованим промптом чат-бот може адаптувати запитання та завдання до конкретного матеріалу, сприяючи систематизації та закріпленню знань. Цей інтелектуальний інструмент сприятиме покращенню якості навчання та

підготовки студентів до іспитів, допомагаючи їм активно впроваджувати отримані знання в практику.

Android-додаток для організації самостійної роботи студентів плануємо вивести на ринок. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для цього випадку мають бути виконані такі етапи робіт:

а) проведення комерційного аудиту науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

б) розрахунок витрат на здійснення науково-технічної розробки;

в) розрахунок економічної ефективності науково-технічної розробки у випадку її впровадження та комерціалізації потенційним інвестором і обґрунтування економічної доцільності комерціалізації потенційним інвестором розробленої у магістерській кваліфікаційній роботі науково-технічної розробки.

5.2 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, які є провідними викладачами випускової кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснімо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного

рівня і комерційного потенціалу розробки та бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
№	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
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 ринку
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 значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної Динаміки	Ринок малий, але має позитивну Динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна Конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	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 фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 5.1

1	2	3	4	5	6
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту.	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 зведемо до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	3	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	2
5. Ринкові переваги (експлуатаційні витрати)	5	5	3
6. Ринкові перспективи (розмір ринку)	3	3	2
7. Ринкові перспективи (конкуренція)	4	5	2
8. Практична здійсненність (наявність фахівців)	3	3	2
9. Практична здійсненність (наявність фінансів)	5	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	2	4
11. Практична здійсненність (термін реалізації)	4	2	4
12. Практична здійсненність (розробка документів)	5	3	4
Сума балів	*СБ₁=49*	*СБ₂=40*	*СБ₃=37*
Середньоарифметична сума балів *СБс*	$СБ_с = \frac{\sum_{i=1}^{3} СБ_1}{3}=42$		

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.

Таблиця 5.3– Науково-технічні рівні та комерційні потенціали розробки

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

Згідно проведених досліджень рівень комерційного потенціалу розробки становить 42 бали, що, згідно таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього.

5.3 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної, дослідно-конструкторської, конструкторсько-технологічної роботи, створенням дослідного зразка і здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- спецустаткування для наукових (експериментальних) робіт;
- програмне забезпечення для наукових (експериментальних) робіт;
- витрати на роботи, які виконують сторонні підприємства, установи і організації;

– інші витрати;

– накладні (загальновиробничі) витрати.

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, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників (Зо) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \qquad (5.2)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

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

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, $T_p=21…23$ дні.

Проведені розрахунки зведемо до таблиці 5.4

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
1	2	3	4	5
Керівник відділу	25 000	1417	25	29762
Науковий працівник	22 000	1247	25	26190
Аспірант	15 000	850	25	17857
Front end розробник	20000	1134	25	23810

Продовження таблиці 5.4

1	2	3	4	5
Back end розробник	20000	1134	25	23810
Системний аналітик	18000	1020	25	21429
Qa тестувальник	17000	964	25	20238
Всього				163095

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1} C_i \cdot t_i \,, \tag{5.3}$$

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

t_i – час роботи робітника на виконання певної роботи, год.

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

$$\mathrm{С_і} = \frac{\mathrm{М_м} \cdot \mathrm{К_і} \cdot \mathrm{К_с}}{\mathrm{Т_р} \cdot \mathrm{Т_{зм}}}, \tag{5.4}$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – 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.

$Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р = 21…23$ *дні*;

$t_{зм}$ – тривалість зміни, год.

$$\mathrm{С_і} = \frac{6700 \cdot 1{,}5 \cdot 2}{21 \cdot 8} = 131.1 \text{ грн}$$

Таблиця 5.5 – 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
Підготовка робочих місць	40	6	2	120	4 800
Встановлення ПЗ	40	5	1,7	102	4080
Налагоджувальник устаткування	40	5	1,7	102	4080
Всього					12960

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = З_о + З_р \cdot \frac{Н_{дод}}{100\%}, \quad (5.5)$$

де $Н_{дод}$ - норма нарахування додаткової заробітної плати.

$$З_{дод} = (163095 + 12960) \cdot 0{,}1 = 17605.5 \text{ грн.}$$

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідниківі робітників за формуло:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (5.6)$$

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

$$З_{дод} = (163095 + 12960 + 17605.5) \cdot 0{,}22 = 42\,605 \text{ грн.}$$

Витрат на сировину, матеріали та комплектуючі не передбачено.

Спецустаткування для наукових (експериментальних) робіт

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i=1}^{к} Ц_i \cdot С_{пр} \cdot К_i \quad (5.7)$$

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

Спр - кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.

$К_i$ - коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_i$ =1,10 ….1,12)

k = кількість найменувань устаткування

Таблиця 5.6 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Комп'ютер	4	30 000	120 000
Принтер	1	3 000	3 000
Всього			123 000

Витрати на придбання спецустаткування становитимуть:

$В_{спец}$= 123 000 ·1,1=135 300 грн.

До статті «Програмне забезпечення для наукових 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 ПЗ розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{к} Ц_{iпрг} \cdot С_{прг} \cdot К_i \quad (5.8)$$

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

$С_{прг}$ - кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

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

k – кількість найменувань програмних засобів. Отримані результати зведемо до таблиці 5.7

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Операційна система Windows	4	8 000	32 000
Microsoft Visual Studio 2022 Professional	4	3 500	14 000
Всього			46 000

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення.

Витрати на придбання програмних засобів становитимуть :

$$В_{прог}=46\ 000\cdot 1,1=50\ 600 \text{ грн.}$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувалися під час (чи для) виконання даного етапу роботи розраховуються за формулою:

$$А = \frac{Ц}{Тв} \cdot \frac{tk}{12}, \quad (5.9)$$

де Ц – загальна балансова вартість всього обладнання, що використовувались для виконання даного етапу роботи, грн.;

$Т_в$ – термін корисного використання, роки;

t_k – термін використання обладнання, приміщень тощо, місяці.

Розрахунки амортизаційних витрат для даного програмного забезпечення зведені в табл. 5.8.

Таблиця 5.8 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Кількість	Балансова вартість, грн.	Термін корисного використання, роки	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Комп'ютер	4	30 000	3	1	3 333
Принтер	1	3 000	3	1	83
Операційна система Windows	4	8 000	2	1	1 333
Microsoft Visual Studio 2022 Professional	4	3 500	2	1	583
	Всього:				5 332

Витрати на силову електроенергію (B_e) розраховують за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_B \cdot К_{впі}}{\eta_i} \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання, кВт;

t_i – тривалість роботи обладнання;

$Ц_е$ - вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії);

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$П_i$ – коефіцієнт корисної дії обладнання $П_i < 1$

$$B_e = \frac{0{,}9 \cdot 8 \cdot 7{,}5 \cdot 0{,}9}{0{,}9} = 54 \text{ грн.}$$

Проведені розрахунки занесемо до таблиці 4.5.

Таблиця 5.9 – Витрати на електроенергію

Найменування операції	Найменування обладнання	Встановлена потужність, кВт	Тривалість операції, год	Кількість днів	Сума, грн
Front end програмування	Комп'ютер	0,9	8	21	1134
Back end програмування	Комп'ютер	0,9	8	21	1134
Системний аналіз	Комп'ютер	0,9	8	21	1134
Qa тестування	Комп'ютер	0,9	8	21	1134
Друк, сканування	Принтер	0,78	2	21	246
				Всього	4 782

Витрати на службові відрядження та витрати на роботи які виконують сторонні підприємства, установи, організації – не передбачено.

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» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{зп}}{100\%}, \quad (5.11)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Інші витрати складатимуть:

$I_в = (163095 + 12960) \cdot 0{,}5 = 88\,027{,}5$ грн.

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» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (5.12)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$В_{нзв} = (163095 + 12960) \cdot 1 = 176\,055$

$$В_{заг} = З_о + З_р + З_{дод} + З_н + В_{спец} + В_{прг} + А_{обл} + В_е + I_{в+} В_{нзв}\ , \qquad (5.13)$$

$В_{заг}$ =163095+12960+17605.5+42605+123000+46000+5332+4782+88027,5+176055=679462

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

$$ЗВ = \frac{В_{заг}}{\mu}, \qquad (5.14)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться настадії:

науково-дослідних робіт, то μ =0,1; технічного проектування, то μ =0,2; розробки конструкторської документації, то μ =0,3; розробки технологій, то μ=0,4; розробки дослідного зразка, то μ =0,5; розробки промислового зразка, то μ=0,7; впровадження, то μ =0,9.

ЗВ= 515009/0,9= 754 958 грн.

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

У даному підрозділі кількісно спрогнозуємо, яку вигоду можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.15)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році;

N - основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість.

Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

p – коефіцієнт, який враховує рентабельність продукту, p = 0,2;

υ – ставка податку на прибуток. У 2023 році — 18%.

Припустимо, що при прогнозованій ціні 700 грн. за послугу доступу до сервісу, в перший рік термін збільшення прибутку складе 3 роки. Після

завершення розробки і її вдосконалення, можна буде підняти його ціну на 100 грн. Кількість наданих послуг також збільшиться: протягом першого року — на 12000 шт., протягом другого року — на 4000 шт., протягом третього року на 1000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (700 + 100)*12000* 0,8333* 0,2 * (1 - 0,18) =1311948 грн.

$\Delta П_2$ = (700 + 100)*(12000+4000)* 0,8333* 0,2 * (1 - 0,18) = 1749263 грн.

$\Delta П_3$ = (700 + 100)*(12000+4000+1000)* 0,8333* 0,2 * (1 - 0,18) = 1858592 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 4919803 грн.

Далі розраховують приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.16)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої МКР, грн;

Т – період часу, протягом якого виявляються результати впровадженої МКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,15;

t – період часу (в роках).

$$ПП = \frac{1311948}{(1+0{,}15)^1} + \frac{1749263}{(1+0{,}15)^2} + \frac{1858592}{(1+0{,}15)^3} = 3688782$$

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково- технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ \qquad (5.17)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки. $k_{інв} = 2….5$

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

$$PV = 2 \cdot 679\,462 = 1\,358\,924 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід(NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (5.18)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.

$E_{абс}$= 3688782– 1358924= 2329858 грн.

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів роботи може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.19)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{2329858}{1358924}} - 1 = 1{,}394 – 1 = 0{,}394$$

Далі слід порівняти розраховану величину $E_в$ з мінімальною (бар’єрною) ставкою дисконтування τ_{min}, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар’єрна) ставка дисконтування визначається за формулою:

$$\tau_{min} = d + f, \quad (5.20)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,9...0,12);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f= (0,05...0,5), але може бути і значно більше.

$$\tau_{min} = 0,12 + 0,08 = 0,2.$$

Оскільки $E_в > \tau_{min}$, то інвестор буде зацікавлений у фінансуванні програмного забезпечення для організації самостійної роботи студента.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (5.21)$$

Для програмного забезпечення організації самостійної роботи студента термін окупності складе:

$$T_{ок} = 1/0,394 = 2,56 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки до розділу

У даному розділі було проаналізовано економічну доцільність розробки програмного засобу для організації самостійної роботи студента з використанням технологій штучного інтелекту. Опитування експертів показало, що запропонована розробка має високий рівень комерційного потенціалу. Вкладання коштів на виконання та впровадження результатів МКР є доцільним. Дана розробка має високий рівень щорічної ефективності та термін окупності складає 2,56 р.

ВИСНОВКИ

У рамках магістерської кваліфікаційної роботи створено Android-додаток для організації самостійної роботи студентів з використанням технологій штучного інтелекту. Цей додаток спрямований на оптимізацію навчального процесу шляхом організації індивідуальної праці студентів за допомогою функціональних можливостей мобільного додатка.

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 магістерської роботи виконано аналіз варіантів управління графічним інтерфейсом додатку, обрані оптимальні та зручні варіанти для користувача. Розроблено метод вирахування коефіцієнта навантаженості студента та метод роботи чат-боту екзаменатора із використанням технологій промпт-інженерії. Розроблено блок-схеми алгоритмів сортування, генерування статистичного звіту та виводу задач відповідно до обраної дати.

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 яких 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. Коваленко О.О. Методи та засоби програмної інженерії для удосконалення системи електронного університету [Електронний ресурс] // «L Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії» - Вінниця 2021 – Режим доступу: https: //conferences.vntu.edu.ua/index.php/znanosv/all-fitki-2021/paper/view/12313
2. Anderson, Lorin W.; Krathwohl, David R., ред. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. New York: Longman. ISBN 978-0-8013-1903-7.
3. What is Android [Електронний ресурс] – Режим доступу: https://www.android.com/what-is-android/
4. ChatGPT [Електронний ресурс] - Режим доступу: https://chat.openai.com/
5. Сергієнко В.П., Малежик М.П., Сіткар Т.В. Комп’ютерні технології в тестуванні: навч. посіб. – Луцьк: СПД Гадяк Жанна Володимирівна, друкарня «Волиньполіграф», 2012. – 290 с.
6. Вергун А. Р., Савенкова Л. В., Чуканова С. О. Програмне забезпечення для перевірки наукових текстів на плагіат: інформаційний огляд. Українська бібліотечна асоціація. Київ : УБА, 2016. - 36 с.
7. Р. М. Пархоменко, Г. Б. Ракитянська Розробка android-додатку “live to learn” для самостійної роботи студента з використанням технологій штучного інтелекту [Електронний ресурс] // «LII науково-технічна конференція підрозділів ВНТУ – 2023» – Вінниця 2023 - Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17425
8. Zhuosheng Zhang, Aston Zhang, Mu Li, Alex Smola. Automatic Chain of Thought Prompting in Large Language Models - Shanghai Jiao Tong University 2022 – 32 с.
9. Peggy Grant, Dale Basye. Personalized Learning: A Guide for Engaging Students with Technology - International Society for Technology in Education, 2014 - 200 с.

10. Nathan Hunter. The Art of Prompt Engineering with chatGPT: A Hands-On Guide, 2023 – 176 с.
11. Р. М. Пархоменко, Г. Б. Ракитянська Роль штучного інтелекту в персоналізації освітнього процесу : розробка чат-боту екзаменатора за допомогою промпт інженерії [Електронний ресурс] // «Міжнародна науково-практична Інтернет-конференція «Електронні інформаційні ресурси: створення, використання, доступ» - Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. – 336 с.
12. Jennifer K. Knight, Michelle Smith. Different but Equal? How Nonmajors and Majors Approach and Learn Genetics - University of Colorado Boulder, 2010 – 17с.
13. Robert C., Martin C. Architecture : a craftsman's guide to software structure and design - Beijing Shi, 2018. 400 с.
14. Humberto Cervantes, Rick Kazman. Designing Software Architectures. A Practical Approach - Print2print, 2016. 292 с.
15. Mark Wickham. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. Apress Berkeley, CA. 2018 – 228 с
16. Matt Neuburg. iOS 14 Programming Fundamentals with Swift - O'Reilly Media, Inc., 2020. – 234 с.
17. Brian Goetz. Java Concurrency in Practice, Pearson Education, Inc. 2020 – 351 с.
18. Dawn Griffiths, David Griffiths. Head First Kotlin: A Brain-Friendly Guide, O'Reilly. 2019 – 480 с.
19. Richard Warburton, Raoul-Gabriel Urma. Real-World Software Development: A Project-Driven Guide to Fundamentals in Java 1st Edition - O'Reilly, 2019. – 190 с.
20. Ian F. Darwin. Android Cookbook: Problems and Solutions for Android Developers – O'Reilly. 2017 – 480 с.
21. Lars Vogel. Eclipse 4 RCP: The complete guide to Eclipse application development. 2013 – 734 с.

22. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/
23. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с.
24. Hans-Jürgen Schönig. Mastering PostgreSQL 15: Advanced techniques to build and manage scalable, reliable, and fault-tolerant database - Packt Publishing, 2023 – 522 с.
25. Daniel Nichter. Efficient MySQL Performance: Best Practices and Techniques - O'Reilly, 2021 – 276 с.
26. Jay A. Kreibich. Using SQLite: Small. Fast. Reliable. Choose Any Three - O'Reilly, 2010 – 526 с.
27. Доун Гріффітс, Девід Гріффітс. Using Constraint Layouts in Android Studio - O'Reilly Media, Inc., 2017. - 228 с.
28. Mark L. Murphy. The busy coder's guide to Android development – CommonsWare, 2008. – 379 с.
29. OkHttpClient [Електронний ресурс]: - Режим доступу: https://square.github.io/okhttp/4x/okhttp/okhttp3/-ok-http-client/
30. Калбертсон Роберт, Браун Кріс, Кобб Гері Швидке тестування - М.: "Вільямс", 2002. - 374 с.

ДОДАТКИ

Додаток А – Технічне завдання

95

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
к.т.н., проф. О. Н. Романюк
"_19_" _вересня_ 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій штучного інтелекту»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доц. Г.Б. Ракитянська
"_19_" _Вересня_ 2023 р.
Виконав:
_____ студент гр. 3ПІ-22м Р.М. Пархоменко
"_19_" _Вересня_ 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій штучного інтелекту».

Галузь застосування – мобільні навчальні системи.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності освітнього процесу студента шляхом організації самостійної роботи за допомогою функціоналу мобільного додатку.

Призначення роботи – розробка методів і програмних засобів роботи Android-додатку з використанням технологій штучного інтелекту.

4. Вихідні дані для проведення НДР

1. Коваленко О.О. Методи та засоби програмної інженерії для удосконалення системи електронного університету [Електронний ресурс] // «L Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії» - Вінниця 2021 – Режим доступу: https://conferences.vntu.edu.ua/index.php/znanosv/all-fitki-2021/paper/view/12313

2. Zhuosheng Zhang, Aston Zhang, Mu Li, Alex Smola. Automatic Chain of Thought Prompting in Large Language Models - Shanghai Jiao Tong University 2022 – 32 с.

3. Peggy Grant, Dale Basye. Personalized Learning: A Guide for Engaging Students with Technology - International Society for Technology in Education, 2014 - 200 с.

4. Richard Warburton, Raoul-Gabriel Urma. Real-World Software Development: A Project-Driven Guide to Fundamentals in Java 1st Edition - O'Reilly, 2019. – 190 с.

5. Технічні вимоги

Вхідні дані – онтологічна модель предметної області; понятійний апарат даної галузі науки; координати вершин графу статистики, база даних задач SQLite, мовна модель штучного інтелекту gpt-3.5-turbo.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до магістерської кваліфікаційної роботи;
– технічне завдання;
– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії і етапи розробки

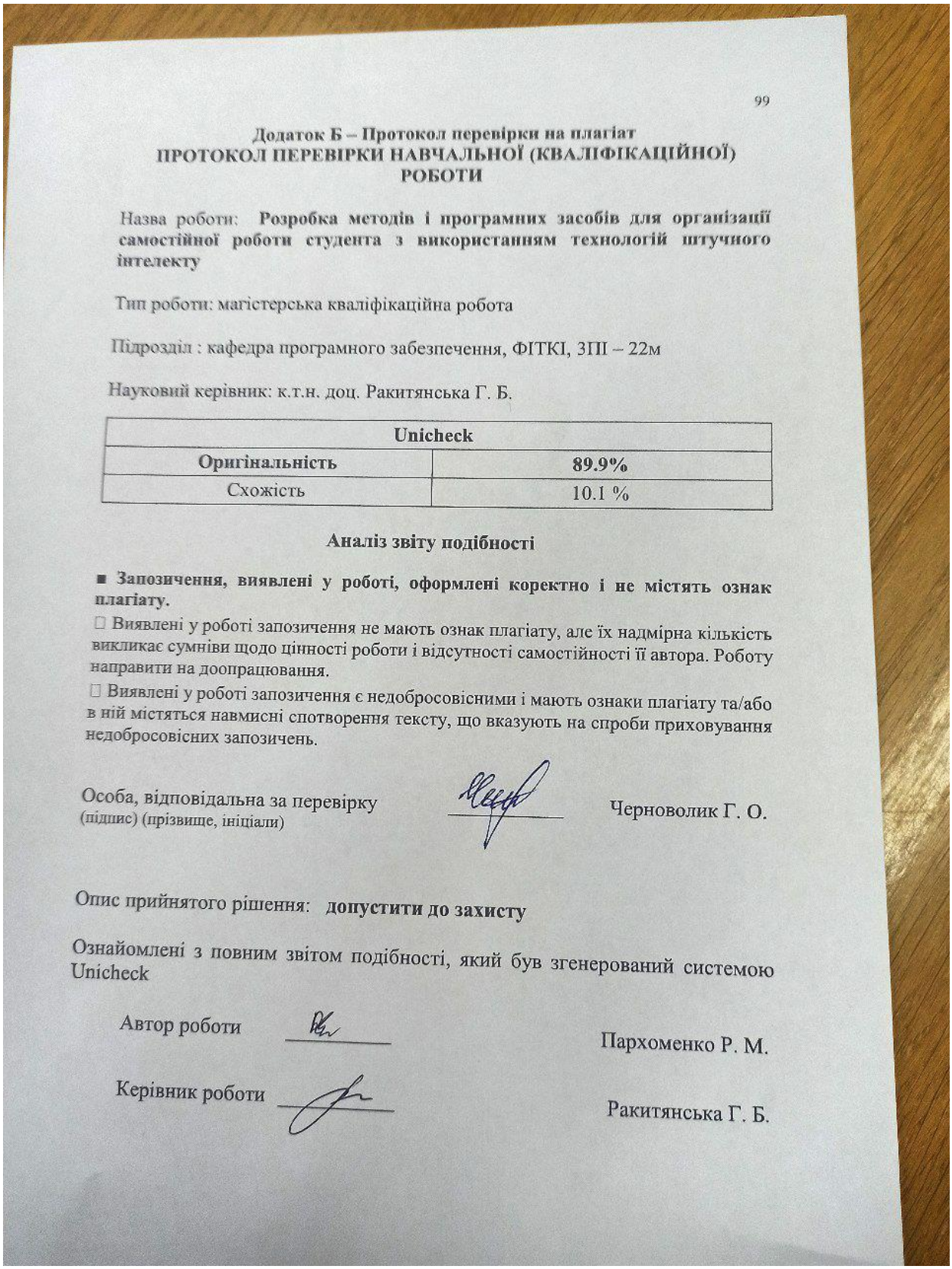
з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання організації самостійної роботи студента	20.09.2023-05.10.2023
2	Розробка методів та програмних засобів для реалізації додатку	06.10.2023-14.10.2023
3	Розробка програмних компонент Android-додатку для організації самостійної роботи студента	15.10.2023-01.11.2023
4	Тестування програмного додатку	01.11.2023-16.11.2023
5	Економічна частина	17.11.2023-01.12.2023

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.

Додаток Б – Протокол перевірки на плагіат



99

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і програмних засобів для організації самостійної роботи студента з використанням технологій штучного інтелекту**

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ЗПІ – 22м

Науковий керівник: к.т.н. доц. Ракитянська Г. Б.

Unicheck	
Оригінальність	**89.9%**
Схожість	10.1 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ 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 (підпис) (прізвище, ініціали) Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Пархоменко Р. М.

Керівник роботи Ракитянська Г. Б.

Додаток В – Лістинг програми

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:dist="http://schemas.android.com/apk/distribution"
        package="info.Parkhomenko.personaldiary">
        <uses-permission android:name="android.permission.INTERNET"/>
        <dist:module
            dist:instant="true" />

    <application
        android:name="info.Parkhomenko.personaldiary.App"
        android:allowBackup="true"
        android:icon="@drawable/agendas"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
android:name="info.Parkhomenko.personaldiary.view.ui.SplashActivity"
android:theme="@style/SplashTheme"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.DiariesActivity"
            android:label="Live to Learn"/>
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.CRUDActivity"
            android:theme="@style/CRUDTheme"
            android:label="Редагування задачі "

android:parentActivityName="info.Parkhomenko.personaldiary.view.ui.DiariesAct
ivity" />
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.DetailActivity"
            android:label="Деталі задачі "

android:parentActivityName="info.Parkhomenko.personaldiary.view.ui.DiariesAct
ivity" />
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.HelperActivity"
            android:label="Помічник"

android:parentActivityName="info.Parkhomenko.personaldiary.view.ui.DiariesAct
ivity" />
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.ExamActivity"
            android:label="Перевірка знань"

android:parentActivityName="info.Parkhomenko.personaldiary.view.ui.DiariesAct
ivity" />
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.RegistationActivity"
```

```
            android:label="Реєстрація "/>
        <activity

android:name="info.Parkhomenko.personaldiary.view.ui.LoginActivity"
            android:label="Вхід "
            />
    </application>

</manifest>
```

Utils.java

```
package info.Parkhomenko.personaldiary.common;

import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.view.Gravity;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import com.yarolegovich.lovelydialog.LovelyChoiceDialog;
import com.yarolegovich.lovelydialog.LovelyStandardDialog;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import info.Parkhomenko.personaldiary.R;
import info.Parkhomenko.personaldiary.data.model.Diary;
import info.Parkhomenko.personaldiary.view.ui.DiariesActivity;

public class Utils {
    public static final String DATE_FORMAT = "yyyy-MM-dd";

    public static void show(Context c,String message){
        Toast.makeText(c, message, Toast.LENGTH_SHORT).show();
    }

    public static boolean validate(EditText... editTexts){
        EditText nameTxt = editTexts[0],descriptionTxt = editTexts[1];

        if(nameTxt.getText() == null ||
nameTxt.getText().toString().isEmpty()){
            nameTxt.setError("Title is Required Please!");
            return false;
        }
        if(descriptionTxt.getText() == null ||
descriptionTxt.getText().toString().isEmpty()){
            descriptionTxt.setError("Description is Required Please!");
            return false;
        }
```

```
        return true;

    }

    public static void clearEditTexts(EditText... editTexts){
        for (EditText editText:editTexts) {
            editText.setText("");
        }
    }
    public static void openActivity(Context c,Class clazz){
        Intent intent = new Intent(c, clazz);
        c.startActivity(intent);
    }

    /**
     * This method will allow us show an Info dialog anywhere in our app.
     */
    public static void showInfoDialog(final AppCompatActivity activity,
String title,
                                      String message) {
        new LovelyStandardDialog(activity,
LovelyStandardDialog.ButtonLayout.HORIZONTAL)
                .setTopColorRes(R.color.indigo)
                .setButtonsColorRes(R.color.darkDeepOrange)
                .setIcon(R.drawable.m_info)
                .setTitle(title)
                .setMessage(message)
                .setPositiveButton("Relax", v -> {})
                .setNeutralButton("Go Home", v -> openActivity(activity,
DiariesActivity.class))
                .setNegativeButton("Go Back", v -> activity.finish())
                .show();
    }
    /**
     * This method will allow us show a single select dialog where we can
select and return an
     * item to an edittext.
     */
    public static void selectDialogItem(Context c,boolean difficulty,final
EditText itemTxt){
        String title = "Вибір категорії";
        String message = "Оберіть категорію для задачі";

         String[] dificulties ={"1","2", "3"};

        if(!difficulty){
            title = "Вибір пора доби";
            message = "Оберіть пору доби";
        }

        ArrayAdapter<String> adapter = new ArrayAdapter<>(c,
                android.R.layout.simple_list_item_1, dificulties);
        new LovelyChoiceDialog(c)
                .setTopColorRes(R.color.darkDeepOrange)
                .setTitle(title)
                .setTitleGravity(Gravity.CENTER_HORIZONTAL)
                .setIcon(R.drawable.m_star)
                .setMessage(message)
                .setMessageGravity(Gravity.CENTER_HORIZONTAL)
                .setItems(adapter, (position, item) -> itemTxt.setText(
item))
                .show();
    }
```

```
    /**
     * This method will allow us convert a string into a java.util.Date
object and
     *  return it.
     */
    public static Date giveMeDate(String stringDate){
        try {
            SimpleDateFormat sdf=new SimpleDateFormat(DATE_FORMAT);
            return sdf.parse(stringDate);
        }catch (ParseException e){
            e.printStackTrace();
            return null;
        }
    }
    /**
     * This method will allow us send a serialized diary objec  to a
specified
     *  activity
     */
    public static void sendDiaryToActivity(Context c, Diary diary,
                                           Class clazz){
        Intent i=new Intent(c,clazz);
        i.putExtra("DIARY_KEY", diary);
        c.startActivity(i);
    }

    /**
     * This method will allow us receive a serialized diary, deserialize it
and return it,.
     */
    public  static Diary receiveDiary(Intent intent, Context c){
        try {
            return (Diary) intent.getSerializableExtra("DIARY_KEY");
        }catch (Exception e){
            e.printStackTrace();
            show(c,"RECEIVING-DIARY ERROR: "+e.getMessage());
        }
        return null;
    }


    public static ArrayList<Diary> getDiariesForThisCategory(List<Diary>
allDiaries,
    String category){
        ArrayList<Diary> filteredDiaries = new ArrayList<>();
        if(allDiaries == null){
            return filteredDiaries;
        }
        for (Diary diary : allDiaries){
            if (diary != null &&
diary.getCategory().equalsIgnoreCase(category)){
                filteredDiaries.add(diary);
            }
        }
        return filteredDiaries;

    }

    public static ArrayList<Diary> getDiariesForThisDate(List<Diary>
allDiaries, String date){
        ArrayList<Diary> filteredDiaries =new ArrayList<>();

        if(allDiaries == null){
            return filteredDiaries;
```

```
        }
        for (Diary diary : allDiaries){
            if (diary != null && diary.getDate().equalsIgnoreCase(date)){
                filteredDiaries.add(diary);
            }
        }
        return filteredDiaries;

    }

    public static ArrayList<List<Diary>>
getAllDiariesGroupedByDates(List<Diary> allDiaries){
        ArrayList<List<Diary>> diariesGroupedByDates =new ArrayList<>();
        List<Diary> remainingDiaries = new ArrayList<>(allDiaries);

        for (Diary diary : allDiaries){
            if (diary != null && diary.getDate() != null){
                ArrayList<Diary> currentGroupDiaries =
Utils.getDiariesForThisDate(
                        remainingDiaries, diary.getDate());
                if(currentGroupDiaries != null && currentGroupDiaries.size()
> 0){
                    if(!diariesGroupedByDates.contains(currentGroupDiaries)){
                        diariesGroupedByDates.add(currentGroupDiaries);
                        for (Diary currentInnerChore : currentGroupDiaries){
                            remainingDiaries.remove(currentInnerChore);
                        }
                    }
                }
            }
        }
        Collections.reverse(diariesGroupedByDates);
        return diariesGroupedByDates;

    }
    @RequiresApi(api = Build.VERSION_CODES.N)
    public static    Map<String, Set<Diary>>
getAllDiariesGroupedByCategory(List<Diary> allDiaries){
        ArrayList<List<Diary>> getAllDiariesGroupedByCategory = new
ArrayList<>();
        List<Diary> remainingDiaries = new ArrayList<>(allDiaries);
        Map<String, Set<Diary>> map2 = new HashMap<>();
        for (Diary allDiary : allDiaries) {
            map2.computeIfAbsent(allDiary.getCategory(), k -> new
HashSet<>()).add(allDiary);
        }

        return map2;

    }
}
```

Diary.java

```
package info.Parkhomenko.personaldiary.data.model;

import androidx.annotation.NonNull;
import androidx.room.ColumnInfo;
import androidx.room.Entity;
import androidx.room.PrimaryKey;

import java.io.Serializable;
```

```
@Entity(tableName = "PersonalDiaryTB")
public class Diary implements Serializable {

    @NonNull
    @PrimaryKey
    @ColumnInfo(name = "id")
    private String id;
    @ColumnInfo(name = "title")
    private String title;
    @ColumnInfo(name = "description")
    private String description;
    @ColumnInfo(name = "date")
    private String date;

    @ColumnInfo(name = "category")
    private String category;
    @ColumnInfo(name = "dificulty")
    private int dificulty;
    @ColumnInfo(name = "userID")
    private String userID;


    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }

    public int getDificulty() {
        return dificulty;
    }
    public void setDificulty(int dificulty) {
        this.dificulty = dificulty;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }


    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }

    public String getDate() {
        return date;
    }
    public void setDate(String date) {
        this.date = date;
    }



    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
```

```
        this.category = category;
    }

    @Override
    public String toString() {
        return getTitle();
    }

    public String getUserID() { return userID; }

    public void setUserID(String userID) { this.userID = userID; }
}
```

DiaryRepository.java

```
package info.Parkhomenko.personaldiary.data.repository;

import android.content.Context;
import android.os.AsyncTask;

import androidx.lifecycle.LiveData;

import java.util.List;
import java.util.concurrent.ExecutionException;

import info.Parkhomenko.personaldiary.data.model.Diary;
import info.Parkhomenko.personaldiary.data.database.MyRoomDB;
import info.Parkhomenko.personaldiary.data.dao.DiaryDAO;

public class DiaryRepository {

    private static DiaryDAO diaryDAO;

    public DiaryRepository(Context context) {
        MyRoomDB database = MyRoomDB.getInstance(context);
        diaryDAO = database.diaryDAO();
    }

    static class SelectAllDataTask extends AsyncTask<Void, Void,
LiveData<List<Diary>>> {
        @Override
        protected LiveData<List<Diary>> doInBackground(Void... voids) {
            return diaryDAO.selectAll();
        }
    }
    public LiveData<List<Diary>> selectAllData() {
        try {
            return new SelectAllDataTask().execute().get();
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }

    static class InsertTask extends AsyncTask<Diary, Void, Long> {
        @Override
        protected Long doInBackground(Diary... diaries) {
            return diaryDAO.insert(diaries[0]);
        }
```

```
    }
    public Long insertData(Diary diary) {
        try {
            return new InsertTask().execute(diary).get();
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }

    static class UpdateTask extends AsyncTask<Diary, Void, Integer> {
        @Override
        protected Integer doInBackground(Diary... diaries) {
            return diaryDAO.update(diaries[0]);
        }
    }
    public Integer update(Diary diary) {
        try {
            return new UpdateTask().execute(diary).get();
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }

    static class DeleteTask extends AsyncTask<Diary, Void, Integer> {
        @Override
        protected Integer doInBackground(Diary... diaries) {
            return diaryDAO.delete(diaries[0]);
        }
    }
    public Integer delete(Diary diary) {
        try {
            return new DeleteTask().execute(diary).get();
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }

    static class DeleteAllTask extends AsyncTask<Void, Void, Integer> {
        @Override
        protected Integer doInBackground(Void... voids) {
            return diaryDAO.deleteAll();
        }
    }
    public Integer deleteAll() {
        try {
            return new DeleteAllTask().execute().get();
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

CRUDActivity.java

```
package info.Parkhomenko.personaldiary.view.ui;


import android.annotation.SuppressLint;
import android.content.Context;
import android.helper.DateTimePickerEditText;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NavUtils;
import androidx.lifecycle.ViewModelProviders;

import java.util.Random;

import info.Parkhomenko.personaldiary.R;
import info.Parkhomenko.personaldiary.common.CacheManager;
import info.Parkhomenko.personaldiary.data.model.Diary;
import info.Parkhomenko.personaldiary.common.Utils;
import info.Parkhomenko.personaldiary.viewmodel.DiaryViewModel;
import io.github.inflationx.viewpump.ViewPumpContextWrapper;

public class CRUDActivity extends AppCompatActivity {


    private EditText titleTxt, descriptionTxt,categoryTxt,difficultyTxt,
categoriesTxt;

    private TextView headerTxt;
    private DateTimePickerEditText dateTxt;
    private final Context c = CRUDActivity.this;
    private DiaryViewModel diaryViewModel;
    private Diary receivedDiary;
    private final String[] dificulties = {"Ранок","Обід","Вечір"};


    private void initializeWidgets() {
        headerTxt = findViewById(R.id.headerTxt);
        titleTxt = findViewById(R.id.titleTxt);
        descriptionTxt = findViewById(R.id.descriptionTxt);
        categoryTxt = findViewById(R.id.categoryTxt);
        dateTxt = findViewById(R.id.dateTxt);
        dateTxt.setFormat(Utils.DATE_FORMAT);
        categoriesTxt = findViewById(R.id.categoryTxt);
        difficultyTxt = findViewById(R.id.dificultyTxt);
    }

    private void listenToEditTextClicks(){

        difficultyTxt.setOnClickListener(v ->
Utils.selectDialogItem(CRUDActivity.this,false,
                difficultyTxt));
    }

    private void insertDiary(String title, String description, String
category, String date,
      int difficulty, String categories) {
        Diary diary = new Diary();
        diary.setId(String.valueOf(System.currentTimeMillis()));
```

```
            diary.setTitle(title);
            diary.setDescription(description);
            diary.setCategory(category);
            diary.setDate(date);
            diary.setDificulty(difficulty);
            diary.setCategory(categories);
            Long result = diaryViewModel.insert(diary);
            if (result != null) {
                if (result > -1) {
                    CacheManager.DIARIES_DIRTY=true;
                    Utils.clearEditTexts(titleTxt, descriptionTxt,dateTxt);
                    Utils.show(this, "INSERT SUCCESSFUL");
                } else {
                    Utils.show(this, "INSERT UNSUCCESSFUL");
                }
            } else {
                Utils.show(this, "UNSUCCESSFUL. ERROR OCCURED");
            }
        }

        private void insertData() {
            String title, description, date,category;
            int difficulty = 1;
            if (Utils.validate(titleTxt, descriptionTxt,difficultyTxt)) {
                title = titleTxt.getText().toString();
                description = descriptionTxt.getText().toString();
                category = categoryTxt.getText().toString();
                difficulty =
    Integer.parseInt(difficultyTxt.getText().toString());
                if (dateTxt.getDate() != null) {
                    date = dateTxt.getFormat().format(dateTxt.getDate());
                } else {
                    dateTxt.setError("Invalid Date");
                    dateTxt.requestFocus();
                    return;
                }
                insertDiary(title, description,
    category,date,difficulty,category);
            }
        }

        private void updateDiary(String title, String description,String
    category, String date,
        int difficulty) {
            receivedDiary.setTitle(title);
            receivedDiary.setDescription(description);
            receivedDiary.setDate(date);
            receivedDiary.setCategory(category);
            receivedDiary.setDificulty(difficulty);
            Integer result = diaryViewModel.update(receivedDiary);
            if (result != null) {
                if (result > 0) {
                    CacheManager.DIARIES_DIRTY=true;
                    Utils.show(this, "UPDATE SUCCESSFUL");
                } else {
                    Utils.show(this, "UPDATE UNSUCCESSFUL");
                }
            } else {
                Utils.show(this, "UNSUCCESSFUL. ERROR OCCURED");
            }
        }

        private void updateData() {
            String title, description, date, category;
```

```
        int difficulty;
        if (Utils.validate(titleTxt, descriptionTxt)) {
            title = titleTxt.getText().toString();
            description = descriptionTxt.getText().toString();
            category = categoryTxt.getText().toString();
            difficulty =
Integer.parseInt(difficultyTxt.getText().toString());

            if (dateTxt.getDate() != null) {
                date = dateTxt.getFormat().format(dateTxt.getDate());
            } else {
                dateTxt.setError("Invalid Date");
                dateTxt.requestFocus();
                return;
            }
            updateDiary(title, description,category, date, difficulty);

        }
    }

    private void deleteData() {
        Integer result = diaryViewModel.delete(receivedDiary);
        if (result != null) {
            if (result > 0) {
                CacheManager.DIARIES_DIRTY=true;
                Utils.show(this, "DELETE SUCCESSFUL");
            } else {
                Utils.show(this, "DELETE UNSUCCESSFUL");
            }
        } else {
            Utils.show(this, "UNSUCCESSFUL. ERROR OCCURED");
        }
    }

    @Override
    public void onBackPressed() {
        Utils.showInfoDialog(this, "Warning", "Are you sure you want to
exit?");
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        if (receivedDiary == null) {
            getMenuInflater().inflate(R.menu.new_item_menu, menu);
            headerTxt.setText("Додавання задачі");
        } else {
            getMenuInflater().inflate(R.menu.edit_item_menu, menu);
            headerTxt.setText("Редагування задачі");
        }
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.insertMenuItem:
                insertData();
                return true;
            case R.id.editMenuItem:
                if (receivedDiary != null) {
                    updateData();
                } else {
                    Utils.show(this, "EDIT ONLY WORKS IN EDITING MODE");
                }
```

```
                return true;
            case R.id.deleteMenuItem:
                if (receivedDiary != null) {
                    deleteData();
                } else {
                    Utils.show(this, "DELETE ONLY WORKS IN EDITING MODE");
                }
                return true;
            case R.id.viewAllMenuItem:
                Utils.openActivity(this, DiariesActivity.class);
                finish();
                return true;
            case android.R.id.home:
                NavUtils.navigateUpFromSameTask(this);
                finish();
                return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void attachBaseContext(Context newBase) {
        super.attachBaseContext(ViewPumpContextWrapper.wrap(newBase));
    }


    @Override
    protected void onResume() {
        super.onResume();
        Diary o = Utils.receiveDiary(getIntent(), c);
        if (o != null) {
            receivedDiary = o;
            titleTxt.setText(receivedDiary.getTitle());
            descriptionTxt.setText(receivedDiary.getDescription());
            categoryTxt.setText(receivedDiary.getCategory());

difficultyTxt.setText(String.valueOf(receivedDiary.getDificulty()));

            Object date = receivedDiary.getDate();
            if (date != null) {
                dateTxt.setDate(Utils.giveMeDate(date.toString()));
            }
        } else {

        }
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_crud);

        diaryViewModel =
ViewModelProviders.of(this).get(DiaryViewModel.class);

        this.initializeWidgets();
        this.listenToEditTextClicks();
    }
}
```

DetailActivity.java

```
    package info.Parkhomenko.personaldiary.view.ui;

    import android.content.Context;
    import android.os.Bundle;
    import android.view.Menu;
    import android.view.MenuItem;
    import android.view.View;
    import android.widget.TextView;

    import androidx.appcompat.app.AppCompatActivity;
    import androidx.core.app.NavUtils;

    import com.google.android.material.appbar.CollapsingToolbarLayout;
    import
com.google.android.material.floatingactionbutton.FloatingActionButton;

    import info.Parkhomenko.personaldiary.R;
    import info.Parkhomenko.personaldiary.data.model.Diary;
    import info.Parkhomenko.personaldiary.common.Utils;
    import io.github.inflationx.viewpump.ViewPumpContextWrapper;

    public class DetailActivity extends AppCompatActivity implements
View.OnClickListener {


        private TextView
titleTV,descriptionTV,dateTV,descriptionTV2,dificultyTV,markAsDoneButton;
        private FloatingActionButton editFAB;
        private Diary receivedDiary;
        private CollapsingToolbarLayout mCollapsingToolbarLayout;


        private void initializeWidgets(){
            titleTV= findViewById(R.id.titleTV);
            descriptionTV= findViewById(R.id.descriptionTV);
            dateTV= findViewById(R.id.dateTV);
            dificultyTV = findViewById(R.id.dificultyTV);
            descriptionTV2= findViewById(R.id.descriptionTV2);
            editFAB=findViewById(R.id.editFAB);
            editFAB.setOnClickListener(this);

mCollapsingToolbarLayout=findViewById(R.id.mCollapsingToolbarLayout);
        }


        private void receiveAndShowData(){
             receivedDiary =
Utils.receiveDiary(getIntent(),DetailActivity.this);

            if(receivedDiary != null){
                titleTV.setText(receivedDiary.getTitle());
                descriptionTV.setText(receivedDiary.getDescription());
                descriptionTV2.setText(receivedDiary.getCategory());
                dateTV.setText(receivedDiary.getDate());

dificultyTV.setText(String.valueOf(receivedDiary.getDificulty()));

                mCollapsingToolbarLayout.setTitle(receivedDiary.getTitle());

mCollapsingToolbarLayout.setExpandedTitleColor(getResources().
```

```
                getColor(R.color.white));
            }
        }


        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.detail_page_menu, menu);
            return true;
        }


        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case R.id.action_edit:
                    Utils.sendDiaryToActivity(this,
receivedDiary,CRUDActivity.class);
                    finish();
                    return true;
                case android.R.id.home:
                    NavUtils.navigateUpFromSameTask(this);
                    finish();
                    return true;
            }
            return super.onOptionsItemSelected(item);
        }


        @Override
        public void onClick(View v) {
            int id =v.getId();
            if(id == R.id.editFAB){
                Utils.sendDiaryToActivity(this,
receivedDiary,CRUDActivity.class);
                finish();
            }
        }

        @Override
        protected void attachBaseContext(Context newBase) {
            super.attachBaseContext(ViewPumpContextWrapper.wrap(newBase));
        }


        @Override
        public void onBackPressed() {
            super.onBackPressed();
            this.finish();
        }


        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_detail);

            initializeWidgets();
            receiveAndShowData();
        }

    }
```

DiariesActivity.java

```
package info.Parkhomenko.personaldiary.view.ui;

import android.content.Context;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AbsListView;
import android.widget.RadioGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.RecyclerView;

import com.github.jhonnyx2012.horizontalpicker.DatePickerListener;
import com.github.jhonnyx2012.horizontalpicker.HorizontalPicker;
import com.google.android.material.bottomnavigation.BottomNavigationView;

import org.joda.time.DateTime;

import java.util.ArrayList;
import java.util.Collections;
```

```
import java.util.List;
import java.util.Map;
import java.util.Set;

import info.Parkhomenko.personaldiary.R;
import info.Parkhomenko.personaldiary.common.CacheManager;
import info.Parkhomenko.personaldiary.common.Utils;
import info.Parkhomenko.personaldiary.data.model.Diary;
import info.Parkhomenko.personaldiary.data.model.Section;
import
info.Parkhomenko.personaldiary.view.adapter.SectionedExpandableLayoutHelper;
import info.Parkhomenko.personaldiary.view.callbacks.ItemClickListener;
import info.Parkhomenko.personaldiary.viewmodel.DiaryViewModel;
import io.github.inflationx.viewpump.ViewPumpContextWrapper;
import lecho.lib.hellocharts.gesture.ZoomType;
import lecho.lib.hellocharts.model.Axis;
import lecho.lib.hellocharts.model.AxisValue;
import lecho.lib.hellocharts.model.Column;
import lecho.lib.hellocharts.model.ColumnChartData;
import lecho.lib.hellocharts.model.PieChartData;
import lecho.lib.hellocharts.model.SliceValue;
import lecho.lib.hellocharts.model.SubcolumnValue;
import lecho.lib.hellocharts.util.ChartUtils;
import lecho.lib.hellocharts.view.ColumnChartView;
import lecho.lib.hellocharts.view.PieChartView;

public class DiariesActivity extends AppCompatActivity
        implements DatePickerListener, ItemClickListener {

    //We define our instance fields
```

```
private RecyclerView rv;
private HorizontalPicker picker;
private BottomNavigationView bottomNavigationView;
private SectionedExpandableLayoutHelper sectionedLayout;
private SectionedExpandableLayoutHelper textlayout;
private boolean defaultPage = true;
private DiaryViewModel diaryViewModel;
private boolean isScrolling = false;
private  ColumnChartView  chartView;
private ColumnChartData columnChartData;
private RadioGroup radioGroup;
private boolean hasLabels = false;
private TextView textView;
private PieChartView pieChart;
private PieChartData data;
/**
 * We initialize our widgets
 */
@RequiresApi(api = Build.VERSION_CODES.N)
private void initializeViews() {
    radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
    textView = (TextView) findViewById(R.id.textView);
   radioGroup.setVisibility(View.INVISIBLE);
   radioGroup.check(R.id.radio_by_date);
   radioGroup.setOnCheckedChangeListener((radioGroup1, i) -> {
      bindDairies();       });
   rv = findViewById(R.id.mRecyclerView);
   sectionedLayout = new
         SectionedExpandableLayoutHelper(this,rv, this,
         2);
```

```
        chartView=(ColumnChartView ) findViewById(R.id.chart);
        chartView.setVisibility(View.INVISIBLE);
        pieChart=(PieChartView)  findViewById(R.id.pieChart);
        pieChart.setVisibility(View.INVISIBLE);
        picker= findViewById(R.id.datePicker);
        picker.setListener(this)
                .setDays(120)
                .setOffset(7)
                .setDateSelectedColor(Color.DKGRAY)
                .setDateSelectedTextColor(Color.WHITE)
                .setMonthAndYearTextColor(Color.DKGRAY)

.setTodayButtonTextColor(getResources().getColor(R.color.colorPrimary))

.setTodayDateTextColor(getResources().getColor(R.color.colorPrimary))
                .setTodayDateBackgroundColor(Color.GRAY)
                .setUnselectedDayTextColor(Color.DKGRAY)
                .setDayOfWeekTextColor(Color.DKGRAY )

.setUnselectedDayTextColor(getResources().getColor(R.color.primaryTextColor))
                .showTodayButton(true)
                .init();
        picker.setBackgroundColor(Color.LTGRAY);
        picker.setDate(new DateTime());
        bottomNavigationView = findViewById(R.id.mBottomNavigation);
        }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private void listenToBottomNavigationClicks(){
        bottomNavigationView.setOnNavigationItemSelectedListener(
```

```
            item -> {
                switch (item.getItemId()) {
                    case R.id.action_home:
                        defaultPage=true;
                        bindDairiesCat();
                        break;
                    case R.id.time_manager:
                        defaultPage=false;
                        bindTimeManager();
                        break;
                    case R.id.action_all_dates:
                        defaultPage=false;
                        bindDairies();
                        break;
                    case R.id.action_helper:
                        defaultPage=false;
                        Utils.openActivity(this, HelperActivity.class);
                        break;
                    case R.id.action_exit:
                        finish();
                        break;
                }
                return false;
            });

    bottomNavigationView.setSelectedItemId(R.id.action_home);
}

private void listenToRecyclerViewScrolls(){
    rv.addOnScrollListener(new RecyclerView.OnScrollListener() {
```

```
        @Override
        public void onScrollStateChanged(@NonNull RecyclerView recyclerView,
int newState) {
            super.onScrollStateChanged(recyclerView, newState);
            if (newState ==
AbsListView.OnScrollListener.SCROLL_STATE_TOUCH_SCROLL) {
                isScrolling = true;
            }
        }

        @Override
        public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int
dy) {
            super.onScrolled(recyclerView, dx, dy);

            if(isScrolling){
                if(bottomNavigationView.getVisibility()== View.VISIBLE){
                    bottomNavigationView.setVisibility(View.GONE);
                }
                isScrolling=false;
            }else{
                Handler handler=new Handler();
                handler.postDelayed(() -> {
                    if(bottomNavigationView.getVisibility()==View.GONE){
                        bottomNavigationView.setVisibility(View.VISIBLE);
                    }
                },3000);
            }
        }
    });
```

```
    }


    private void bindTimeManager(){
        picker.setVisibility(View.GONE);
        radioGroup.setVisibility(View.INVISIBLE);
        textView.setVisibility(View.GONE);

        sectionedLayout=new
                SectionedExpandableLayoutHelper(this, rv, this,
                2);

        List<Column> columns = new ArrayList<Column>();
        List<Diary> diaries = CacheManager.ALL_DIARIES_MEMORY_CACHE;
        ArrayList<List<Diary>> diariesLists =
Utils.getAllDiariesGroupedByDates(diaries);
        Collections.reverse(diariesLists);
        List<SubcolumnValue> values;
        int i=0;
        List<AxisValue> axisValues = new ArrayList<AxisValue>();
        for (List<Diary> list: diariesLists){
            if(list != null && list.size()>0){
                String dateSection = list.get(0).getDate();
                values = new ArrayList<SubcolumnValue>();
                values.add(new SubcolumnValue(list.size(), ChartUtils.pickColor()));


                axisValues.add(new AxisValue(i).setLabel(dateSection));
```

```
            columns.add(new Column(values).setHasLabelsOnlyForSelected(true));

        i=i+1;
        }
    }

    columnChartData = new ColumnChartData(columns);
    columnChartData.setAxisXBottom(new Axis(axisValues));

columnChartData.setAxisYLeft(Axis.generateAxisFromRange(0,diariesLists.size()
+2,1).setHasLines(true));

    chartView.setColumnChartData(columnChartData);
    chartView.setValueSelectionEnabled(true);
    chartView.setZoomType(ZoomType.HORIZONTAL);
    chartView.setVisibility(View.VISIBLE);
    bindPieChart();

}


@RequiresApi(api = Build.VERSION_CODES.N)
private void bindDairiesCat() {
    List<Diary> diaries = CacheManager.ALL_DIARIES_MEMORY_CACHE;
    radioGroup.setVisibility(View.INVISIBLE);
    chartView.setVisibility(View.INVISIBLE);
    pieChart.setVisibility(View.INVISIBLE);
    picker.setVisibility(View.VISIBLE);
```

```
        textView.setVisibility(View.VISIBLE);

        TextView textView = findViewById(R.id.textView);
        List<Diary> todayDiaries = Utils.getDiariesForThisDate(diaries,
                CacheManager.SELECTED_DATE);

        int sum =0;
        for (Diary item : todayDiaries)
        {
           sum+=  item.getDificulty();

        }
        int res = 240 - ((sum*30));
        if (res <0){
           textView.setText("Коефіцієнт навантаженості на цей день перевищує норму, спробуйте розвантажити цей день, перенесом завдань на інший день");
        } else if (res >0 && todayDiaries.size()!=0) {
           textView.setText("Коефіцієнт навантаженості на цей день в межах норми, виконуйте завдання за планом");
        } else if (todayDiaries.size()==0) {
           textView.setText("На цей день задач немає, можете відпочити або використати цей день щоб розвантажити себе в інший");
        }

        Map<String, Set<Diary>> diariesLists = Utils.getAllDiariesGroupedByCategory(todayDiaries);

        sectionedLayout = new
                SectionedExpandableLayoutHelper(this, rv, this,
```

```
                2);
        List<Diary> List = null;
        for (Map.Entry<String, Set<Diary>> entry : diariesLists.entrySet()) {
            String category = entry.getKey();

            if (category == null || category.length() == 0) {
                category = "Без категрої";

            }
            List = new ArrayList<Diary>();
            List.addAll(entry.getValue());
            sectionedLayout.addSection(category + " (" + List.size() + ")",
(ArrayList<Diary>) List);

        }

        sectionedLayout.notifyDataSetChanged();
    }


    private void bindPieChart(){

        List<Diary> diaries = CacheManager.ALL_DIARIES_MEMORY_CACHE;

        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.N) {
            Map<String, Set<Diary>> diariesLists =
Utils.getAllDiariesGroupedByCategory(diaries);
```

```
        List<SliceValue> values = new ArrayList<SliceValue>();
        for (Map.Entry<String, Set<Diary>> entry : diariesLists.entrySet()) {
            String category = entry.getKey();

            SliceValue sliceValue = new SliceValue( entry.getValue().size(),
ChartUtils.pickColor());

            sliceValue.setLabel(category + " - " + entry.getValue().size());
            values.add(sliceValue);



        }



        data = new PieChartData(values);
        data.setHasLabels(true);

        data.setHasLabelsOnlyForSelected(false);
        data.setHasLabelsOutside(false);
        data.setHasCenterCircle(false);
        pieChart.setPieChartData(data);
        pieChart.setVisibility(View.VISIBLE);
        }
    }
```

```
    @RequiresApi(api = Build.VERSION_CODES.N)
    private void bindDairies(){
        List<Diary> diaries = CacheManager.ALL_DIARIES_MEMORY_CACHE;
        chartView.setVisibility(View.INVISIBLE);
        pieChart.setVisibility(View.INVISIBLE);
        radioGroup.setVisibility(View.VISIBLE);
        picker.setVisibility(View.GONE);
        textView.setVisibility(View.GONE);
        switch ( radioGroup.getCheckedRadioButtonId()) {
            case R.id.radio_by_date:
                ArrayList<List<Diary>> diariesByDateLists =
Utils.getAllDiariesGroupedByDates(diaries);
                sectionedLayout = new
                        SectionedExpandableLayoutHelper(this, rv, this,
                        2);
                for (List<Diary> list: diariesByDateLists){
                    if(list != null && list.size()>0){
                        String dateSection = list.get(0).getDate();
                        sectionedLayout.addSection(dateSection + " (" + list.size() + ")",
(ArrayList<Diary>) list);
                    }
                }
                break;
            case R.id.radio_by_category:
                Map<String, Set<Diary>> diariesLists =
Utils.getAllDiariesGroupedByCategory(diaries);
                sectionedLayout = new
                        SectionedExpandableLayoutHelper(this, rv, this,
                        2);
```

```
            for (Map.Entry<String, Set<Diary>> entry : diariesLists.entrySet()) {
                String category = entry.getKey();

                if(category==null || category.length()==0){
                    category="Без категорії";

                }
                List<Diary> List = new ArrayList<Diary>();
                List.addAll(entry.getValue());
                sectionedLayout.addSection(category + " (" + List.size() + ")",
(ArrayList<Diary>) List);
            }
            break;
        default:
            break;
    }
    sectionedLayout.notifyDataSetChanged();
}

@RequiresApi(api = Build.VERSION_CODES.N)
private void reloadDiaries() {
    diaryViewModel.getDiariesLiveData().observe(this, diaries -> {
        if (diaries != null && diaries.size() > 0) {
            CacheManager.ALL_DIARIES_MEMORY_CACHE = diaries;
            CacheManager.DIARIES_DIRTY=false;
            bindDairiesCat();
        }else{
            CacheManager.ALL_DIARIES_MEMORY_CACHE.clear();
        }
    });
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.diaries_page_menu, menu);
    return true;
}

@RequiresApi(api = Build.VERSION_CODES.N)
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_new:
            Utils.openActivity(this, CRUDActivity.class);
            finish();
            return true;
        case R.id.action_exam:
            defaultPage=false;
            Utils.openActivity(this, ExamActivity.class);
            return true;

        case R.id.action_switch_view:
            defaultPage = !defaultPage;
          bindDairies();

            return true;
        case R.id.action_exit:
            this.finish();
            return true;
```

```
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void attachBaseContext(Context newBase) {
    super.attachBaseContext(ViewPumpContextWrapper.wrap(newBase));
}

@Override
public void onBackPressed() {
    super.onBackPressed();
    this.finish();
}

    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    public void onDateSelected(DateTime dateSelected) {
        String year = String.valueOf(dateSelected.getYear());
        String month = String.valueOf(dateSelected.getMonthOfYear());
        String day = String.valueOf(dateSelected.getDayOfMonth());

        if(dateSelected.getMonthOfYear() < 10){
            month = "0" + month ;
        }
        if(dateSelected.getDayOfMonth() < 10){
            day = "0" + day;
        }
        CacheManager.SELECTED_DATE = year + "-" + month + "-" + day;
```

```
        if(CacheManager.DIARIES_DIRTY){
            reloadDiaries();
        }else{
//          bindDairies();
            bindDairiesCat();
        }

    }

@Override
public void itemClicked(Diary diary) {
    Utils.sendDiaryToActivity(this, diary,DetailActivity.class);
}

@Override
public void itemClicked(Section section) {

   Utils.show(this,section.getName()+" clicked");
}

@RequiresApi(api = Build.VERSION_CODES.N)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_diaries);

    diaryViewModel = ViewModelProviders.of(this).get(DiaryViewModel.class);

    this.initializeViews();
    this.listenToBottomNavigationClicks();
```

```
        this.listenToRecyclerViewScrolls();
    }
}
```

ExamActivity.java

```
package info.Parkhomenko.personaldiary.view.ui;


import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;

import info.Parkhomenko.personaldiary.R;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class  ExamActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    TextView welcomeTextView;
    EditText messageEditText;
    ImageButton sendButton;
    List<Message> messageList;
    JSONArray messageArray;
    MessageAdapter messageAdapter;
    int i =0;
    public static final MediaType JSON
            = MediaType.get("application/json; charset=utf-8");
    OkHttpClient client = new OkHttpClient.Builder()
            .readTimeout(90, TimeUnit.SECONDS)
            .build();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_exam);
        messageList = new ArrayList<>();
```

```
        messageArray = new JSONArray();
        recyclerView = findViewById(R.id.recycler_view);
        welcomeTextView = findViewById(R.id.welcome_text);
        messageEditText = findViewById(R.id.message_edit_text);
        sendButton = findViewById(R.id.send_btn);

        //setup recycler view
        messageAdapter = new MessageAdapter(messageList);
        recyclerView.setAdapter(messageAdapter);
        LinearLayoutManager llm = new LinearLayoutManager(this);
        llm.setStackFromEnd(true);
        recyclerView.setLayoutManager(llm);

        sendButton.setOnClickListener((v) -> {
            String question = messageEditText.getText().toString().trim();

            addToChat(question, Message.SENT_BY_ME);

            messageEditText.setText("");
            callAPI(question);
            welcomeTextView.setVisibility(View.GONE);
        });
    }

    void addToChat(String message, String sentBy) {
        runOnUiThread(new Runnable() {

            @Override
            public void run() {
                messageList.add(new Message(message, sentBy));
                messageAdapter.notifyDataSetChanged();

recyclerView.smoothScrollToPosition(messageAdapter.getItemCount());

            }
        });
    }

    void addResponse(String response) {
        messageList.remove(messageList.size() - 1);
        addToChat(response, Message.SENT_BY_BOT);
    }

    void callAPI(String question) {
        //okhttp
        messageList.add(new Message("Typing... ", Message.SENT_BY_BOT));

        JSONObject jsonBody = new JSONObject();
        JSONObject obj = new JSONObject();
        try {
            jsonBody.put("model", "gpt-3.5-turbo");
            obj.put("role", "user");
            if (i<1) {
                obj.put("content", "You are an examiner. Please ask me some
questions for the topic: " + question +
                        "\\nFor example:\n" +
                        "1. You asking\n" +
                        "2. I answer\n" +
                        "3. you're saying if my answer was correct. If I was
incorrect say \"Incorrect!\" " +
                        "and critique my response to each question and
provide example answers then ask another question. " +
                        "If I was correct say \"Correct!\" and ask another
question.\n" +
```

```
                        "Begin by asking the first question and then waiting
for my response. ");
                i++;
            } else {
                obj.put("content", question);
            }
            messageArray.put(obj);
            jsonBody.put("messages", messageArray);
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        RequestBody body = RequestBody.create(jsonBody.toString(), JSON);
        Request request = new Request.Builder()
                .url("https://api.openai.com/v1/chat/completions")
                .header("Authorization", "Bearer sk- ")
                .post(body)
                .build();

        client.newCall(request).enqueue(new Callback() {

            @Override
            public void onFailure(@NonNull Call call, @NonNull IOException e)
{
                addResponse("Failed to load response due to " +
e.getMessage() + "Sent the type of error to HelpLivetoLearn@gmail.com");
            }

            @Override
            public void onResponse(@NonNull Call call, @NonNull Response
response) throws IOException {
                if (response.isSuccessful()) {
                    JSONObject jsonObject;
                    try {
                        jsonObject = new
JSONObject(response.body().string());
                        JSONArray jsonArray =
jsonObject.getJSONArray("choices");
                        String result =
jsonArray.getJSONObject(0).getJSONObject("message").getString("content");
                        addResponse(result.trim());
                        JSONObject obj = new JSONObject();

                        obj.put("role", "assistant");
                        obj.put("content", result);
                        messageArray.put(obj);

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }


                } else {
                    addResponse("Failed to load response due to " +
response.body().toString());
                }
            }
        });
    }
}
```

HelperActivity.java

```
package info.Parkhomenko.personaldiary.view.ui;


import androidx.annotation.NonNull;
```

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;

import info.Parkhomenko.personaldiary.R;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class  HelperActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    TextView welcomeTextView;
    EditText messageEditText;
    ImageButton sendButton;
    List<Message> messageList;
    JSONArray messageArray ;
    MessageAdapter messageAdapter;
    public static final MediaType JSON
            = MediaType.get("application/json; charset=utf-8");
    OkHttpClient client = new OkHttpClient.Builder()
            .readTimeout(90, TimeUnit.SECONDS)
            .build();


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_helper);
        messageList = new ArrayList<>();
        messageArray= new JSONArray();
        recyclerView = findViewById(R.id.recycler_view);
        welcomeTextView = findViewById(R.id.welcome_text);
        messageEditText = findViewById(R.id.message_edit_text);
        sendButton = findViewById(R.id.send_btn);

        //setup recycler view
        messageAdapter = new MessageAdapter(messageList);
        recyclerView.setAdapter(messageAdapter);
        LinearLayoutManager llm = new LinearLayoutManager(this);
        llm.setStackFromEnd(true);
        recyclerView.setLayoutManager(llm);

        sendButton.setOnClickListener((v)->{
            String question = messageEditText.getText().toString().trim();
            addToChat(question,Message.SENT_BY_ME);
```

```
            messageEditText.setText("");
            callAPI(question);
            welcomeTextView.setVisibility(View.GONE);
        });
    }

    void addToChat(String message,String sentBy){
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                messageList.add(new Message(message,sentBy));
                messageAdapter.notifyDataSetChanged();

recyclerView.smoothScrollToPosition(messageAdapter.getItemCount());

            }
        });
    }

    void addResponse(String response){
        messageList.remove(messageList.size()-1);
        addToChat(response,Message.SENT_BY_BOT);
    }

    void callAPI(String question){
        //okhttp
        messageList.add(new Message("Typing... ",Message.SENT_BY_BOT));
        JSONObject jsonBody = new JSONObject();
        JSONObject obj = new JSONObject();
        try {
            jsonBody.put("model","gpt-3.5-turbo");
            obj.put("role", "user");
            obj.put("content", question);
            messageArray.put(obj);
            jsonBody.put("messages",messageArray);
        } catch (JSONException e) {
            throw new RuntimeException(e);
        }
        RequestBody body = RequestBody.create(jsonBody.toString(),JSON);
        Request request = new Request.Builder()
                .url("https://api.openai.com/v1/chat/completions")
                .header("Authorization","Bearer ")
                .post(body)
                .build();
        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(@NonNull Call call, @NonNull IOException e)
{
                addResponse("Failed to load response due to "+e.getMessage()
+ "Sent the type of error to HelpLivetoLearn@gmail.com");
            }

            @Override
            public void onResponse(@NonNull Call call, @NonNull Response
response) throws IOException {
                if(response.isSuccessful()){
                    JSONObject  jsonObject;
                    try {
                        jsonObject = new
JSONObject(response.body().string());
                        JSONArray jsonArray =
jsonObject.getJSONArray("choices");
                        String result =
jsonArray.getJSONObject(0).getJSONObject("message").getString("content");
```

```
                        addResponse(result.trim());
                        JSONObject obj = new JSONObject();

                        obj.put("role", "assistant");
                        obj.put("content", result);
                        messageArray.put(obj);

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }


                }else{
                    addResponse("Failed to load response due to
"+response.body().toString());
                }
            }
        });



    }

}
```

SplashActivity.java

```
package info.Parkhomenko.personaldiary.view.ui;

import android.content.Context;
import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProviders;

import info.Parkhomenko.personaldiary.R;
import info.Parkhomenko.personaldiary.common.CacheManager;
import info.Parkhomenko.personaldiary.common.Utils;
import info.Parkhomenko.personaldiary.viewmodel.DiaryViewModel;
```

```
import io.github.inflationx.viewpump.ViewPumpContextWrapper;

public class SplashActivity extends AppCompatActivity {

    private ImageView mLogo;
    private TextView mainTitle, subTitle;

    private void initializeWidgets() {
        mLogo = findViewById(R.id.mLogo);
        mainTitle = findViewById(R.id.mainTitle);
        subTitle = findViewById(R.id.subTitle);
    }



    private void proceed() {
        Thread t = new Thread() {
            @Override
            public void run() {
                try {
                    sleep(500);
                    Utils.openActivity(SplashActivity.this, RegistationActivity.class);
                    finish();
                    super.run();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        };
        t.start();
    }
```

```
    private void preloadDiaries() {
        DiaryViewModel diaryViewModel =
ViewModelProviders.of(this).get(DiaryViewModel.class);
        diaryViewModel.getDiariesLiveData().observe(this, diaries -> {
            if (diaries != null && diaries.size() > 0) {
                CacheManager.ALL_DIARIES_MEMORY_CACHE = diaries;
                CacheManager.DIARIES_DIRTY = false;
            }
            proceed();
        });
    }

    private void showSplashAnimation() {
        Animation bottomToTop = AnimationUtils.loadAnimation(this,
R.anim.top_to_bottom);
        mLogo.startAnimation(bottomToTop);

        Animation fadeIn = AnimationUtils.loadAnimation(this, R.anim.fade_in);
        mainTitle.startAnimation(fadeIn);
        subTitle.startAnimation(fadeIn);

        this.preloadDiaries();
    }

    @Override
    protected void attachBaseContext(Context newBase) {
        super.attachBaseContext(ViewPumpContextWrapper.wrap(newBase));
    }
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        this.initializeWidgets();
        this.showSplashAnimation();
    }

}
```

Додаток Г – Ілюстративна частина

РОЗРОБКА МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ОРГАНІЗАЦІЇ САМОСТІЙНОЇ РОБОТИ СТУДЕНТА З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

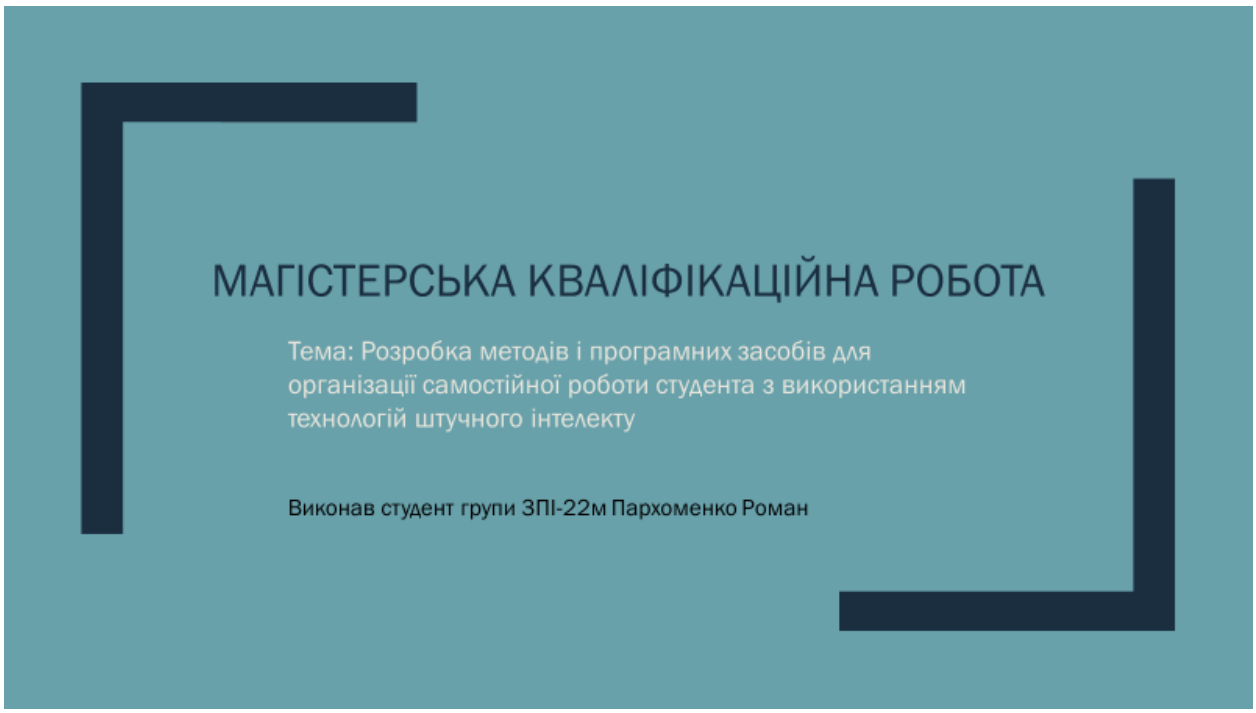


Рисунок 1 – Назва робооти

Рисунок 2 – Актуальність розробки

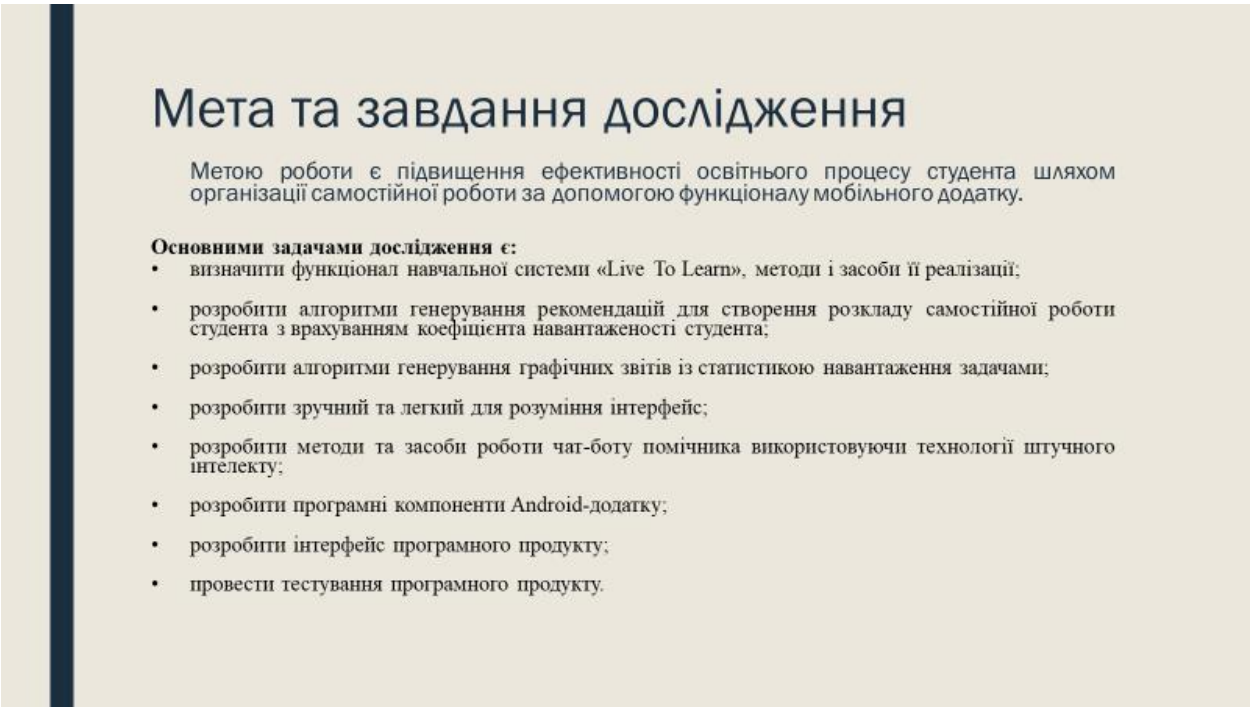


Рисунок 3 – Мета та завдання розробки

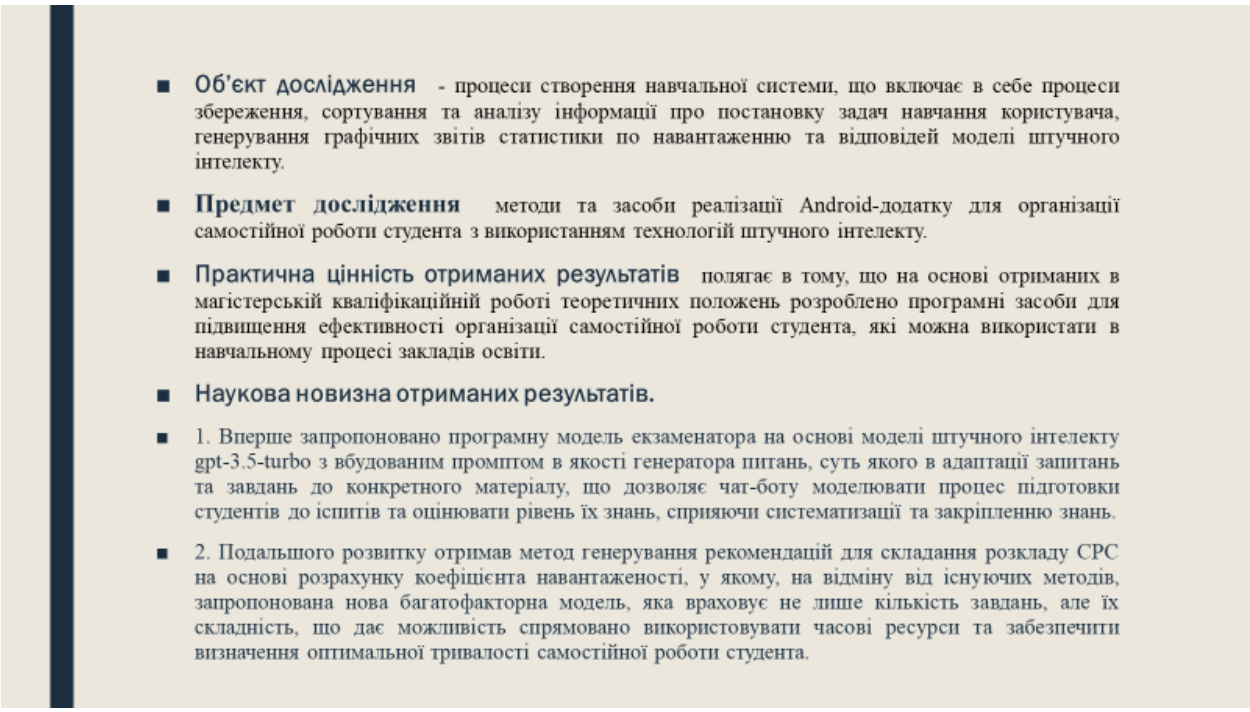


Рисунок 4 – Об`єкт і предмет дослідження, практична цінність та наукова новизна отриманих результатів

Порівняльний аналіз аналогів

Критерій	Smart Study	Vaia	Focus To-Do	JetIQ	Live To Learn
Створення власних категорій задач	+	-	-	-	+
Безкоштовність повного функціоналу	+	-	-	+	+
Використання технологій штучного інтелекту	-	+	-	-	+
Автономна робота без мережі Internet	+	-	+	-	+
Кросплатформність	+	+	+	+	-
Можливість перевірки знань за допомогою тестів	-	+	-	-	+
Підсумковий результат	4	3	2	2	5

Рисунок 5 – Порівняльний аналіз аналогів

Концептуальна схема програмної системи

Рисунок 6 – Концептуальна схема програмної системи

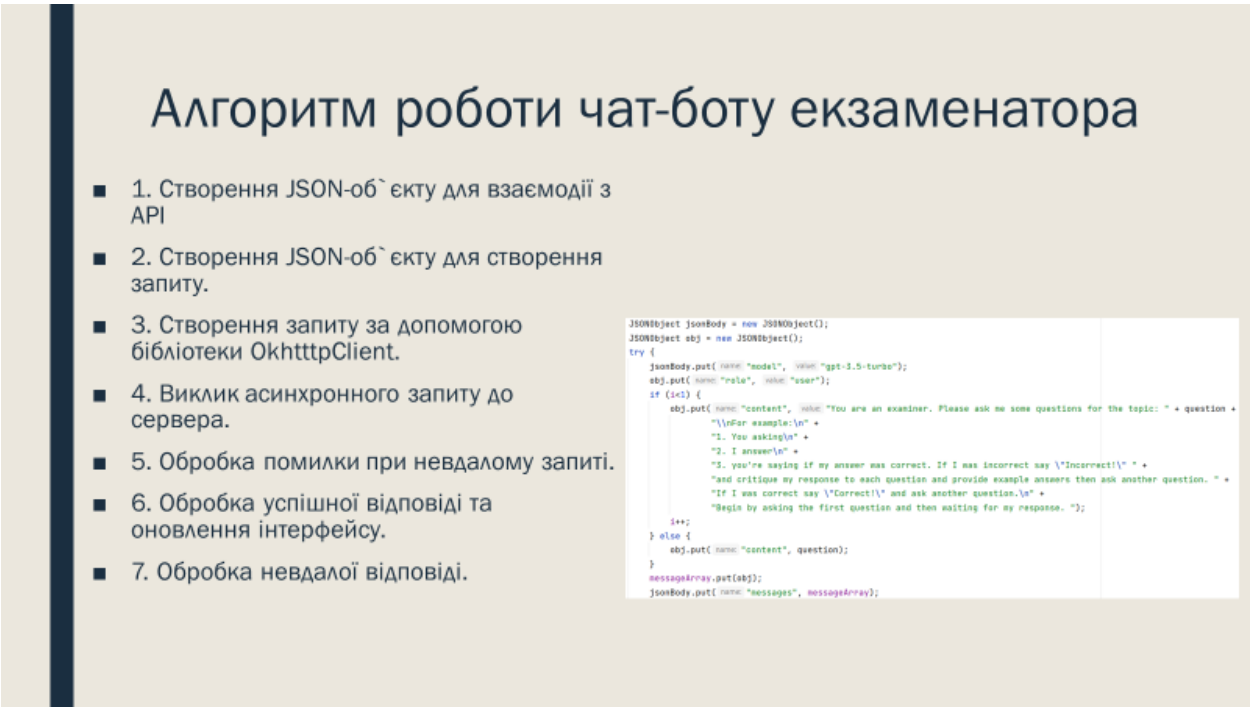


Рисунок 7 – Алгоритм роботи чат-боту екзаменатора

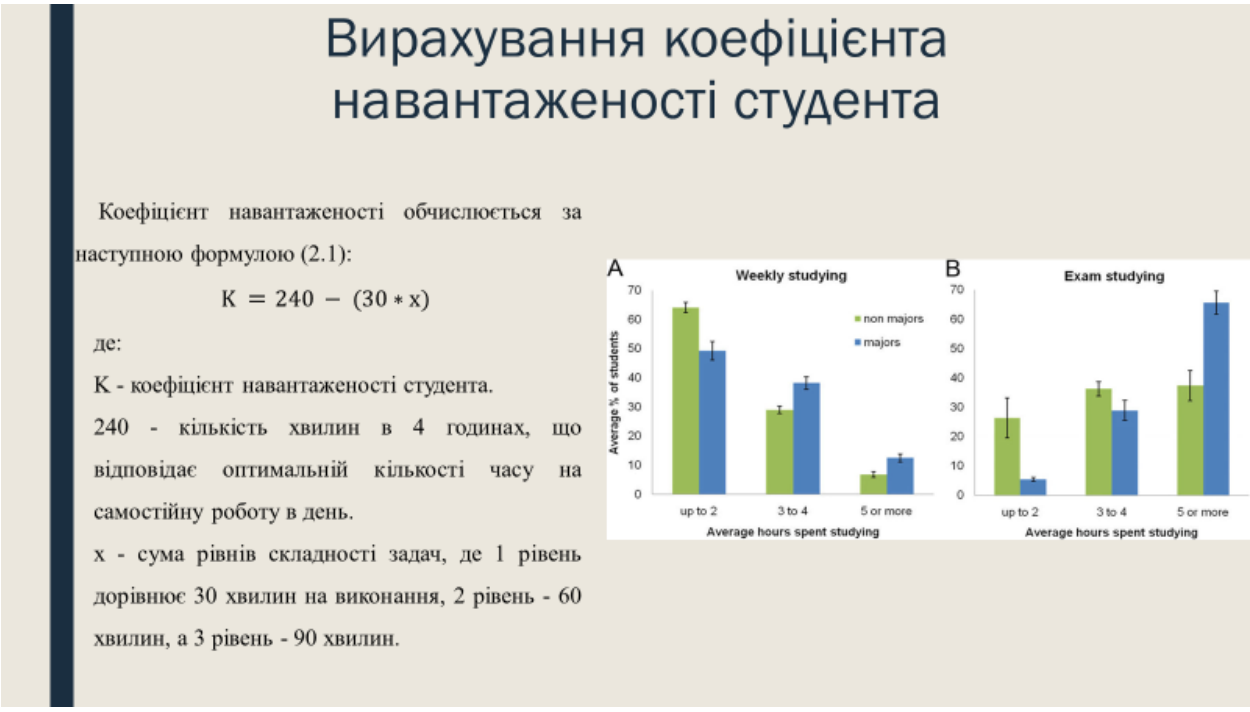


Рисунок 8 – Вирахування коефіцієнта навантаженості студента

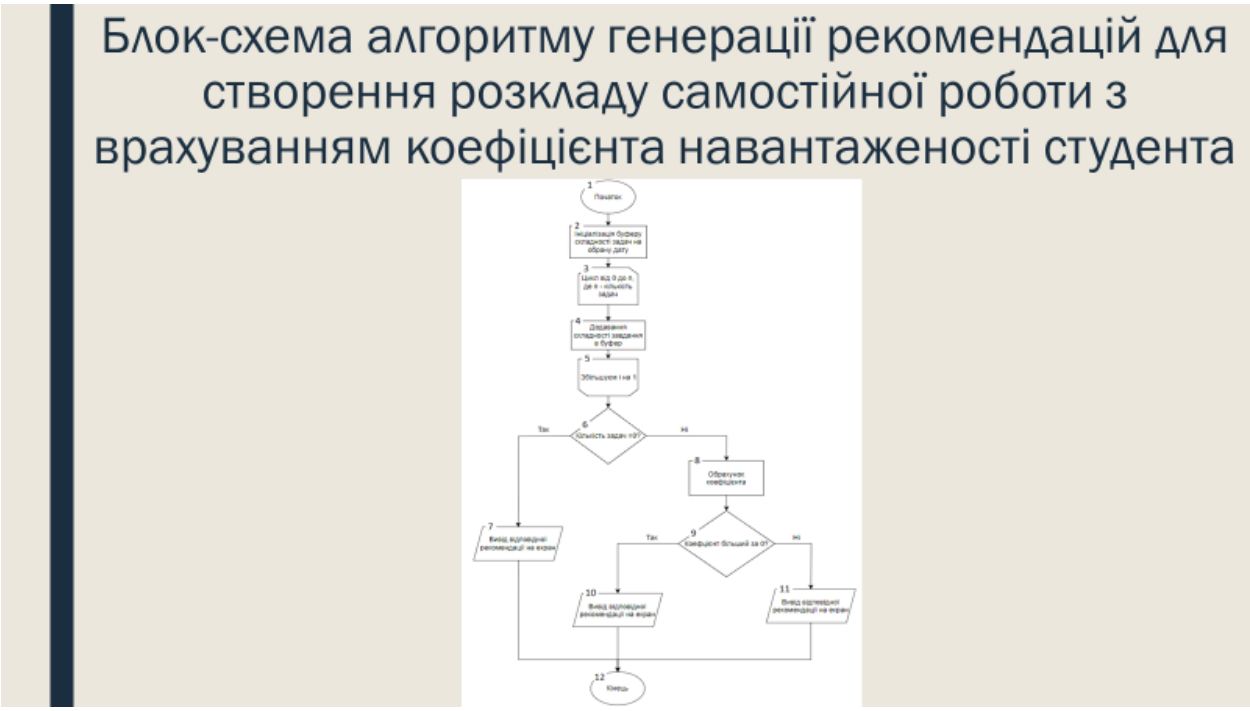


Рисунок 9 – Блок-схема алгоритму генерації рекомендацій для створення розкладу самостійної роботи з врахуванням коефіцієнта навантаженості студента

Рисунок 10 - Блок-схема алгоритму сортування задач за категоріями

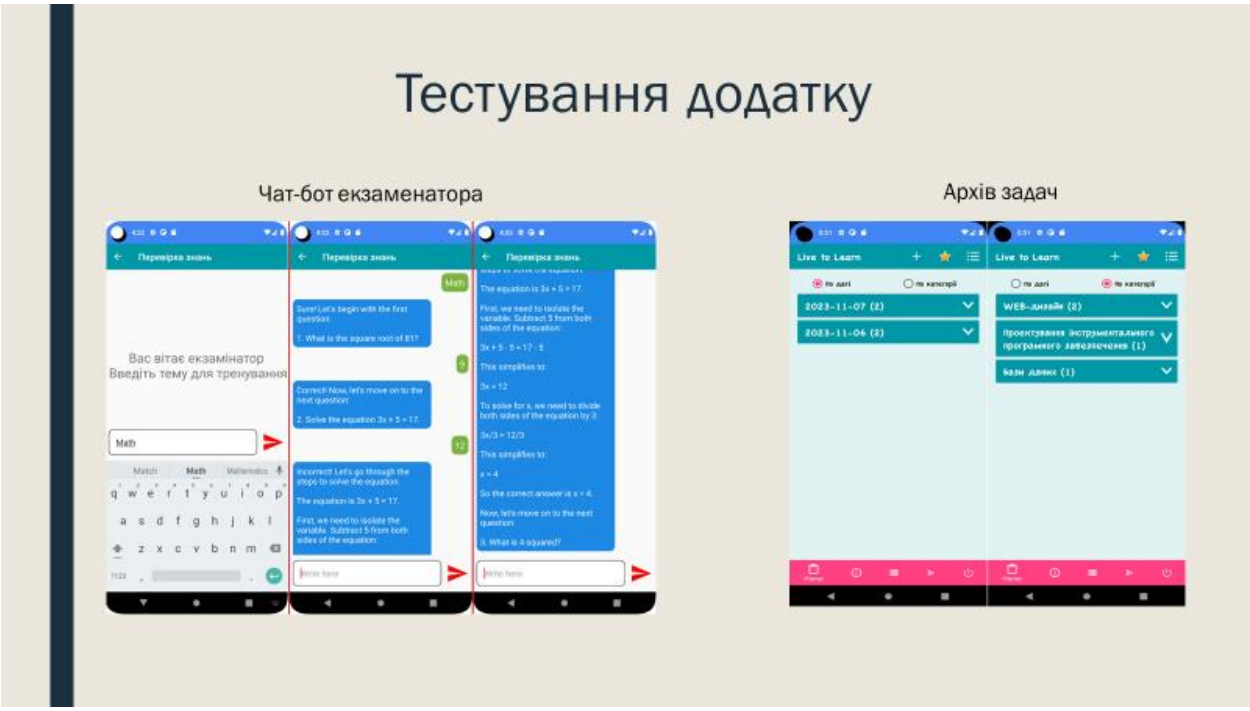


Рисунок 11 – Тестування додатку (чат-бот екзаментатора, архів задач)

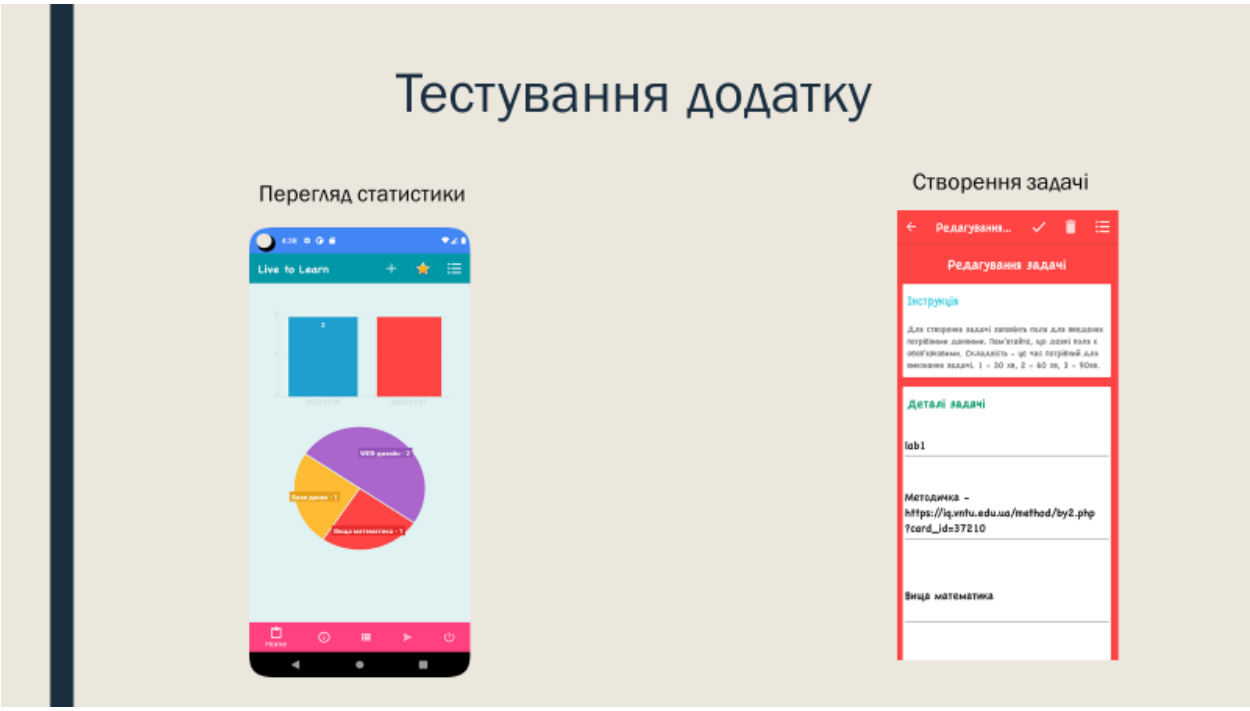


Рисунок 12 – Тестування додатку (перегляд статистики, створення задачі)

Рисунок 13 - Апробація матеріалів магістерської кваліфікаційної роботи

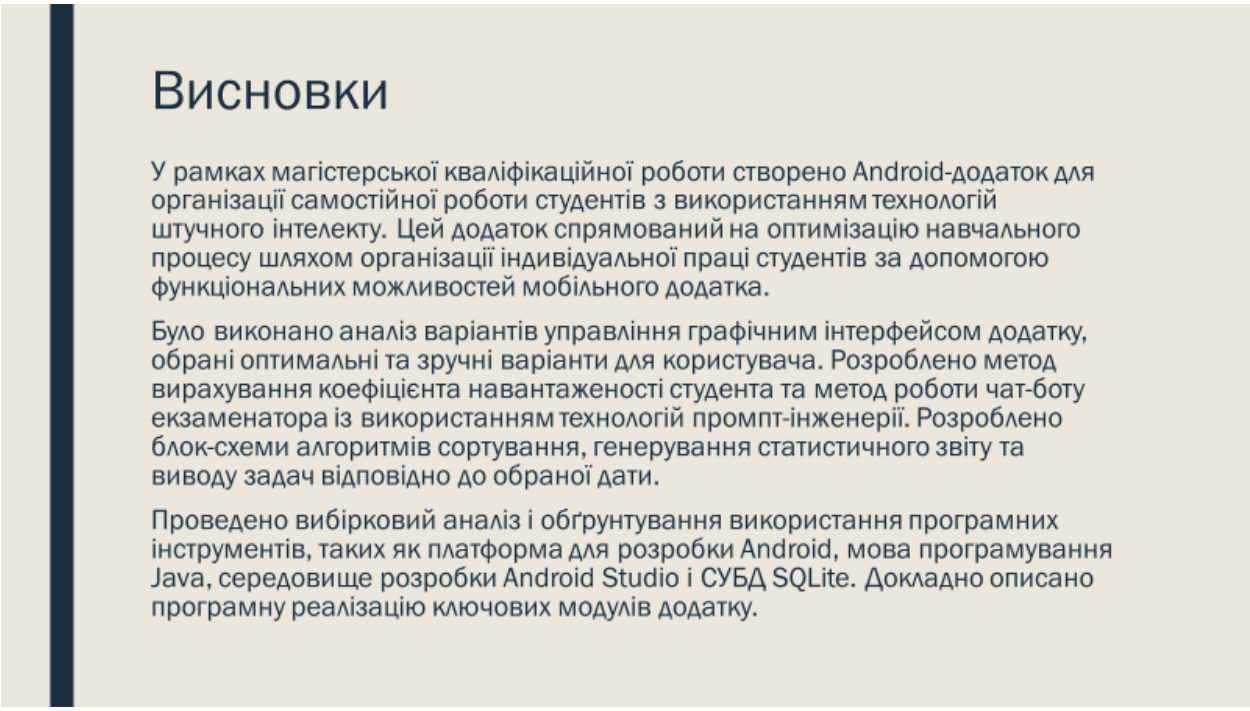


Рисунок 14 – Висновки

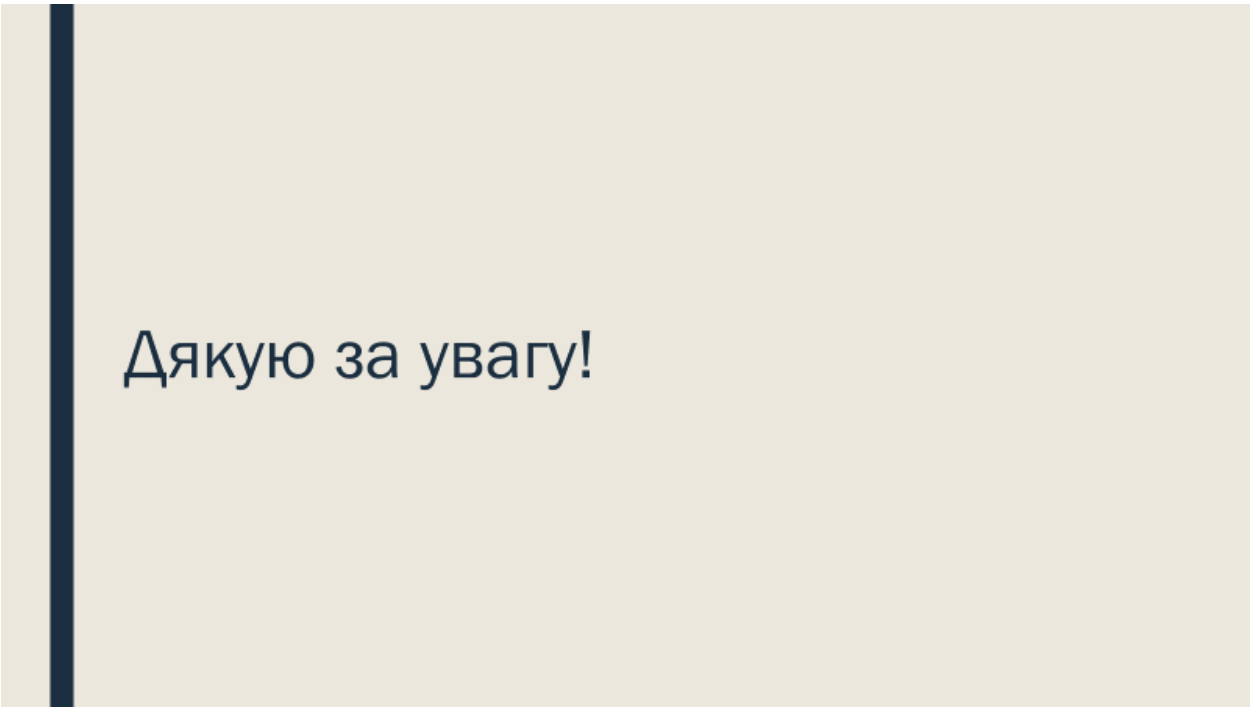


Рисунок 15 – Фінальний слайд