Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методу і програмного засобу для моніторингу діяльності персоналу»

Виконав: студент II курсу
групи ІПІ-22М спеціальності
121 – Інженерія програмного забезпечення
_____ Поліщук М.О.

Керівник: к.т.н., доцент кафедри ПЗ
_____ Бабюк Н.П.

«18» грудня 2023 р.

Опонент: к.т.н., доцент кафедри ЗІ
_____ Малиновський В.І.

«18» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
_____ д.т.н., проф. Романюк О.Н.
«18» грудня 2023 р.

ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
к.т.н., професор Романюк О. Н.
«19» вересня 2023 р.

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

Поліщуку Миколі Олеговичу

1. Тема роботи – Розробка методу і програмного засобу для моніторингу діяльності персоналу.

Керівник роботи: Бабюк Наталя Петрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. №247.

2. Строк подання студентом роботи 05.12.2023 р.

3. Вихідні дані до роботи: методи для моніторингу діяльності персоналу, блок-схеми алгоритмів роботи програми, таблиці, що містять інформацію про оцінювання продуктивності роботи персоналу.

4. Зміст текстової частини: аналіз сучасного стану питання, порівняльний аналіз аналогів; аналіз методів вирішення поставленої задачі; постановка задач на проектування, удосконалення методу моніторингу діяльності персоналу, розробка алгоритмів програмного засобу; варіантний аналіз та обґрунтування засобів для реалізації програмного засобу; розробка серверної частини програмного додатку; розробка інтерфейсу програмного засобу; тестування роботи додатку; розробка інструкції користувача; економічна частина.

5. Перелік ілюстративного матеріалу: мета роботи, об'єкт та предмет дослідження, основні задачі дослідження, порівняльний аналіз аналогів, розробка методу; алгоритм загальної роботи програми; вибір засобів для реалізації програмного засобу; реалізація методу моніторингу діяльності персоналу; демонстрація вигляду розробленої програми; економічна доцільність розробки; висновки, публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Бабюк Н.П., к.т.н., доц. каф. ПЗ	19.09.23	05.12.23
5	Причепа., к.е.н., проф. каф. ЕПВМ	20.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач дослідження.	20.09.23 30.09.23	вик
2	Розробка структури та алгоритмів програмного продукту	01.10.23 20.10.23	вик
3	Розробка модулів програмного засобу	23.10.23 05.11.23	вик
4	Тестування розробленого програмного засобу	05.11.23 10.11.23	вик
5	Економічна частина	20.11.23 01.12.23	вик

Студент (підпис) **Поліщук М.О.** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) **Бабюк Н. П.** (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Поліщук М.О. Методи і програмні засоби для моніторингу діяльності персоналу: магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 100 с.

На укр. мові. Бібліогр.: 31 назв; рис.: 30; табл. 10.

У магістерській кваліфікаційній роботі розроблено мобільний додаток для моніторингу діяльності персоналу, заявки якому користувач керує створеними завданнями та вказує час на їх виконання. Завдяки заміні методів слідкування за активним часом екрану пристрою на свободу вибору часових проміжків було отримано вищий рівень приватності та надано користувачу можливість налаштування свого власного рівня продуктивності. Індивідуальний рівень продуктивності працівника дає можливість більш точного прогнозу виконнаня робочого плану.

Розробка виконана мовою програмування Dart та її фреймворком Flutter. База даних була реалізована за допомого хмарного сервісу Firebase – Firestore. Середою розробки було обрано Visual Studio Code. Розроблений застосунок є простим та надійним у використанні, повністю задовольняючи при цьому всі висунуті до нього вимоги. Додаток був протестований, та перевірений на відсутність помилок та коректне виконання поставлених функціональних вимог, в ході тестування констатовано досягнення мети роботи.

Отримані результати можуть бути використати для продовження, розвитку та розширення створеного додатку а також створення аналогічних застосунків метою яких є моніторинг діяльності персоналу.

Ключові слова: Dart, Flutter, Authentication, Ecommerce, Firebase, Todo, API.

ANNOTATION

UDC 004.42

Polishchuk M.O. Methods and software tools for monitoring personnel activity: master's qualification thesis on specialty 121 - Software engineering, educational program - Software engineering. Vinnytsia: VNTU, 2023. 100 p.

In Ukrainian speech Bibliogr.: 31 titles; Fig.: 30; table 10.

In the master's qualification work, a mobile application was developed for monitoring personnel activity, the application of which the user manages the created tasks and specifies the time for their completion. By replacing the methods of monitoring the active screen time of the device with the freedom to choose time intervals, a higher level of privacy was obtained and the user was given the opportunity to set his own level of productivity. The individual level of employee productivity makes it possible to make a more accurate forecast of the execution of the work plan.

The development was carried out using the Dart programming language and its Flutter framework. The database was implemented using the Firebase - Firestore cloud service. Visual Studio Code was chosen as the development environment. The developed application is simple and reliable to use, fully satisfying all the requirements set for it. The application was tested and verified for the absence of errors and the correct fulfillment of the set functional requirements, during the testing it was confirmed that the goal of the work had been achieved.

The obtained results can be used for the continuation, development and expansion of the created application, as well as for the creation of similar applications whose purpose is to monitor personnel activity.

Keywords: Dart, Flutter, Authentication, Ecommerce, Firebase, Todo, API.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження.

Віддзеркалення актуальності теми у сучасному бізнес-світі, де конкуренція на ринку праці змушує компанії активно шукати інноваційні методи управління персоналом є однією з багатьох причин для вибору самі цієї теми дослідження. Розробка ефективного методу та програмного засобу для моніторингу діяльності працівників може стати важливим інструментом для оптимізації продуктивності та досягнення стратегічних цілей.

Вибір теми обумовлений сучасними тенденціями в управлінні персоналом, де зміни в робочих стандартах та зростання вимог до якості праці ставлять перед компаніями завдання розвитку ефективних методів контролю та підтримки персоналу. Впровадження високотехнологічних інструментів, які дозволяють моніторити діяльність персоналу, може сприяти адаптації до сучасних викликів та збереженню конкурентної переваги.

Обрана тема визначається відсутністю комплексного підходу до моніторингу діяльності персоналу в більшості компаній. Якщо існує відчуття, що наразі відсутній одиночний метод чи програмний засіб, який би враховував всі аспекти діяльності персоналу, це стає стимулом для проведення дослідження та розробки новаторських рішень у цій області.

Загалом, вибір теми дослідження обумовлений потребою висвітлити та вирішити актуальні завдання управління персоналом, використовуючи сучасні технології та методи, що має потенційно значущі практичні вигоди для більшості підприємств.

Зв’язок роботи з науковими програмами, планами, темами.

Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження.

Метою магістерської кваліфікаційної роботи є розробка мобільного

додатку для моніторингу діяльності персоналу у групах працівників, що дасть можливість оптимізувати час та результати виконуваних цілей та задач для працівників. А також покращить можливість керівництва планувати та розподіляти робоче навантаження між працівниками, зважаючи на реальні результати вимірювання часу, що йде на виконання поствлених завдань.

Основними задачами дослідження є:

– аналіз предметної області, ключових термінів та понять;

– удосконалення методу моніторингу діяльності;

– розробка алгоритмів програмного засобу для моніторингу діяльності персоналу;

– розробка простого та зрозумілого інтерфейсу для користувача;

– реалізація методу та програмногозасобу для моніторингу діяльності персоналу;

– тестування створеного програмного продукту.

Об’єкт дослідження – комплексне вивчення та аналіз процесів, пов'язаних з ефективним впровадженням інноваційних методів та програмного забезпечення для моніторингу роботи працівників в корпоративному середовищі. Основним об'єктом дослідження є діяльність персоналу, включаючи різноманітні аспекти, такі як робочий час, продуктивність, виконання завдань та взаємодія між співробітниками а також процес розробки програмного засобу та реалізація методу для моніторингу роботи працівників із використанням мови програмування Dart та її фреймворку Flutter. Для реалізації серверної частини та аутентифікації працівників передбачено використання сервісів Firebase.

Предметом дослідження є системний аналіз і вдосконалення існуючих методів та програмного забезпечення, спрямованих на ефективний моніторинг та управління діяльністю персоналу в сучасних корпоративних умовах. Предметом дослідження є комплексні аспекти, пов'язані з організацією робочого часу, оцінкою продуктивності, аналізом виконання завдань, взаємодією між співробітниками та іншими параметрами, що визначають ефективність роботи

персоналу.

Методи дослідження. У процесі досліджень використовувались різноманітні методи дослідження для отримання обґрунтованих та цілеспрямованих результатів:

- Ретельний огляд наукових статей, книг та інших джерел для засвоєння існуючих методів та програмних засобів моніторингу персоналу, аналіз тенденцій у цій галузі та ідентифікація прогалин для подальшого дослідження.
- Анкетування та опитування працівників і керівників для отримання відгуків щодо існуючих методів контролю та їх ефективності, а також для виявлення потреб та вимог до нових програмних засобів.
- Впровадження нових методів та програмного забезпечення в обраній компанії для оцінки їхньої працездатності та впливу на діяльність персоналу. Збір даних перед, під час та після впровадження для порівняльного аналізу.
- Аналіз успішних випадків впровадження або вдосконалення систем моніторингу в інших організаціях з подальшим висвітленням кращих практик та можливих стратегій.
- Консультації з експертами в галузі управління персоналом, програмування та інформаційних технологій для отримання цінних порад та рекомендацій.

Наукова новизна отриманих результатів:

Подальшого розвитку набув метод моніторингу виконання задач персоналом, що виявляє проблемні завдання, який мінімізує ризики при реалізації проекту і завдяки якому пришвидшується процес огляду та виявлення взаємопов'язаних задач і їх тематики.

Практична цінність отриманих результатів. Практичне значення магістерської кваліфікаційної роботи полягає у використанні розробленого програмного продукту для моніторингу діяльності членів групи працівників

керіництвом, що забезпечить оптимізацію планів навантаження, що в свою чергу покращує ставлення працівників до своїх обов’язків та допомагає підвищити ефективність виконання робочих завдань не впливаючи негативно на психологічний стан виконавців.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. Автору належать такі результати: розробка алгоритмів та візуальної частини додатку, а також інтеграція сервісів Firebase для авторизації користувачів та зберігання даних у хмарному сервісі Firestore.

Апробація матеріалів магістерської кваліфікаційної роботи.

Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Міжнародній науково-практичній інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ» (МН-2024)

Публікації. Основні результати досліджень опубліковано в 1 науковій праці, що входить у матеріали Міжнародної науково-практичної інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ» (МН-2024) [1].

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз сучасного стану питання

Сучасні комерційні проекти неможливі без взаємозв’язку великої кількості спеціалізованих відділів працівників та їх персоналу між собою. Для більшого розуміння актуальності проведення дослідження та розробки на тему моніторингу діяльності робітників необхідно провести аналіз вже існуючих методів, що дають змогу оцінювати та покращувати поточні результати продуктивності та успішності.

На даний момент виділяють 7 основних методів моніторингу, що дають можливість проводити більш об’єктивну та різнорівневу оцінку виконання поставлених завдань:

1. Системи контролю робочого часу (Time Tracking Systems):

Системи такого типу використовуються для реєстрації робочого часу працівників. Електронні інструменти та програми дозволяють точно фіксувати час, проведений працівниками над конкретними завданнями чи проектами, що дає можливість об’єктивно оцінити час, витрачений на винання задач різного типу та складності.

2. Системи моніторингу продуктивності (Productivity Monitoring Systems):

Такі системи спрямовані на вимірювання та аналіз ефективності виконання завдань працівниками. Включають у себе відстеження використання програм, кількість виконаних завдань та обсяг продуктивної роботи.

3. Аналіз використання комп'ютера (Computer Usage Analysis):

Дають можливість реалізувати спостереження за активністю на комп'ютері працівника, включаючи використання програм, часову активність, перерви та час, витрачений на різні завдання. Такі рішення завжди сприймаються негативно

працівниками як штатного колективу так і найманими працівниками поза штатом, через постійний жорсткий контроль з боку роботодаця.

4. Системи електронного моніторингу працівників (Employee Monitoring Systems):

Такі системи включають широкий спектр інструментів, таких як відеоспостереження, моніторинг комунікацій, відслідковування робочих процесів тощо. Застосовуються для отримання детальної інформації про робочу діяльність.

5. Оцінка завдань та ключових показників продуктивності (Task and KPI Evaluation):

Є одними з найзастосованіших та застосовується для визначення та вимірювання результативності працівників шляхом визначення ключових показників продуктивності та виконання конкретних завдань.

6. Системи звітності та зналізу (Reporting and Analytics Systems):

Використовуються для збору та обробки даних з метою створення звітів та аналізу результатів моніторингу, що допомагає приймати управлінські рішення. В більшості випадків икористовуються з методами оцінки завдань та ключових показників продуктивності для більш точного та детального аналізу отриманих результатів.

7. Використання інструментів штучного інтелекту (AI-powered Monitoring Tools):

Сучасні інструменти використовують алгоритми штучного інтелекту для аналізу та передбачення продуктивності, а також для розпізнавання патернів поведінки робітників.

Кожен з цих методів має свої переваги та обмеження, і їхнє використання повинно бути обґрунтоване конкретними цілями моніторингу. Важливим є не лише вибір конкретних методів моніторингу, але й збалансоване врахування прав та інтересів працівників, забезпечення конфіденційності та дотримання етичних норм. Через надмірний контроль працівники втрачають довіру та

лояльність до керівництва а з ними і бажання працювати ефективно, розуміючи, що при підвищенні рівня виконання поставлених планів збільшиться не обсяг заохочень за плідну працю а лише поставлений план задач для виконання.

1.2 Порівняльний аналіз аналогів

Для перевірки актуальності розробки було проведено порівняльний аналіз найпопулярніших додатків для моніторингу роботи персоналу.

Оскільки монітринг окремої групи працівників є розповсюдженою потребою, продуктові компанії пропонують свої рішення. На момент проведення порівняння аналогів вже існує багато мобільних додатків для моніторингу діяльності персоналу, які дозволяють керуючому персоналу відстежувати робочий час, продуктивність та інші показники.

В якості аналогів було розглянуто такі продукти:

1. Hubstuff
2. Clockify
3. DeskTime

Через велику кількість функціональних можливостей та відмінностей в використовуваних методах моніторингу, обраних для порівняння аналогів було вирішено навести по декілька позитивних та негативних сторін кожного з додатків для більш об’єктивного порівняння.

Hubstuff - Одна програма для автоматизації процесів відстеження робочого часу, управління робочою силою та показників продуктивності. Використовується віддаленими та виїзними працівниками, міжнародними командами та фрілансерами в понад 15 галузях, щоб відстежувати час, отримувати докази роботи, керувати графіками та автоматизувати нарахування заробітної плати [2].

Позитивні сторони додатку Hubstaff:

- Комплексність та багатофункціональність:

Hubstaff пропонує широкий спектр функцій, таких як відстеження робочого часу, скріншоти екрану, GPS-відстеження та аналіз продуктивності. Це дозволяє комплексно відслідковувати та оцінювати робочу діяльність персоналу.

- Гнучкість та адаптивність:

Hubstaff дозволяє вам вибирати тільки ті функції, які необхідні для конкретних потреб вашої організації. Гнучкі налаштування роблять його придатним для різноманітних типів бізнесу та розмірів команд.

- Інтеграція та сумісність:

Hubstaff має можливість інтеграції з багатьма іншими інструментами та сервісами, що спрощує обмін даними та підвищує ефективність управління. Вона підтримує інтеграцію з популярними проектними та бухгалтерськими системами.

Негативні сторони додатку Hubstaff:

- Високі витрати для великих команд:

Для великих команд або підприємств, вартість використання Hubstaff може бути високою. Особливо з урахуванням додаткових функцій та користувальницьких ліцензій, що може впливати на бюджет.

- Складність для неініційованих користувачів:

Для деяких користувачів може виникнути складність у використанні всіх функцій Hubstaff через їхню обширність. Це може вимагати часу на навчання та адаптацію.

- Можливі питання конфіденційності:

Системи моніторингу, такі як відстеження екрану та скріншоти, можуть викликати питання щодо конфіденційності та приватності працівників. Це може впливати на робочу атмосферу та відносини у колективі.

На рисунку 1.1 зображено інтерфейс додатку Hubstaff.

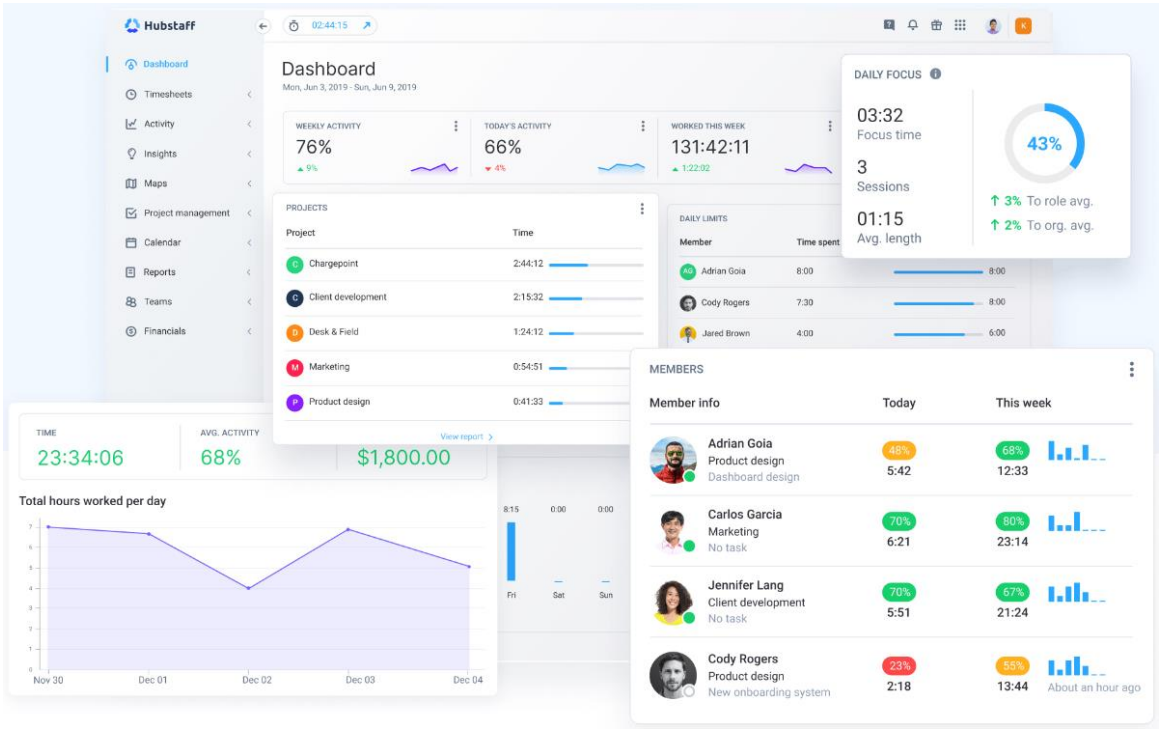

Рисунок 1.1 – Користувацький інтерфейс додатку Hubstuff

Clockify - програмне забезпечення для відстеження часу, яким користуються мільйони людей, це програма для відстеження часу та табелів, яка дозволяє відстежувати робочі години в проектах, має необмежену кількість користувачів та є повністю безкоштовною. Дає можливість відстежувати продуктивність, відвідуваність і оплачувані години за допомогою простого відстеження часу та табеля[3].

Позитивні сторони додатку Clockify:

- Безкоштовна версія з широким функціоналом:

Clockify пропонує безкоштовну версію, яка включає в себе важливі функції, такі як відстеження робочого часу, звіти та категорії проектів. Це дозволяє невеликим компаніям та фрілансерам ефективно використовувати додаток без значних витрат.

- Простий та інтуїтивно зрозумілий інтерфейс:

Clockify володіє простим та легким у використанні інтерфейсом. Його інтуїтивно зрозумілі функції дозволяють користувачам швидко розпочати відстеження робочого часу та створювати звіти без додаткового навчання.

- Можливість інтеграції та розширення функціоналу:

Clockify підтримує інтеграцію з іншими популярними інструментами та сервісами, що полегшує обмін даними та забезпечує зручність в роботі. Також існують розширення для браузерів, що дозволяє зручно взаємодіяти з додатком.

Негативні сторони додатку Clockify:

- Обмеження безкоштовної версії:

Безкоштовна версія Clockify має свої обмеження, такі як обмежена кількість користувачів та проєктів. Для більших команд або проектів може знадобитися перехід до платної версії.

- Відсутність розширених функцій в безкоштовній версії:

Деякі розширені функції, такі як автоматичний відлік часу та додаткові звіти, доступні лише в платних версіях. Це може обмежити можливості користувачів без оплати підписки.

- Можливі проблеми із синхронізацією та затримками:

Деякі користувачі повідомляли про можливі проблеми із синхронізацією та затримками у поновленні даних. Це може вплинути на точність відстеження робочого часу. На рисунку 1.2 зображено інтерфейс додатку Clockify.

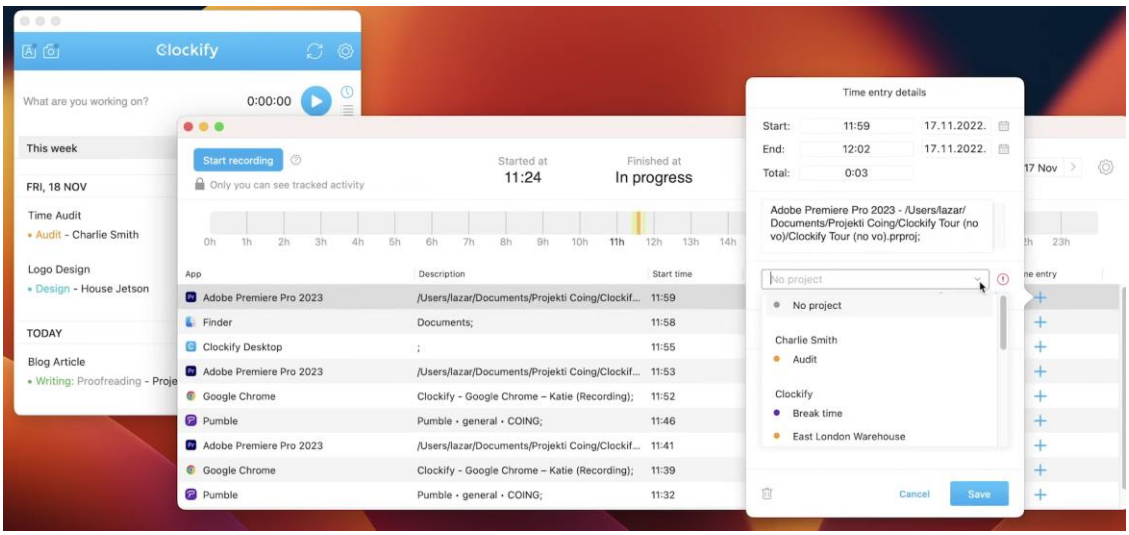

Рисунок 1.2 – Користувацький інтерфейс додатку Clockify

DeskTime - додаток для відстеження часу, це повнофункціональний реєстратор продуктивності, який дозволяє швидко та легко відстежувати власний робочий час або робочий час співробітників. Завдяки двом простим варіантам під час роботи далеко від комп’ютера, відстеження часу на основі проекту та відстеження часу вручну, ви обов’язково зафіксуєте кожну важливу хвилину робочого дня. [4].

Позитивні сторони додатку DeskTime:

- Автоматичне відстеження активності:

DeskTime пропонує автоматичне відстеження активності користувачів, включаючи час активності на комп'ютері та перерви. Це дозволяє точно вимірювати час, витрачений на роботу та відпочинок.

- Детальні звіти та аналіз продуктивності:

DeskTime надає розгорнуті звіти та аналіз продуктивності, що включає в себе інформацію про активність, відвідані сайти, застосунки та інше. Це дозволяє керівникам отримувати інсайти в робочі звички та ефективність персоналу.

- Опції керування проектами та завданнями:

DeskTime має вбудовані інструменти для керування проектами та завданнями, що дозволяє ефективно організовувати робочий процес. Функції, такі як розподіл часу на проекти та завдання, полегшують відстеження прогресу та витрати часу.

Негативні сторони додатку DeskTime:

- Високі витрати для більших команд:

Для великих компаній або команд, вартість використання DeskTime може бути високою, особливо при необхідності розширених функцій та додаткових ліцензій.

- Потенційні питання приватності:

Системи відстеження, особливо з детальним аналізом відвіданих сайтів та застосунків, можуть викликати питання щодо приватності та довіри серед працівників.

- Обмежена гнучкість для роботи з дому:

DeskTime може бути менш гнучким у відстеженні робочого часу для тих, хто працює з дому. Особливо, якщо співробітники використовують різні пристрої або розташовані в різних місцях. На рисунку 1.3 зображено інтерфейс додатку DeskTime

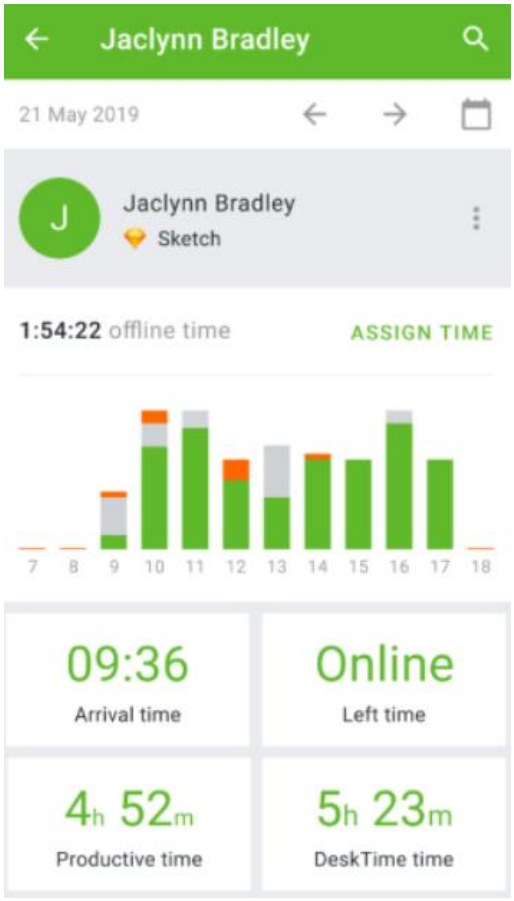

Рисунок 1.3 – Користувацький інтерфейс додатку DeskTime

Отже, провівши порівняння додатків наведених вище, було створено таблицю порівняння аналогів - таблиця 1.1. У вказаній нижче таблиці вказано результати порівняння аналогів за найважливішими для розроблюваного додатку.

Таблиця 1.1 – Порівняння аналогів

Критерій	Hubstuff	Clockify	DeskTime	ActiveTime
Детальні звіти	+	+	+	+
Простий у використанні	-	-	-	+
Повністю безкоштовний	-	-	-	+
Відсутність вбудованої реклами	+	+	+	+
Безкоштовна версія	-	+/-	+/-	+
Інтеграція	+	+	+	-
Приватність та довіра	-	-	-	+

Під час проведення порівняльного аналізу аналогів, було прийнято до уваги саме основні критерії які впливають на продуктивність та зручність використання.

Головним критерієм для порівняння було обрано можливість формувати детальні звіти. Усі проаналізовані додатки, мають за основну мету надати користувачу можливість проаналізувати час, проведений за виконанням різних завдань та зібрати статистику чи вивести звітність.

Також до уваги було взято критерій простоти у використанні. Цей критерій є не менш важливим, оскільки час на підготовку до роботи залежить напряму від складності додатку. Складність використання додатку залежить прямопропорційно до кількості функціоналу та реалізованих можливостей. Щоб

збільшити діапазон цільової аудиторії можна поступитись частиною другорядних можливостей, що в свою чергу зменшить час інтеграції нової системи у використання працівниками.

Критерії, що великою мірою впливають на кількість завантажень та бажання користуватись додатком це приватність та довіра до керівництва. Реалізація функціоналу для фото/відео фіксації екрану працівників та доступ до пристрою в режимі реального часу взагалі не залишає приватності під час роботи, що негативно впливає на довіру до керівництва та знижує бажання працювати якісно в комфортному темпі.

Наявність реклами та додаткового платного контенту також є негативним фактором при проведенні аналізу додатку. При поділі додатку на платну та безкоштовну версії та, що не приносить розробнику прибутку, стає не ефективною для користувача, що змушує до придбання повного видання або ж до пошуку альтернатив.

Отже, відповідно до таблиці порівняння характеристик, розробка власного програмного продукту є доцільною. В результаті розробки буде отримано програмний продукт, який анулює всі перераховані недоліки існуючих рішень і забезпечить більш простий та ефективний спосіб виконання поставлених задач та цілей для моніторингу діяльності.

1.3. Аналіз методів вирішення поставленої задачі

Після детального аналізу конкуруючих рішень та зібраних даних було вирішено утриматися від впровадження функцій, які могли б підірвати довіру та відчуття конфіденційності серед керівництва. Зосереджуючись на головному завданні, було прийнято рішення розробити систему, що надає працівникам можливість самостійно додавати інформацію про свої завдання, вкладати деталізований опис та вказувати очікуваний час на їх виконання. Це сприяє

більш точній оцінці власної продуктивності та мотивує співробітників краще слідкувати за якістю виконання своєї роботи.

З метою поліпшення комфорту користувачів, в розробці було зібрано ряд корисних інструментів. Серед них варіативні кольорові схеми оформлення та можливість перекладу інтерфейсу з української на англійську мову. Не менш важливим елементом стала візуалізація розподілу часу між різними завданнями через застосування кругових діаграм. Ця функціональність допоможе зрозуміти, які завдання є найбільш пріоритетними та вимагають більше уваги.

1.4 Постановка задач на проектування

Магістерська кваліфікаційна робота присвячена розробці програмного засобу для моніторингу діяльності персоналу: його алгоритмів і інтерфейсу. Для його інтерфейсу було висунуто такі вимоги:

- Мінімалістичний і зручний в користуванні;
- Лаконічний дизайн з помірною кольоровою гамою;
- Наявність кругових діаграм для спрощення сприйняття обсягу навантаження;

Для алгоритмів додатку було висунуто такі вимоги:

- Розробити алгоритм створення задачі;
- Розробити алгоритм збереження задачі;
- Розробити алгоритм редагування задачі;
- Розробити алгоритм видалення задачі;
- Розробити програмний продукт, призначений для вирішення поставлених проблем;
- Провести тестування програмного продукту.

1.5 Висновки

Проведений глибокий аналіз сучасного стану проблематики підтвердив актуальність обраного дослідження. У рамках дослідження був проведений детальний порівняльний аналіз вже існуючих аналогів розглядуваного продукту з метою виявлення можливостей для його оптимізації та удосконалення для подальшої ефективної комерційної реалізації. Після вивчення сильних та слабких сторін розглядуваних аналогів, було виявлено ключові аспекти, які можуть бути вдосконалені в новому продукті. Ці відомості використані для розробки докладних функціональних та нефункціональних вимог до майбутнього продукту. Крім того, були сформульовані конкретні завдання та методологія для ефективного проектування нового програмного продукту.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз стану галузі моніторингу праці

У сучасному світі галузь моніторингу діяльності персоналу швидко розвивається, пристосовуючись до нових викликів та інновацій. Однією з основних тенденцій є розширення функціональності програмних засобів для моніторингу, які стають більш комплексними та виваженими. Останні роки спостерігається важливий крок у напрямку використання штучного інтелекту (ШІ) у розробці програм для моніторингу, що дозволяє аналізувати великі обсяги даних та надавати корисні прогнози стосовно продуктивності та ефективності роботи персоналу[5].

Штучний інтелект (artificial intelligence, AI) — це метод змусити комп'ютер чи програмне забезпечення «мислити» як людський мозок. Це досягається шляхом вивчення закономірностей роботи людського мозку та аналізу когнітивних процесів. Результатом цих досліджень є розробка інтелектуального програмного забезпечення та систем[6].

Використання штучного інтелекту (ШІ) для моніторингу продуктивності персоналу представляє інноваційний підхід, який може суттєво поліпшити управління та ефективність команди. Однією з ключових тенденцій є розширення функціональності програмних засобів, де ШІ використовується для аналізу великої кількості даних, таких як робочий час, завдання, та надає прогнози щодо продуктивності. Це дозволяє не лише відстежувати, але і передбачати результативність працівників.

Додатково, ШІ може надавати персоналізовані рекомендації, враховуючи індивідуальні особливості кожного працівника. Автоматизація рутинних завдань, таких як обробка даних та відповіді на типові запитання, дозволяє звільнити час для більш творчих завдань. ШІ також може виявляти точки

оптимізації в робочих процесах, аналізуючи взаємодію працівників та вплив різних факторів.

Ще однією ключовою тенденцією є активний розвиток мобільних додатків для моніторингу діяльності. Застосунки стають все більш доступними та мобільними, надаючи можливість відстежувати робочий час та продуктивність з будь-якого мобільного пристрою, що важливо для сучасних команд, які працюють з різних місць.

У контексті конфіденційності та приватності спостерігається збільшення уваги до етичних аспектів використання інструментів моніторингу. Компанії шукають баланс між необхідністю контролю та захистом приватності працівників, враховуючи сучасні стандарти та законодавство. З врахуванням етичних аспектів, використання ШІ в моніторингу продуктивності повинно бути збалансованим, з урахуванням конфіденційності та захисту приватності працівників. Загалом, цей підхід відкриває широкі можливості для оптимізації та підвищення ефективності робочих процесів у сучасному бізнес-середовищі.

Однак разом із зростанням можливостей таких систем, виникають нові виклики. Питання конфіденційності, етики використання та вирішення проблем щодо приватності визначаються як важливі пункти у галузі моніторингу діяльності персоналу. Відсутність чіткого юридичного регулювання може призвести до спірних питань та непорозумінь між роботодавцем та працівником масштабів цілої компанії.

У контексті моніторингу діяльності персоналу, майнинг може бути дуже корисним та перспективним інструментом.

Технології Text Mining - набір методів, призначених для здобуття інформації з текстів на основі сучасних ІКТ, що дає змогу виявити закономірності, які можуть приводити до отримання корисної інформації і нових знань користувачами. Це інструмент, який дає можливість аналізувати великі обсяги інформації у пошуках тенденцій, шаблонів і взаємозв'язків, здатних допомогти у прийнятті стратегічних рішень[7].

Слід відзначити, що аналіз тексту знаходиться на перетині розвитку двох важливих напрямків: аналізу здобуття даних та методів обробки текстів. Історично розвиток технологій для аналізу даних відбувався паралельно з розробкою інструментів для текстового майнингу. Такі методології, як класифікація та кластеризація, які мали своє походження в аналізі здобуття даних, стали неодмінною частиною Text Mining. У свою чергу, Text Mining розширив можливості аналізу тексту: він став можливим для автоматичного реферування текстів, виявлення феноменів, а також понять і фактів в текстах[7]. Сучасні системи Text Mining дозволяють виявляти та аналізувати шаблони у текстах, здійснювати автоматизований розподіл інформації за різними профілями та створювати комплексні огляди документів. Таким чином, Text Mining відкриває нові перспективи для семантичного пошуку документів та ефективного аналізу текстової інформації.

З метою більш чіткого розуміння, можна розподілити Text Mining на основні елементи:

- Процес Feature (Entity) Extraction - відноситься до вилучення з текстових матеріалів конкретних слів або їх комбінацій, які з точки зору користувача є ключовими для характеризації змісту документа. Серед таких елементів можна виділити інформацію про осіб, організації, географічні локації, терміни ПрО та інші важливі фрази. Якщо говорити про Feature (Entity) Association Extraction, то тут йдеться про більш складні сукупності слів, які технологічно об'єднані і можуть відображати глибший контекст або взаємозв'язок між різними елементами тексту;
- Автоматичне реферування або анотування (Summarization) - відноситься до процесу створення стислого огляду або конденсованої версії документа на основі його повного текстового вмісту;

- Класифікація (Classification) - використовує статистичні методи та кореляції для формування специфічних правил, які дозволяють системі ефективно розподіляти документи до певних попередньо визначених категорій або груп;
- Кластеризація (Clustering) - базується на характеристиках документів і використовує комбінацію лінгвістичних та математичних підходів, не обмежуючись заздалегідь встановленими категоріями чи групами;
- Відповіді на питання (question answering) - автоматичне виділення та надання конкретних відповідей на запитання, базуючись на аналізі вхідного тексту та взаємодії з базою даних чи іншими джерелами інформації;
- Тематичне індексування тексту - відноситься до процесу призначення ключових тем або категорій до документів або їх частин. Цей підхід дозволяє структурувати і організовувати великі обсяги інформації, спрощуючи навігацію та пошук конкретних документів за їх тематикою;
- Пошук за ключовими словами відноситься до методу пошуку інформації, де користувач вводить конкретні терміни або фрази, які потім використовуються для виявлення відповідних документів або вмісту. Цей метод є одним з найпоширеніших і ефективних способів пошуку в інформаційних системах, дозволяючи швидко локалізувати релевантну інформацію серед великої кількості даних.
- Побудова семантичної мережі, або аналіз зв'язків, відноситься до процесу, який створює візуальне представлення взаємозв'язків між різними елементами документа. Цей метод виявляє та аналізує ключові фрази або дескриптори в тексті, а також встановлює, як ці елементи пов'язані між собою. Він може розпізнавати різноманітні сутності, факти, події та інші аспекти тексту, дозволяючи користувачам ефективно шукати та навігувати в масиві інформації. Цей підхід є високорівневим і дозволяє глибоко аналізувати та розуміти зміст документів, витягаючи з них найважливішу інформацію.

Провевши огляд технології Text Mining та проаналізувавши основні елементи данної технології можна стверждувати, що майнінг, або видобуток даних, представляє собою процес автоматизованого виявлення цінної інформації або зв'язків в великих наборах даних. Зокрема, використання майнингу дозволяє ефективно аналізувати великі обсяги даних, зібраних про діяльність персоналу. Це може включати в себе аналіз часових рядів робочих годин, оцінок продуктивності, відгуків колег, а також використання різних інструментів для класифікації та кластеризації даних. Завдяки майнингу можна виявити складні зв'язки та закономірності, які не завжди очевидні при поверхневому аналізі. Крім того, використання майнингу може допомогти у виявленні аномалій або незвичайних патернів у діяльності персоналу, що є корисним для розробки програмного засобу моніторингу. Таким чином, майнинг даних в контексті даної роботи може відігравати ключову роль у глибокому аналізі діяльності персоналу та створенні ефективних інструментів для її моніторингу та управління.

Всі ці тенденції формують основу для подальших досліджень у магістерській роботі, відкриваючи можливість розглядати сучасні виклики та шукати новаторські рішення з метою покращення систем моніторингу та управління персоналом.

2.2 Розробка структури інтерфейсу для мобільного додатку

Дизайн інтерфейсу відіграє ключову роль у взаємодії користувача з додатком, тому його створення потребує глибокого розуміння та уваги до деталей. Комфорт та ефективність використання додатку в значній мірі залежать від того, наскільки інтуїтивно зрозумілим і доступним є його інтерфейс. Ключовий етап цього процесу полягає у визначенні конкретних функцій та можливостей, які мають бути відображені в додатку. Після цього вибір кольорової палітри стає не менш важливим аспектом, оскільки кольори можуть

впливати на сприйняття та настрій користувача, а також відображати та підкреслювати конкретні функціональні можливості додатку.

Теорія кольору є практичним злиттям мистецтва і науки, використовується для визначення гармонійного поєднання різних кольорів. Основу теорії складає Коло кольорів, де можна обрати будь-які кольори, і їх поєднання вважається практично ідеальним[8].

Зручність використання програмного продукту відіграє важливу роль у його популярності. Термін "юзабіліті" відноситься до концепції розробки інтерфейсів програмного забезпечення, яка ставить за мету досягнення максимальної психологічної та зорової зручності для користувача. За стандартами ISO, використовність визначається як "ступінь, з якою продукт може бути використаний визначеними користувачами для досягнення визначених цілей з ефективністю, продуктивністю та задоволеністю в конкретному контексті використання".

Сьогодні термін "юзабіліті" все частіше вживається як синонім до слова "ергономіка" у відношенні таких продуктів, як побутова електроніка або засоби зв'язку. У більш широкому контексті він може використовуватися для оцінки ефективності дизайну як механічних об'єктів і інструментів (таких як мишка, стілець, настільна лампа, дистанційний пульт), так і графічного дизайну (такого як меню телевізора, меню та система навігації на сайті, інтерфейс програмного забезпечення). Отже, найвлучнішим перекладом слова "usability" є "зручність і простота використання (застосування)", "дружелюбність" і навіть "практичність"[9].

Структура інтерфейсу головної сторінки розроблюваного додатку для моніторингу зображена на рисунку нижче(див. рисунок 2.1).

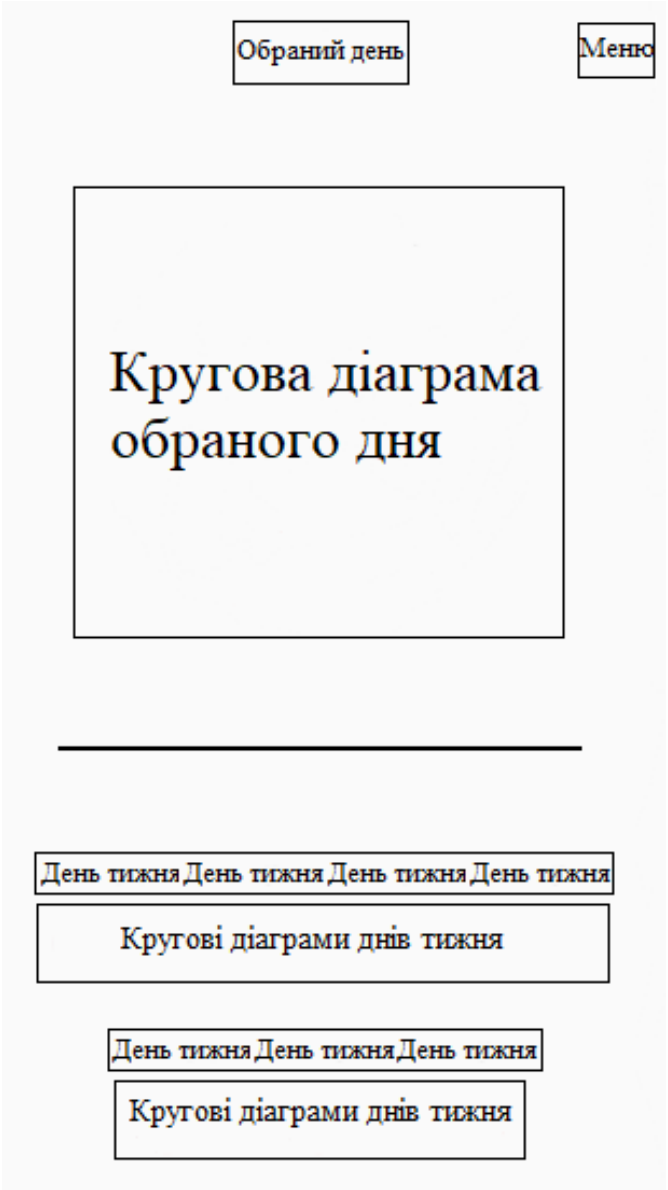

Рисунок 2.1 - Структура інтерфейсу головної сторінки додатку

На головній сторінці додатку розміщені кругові діаграми поточного тижня з відповідними підписами днів. Кожна з діаграм складається з секторів, що відповідають часу вказаному на виконання поставлених задач.

Обираючи день тижня, натискаючи на діаграму, вона переміщуються на центр екрану та збільшується у маштабі. Нижче діаграми з’являється список завдань за обраний день тижня з можливостями: розпочати виконання завдання, редагувати завдання та можливості видалити завдання на рисунку нижче(див. рисунок 2.2).

Рисунок 2.2 - Структура інтерфейсу сторінки обраного дня тижня

Можливість редагування та видалення завдань зі списку здійснюється шляхом виконання жесту слайд ліворуч. При проведенні назви завдання ліворуч з’являються інтуїтивно зрозумілі кнопки з відповідними функціями.

Можливість почати виконання завдання та запустити таймер відліку часу відведеного на його виконання, відповідно, здійснюється шляхом виконання жесту слайд праворуч, затиснувши назву завдання.

У верхньому лівому кутку екрану знаходиться кнопка повернення на попередню, головну, сторінку.

У верхньому правому кутку знаходиться кнопка меню, при натисненні на яку з’являється меню з можливостями видалити усі завдання за обраний день чи додати до списку нове завдання.

При виборі можливості створення нового завдання відкривається сторінка для заповнення основних полів нового завдання, таких як: назва завдання, опис завдання, та виставлення часу виконання завдання. Також буде необхідно обрати день чи декілька днів, в які завдання має бути виконане.(див. рисунок 2.3).

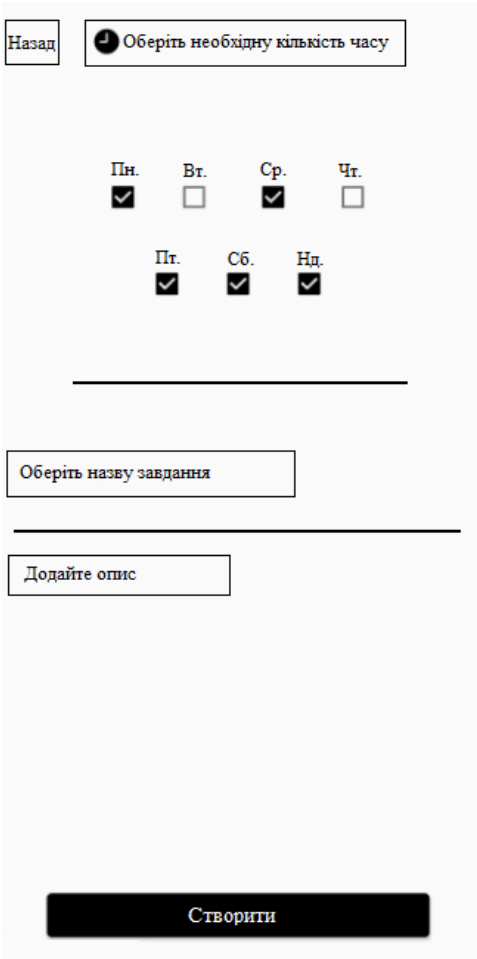

Рисунок 2.3 - Структура інтерфейсу сторінки створення нового завдання

При виборі кількості часу, необхідного для виконання завдання, відкривається віджет для вибору кількості годин та хвилин шляхом гортання коліщатка зі значеннями (див. рисунок 2.4).

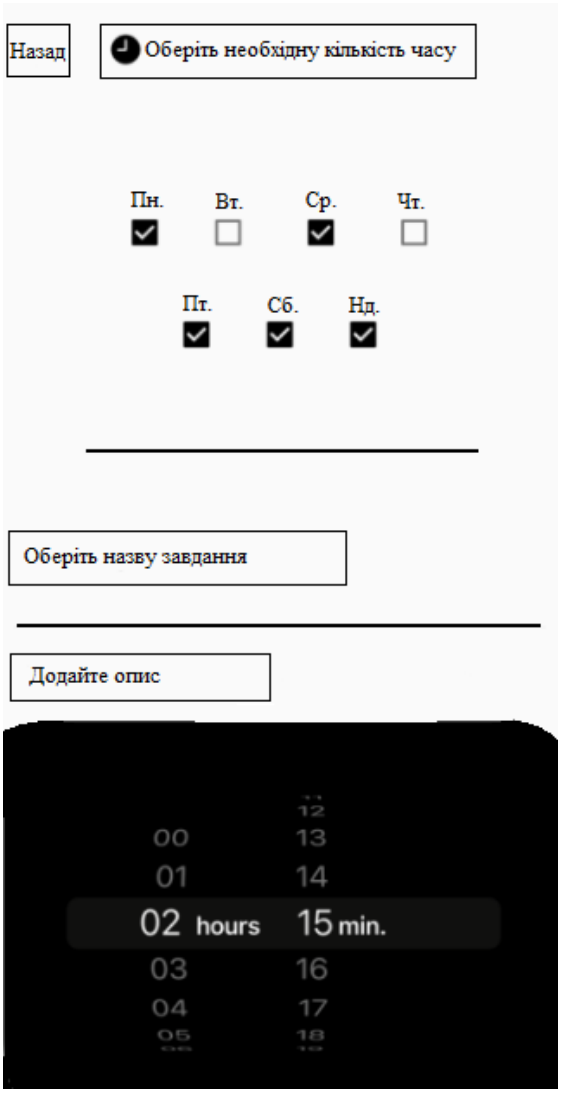

Рисунок 2.4 - Структура інтерфейсу сторінки створення нового завдання - вибір часу

2.3 Удосконалення методу моніторингу виконання задач працівниками

У процесі виконання завдання працівник може стикнутися з труднощами, які можуть призвести до неможливості успішного завершення завдання або порушення встановлених термінів. Також під час обговорення завдання між працівниками може виникнути конфліктна ситуація, що створює ризик порушення термінів виконання. В обох випадках характерні аномальні емоційні реакції при обговоренні задачі.

Своєчасне втручання менеджера проекту у процес виконання завдання може допомогти зберегти робочий час та зменшити ризики зриву термінів. Тобто, необхідно враховувати можливість порушення термінів виконання завдання при його оцінці ризиків.

Для виявлення проблемних завдань пропонується використовувати аналіз тональності тексту, який виконується при описі завдання працівником (сантимент-аналіз). В процесі обговорення завдання загалом тональність коментарів може змінюватися залежно від успішності його виконання, причому для оцінки ризиків більш важливі останні коментарі, аніж перші.

Наприклад, на початку виконання завдання працівники можуть стикатися з труднощами, через що коментарі мають виражене негативне забарвлення. Однак з часом проблема може бути вирішена, і тональність коментарів може змінитися на нейтрально-позитивну. При оцінці ризиків невчасного виконання завдання важливо враховувати цей аспект. Таким чином, для оцінки ризику невчасного виконання завдання рекомендується розраховувати зважену інтегральну оцінку тональності кожного текстового документа, що відноситься до завдання(див. рисунок 2.5).

$$R = \frac{2}{N * (N+1)} \sum_{i=1}^{N} i * c_i,$$

Рисунок 2.5 – Формула розахунку інтегральної оцінки тональності

де c_i – показник compound i – ого документа, N – кількість документів, R – інтегральний показник ризику.

Потім потрібно від числового значення перейти до якісної оцінки ступеня ризику. Для цього пропонується таблиця переходу (табл.2.1).

Таблиця 2.1 – Таблиця переходу для визначення якісної оцінки ступеня ризику невиконання завдання.

Рівень ризику	Опис	Значення інтегрального показника
Ймовірний	Невиконання або зрив термінів виконання задачі відбувається у більшості випадків	[-1; 0)
Звичайний	Зрив термінів виконання зазвичай не відбувається	[0; 0.35]
Низький	Невиконання або зрив термінів виконання практично не відбувається	(0.35; 1]

Визначення ключових тематик у дискусіях відіграє важливу роль у розумінні сутності обговорюваних питань та завдань. Для ефективного виявлення цих тем можна застосовувати метод латентно-семантичного аналізу, або LSA. Цей метод, базуючись на обробці природної мови, взаємодіє з бібліотекою текстових документів, встановлюючи зв'язки між конкретними термінами та їх контекстом. Додатково, LSA дозволяє ідентифікувати специфічні теми або напрямки, які присутні у вивчених документах. Цей аналітичний підхід вже успішно використовується для створення баз даних знань та генерації когнітивних моделей, поглиблюючи розуміння змісту і контексту обговорення.

Латентно-семантичний аналіз - це комплексний процес, який включає в себе декілька ключових етапів для глибокого аналізу текстової інформації. Перший етап цього процесу - підготовка корпусу документів. На цьому етапі текстові документи розбиваються на окремі лексеми або слова. Далі в процесі очищення документа від "шуму" відбувається вилучення незначущих лексем-сполучників, а також пунктуаційних знаків, які можуть завадити коректному аналізу. Останнім кроком на цьому етапі є нормалізація слів, що полягає в приведенні їх до їхньої базової, або початкової форми, що спрощує подальший аналіз та порівняння між різними документами. Після завершення цих процедур розпочинається наступний етап латентно-семантичного аналізу, спрямований на

створення терм-документної матриці та виявлення глибоких залежностей між термінами і документами.

На наступному етапі створюється терм-документна матриця, яка є ключовою структурою для аналізу текстової інформації. У цій матриці кожен рядок відповідає окремому документу, тоді як кожен стовпець відображає конкретне слово, яке з'являється у цих документах. Важливою характеристикою кожного слова у цій матриці є його вага, яка визначається за допомогою спеціальної метрики, відомої як TF-IDF (Term Frequency-Inverse Document Frequency). Ця метрика допомагає враховувати, наскільки часто зустрічається слово у конкретному документі в порівнянні з його загальною частотою в корпусі текстів, надаючи таким чином більш точне розуміння важливості кожного терміну в контексті аналізу.

$$tf - idf(r, d, D) = tf(t, d) \times idf(t, D)$$

$$tf(t, d) = \frac{n_t}{\sum_k n_k} \qquad idf(t, D) = \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$$

Рисунок 2.6 – Формула створення терм-документної матриці

У подальшому процесі аналізу використовується метод сингулярного розкладу для детального розгляду отриманої терм-документної матриці. Цей метод не просто допомагає розкрити потенційні відносини між документами, але й дозволяє глибше зануритися у контекст та семантику текстового матеріалу. Шляхом аналізу сингулярного розкладу можна виявити, які терміни тісно пов'язані між собою в різних документах, а також які ключові теми або концепції переважають у текстах. Це відкриває можливість кластеризації документів на основі їхньої семантики та тематики, в результаті чого можна визначити домінуючі теми або проблематику, що домінувала на конкретному етапі дослідження або розробки. Такий підхід дозволяє витягти цінні інсайти з текстових матеріалів, зосереджуючись на актуальних та ключових аспектах, які

були висвітлені у завданнях або коментарях до них. Приклад виявлених тем зображено на рисунку 2.7.

Рисунок 2.7 – Виявлені методом LSA теми

З метою оцінки ефективності визначено наступні критерії:

- швидкість виявлення проблемних завдань;

- швидкість виявлення взаємопов'язаних завдань.

Для аналізу поточного стану розробки проекту були залучені два досвідчених експерта у галузі проектного менеджменту , які мають більше 5 років практики у керуванні проектами. Проведено два експерименти, в межах кожного з яких експерт у ролі проектного менеджера оцінює поточний стан розробки проекту, використовуючи:

- звичайну Kanban-дошку із переліком завдань;

- розроблений метод для моніторингу процесу розробки на основі аналізу текстових артефактів.

Під час експерименту експерт у ролі проектного менеджера переглядає список завдань та їх зміст (текст опису та коментарів) з метою виявлення завдань, що взаємодіють між собою. Основною метою є зрозуміння проектним

менеджером структури взаємозв'язків між завданнями та переліку тем, до яких відноситься кожне завдання. Спочатку експеримент з виявлення взаємопов'язаних завдань проводиться за допомогою класичної Kanban-дошки. Після цього аналогічний експеримент виконується за допомогою розробленого методу(див. рисунок 2.8).

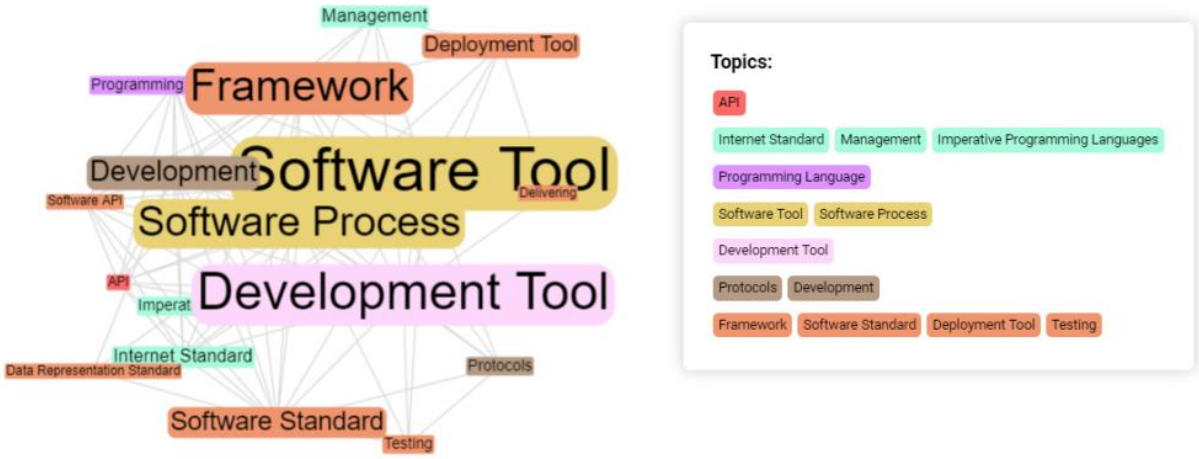

Рисунок 2.8 – Виявлені категорії та підкатегорії виявлених термінів

У межах першого експерименту було отримано дані, що використання розробленого методу пришвидшує процес огляду та виявлення взаємопов'язаних задач і їх тематики у 1.3 рази для першого експериментатора та у 1.24 рази для другого експериментатора. У рамках другого експерименту бачимо, що використання розробленого методу пришвидшило процес огляду та виявлення взаємопов'язаних задач і їх тематики у 1.18 рази для першого експерта та у 1.16 разів для другого експерта. Підсумовуючи, розрахуємо інтегральну оцінку часу, витраченого на оцінку ступеня ризику задачі та виявлення взаємопов'язаних задач з їх темами для двох експериментів сумарно.

Kanban-дошка – 825 хвилин

Розроблений метод – 678 хвилин

Отже, можна зробити висновок, що вдосконалений метод моніторингу виконання задач працівниками значно пришвидшить роботу проектним менеджерам після його програмної реалізації.

2.3 Візуальне модулювання роботи додатку

Перш ніж приступити до втілення будь-якого функціоналу, рекомендується його детально описати, використовуючи діаграми. Зображення можливостей та особливостей розроблюваного продукту забезпечує можливість знаходження більш об’єктивних та ефективних рішень у більшості сценаріїв. Використання діаграм у стандарті UML сприяє більш чіткому розумінню слабких сторін функціоналу та відкриває перспективи для його подальшого удосконалення. Цей підхід дозволяє більш системно аналізувати функціонал, що полегшує виявлення його можливих недоліків та визначення шляхів вдосконалення.

Діаграми послідовності виокремлюються як ефективний засіб формалізації сценаріїв використання. Однією з їх ключових переваг є можливість визначення взаємодіючих компонентів та опис потоків повідомлень на початкових етапах розробки сценаріїв. Ці компоненти та повідомлення в подальшому перетворюються у конкретні класи (об’єкти) та методи цих об’єктів. В результаті цього процесу негайно визначається модель системи подій (Actions), яку ці класи (об’єкти) будуть підтримувати та обробляти. Використання діаграм послідовності сприяє визначенню взаємозв’язків між компонентами системи та створенню базового каркасу для подальшого проектування та реалізації[9].

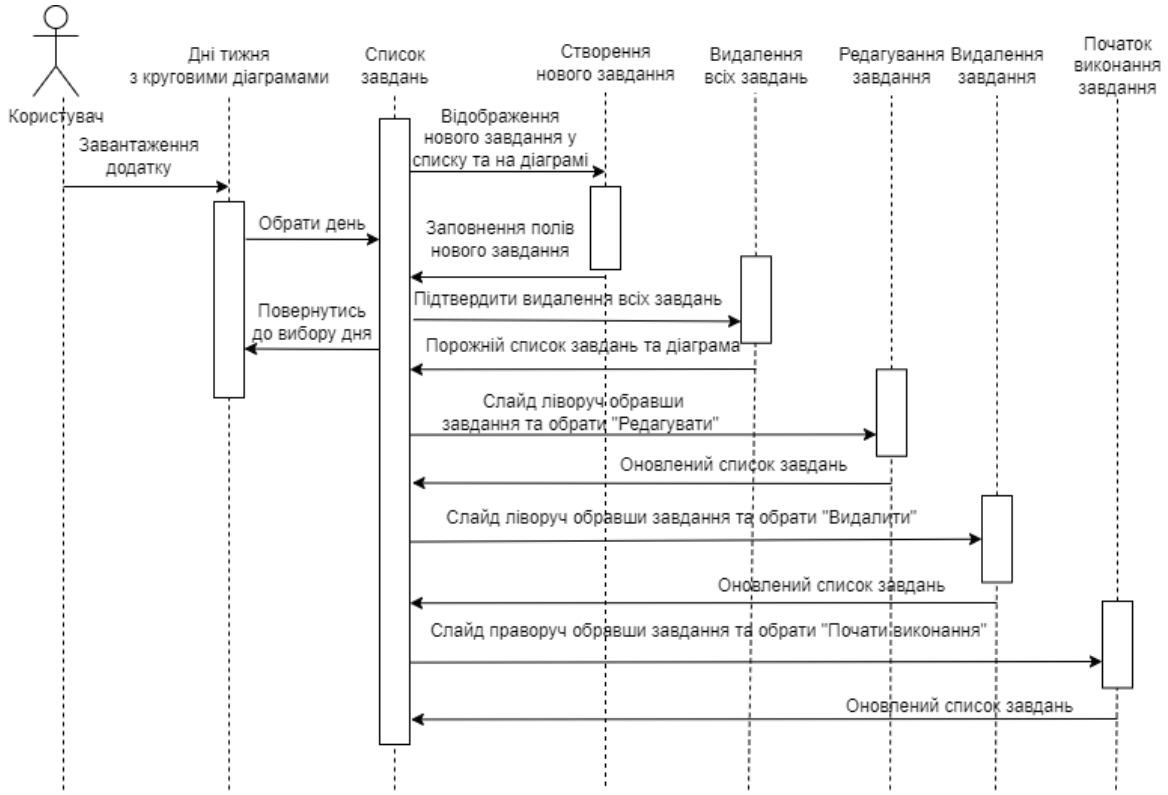

Рисунок 2.5 – Діаграма послідовності додатку

На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення(див. рисунок 2.5).

Діаграма послідовності відображає послідовність взаємодій між різними об'єктами або компонентами системи в рамках конкретного сценарію використання. Діаграма послідовності надає чіткий вигляд послідовності викликів та відповідей між об'єктами в рамках конкретного сценарію, допомагаючи розібратися у логіці взаємодії та визначити можливість подальших вдосконалень[10].

Діаграма діяльності у мові моделювання UML є засобом візуального відображення структури та порядку виконання дій. У цьому контексті граф діяльностей є варіантом графу станів скінченного автомату, де кожна вершина представляє конкретну дію, а переходи відбуваються після завершення

виконання цих дій. Використання діаграм діяльності дозволяє не лише візуалізувати послідовність дій, але й чітко визначати залежності між ними, вказуючи на логіку виконання процесу. Відзначаючи конкретні дії як вершини та встановлюючи переходи між ними, діаграма діяльності уможливлює зрозуміле та систематичне представлення ланцюжка подій та розгортання дій в процесі взаємодії системи чи програмного продукту.

Дія, в контексті специфікації, є ключовою одиницею, яка визначає поведінку системи. Здійснюючи обробку множини вхідних сигналів, дія перетворює їх у відповідну множину вихідних сигналів. Важливо відзначити, що одна або обидві з цих множин можуть залишатися порожніми. Виконання дії еквівалентне виконанню конкретної операції. Також, виконання діяльності означає виконання окремої діяльності, включаючи в себе виконання всіх дій, що утворюють цю діяльність. Кожна окрема дія в рамках діяльності може виконуватись один, два або більше разів під час виконання діяльності. Умовою є те, що дії повинні отримувати, обробляти та тестувати дані, при цьому деякі дії можуть вимагати конкретної послідовності виконання. Важливо відзначити, що специфікація діяльності на більш високих рівнях сумісності може дозволяти виконання різних (логічних) потоків та містити механізми синхронізації для забезпечення правильного порядку виконання дій[11].

Приймаючи до уваги переваги діаграми діяльності, було прийнято рішення про створення діаграми для створення чіткішого розуміння створюваного додатку. На створеній нижче діаграмі діяльності описано можливості додатку та послідовність доступних дій з боку користувача(див. рисунок 2.6).

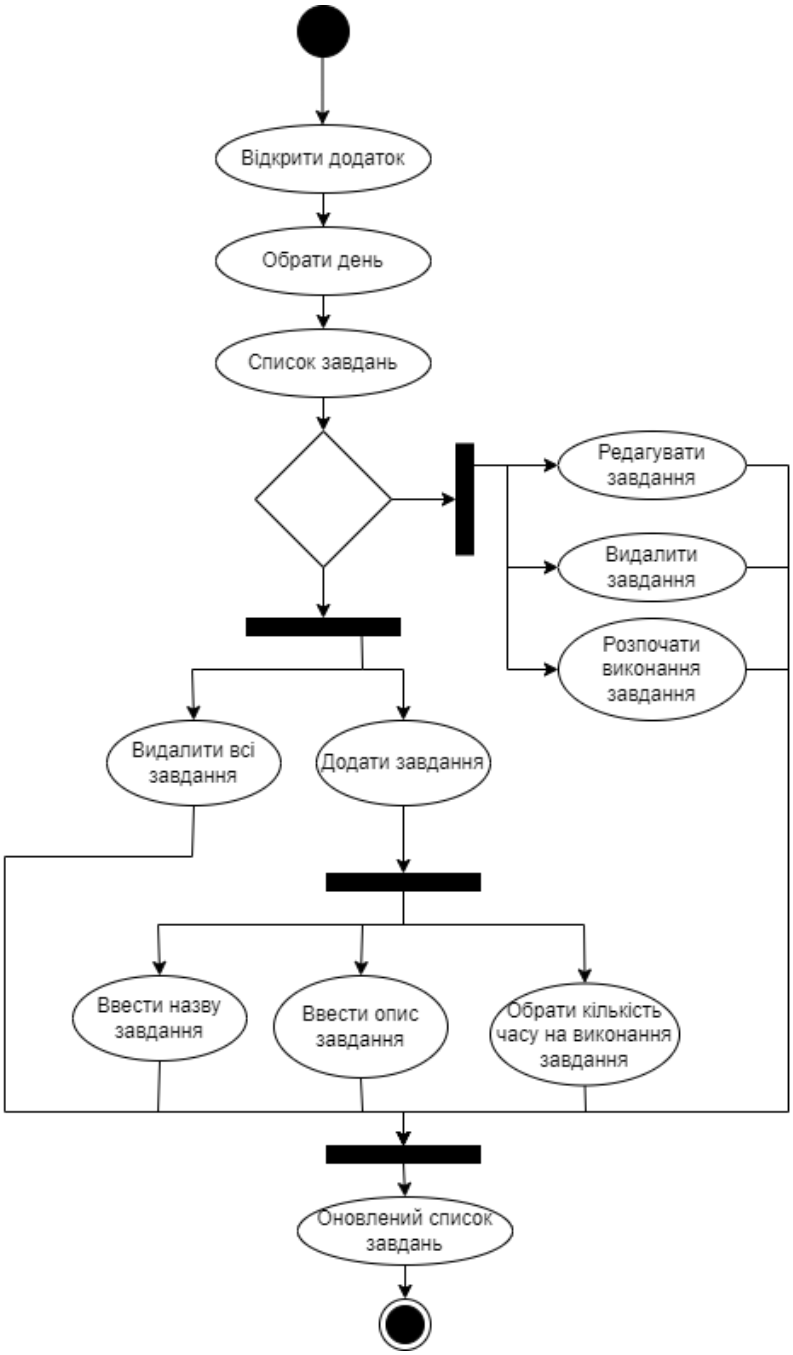

Рисунок 2.6 – Діаграма діяльності додатку

2.4 Розробка алгоритму роботи програми

Блок-схема алгоритму визначає графічну модель процесів, які повинен виконувати додаток, візуалізуючи їх у вигляді окремих функціональних блоків, кожен із яких відповідає за виконання конкретних завдань(див. рисунок 2.7).

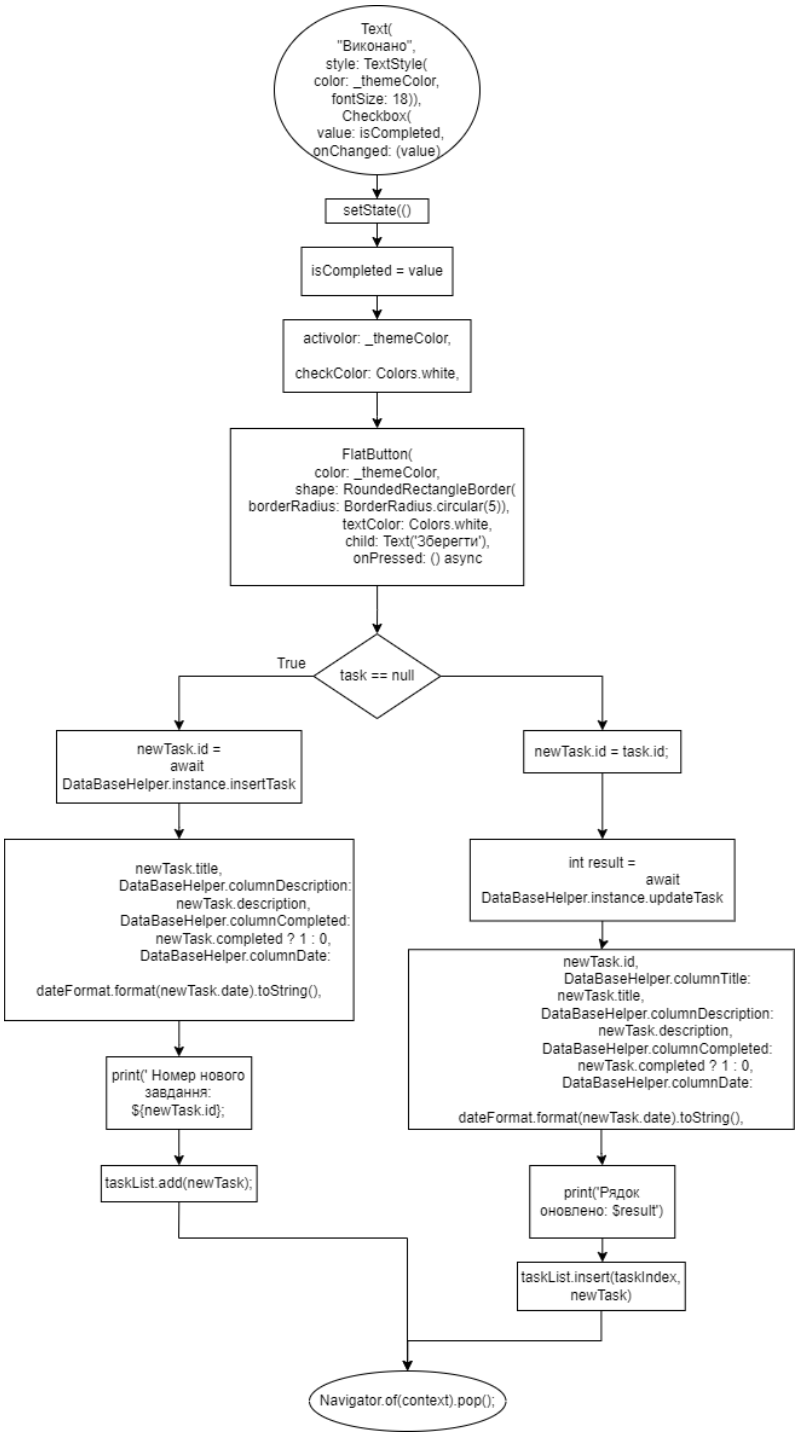

Рисунок 2.7 - Блок-схема роботи функції створення нових завдань

Однією з ключових функцій додатку є операція створення та редагування завдань. Ця функція відповідає за ініціацію вікна для створення нових завдань та редагування вже існуючих. Вона надає можливість відкривати вікно для створення нових завдань та вдосконалення існуючих, використовуючи жест свайп вліво для швидкого доступу до опцій обраної події. Таким чином, блок-

схема дозволяє візуально розібратися в структурі додатку та зрозуміти, як різні функціональні блоки взаємодіють для забезпечення заданих операцій.

2.5 Висновок

У другому розділі магістерської кваліфікаційної роботи було виконано високоякісний аналіз сучасного стану галузі, що стосується моніторингу продуктивності персоналу. Піддавши докладному аналізу, було не лише ідентифіковано основні тенденції в цій області, але й обґрунтовано стратегічний вибір інтерфейсу для програмного додатку, враховуючи найсучасніші рішення та практики.

Крім того, в рамках цього розділу було вивчено та визначено алгоритми роботи створюваного додатку, для забезпечення оптимальної ефективності та користувацького комфорту. Провівши аналіз вимог до кінцевої версії мобільного додатку, було розроблено діаграми діяльності та послідовності, які висвітлюють не лише основні функціональності, але і взаємодії між ними.

Було проаналізовано важливість текст-майнінгу особливо в контексті діяльності персоналу. Text mining не тільки ефективний для обробки даних про робочий час, продуктивність та відгуки колег, але й для виявлення глибоких зв'язків та закономірностей, які іноді залишаються непоміченими за стандартним підходом. Було визначено, що майнинг може бути корисним для виявлення аномалій у роботі персоналу, що стало важливим аспектом для розробки програмного засобу моніторингу. Таким чином, у рамках зазначеної роботи, майнинг даних вже відігравав ключову роль у глибокому аналізі та управлінні діяльністю персоналу.

Зокрема, важливою частиною роботи була розробка блок-схеми для однієї з ключових функцій додатку, а саме для алгоритму створення та редагування подій. Ця блок-схема стала інструментом для детального розуміння логіки та послідовності операцій, що відбуваються в процесі взаємодії з даною функцією.

3. РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ЗАСОБУ

3.1 Варіантний аналіз та обґрунтування вибору засобів реалізації програмного засобу

При виборі технологій розробки програмного продукту в першу чергу необхідно звернути увагу на самих користувачів та мобільні пристрої якими вони користуються.

Звертаючись до статистичних даних 2022 року, частка ринку iOS та Android у США становить 59,17% iOS та 40,54% Android. У Японії частка ринку iOS становить майже 63%. Так само у Великій Британії iOS з великим відривом перевершує Android. Основна причина того, що iOS є першим вибором споживачів на розвинених ринках, – це їхня купівельна спроможність[12].

При прийнятті рішення у виборі технології для розробки мобільного додатку, було визначено, що кращим варіантом буде використання кросплатформенного підходу, а не традиційного нативного методу. Цей вибір було обґрунтовано декількома причинами, включаючи зростання популярності користувачів на платформі iOS. Крім того, кросплатформна розробка дозволяє отримати численні переваги, які можуть бути важливими у виборі методу розробки. Так, кросплатформні рішення мають ряд вагомих плюсів у порівнянні з нативними підходами до розробки:

- Вартість розробки:

Кросплатформенна розробка має один кодбейз може бути використаний для розробки додатків для обох основних мобільних платформ (iOS та Android). Це дозволяє зменшити витрати на розробку і підтримку двох окремих проектів.

Нативна розробка, у свою чергу, для кожної платформи має окремий код, що може призвести до збільшення вартості проекту.

- Швидкість розробки:

Кросплатформенна розробка для обох платформ відбувається одночасно, що прискорює виготовлення нових функцій і випуск оновлень.

Нативна розробка вимагає своєї окремої розробки для кожної платформи, що може призвести до затримок у випуску нових функцій.

- Загальна архітектура:

У кросплатформенній розробці один кодбейз може використовувати спільні бібліотеки та фреймворки, що спрощує утримання та знижує ймовірність помилок.

Нативна розробка має свою власну архітектуру та вимоги до кожної з платформ, що може призвести до більших труднощів у підтримці та оновленні.

- Швидкість випуску:

Одноразова кросплатформенна розробка може призвести до швидшого випуску додатка на ринок, оскільки весь функціонал одразу доступний для обох платформ.

Нативна розробка вимагає відділеної розробки для кожної платформи, що може призвести до затримок у випуску нових версій.

- Легше тестування:

Тестування кросплатформенного проекту здійснюється на одному кодбейзі, що робить процес тестування більш ефективним та менш часоємким.

У нативній розробцінеобхідно вести відокремлене тестування для кожної платформи, що може призвести до подвійних витрат часу і ресурсів.

Зважаючи на переваги кросплатформенної розробки, було прийнято рішення використовувати стек технологій пов’язаних саме з нею з метою охоплення більшої цільової аудиторії та покращення якості кінцевого продукту.

Для глибшого та всебічного аналізу технологічних можливостей, що могли б вплинути на розробку мобільного додатку, було проведено аналіз трьох провідних технологій у сфері кросплатформенної розробки мобільних додатків: React Native, Flutter та Xamarin. У цьому порівняльному дослідженні було розглянуто різноманітні аспекти кожної з технологій, такі як мова

програмування, рівень продуктивності, а також унікальні та спільні характеристики, щоб визначити, яка з них найкраще підходить для потреб розробки поточного проекту:

1. React Native:

   Мова програмування: JavaScript або TypeScript.

   Основні переваги:

- Велика спільнота розробників і обширний набір готових компонентів.
- Використовує React, що полегшує для веб-розробників входження в мобільну розробку.
- Гнучка інтеграція з нативними модулями.

  Недоліки:

- Продуктивність може бути менше, порівняно з нативною розробкою.
- Деяка залежність від сторонніх модулів.

2. Flutter:

   Мова програмування: Dart.

   Основні переваги:

- Висока продуктивність завдяки компіляції в нативний код.
- Один кодбейз для обох платформ.
- Гарний інструментарій для створення красивого інтерфейсу користувача.

  Недоліки:

- Невелика спільнота порівняно з React Native.
- Є вивчення мови Dart, що може бути новим для деяких розробників.

3. Xamarin:

   Мова програмування: C#.

   Основні переваги:

- Велика інтеграція з екосистемою .NET.
- Доступ до нативних API і бібліотек .NET.
- Висока продуктивність завдяки компіляції в нативний код.

  Недоліки:

- Обмежена безкоштовна версія Xamarin обмежена в можливостях.
- Менша спільнота порівняно з іншими фреймворками.

Загальний висновок:

На основі проведенного порівняльного аналізу можна зробити висновок, що React Native, на перший погляд, представляє собою найбільш оптимальний варіант для нашого проекту. Хоча Flutter пропонує відмінну продуктивність та захоплюючий дизайн, він також вимагає знання мови програмування Dart, що може створити додатковий поріг входу для команди. З іншого боку, Xamarin виявляється привабливим для розробників, які вже знайомі з .NET технологіями, проте він може мати свої обмеження, якщо команда не має відповідного досвіду або експертизи в цій області.

Інтеграція Firebase з Flutter може дозволити команді ефективно оптимізувати та спростити процес розробки, зокрема в контексті взаємодії з серверними ресурсами. Firebase, як платформа від Google, пропонує широкий спектр інструментів та сервісів, спрямованих на підтримку розробки та розгортання як мобільних, так і веб-додатків. Її гнучкість та можливості можуть виявитися незамінними для команд, які прагнуть зосередитися на якості та швидкості розробки без зайвих технічних труднощів[13]. При розробці кросплатформенного мобільного додатку на Flutter використання Firebase має велику кількість перваг. Переваги використання платформи Firebase наведено нижче:

- База даних в реальному часі (Realtime Database): Firebase пропонує базу даних в реальному часі, яка ідеально підходить для синхронізації даних між різними пристроями та користувачами. Це особливо корисно для додатків, де важливо миттєво отримувати оновлення, такі як чати чи реальні групові додатки.
- Аутентифікація користувача: Firebase має інструменти для реалізації безпеки та аутентифікації користувачів. Ви можете використовувати

Firebase Authentication для зручного входу користувачів через різні соціальні мережі або електронні адреси.

- Хмарні зберігання (Cloud Storage): Firebase дозволяє зберігати та обробляти файли, такі як зображення чи відео, за допомогою Cloud Storage. Це може значно спростити завдання роботи з мультимедійним контентом у вашому додатку.
- Хмарні функції (Cloud Functions): Ви можете використовувати Cloud Functions для виконання коду на стороні сервера безпосередньо на серверах Firebase. Це може бути корисно для обробки подій, таких як взаємодія з базою даних після певних подій у додатку.
- Інтеграція з іншими сервісами Google: Firebase добре інтегрується з іншими сервісами Google, такими як Google Analytics, AdMob, Cloud Messaging тощо, що дозволяє вам використовувати весь спектр інструментів для забезпечення ефективності та рекламного взаємодії.
- Простота використання та документація: Firebase відомий своєю простотою в використанні. Офіційна документація є досить повною та доступною, що спрощує інтеграцію з вашим Flutter-додатком.

З урахуванням позитивного досвіду використання мови програмування Dart та її фреймворку Flutter у попередніх проектах, було прийнято рішення віддати перевагу цьому конкретному стеку технологій для створення кросплатформенного додатку. Вивчення та ефективне застосування Flutter, поєднане з потужними можливостями Firebase, створює оптимальне середовище для швидкого та продуктивного розвитку розроблювано програмного продукту.

3.2 Реалізація авторизації користувачів за допомогою сервісів Firebase

Для інтеграції можливостей Firebase у проект необхідно провести додаткові кроки з включення цієї платформи. Завдяки описаним нижче діям буде отримано доступ до розширеного набору інструментів та сервісів Firebase, які значно розширять можливості мобільного додатку.

Першим кроком буде додавання пакету firebase_auth до файлу pubspec.yaml та виконання команди у консолі середи розробки: «flutter pub get» для встановлення(див. рисунок 3.1).

pubspec.yaml

```
name: activetime
description: An advanced, minimalist and powerful time management application.
publish_to: "none"

version: 1.0.0+1

environment:
  sdk: ">=2.12.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  hive: ^2.0.4
  hive_flutter: ^1.1.0
  flutter_bloc: ^8.1.3
  firebase_auth: ^4.0.1
  logger:
  syncfusion_flutter_charts: ^23.2.6
  flutter_slidable: ^3.0.1
  cupertino_icons: ^1.0.2
  audioplayers: ^5.2.1
```

Рисунок 3.1 – Вигляд залежностей файлу pubspec.yaml

Процес ініціалізації Firebase в додатку починається у основному файлі проекту, який має назву main.dart. Окрім цього, створення ключового елемента додатку, такого як сторінка авторизації, вимагає використання специфічних інструментів Firebase, зокрема Firebase Auth. Цей інструмент дозволяє

користувачам легко виконати вхід або зареєструватися в додатку. Коли користувач вибирає опцію входу за допомогою електронної пошти та пароля, використовуються конкретні методи Firebase Auth для забезпечення безпеки та автентифікації даних(див. рисунок 3.2).

```
import 'package:firebase_auth/firebase_auth.dart';

Future<void> signInWithEmailAndPassword(String email, String password)
  try {
    await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: email,
      password: password,
    );
    print("Signed in successfully!");
  } on FirebaseAuthException catch (e) {
    print("Error signing in: $e");
  }
}
```

Рисунок 3.2 – Вигляд реалізації операцій авторизації

Для відстеження змін у стані авторизації використовується потік authStateChanges. Це дозволяє адаптувати інтерфейс залежно від того, чи користувач ввійшов у систему чи вийшов з неї(див. рисунок 3.3).

```
FirebaseAuth.instance.authStateChanges().listen((User? user) {
  if (user == null) {
    print("User is signed out");
  } else {
    print("User is signed in: ${user.uid}");
  }
});
```

Рисунок 3.3 – Вигляд реалізації операцій авторизації

Використано різні віджети, такі як TextField, FlatButton та інші для створення користувацького інтерфейсу для введення електронної пошти, пароля та кнопок для входу чи реєстрації. Додано функціонал для виходу з облікового запису користувача за допомогою signOut():(див. рисунок 3.4).

```
Future<void> signOut() async {
  await FirebaseAuth.instance.signOut();
  print("User signed out");
}
```

Рисунок 3.4 – Вигляд реалізації операцій авторизації

Завдяки описаним етапам було реалізовано авторизацію використовуючи Firebase. При успішній реєстрації, в розділі авторизації буде відображено користувачів додатку. Завдяки Firebase є можливість керувати користувачами зі сторінки додатку на сайті Firebase(див. рисунок 3.5).

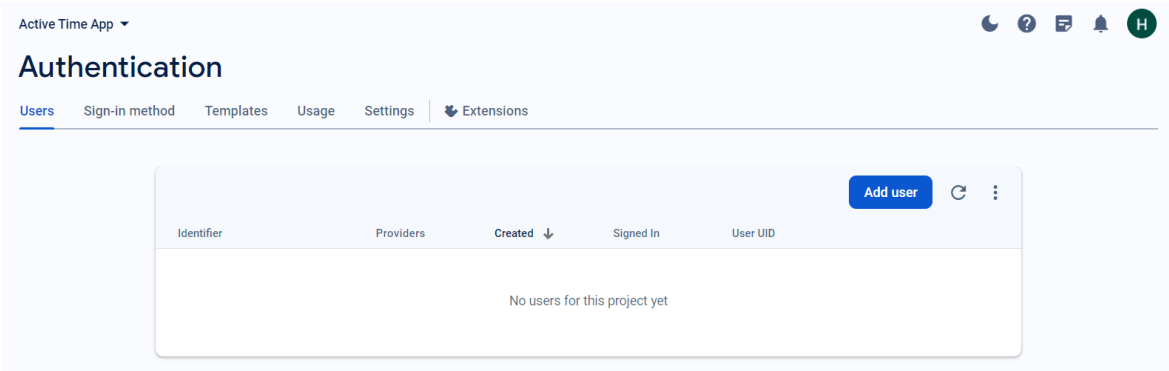

Рисунок 3.5 – Сторінка управління авторизацією створеного додатку

3.3 Розробка програмних модулів мобільного додатку

Flutter представляє собою сучасний відкритий фреймворк, спроектований для створення мобільних, веб-додатків та десктоп-додатків. Його унікальні характеристики та особливості гарантують не тільки високу ефективність розробки, але й гнучкість у конфігурації, що дозволяє розробникам створювати застосунки, які відповідають найсучаснішим стандартам. Крім того, завдяки своїй оптимізованій архітектурі, Flutter забезпечує високий рівень продуктивності, що робить його одним з першовиборних інструментів для багатьох розробників.

Flutter відзначається унікальним підходом до розробки, пропонуючи використання мови програмування Dart для створення не лише клієнтської, але й серверної частини застосунку. Така єдність мови програмування на обох етапах розробки значно спрощує процес, оскільки відпадає необхідність переходу між різними технологіями та мовами. Крім того, це надає розробникам можливість глибше зануритися у особливості Dart, розширюючи свої професійні навички та прискорюючи процес розвитку проектів.

Flutter справляє враження своєю багатофункціональністю, пропонуючи широкий спектр готових віджетів, які спрощують процес розробки. Більш того, завдяки своїй багатошаровій архітектурі, Flutter забезпечує систематизацію коду, роблячи його більш структурованим, зрозумілим і легко підтримуваним. Ціла концепція Flutter базується на ідеї, що кожен елемент є віджетом, що призводить до формування додатку у вигляді дерева віджетів, що спрощує його управління та розширення. [14].

Головними елементами додатку можна вважати головну сторінку, яку користувач бачить після авторизації та сторінку створення нового завдання. На головному екрані клієнт бачить візуалізацію розподіленого часу за поточну дату у вигляді кругової діаграми, кожен сектор на якій зображає окрему задачу. Трохи

нижче, по центру екрану, користувач бачить кожен день тижня зі зменшеними круговими діаграмами за відповідні дні в тижні, що зображать наванатженість завданнями та дають змогу оцінити обсяг роботи(див. рисунок 3.6). Також в верхньому правому куті екрану розташовано кнопку входу в меню, якою було реалізовано можливості видалення всіх завдань за обраний день та створення нового завдання. Частина коду головної сторінки продемонстрована на рисунку 3.9, увесь код додатку вказаний в додатку В.

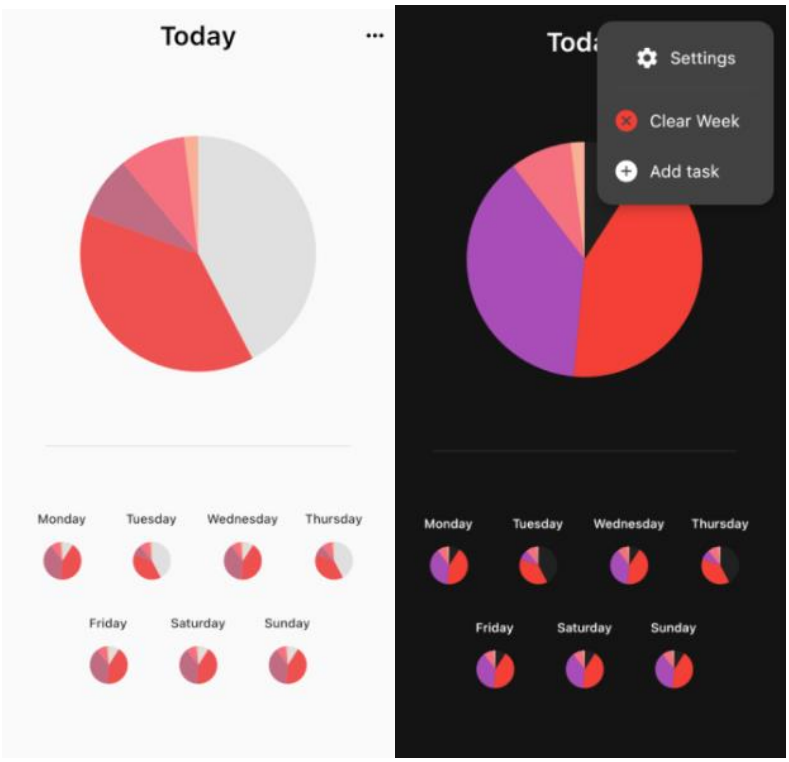

Рисунок 3.6 – Головна сторінка додатку

При натисканні на кнопку створення нового завдання, відкривається окрема сторінка, призначенна для внесення данних нового завдання. На відкритій сторінці створення нового завдання необхідно ввести назву та опис завдання, вказати дні тижня у які це завдання має бути виконане а також обрати

кількість виділеного часу для виконання. На рисунку 3.7 було зображено сторінку створення нового завдання та віджет вибору часу.

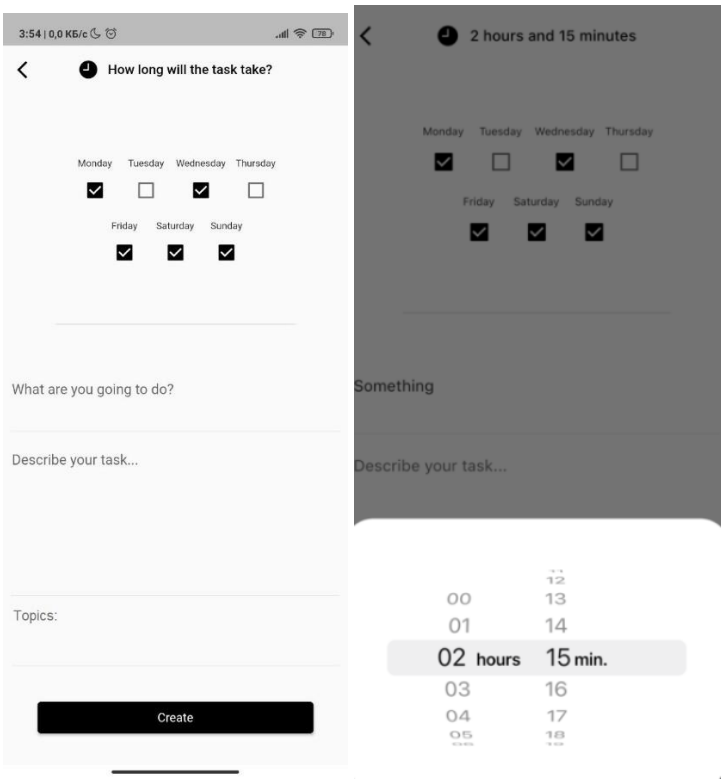

Рисунок 3.7 – Виконання функції створення нового завдання та виставлення часу на виконання

На сторінці створення завдання також відображено теми та підтеми, до яких буде віднесено завдання, використовуючи розроблений метод моніторингу виконання задач. Завдяки цьому підходу користувач може чітко визначити, до якої конкретної категорії чи підкатегорії відноситься його завдання, що сприяє ефективному контролю та структуризації процесу виконання роботи. На рисунку 3.8 було зображено частину коду реалізованої сторінки створення завдання.

```dart
class CreateTaskPage extends VTStatefulWidget {
  final ValueListenable<Box<Task>> todaysBox;
  final Day? selectedDay;

  CreateTaskPage({
    Key? key,
    required this.todaysBox,
    this.selectedDay,
  }) : super(key: key);

  @override
  CreateTaskPageState createState() => CreateTaskPageState();
}

class CreateTaskPageState extends VTState<CreateTaskPage> {
  final localDbService = LocalDBService();
  final viewUtils = ViewUtils();

  final formKey = GlobalKey<FormState>();
  final titleTextController = TextEditingController();
  final desTextController = TextEditingController();

  Duration durationOfTask = Duration.zero;

  bool? monday = false,
      tuesday = false,
      wednesday = false,
      thursday = false,
      friday = false,
      saturday = false,
      sunday = false;
```

```dart
Future<void> showTimePicker() async {
  showModalBottomSheet(
    context: context,
    builder: (builder) {
      return SizedBox(
        height: MediaQuery.of(context).copyWith().size.height / 3,
        child: SizedBox.expand(
          child: CupertinoTimerPicker(
            mode: CupertinoTimerPickerMode.hm,
            minuteInterval: 1,
            secondInterval: 1,
            initialTimerDuration: durationOfTask,
            onTimerDurationChanged: (newTime) {
              setState(() => durationOfTask = newTime);
            },
          ), // CupertinoTimerPicker
        ), // SizedBox.expand
      ); // SizedBox
    },
  );
}

Widget dayChecker(int i) {
  return Padding(
    padding: const EdgeInsets.all(5),
    child: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        Text(
          ViewUtils().rightDayNameGenerator(i, vt, context),
          style: const TextStyle(fontSize: 11),
        ), // Text
        Checkbox(
```

Рисунок 3.8 – Зображення частини коду сторінки створення завдань

На рисунку 3.8 було зображено віджет CreateTaskPage, що відповідає за відображення та реалізовану логіку сторінки створення нових завдань. Оскільки у Flutter всі елементи є віджетами, відповідоно їх комбінація являє собою деревовидну структуру, що має назву дерево віджетів. Тому віджет CreateTaskPage є кореневим для зображеної сторінки та включає в себе інші віджети, такі як віджети ShowTimePicker та dayCheker, які в свою чергу відображають віджет вибору часу, виділеного на завдання, та можливість обрати дати виконання завдання відповідно. Повністю код віджета було винесено в додаток В.

На рисунку 3.9 було зображено частину віджету DayChart, що відповідає за відображення списку днів тижня, в яких зерігаються списки виставлених завдань. Код додатку вказаний в додатку В.

```
class DayChart extends VTStatefulWidget {
  List<Task> tasks;
  final bool isTooltipBehaviorEnabled;

  DayChart({
    Key? key,
    required this.tasks,
    this.isTooltipBehaviorEnabled = false,
  }) : super(key: key);

  @override
  _DayChartState createState() => _DayChartState();
}

class _DayChartState extends VTState<DayChart> {
  final viewUtils = ViewUtils();

  dynamic currentPalette(String? themeName) {
    var palettes = {
      's/2': DayChartColorPalettes.s2Palette,
      'dark': DayChartColorPalettes().darkPalette,
      'default': DayChartColorPalettes().lightPalette
    };

    return palettes[themeName];
  }
```

Рисунок 3.9 – Частина коду головної сторінки додатку

В меню, в верхньому правому куті екрану було також створену кнопку налаштувань(див. рисунок 3.6), завдяки якій, користувач може відкрити сторінку з налаштуваннями додатку, на якій є змога обрати одну з двох кольорових тем: світлу чи темну, та обрати одну з двох мов на вибір: англійську чи українську. Частина коду реалізації сторінки налаштувань зображена на рисунках 3.10 та 3.11.

Можливість зміни кольору теми додатку та мови інтерфейсу додатку було реалізовано будованною функціональністю, яка надає користувачам можливість адаптувати візуальний досвід за допомогою зміни кольорової схеми теми. Ця опція дозволяє користувачам вибрати тон, який найкраще відповідає їхнім особистим вподобанням або умовам користування. Крім того, щодо мови інтерфейсу, Flutter також підтримує мультиязичність, що надає змогу користувачам з легкістю переключатися між різними мовами інтерфейсу в залежності від своїх уподобань.

```dart
class ThemeSelectorWidget extends VTStatelessWidget {
  final Function(int) updateState;
  final Map<int, Widget> themeSegments;
  final int themeSegmentedValue;

  ThemeSelectorWidget({
    Key? key,
    required this.updateState,
    required this.themeSegments,
    required this.themeSegmentedValue,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.appearance'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
          ), // Text
        ), // Align
        const SizedBox(height: 15),
        SizedBox(
          width: 300,
          child: CupertinoSlidingSegmentedControl(
            padding: const EdgeInsets.all(0),
            groupValue: themeSegmentedValue,
            children: themeSegments,
            onValueChanged: (dynamic i) {
              final values = {
```

Рисунок 3.10 – Частина коду сторінки налаштувань додатку, віджет вибору кольорів теми

```dart
class LangSelectorWidget extends VTStatelessWidget {
  final Function(int) updateState;
  final Map<int, Widget> langSegments;
  final int langSegmentedValue;

  LangSelectorWidget({
    Key? key,
    required this.updateState,
    required this.langSegments,
    required this.langSegmentedValue,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.lang'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWei
          ), // Text
        ), // Align
        const SizedBox(height: 15),
        SizedBox(
          width: 500,
          child: CupertinoSlidingSegmentedControl(
            padding: const EdgeInsets.all(0),
            groupValue: langSegmentedValue,
            children: langSegments,
            onValueChanged: (dynamic i) {
              changeLanguage(i, context);
              updateState.call(i);
            },
          ), // CupertinoSlidingSegmentedControl
        ), // SizedBox
      ],
    ); // Column
  }

  void changeLanguage(int i, BuildContext context) {
    var args = {
      0: () => BlocProvider.of<PreferenceCubit>(context).changeLang(
```

Рисунок 3.11 – Частина коду сторінки налаштувань додатку, віджет вибору мови додатку

Дерево віджетів у Flutter представляє собою ієрархічну структуру, де кожен віджет є елементом батьківського віджета, утворюючи дерево з кореневим вузлом, який є головним віджетом додатку. Однією з ключових особливостей цього дерева є відкладене будівництво віджетів, що означає, що вони конструюються лише тоді, коли стають видимими або коли змінюється їхній стан. Це сприяє ефективності та оптимізації продуктивності додатку, дозволяючи використовувати ресурси тільки там, де вони дійсно потрібні.

3.5 Реалізація методу моніторингу виконання задач працівниками

При натисканні на кнопку створення нового завдання, відкривається окрема сторінка, призначенна для внесення данних нового завдання. На відкритій сторінці створення нового завдання необхідно ввести назву та опис завдання, вказати дні тижня у які це завдання має бути виконане а також обрати кількість виділеного часу для виконання. На рисунку 3.12 було зображено сторінку створення нового завдання та віджет вибору часу.

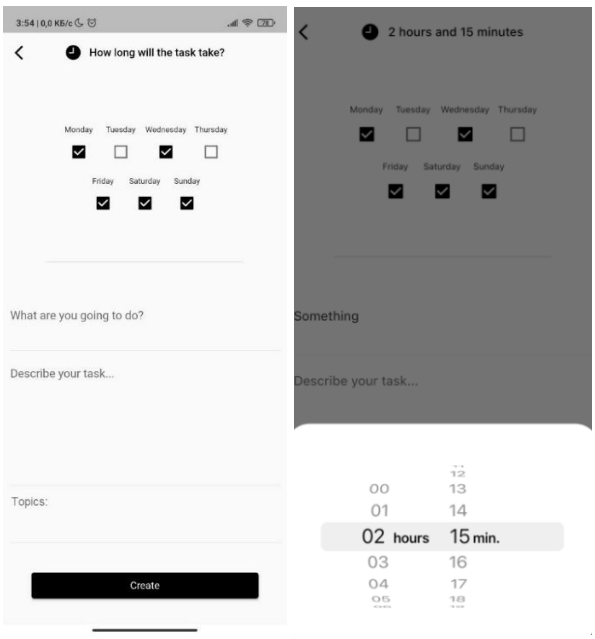

Рисунок 3.12 – Виконання функції створення нового завдання та виставлення часу на виконання

Для реазаціїї розробленого методу моніторингу виконання завдань, було реалізовано окремий блок «Topics:», що зображений на рисунку 3.12, для тих тем та підтем, до яких відноситься створюване завдання при введені його опису.

Відображення тем та підтем було реалізоване сами при введені опису завдання з урахуванням, що можливості текст майнінгу стануть у нагоді користувачам при описі події: чим конструктивніший опис – тим чіткіше метод відобразить тематику завдання. До підтвердження створення завдання користувач, зважаючи на відображені теми, має змогу одразу змінити введений ним опис, у випадку, якщо він помилковий чи описує завдання неточно.

На рисунку 3.13 було зображено частину коду, що відповідає за відображення тем та підтем, згенерованих розробленим методом. Весь код зображено в додатку В.

```
void createTask() {
  if (!formKey.currentState!.validate()) {
    return;
  }

  if (durationOfTask == Duration.zero) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        backgroundColor: Colors.red,
        content: Text(vt.intl.of(context)!.fmt('error.not_found_duration')),
      ), // SnackBar
    );
    return;
  }

  Task task = Task(
    uniquekey: '',
    title: titleTextController.text,
    description: desTextController.text,
    hours: durationOfTask.inHours,
    minutes: durationOfTask.minute,
  );

  for (var i in managableDaysIndexes!) {
    List<Task> tasksList =
        localDbService.rightBoxByCheckBoxId(i).values.toList();
    Duration remainingTime =
        ViewUtils.fullDay - viewUtils.calculateTotalDuration(tasksList);

    if (task.duration > remainingTime) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          backgroundColor: Colors.red,
```

```
for (var i in managableDaysIndexes!) {
  List<Task> tasksList =
      localDbService.rightBoxByCheckBoxId(i).values.toList();
  Duration remainingTime =
      ViewUtils.fullDay - viewUtils.calculateTotalDuration(tasksList);

  if (task.duration > remainingTime) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        backgroundColor: Colors.red,
        content: Text(
          vt.intl
              .of(context)!
              .fmt('error.selected_duration_more_than_remaining'),
        ), // Text
      ), // SnackBar
    );
  } else {
    localDbService.rightBoxByCheckBoxId(i).add(task.copyWith(
          uniquekey: localDbService.rightTaskKeyCheckBoxId(i),
        ));
    ScaffoldMessenger.of(context).removeCurrentSnackBar();
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => Dashboard()),
    );
```

Рисунок 3.13 – Частина коду візуалізації розробленого методу

3.4 Висновок

У третьому розділі магістерської дипломної роботи було здійснено глибокий аналіз різноманітних варіантів технологічних засобів, що могли б бути використані для реалізації програмного продукту. Цей аналіз включав детальне

дослідження як нативних підходів до розробки, так і кросплатформенних рішень. Під час порівняльного аналізу було ретельно розглянуто переваги та обмеження кожного з цих підходів. З урахуванням отриманих даних та врахуванням специфіки завдань, реалізація яких була проведена, було прийняте обґрунтоване рішення на користь кросплатформенної розробки як оптимального технологічного рішення для даного проекту.

Для детального аналізу та вибору найбільш підходящої технології розробки для розроблюваного проекту, було проведено порівняльне дослідження трьох ключових платформ для кросплатформенної мобільної розробки: React Native, Flutter та Xamarin. Кожна з цих технологій має свої унікальні характеристики, такі як мови програмування, швидкодія, а також специфічні для них функціональні можливості. У процесі аналізу ми розглянули ряд критеріїв, включаючи ефективність, гнучкість та загальну продуктивність. В результаті глибокого аналізу та зважження усіх переваг та недоліків кожної платформи, ми прийняли обґрунтоване рішення вибрати комбінацію Firebase та Flutter як найоптимальніше рішення для наших потреб. Було реалізовано авторизацію засобами сервісів Firebase та наведено покрокову інструкцію з інтеграції до Flutter.

У рамках проведеного дослідження та розробки додатку, було приведено конкретні відрізки коду, які ілюструють роботу ключових компонентів та основний функціонал програми. Крім того, було детально розглянуто та проаналізовано взаємодію користувача з головною сторінкою додатку, а також вивчили процес створення нових завдань в контексті цього інтерфейсу. Не менш важливим було дослідження побудови ієрархії компонентів у Flutter, що забезпечує гнучкість та зручність при створенні користувацького інтерфейсу. Щоб надати читачу можливість глибше зануритися у тему та детальніше ознайомитися з кодом, було вирішено включити повний лістинг коду у додаток В до даної роботи.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАСОБУ

4.1 Аналіз методів тестування

Тестування програмного забезпечення - це комплексний процес, спрямований на перевірку відповідності програмного продукту очікуваним вимогам та виявлення потенційних дефектів. Цей процес охоплює виконання компонентів програмної системи, використовуючи як ручні методи, так і автоматизовані інструменти, з метою оцінки різних властивостей програми. Головна мета тестування полягає в виявленні помилок, прогалин або невідповідностей вимогам, що дозволяє уточнити та покращити функціональність продукту відповідно до визначених фактичних вимог. Цей процес є важливим етапом в розробці програмного забезпечення, сприяючи вдосконаленню якості та надійності програми [15].

З метою отримання об’єктивних результатів аналізу методів тестування програмного забезпечення було розглянуто методи тестування "білої скриньки", "сірої скриньки" та "чорної скриньки", що представляють різні підходи до тестування програмного забезпечення, кожен з яких має свої характеристики та використовується в різних ситуаціях.

Тестування методом «білої скриньки» - метод тестування програмного забезпечення, який передбачає, що внутрішня структура, пристрій чи реалізація системи відомі тестувальнику. Вибираються вхідні значення, грунтуючись на знанні коду, який буде їх обробляти. Точно так само ми знаємо, яким повинен бути результат цієї обробки. Знання всіх особливостей тестованої програми та її реалізації - обов’язкові для цієї техніки. Тестування «білої скриньки» - поглиблення у код системи, за межі її зовнішніх інтерфейсів[16].

Тестування методом сірого ящика - метод тестування програмного забезпечення, який передбачає, комбінацію White Box і Black Box підходів. Тобто, внутрішній устрій програми нам відомо лише частково. Передбачається,

наприклад, доступ до внутрішньої структури та алгоритмів роботи ПЗ для написання максимально ефективних тест-кейсів, але саме тестування проводиться за допомогою техніки чорного ящика, тобто, з позиції користувача[16].

Тестування методом "чорної скриньки", також відоме як тестування, засноване на специфікації або тестування поведінки, є технікою, призначеною для перевірки зовнішнього функціоналу програмного продукту. У цьому методі тестування тестувальник взаємодіє з системою, враховуючи її як чорну скриньку, не розглядаючи внутрішню реалізацію чи вихідний код. Фокус робиться на перевірці, як програма реагує на вхідні дані та як вона генерує вихідні результати. Цей підхід дозволяє оцінити відповідність програми визначеним вимогам, виявити можливі дефекти в її функціональності та забезпечити високий рівень коректності та зручності використання для кінцевого користувача. Тестування "чорної скриньки" виявляється особливо ефективним при оцінці зовнішньої взаємодії із системою та визначенні відповідності її функціональності визначеним стандартам і очікуванням користувача[16].

Як результат проведеного порівняння, було прийнято рішення використовувати метод "чорної скриньки" для тестування мобільного додатку. Обраний метод має кілька обґрунтованих причин. По-перше, цей підхід дозволяє зосередитися на зовнішньому функціоналі додатку, визначаючи, як система взаємодіє з користувачем та навколишнім середовищем. По-друге, тестування "чорної скриньки" виявляється ефективним при валідації вихідних результатів та перевірці функціональності, не вдаваючись у деталі внутрішньої реалізації. Це важливо для підтримання гнучкості та незалежності від внутрішньої архітектури додатку. Крім того, метод "чорної скриньки" дозволяє виявити можливі дефекти в інтерфейсі, взаємодії та зручності використання для кінцевого користувача.

4.2 Тестування розробленого програмного продукту

Для більш детальної перевірки та тестування додатку, було обрано рішення тестувати розроблюваний додаток з використанням списку функціональних можливостей, доступних користувачу, що дозволяє перевірити, чи відповідає програмний продукт очікуванням користувача.

Список функціональних можливостей користувача в порядку їх доступності при звичайному користуванні:

1. Відкрити додаток.
2. Авторизуватись/Зареєструватись.
3. Обрати потрібний день.
4. Створити нове завдання.
5. Видалити всі завдання.
6. Зайти в налаштування та змінити мову.
7. Зайти в налаштування та змінити кольорову тему.

Відповідо до списку функціональних можливостей, було проведено регресійне тестування тастворено тест кейси.

Відкриваючи додаток, користувач бачить першу сторінку – сторінку авторизації. На сторінці авторизації користувач може авторизуватись, відновити втрачений пароль та зареєструватись(див. рисунок 4.1).

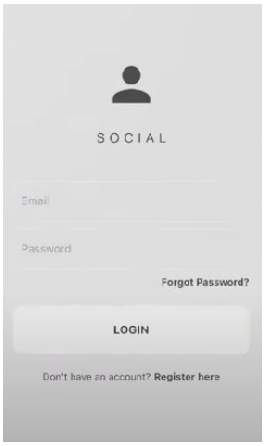

Рисунок 4.1 – сторінка авторизації

Для перевірки можливості реєстрації та авторизації було створено відповідні тест кейси, що описані в таблиці 4.1 та таблиці 4.2.

В таблиці 4.1 описано можливі тестові випадки при реєстрації нового користувача.

Таблиця 4.1 – Виконання Test case реєстрація користувача

№	Назва	Кроки	Очікуваний результат	Статус
1	Введення валідних значень електронної пошти та паролю	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2.Натиснути на кнопку реєстрації.	З’являються поля для введеня електронної пошти та паролю нового користувача.	
		3. Ввести значення пошти та паролю.	Відповідні поля заповнюються.	
		4. Натиснути на кнопку підтвердження.	Введені дані збережені та користувач бачить головну сторінку додатку.	
2	Введення валідного значення електронної пошти та	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано

Продовження таблиці 4.1 – Виконання Test case реєстрація користувача

	невалідне значення паролю	2.Натиснути на кнопку реєстрації.	З’являються поля для введення електронної пошти та паролю нового користувача.	
		3. Ввести валідне значення пошти на не правильний пароль	Поля заповнені введеними даними.	
		4. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідні параметри паролю.	
3	Введення невалідних значень електронної пошти та паролю	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2.Натиснути на кнопку реєстрації.	З’являються поля для введення електронної пошти та паролю нового користувача.	
		3. Ввести невалідне значення пошти на не правильний пароль	Поля заповнені введеними даними.	

Продовження таблиці 4.1 – Виконання Test case реєстрація користувача

		4. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідні параметри введених пошти та паролю.	
4	Залишити поля пошти та паролю пустими	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2.Натиснути на кнопку реєстрації.	З’являються поля для введення електронної пошти та паролю нового користувача.	
		3. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	
5	Залишити поле пошти пустим	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2.Натиснути на кнопку реєстрації.	З’являються поля для введення електронної пошти	

Продовження таблиці 4.1 – Виконання Test case реєстрація користувача

			та паролю нового користувача.	
		3. Ввести лише пароль	Поле з паролем буде заповнене користувацькими даними, але з’явиться повідомлення про необхідність введення пошти	
		4. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	
6	Залишити поле паролю пустим	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2.Натиснути на кнопку реєстрації.	З’являються поля для введення електронної пошти та паролю нового користувача.	
		3. Ввести лише пошту	Поле з поштою буде заповнене користувацькими	

Продовження таблиці 4.1 – Виконання Test case реєстрація користувача

			даними, але з’явиться повідомлення про необхідність введення пароля	
		4. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	

Після вдалої реєстрації, вже існуючий користувач, при повторному вході в додаток має авторизуватись. Для перевірки корректності роботи авторизаціїї додатку було створено тест кейс наведений в таблиці 4.2.

Таблиця 4.2 – Виконання Test case авторизації користувача

№	Назва	Кроки	Очікуваний результат	Статус
1	Введення првильних значень електронної пошти та паролю	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2. Ввести значення пошти та паролю.	Відповідні поля заповнюються.	

Продовження таблиці 4.2 – Виконання Test case авторизації користувача

		3. Натиснути на кнопку підтвердження.	Введені дані підтверджені та користувач бачить головну сторінку додатку.	
2	Введення правильного значення електронної пошти та помилкове значення паролю	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2. Ввести правильне значення пошти на не правильний пароль	Поля заповнені введеними даними.	
		3. Натиснути кнопку підтвердження	З’являється повідомлення про помилку при введені паролю.	
3	Введення значень електронної пошти та паролю з помилками	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2. Ввести не правильне значення	Поля заповнені введеними даними.	

Продовження таблиці 4.2 – Виконання Test case авторизації користувача

		пошти на не правильний пароль		
		3. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про відсутність зареєстрованих даних пошти та паролю.	
4	Залишити поля пошти та паролю пустими	1. Відкрити додаток	Додаток відкривається. Відображається сторінка авторизації	Виконано
		2.Натиснути на кнопку авторизації.	З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	
5	Залишити поле пошти пустим	1. Відкрити додаток.	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2. Ввести лише пароль	Поле з паролем буде заповнене користувацькими даними, з’явиться повідомлення про	

Продовження таблиці 4.2 – Виконання Test case авторизації користувача

			необхідність введення пошти	
		3. Натиснути кнопку підтвердження	Введений пароль зникне. З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	
6	Залишити поле паролю пустим	1. Відкрити додаток.	Додаток відкривається. Відображається сторінка авторизації.	Виконано
		2. Ввести лише пошту	Поле з поштою буде заповнене користувацькими даними, але з’явиться повідомлення про необхідність введення пароля	
		3. Натиснути кнопку підтвердження	З’являється повідомлення з інформацію про необхідність заповнення всіх полів.	

Після авторизації чи реєстрації користувач бачить головну сторінку додатку, на якій відображається кругова діаграма з завданнями за поточну дату та зменшені діаграми за кожен день тижня, натиснувши на які користувач побачить обраний день тижня та список завдань за обраний день. На рисунку 4.2 зображено головну сторінку додатку.

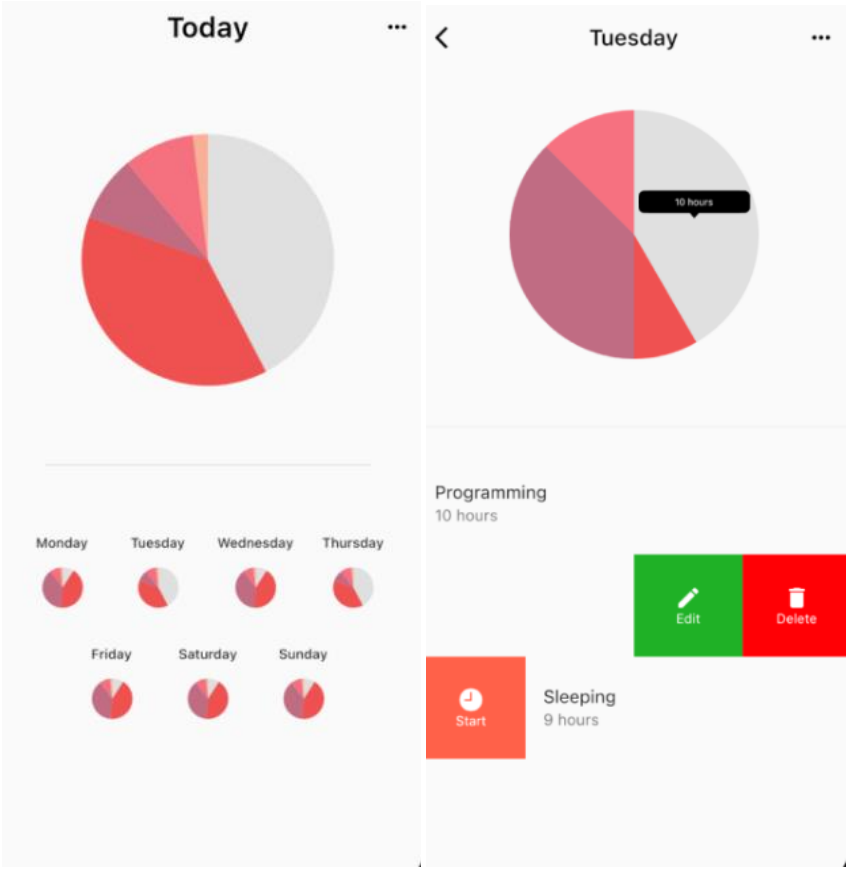

Рисунок 4.2 – Головна сторінка додатку та сторінка обраного дня

Також на рисунку 4.2 зображено сторінку обраного дня та список завдань з якими користувач може взаємодіяти: редагувати видаляти та почати відлік часу, виділеного на виконання завдання.

На рисунку 4.3 було зображено відкрите меню у верхньому правому кунтку екрану, що даже можливість видалити всі завдання та сторити нове завдання.

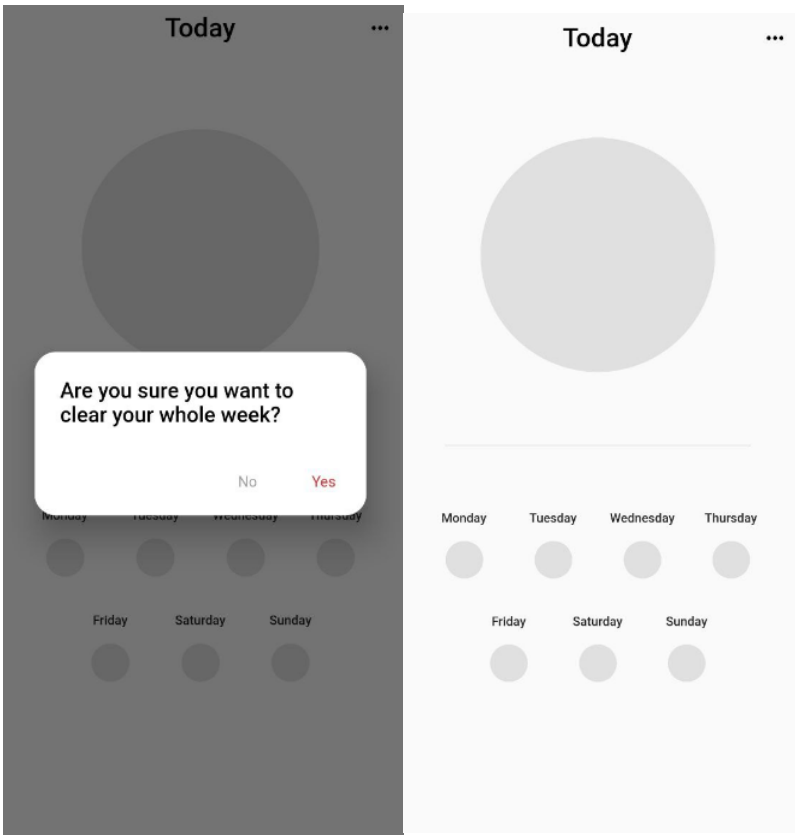

Рисунок 4.3 – Виконання функції «Видалити всі завдання»

При натисканні на кнопку створення нового завдання, відкривається окрема сторінка, призначенна для внесення данних нового завдання. На відкритій сторінці створення нового завдання необхідно ввести назву та опис завдання, вказати дні тижня у які це завдання має бути виконане а також обрати кількість виділеного часу для виконання. На рисунку 4.4 було зображено сторінку створення нового завдання та віджет вибору часу.

На сторінці створення завдання також відображено теми та підтеми, до яких буде віднесено завдання, використовуючи розроблений метод моніторингу виконання задач. Завдяки цьому підходу користувач може чітко визначити, до якої конкретної категорії чи підкатегорії відноситься його завдання, що сприяє ефективному контролю та структуризації процесу виконання роботи.

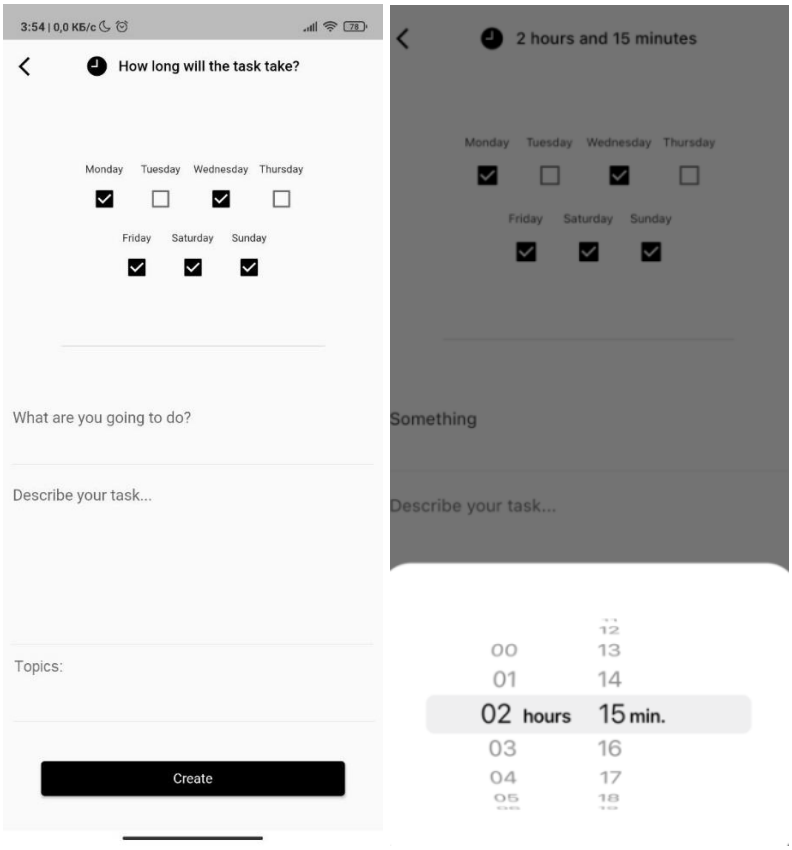

Рисунок 4.4 – Виконання функції створення нового завдання та виставлення часу на виконання

Вибір часу, описаний на рисунку 4.4, було реалізовано не окремою сторінкою а віджетом з двома колами, перше відповідає за вибір кількості годин, а друге за вибір кількості хвилин. Таким чином не переходячи зі сторінки створення нового завдання було реалізовано віджет встановлення кількості часу.

Для створення завдання, після заповнення полів назви та опису, вибору кількості часу та вибору днів виконання, залишається підтвердити сторення завдання натиснувши на кнопку створення завдання, розміщену в нижній частині екрану сторінки сторення завдання.

Використовуючи меню, є можливість перейти на сторінку налаштувань (див. рисунок 4.5).

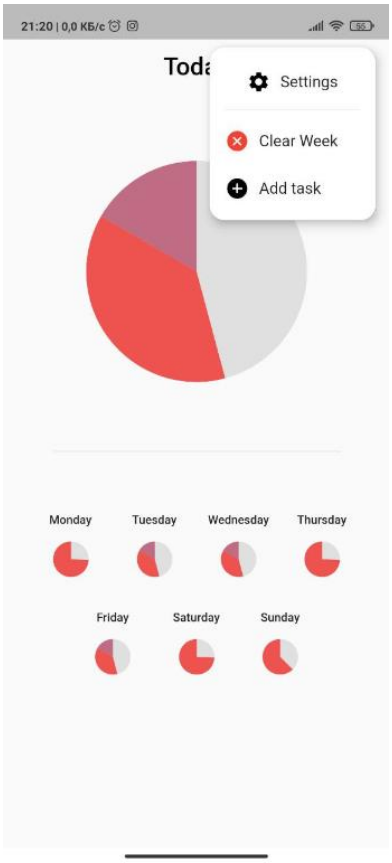

Рисунок 4.5 – Вигляд меню додатку

В меню налаштувань було реалізовано можливість зміни теми додатку та його мови, а також можливість вимикати анімації віджетів додатку.

Можливість зміни кольорових тем в мобільному додатку зі світлої на темну має ряд практичних переваг, спрямованих на поліпшення користувацького досвіду та забезпечення оптимальних умов використання.

Зміна тем впливає на зовнішній вигляд додатку, але також має значний вплив на зручність та функціональність. Темний режим дозволяє оптимізувати взаємодію з додатком в різних умовах. Реалізацію можливості зміни кольорової теми додатку, що було продемонстровано на рисунках зображених вище та на рисунку 4.6, що зображений нижче.

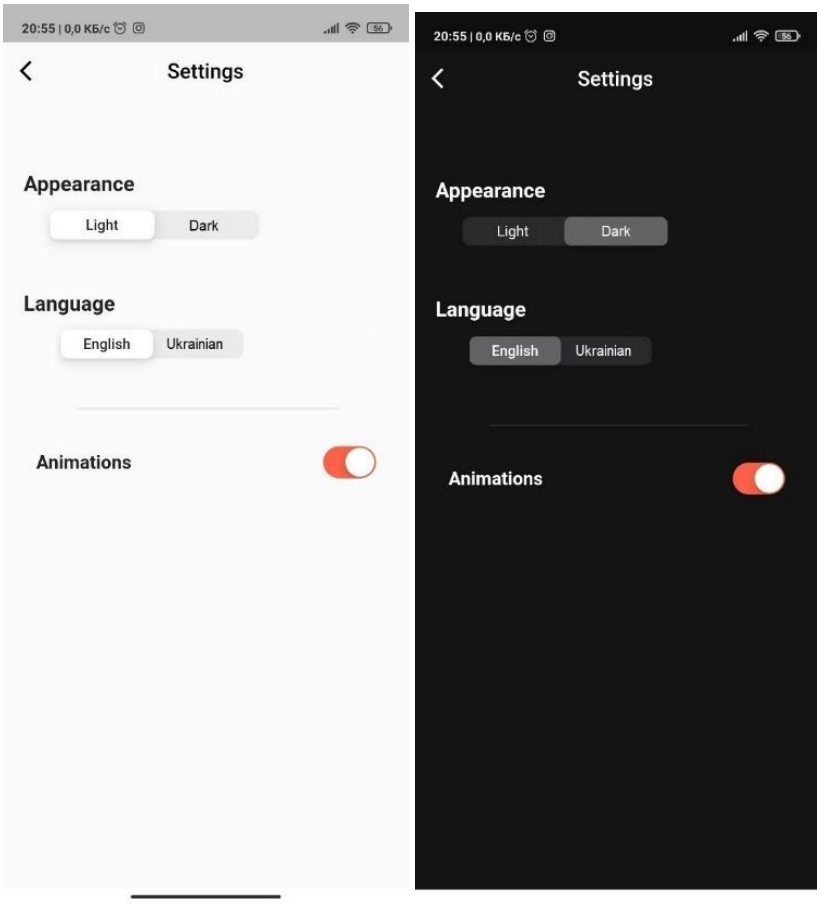

Рисунок 4.6 – Зміна кольорової теми додатку на сторінці налаштувань

Можливість зміни мови інтерфейсу, що зображена на рисунку 4.6, з англійської на українську в мобільному додатку має численні переваги для користувачів. Забезпечення інтерфейсу на рідній мові покращує користувацький досвід, роблячи додаток більш доступним та привабливим для широкого кола людей.

Крім того, зміна мови сприяє глобалізації продукту та розширенню аудиторії, створюючи умови для його успішного використання в англомовних регіонах. Виявляючи увагу до індивідуальних потреб користувачів створюється більше можливостей для задоволення їхніх унікальних очікувань та вподобань.

4.3 Висновок

У четвертому розділі магістерської кваліфікаційної роботи детально було розглянуто аналіз доступних методів тестування. З усіх розглянутих можливостей, було обрано та зосереджено увагу на методі "чорної скриньки". З метою систематичної перевірки розроблених рішень мобільного додатку, було складено докладний список його функціональних можливостей. Цей список було використано як основу для проведення тестування, особливо у сферах модулів авторизації та реєстрації користувачів. Для цих конкретних модулів було застосовано тестові кейси, які дозволили переконатися у їхній надійності та коректності роботи.

У розділі також було надано докладний огляд та характеристику реалізованих функціональних можливостей додатку, а також наведено візуальні матеріали, що демонструють його роботу в рамках описаних головних функціональних можливостей.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного аудиту науково-технічної розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності[17].

Результатом магістерської кваліфікаційної роботи «Метод і програмний засіб для моніторингу діяльності персоналу» є програмний продукт у вигляді веб додатку для керуванням контенту.

Для проведення технологічного аудиту залучено трьох незалежних експертів: Чорноволик Галина Олександрівна (к.т.н. доц. кафедри ПЗ ВНТУ), Причепа Ірина Валеріївна (к.т.н. доц. кафедри ЕП ВНТУ), Бабюк Наталя Петрівна (к.т.н. доц. кафедри ПЗ ВНТУ).

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено 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

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите-р.	0	1	2	3	4
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
Практична здійсненність					
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
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

Продовження таблиці 5.1

Крите-р.	0	1	2	3	4
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, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Бабюк	2 – Чорноволик	3 – Причепа
	Бали, виставлені експертами:		
1	4	3	4
Ринкові переваги (недоліки):			
2	3	2	3
3	4	4	4
4	4	3	4
5	3	4	3
Ринкові перспективи			
6	2	3	3
7	3	3	3
Практична здійсненність			
8	4	4	4
9	4	4	4
10	4	4	4
11	4	4	4
12	3	4	4
Сума балів	$СБ_1$=44	$СБ_2$=44	$СБ_3$=44
Середньоарифметична сума балів $\overline{СБ}$	44		

За даними таблиці 5.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 44 бали, що відповідає рівню «високий».

В якості аналога для розробки було обрано сервіс моніторнгу продуктивності працівників Hubstaff. Основними недоліками аналога є можливе відчуття втрати приватності та постійного контролю з боку співробітників. Постійний моніторинг може впливати на комфорт та відносини в колективі. Щоб відправляти дані на сервери Hubstaff, програма потребує доступу до Інтернету. Це може викликати навантаження на мережу та впливати на швидкість роботи інших інтернет-залежних додатків. Деякі функції Hubstaff можуть бути обмеженими без доступу до Інтернету, що може стати проблемою в умовах обмеженого чи відсутнього інтернет-з'єднання. Також до недоліків можна віднести вартість використання Hubstaff, що може бути високою для невеликих бізнесів і підприємців, особливо якщо компанія знаходиться на ранніх стадіях розвитку та обмеженого бюджету.

У розробці дані проблеми вирішується використанням бібліотек та фреймворків, які мають на меті забезпечити оптимізацію роботи розроблюваного продукту. Також програмний продукт випереджає аналог за такими параметрами як мінімізація інтенсивності використання ресурсів шляхом

оптимізації коду та ресурсозберігаючих технік та добре спроектований інтерфейс з фокусом на інтуїтивність та легкість використання. У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту.

Таблиця 5.4 - Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Функціональність	90%	90%	1
Надійність	80%	80%	1
Сумісність	50%	70%	1,4
Супровід	70%	70%	1
Економія ресурсів і часу	50%	80%	1,6
Простота використання	60%	80%	1,3

Нова розробка може використовуватися як у приватних підприємствах, так і в інших сферах інтернет-технологій. Безпосереднім споживачем є працівники, займаючі керівні посади, що мають обов'язком керування працівниками та їх навантаженням, які прагнуть оптимізувати та автоматизувати управління через використання цієї системи.

Технічна готовність складає 95%. Була розроблена архітектура програмного продукту, розроблено прототип, який частково покриває необхідний функціонал. На даний момент існує декілька зацікавлених сторін з комерціалізації розробки.

5.2 Розрахунок витрат на здійснення науково-технічної розробки.

Проведемо розрахунок витрат на здійснення науково-технічної розробки. Обчислимо основну заробітну $З_о$ розробника, за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t[грн], \quad (5.1)$$

де М – місячний посадовий оклад конкретного розробника, грн;

$Т_р$ – число робочих днів в місяці –24;

t – число днів роботи розробника.

Обчислимо заробітну плату розробника з місячним окладом 17000 грн а також дизайнера з місячним окладом в 15000 грн. Обидва працівники мають оданкову кількість робочих днів у місяці – 24. Число днів роботи розробки 72.

$$З_о = \frac{17000}{24} \cdot 72 = 51000{,}00(\text{грн}).$$

Результати розрахунків зведемо до таблиці 5.5.

Таблиця 5.5 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	17000,00	708,33	72	51000,00
Дизайнера	15000,00	625,00	72	45000,00
Всього				96000,00

Обчислимо додаткову заробітну плату розробників $З_д$. Розрахуємо її як 10% від основної заробітної плати за формулою (5.2):

$$З_д = З_о \cdot 0{,}11\ [грн]. \quad (5.2)$$

$$З_д = 96000 \cdot 0,11 = 10560(грн).$$

Згідно діючого законодавства нарахування на заробітну плату (Єдиний соціальний внесок) складають 22% від суми основної та додаткової заробітної плати, як подано формулою (5.3):

$$Н_з = (З_{о\ р} + З_д) \cdot 0,22 \text{ [грн]}. \quad (5.3)$$

$$Н_з = (96000+10560) \cdot 0,22 = 98323,2 \text{ (грн)}.$$

Обчислимо амортизацію обладнання, що використовувались для розробки. В спрощеному вигляді амортизаційні відрахування розраховується за формулою(5.4):

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В} \text{ [грн]}, \quad (5.4)$$

де $Ц$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, грн;

Т – фактична тривалість використання, міс;

$Т_В$ – термін використання обладнання, приміщень тощо, роки.

Розрахуємо амортизаційні витрати на ноутбук та графічний планшет. Балансова вартість ноутбука становить 16000 грн, а термін його використання – 6 років, а фактична тривалість використання 3 місяці.

Балансова вартість графічного планшету становить 8000 грн, а термін його використання – 6 років, а фактична тривалість використання 3 місяці

$$А = \frac{16000 \cdot 3}{12 \cdot 6} = 666,66(грн).$$

Величина амортизаційних відрахувань наведена в таблиці 5.6.

Таблиця 5.6 – Величина амортизаційних відрахувань

№	Найменування	Балансова вартість, грн	Термін використання, р	Фактична тривалість використання, міс.	Величина амортизаційних відрахувань, грн
1	Ноутбук	16000	6	3	666,66
2	Графічний планшет	8000	5	3	400
Всього:					1066,66

Обчислимо витрати на силову електроенергію за формулою (5.5):

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i} \quad (5.5)$$

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

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 3,45 грн;

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

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

$B_e = 0,4 \cdot 504,0 \cdot 3,45 \cdot 0,95 / 0,97 = 681,17$ *грн.*

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Інші витрати ($B_{ін}$) охоплюють: витрати на Інтернет, управління організацією, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, тощо. Інші витрати приймаємо як 100% від суми основної заробітної плати розробників.

Величину інших витрат розрахуємо за формулою (5.6):

$$B_{ін} = З_о \cdot 50\% \ [грн]. \quad (5.6)$$

$$B_{ін} = 51000 \cdot 0,5 = 25500 \text{ (грн)}.$$

Обчислимо витрати на виконання даної роботи, що є сумою всіх попередніх витрат.

$$B = З_о + З_р + З_д + Н_з + А + В_е + В_{ін} \ [грн].$$

В=51000+10560+98323,2 +1066,66+681,17+25500= 187131,03(грн).

Виконаємо прогнозування загальних витрат *(ЗВ)* на виконання та впровадження результатів виконаної науково-технічної роботи за формулою(5.7):

$$ЗВ = В_{заг} / \beta \ [грн], \quad (5.7)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Якщо розробка знаходиться на стадії впровадження, то β ≈ 0,9.

ЗВ = 187131,03/0,9=207923.36 (грн).

Витрати на виконання наукової роботи та впровадження її результатів становитиме 207923.36грн.

5.3 Розрахунок економічної ефективності та обґрунтування економічної доцільності комерціалізації науково-технічної розробки.

Узагальнюючим результатом, який може отримати потенційний інвестор від комерціалізації науково-технічної розробки, є щорічне збільшення величини

чистого прибутку. Зростання чистого прибутку надає можливість інвестору отримувати додаткові кошти, покращувати фінансові результати діяльності.

Виконання даної наукової роботи та впровадження її результатів складає приблизно до 1 року. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

Збільшення чистого прибутку у потенційного інвестора протягом декількох років розраховується за формулою (5.8):

$$\Delta П_i = (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \quad (5.8)$$

де $\Delta Ц_o$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N – основний кількісний показник, який визначає величину попиту на аналогічні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої науково-технічної розробки у році до впровадження результатів;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. ρ = 0,3;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у ϑ = 18%.

В результаті впровадження наукової розробки покращується якість певного продукту, що дозволяє підвищити ціну його реалізації на 600 грн.

Кількість користувачів збільшиться протягом першого року на 600, другого – 950, третього – 1200. Реалізація продукції до впровадження результатів наукової розробки складала 1 користувача, а її ціна становила – 7500 грн.

Розрахуємо показник прибутку впродовж трьох років відносно базового.

Збільшення чистого прибутку $\Delta П_1$ протягом першого року складе:

$\Delta П_1 = (550 \cdot 29\,500 + (7500 + 550) \cdot 600) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 4316102{,}35$ (грн).

Обчислимо збільшення чистого прибутку $\Delta П_2$ протягом другого року:

$\Delta П_2 = (550 \cdot 29\,500 + (7500 + 550) \cdot (600+950)) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 5883777{,}15$(грн).

Збільшення чистого прибутку $\Delta П_3$ протягом третього року становитиме:

$\Delta П_3 = (550 \cdot 29\,500 + (7500 + 550) \cdot (600+950+1200)) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 7863997{,}93$ (грн).

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку.

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

$$PV = k_{\text{інв}} \cdot \text{ЗВ}$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. $k_{\text{інв}} = 2$.

$$PV = 2 \cdot 207923.36 = 415846{,}72 \text{ (грн)}.$$

Чистий приведений дохід для інвестора розраховуємо за формулою (5.9):

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \text{ [грн]}, \quad (5.9)$$

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

т – період часу, протягом якого виявляються результати впровадженої роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{4316102,35}{(1+0,1)^1} + \frac{5883777,15}{(1+0,1)^2} + \frac{7863997,93}{(1+0,1)^3} = 17694693.20(\text{грн}).$$

Розрахуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою(5.10):

$$Е_{абс} = (ПП - PV), \text{ [грн]}, \quad (5.10)$$

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

PV – теперішня вартість інвестицій PV = ЗВ, грн.

$$E_{абс} = 17694693.20 - 415846,72 = 17278846.48(\text{грн}).$$

Так як $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту.

На п’ятому етапі розрахуємо відносну (щорічну) ефективність вкладених у наукову розробку інвестицій $E_в$. Для цього використаємо формулу (5.11):

$$\mathrm{E_в} = \sqrt[\text{Тж}]{1 + \frac{\mathrm{E_{абс}}}{PV}} - 1 \qquad (5.11)$$

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

PV теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{17278846.48}{415846.72}} - 1 = 2{,}49 \text{ або } 249\%.$$

Розраховану величину $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладати не вигідно. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування визначається за формулою (5.12):

$$\tau = d + f, \qquad (5.12)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = (0,09);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau = 0{,}09 + 0{,}1 = 0{,}19$$

Оскільки $E_в$ = 249% > τ мін = 0,19 = 19%, то потенційний інвестор може бути зацікавлений у фінансуванні науково-технічної розробки.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ можна розрахувати за формулою (5.13):

$$T_{ок} = \frac{1}{E_в}, \tag{5.13}$$

$$T_{ок} = \frac{1}{2{,}49} = 0{,}40\ (роки),$$

Термін окупності становить 0,40 року ($T_{ок}$ <3…5 років), що свідчить, що фінансування даної наукової розробки є доцільним.

5.4 Висновки

У п’ятому розділі було виконано оцінювання комерційного потенціалу розробки. Проведено комерційний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки є високим.

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що підтверджує перспективність розробки. Програмний продукт має кращі функціональні показники, а тому є конкурентоспроможним товаром, а існуючі переваги нової розробки дозволять швидко поширити її на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-технічної роботи загальні витрати на розробку складають 207923.36грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 17278846.48 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 249%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 19%.

Це означає потенційну зацікавленість інвесторів у комерціалізації нового програмного продукту.

Отже, розрахунок ефективності вкладених інвестицій та періоду їх окупності показав, що фінансування розробки є доцільним, адже інвестиції буде повернуто в термін до 1 року.

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи було розроблено мобільний додаток для моніторингу діяльності персоналу. Основною метою розробки була оптимізація часу та реалізація цілей та задач для працівниців. А також покращити можливості керівництва планувати та розподіляти робоче навантаження між працівниками, зважаючи на резільтати замірів часу, що йде на виконання завдань працівниками.

У процесі розробки програмного продукту була створена модель, яка включає покроковий алгоритм роботи програми та методи реалізації програмного продукту.

Було проведено варіативний аналіз стану галузі моніторингу праці та на основі проведеного аналізу виокремлено актуальні питання галузі та спірні моменти популрних аналогів з метою уникнення вже існуючих проблем аналогів та підвищення якості кінцевого продукту. Обгрунтовано вибір програмного забезпечення для реалізації програмного продукту. Розроблено інтерфейс програмного засобу, попередньо зробивши аналіз та вибір інтерфейсу.

Проведено тестування розробленого програмного забезпечення.

Мобільний додаток було розроблено з використанням мови програмування Dart та її фреймворком Flutter, з метою забезпечення якісного кросплатфоременного рішення та майбутнього розширення на платформу iOS. Розробка модулів аутентифікації та підключення хмарної бази даних для зберігання користувацьких даних були реалізовані використовуючи доступні безкоштовні сервіси Firebase.

Виконано економічний аналіз та розрахунки, результати яких підтвердили доцільність розробки та економічну вигідність як для комерційних установ так і для інвесторів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М. О. Поліщук, Н.П. Бабюк «Методи та програмні засоби для моніторингу виконання робочих завдань працівників IT-підприємств» - ВНТУ, 2023. Молодь в науці: дослідження, проблеми, перспективи (МН-2024).
2. Hubstuff [Електронний ресурс] - https://hubstaff.com/
3. Clockify [Електронний ресурс] - https://clockify.me/
4. DeskTime [Електронний ресурс] – https://desktime.com/features/mobile-time-tracking
5. Все, що вам потрібно знати про моніторинг співробітників [Електронний ресурс] - https://yaware.com.ua/uk/blog/vse-shho-vam-potribno-znati-pro-monitoring-spivrobitnikiv/
6. Використання штучного інтелекту (ШІ) для моніторингу продуктивності [Електронний ресурс] - https://ts2.space/uk/%D1%80%D0%BE%D0%BB%D1%8C-%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%BE%D0%B3%D0%BE-%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%83-%D0%B2-%D1%83%D0%BF%D1%80%D0%B0% D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%96-3/#gsc.tab=0
7. Штучний інтелект [Електронний ресурс] - https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi
8. Text Mining [Електронний ресурс] - https://pidru4niki.com/1722020647790/informatika/intelektualni_tehnologiyi_text_mining
9. Г.Ігнатов, Р.Михальча. Текст майнінг. Інтелектуальний аналіз тексту: дизайн досліджень, збір даних та методи аналізу / 2021 / Гуманітарний центр / 344 с.
10. Теорія кольору [Електронний ресурс] - https://sebweo.com/teoriya-koloru-poradi-dlya-nathnennya-veb-dizajneram
11. Використовність [Електронний ресурс] - https://it-rating.ua/yuzabiliti-tse-scho-rozumiyut-pid-terminom-yuzabiliti

12. Застосування UML [Електронний ресурс] - https://dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy
13. Діаграма послідовності [Електронний ресурс] - https://www.maxzosim. com/ sequence-diagrams/
14. Діаграма діяльності [Електронний ресурс] - http://ni.biz.ua/6/6_3/ 6_33177_ diagramma-deyatelnosti-activity-diagram.html[Електронний ресурс]
15. Сучасні мобільні операційні системи [Електронний ресурс] - https://polaridad.es/uk/que-es-un-sistema-operativo-movil/
16. Статистичні дані використання мобільних операційних систем [Електронний ресурс] - https://root-nation.com/ua/news-ua/it-news-ua/ua-android-ios-statistika-2022/
17. Firebase [Електронний ресурс] - https://firebase.google.com/products-build
18. Особливості Flutter [Електронний ресурс] https://flutter.dev/
19. L.Crispin, J.Gregory. Agile Testing: A Practical Guide for Testers and Agile Teams / Addison-Wesley Professional; 1st edition / 2008 / 576 pages.
20. What is Software Testing? Definition, Basics & Types [Електронний ресурс]: https://www.guru99.com/software-testing-introduction-importance.html
21. Методи тестування [Електронний ресурс]: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/
22. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А. Технічне завдання

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

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"20 "вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методу і програмного засобу для моніторингу діяльності персоналу»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. каф. Бабюк Н.П.
" 20 " вересня 2023 р.

Виконав:
студент гр. 1ПІ-22М Поліщук М.О.
" 20 " вересня 2023 р.

Вінниця – 2023 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу і програмного засобу для моніторингу діяльності персоналу».

Галузь застосування – моніторинг діяльності персоналу.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від «18» вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є покращення рівня зручності моніторингу роботи персоналу та оптиміцація часу витраченого на виконання робочих завдань, шляхом розробки мобільного застосунку з використанням програми Visual Studio Code, сервісу Firebase та мовою програмування Dart та її фреймворку Flutter, завдяки вибору стеку технологій кросплатформенної розробки є можливість продовжити розвиток проекту на деклька цілоьвих платформ з метою збільшення доступності додатку до завантаження.

Призначення роботи – моніторинг діяльності персоналу.

4. Вихідні дані для проведення НДР

1. М. О. Поліщук, Н.П. Бабюк «Методи та програмні засоби для моніторингу виконання робочих завдань працівників ІТ-підприємств» - ВНТУ, 2023. Молодь в науці: дослідження, проблеми, перспективи (МН-2024).
2. Штучний інтелект [Електронний ресурс]: https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vidi-ta-skladovi

3. Text Mining [Електронний ресурс]: https://pidru4niki.com/1722020647790/informatika/intelektualni_tehnologiyi_text_mining

5. Технічні вимоги

МКР повинна задовольняти такі вимоги:

– запропонувати нові методи моніторингу діяльності персоналу;

– розробити програмні компоненти мобільного додатку на основі запропонованих методів;

– розробити основний алгоритм з використанням методу моніторингу діяльності персоналу;

– вихідні дані – метод моніторингу діяльності персоналу;

– результат роботи, метод моніторингу діяльності персоналу та мобільний додаток для моніторингу діяльності персоналу.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки.

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Обгрунтування вибору методу розробки та постановка задач дослідження	20.09.23 30.09.23
2	Розробка структури та алгоритмів програмного продукту	01.10.23 20.10.23
3	Розробка модулів програмного засобу	23.10.23 05.11.23
4	Тестування розробленого програмного засобу	05.11.23 10.11.23
5	Економічна частина	10.11.23 20.11.23
6	Оформлення матеріалів до захисту магістерської кваліфікаційної роботи	20.11.23 01.12.23

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 роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Метод і програмний засіб для моніторингу діяльності персоналу

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 22М

Unicheck	
Оригінальність	**92,1%**
Схожість	7,9%

Науковий керівник: к.т.н., доцент кафедри ПЗ Бабюк Н. П.

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 _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Поліщук М.О.

Керівник роботи Бабюк Н. П.

Додаток В Лістинг мобільного додатку

Лістинг модуля main.dart

```
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:vtime/core/services/local_db_service.dart';
import 'package:vtime/core/utils/intl.dart';

import 'app.dart';
import 'core/model/task.dart';
import 'core/vt.dart';


void main() async {
  await Hive.initFlutter();
  Hive.registerAdapter(TaskAdapter());

  await Hive.openBox('preferences');

  await Hive.openBox<Task>('monday');
  await Hive.openBox<Task>('tuesday');
  await Hive.openBox<Task>('wednesday');
  await Hive.openBox<Task>('thursday');
  await Hive.openBox<Task>('friday');
  await Hive.openBox<Task>('saturday');
  await Hive.openBox<Task>('sunday');

  final VT vt = VT();

  vt.intl = Intl();
  vt.localDbService = LocalDBService();

  vt.intl.locale = const Locale('en');
  vt.intl.supportedLocales = languages;

  runApp(App());
}
```

Лістинг модуля програми settings.dart

```
import 'package:audioplayers/audioplayers.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

import 'package:vtime/core/cubits/preference_cubit.dart';
import 'package:vtime/core/utils/widgets.dart';
import 'package:vtime/view/widgets/components/custom_switchers.dart';

import 'widgets/components/appbars.dart';
```

```dart
import 'widgets/utils.dart';

class Settings extends VTStatefulWidget {
  Settings({Key? key}) : super(key: key);

  @override
  _SettingsState createState() => _SettingsState();
}

class _SettingsState extends VTState<Settings> {
  late PreferenceCubit preferenceCubit;

  int themeSegmentedValue = 0, langSegmentedValue = 0;
  bool isAnimationsEnabled = true;
  String selected = 'Nonimooley';

  final langSegments = const <int, Widget>{
    0: Text('English'),
    1: Text('Türkçe'),
    2: Text('Русский'),
    3: Text('ქართული')
  };

  final alarmSounds = [
    'Nonimooley',
    'Crystalie',
    'Favour',
    'Violet',
    'SMS',
    'Points',
    'Iris',
    'Crystal',
    'Harmonics',
    'Marigold'
  ];

  dynamic detectTheme() {
    switch (preferenceCubit.state.themeName) {
      case 'default':
        return themeSegmentedValue = 0;
      case 'dark':
        return themeSegmentedValue = 1;
      case 's/2':
        return themeSegmentedValue = 2;
    }
    return 0;
  }

  dynamic detectLang() {
    switch (preferenceCubit.state.langCode) {
```

```
    case 'en':
      return langSegmentedValue = 0;
    case 'ua':
      return langSegmentedValue = 1;
  }
  return 0;
}

void detectAlarmSound() async {
  var val = await preferenceCubit.currentAlarmSound;
  selected = val!;
}

void detectStateOfAnimations() async {
  var val = await preferenceCubit.isAnimationsEnabled;
  isAnimationsEnabled = val!;
}

@override
void initState() {
  preferenceCubit = BlocProvider.of<PreferenceCubit>(context);

  detectTheme();
  detectLang();
  detectAlarmSound();
  detectStateOfAnimations();

  super.initState();
}

@override
Widget build(BuildContext context) {
  var themeSegments = <int, Widget>{
    0: Text(vt.intl.of(context)!.fmt('prefs.appearance.light')),
    1: Text(vt.intl.of(context)!.fmt('prefs.appearance.dark')),
    2: const Text('S/2')
  };
  return Scaffold(
    appBar: TransparentAppBar(
      titleWidget: Text(vt.intl.of(context)!.fmt('prefs.settings')),
      onLeadingTap: () => Navigator.pop(context),
    ),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(20),
      child: Center(
        child: Column(
          children: [
            const SizedBox(height: 50),
            ThemeSelectorWidget(
              themeSegmentedValue: themeSegmentedValue,
```

```
                themeSegments: themeSegments,
                updateState: (i) => setState(() => themeSegmentedValue = i),
              ),
              const SizedBox(height: 50),
              LangSelectorWidget(
                langSegmentedValue: langSegmentedValue,
                langSegments: langSegments,
                updateState: (i) => setState(() => langSegmentedValue = i),
              ),
              const SizedBox(height: 50),
              AlarmSongSelectorWidget(
                updateState: (val) => setState(() => selected = val),
                selected: selected,
                alarmSounds: alarmSounds,
              ),
              const SizedBox(height: 30),
              ViewUtils.divider,
              const SizedBox(height: 30),
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 10),
                child: SwitcherTile(
                  title: vt.intl.of(context)!.fmt('prefs.animations'),
                  switcherValue: isAnimationsEnabled,
                  onChanged: (v) {
                    setState(() => isAnimationsEnabled = v);
                    preferenceCubit.changeStateOfAnimations(v);
                  },
                  mainAxisAlignment: MainAxisAlignment.spaceBetween,
                  titleStyle: const TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class AlarmSongSelectorWidget extends VTStatefulWidget {
  final Function(dynamic) updateState;
  final String selected;
  final List<String> alarmSounds;

  AlarmSongSelectorWidget({
    Key? key,
    required this.updateState,
```

```
    required this.selected,
    required this.alarmSounds,
  }) : super(key: key);

  @override
  _AlarmSongSelectorWidgetState createState() =>
      _AlarmSongSelectorWidgetState();
}

class _AlarmSongSelectorWidgetState extends VTState<AlarmSongSelectorWidget> {
  final audioPlayer = AudioPlayer();
  final player = AudioCache(prefix: 'assets/alarms/');

  @override
  void initState() {
    player.fixedPlayer = audioPlayer;
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.alarmSound'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
          ),
        ),
        const SizedBox(height: 15),
        FractionallySizedBox(
          widthFactor: .95,
          child: ElevatedButton(
            style: ViewUtils().pomodoroButtonStyle(context),
            onPressed: () => showPicker(context),
            child: Text(
              widget.selected,
              style: TextStyle(color: ViewUtils().pomodoroOrange(context)),
            ),
          ),
        ),
      ],
    );
  }

  Future<void> showPicker(BuildContext context) async {
    showModalBottomSheet(
      context: context,
```

```
      builder: (builder) {
        return SizedBox(
          height: MediaQuery.of(context).copyWith().size.height / 3,
          child: SizedBox.expand(
            child: CupertinoPicker.builder(
              itemExtent: 40,
              childCount: widget.alarmSounds.length,
              onSelectedItemChanged: (index) {
                player.play('${widget.alarmSounds[index]}.mp3');
                BlocProvider.of<PreferenceCubit>(context).changeAlarmSound(
                  widget.alarmSounds[index],
                );
                widget.updateState.call(widget.alarmSounds[index]);
              },
              itemBuilder: (_, i) => Center(
                child: Text(
                  widget.alarmSounds[i],
                  style: context
                      .read<PreferenceCubit>()
                      .state
                      .theme!
                      .primaryTextTheme
                      .headline6,
                ),
              ),
            ),
          ),
        );
      },
    );
  }
}

class ThemeSelectorWidget extends VTStatelessWidget {
  final Function(int) updateState;
  final Map<int, Widget> themeSegments;
  final int themeSegmentedValue;

  ThemeSelectorWidget({
    Key? key,
    required this.updateState,
    required this.themeSegments,
    required this.themeSegmentedValue,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
```

```
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.appearance'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
          ),
        ),
        const SizedBox(height: 15),
        SizedBox(
          width: 300,
          child: CupertinoSlidingSegmentedControl(
            padding: const EdgeInsets.all(0),
            groupValue: themeSegmentedValue,
            children: themeSegments,
            onValueChanged: (dynamic i) {
              final values = {
                0: () {
                  BlocProvider.of<PreferenceCubit>(context)
                      .changeTheme('default');
                },
                1: () {
                  BlocProvider.of<PreferenceCubit>(context).changeTheme('dark');
                },
                2: () {
                  BlocProvider.of<PreferenceCubit>(context).changeTheme('s/2');
                }
              };

              values[i]!.call();
              updateState.call(i);
            },
          ),
        ),
      ],
    );
  }
}

class LangSelectorWidget extends VTStatelessWidget {
  final Function(int) updateState;
  final Map<int, Widget> langSegments;
  final int langSegmentedValue;

  LangSelectorWidget({
    Key? key,
    required this.updateState,
    required this.langSegments,
    required this.langSegmentedValue,
  }) : super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.lang'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
          ),
        ),
        const SizedBox(height: 15),
        SizedBox(
          width: 500,
          child: CupertinoSlidingSegmentedControl(
            padding: const EdgeInsets.all(0),
            groupValue: langSegmentedValue,
            children: langSegments,
            onValueChanged: (dynamic i) {
              changeLanguage(i, context);
              updateState.call(i);
            },
          ),
        ),
      ],
    );
  }

  void changeLanguage(int i, BuildContext context) {
    var args = {
      0: () => BlocProvider.of<PreferenceCubit>(context).changeLang('en'),
      1: () => BlocProvider.of<PreferenceCubit>(context).changeLang('tr'),
      2: () => BlocProvider.of<PreferenceCubit>(context).changeLang('ru'),
      3: () => BlocProvider.of<PreferenceCubit>(context).changeLang('ka'),
    };
    args[i]!.call();
  }
}
```

Лістинг модуля програми day_veiw.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter/foundation.dart';
import 'package:hive/hive.dart';
import 'package:vtime/core/model/day.dart';
import 'package:vtime/core/model/task.dart';
import 'package:vtime/core/services/local_db_service.dart';
import 'package:vtime/core/utils/widgets.dart';
```

```dart
import 'create.dart';
import 'widgets/day_chart.dart';
import 'widgets/task_card.dart';
import 'widgets/components/appbars.dart';
import 'widgets/utils.dart';

class DayView extends VTStatefulWidget {
  final Day day;
  final ValueListenable<Box<Task>>? dayBox;

  DayView({
    Key? key,
    required this.day,
    required this.dayBox,
  }) : super(key: key);

  @override
  _DayViewState createState() => _DayViewState();
}

class _DayViewState extends VTState<DayView> {
  List<Task> tasks = [];

  /// Removes element from [oldIndex], and inserts removed element to [newIndex].
  /// After the local list reordering, it clears old database elements and fills it
by new ones.
  void onReorder(int oldIndex, int newIndex, Box<Task> box) {
    if (oldIndex < newIndex) newIndex -= 1;

    var item = tasks.removeAt(oldIndex);
    tasks.insert(newIndex, item);

    setState(() {});

    for (var i = 0; i < tasks.length; i++) {
      box.delete(tasks[i].key);

      if (tasks[i].title != remainingTimeFillerTaskCode) box.add(tasks[i]);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: TransparentAppBar(
        onLeadingTap: () => Navigator.pop(context),
        titleWidget: Text(widget.day.title ?? '404'),
        action: Padding(
          padding: const EdgeInsets.only(right: 15),
```

```
          child: SettingsPopUpMenu(todaysBox: widget.dayBox!, day: widget.day),
        ),
      ),
      body: ValueListenableBuilder<Box<Task>>(
        valueListenable: widget.dayBox!,
        builder: (contxt, box, _) {
          tasks = box.values.toList().cast<Task>();

          return SingleChildScrollView(
            child: Column(
              children: [
                DayChart(tasks: tasks, isTooltipBehaviorEnabled: true),
                const SizedBox(height: 15),
                const Divider(),
                const SizedBox(height: 15),
                ReorderableListView(
                  shrinkWrap: true,
                  physics: const NeverScrollableScrollPhysics(),
                  onReorder: (oldI, newI) => onReorder(oldI, newI, box),
                  children: [
                    for (var i = 0; i < tasks.length; i++)
                      TaskCard(
                        task: tasks[i],
                        dayBox: box,
                        onDismissed: () => box.delete(tasks[i].key),
                        key: UniqueKey(),
                      ),
                  ],
                ),
              ],
            ),
          );
        },
      ),
    );
  }
}

class SettingsPopUpMenu extends VTStatelessWidget {
  final Day? day;
  final ValueListenable<Box<Task>> todaysBox;

  SettingsPopUpMenu({
    Key? key,
    required this.todaysBox,
    this.day,
  }) : super(key: key);

  final localDbService = LocalDBService();
  final viewUtils = ViewUtils();
```

```
@override
Widget build(BuildContext context) {
  return PopupMenuButton(
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
    child: const Icon(Icons.more_horiz),
    onSelected: (val) => onPopUpItemSelected(context, val),
    itemBuilder: (_) => [
      PopupMenuItem(
        value: 0,
        child: Row(
          children: [
            const Icon(CupertinoIcons.clear_circled_solid, color: Colors.red),
            const SizedBox(width: 10),
            Text(vt.intl.of(context)!.fmt('act.clearDay')),
          ],
        ),
      ),
      PopupMenuItem(
        value: 1,
        child: Row(
          children: [
            const Icon(CupertinoIcons.add_circled_solid),
            const SizedBox(width: 10),
            Text(vt.intl.of(context)!.fmt('act.addTask')),
          ],
        ),
      ),
    ],
  );
}

void onPopUpItemSelected(BuildContext context, dynamic seleted) {
  var methods = {
    0: () => viewUtils.alert(
          context,
          vt,
          title: vt.intl.of(context)!.fmt('clear.day.title'),
          onAct: () {
            localDbService.clearDay(day!.dayIndex!);
            Navigator.pop(context);
          },
        ),
    1: () => Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => CreateTaskPage(
              todaysBox: todaysBox,
              selectedDay: day,
            ),
```

```
            ),
          ),
    };

    methods[seleted]!.call();
  }
}
```

Лістинг модуля програми dashboard.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:vtime/core/model/day.dart';
import 'package:vtime/core/model/task.dart';
import 'package:vtime/core/services/local_db_service.dart';
import 'package:vtime/core/utils/utils.dart';
import 'package:vtime/core/utils/widgets.dart';
import 'package:hive_flutter/hive_flutter.dart';

import 'day_view.dart';
import 'create.dart';
import 'settings.dart';
import 'widgets/mini_day_card.dart';
import 'widgets/components/appbars.dart';
import 'widgets/day_chart.dart';
import 'widgets/utils.dart';

class Dashboard extends VTStatefulWidget {
  Dashboard({Key? key}) : super(key: key);

  @override
  _DashboardState createState() => _DashboardState();
}

class _DashboardState extends VTState<Dashboard> {
  final titleTextController = TextEditingController();
  final localDbService = LocalDBService();

  late Day today = weekDays(vt, context)[0];
  ValueListenable<Box<Task>>? todaysBox;

  // Just contains cases appertitate to week days.
  // And listens now(WeekDay) and generates right function calling.
  void _refreshContent() {
    var cases = {
      DateTime.monday: () {
        today = weekDays(vt, context)[0];
        todaysBox = localDbService.rightListenableValue(
          weekDays(vt, context)[0],
```

```
      );
    },
    DateTime.tuesday: () {
      today = weekDays(vt, context)[1];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[1],
      );
    },
    DateTime.wednesday: () {
      today = weekDays(vt, context)[2];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[2],
      );
    },
    DateTime.thursday: () {
      today = weekDays(vt, context)[3];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[3],
      );
    },
    DateTime.friday: () {
      today = weekDays(vt, context)[4];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[4],
      );
    },
    DateTime.saturday: () {
      today = weekDays(vt, context)[5];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[5],
      );
    },
    DateTime.sunday: () {
      today = weekDays(vt, context)[6];
      todaysBox = localDbService.rightListenableValue(
        weekDays(vt, context)[6],
      );
    },
  };

  return cases[DateTime.now().weekday]!.call();
}

@override
Widget build(BuildContext context) {
  _refreshContent();
  return Scaffold(
    appBar: TransparentAppBar(
      disableLeading: true,
      action: Padding(
```

```
          padding: const EdgeInsets.only(right: 15),
          child: SettingsPopUpMenu(todaysBox: todaysBox!),
        ),
        titleWidget: GestureDetector(
          onTap: () => openNewDay(0, context, day: today, todaysBox: todaysBox),
          child: Text(
            vt.intl.of(context)!.fmt('today'),
            style: const TextStyle(fontWeight: FontWeight.w600, fontSize: 25),
          ),
        ),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.symmetric(vertical: 30),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ValueListenableBuilder<Box<Task>>(
              valueListenable: todaysBox!,
              builder: (context, box, _) {
                final tasks = box.values.toList().cast<Task>();
                return DayChart(tasks: tasks, isTooltipBehaviorEnabled: true);
              },
            ),
            const SizedBox(height: 30),
            ViewUtils.divider,
            const SizedBox(height: 50),
            WeekView(weeks: weekDays(vt, context)),
          ],
        ),
      ),
    );
  }
}

// Method which navigates to DayView with given parameters.
void openNewDay(
  int i,
  BuildContext context, {
  List? weeks,
  Day? day,
  dynamic todaysBox,
}) {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => DayView(
        day: weeks?[i] ?? day,
        dayBox: (weeks != null)
            ? LocalDBService().rightListenableValue(weeks[i])
            : todaysBox,
```

```
      ),
    ),
  );
}

// A widget which has mini day charts of all week.
class WeekView extends VTStatelessWidget {
  final List<Day>? weeks;
  WeekView({Key? key, this.weeks}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        children: [
          Wrap(
            children: [
              for (var i = 0; i < 4; i++)
                MiniDayChart(
                  title: ViewUtils().rightDayNameGenerator(i, vt, context),
                  onTap: () => openNewDay(i, context, weeks: weeks),
                  todaysBox: LocalDBService().rightListenableValue(weeks![i]),
                ),
            ],
          ),
          Wrap(
            children: [
              for (var i = 4; i < 7; i++)
                MiniDayChart(
                  todaysBox: LocalDBService().rightListenableValue(weeks![i]),
                  title: ViewUtils().rightDayNameGenerator(i, vt, context),
                  onTap: () => openNewDay(i, context, weeks: weeks),
                ),
            ],
          ),
        ],
      ),
    );
  }
}

class SettingsPopUpMenu extends VTStatelessWidget {
  final ValueListenable<Box<Task>> todaysBox;
  SettingsPopUpMenu({Key? key, required this.todaysBox}) : super(key: key);

  final localDbService = LocalDBService();
  final viewUtils = ViewUtils();

  @override
  Widget build(BuildContext context) {
```

```
  return PopupMenuButton(
    shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
    child: const Icon(Icons.more_horiz),
    onSelected: (val) => onPopUpItemSelected(context, val),
    itemBuilder: (_) => [
      PopupMenuItem(
        value: 0,
        child: Column(
          children: [
            const SizedBox(height: 15),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                const Icon(Icons.settings),
                const SizedBox(width: 10),
                Text(vt.intl.of(context)!.fmt('prefs.settings')),
              ],
            ),
            const SizedBox(height: 8),
            const Divider(),
          ],
        ),
      ),
      PopupMenuItem(
        value: 1,
        child: Row(
          children: [
            const Icon(CupertinoIcons.clear_circled_solid, color: Colors.red),
            const SizedBox(width: 10),
            Text(vt.intl.of(context)!.fmt('act.clearWeek')),
          ],
        ),
      ),
      PopupMenuItem(
        value: 2,
        child: Row(
          children: [
            const Icon(CupertinoIcons.add_circled_solid),
            const SizedBox(width: 10),
            Text(vt.intl.of(context)!.fmt('act.addTask')),
          ],
        ),
      ),
    ],
  );
}

void onPopUpItemSelected(BuildContext context, dynamic seleted) {
  var methods = {
    0: () => Navigator.push(
```

```
            context,
            MaterialPageRoute(builder: (context) => Settings()),
          ),
      1: () => viewUtils.alert(
            context,
            vt,
            title: vt.intl.of(context)!.fmt('clear.week.title'),
            onAct: () {
              localDbService.clearWeek();
              Navigator.pop(context);
            },
          ),
      2: () => Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => CreateTaskPage(todaysBox: todaysBox),
            ),
          ),
    };

    methods[seleted]!.call();
  }
}
```

Лістинг модуля програми appbars.dart

```
import 'package:flutter/material.dart';

class TransparentAppBar extends StatelessWidget implements PreferredSizeWidget {
  final bool disableLeading;
  final Function? onLeadingTap;
  final Widget? titleWidget;
  final Widget? action;

  const TransparentAppBar({
    Key? key,
    this.disableLeading = false,
    this.onLeadingTap,
    this.titleWidget,
    this.action,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return AppBar(
      title: titleWidget,
      centerTitle: true,
      actions: (action != null) ? [action!] : null,
      leading: disableLeading
```

```
          ? const SizedBox.shrink()
          : IconButton(
              icon: const Icon(Icons.arrow_back_ios),
              onPressed: () => onLeadingTap!(),
            ),
    );
  }

  @override
  Size get preferredSize => const Size.fromHeight(55);
}
```

Лістинг модуля програми create.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:hive/hive.dart';
import 'package:vtime/core/model/day.dart';

import 'package:vtime/core/model/task.dart';
import 'package:vtime/core/services/local_db_service.dart';
import 'package:vtime/core/utils/widgets.dart';
import 'package:vtime/view/widgets/components/buttons.dart';
import 'package:vtime/view/widgets/utils.dart';

import 'dashboard.dart';
import 'widgets/components/appbars.dart';

class CreateTaskPage extends VTStatefulWidget {
  final ValueListenable<Box<Task>> todaysBox;
  final Day? selectedDay;

  CreateTaskPage({
    Key? key,
    required this.todaysBox,
    this.selectedDay,
  }) : super(key: key);

  @override
  CreateTaskPageState createState() => CreateTaskPageState();
}

class CreateTaskPageState extends VTState<CreateTaskPage> {
  final localDbService = LocalDBService();
  final viewUtils = ViewUtils();

  final formKey = GlobalKey<FormState>();
  final titleTextController = TextEditingController();
```

```
final desTextController = TextEditingController();

Duration durationOfTask = Duration.zero;

bool? monday = false,
    tuesday = false,
    wednesday = false,
    thursday = false,
    friday = false,
    saturday = false,
    sunday = false;

List<int>? managableDaysIndexes = [];

@override
void initState() {
  if (widget.selectedDay != null) {
    setSelectedDay(widget.selectedDay!.dayIndex!);
  }
  super.initState();
}

// Sets selected day's value to true.
// So it becomes already selected when user navigates here from any concrete day.
void setSelectedDay(int i) {
  var values = {
    0: () => monday = true,
    1: () => tuesday = true,
    2: () => wednesday = true,
    3: () => thursday = true,
    4: () => friday = true,
    5: () => saturday = true,
    6: () => sunday = true,
  };

  addIndexToList(val: i);

  values[i]!.call();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    extendBodyBehindAppBar: true,
    appBar: appBar(context),
    bottomNavigationBar: SaveButton(
      title: vt.intl.of(context)!.fmt('act.create'),
      onTap: createTask,
    ),
    body: SingleChildScrollView(
```

```
      padding: const EdgeInsets.all(10),
      child: Form(
        key: formKey,
        child: Column(
          children: [
            const SizedBox(height: 150),
            Wrap(children: [for (var i = 0; i < 4; i++) dayChecker(i)]),
            Wrap(children: [for (var i = 4; i < 7; i++) dayChecker(i)]),
            const SizedBox(height: 50),
            ViewUtils.divider,
            const SizedBox(height: 50),
            TextFormField(
              controller: titleTextController,
              maxLines: 2,
              decoration: ViewUtils().nonBorderInputDecoration(
                hint: vt.intl.of(context)!.fmt('task.title.hint'),
              ),
              validator: (v) {
                if (v!.isEmpty) {
                  return vt.intl
                      .of(context)!
                      .fmt('error.title_field_validation');
                }

                return null;
              },
            ),
            const SizedBox(width: 15),
            const Divider(),
            const SizedBox(width: 15),
            TextFormField(
              controller: desTextController,
              minLines: 1,
              maxLines: 15,
              decoration: ViewUtils().nonBorderInputDecoration(
                hint: vt.intl.of(context)!.fmt('task.desc.hint'),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

TransparentAppBar appBar(BuildContext context) {
  return TransparentAppBar(
    titleWidget: GestureDetector(
      onTap: () => showTimePicker(),
      child: Wrap(
```

```
        crossAxisAlignment: WrapCrossAlignment.center,
        children: [
          const Icon(CupertinoIcons.clock_fill),
          const SizedBox(width: 10),
          Text(
            durationOfTask.toHumanLang(vt, context),
            style: const TextStyle(fontSize: 15),
          )
        ],
      ),
    ),
    onLeadingTap: () {
      ScaffoldMessenger.of(context).removeCurrentSnackBar();
      Navigator.pushAndRemoveUntil(
        context,
        MaterialPageRoute(builder: (context) => Dashboard()),
        (route) => false,
      );
    },
  );
}

Future<void> showTimePicker() async {
  showModalBottomSheet(
    context: context,
    builder: (builder) {
      return SizedBox(
        height: MediaQuery.of(context).copyWith().size.height / 3,
        child: SizedBox.expand(
          child: CupertinoTimerPicker(
            mode: CupertinoTimerPickerMode.hm,
            minuteInterval: 1,
            secondInterval: 1,
            initialTimerDuration: durationOfTask,
            onTimerDurationChanged: (newTime) {
              setState(() => durationOfTask = newTime);
            },
          ),
        ),
      );
    },
  );
}

Widget dayChecker(int i) {
  return Padding(
    padding: const EdgeInsets.all(5),
    child: Column(
      mainAxisSize: MainAxisSize.min,
      children: [
```

```
        Text(
          ViewUtils().rightDayNameGenerator(i, vt, context),
          style: const TextStyle(fontSize: 11),
        ),
        Checkbox(
          key: Key('subCheckBox.$i'),
          value: rightDayValueGenerator(i),
          onChanged: (v) {
            setState(() => valueControllerMethod(v, i));
            addIndexToList(val: i);
          },
        ),
      ],
    ),
  );
}

// Returns right value by gave checkbox index.
bool rightDayValueGenerator(int index) {
  var values = {
    0: monday,
    1: tuesday,
    2: wednesday,
    3: thursday,
    4: friday,
    5: saturday,
    6: sunday,
  };
  return values[index]!;
}

// Returns right checkbox value changing by gave checkbox index.
void valueControllerMethod(bool? v, int i) {
  var rightMethodSelector = {
    0: () => monday = v,
    1: () => tuesday = v,
    2: () => wednesday = v,
    3: () => thursday = v,
    4: () => friday = v,
    5: () => saturday = v,
    6: () => sunday = v,
  };

  rightMethodSelector[i]!.call();
}

void addIndexToList({dynamic val}) {
  if (managableDaysIndexes!.contains(val)) {
    managableDaysIndexes!.remove(val);
    return;
```

```
  }

  managableDaysIndexes!.add(val);
}

// Creates tasks for appertitate to selected days.
void createTask() {
  if (!formKey.currentState!.validate()) {
    return;
  }

  if (durationOfTask == Duration.zero) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        backgroundColor: Colors.red,
        content: Text(vt.intl.of(context)!.fmt('error.not_found_duration')),
      ),
    );
    return;
  }

  Task task = Task(
    uniquekey: '',
    title: titleTextController.text,
    description: desTextController.text,
    hours: durationOfTask.inHours,
    minutes: durationOfTask.minute,
  );

  for (var i in managableDaysIndexes!) {
    List<Task> tasksList =
        localDbService.rightBoxByCheckBoxId(i).values.toList();
    Duration remainingTime =
        ViewUtils.fullDay - viewUtils.calculateTotalDuration(tasksList);

    if (task.duration > remainingTime) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          backgroundColor: Colors.red,
          content: Text(
            vt.intl
                .of(context)!
                .fmt('error.selected_duration_more_than_remaining'),
          ),
        ),
      );
    } else {
      localDbService.rightBoxByCheckBoxId(i).add(task.copyWith(
            uniquekey: localDbService.rightTaskKeyCheckBoxId(i),
          ));
```

```
        ScaffoldMessenger.of(context).removeCurrentSnackBar();
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => Dashboard()),
        );
      }
    }
  }
}
```

Лістинг модуля програми set_up.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:vtime/core/cubits/preference_cubit.dart';
import 'package:vtime/core/utils/widgets.dart';
import 'package:vtime/view/dashboard.dart';

class AppSetup extends VTStatefulWidget {
  AppSetup({Key? key}) : super(key: key);

  @override
  _AppSetupState createState() => _AppSetupState();
}

class _AppSetupState extends VTState<AppSetup> {
  final langSegments = const <int, Widget>{
    0: Text('English'),
    1: Text('Türkçe'),
    2: Text('Русский'),
    3: Text('ქართული')
  };

  int themeSegmentedValue = 0;
  int langSegmentedValue = 0;

  @override
  Widget build(BuildContext context) {
    var themeSegments = <int, Widget>{
      0: Text(vt.intl.of(context)!.fmt('prefs.appearance.light')),
      1: Text(vt.intl.of(context)!.fmt('prefs.appearance.dark')),
      2: const Text('S/2')
    };

    return Scaffold(
      appBar: AppBar(title: Text(vt.intl.of(context)!.fmt('app.setup'))),
      floatingActionButtonLocation:
          FloatingActionButtonLocation.miniCenterFloat,
```

```
    floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.done), onPressed: completeSetup),
    body: SingleChildScrollView(
      padding: const EdgeInsets.all(20),
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const SizedBox(height: 130),
            themeSelecting(themeSegments),
            const SizedBox(height: 100),
            langSelecting(),
            const SizedBox(height: 30),
          ],
        ),
      ),
    ),
  );
}

Column themeSelecting(themeSegments) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Align(
        alignment: Alignment.centerLeft,
        child: Text(
          vt.intl.of(context)!.fmt('prefs.appearance'),
          style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
        ),
      ),
      const SizedBox(height: 15),
      SizedBox(
        width: 300,
        child: CupertinoSlidingSegmentedControl(
          padding: const EdgeInsets.all(0),
          groupValue: themeSegmentedValue,
          children: themeSegments,
          onValueChanged: (dynamic i) {
            changeTheme(i);
            setState(() => themeSegmentedValue = i);
          },
        ),
      ),
    ],
  );
}

Column langSelecting() {
  return Column(
```

```
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(
            vt.intl.of(context)!.fmt('prefs.lang'),
            style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
          ),
        ),
        const SizedBox(height: 15),
        SizedBox(
          width: 500,
          child: CupertinoSlidingSegmentedControl(
            padding: const EdgeInsets.all(0),
            groupValue: langSegmentedValue,
            children: langSegments,
            onValueChanged: (dynamic i) {
              changeLanguage(i);
              setState(() => langSegmentedValue = i);
            },
          ),
        ),
      ],
    );
  }

  void completeSetup() {
    changeTheme(themeSegmentedValue);
    changeLanguage(langSegmentedValue);

    Navigator.pushAndRemoveUntil(
      context,
      MaterialPageRoute(builder: (context) => Dashboard()),
      (route) => false,
    );
  }

  void changeLanguage(int i) {
    var args = {
      0: () => BlocProvider.of<PreferenceCubit>(context).changeLang('en'),
      1: () => BlocProvider.of<PreferenceCubit>(context).changeLang('tr'),
      2: () => BlocProvider.of<PreferenceCubit>(context).changeLang('ru'),
      3: () => BlocProvider.of<PreferenceCubit>(context).changeLang('ka'),
    };
    args[i]!.call();
  }

  void changeTheme(int i) {
    final values = {
      0: () {
```

```
        BlocProvider.of<PreferenceCubit>(context).changeTheme('default');
      },
      1: () {
        BlocProvider.of<PreferenceCubit>(context).changeTheme('dark');
      },
      2: () {
        BlocProvider.of<PreferenceCubit>(context).changeTheme('s/2');
      }
    };

    values[i]!.call();
  }
}
```

Лістинг модуля програми custom_swithers.dart

```
import 'package:flutter/cupertino.dart';
import 'package:vtime/core/utils/widgets.dart';
import 'package:vtime/view/widgets/utils.dart';

class SwitcherTile extends VTStatelessWidget {
  final String title;
  final bool switcherValue;
  final Function(bool) onChanged;
  final Color? switcherColor;
  final double spaceAround;
  final MainAxisAlignment mainAxisAlignment;
  final TextStyle titleStyle;

  SwitcherTile({
    Key? key,
    required this.title,
    required this.switcherValue,
    required this.onChanged,
    this.switcherColor,
    this.spaceAround = 10,
    this.mainAxisAlignment = MainAxisAlignment.center,
    this.titleStyle = const TextStyle(fontWeight: FontWeight.bold),
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: mainAxisAlignment,
      children: [
        Align(
          alignment: Alignment.centerLeft,
          child: Text(title, style: titleStyle),
        ),
        SizedBox(width: spaceAround),
```

```
        CupertinoSwitch(
          activeColor: switcherColor ?? ViewUtils().pomodoroOrange(context),
          value: switcherValue,
          onChanged: onChanged,
        ),
      ],
    );
  }
}
```

Лістинг модуля програми buttons.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:vtime/core/cubits/preference_cubit.dart';
import 'package:vtime/core/utils/widgets.dart';

class SaveButton extends VTStatelessWidget {
  final String title;
  final Function onTap;

  SaveButton({
    Key? key,
    required this.title,
    required this.onTap,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Transform.translate(
      offset: Offset(0, -1 * MediaQuery.of(context).viewInsets.bottom),
      child: Padding(
        padding: const EdgeInsets.symmetric(vertical: 30),
        child: FractionallySizedBox(
          widthFactor: .8,
          child: ElevatedButton(
            onPressed: () => onTap(),
            child: Text(
              title,
              style: BlocProvider.of<PreferenceCubit>(context)
                  .state
                  .theme!
                  .textTheme
                  .button,
            ),
          ),
        ),
      ),
    );
  }}
```

Додаток Г Ілюстративна частина

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ СТВОРЕННЯ МЕСЕНДЖЕРА З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

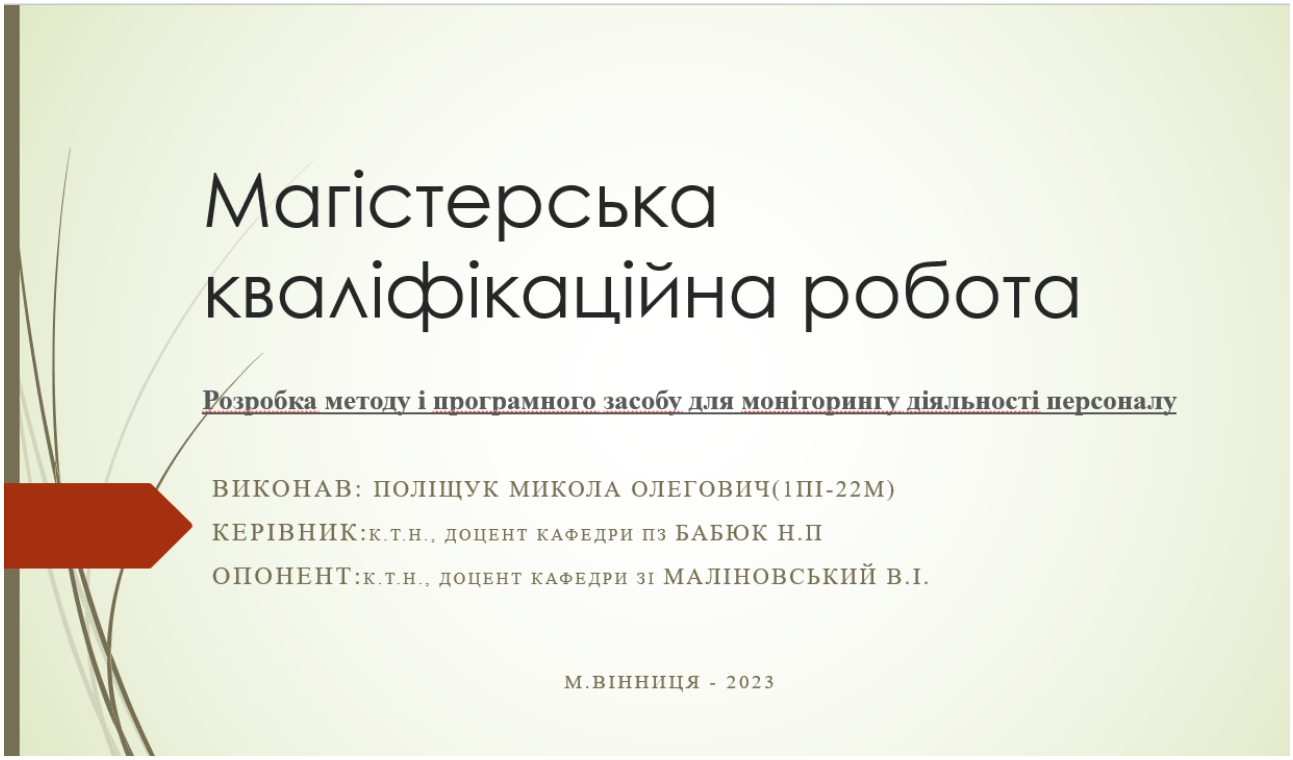

Рисунок Г.1 - Слайд презентації №1

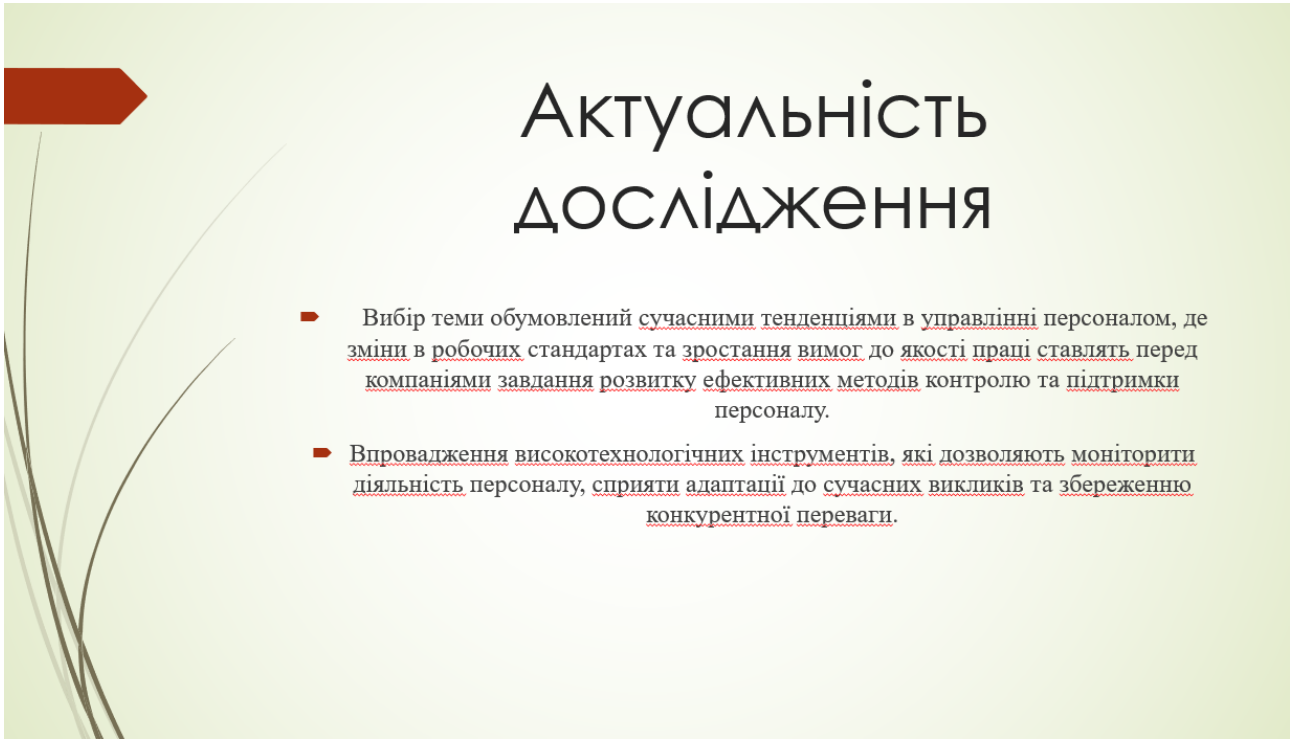

Рисунок Г.2 - Слайд презентації №2

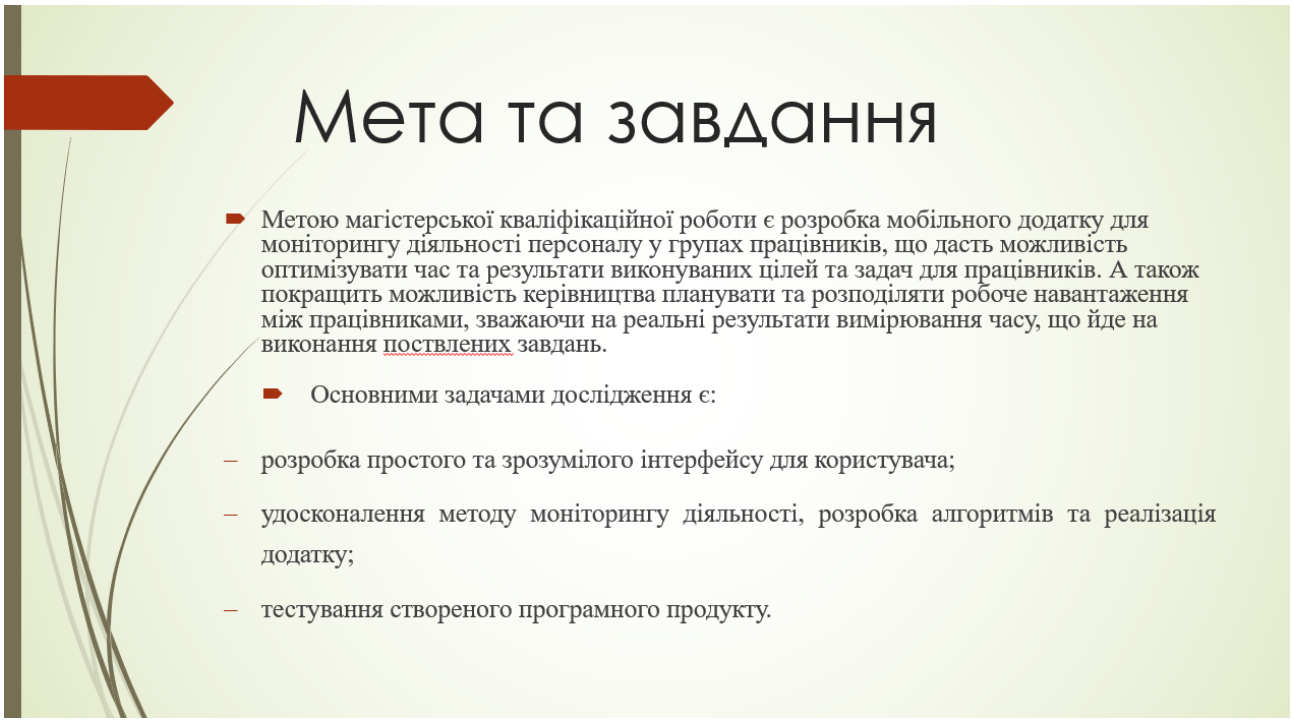

Рисунок Г.3 - Слайд презентації №3

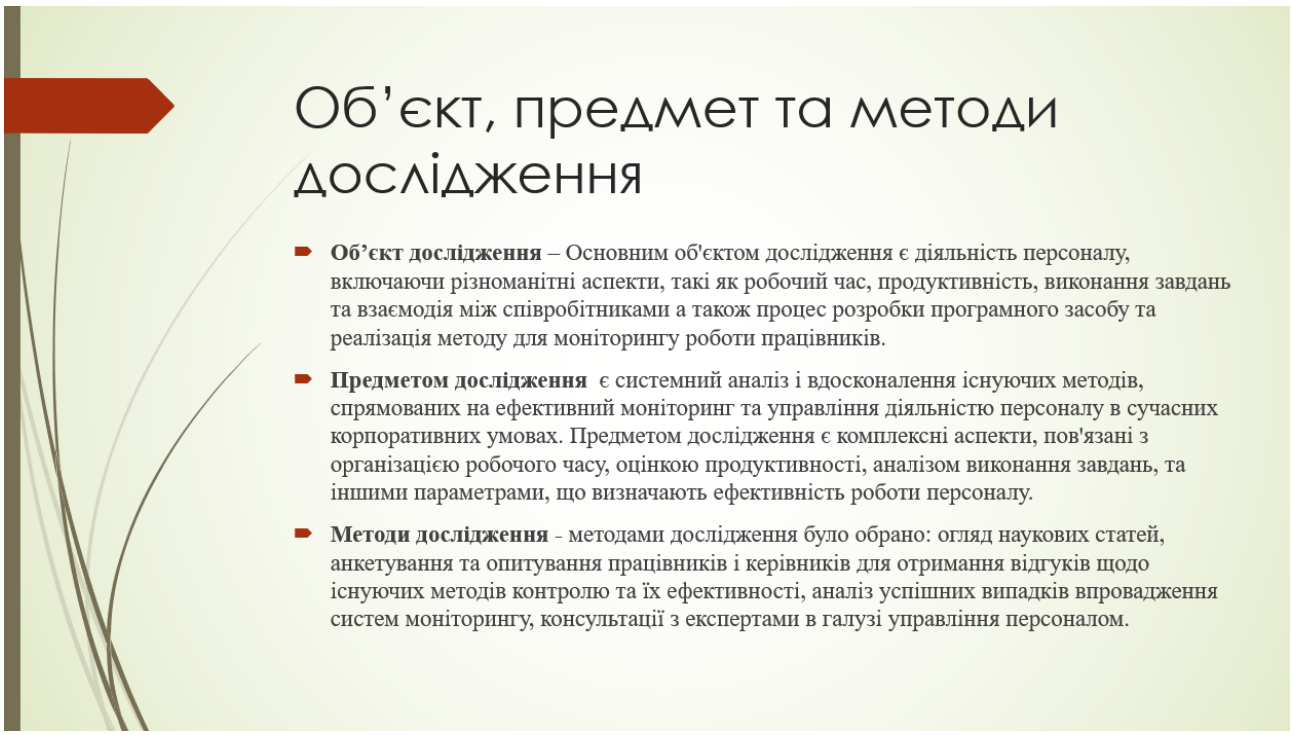

Рисунок Г.4 - Слайд презентації №4

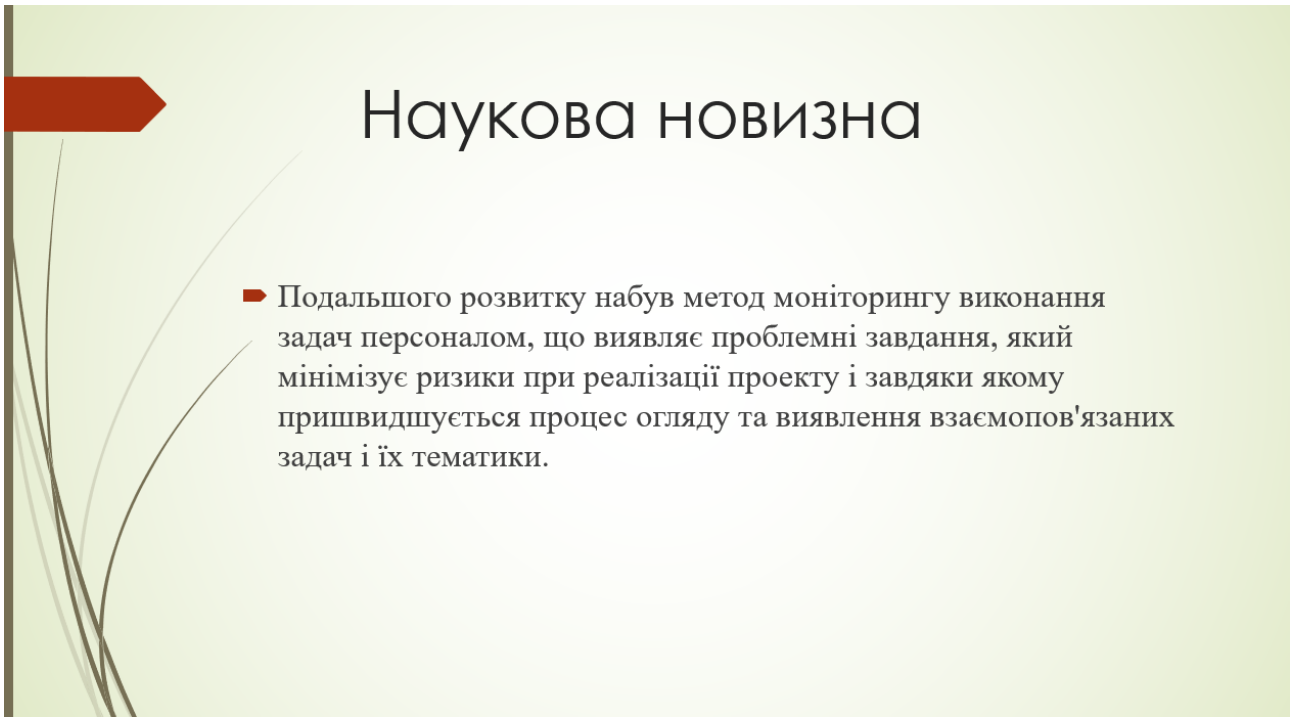

Рисунок Г.5 - Слайд презентації №5

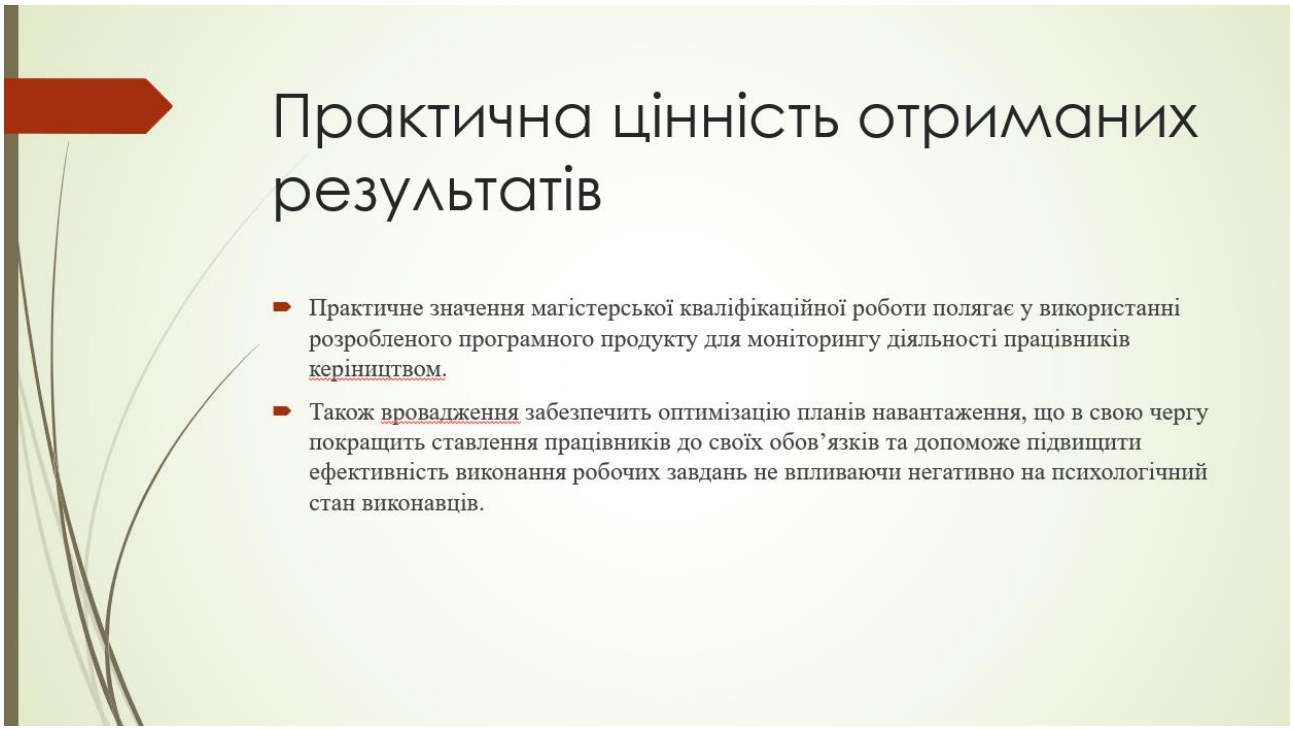

Рисунок Г.6 - Слайд презентації №6

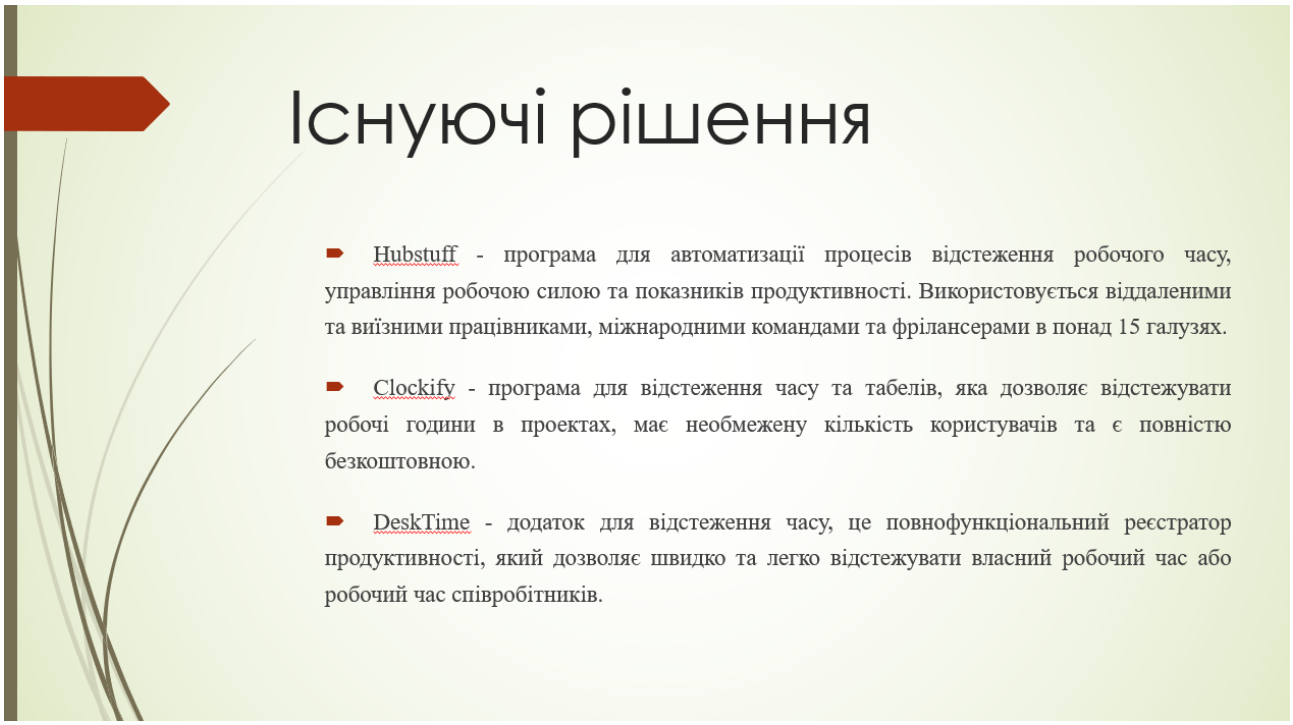

Рисунок Г.7 - Слайд презентації №7

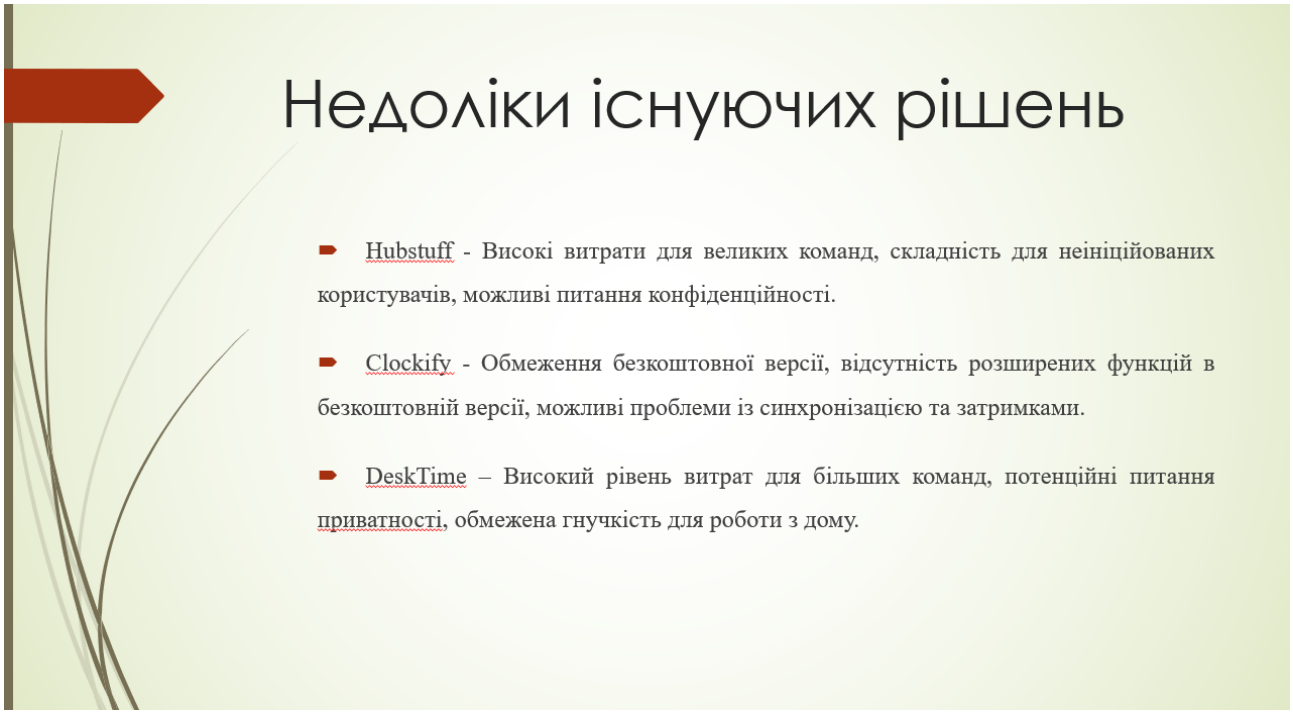

Рисунок Г.8 - Слайд презентації №8

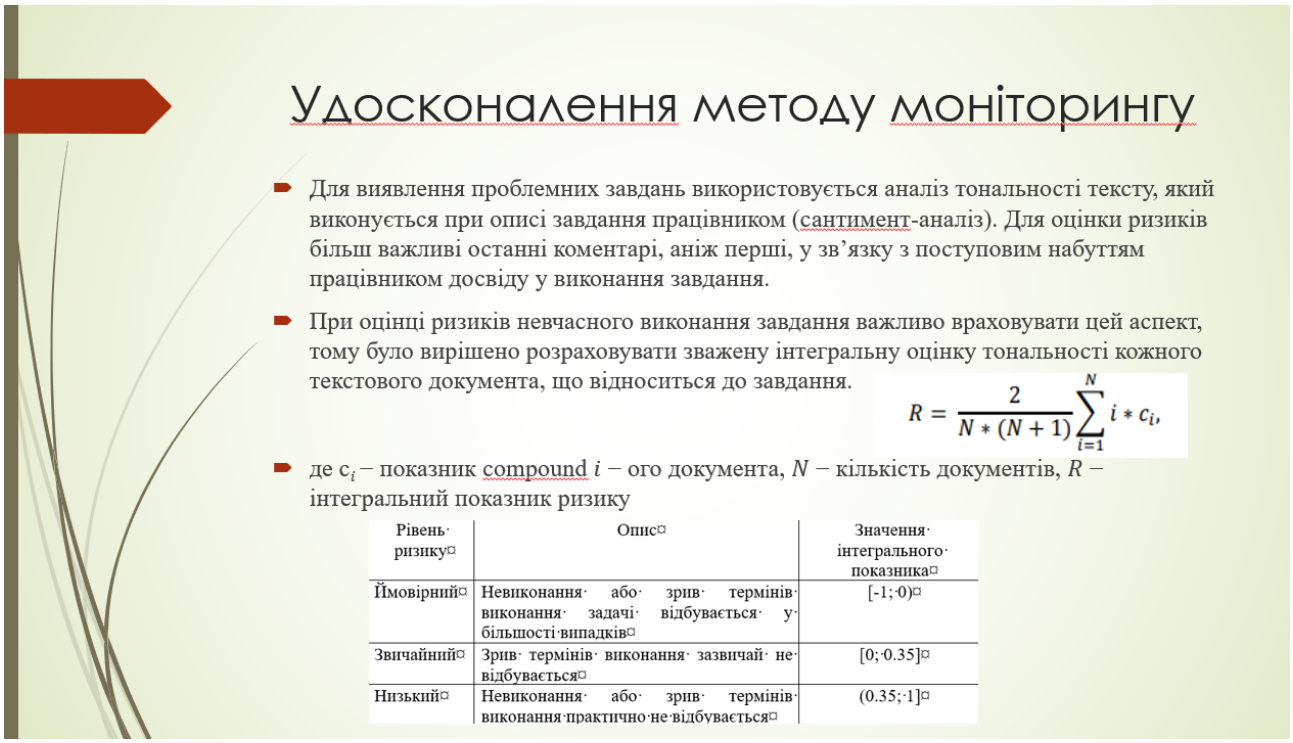

Рисунок Г.9 - Слайд презентації №9

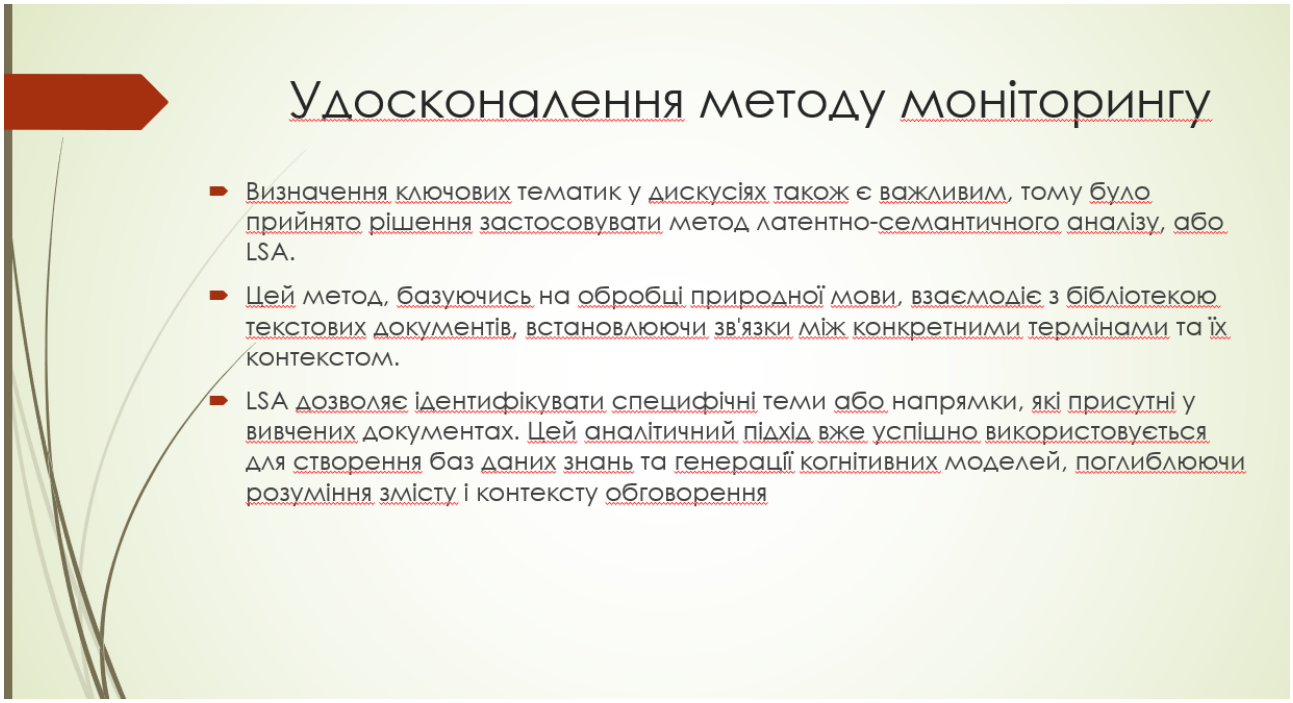

Рисунок Г.10 - Слайд презентації №10

Удосконалення методу моніторингу

- Латентно-семантичний аналіз - це комплексний процес з декількох ключових етапів:
- Перший етап - підготовка корпусу документів (розбиття на лексеми, очиска від шуму та поверненя до початкофої форми слів)
- Другий етап - створення терм-документної матриці та виявлення глибоких залежностей між термінами(створюється терм-документна матриця, вирахування ваги слів)
- У подальшому процесі аналізу використовується метод сингулярного розкладу для детального розгляду отриманої терм-документної матриці. Цей метод не просто допомагає розкрити потенційні відносини між документами, але й дозволяє глибше зануритися у контекст та семантику текстового матеріалу(можна виявити, які терміни тісно пов'язані між собою в різних документах).

Рисунок Г.11 - Слайд презентації №11

Діаграма послідовності додатку

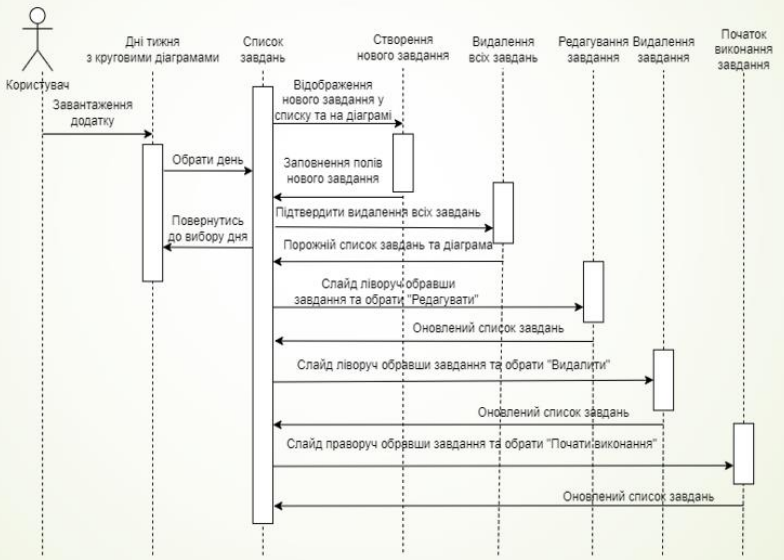

Рисунок Г.12 - Слайд презентації №12

Рисунок Г.13 - Слайд презентації №13

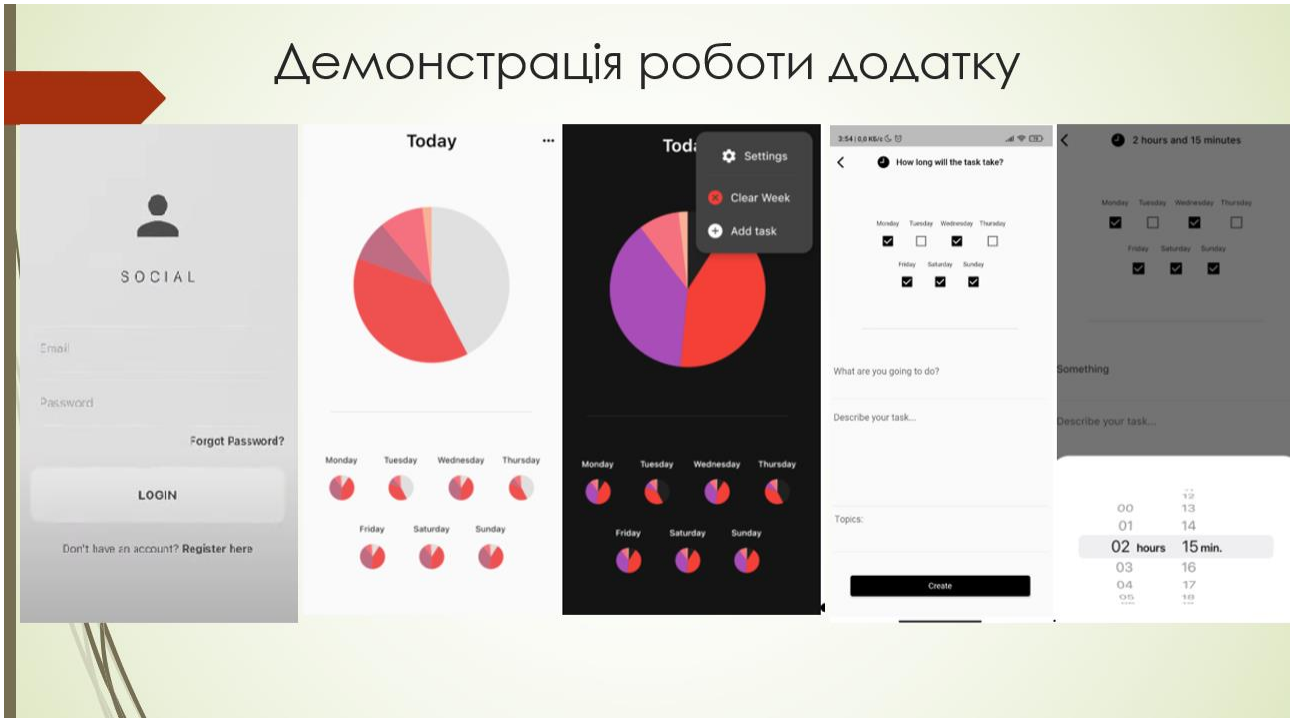

Рисунок Г.14 - Слайд презентації №14

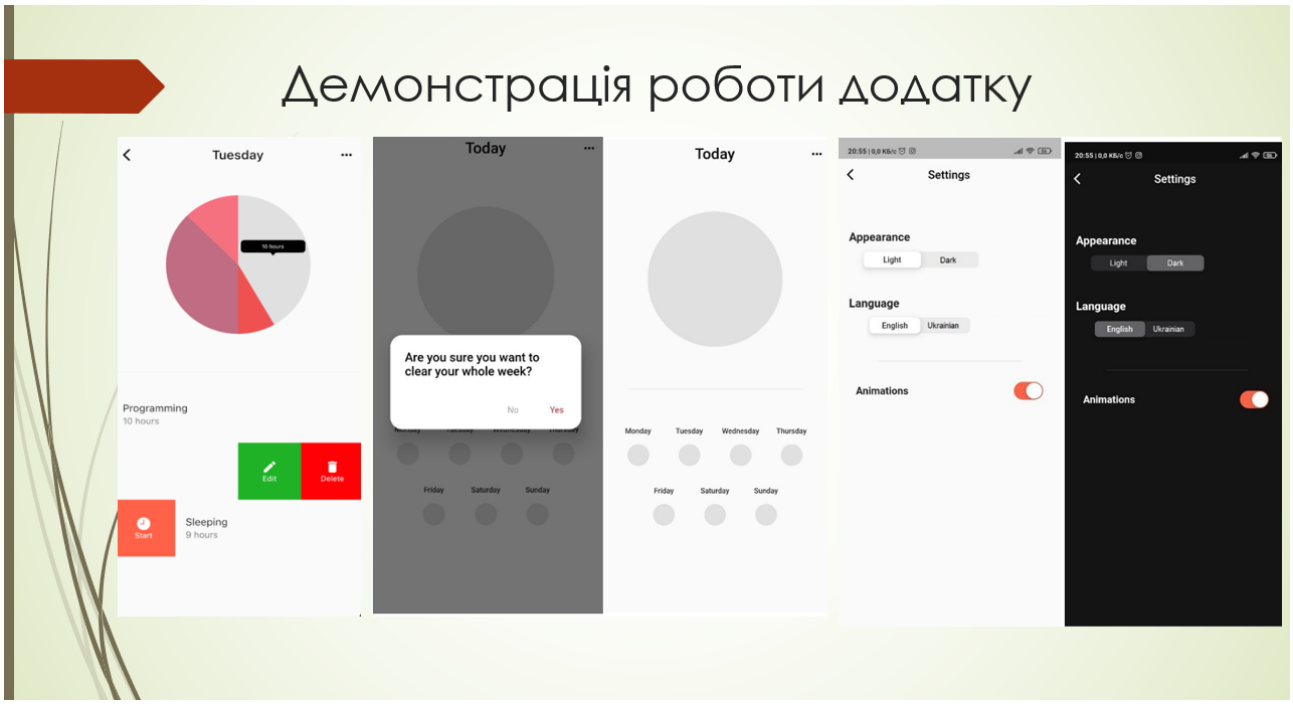

Рисунок Г.15 - Слайд презентації №15

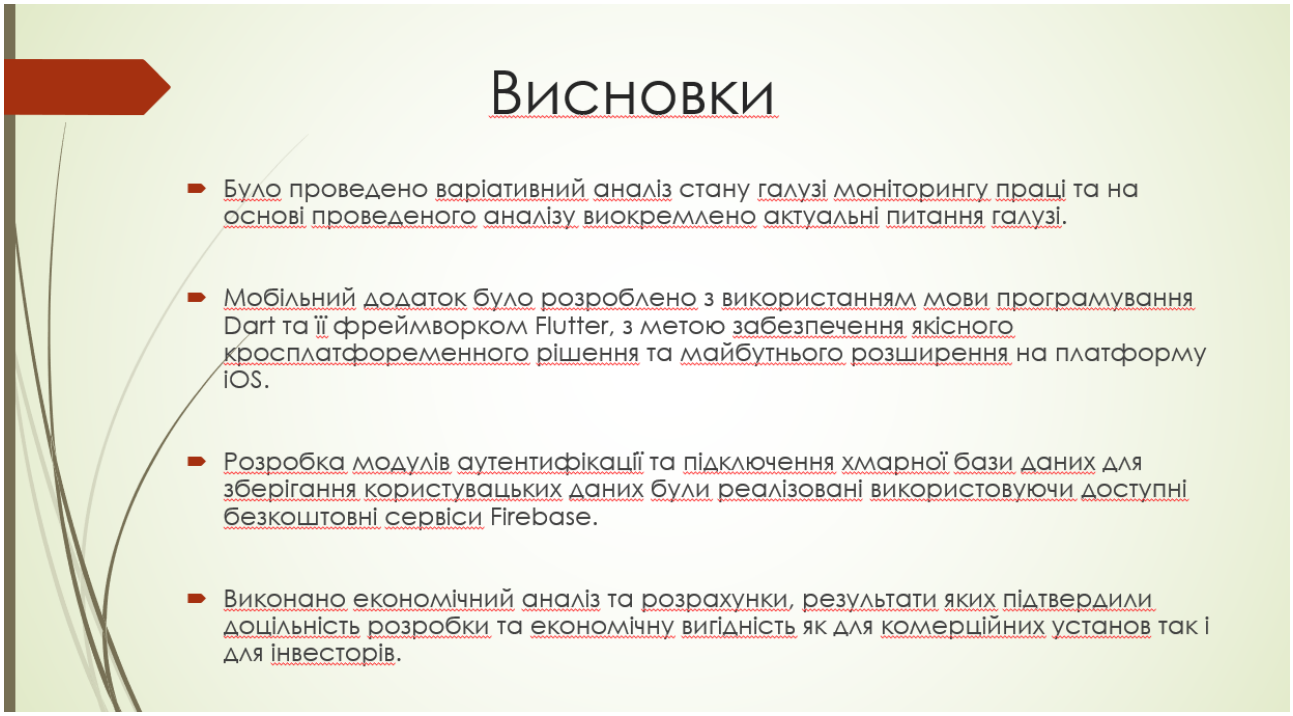

Рисунок Г.16 - Слайд презентації №16

Рисунок Г.17 - Слайд презентації №17