Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних.»

Виконав: студент 2 курсу,
групи 2АКІТ-22м спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

Владислав ПОДОЛЯНИН .
Ім'я ПРІЗВИЩЕ

Керівник: к.т.н., доцент, доцент кафедри КСУ
степінь, вчання, посада
Олег КОВАЛЮК
Ім'я ПРІЗВИЩЕ
«01» 12 2023 р.

Опонент: к.т.н., доцент каф. АІІТ
Роман МАСЛІЙ
Ім'я ПРІЗВИЩЕ
«06» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
_Марія ЮХИМЧУК
«07» 12 2023

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Подолянину Владиславу Іванович .
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних

керівник роботи Ковалюк Олег Олександрович
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи " 1 " грудня 2023 року
3. Вихідні дані до роботи: можливість використання мобільного додатку на різних пристроях андроїд; база даних спроектована спеціально під мобільний додаток; можливість записувати, редагувати, та видаляти задачі до моменту виконання
4. Зміст текстової частини: вступ, аналіз предметної області та огляд аналогів, огляд структури технологій та функціональності мобільного додатку, проєктування інтерфейсу та бази даних
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): об'єкт, предмет та мета дослідження, задачі дослідження, блок-схема порядку дій, основні системи управління і обліку завдань, середовище тестування Android Studio, зовнішній вигляд мобільного додатку, тестування програми, економічна частина, висновки

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Бурєннікова Н. В., д.е.н., професор кафедри ЕПВМ.		

2. Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз мов програмування для розробки мобільних додатків	10.10. 2023	11.10.2023	
2	Аналіз мов програмування для розробки бази даних для мобільного додатку	12.10. 2023	14.10.2023	
3	Огляд та аналіз аналогів у сфері розподілу та обліку завдань	15.10. 2023	24.10.2023	
4	Розробка дизайну інтерфейсу додатку, та огляд функціональності додатку	25.10. 2023	29.10.2023	
5	Розробка і тестування програмного забезпечення та баз даних	30.10.2023	04.11.2023	
6	Розрахунок економічної частини	05.11.2023	20.11.2023	
7	Графічні матеріали	21.11.2023	27.11.2023	

Студент Владислав ПОДОЛЯНИН
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи Олег КОВАЛЮК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 621.374.415

Подолянин В. І. Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2023. На укр. мові.: 125ст., 55 джерел, 4 розділи.

Ця магістерська робота присвячена розробці системи планування та обліку завдань, яка має на меті полегшити і удосконалити управління завданнями в різних сферах діяльності. Основний акцент роботи робиться на розробці дизайну програмного забезпечення та відповідної бази даних для ефективного функціонування системи.

Робота включає детальний аналіз та розробку інтуїтивно зрозумілого та ефективного користувацького інтерфейсу, що дозволяє з легкістю планувати та відстежувати завдання. Також включає реалізацію різноманітних функцій, таких як планування завдань, пріоритезація, нагадування та спрощений облік прогресу завдань.

Важливою складовою є проектування та впровадження ефективної бази даних для зберігання та управління інформацією про завдання, забезпечуючи швидкий доступ та надійність.

Ця робота не лише спрямована на теоретичне вивчення проблеми, але й на практичну реалізацію розробленої системи. Висновки, які надаються у роботі, відображають важливий внесок у сучасні технології та можуть служити основою для подальших досліджень у цьому напрямку.

Дизайн додатку розроблено за допомогою програмного забезпечення Figma, база даних для програми розроблена за допомогою бібліотеки Room.

Ключові слова: автоматизована система, управління, планування, мобільний додаток, база даних, Figma.

ANNOTATION

UDC 621.374.415

V. I. Podolianyn. Development of an automated system for planning and accounting tasks. Part 1. Mobile application and database design development. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023. In Ukrainian. languages: 125 p., 55 sources, 4 chapters.

This master's thesis is devoted to the development of a task planning and accounting system, which aims to facilitate and improve task management in various fields of activity. The main emphasis of the work is on the development of the software design and the corresponding database for the efficient functioning of the system.

The work includes detailed analysis and development of an intuitive and efficient user interface that allows for easy planning and tracking of tasks. It also includes the implementation of various functions such as task scheduling, prioritization, reminders and simplified task progress accounting.

An important component is the design and implementation of an effective database for storing and managing task information, ensuring rapid access and reliability.

This work is aimed not only at the theoretical study of the problem, but also at the practical implementation of the developed system. The conclusions presented in the work reflect an important contribution to modern technologies and can serve as a basis for further research in this direction.

The design of the application was developed using the Figma software, the database for the application was developed using the Room library.

Keywords: automated system, management, planning, mobile application, database, Figma.

ЗМІСТ

ВСТУП

Актуальність. У сучасному темпі життя, який характеризується стрімким технологічним розвитком та постійними змінами, на передній план виходить потреба у максимальній ефективності та оптимізації робочих процесів. В умовах постійного руху та завдань, які розвиваються високою швидкістю, забезпечення ефективного планування та раціонального управління завданнями стає завданням першочергового значення.

Однією з ключових відповідей на цей виклик стає розробка автоматизованих систем планування завдань[1], спрямованих на використання в сучасних смартфонах. Ці рішення не просто відзеркалюють сучасні реалії, де смартфони стали невід'ємною частиною повсякденного життя, але й активно сприяють підвищенню продуктивності та організації робочого та особистого часу.

У світлі нестабільного ритму сучасного життя та надзвичайно великого обсягу завдань інтелектуальні системи планування визначаються як ключовий елемент оптимізації та успішного вирішення рутинних справ. Вони виявляються не лише ефективними інструментами для створення та відстеження розкладу завдань, але й активною підтримкою, що надає користувачам цінні рекомендації, оптимальні рамки для виконання завдань, аналізує їх продуктивність та вносить рекомендації для покращення управління часом.

В світі, де технології розвиваються на величезній швидкості, інтелектуальні системи планування знаходять своє місце як необхідний керований елемент справжнього успіху та високої продуктивності.

Мета роботи. Основною метою роботи є покращити можливості та комфорт користувачів у плануванні завдань.

Завдання роботи.

- Провести аналіз систем планування та обліку завдань;
- Провести аналіз технологій та функціональності мобільного додатку;

- Виконати проєктування інтерфейсу та бази даних;
- Оцінити економічні показники розробки.

Об'єкт дослідження. Об'єктом дослідження є процеси планування та обліку завдань в умовах сучасного інформаційного середовища, зокрема, на платформі смартфонів.

Предмет дослідження. Предметом дослідження є розробка дизайну програми та бази даних для автоматизованої системи, спрямованої на зручне та ефективне планування та відстеження завдань.

Новизна роботи. Інноваційність розробки полягає у зручному візуальному представленні запланованих завдань для смартфонів, що визначає новизну цієї роботи. Інноваційний підхід до інтерфейсу користувача та оптимізована база даних роблять систему відмінною від існуючих аналогів.

Практична цінність. Розроблена система буде корисною для широкого кола користувачів, які прагнуть до підвищення своєї продуктивності та ефективності управління завданнями, особливо в умовах активного використання смартфонів.

Цінність роботи полягає в розробці програмного забезпечення яке може покращити можливості та комфорт користувачів у плануванні завдань.

Апробація. Результати дослідження та розроблену систему апробовано шляхом тестування на реальних користувачах та отримання їхнього фідбеку. Це дозволить вдосконалити систему та забезпечити відповідність її функціоналу реальним потребам користувачів.

Публікації: Подолянин В. І.., Ковалюк О.О. «Короткий огляд розробки автоматизованої системи планування і обліку завдань», «LIII Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету НТКП-ВНТУ», 2023.

1 АНАЛІЗ СИСТЕМ ПЛАНУВАННЯ ТА ОБЛІКУ ЗАВДАНЬ. ОГЛЯД АНАЛОГІВ

1.1 Аналіз проблеми автоматизованої системи планування та обліку завдань

Проблематика автоматизованих систем планування та обліку завдань виникає внаслідок різних факторів, які включають технічні обмеження, соціально-психологічні аспекти та вимоги сучасного способу життя. Ось кілька ключових аспектів проблематики:

1. Технічні обмеження:

- *Сумісність та інтеграція:* Багато існуючих систем не завжди ефективно інтегруються з іншими популярними інструментами (наприклад, календарями, електронною поштою[2]), що може обмежувати їхню корисність.
- *Обмеження платформ[3]:* Деякі системи можуть бути обмежені певними платформами або не мати зручного інтерфейсу для мобільних пристроїв.

2. Соціально-психологічні аспекти:

- *Важкість впровадження:* Користувачі можуть виявляти опір впровадженню нових систем, особливо якщо вони не є інтуїтивними та не надають очевидних переваг у порівнянні із старими методами планування.
- *Стрес та надмірне завантаження:* Недостатній контроль над завданнями та перевантаженість інформацією можуть викликати стрес та погіршити продуктивність.

3. Вимоги сучасного способу життя:

- *Мобільність:* Зростання мобільності сучасних користувачів вимагає наявності зручних та ефективних засобів управління завданнями на мобільних пристроях.
- *Індивідуалізація:* Кожен користувач має власні потреби та пріоритети, тому інструменти планування повинні бути гнучкими та індивідуалізованими.

4. Призначення та ефективність:

- *Недостатня ефективність:* Деякі системи можуть виявитися надто складними або недостатньо ефективними, що робить їх менш затребуваними серед користувачів.
- *Відсутність адаптації[4]:* Системи не завжди здатні адаптуватися до змін у структурі завдань або особистих пріоритетів користувача.

Розв'язання цих проблем передбачає не лише розробку технічно ефективних систем, але й врахування соціокультурних аспектів[5] та потреб користувачів. Ефективна автоматизована система планування повинна бути інтегрованою, легко використовуваною та спрямованою на покращення продуктивності та якості життя користувачів.

1.2 Аналіз предметної області розробки бази даних мобільного додатку

В епоху стрімкого технологічного розвитку мобільні телефони стали невід'ємною частиною сучасного життя, перетворивши спосіб, яким люди взаємодіють із світом. Завдяки їхній портативності[6], потужності та мережевій доступності, мобільні телефони визначають новий стандарт для зручності та швидкості в області інформаційних технологій.

З кожним роком мобільні телефони здобувають все більше популярності, стаючи надійними компаньйонами у повсякденних справах та вирішенні різноманітних завдань. Їх відмінна обробка інформації, функціональність та мобільність зробили їх невід'ємною частиною життя мільйонів людей по всьому світу.

Все більше користувачів віддають перевагу використанню смартфонів для доступу до Інтернету[7], спілкування, вирішення завдань та розваг. Інноваційні технології, які вони вміщують, роблять їх не лише засобом зв'язку, але і мобільним інформаційним центром, який об'єднує в собі широкий спектр функцій.

Зростання залежності від мобільних телефонів породжує потребу у високоефективних програмах, призначених для розв'язання різних завдань. В умовах активного темпу життя та постійної потреби в зручних інструментах, розробка програм саме для мобільних телефонів стає нагальною задачею.

Розробка програм для мобільних телефонів відкриває безмежні можливості для полегшення щоденних завдань, покращення продуктивності та забезпечення нових можливостей розваг. Завдяки високій доступності та потужним функціоналом, такі програми можуть перетворити смартфон в особистий асистент[8].

У цьому контексті, розробка програм для мобільних телефонів стає стратегічно важливою, забезпечуючи користувачам зручний та ефективний інструментарій для різних потреб. Високий рівень мобільності, поєднаний з потужністю сучасних смартфонів, визначає цю область розробки як ключову для подальшого розвитку технологій та вдосконалення користувацького досвіду.

Часто для додатку потрібно використовувати базу даних. Для цього використовують SQLite.

SQLite є вбудованою реляційною базою даних, яка часто використовується в Android-додатках. Для спрощення роботи з SQLite використовують бібліотеку Room Persistence, яка надає абстракції та зручний інтерфейс[9].

Room - це бібліотека для роботи з базами даних SQLite в мові програмування Kotlin, яка входить в склад Android Jetpack. Вона надає високорівневий доступ до SQLite, спрощуючи роботу з базами даних у мобільних додатках. [10]

Він має такі переваги:

Підтримка Kotlin:

Room оптимізована для використання в мові програмування Kotlin[11], що робить її зручною та ефективною для розробників Kotlin.

Компактність та Простота:

Бібліотека дозволяє використовувати анотації для визначення сутностей та запитів, що робить код компактним та легким для розуміння.

Підтримка Асинхронних Операцій[12]:

Room автоматично використовує асинхронні запити, що сприяє ефективній роботі з базою даних, не блокуючи основний потік виконання.

Перевірка Під Час Компіляції:

Запити до бази даних перевіряються під час компіляції[13], що дозволяє виявляти помилки на етапі розробки.

Для розробки дизайну для програм використовують інструмент для дизайну Figma. Це онлайн-інструмент для дизайну та прототипування, який дозволяє командам спільно працювати над проектами. Він використовується для створення веб-дизайну, мобільних додатків, інтерфейсів користувача та інших дизайн-завдань.[14]

Інтерфейс Користувача Figma:

- Канвас: Це робоча область, де створюється та редагується дизайн.[15]
- Інструменти: Figma має різні інструменти для створення фігур, ліній, тексту, іконок та інших елементів дизайну.
- Бібліотеки: Можна використовувати власні компоненти та стилі або імпортувати готові бібліотеки.[16]

Компоненти та Композиції:

- Figma сприяє використанню компонентів, що полегшує створення повторюваних елементів і дозволяє легко змінювати їхні стилі та властивості.
- Композиції дозволяють створювати складніші елементи, об'єднуючи групи компонентів в один.

Прототипування:

- Figma надає інструменти для створення прототипів та взаємодії між екранами. Можна додавати переходи, анімацію та визначати поведінку додатка.[17]

Робота в Команді:

- Figma - це хмарна платформа, що дозволяє різним членам команди одночасно працювати над проектом. Зміни в реальному часі, коментарі та можливість спільного редагування полегшують командну роботу.[18]

Експорт та Успішна Імплементація:

- Після завершення дизайну можна експортувати свої роботи в різні формати (PNG, SVG)[19][20] або навіть експортувати код CSS[21].

Відгуки та Тестування:

- Figma дозволяє збирати відгуки від команди або клієнтів, а також тестувати прототипи перед реалізацією.

Інтеграція з Іншими Інструментами:

- Figma може інтегруватися з іншими популярними інструментами для дизайну та розробки, такими як Zeplin[22] або Adobe XD.

Figma став популярним інструментом для розробки мобільних додатків, завдяки своїм потужним функціоналом та можливостям спільної роботи. Він полегшує весь процес від ідеї до реалізації дизайну.

Також окрім Figma часто використовують Adobe XD, який ще старшим аналогом, але в останній час віддає свої позиції молодшій програмі.

Adobe XD використовують як інструмент для розробки прототипів та дизайну інтерфейсів користувача (UI) від Adobe. Він спроектований для дизайнерів, розробників і всіх, хто працює з мобільними додатками та веб-сайтами.[23] Ось кілька ключових рис Adobe XD:

Прототипування:

Adobe XD надає можливість швидко створювати інтерактивні прототипи для перевірки функціоналу та взаємодії елементів інтерфейсу.

Дизайн інтерфейсу:

Інструменти для створення векторного дизайну дозволяють легко створювати іконки, кнопки, макети та інші елементи інтерфейсу.

Анімація:

Adobe XD підтримує створення анімацій для переходів між екранами та інші ефекти для покращення користувацького досвіду.

Загальна взаємодія з Adobe Creative Cloud[24]:

Легка інтеграція з іншими продуктами Adobe, такими як Photoshop[25] та Illustrator[26], для зручного обміну ресурсами і проектами.

Бібліотеки ресурсів:

Зручна можливість створення та використання бібліотек ресурсів для швидкого доступу до стандартних елементів дизайну.

Спільна робота:

Adobe XD дозволяє спільно працювати над проектами у реальному часі, що дозволяє командам швидше взаємодіяти та вносити зміни.

Відстеження змін:

Можливість відстежувати зміни у проекті, щоб легко повертатися до попередніх версій дизайну.

Adobe XD є популярним вибором серед дизайнерів та розробників завдяки його інтуїтивному інтерфейсу, зручності використання та потужному функціоналу для створення сучасних та інтерактивних інтерфейсів.

1.3 Аналіз існуючих автоматизованих систем планування у сфері мобільних додатків

1.3.1 Аналіз історичного розвитку планувального спорядження

Історія розвитку методів планування включає в себе різні засоби та підходи, які еволюціонували разом із змінами технологій та суспільних умов. Нижче наведені основні етапи розвитку.

- Папір та Олівець:

Початок історії планування пов'язаний з використанням звичайного паперу та олівця. Люди записували свої плани, завдання та нагадування, використовуючи різні типи записних книг і блокнотів. Цей простий, але ефективний спосіб планування забезпечував фізичну форму розпису та взаємодії зі словами, що визначало особистий характер інформації. З часом цей традиційний метод планування став інтегруватися з сучасними технологіями, розширюючи можливості та функціоналість планування в онлайн-середовищі. Однак коріння цього процесу залишаються в практиці використання паперу та ручки, що відображає не лише спосіб фіксації завдань, але й культурні та історичні аспекти планування.

- Персональні Органайзери:

У другій половині 20-го століття виникла ера персональних органайзерів, що стала ключовим кроком у розвитку методів планування та організації життя. Філософія цих органайзерів базувалася на ідеї об'єднання різних функцій, таких як календар, списки завдань і контакти, в одному компактному пристрої. Два видатних представника цього періоду – Filofax[27] та Day-Timer[28] – стали справжніми символами організації життя.

Filofax заснований у Великобританії у 1921 році, Filofax став першим широко використовуваним органайзером. Його унікальність полягала в можливості використання різних блокнотів та вкладишів для персоналізації залежно від потреб користувача. Цей системний підхід дозволяв людям вести детальні записи, планувати завдання та створювати власні системи відстеження.

Day-Timer у 1947 році в США виник Day-Timer – перший органайзер, який визначав стандарт "день на сторінці". Ця концепція дозволяла користувачам детально планувати свої денні завдання та події. Day-Timer став популярним завдяки своїй компактності та зручності в веденні записів.

Персональні органайзери цього часу забезпечили ефективні інструменти для керування часом та завданнями, допомагаючи людям у веденні бізнесу, управлінні

робочими та особистими обов'язками. Спроектовані для покращення продуктивності та організації, вони стали популярними символами стилю та функціональності в робочому та особистому житті.Електронні Комп'ютерні

- Програми:

У 80-90-х роках 20-го століття виникла нова ера електронних органайзерів та програмного забезпечення для керування завданнями. Під впливом технологічних інновацій та розвитку персональних комп'ютерів, було створено декілька відомих програм, спрямованих на покращення організації часу та завдань користувачів.

Microsoft Schedule+[29] розроблений Microsoft, Schedule+ став частиною серії продуктів Microsoft Office і надавав користувачам інтегрований підхід до планування та керування завданнями.

Lotus Organizer[30] відомий як один з перших електронних органайзерів, Lotus Organizer був частиною популярного пакету програм Lotus. Він включав календар, списки завдань, контактні менеджери та інші інструменти для організації інформації.

DayTimer Organizer програмне забезпечення від компанії Day-Timer, що розробляла популярні паперові органайзери, було адаптоване для електронного використання. Воно надавало схожі можливості, але в електронному форматі.

Ці програми забезпечували користувачів електронними інструментами для планування, замінюючи або доповнюючи традиційні паперові методи. Вони визначили перехід від ручного ведення записів до використання електронних пристроїв для керування часом і завданнями.

- Початок Інтернету:

У 2000-х роках з появою Інтернету та використанням онлайн-технологій виникло безліч онлайн сервісів для керування завданнями та планування часу. Деякі з них стали популярними і використовуються й дотепер.

Remember The Milk це онлайн сервіс який став популярним завдяки своїй простоті та можливостям синхронізації з різними пристроями. Він дозволяє користувачам створювати списки завдань, нагадування та планувати події.

Toodledo надає розширені можливості для організації завдань, включаючи пріоритети, теги, терміни та інші аспекти. Сервіс також пропонує аналітичні засоби для відстеження продуктивності.

Google Calendar запущений у 2006 році, Google Calendar вирізняється своєю інтеграцією з іншими сервісами Google та можливістю ділитися календарями з іншими користувачами. Завдяки хмарній синхронізації, дані доступні на різних пристроях. [31]

Apple Calendar (відомий раніше як iCal) створений ще в 2002 році, входить у склад операційних систем Apple. Він вирізняється інтеграцією з екосистемою Apple та іншими додатками, що забезпечує зручне управління подіями та завданнями для користувачів пристроїв Apple. [32]

Два останніх календарі вбудовані в екосистеми Google та Apple відповідно, що дозволяє користувачам зручно взаємодіяти з ними та використовувати їх у поєднанні з іншими сервісами та додатками від обраного виробника. Вони надають не лише можливості планування, а й ефективного управління часом, адаптовані до конкретних екосистем та потреб користувачів.

- Клієнтські Додатки та Хмарові Сервіси:

З розвитком технологій хмарового зберігання, такі сервіси, як Evernote[33], Todoist[34], Asana[35], Trello[36] та інші, надають користувачам зручні інструменти для ведення списків завдань, створення проектів та співпраці в команді.

Evernote дозволяє зберігати та організовувати нотатки, файли та фотографії. Todoist — зручний для створення списків завдань та їх відстеження, а також має синхронізацію на різних платформах. Asana використовується для управління проектами, планування завдань та співпраці в командах. Trello пропонує організацію завдань у вигляді дошок та карток, полегшуючи керування проектами

та командною роботою. Ці інструменти спрямовані на забезпечення ефективного планування та відстеження завдань на різних рівнях — від особистих списків до групових проектів.

- Інтелектуальні Помічники та Штучний Інтелект[37]:

Останнім часом розвиваються інтелектуальні помічники та додатки з елементами штучного інтелекту. Наприклад, вони можуть пропонувати оптимальний час для виконання завдань, аналізувати продуктивність та робити рекомендації для кращого управління часом. Вони не обмежуються простою збереженістю та відстеженням завдань, але й впроваджують інтелектуальні функції для оптимізації управління часом та підвищення продуктивності. Такі додатки можуть аналізувати користувачів та пропонувати оптимальний час для завдань, відстежувати продуктивність, надавати персоналізовані поради та адаптуватися до змін у розкладі. Інтеграція елементів штучного інтелекту розширює можливості додатків, забезпечуючи більш інтелектуальний та персоналізований підхід до управління часом. Ці інновації сприяють підвищенню ефективності та зручності використання таких додатків.

Зараз люди мають доступ до широкого спектру інструментів для планування, які можна використовувати на різних етапах розвитку технологій та особистих потреб користувачів. Засоби планування стали не тільки ефективними, а й більш адаптованими до сучасного способу життя та роботи.

1.3.2 Аналіз аналогів

Any.do

Програма має простий зовнішній вигляд та функціонал, що полегшує роботу з нею для пересічного користувача. Вона дозволяє ставити задачі на певний день та час і створювати нагадування.[38]

Ось основні можливості програми:

- Безпечно синхронізується між мобільним телефоном, робочим столом, Інтернетом і планшетом. Завжди синхронізує всі списки справ, завдання, нагадування, календар і порядок денний.
- Нагадування на запланований час, або на прибуття до певного місця.
- Можливість працювати разом зі спільними списками та призначеними завданнями, щоб краще співпрацювати.
- Можливість синхронізації в режимі реального часу з календарем свого телефону, календарем Google, подіями Facebook, календарем Outlook або будь-яким іншим календарем.
- Список завдань Any.do також чудово підходить для покупок у продуктовому магазині. Прямо у списку справ є зручний варіант списку покупок, яким можна поділитися з іншими.

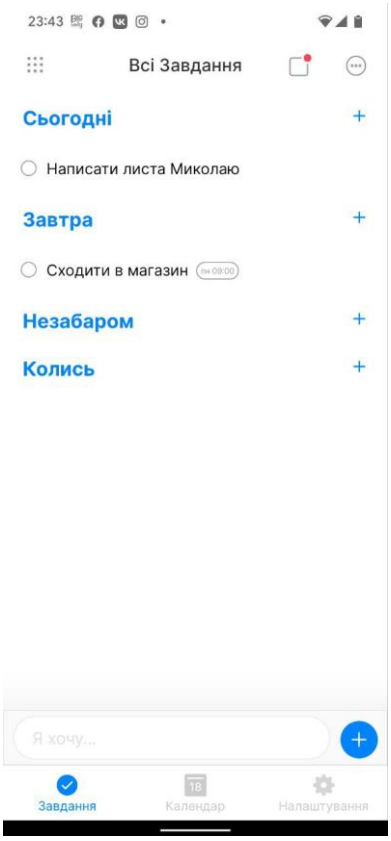


Рисунок 1.2 – Список завдань у програмі Any.do

Також є можливість редагувати подію, додавати підзадачі та писати примітки.

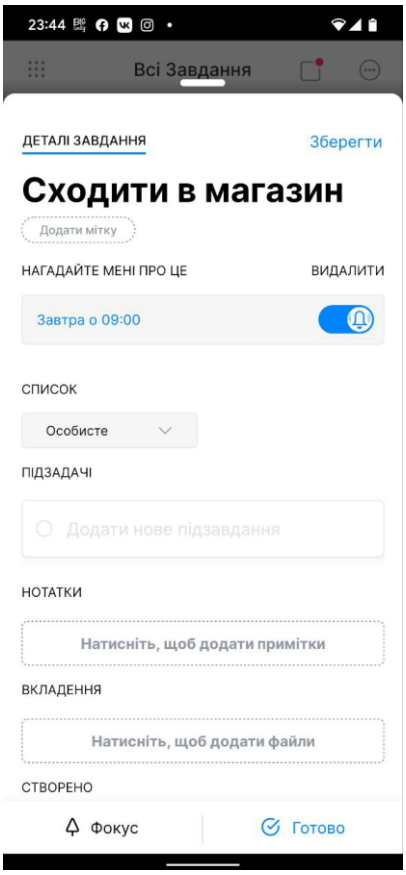


Рисунок 1.3 – Редагування завдання в програмі Any.do

У цієї програми є свої недоліки:

- Вартість преміум-версії:

Деякі користувачі можуть вважати вартість преміум-функцій дещо високою, особливо порівняно з конкурентами, які можуть пропонувати подібні можливості безкоштовно або за меншу плату.

- Обмежені можливості безкоштовної версії:

Безкоштовна версія Any.do може мати обмежені можливості порівняно з преміум-версією. Це може обмежити користувачів у доступі до певних функцій.

- Синхронізація обмежена в безкоштовній версії:

У безкоштовній версії можуть бути обмеження щодо синхронізації між пристроями або іншими обмеженнями щодо хмарового зберігання даних.

- Можливі труднощі з інтерфейсом:

Деякі користувачі можуть зазнавати труднощів з інтерфейсом чи UX, який може не відповідати їхнім особистим вподобанням.

- Відсутність деяких продвинутих функцій:

Деякі користувачі можуть виявити, що деякі продвинуті функції або інтеграції, які доступні в інших програмах, відсутні у Any.do.

Sectograph

Sectograph – це планувальник часу, який візуально відображає список завдань і подій на день у вигляді 12-годинної кругової діаграми - циферблат годинника.[39]

Список подій календаря проєктується у вигляді кругової діаграми в програмі та на віджеті головного екрана.

Події — це сектори, початок і тривалість яких можна чітко відстежувати за допомогою спеціальних дуг, щоб слідувати за планом.

Поєднання календаря та аналогового годинника дає візуальне уявлення про роботу, дозволяючи ефективно планувати та розраховувати свій день.

Серед недоліків можна позначити:

- Специфічне використання:

Специфічність графічного представлення часу у вигляді секторів годинника може не всім користувачам підходити. Деякі віддають перевагу традиційному текстовому або календарному вигляду.

- Обмежені функції:

У порівнянні з деякими іншими календарними додатками Sectograph може виглядати обмеженим за функціональністю, особливо якщо користувач очікує більш розвинених можливостей.

- Не підходить для складних розкладів:

Для людей із складними графіками чи численними подіями програма може бути менш зручною, оскільки її графічне представлення може стати перенасиченим.

- Не для всіх відомостей:

Деякі користувачі можуть вважати, що відсутність певних деталей чи інтеграції з іншими додатками обмежує їхні можливості планування.

- Залежність від особистого смаку:

Як і з будь-яким іншим додатком, смак та вибір користувача грають важливу роль. Те, що комусь подобається, іншому може не сподобатися.

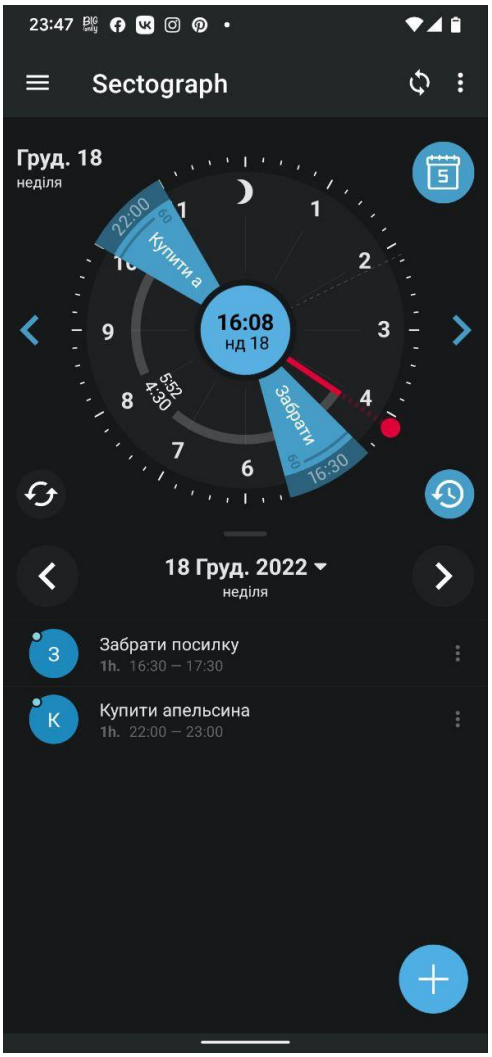


Рисунок 1.5 – Список завдань у програмі Sectograph

ToDo List & Tasks

Проста програма для найважливіших завдань. Легкий та інтуїтивно зрозумілий дизайн. Додаток має вбудовану функцію для роботи з додатком однією рукою, а його інтерфейс користувача з легкою анімацією не залишить вас байдужим.[40]

Для чого використовувати:

• За допомогою функції «швидкого додавання»: можна фіксувати та планувати завдання, коли вони спадають на думку.

• Додаючи завдання до списку справ, приходить сповіщення про майбутнє завдання, щоб не пропустити його.

• Не забувати головні задачі на день.

Серед недоліків потрібно зазначити:

- Некомфортний інтерфейс:

  Погано прорублений інтерфейс користувача, у якому деякі кнопки знаходяться не там де було б комфортно. Деякі написи погано видно через недоречний колір шрифту.

- Слабкий функціонал додатку:

  Відсутність розширених функцій, таких як пріоритизація чи категоризація задач, може стати незручною для управління складними списками справ. Додавання задач виглядає просто, але може бути обмеженим у гнучкості для деталізації задач.

На рисунках 1.7 та 1.8 можна побачити зовнішній вигляд програми та функціонал які вона надає.

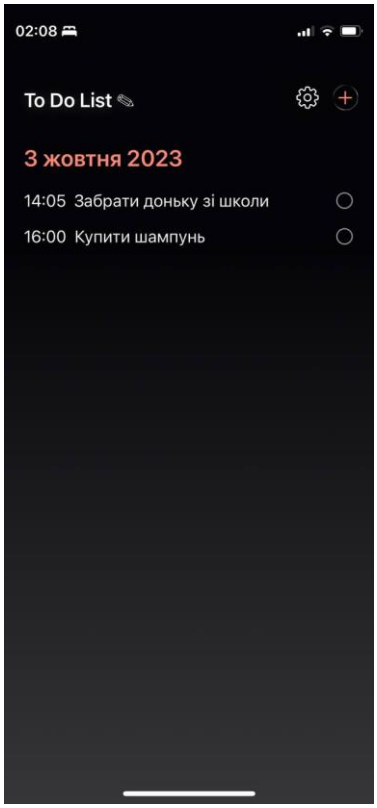


Рисунок 1.7 – Список завдань у програмі ToDo List & Tasks

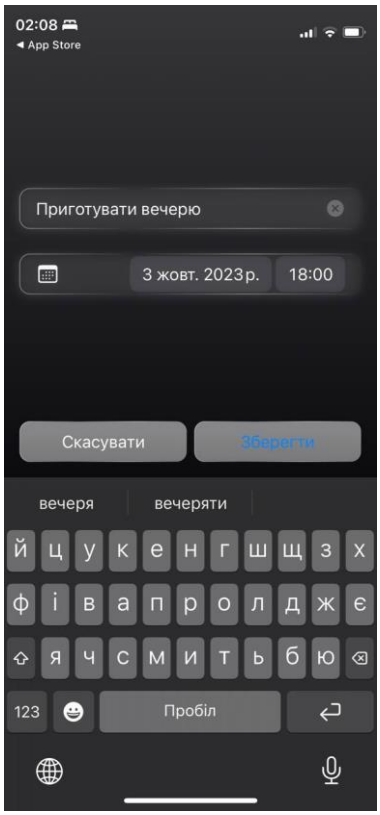


Рисунок 1.8 – Створення завдання в програмі ToDo List & Tasks

1.3.3 Перелік важливих аспектів врахування під час розробки мобільного додатку планувальника

Розробка мобільного додатка планувальника вимагає уважного підходу та врахування ряду важливих аспектів. Нижче подано ключові пункти, які важливо врахувати під час цього процесу:

Аналіз Цільової Аудиторії:

Ретельний аналіз та розуміння потреб цільової аудиторії. Це дозволяє створити функціонал, який відповідає вимогам користувачів та вирішує їхні проблеми.[41]

- Унікальність та Цінність:

Додаток повинен мати унікальні функції або пропонувати унікальний спосіб вирішення завдань планування. Важливо, щоб він надавав конкретну цінність для користувачів.

- Зручний та Інтуїтивний Інтерфейс:

Простота та зручність інтерфейсу впливають на користувальницький досвід. Потрібно переконатися, що користувачі можуть легко розуміти, як використовувати додаток без зайвих зусиль.

- Множинні Платформи:

Потрібно розглянути можливість розробки додатку для різних мобільних платформ, таких як Android[42] та iOS[43], для максимального охоплення аудиторії.

- Безпека та Конфіденційність:

Забезпечення захисту конфіденційності даних користувачів та використання найсучасніших методів безпеки.

- Синхронізація та Резервне Копіювання:

Можливість синхронізації даних між різними пристроями користувача та автоматичне резервне копіювання даних[44].

- Гнучкість та Розширюваність:

Розробка додатка таким чином, щоб його можна було легко розширювати та адаптувати до змін потреб користувачів чи ринкових умов.

- Оптимізація Для Роботи Офлайн:

Можливість користувачів використовувати додаток навіть без підключення до Інтернету.

- Аналітика та Відгуки:

Додавання аналітичних інструментів для відстеження використання додатка та отримання зворотнього зв'язку від користувачів.

- Управління Витратами та Монетизація:

Потрібно розглядати питання монетизації та витрат на розробку та підтримку додатка.

- Підтримка та Оновлення:

Забезпечення регулярних оновлень для додавання нових функцій та виправлення помилок, а також підтримка користувачів.

- Відповідність Законодавству та Політикам Магазинів Додатків:

Врахування правил та політик магазинів додатків (Google Play[45], App Store[46]) та відповідність їм.

1.4 Висновки

Було виконано аналіз проблеми автоматизованих систем планування та обліку завдань. В аналізі існуючих автоматизованих систем планування у сфері мобільних додатків були визначені переваги та недоліки існуючих рішень, що визначило простір для вдосконалення та новаторського підходу.

Дослідження історичного розвитку планувального спорядження виявило ключові пункти, які суттєво впливали на формування та еволюцію таких систем.

Аналіз аналогів підкреслив важливість інновацій та додаткових функціональних можливостей для задоволення різноманітних потреб користувачів.

Перелік важливих аспектів, які слід враховувати під час розробки мобільного додатку планувальника, був узагальнений і включав в себе різноманітні функції та характеристики для забезпечення високої ефективності та зручності користування.

2 АНАЛІЗ ТЕХНОЛОГІЙ ТА ФУНКЦІОНАЛЬНОСТІ МОБІЛЬНОГО ДОДАТКУ

2.1 Функціональність мобільного додатку

Кожен створений додаток повинен визначатися своїм призначенням та функціональністю для досягнення результату та виконання своєї основної мети. У даному випадку, розроблений мобільний додаток для планування задач є інструментом, спрямованим на оптимізацію управління часом та завданнями користувача.

Основна функція додатка полягає в передачі введених даних в базу даних та їхньому подальшому сортуванні за часом виконання. Початково користувач вводить завдання, термін виконання та примітки, а додаток візуалізує ці задачі на шкалі дня для кращого розуміння. При необхідності редагування, користувач може вибрати конкретний день та редагувати завдання зі списку.

Додаток представляє собою комплексну систему з можливістю додавання, організації та відслідковування завдань. Кожна задача може бути детально описана, включаючи додаткові дані для більшої контекстної інформації. Функції сортування, фільтрації, маркування та категоризації полегшують організацію та доступ до задач.

Система використовує базу даних, основним елементом якої є бібліотека Room для зберігання даних про справи та задачі. Також реалізовано функціонал для аналізу продуктивності користувача на основі виконаних завдань.

Додаток надає інтерактивність через інтуїтивний і “userfriendly”[46] інтерфейс. Постійне оновлення та додавання нових функцій розглядається як необхідний етап для підтримки високого рівня зручності користування та розвитку функціоналу.

У ході програмного проектування ураховано можливі виняткові ситуації та проблеми, а система розроблена з урахуванням можливості ефективного реагування на непередбачені обставини. Планується постійне розширення

функціоналу для покращення зручності користування та попередження непередбачуваних аномалій.

Загальною метою є створення ефективного інструменту для управління завданнями та часом, здатного динамічно адаптуватися до змінних потреб користувачів та уникнути негативних наслідків від несподіваних сценаріїв використання.

Для того щоб планувати розробку потрібно змоделювати яка має бути взаємодія між користувачами та компонентами програми. Для таких цілей існують *діаграми прецедентів*. Таблиця прецедентів або діаграма прецедентів (Use Case Diagram) є одним із видів діаграм в мові моделювання UML (Unified Modeling Language), яка використовується для візуалізації функціональності системи та взаємодій між системою та зовнішніми сутностями (акторами). Основною метою цієї діаграми є зазначення того, як система буде взаємодіяти з різними сутностями (користувачами чи іншими системами).

Основні компоненти таблиці прецедентів:

- Актори (Actors):

Актори представляють сутності, які взаємодіють з системою. Це можуть бути користувачі, інші системи або зовнішні сервіси.

- Прецеденти (Use Cases):

Прецеденти визначають функціональні можливості системи або операції, які вона може виконувати.

- Зв'язки між акторами і прецедентами:

Зв'язки вказують на те, як актори взаємодіють з прецедентами. Наприклад, лінія між актором і прецедентом може вказувати на участь актора в конкретному функціоналі чи операції.

- Система:

Вона представлена прямокутником, який містить всі прецеденти системи та всі актори.

Приклад звичайних прецедентів може включати такі операції, як "Створити обліковий запис", "Здійснити покупку", "Видалити елемент" тощо.

Таблиці прецедентів є корисним інструментом для розуміння функціональних вимог системи та її взаємодій з акторами.

Блок-схема порядку дій (Flowchart) – це графічне представлення послідовності операцій або дій в системі чи програмі. Вона використовує стандартні графічні символи для представлення кожної дії та логічних зв'язків між ними. Блок-схеми використовуються для візуалізації та розуміння алгоритмів, процесів чи програмних логік.

Основні елементи блок-схеми порядку дій:

- Процес (Process):

Представляє конкретну дію чи операцію, яка виконується.

- Рішення (Decision):

Вказує на точку вибору між двома чи більше альтернативними шляхами. Зазвичай має дві гілки: "так" та "ні".

- Термінальний блок (Terminal):

Позначає початок або завершення алгоритму чи процесу. Зазвичай містить фрази типу "Початок" або "Кінець".

- Введення/Виведення (Input/Output):

Вказує на введення або виведення даних від або до користувача чи інших систем.

- Сполучний лінійний блок (Connector):

Використовується для з'єднання різних частин блок-схеми, які розташовані на різних сторінках.

- З'єднання (Off-page Connector):

Показує з'єднання з іншими частинами блок-схеми на інших сторінках.

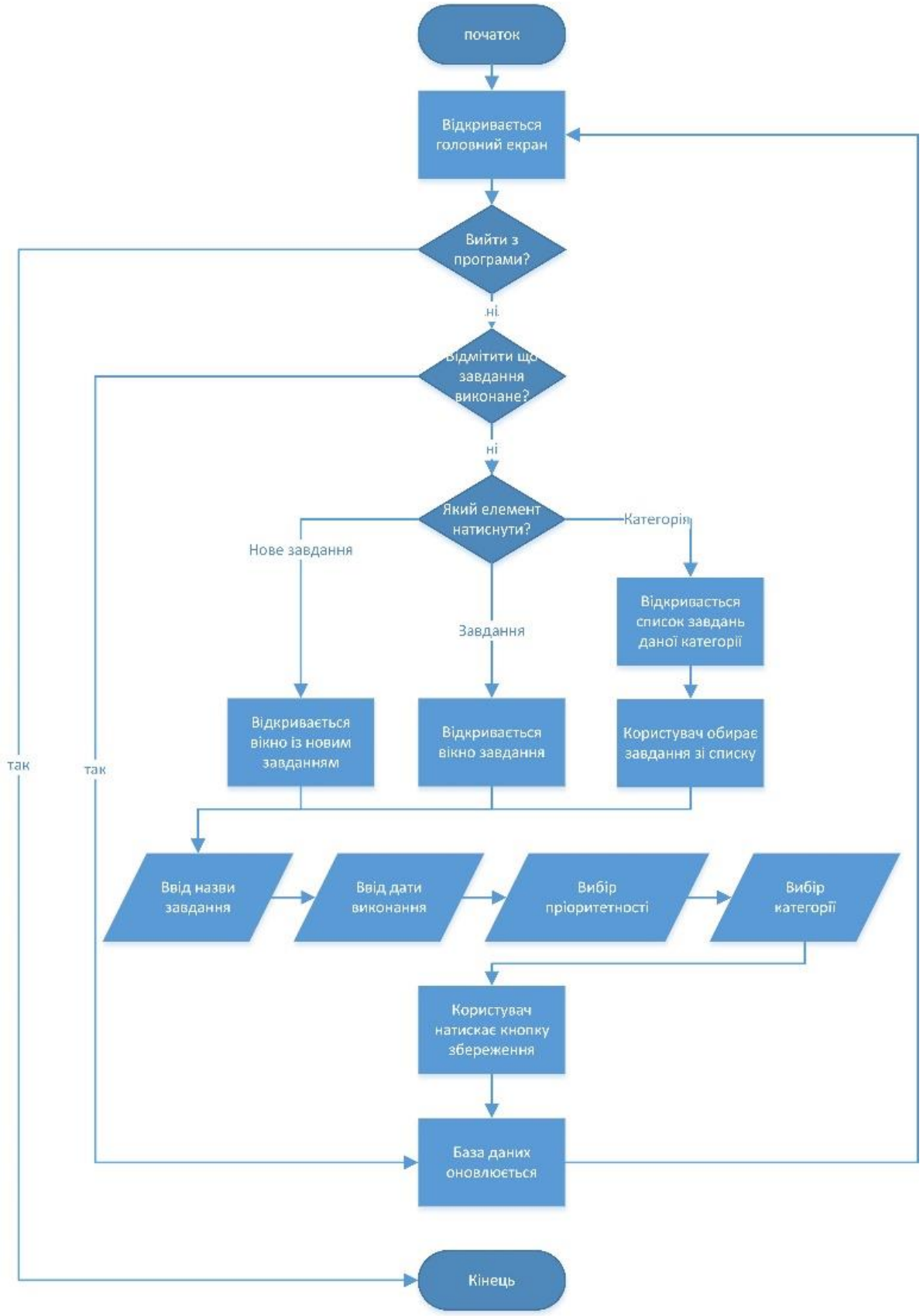


Рисунок 2.1 - Блок схема порядку дій

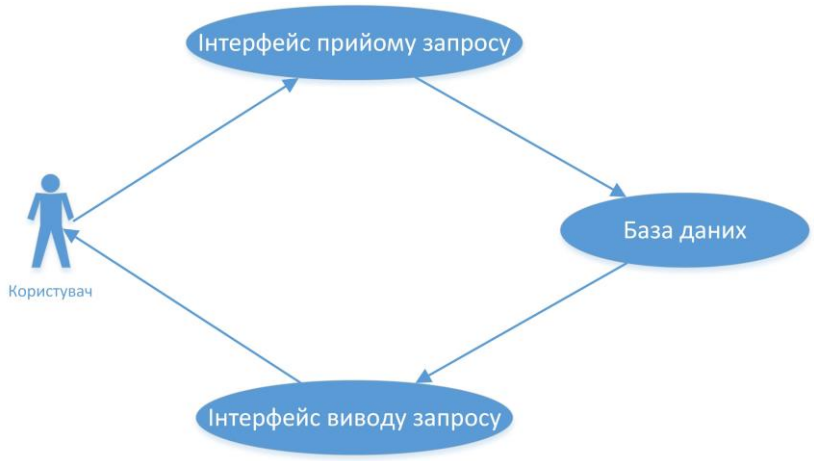


Рисунок 2.2 – Модель взаємодії між користувачем та базою даних

Рисунок 2.3 – Модель взаємодії між користувачем та базою даних

На рисунку 2.1 зображена блок схема порядку дій програмою, при різних умовах.

На рисунку 2.2 зображена спрощена діаграма прецедентів предметної галузі.

На рисунку 2.3 зображена діаграма прецедентів, у якій зображено модель взаємодії між користувачем та базою даних.

2.2 Обгрунтування вибору операційної системи

Сучасний світ мобільних додатків пропонує розробникам можливість обрати операційну систему для свого продукту серед двох основних платформ: Android та iOS. Обираючи оптимальну платформу, розробники стикаються з питанням, для кого саме вони створюють додаток та які особливості повинен мати цей продукт.

Android і iOS мають значну частку ринку, і кожна з них має свої унікальні риси. Android відзначається великою кількістю різних пристроїв, що працюють під цією ОС, що може становити виклик для уніфікації дизайну та функціональності. З іншого боку, розробка для Android є більш гнучкою, оскільки вона базується на відкритих стандартах та відзначається меншою кількістю обмежень від виробників пристроїв.

iOS, навпаки, володіє обмеженим рядом пристроїв, що полегшує уніфікацію та стандартизацію розробки. Ця платформа також славиться вищим рівнем безпеки та спрощеною процедурою розгортання додатків.

Однак важливо враховувати, що база користувачів кожної платформи має свої унікальні вимоги та очікування. Android має більше користувачів у світі, особливо в країнах з економікою в розвитку, тоді як iOS-користувачі відомі своєю вищою платоспроможністю. Вибір платформи може визначити цільову аудиторію та рівень доходу від додатку.

Переваги та недоліки розробки для Android та iOS визначаються конкретними завданнями та потребами проекту. Вибір між ними може залежати від багатьох факторів, таких як бюджет, терміни, географічні переваги користувачів та технічні особливості. Важливо враховувати ці фактори для успішної розробки та впровадження мобільного додатку.

Вибір розпочати розробку для операційної системи Android та згодом розширити на iOS – це стратегічний хід, обумовлений кількома важливими факторами. Зокрема, Android є великим ринком, де присутній значний обсяг користувачів. Розпочати з цієї платформи дозволяє швидше набрати аудиторію та зробити продукт доступним для широкого кола людей.

Оскільки Android має свої особливості, такі як різноманіття пристроїв та відкритий код, розробка для цієї ОС дозволяє набути досвіду вирішення різних технічних викликів та адаптувати додаток до різних пристроїв. Це також може допомогти збирати відгуки від користувачів та поліпшувати продукт на ранніх етапах.

Поступовий перехід на iOS, коли база користувачів виросте, має сенс з погляду розширення аудиторії та залучення більш платоспроможної цільової групи. Користуючись досвідом та знаннями, набутими під час розробки для Android, можна більш ефективно впроваджувати додаток на iOS, звертаючи увагу на особливості цієї платформи та споживацьких звичаїв користувачів.

Такий подвійний підхід дозволяє оптимально використовувати ресурси та стратегічно будувати ріст продукту відповідно до характеристик кожної з платформ.

2.3 Обгрунтування засобів розробки дизайну програми

Розробка дизайну мобільного додатку є важливим етапом у творенні високопродуктивних та зацікавлюючих продуктів для сучасних користувачів. У

цьому процесі використовуються різноманітні інструменти, що дозволяють створювати естетичні та функціональні інтерфейси, а також забезпечують зручність взаємодії з додатком.

Вибір правильного інструменту для розробки дизайну визначається рядом чинників, таких як зручність в роботі, можливості спільної роботи команди, ефективність створення прототипів та взаємодії з розробниками. У цьому контексті інструменти, такі як Figma, Adobe XD, Sketch та інші, стають невід'ємною частиною творчого процесу.

Розробка дизайну не лише забезпечує естетичний вигляд додатку, але й спрямована на створення інтуїтивних та зручних інтерфейсів, що підвищують задоволення користувачів від використання мобільних додатків. У подальшому, такий підхід до дизайну сприяє популярності та конкурентоспроможності продукту на ринку мобільних додатків.

Adobe XD (Experience Design) - це інструмент для розробки дизайну та створення прототипів мобільних та веб-додатків. Він розроблений компанією Adobe і забезпечує дизайнерам та розробникам зручний інтерфейс для взаємодії з проектами та реалізації їх ідей.

Основні можливості Adobe XD для розробки дизайну мобільних додатків включають:

- Інтерфейс та Організація:

Adobe XD пропонує простий та інтуїтивний інтерфейс, що сприяє зручному розміщенню та організації об'єктів, елементів та шарів у проекті.

- Дизайн та Прототипування:

Користувачі можуть створювати макети та додавати елементи дизайну, такі як текст, форми, зображення та іконки. Adobe XD підтримує швидке та легке створення прототипів для тестування взаємодії та навігації.

- Анімації:

Інструмент дозволяє додавати анімації до прототипів, щоб краще відобразити вигляд та поведінку додатка.

- Спільна робота:

Adobe XD підтримує спільну роботу в команді. Користувачі можуть додавати коментарі, взаємодіяти та обмінюватися проектами, спрощуючи процес спільної розробки.

- Відстеження відгуків:

Завдяки можливості відстеження відгуків, дизайнери можуть отримати конструктивну критику від команди чи клієнтів прямо в інтерфейсі програми.

Хоча Adobe XD є потужним інструментом для розробки дизайну мобільних додатків, він має деякі недоліки:

- Обмежена анімація:

В порівнянні з іншими інструментами для дизайну мобільних додатків, Adobe XD може мати обмежені можливості створення складних анімацій та переходів.

- Невеликий вибір шрифтів та інструментів малювання:

У порівнянні з іншими графічними редакторами, Adobe XD має менший вибір шрифтів та інструментів для малювання.

- Обмежений функціонал для дизайну веб-сайтів:

Хоча Adobe XD призначений для дизайну мобільних додатків та веб-сайтів, його функціонал для веб-дизайну може бути менш розгорнутим у порівнянні з іншими інструментами.

- Неінтегрована робота з джерелами даних:

Adobe XD не надає інтегрованих інструментів для роботи з реальними даними чи базами даних, що може бути недоліком при створенні реалістичних прототипів.

- Відсутність можливості експорту коду:

У порівнянні з іншими інструментами, Adobe XD не надає вбудованих можливостей експорту коду, що може ускладнити взаємодію між дизайнерами та розробниками.

- Вартість:

Adobe XD може вимагати підписки на платформу Creative Cloud для повного доступу до всіх функцій та можливостей. Для деяких користувачів це може стати фінансовим обмеженням, особливо для невеликих команд чи самостійних дизайнерів.

Також для розробки дизайну додатків часто використовують інструмент *Figma*. Перша і основна особливість даного інструменту це його вартість. Базова версія є безкоштовною і її може бути достатньо для більшості розробників. Але крім того для цього інструменту розробляють та продають спецільні адони які розширюють функціональність інструменту. Але навіть враховучи ціну цих інструментів, розробка дизайну на основі середовища Figma є дешевшою за розробку на основі її аналогів.

Нижче перераховано лише частина з корисних адонів, які можна використати при розробці дизайну:

- Figmify:

Додає можливість швидко вставляти зображення та створювати анімації безпосередньо в Figma.

- Zeplin:

Дозволяє ефективно обмінюватися дизайном між дизайнерами та розробниками, додаючи можливість експортувати стилі та ресурси в Zeplin.

- Overflow Figma Plugin:

Допомагає створювати прототипи з Figma та експортувати їх в Overflow для подальшого створення інтерактивних прототипів.

- Auto Layout:

Вбудована функція, але вона дозволяє легко створювати автоматичні макети для адаптивного дизайну.

- Figmify Design Lint:

Аналізує дизайн та надає рекомендації щодо покращень, таких як оптимізація розміщення елементів чи стилізація.

- Stark:

Забезпечує інструменти для перевірки доступності та контрастності кольорів у дизайні.

- Content Buddy:

Дозволяє легко генерувати фіктивний контент для дизайну, включаючи текст, зображення та дані.

- Unsplash:

Дозволяє швидко вставляти зображення з Unsplash безпосередньо в Figma.

- Google Sheets Sync:

Синхронізує дані між Google Sheets та Figma, корисно для роботи з текстовим контентом.

- Map Maker:

Дозволяє вбудовувати карти у макети для візуалізації географічної інформації.

Серед недоліків можна відмітити такі моменти:

- Залежність від Інтернету:

Figma використовує хмарну інфраструктуру, і його повна функціональність вимагає підключення до Інтернету. Без зв'язку з мережею можуть виникнути обмеження у роботі.

- Проблеми із швидкістю:

Залежно від швидкості Інтернет-з'єднання та потужності обладнання, працювати у Figma може бути повільним, особливо при великих проектах з великою кількістю об'єктів.

- Приватність даних:

Оскільки Figma базується на хмарних технологіях, користувачі можуть мати питання щодо приватності даних, зокрема, як проекти зберігаються та обробляються.

- Обмеження у безкоштовному варіанті:

Безкоштовний план Figma має свої обмеження, такі як обмеження на кількість проектів та команд. Деякі продвинуті функції можуть бути доступні лише за додаткову плату.

- Вивчення інтерфейсу:

Для новачків інтерфейс Figma може здаватися трошки складним, і для повного використання всіх можливостей може знадобитися трошки часу та навчання.

- Вартість для команд:

Якщо працюють в команді, вартість підписок для кожного користувача може зрости, що робить його менш доступним для деяких бюджетів.

Sketch є популярним інструментом для дизайну серед користувачів macOS. Ось кілька ключових аспектів та характеристик інструменту Sketch:

- Векторний дизайн:

Інструмент спеціалізується на векторному дизайні, що дозволяє створювати масштабовані та високоякісні зображення для різних пристроїв.

- Спрощений інтерфейс:

Sketch володіє інтуїтивно зрозумілим інтерфейсом, що полегшує вивчення та використання новими користувачам.

- Бібліотеки та символи:

Sketch підтримує використання бібліотек та символів, що дозволяє створювати уніфіковані та легко оновлювані елементи дизайну.

- Зручний експорт:

Sketch дозволяє зручно експортувати графіку для подальшої реалізації в розробці мобільних додатків або веб-сайтів.

- Розширюваність:

Sketch підтримує розширення за допомогою плагінів, що дозволяє розширити його функціональність та пристосувати до власних потреб.

- Колаборація та обмін файлами:

Можливості співпраці: Інструмент надає можливості для спільної роботи та обміну дизайн-файлами, що дозволяє командам працювати над проектом.

- Вартість:

Одноразова оплата: Для користувачів, які використовують macOS, Sketch пропонує одноразовий платіж за ліцензію, що може бути вигідним порівняно з підпискою на інші інструменти.

Хоча Sketch має багато переваг, слід враховувати, що він ексклюзивний для macOS, і користувачі, які використовують інші операційні системи, можуть вибрати інші альтернативи.

Серед перерахованих вище інструментів було обрано інструмент Figma через його функціональність та хорошу оптимізацію роботи під розробку дизайну та анімацій для мобільних додатків. Також серед переваг є ціна, так як базова версія є безкоштовною, а додаткові інструменти є дешевшими за аналогів.

2.4 Обгрунтування використання бази даних та її надбудови.

При розробці програм для смартфонів часто використовують бази даних для більш структурованого вигляду даних. Особливо якщо це стосується програми в якій мають зберігатися тижневі плани та їх дані.

При розробці бази даних варто звернути увагу на різні структури баз даних.

Ієрархічна база даних — це структура даних, організована у вигляді ієрархії чи дерева, де дані представлені у вигляді вузлів та зв'язків між ними. Основна ідея полягає у впорядкуванні даних у вигляді батьківських та дочірніх вузлів, що утворює ієрархічну структуру.

Основні поняття і їх характеристики ієрархічних баз даних:

- Сегмент (Segment):

Один запис у базі даних, який може включати в себе дані або вказівки на інші сегменти.

- Запис (Record):

Група пов'язаних сегментів, яка представляє собою одну одиницю інформації.

- Сет (Set):

Група записів, що мають спільний батьківський сегмент. Сет може розглядатися як піддерево ієрархії.

- Реляція батько-дитина:

Вказує на те, що один сегмент (батьківський) має відношення з іншим сегментом (дитячим) на рівні нижче.

- Корінь ієрархії:

Вершина дерева, яка не має батьківського сегмента.

- Лист (Leaf):

Сегмент, який не має дочірніх сегментів.

- Шлях (Path):

Спосіб знаходження конкретного сегмента в ієрархії.

Переваги ієрархічних баз даних включають швидкий доступ до даних за допомогою ієрархічної структури, простоту та ефективність в роботі з великими обсягами даних. Однак вони також мають обмеження, такі як складність у зміні структури та неефективність для деяких видів запитів, що може бути важливим у сучасних інформаційних системах.

Недоліком можна назвати надлишковість даних.

Мережева структура бази даних є однією з концепційних моделей організації даних, в якій взаємозв'язки між записами формують графову структуру. Ця модель була популярною в 1960-1970-х роках та представлена як альтернатива ієрархічній базі даних. Основні компоненти мережевої бази даних включають записи, вузли та відносини.

Основні характеристики мережевої бази даних:

- Записи (Records):

Представляють собою набори даних, які можуть містити різноманітні типи інформації.

- Вузли (Nodes):

Це елементи даних, які можуть бути пов'язані між собою.

- Відносини (Relationships):

Визначають зв'язки між вузлами, показуючи, як один вузол пов'язаний з іншим.

- Множества (Sets):

Групи вузлів, які можуть мати спільний батьківський вузол.

- Вузли-посилання (Link Nodes):

Вузли, що вказують на інші вузли, визначаючи зв'язки.

- Шлях (Path):

Спосіб знаходження конкретного вузла в графі.

- Множини вузлів (Node Sets):

Групи вузлів, які мають спільний батьківський вузол.

Мережева модель дозволяє ефективно представляти багатовідносинні дані та допускає більш гнучкий доступ до інформації, ніж ієрархічна модель. Однак вона може бути складною у використанні та управлінні через складність зв'язків та ризик дублювання даних. Мережева структура бази даних досі може застосовуватись в окремих випадках, але загалом її популярність втратила з часом на користь більш сучасних моделей даних, таких як реляційна база даних.

Об'єктно-орієнтована база даних (ООБД) – це система управління базами даних, яка використовує об'єктно-орієнтований підхід до моделювання та організації даних. Вона забезпечує зберігання та обробку об'єктів, які представляють реальні або абстрактні об'єкти, і використовує об'єктно-орієнтовану модель даних для опису взаємозв'язків і мову запитів.

Основні характеристики об'єктно-орієнтованих баз даних:

- Класи та об'єкти:

Дані організовані в класи (типи) та об'єкти, що є конкретними представниками цих класів. Кожен об'єкт може мати свої атрибути та методи.

- Успадкування:

Включає можливість успадкування атрибутів та методів від батьківського класу до підкласів.

- Поліморфізм:

Дозволяє використовувати об'єкти підкласів так, якщо вони є об'єктами їхнього батьківського класу.

- Інкапсуляція:

Забезпечує обмежений доступ до деяких атрибутів та методів об'єкта, забезпечуючи концепцію "прихованості деталей реалізації".

- Асоціації та агрегації:

Дозволяє визначати взаємозв'язки між об'єктами, такі як асоціації (зв'язки) та агрегації (включення об'єктів в інші об'єкти).

- Мови запитів:

Використовує спеціалізовані мови запитів, такі як OQL (Object Query Language), для взаємодії та отримання даних.

- Довговічність:

Здатна зберігати об'єкти та їхні взаємозв'язки у базі даних навіть після припинення роботи програми.

Об'єктно-орієнтовані бази даних застосовуються там, де важлива гнучкість моделювання та потрібна відмінна від реляційних баз даних концепція роботи з об'єктами. Вони часто використовуються в розробці програмного забезпечення, де важлива об'єктно-орієнтована парадигма програмування.

Реляційна база даних (РБД) - це структурована колекція даних, яка організована за допомогою реляційної моделі даних. Вона використовує таблиці для представлення і збереження даних та взаємозв'язків між ними. Реляційні бази даних є широко поширеними і використовуються в різних сферах, включаючи бізнес, науку, технології та інші галузі.

Основні характеристики реляційних баз даних:

- Таблиці:

Дані організовані у вигляді таблиць, де кожна таблиця відображає конкретний тип інформації.

- Рядки та колонки:

Таблиці складаються з рядків (кортежів) та колонок (атрибутів). Кожен рядок представляє запис даних, а кожна колонка - конкретний атрибут.

- Ключі:

Визначаються первинні та зовнішні ключі для унікальності записів та забезпечення взаємозв'язків між таблицями.

- Нормалізація:

Процес розбиття таблиць на менші, пов'язані за допомогою ключів, для уникнення аномалій та забезпечення ефективності.

- Мова запитів SQL:

Використовує мову SQL (Structured Query Language) для взаємодії з базою даних, виконання запитів та здійснення змін.

- Транзакції:

Підтримка транзакцій для забезпечення атомарності, цілісності та відновлення бази даних після помилок.

- Відношення:

Таблиці можуть мати взаємозв'язки між собою, що дозволяє ефективно організовувати та аналізувати дані.

Переваги реляційних баз даних включають структурованість, простоту використання, високий рівень нормалізації та відмінну підтримку транзакцій. Однак вони можуть бути менш ефективними для деяких завдань, де даними управляють більш складні взаємозв'язки, або в разі великого обсягу даних.

Реляційна база даних добре підійде під задачі які стоять перед розробкою даного додатку, так як всі дані про задачі та їхні дані добре будуть структуруватись в табличних формах.

На даному етапі розробки планується розробляти локальну базу даних із можливим розширенням її до хмарної. Серед локальних баз даних які можна використати при розробці мобільного додатку це SQLite, Room Database, Couchbase Lite, Core Data, FMDB та інші.

SQLite - це легка та вбудована система управління базами даних (СУБД)[48], яка широко використовується в різних мобільних, веб-та вбудованих додатках. Основні особливості та характеристики SQLite включають:

- Вбудована БД:

SQLite є вбудованою СУБД, що означає, що вона не вимагає окремого сервера та використовується безпосередньо в межах програми, яка звертається до бази даних.

- Легкість та Простота:

SQLite відома своєю легкістю та простотою використання. База даних універсальна, може бути використана в різних типах додатків.

- Безсерверність:

Оскільки SQLite є безсерверною базою даних, можна працювати з нею, не потрібно запускати окремий процес сервера.

- Невеликий Розмір:

Бібліотека SQLite має невеликий розмір, що робить її ідеальним вибором для обмежених ресурсів пристроїв, таких як мобільні телефони та вбудовані системи.

- Транзакції:

SQLite підтримує транзакції, що дозволяє атомарно виконувати групу SQL-операцій.

Крос-платформеність:

SQLite підтримується на різних платформах, включаючи Android, iOS, Windows[49], Linux[50], macOS[51] і багато інших.

Динамічні Типи Даних:

В SQLite використовуються динамічні типи даних, що означає, що тип даних колонки може змінюватися під час життєвого циклу бази даних.

Підтримка SQL:

SQLite повністю сумісна з мовою SQL, що робить її зручною для тих, хто знайомий з SQL-запитами.

Як вказано вище, SQLite використовується в різних системах, але якщо говорити про розробку мобільного додатку на OS Android, варто звернути увагу за надбудову для SQLite під назвою Room Database.

Room - це частина Android Architecture Components[52], яка надає абстракцію бази даних та використовується для спрощення роботи з SQLite в Android-додатках. Вона пропонує високорівневий інтерфейс для роботи з базою даних, забезпечуючи при цьому ефективну інтеграцію з іншими компонентами архітектури.

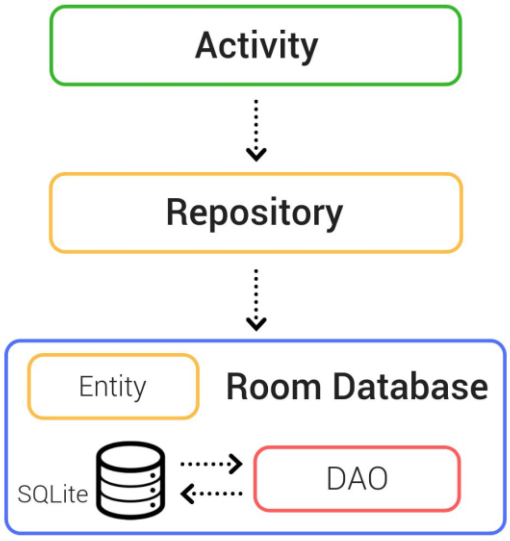


Рисунок 2.2 – Діаграма архітектури Room

Якщо говорити про принцип роботи цієї надбудови, то: клас бази даних надає програмі екземпляри DAO(Data access objects)[53], пов’язані з цією базою даних. У свою чергу, програма може використовувати DAO для отримання даних із бази даних як екземплярів пов’язаних об’єктів сутності даних. Програма також може використовувати визначені сутності даних для оновлення рядків із відповідних таблиць або для створення нових рядків для вставки. Рисунок 2.2 ілюструє взаємозв’язок між різними компонентами Room.[54]

Couchbase Lite - це легковагова база даних для мобільних та вбудованих додатків. Основна функціональність Couchbase Lite включає в себе роботу в офлайн-режимі, реплікацію даних та можливість локального збереження та

опрацювання даних без постійного з'єднання з сервером бази даних. Ось деякі ключові аспекти Couchbase Lite:

- Легковаговість та мобільність:

Couchbase Lite оптимізований для використання на мобільних пристроях, має низький вплив на ресурси та може ефективно працювати в умовах обмеженої мережевої доступності.

- Офлайн-режим:

Однією з ключових особливостей Couchbase Lite є підтримка роботи в офлайн-режимі. Додатки можуть локально зберігати дані та працювати з ними, навіть якщо немає з'єднання з мережею.

- Реплікація даних:

Couchbase Lite надає механізм реплікації, який дозволяє синхронізувати дані між мобільними пристроями та сервером бази даних. Це робить можливим обмін та збереження актуальних даних на різних пристроях.

- Документ-орієнтована модель даних:

Couchbase Lite працює з даними в форматі документів, що дозволяє легко зберігати та взаємодіяти з ними, використовуючи JSON-структури.

- Можливості пошуку:

Couchbase Lite підтримує можливості пошуку та фільтрації даних, що спрощує роботу з великими обсягами інформації.

- Широкі можливості інтеграції:

Couchbase Lite підтримує кілька мов програмування (Java, C#, Swift, інші) та може використовуватись в різних платформах, включаючи Android та iOS.

- Висока продуктивність та масштабованість:

Couchbase Lite розроблено для забезпечення високої продуктивності та масштабованості в умовах роботи з мобільними додатками та вбудованими системами.

Core Data - це фреймворк для управління об'єктами та даними в додатках, розроблених для платформ iOS та macOS. Основною його метою є забезпечення швидкого та ефективного доступу до даних, а також спрощення завдань роботи з базою даних.

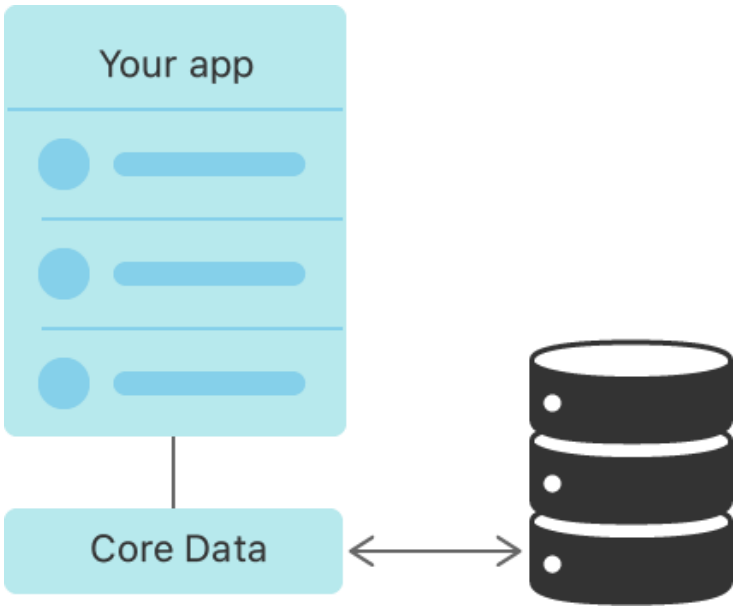


Рисунок 2.3 – Спрощена схема роботи Core Data

Основні аспекти Core Data:

- Об'єктно-орієнтований підхід:

Core Data працює на рівні об'єктів, що дозволяє розробникам працювати з даними, використовуючи звичайні класи об'єктів мови програмування Swift або Objective-C. Це робить роботу з даними більш зрозумілою та зручною.

- Можливості зберігання:

Core Data дозволяє зберігати дані в SQLite базі даних, бінарних файлах або навіть в оперативній пам'яті, забезпечуючи гнучкість вибору типу сховища.

- Керування версіями:

Механізм версіонування в Core Data дозволяє легко впроваджувати зміни в схему бази даних, забезпечуючи зручність у підтримці та розвитку додатків з плинним ростом.

- Керування зв'язками:

Core Data автоматично вирішує проблеми зв'язків між об'єктами, дозволяючи легко визначати та використовувати взаємозв'язки між об'єктами.

- Кешування та продуктивність:

Core Data використовує ефективні механізми кешування та відстроченого зберігання для оптимізації продуктивності та швидкості доступу до даних.

FMDB - це обгортка (wrapper) навколо бібліотеки SQLite для мов програмування Objective-C та Swift. Вона дозволяє розробникам легко використовувати SQLite для роботи з базою даних в додатках для платформ iOS та macOS.

Основні характеристики та особливості FMDB:

- Легкість використання:

FMDB надає простий та зрозумілий інтерфейс для виконання операцій з базою даних SQLite. Це робить його ідеальним вибором для розробників, які шукають простий спосіб роботи з SQLite.

- Об'єктно-орієнтований підхід:

Інтерфейс FMDB гарно вписується в об'єктно-орієнтований стиль програмування мов Objective-C та Swift, що спрощує взаємодію з бібліотекою.

- Підтримка запитів SQL:

Можна виконувати будь-які SQL-запити через FMDB, що надає більшу гнучкість при роботі з базою даних.

- Безпека:

FMDB допомагає уникнути SQL-ін'єкцій та інших загроз безпеки, надаючи безпечний шлях для виконання SQL-запитів.

- Підтримка черги завдань та багатозадачності:

Використання FMDB у додатку дозволяє безпечно працювати з базою даних навіть в умовах багатозадачності та паралельного виконання операцій.

- Автоматичне виправлення помилок:

FMDB надає механізми для обробки помилок та виправлення їх, що робить його стійким до невірно виконаних операцій з базою даних.

FMDB часто вибирається розробниками, які шукають легкий інструмент для роботи з SQLite у своїх додатках для iOS та macOS, і відмінно підходить для проектів різного розміру та складності.

Так як операційна система для розробки додатку було обрано OS Android, потрібно обирати серед відповідних СУБД. Вибір зупинився на Room Database. Її краще використовувати ніж SQLite, так як це обгортка, яка має ті самі можливсті, але сильно спрощує роботу при розробці баз даних. Якщо порівнювати його із Couchbase Lite, то Room Database має кращу підтримку для розробки СУБД для OS Android. Крім того Couchbase Lite використовує JSON-структури, що може працювати дещо повільніше за схему SQLite.

2.5 Висновки

В результаті обгрунтування основних методів розробки було вирішено використати такі методи: для розробки дизайну мобільного додатку має використовуватись інструмент Figma через його ціну та функціональність. Для розробки бази даних має використовуватись система управління базами даних SQLite, так як це єдина офіційно прийнята система для Android. Але для спрощення розробки має використовуватись надбудова Room Database.

3 ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ ТА БАЗИ ДАНИХ

3.1 Вибір та аналіз інструментів розробки

Вибір інструментів для розробки програмного забезпечення є ключовим етапом у процесі створення ефективної та функціональної системи. Сучасна індустрія розробки пропонує різноманіття інструментів, які охоплюють всі аспекти та етапи проекту - від розробки дизайну та програмного коду до вивчення та оптимізації баз даних.

Важливість правильного вибору інструментів важко переоцінити. Кожен етап розробки вимагає специфічного підходу, та ефективність всього процесу значно залежить від відповідних інструментів. Обираючи інструментарій для розробки дизайну, програмного забезпечення та бази даних, необхідно враховувати велику кількість факторів, таких як потреби проекту, технічні вимоги, масштабність, зручність використання, а також можливості взаємодії інструментів між собою.

Ретельний аналіз та вибір інструментів відіграють критичну роль у забезпеченні успішного виконання завдань та досягненні поставлених цілей проекту. Врахування всіх аспектів інструментів для дизайну, програмування та баз даних є необхідним етапом у визначенні оптимального та працездатного стеку технологій для подальшої розробки програмного продукту.

3.1.1 Вибір інструменту для розробки дизайну мобільного додатку

В якості інструменту для розробки дизайну мобільного додатку було обрано Figma. Figma є потужним та універсальним інструментом, який відзначається кількома ключовими перевагами.

По-перше, Figma є онлайн-платформою, що дозволяє легко співпрацювати з членами команди та здійснювати редагування в реальному часі. Це особливо важливо при роботі в розподілених командах або в умовах, коли важлива швидкість реагування на зміни.

По-друге, Figma дозволяє створювати дизайн для різних платформ, включаючи мобільні додатки, забезпечуючи при цьому адаптивність та гнучкість. Його інтерфейс дружелюбний і доступний, що робить його привабливим для дизайнерів різного рівня досвіду.

По-третє, Figma відрізняється можливістю прототипування, що сприяє візуалізації функціональності та взаємодії в мобільному додатку перед його розробкою. Це робить процес розробки більш ітеративним та сприяє виправленню можливих недоліків на ранніх етапах.

Нарешті, Figma відзначається можливістю зберігати проекти в хмарі, що дозволяє легко здійснювати доступ до даних та працювати з ними з різних пристроїв.

Узагальнюючи, Figma обрано як інструмент для розробки дизайну мобільного додатку через його зручність у співпраці, можливості адаптивного та прототипного дизайну, а також легкий доступ до проектів через хмарну інфраструктуру.

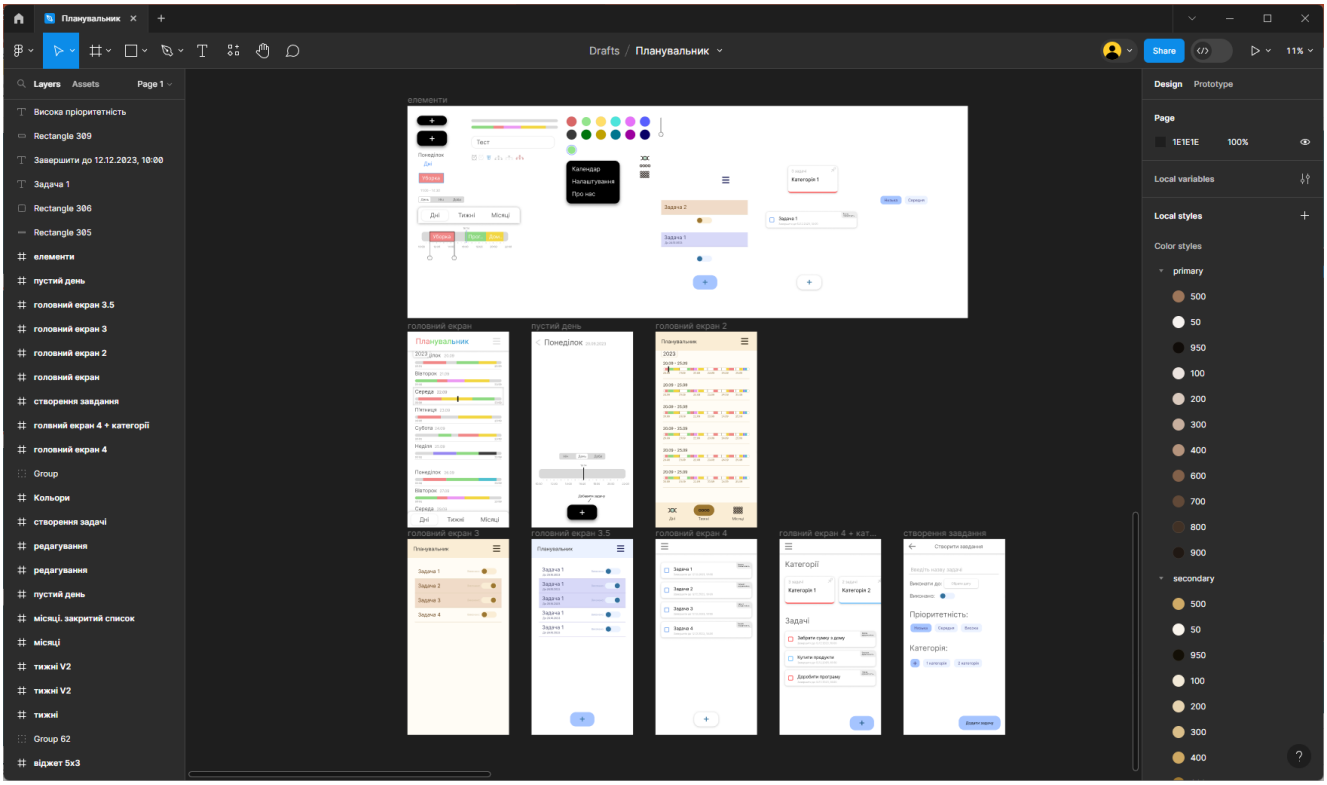


Рисунок 3.1 – Середовище розробки дизайну Figma

3.1.2 Вибір середовища для розмітки дизайну мобільного додатку

Android Studio є вибором багатьох розробників мобільних додатків, і вибір цього інтегрованого середовища розробки (IDE) для розмітки зовнішнього вигляду за допомогою XML має кілька вагомих причин.

По-перше, Android Studio є офіційним інструментом розробки для Android, розробленим компанією Google. Це гарантує високий рівень сумісності та підтримку для всіх новітніх функцій та змін в Android-екосистемі.

По-друге, Android Studio надає потужний та зручний інтерфейс для роботи з XML-розміткою. Вбудована підтримка автодоповнення, шаблонів коду та валідації дозволяє швидко та ефективно створювати складні макети екранів для мобільних додатків.

По-третє, Android Studio інтегрується з різними інструментами для відлагодження, тестування та визначення продуктивності, що сприяє розвитку високоякісних та ефективних додатків.

По-четверте, велика спільнота розробників використовує Android Studio, що означає наявність широкого спектру онлайн-ресурсів, форумів та документації, які можуть бути корисними при вирішенні проблем та отриманні порад щодо розробки.

Узагальнюючи, обираючи Android Studio для розробки зовнішнього вигляду за допомогою XML, ви отримуєте надійне та потужне середовище, спеціально налаштоване для роботи з Android-додатками, яке сприяє продуктивності та ефективності розробницького процесу.

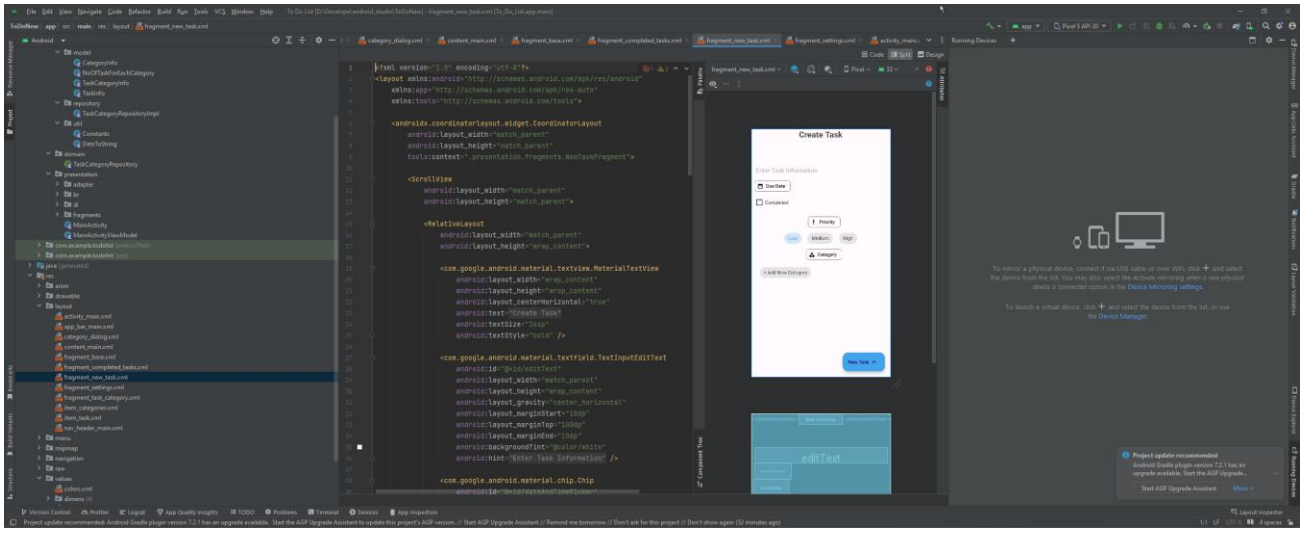


Рисунок 3.2 – Середовище розмітки зовнішнього вигляду програми в Android Studio

Також Android Studio дозволяє тестувати розроблений мобільний додаток не маючи смартфон. Мобільні розробники широко використовують програмні емулятори, які в реальному часі відображають функціонал додатка на віртуальному телефоні. Після кожного збереження коду внесені зміни відображаються на віртуальному телефоні, що робить цей метод тестування належним та

обгрунтованим. Оскільки компіляція проекту в додаток для мобільного телефону може займати значний час, використання емуляторів дозволяє зекономити час і робить процес розробки більш ефективним.

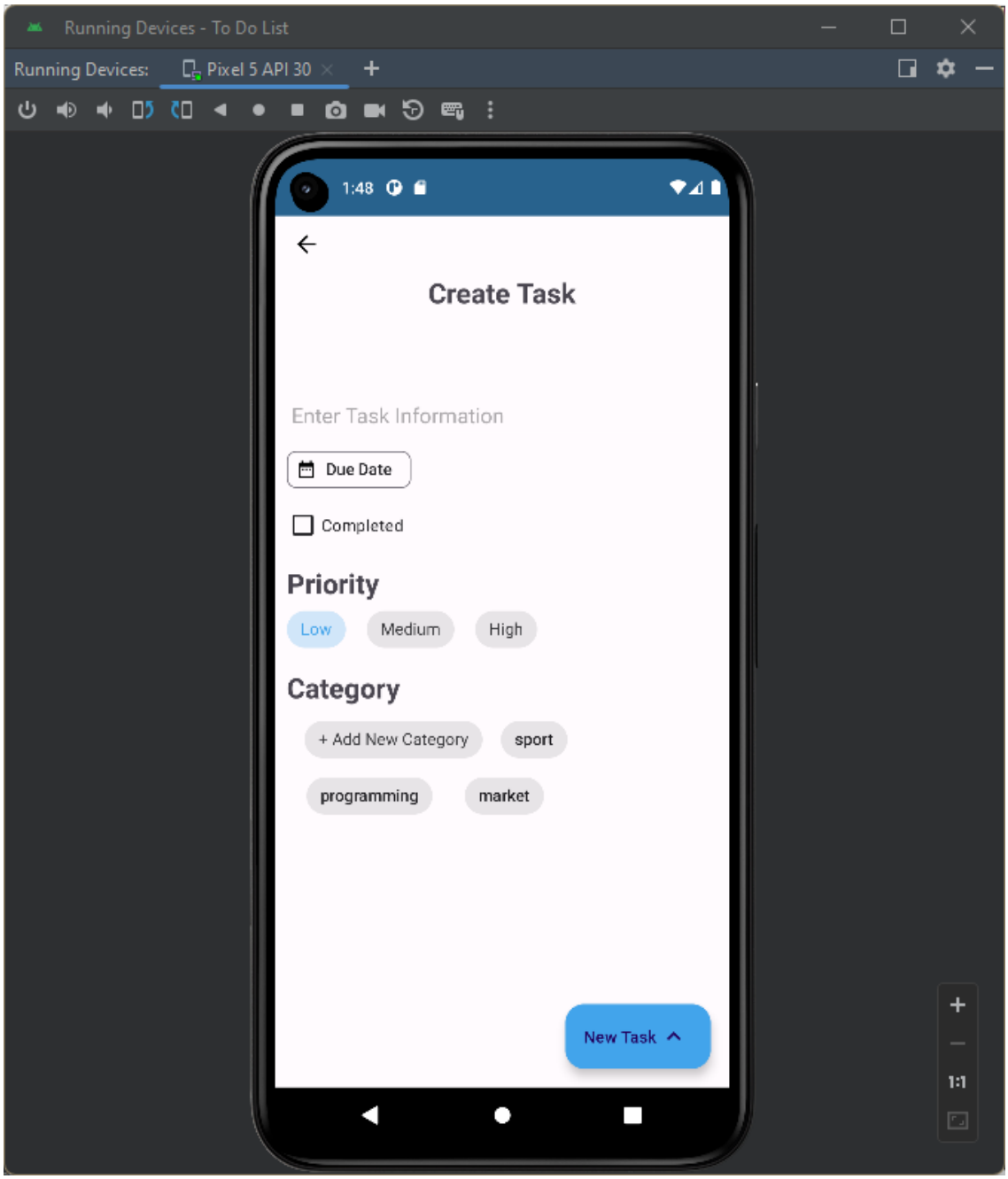


Рисунок 3.3 – Емульований пристрій Pixel 5

3.1.3 Вибір інструментів для роботи з базами даних

Для розробки мобільного додатку на операційній системі Android було обрано SQLite та Room Database, що є добрим та обгрунтованим вибором.

SQLite - це легкий та ефективний реляційний двигун баз даних, який добре підходить для мобільних додатків через свою невелику вагу та простоту використання. Його вбудована підтримка SQL дозволяє легко взаємодіяти з базою даних, використовуючи стандартні запити.

Room Database - це вищий рівень абстракції над SQLite, який надає зручний інтерфейс для роботи з базою даних в межах Android додатків. Room спрощує роботу з SQLite, надаючи анотації для визначення сутностей бази даних, а також автоматично генерує SQL-запити на їх створення та управління. Це сприяє швидкій і раціональній розробці та підтримці бази даних.

Однією з основних переваг використання Room Database є його спрощений підхід до роботи з асинхронними запитами.

Узагальнюючи, використання SQLite та Room Database для розробки мобільного додатку на Android є відмінним рішенням, яке забезпечить ефективне та оптимізоване управління базою даних у проекті.

3.2 Робота над дизайном, розміткою та базою даних

3.2.1 Розробка дизайну додатку

Розробка дизайну для мобільного додатку має величезне значення з кількох ключових причин.

Дизайн є не лише естетичним елементом, але і важливим фактором для привертання уваги користувачів. Привабливий та естетичний інтерфейс створює позитивне враження і сприяє першій взаємодії з додатком.

Зручність використання також залежить від дизайну. Інтуїтивний та логічний розміщення елементів інтерфейсу сприяє зручності користування та взаємодії.

Дизайн визначає ідентичність бренду. Використання оригінального стилю, кольорів та логотипу допомагає формувати впізнаваність та створює унікальний бренд.

Адаптивність дизайну до різних розмірів екранів та пристроїв дозволяє додатку виглядати оптимально на різних платформах.

Розробка дизайну також допомагає визначити, як різні функції додатку будуть виглядати та взаємодіяти, полегшуючи планування та розробку. Загалом, розробка дизайну є важливим кроком для створення успішного та ефективного мобільного додатку.

Початковою ідеєю була розробка програмного продукту де задачі зображуються у вигляді часової діаграми.

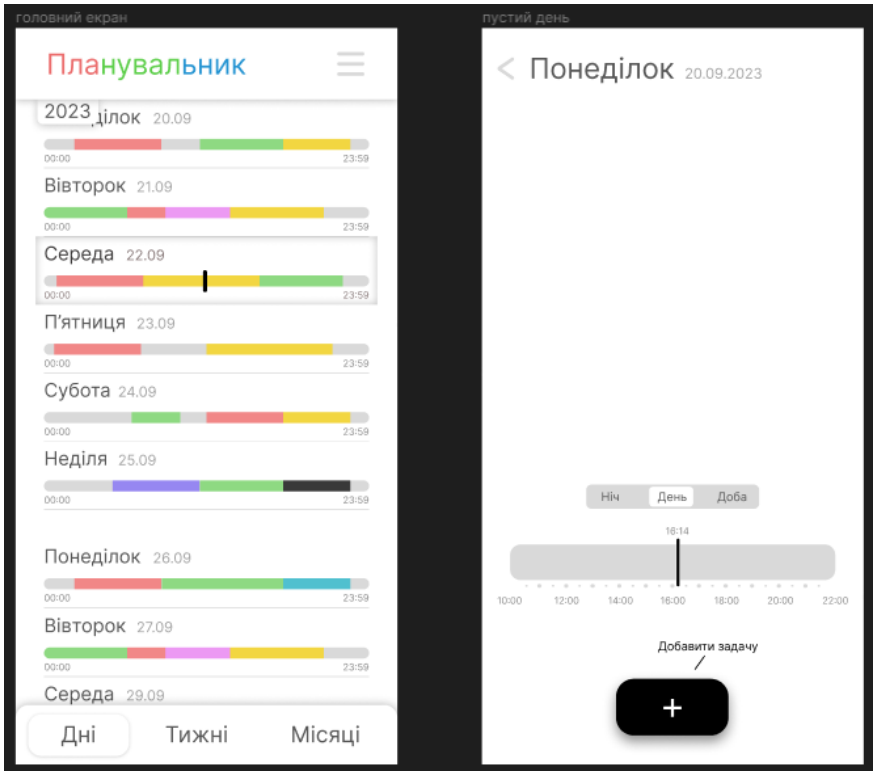


Рисунок 3.4 – Макет головного екрану

Додаток продовжував розроблятись із таким направленням, дизайн дороблявся.

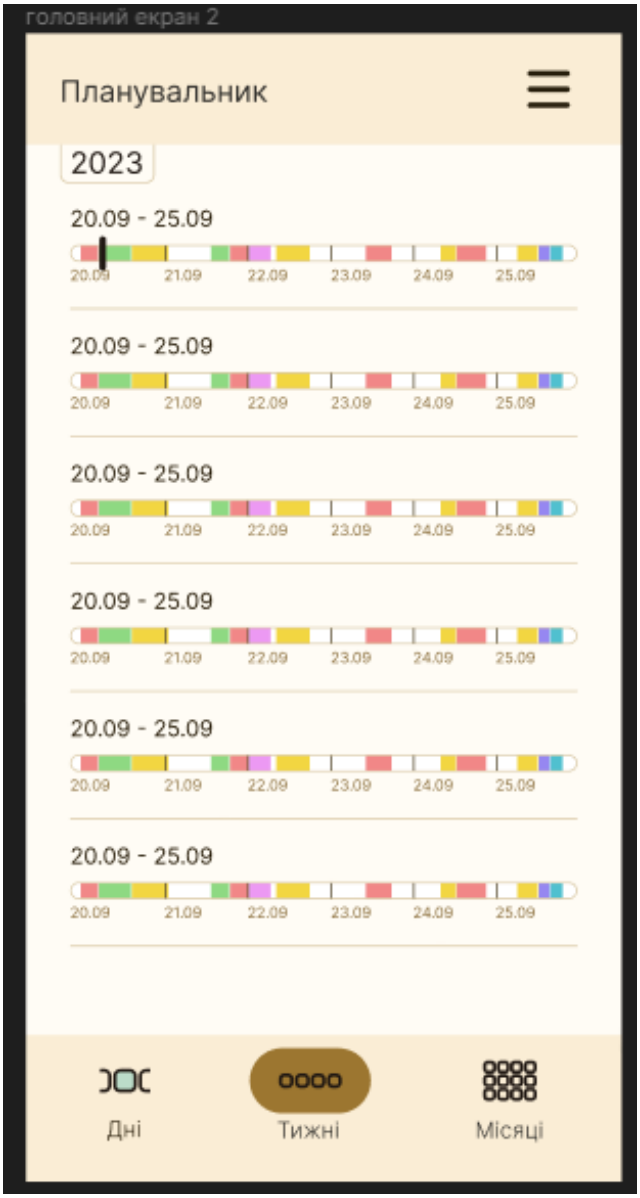


Рисунок 3.5 – Макет дороблeного головного екрану

Так як дизайн тримає в собі багато різних елементів, їх було зібрано в одне місце.

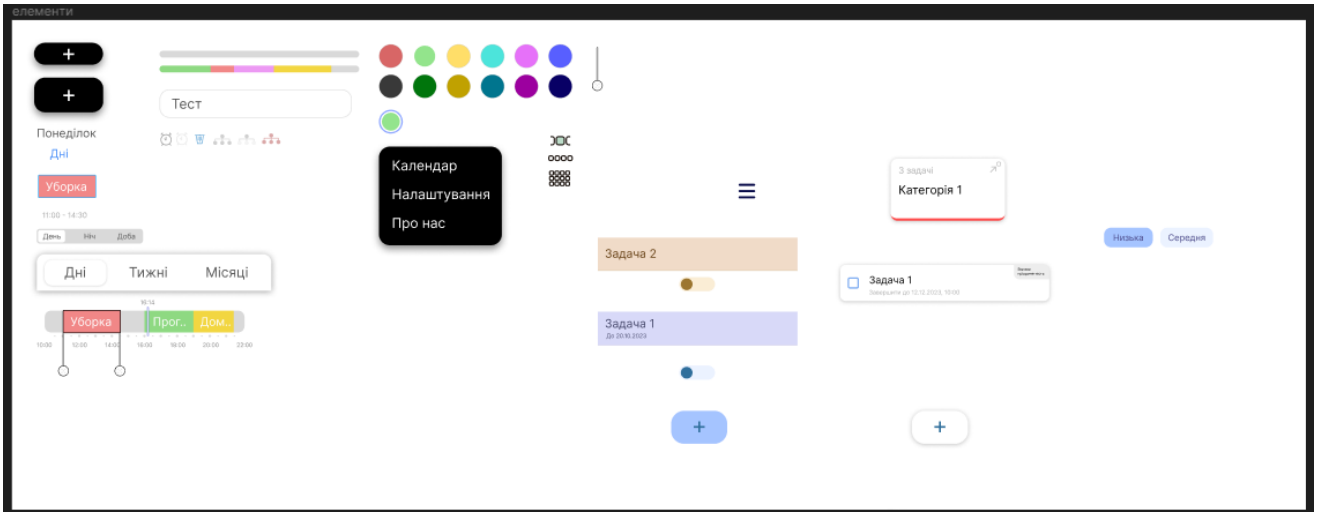


Рисунок 3.6 – Полотно зі всіма елементами

Але з часом стало зрозуміло що даний функціонал буде сильно навантажувати інтерфейс користувача, то ж було прийнято рішення розробити інший дизайн.

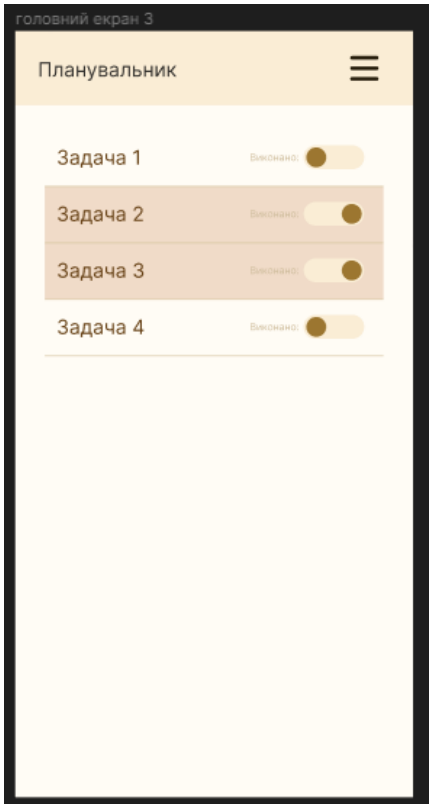


Рисунок 3.7 – Макет головного екрану другої версії

Цю ідею було розвинено далі. Було розроблено різні варіації зовнішнього вигляду програми. Вона повинна бути зручною та виглядати так, щоб користувач одразу розумів для чого цей додаток і як ним користуватись.

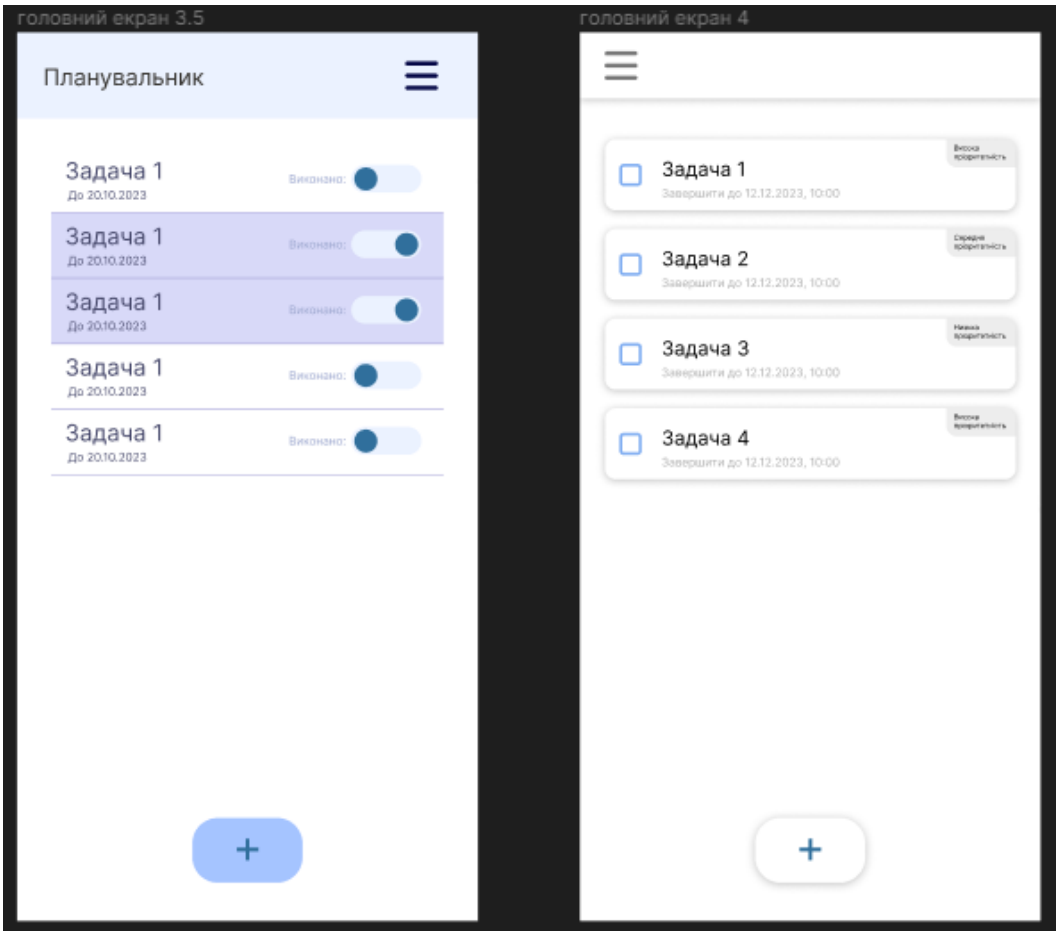


Рисунок 3.8 – Макет дорobленого головного екрану другої версії

Водночас було додано нові функції, такі як пріоритетність задачі і час до якого потрібно її виконати.

Було вирішено добавити категорії та інтерфейс для них. Категорії мають свої кольори для кращої орієнтації. Ці кольори відображуються у кожній задачі.

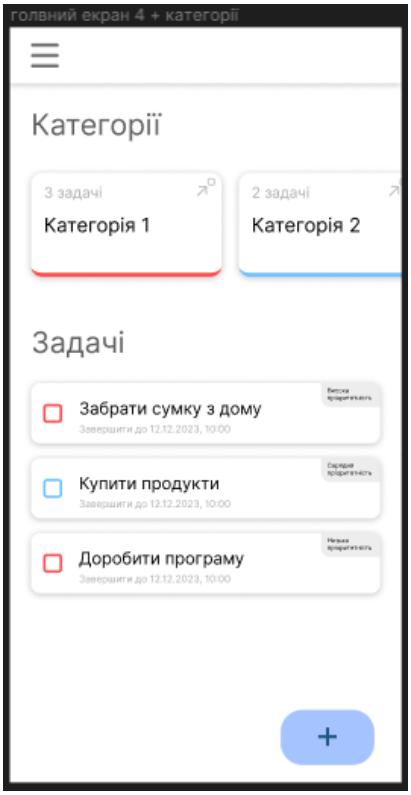


Рисунок 3.9 – Макет головного екрану з категоріями

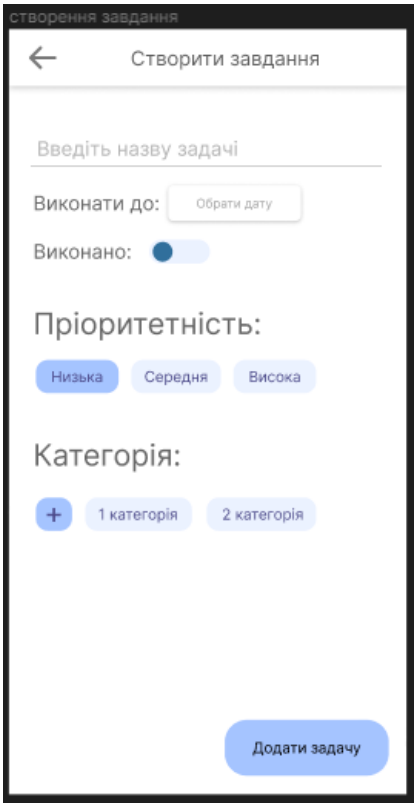


Рисунок 3.10 – Макет вікна для створення задачі

На рисунку 3.10 можна побачити розроблений макет вікна для створення задачі.

3.2.2 Розмітка в Android Studio

Розмітка в Android Studio використовує мову розмітки XML (eXtensible Markup Language)[55], яка дозволяє описувати структуру та вигляд елементів інтерфейсу користувача в мобільних додатках для операційної системи Android.

Кожен макет (layout) визначає структуру UI-компонентів, таких як кнопки, тексти, зображення, тощо, і їхні взаємовідношення на екрані пристрою. XML-розмітка дозволяє розміщувати та конфігурувати ці компоненти за допомогою тегів та атрибутів.

Нижче наведено для прикладу частину коду розмітки для вікна налаштувань.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            tools:context=".presentation.fragments.SettingsFragment">

            <com.google.android.material.textview.MaterialTextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="15dp"
                android:layout_marginBottom="10dp"
                android:text="@string/settings"
                android:textSize="22sp"
                android:textStyle="bold" />
```

```
            <com.google.android.material.textview.MaterialTextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:text="@string/notification"
                android:textSize="16sp"
                android:textStyle="bold" />

            <com.google.android.material.card.MaterialCardView
                style="@style/Widget.App.CardView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:elevation="8dp"
                app:strokeColor="@color/light_grey"
                app:strokeWidth="1dp">

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:padding="15dp"
                    android:text="@string/low_priority_tasks" />


<com.google.android.material.switchmaterial.SwitchMaterial
                    android:id="@+id/low_tasks"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="end"
                    app:thumbTint="@drawable/switch_thumb_selector"
                    app:trackTint="@drawable/switch_track_selector"/>

            </com.google.android.material.card.MaterialCardView>
…
```

У цій розмітці створюється основне полотно *layout*, що позначає собою шар, або вікно. У ньому створюється елемент *ScrollView*, який дозволяє користувачеві пролистати список якщо список буде більший за розмір екрану. В ньому прописуються такі атрибути як *android:layout_width* та *android:layout_height*, які позначають ширину та висоту об’єкту. В даному випадку атрибути мають значення *match_parent*, що означає що елемент буде шириною та висотою такою ж як і у батьківського елемента, тобто на весь екран.

Далі, всередині елементу прописується ще один елемент *LinearLayout*, який дозволяє створювати горизонтальні або вертикальні списки. В даному випадку використовується вертикальний список, який прописується за допомогою атрибуту з таким значенням *android:orientation="vertical"*. Елемент має наступний атрибут із значенням *android:layout_height="wrap_content"*, що означає що висота елемента буде залежати від кількості елементів всередині. Атрибут *tools:context=".presentation.fragments.SettingsFragment"* додає активість **SettingsFragment** який буде використовуватись у даному вікні. В ньому прописані потрібні для роботи функції.

Всередині *LinearLayout* першим елементом прописано *MaterialTextView*, який потрібен для виводу тексту на екран. В атрибуті *android:text* прописано значення *"@string/settings"*. Тобто цей атрибут потрібен для виводу тексту “Settings”. В значенні атрибуту можна було прописати просто “Settings”, але але буде помилка з точки зору розробки додатку, так як текст може мінятись і для того щоб додаток можна було переводити на різні мови, потрібно виписати словник в окремий файл. Вони виписуються у файлі під назвою *strings*. До нього можна звертатись за допомогою *"@string/[слово чи фраза]"*. Слово для цього елемента в словнику виглядає так: *<string name="settings">Settings</string>*. Решта атрибутів описують як напис буде виглядати, та де знаходитись.

Наступним елементом йде *MaterialTextView*, який потрібен для виводу назви списку *Notification***.**

Далі йде елемент *MaterialCardView*, який являє собою компонент інтерфейсу користувача в Android, який надає блок, що виглядає як картка або папка, з округленими кутами та тінями. Це дозволяє створювати ефектні та структуровані макети для відображення інформації чи групування елементів.

Атрибути в цьому елементі встановлюють зовнішній вигляд карти та її розміри. Окремо можна виділити атрибути *android:elevation="8dp"*. Він

налаштовує розмір тіні від карти та *app:strokeWidth="1dp"*, який визначає товщину контурів карти.

Всередині карти знаходяться два елементи: *TextView* та *SwitchMaterial*. Про перше вже було розписано вище, тому зупинимось на *SwitchMaterial*.

SwitchMaterial - це вдосконалена версія елемента вмикання (*Switch*) у бібліотеці Material Design для Android. У свою чергу *Switch* це елемент інтерфейсу користувача в Android, який дозволяє користувачам зручно перемикати між двома станами: увімкнено і вимкнено. Це найпростіший спосіб дозволити користувачам змінювати бінарний параметр у додатку.

В цьому випадку елементу прописано атрибут *android:id="@+id/low_tasks"*. Він надає елементу свій ID для того щоб до нього можна було звернутись через функцію Kotlin. Атрибут *android:layout_gravity="end"* вказує елементу що він повинен знаходитись в кінці батьківського елементу, тобто справа. Атрибути *app:thumbTint="@drawable/switch_thumb_selector"* та *app:trackTint="@drawable/switch_track_selector"* визначають зміну кольору при натисканні пальцями та селектором(мишкою) відповідно. Вони дістають розмітку з інших файлів:

switch_thumb_selector:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:color="@color/dark_blue"
        android:state_checked="true"/>
    <item android:color = "@color/white"/>
</selector>
```

switch_track_selector:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item android:color="@color/light_blue"
    android:state_checked="true"/>
```

```
    <item android:color = "@color/light_grey"/>
</selector>
```

На рисунку 3.11 зображено вікно для якого частково була розписана розмітка вище.

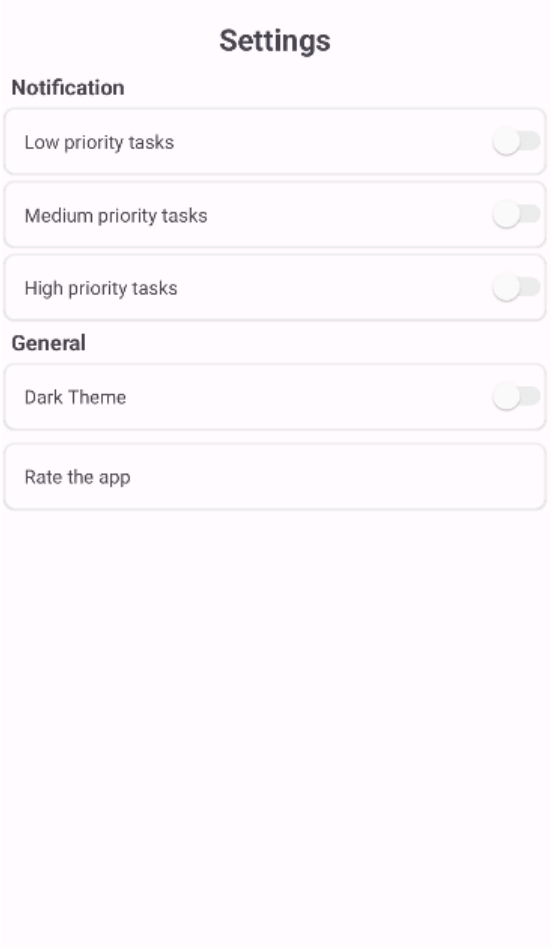


Рисунок 3.11 – Вікно налаштувань

3.2.3 База даних

База даних для мобільного додатку є збереженим електронним набором даних, який дозволяє додатку ефективно організовувати, зберігати та взаємодіяти з інформацією. Основні функції бази даних для мобільного додатку включають:

- Збереження Даних: База даних дозволяє зберігати різноманітні типи даних, такі як тексти, числа, зображення тощо.

- Організація та Структурування: Дані можна організовувати у вигляді таблиць та зв'язків, що допомагає покращити структуру та швидкість доступу до інформації.
- Швидкий та Ефективний Запит: Запити до бази даних дозволяють отримувати швидкий доступ до потрібної інформації, а також фільтрувати, сортувати та аналізувати дані.
- Можливість Оновлення та Модифікації: База даних дозволяє додатку зручно оновлювати, додавати, видаляти та модифікувати інформацію.
- Забезпечення Взаємодії: Додатки можуть використовувати базу даних для взаємодії з іншими компонентами, обміну даними та зберігання стану програми.
- Оптимізація Ресурсів: Збереження даних у базі даних допомагає оптимізувати використання ресурсів пристрою та поліпшує продуктивність додатку.

База даних додатку має наступну структуру, яка накреслена на рисунку 3.12.

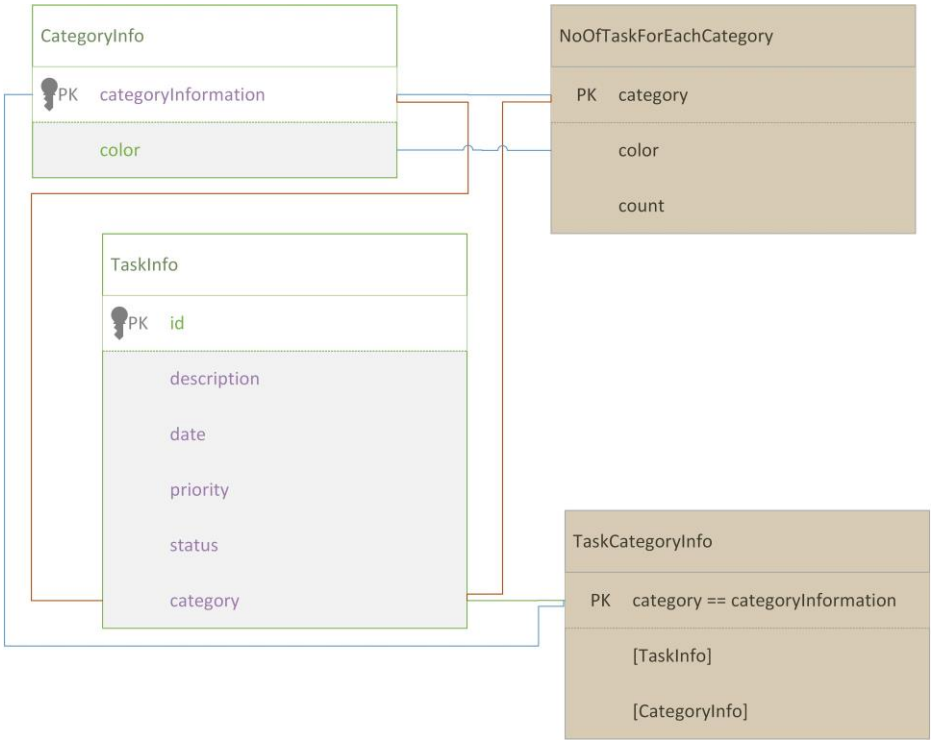


Рисунок 3.12 – Схема структури бази даних додатку

В розробленій базі даних є 4 таблиці. 2 з них є фізично створеними та постійними. Інші 2 створюються під час роботи додатку для зручної взаємодії із даними.

CategoryInfo

Ця таблиця записує категорії та кольори які їх позначають. Вона пов’язана безпосередньо із *TaskInfo* та *NoOFTaskForEachCategory*. Ключем у цій таблиці є назва категорії, стовбець якої називається *categoryInformation*. У стовбець *color* записуєься код кольору у вигляді #RRGGBB.

TaskInfo

Ця таблиця записує інформацію про завдання. Ключем даної таблиці є стовбець *id* у якому записується порядкове значення задачі. Далі таблиця має наступні стовбці:

description - назва задачі.

date – час у який встановлено дедлайн задачі

priority – пріоритетність задачі (Low/Medium/High)

status – статус виконання задачі

category – категорія задачі

Ця таблиця пов’язана із таблицями *CategoryInfo* та *NoOFTaskForEachCategory*.

NoOFTaskForEachCategory

Уявна таблиця яка створюється під час відтворення певних сценаріїв. Зроблена для того щоб можна було відобразити кількість задач певної категорії. Ключовий стовбець *category* у якому записується назва категорії. Інший стовбець *color* записує колір категорії. Стовбець *count* записує кількість задач які є в цій категорії.

TaskCategoryInfo

Уявна таблиця яка створюється щоб об’єднати стовбці двух таблиць: *CategoryInfo* та *TaskInfo*. Ключові стовбці утворюються за допомогою стовбця *category* з таблиці *TaskInfo* та стовбця *categoryInformation* з таблиці *CategoryInfo*.

Room Database дозволяє зберігати запити до таблиць у функціях. Що полегшує роботу з базою даних через Kotlin.

Для прикладу можна використати функцію *getCompletedTask()*.

Він створюється наступним чином:

```
@Query("SELECT * " +
        "FROM taskInfo " +
        "WHERE status = 1 " +
        "ORDER BY date")
fun getCompletedTask(): LiveData<List<TaskCategoryInfo>>
```

Анотація *@Query* позначає запит по якому буде працювати функція. У запиті дістаються рядки з таблиці *taskInfo* де статус виконання завданні = 1, тобто вони виконані. Сортується запит за датою.

Після виконання функції ці рядки вносяться в уявну таблицю *TaskCategoryInfo*.

Є функції для який запит писати не обов’язково. Room Database дозволяє це робити. Це видно на прикладі створення наступної функції:

```
@Delete
suspend fun deleteCategory(categoryInfo: CategoryInfo)
```

Вона буде використовуватись для того щоб видалити певну категорію із таблиці *CategoryInfo*. Для цього потрібно буде ввести назву категорії у параметрах.

Room Database сильно полегшує роботу з базою даних. Для прикладу, попередню функцію Room конвертує наступним чином для того щоб цей запит спрацював:

```
@Override
public Object deleteCategory(final CategoryInfo categoryInfo,
    final Continuation<? super Unit> continuation) {
  return CoroutinesRoom.execute(__db, true, new Callable<Unit>() {
    @Override
    public Unit call() throws Exception {
      __db.beginTransaction();
      try {
        __deletionAdapterOfCategoryInfo.handle(categoryInfo);
        __db.setTransactionSuccessful();
        return Unit.INSTANCE;
      } finally {
        __db.endTransaction();
      }
    }
  }, continuation);
}
```

Незрівнянно коротше і простіше виглядає попередній вираз.

3.3 Висновки

У даному розділі було проведено детальний аналіз мобільного додатку для планування та виконання завдань. Зосереджуючись на процесі розробки та взаємодії з інтерфейсом, було показано, як працювати з розміткою в Android Studio, подаючи конкретні приклади з додатку. Використовуючи XML-розмітку, забезпечено зручний та естетичний вигляд додатку, що сприяє зручному користуванню.

Особливу увагу приділено роботі з базою даних. Було обрано SQLite для зберігання та організації даних додатку, а Room Database використовувався для забезпечення відносин та структури інформації. Це дозволило зручно керувати завданнями та категоріями, а також легко взаємодіяти з даними.

Надалі, робота висвітлила важливі аспекти розробки мобільних додатків, такі як вибір інструментів для розробки дизайну (Figma) та робота з розміткою в

Android Studio. Ці кроки є ключовими для створення функціональних та зручних додатків, які відповідають потребам користувачів.

Узагальнюючи, розділ розкрито основні аспекти розробки мобільного додатку, зосереджуючись на важливих кроках та інструментах, які допомагають створити продуктивний та ефективний додаток для планування завдань.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

В сучасному світі, де час – це один із найцінніших ресурсів, задачі планування і обліку завдань стають надзвичайно важливими. Незалежно від того, чи ви студент, професіонал у сфері бізнесу чи просто людина, яка шукає спосіб ефективніше управляти своїм часом, автоматизована система планування стане надійним помічником. Вона допомагає організовувати завдання, визначати пріоритети та раціонально розподіляти час для досягнення максимальної продуктивності.

Однією з основних проблем, які стикаються користувачі, є недостатня ефективність традиційних методів планування, таких як записи вручну або використання паперових щоденників. Ці методи вимагають багато часу і зусиль, і часто не забезпечують гнучкість і зручність в управлінні завданнями. Тому розробка автоматизованої системи планування стає актуальним завданням, яке може революціонізувати спосіб, яким ми ведемо своє повсякденне життя і роботу.

Основною метою цієї роботи є розробка програмного забезпечення, яке допоможе користувачам ефективніше планувати, відстежувати і виконувати завдання. Вона буде спрямована на полегшення процесу планування і оптимізацію використання часу, щоб досягти кращих результатів. Завдяки цій роботі, ми маємо намір внести вагому спільний внесок у покращення управління часом та ефективності роботи кожного користувача.

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Магістерська кваліфікаційна робота за темою тему «Розробка автоматизованої системи планування і обліку завдань.»: передбачає розробку програмного

забезпечення, яке може задовільнити окремих осіб або компаній розраховуючи свій час та плани на поточний день/тиждень/місяць.

Проведемо оцінювання комерційного потенціалу даної розробки. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу буде проведено на основі відгуку трьох експертів: Ковалюк Олег Олександрович – науковий керівник, доцент кафедри комп`ютерних систем управління ВНТУ, кандидат технічних наук; Кавецький Вячеслав Валерійович кандидат економічних наук, доцент кафедри економіки підприємства і виробничого менеджменту ВНТУ; Галущак Анастасія Володимирівна – асистент кафедри комп'ютерних наук ВНТУ; із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із таблицею 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	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 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	Критерій	Критерій	Критерій	Критерій	Критерій
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

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	Критерій	Критерій	Критерій	Критерій	Критерій
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки занесемо у таблицю 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Ковалюк О. О.	Кавецький В. В.	Галущак А. В.
	Бали, виставлені експертами:		
1	3	2	3
2	3	2	3
3	3	4	3
4	3	3	4
5	3	3	2
6	3	3	2
7	4	3	2
8	3	3	3
9	3	3	2
10	4	4	4
11	4	3	3
12	3	3	2
Сума балів	$СБ_1$=39	$СБ_2$=36	$СБ_3$=33
Середньоарифметична сума балів	*(39+36+33) / 3 = 36*		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 − Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків	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 36 балів. Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є вище середнього. Дана розробка буде корисним продуктом для людей в яких власний час дорожче за гроші, а гроші можна отримати за час. Відносно конкурентів вона буде кращою завдяки своєму зручному інтерфейсу. Основний елемент програми, який дозволяє бачити користувачеві свої задачі у так званому таймлайні, дозволяє програмному забезпеченню виділятися на фоні конкурентів, та надавати додаткові можливості при користуванні програмою, а значить і при плануванні свого часу.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення. Процес прогнозування можна розділити на три етапи:

- розрахунок витрат на виконавців даного розділу роботи;
- розрахунок загальних витрат на виконання роботи;
- прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат, основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \qquad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (дослідника), грн;

$T_р$ – число робочих днів в місяці, 22 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проєкту, дизайнер, та спеціаліст по базам даних	12000	550	52	28600
Експерт у предметній галузі	19000	400	30	25900
Програміст-спеціаліст	23000	1100	50	55000
Всього				109500

Додаткова заробітна плата прийнято розраховувати як 15 % від основної заробітної плати розробників та робітників:

$$З_д = З_о * \frac{15\%}{100\%}; \quad (4.2)$$

$$З_д = 109500 * \frac{15\%}{100\%} = 16425 \text{ (грн.)}.$$

Нарахування на заробітну плату $Н_{ЗП}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується за формулою:

$$Н_з = (З_о + З_д) * \frac{\beta}{100\%}; \quad (4.3)$$

де $З_0$ – основна заробітна плата розробника, грн;

$З_Д$ – додаткова заробітна плата розробника, грн;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (109500 + 16425) * \frac{22\%}{100\%} = 27703{,}5 \text{ (грн.)}.$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір для принтера А4 Zoom 500шт., щільність 80 г/м²	260	1	-	-	268
Ручка масляна FINEGRIP Cello синя 5 шт в упак Cello	42	2	-	-	92,4

Продовження таблиці 4.5

Маркер перманентний Buromax BM.8703-01	7,62	4	-	-	33,528
Всього:					393,928

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \tag{4.4}$$

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

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

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

k – кількість найменувань програмних засобів.

Отримані результати необхідно звести до таблиці.

Таблиця 4.6 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11 Pro	1	10700	11877
Прикладний пакет Microsoft Office	1	7790	8646,9
Програмне середовище розробки Android Studio	1	100	111

Продовження таблиці 4.6

Програмне середовище для створення макетів Figma	1	120	133,2
Всього:			20768,1

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{\text{Ц}}{\text{Т}_\text{в}} * \frac{t_{\text{вик}}}{12} \text{ [грн.]}, \quad (4.5)$$

де *Ц* – балансова вартість обладнання, грн;

Т – термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ – термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на смартфон балансова вартість якого становить 14000 грн, термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 3 міс.

$$\text{А}_{\text{обл}} = \frac{14000}{2} * \frac{3}{12} = 1750 \text{ (грн)}.$$

Визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.7. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$\text{А}_{\text{н.р.}} = \text{Ц}_{\text{н.р.}} * \text{Н}_\text{а} * \frac{t_{\text{вик}}}{12}. \quad (4.6)$$

Проте, вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю.

Таблиця 4.7 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна периферія	26300	2	3	3287,5
Смартфон для тестування	14000	2	3	1750
Приміщення	400000	20	3	5000
Всього				10037,5

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільчих компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільчих компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В * П * Ф * К_п, \quad (4.7)$$

де *В* – вартість 1 кВт-години електроенергії для 1 класу підприємства, *В* = 7,5 грн/кВт;

П – встановлена потужність обладнання, кВт. *П* = 0,5 кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9,

Використаємо дану формулу для розрахунку споживання електроенергії комп'ютером:

$$В_е = 7{,}5 * 0{,}5 * 250 * 0{,}9 = 843{,}75 \text{ (грн.)}.$$

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Комп'ютер та комп'ютерна периферія	0,5	250	843,75
Смартфон	0,005	120	4,05
Всього:			847,8

Статті «Витрати на службові відрядження» немає, то ж витрат на них не передбачено.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$І_в = (З_о * З_р) * \frac{Н_{ів}}{100\%}, \quad (4.8)$$

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

$$І_в = 109500 * \frac{110\%}{100\%} = 120450 \text{ (грн.)}.$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з

набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$\text{Н}_{\text{нзв}} = \left(\text{З}_{\text{о}} * \text{З}_{\text{р}}\right) * \frac{\text{Н}_{\text{ів}}}{100\%}, \qquad (4.9)$$

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

$$\text{Н}_{\text{нзв}} = 109500 * \frac{120\%}{100\%} = 131400\ (\text{грн.}).$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$\text{В}_{\text{заг}} = 109500 + 16425 + 27703{,}5 + 393{,}928 + 20768{,}1 + 10037{,}5 + 847{,}8 + 120450 + 131400 = 437\,252{,}828(\text{грн.}).$$

Фінальним етапом проведемо обрахунок загальних витрат на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$\text{ЗВ} = \frac{\text{В}_{\text{заг}}}{\eta}\,[\text{грн.}], \qquad (4.10)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії:

– науково-дослідних робіт, то η=0,1;
– технічного проектування, то η=0,2;

– розробки конструкторської документації, то η=0,3;
– розробки технологій, то η=0,4;
– розробки дослідного зразка, то η=0,5;
– розробки промислового зразка, то η=0,7;
– впровадження, то η=0,9.

Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = \frac{437\ 252{,}828}{0{,}9} = 486\ 139{,}809(грн).$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Витрати на провадження дослідження складаються з витрат на провадження розробок за напрямками «Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних» та «Розробка автоматизованої системи планування і обліку завдань. Частина 2. Розробка мобільного додатку». Сумарні витрати складуть:

$$ЗВ = 619\ 839{,}79 + 486\ 139{,}809 = 1\ 105\ 979{,}6$$

Збільшення чистого прибутку є позитивним результатом, який може отримати інвестор при вкладені певного об’єму ресурсів. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації розроблювального інформаційного продукту.

В даній ситуації розглядається розробка чи суттєве вдосконалення програмного засобу. Тоді майбутній економічний ефект буде формуватися на основі таких даних: ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик; N –кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки; $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.11)$$

де $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

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

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні 300 грн за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 30000 шт., протягом другого року – на 20000 шт., протягом третього року на 18000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П1 = (500 * 100000 + (0 + 300) * 30000) * 0{,}8333 * 0{,}49 * (1 - 0{,}18) = \\ = 19\,754\,376{,}46\ (\text{грн.}),$$

$$\Delta П2 = (500 * 100000 + (300 + 100) * (30000 + 20000)) * 0{,}8333 * 0{,}49 * (1 - 0{,}18) \\ = 23\,437\,395{,}8(\text{грн.}),$$

$$\Delta П3 = (500 * 100000 + (400 + 100) * (30000 + 20000 + 18000)) * 0{,}8333 * 0{,}49 \\ * (1 - 0{,}18) = 28\,124\,874{,}96(\text{грн.}).$$

Отже, відповідно до проведених розрахунків, прогнозований комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Проведемо розрахунок ефективності вкладених інвестицій та періоду їх окупності.

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

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.12)$$

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

T – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

$$\text{ПП} = \left(\frac{19\,754\,376{,}46}{(1+0{,}15)^1}\right) + \left(\frac{23\,437\,395{,}8}{(1+0{,}15)^2}\right) + \left(\frac{28\,124\,874{,}96}{(1+0{,}15)^3}\right) =$$
$$= 17\,177\,718{,}66 + 17\,722\,038{,}41 + 18\,492\,561{,}82 = 53\,392\,318{,}89\ (\text{грн.}).$$

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

$$PV = k_{\text{інв}} * \text{ЗВ}, \qquad (4.13)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

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

$$PV = 2 * 1\,105\,979{,}6 = 2\,211\,959{,}2(\text{грн.}).$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{Е}_{\text{абс}} = \text{ПП} - PV, \qquad (4.14)$$

$$\text{Е}_{\text{абс}} = 53\,392\,318{,}89 - 2\,211\,959{,}2 = 51\,180\,359{,}7\ (\text{грн.}).$$

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

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1, \qquad (4.15)$$

де $T_{ж}$ – життєвий цикл наукової розробки, роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{51\,180\,359{,}7}{2\,211\,959{,}2}} - 1 = 1{,}9.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (4.16)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

f– показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}14 + 0{,}4 = 0{,}54.$$

Оскільки $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}, \qquad (4.17)$$

$$\mathrm{T_{ок}} = \frac{1}{1{,}9} = 0{,}53(\text{р.}).$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,53 роки, то фінансування даної наукової розробки є доцільним.

4.4 Висновки

Економічна частина роботи містить розрахунок витрат на розробку нового програмного продукту. Загальна сума витрат складає 1 105 979,6 гривень.

Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні цієї розробки.

Термін окупності розробки програмного забезпечення складає 0,53 роки. Тому інвестування в розробку автоматизованої системи планування і обліку завдань є доцільним для інвесторів.

ВИСНОВКИ

Проведений аналіз проблем автоматизованих систем планування та обліку завдань, детальний розгляд предметної області мобільних додатків та аналіз існуючих систем створили основу для вдосконалення та новаторського підходу в розробці. Проведено дослідження історичного розвитку та аналіз аналогів вказали на важливість інновацій та виокремлення функціональних можливостей для задоволення різноманітних потреб користувачів. Перелік аспектів для розробки мобільного додатку планувальника узагальнює різноманітні функції та характеристики, спрямовані на високу ефективність та комфорт користування. Аналіз та вибір API дозволили використовувати ключові інструменти для поліпшення функціональності та взаємодії з іншими системами, підкреслюючи комплексний підхід до розробки мобільних додатків планувальників.

Обрані методи розробки, такі як використання Figma для дизайну, SQLite для бази даних та Room Database для спрощення розробки, обрані на основі їхньої вартості та функціональності. Підходячи до дизайну через Figma забезпечує ефективність та комфортну роботу. Вибір SQLite для бази даних обумовлений офіційним статусом для Android, а Room Database допомагає структурувати інформацію та керувати завданнями та категоріями зручно та ефективно.

Детальний аналіз розробки мобільного додатку для планування задач підкреслив важливість роботи з розміткою в Android Studio та вибір оптимальних інструментів. Використання XML-розмітки у Android Studio дозволило створити зручний та естетичний інтерфейс. Робота з базою даних через SQLite та Room Database спрощує організацію та взаємодію з даними. Вибір інструментів для розробки дизайну та роботи з базою даних відображає важливі етапи розробки, спрямовані на створення функціональних та зручних додатків для задоволення потреб користувачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Що таке ERP система, IT компанія «Розумний кубик», Альбіна Белякова, 2018р. [Електронний ресурс]– URL: https://kubik.com.ua/ua/articles/baserp-vs-1cupp (Дата звернення 09.09.2023)
2. Що таке електронна пошта?, FutureNow, 2020р. [Електронний ресурс] – URL: https://futurenow.com.ua/elektronna-poshta-tse-shho-take-e-mail-vse-shho-vam-treba-znaty/ (Дата звернення 09.09.2023)
3. Комп'ютерна платформа, Wikipedia, 2022р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0 (Дата звернення 09.09.2023)
4. Адаптація, ВУЕ, С. В. Межжерін [Електронний ресурс] – URL: https://vue.gov.ua/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B0%D1%86%D1%96%D1%8F (Дата звернення 09.09.2023)
5. Соціальна психологія, Лідія Орбан-Лембрик, 2006р.
6. Портативний, Словник іншомовних слів, Володимир Лук'янюк [Електронний ресурс] – URL: https://www.jnsm.com.ua/cgi-bin/u/book/sis.pl?Article=14781&action=show (Дата звернення 10.09.2023)
7. Що таке Інтернет?, FutureNow, 2019р. [Електронний ресурс] – URL: https://futurenow.com.ua/shho-take-internet/ (Дата звернення 10.09.2023)
8. Віртуальний помічник, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BF%D0%BE%D0%BC%D1%96%D1%87%D0%BD%D0%B8%D0%BA (Дата звернення 10.09.2023)

9. About SQLite, SQLite Comp., 2023р. [Електронний ресурс] – URL: https://www.sqlite.org/about.html (Дата звернення 14.09.2023)
10. Room Persistent Library, Geeks for Geeks, 2022р. [Електронний ресурс] – URL: https://www.geeksforgeeks.org/introduction-to-room-persistent-library-in-android/ (Дата звернення 14.09.2023)
11. Kotlin, JetBrains [Електронний ресурс] – URL: https://kotlinlang.org/ (Дата звернення 14.09.2023)
12. What does asynchronous mean?, Tech Target, Ben Lutkevich, 2019р. [Електронний ресурс] – URL: https://www.techtarget.com/searchnetworking/definition/asynchronous (Дата звернення 14.09.2023)
13. Compilation, Computer Hope, 2022р. [Електронний ресурс] – URL: https://www.computerhope.com/jargon/c/compilat.htm (Дата звернення 14.09.2023
14. Figma, Toptal, LLC, Ben Kopf [Електронний ресурс] – URL: https://www.toptal.com/designers/ui/figma-design-tool (Дата звернення 14.09.2023)
15. Canvas, Graphic Picra Inc. [Електронний ресурс] – URL: https://www.graphic.com/docs/canvas (Дата звернення 15.09.2023)
16. What is a Programming Library?, Career Foundry, Rachel Meltzer, 2023р. [Електронний ресурс] – URL: https://careerfoundry.com/en/blog/web-development/programming-library-guide/ (Дата звернення 15.09.2023)
17. Figma Prototyping, Figma [Електронний ресурс] – URL: https://www.figma.com/prototyping/ (Дата звернення 15.09.2023)
18. Figma cloud technology, Best Website Builders, Kathy McFarland, 2022р. [Електронний ресурс] – URL: https://www.websitebuilderinsider.com/what-cloud-does-figma-use/ (Дата звернення 15.09.2023)

19. PNG, Adobe [Електронний ресурс] – URL: https://www.adobe.com/creativecloud/file-types/image/raster/png-file.html (Дата звернення 15.09.2023)
20. SVG, Adobe [Електронний ресурс] – URL: https://www.adobe.com/creativecloud/file-types/image/vector/svg-file.html (Дата звернення 15.09.2023
21. CSS, Mozilla org., MDN contributors, 2023р.[Електронний ресурс] – URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS (Дата звернення 15.09.2023)
22. Zeplin, Zeplin io [Електронний ресурс] – URL: https://zeplin.io/ (Дата звернення 26.09.2023)
23. Adobe XD, Adobe [Електронний ресурс] – URL: https://www.adobe.com/ru/products/xd/learn/get-started/what-is-adobe-xd-used-for.html (Дата звернення 26.09.2023)
24. Adobe Creative Cloud, Adobe [Електронний ресурс] – URL: https://www.adobe.com/ua/creativecloud.html (Дата звернення 26.09.2023)
25. Adobe Photoshop, Adobe [Електронний ресурс] – URL: https://www.adobe.com/ua/products/photoshop.html (Дата звернення 26.09.2023)
26. Adobe Illustrator, Adobe [Електронний ресурс] – URL: https://www.adobe.com/ua/products/illustrator.html (Дата звернення 27.09.2023)
27. Filorax Organisers, FILOFAX - FLB GROUP [Електронний ресурс] – URL: https://filofax.com/collections/organisers (Дата звернення 27.09.2023)
28. Day Timer Planner, Day Timer [Електронний ресурс] – URL: https://www.daytimer.com/ (Дата звернення 27.09.2023)
29. Microsoft Outlook, Microsoft, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Microsoft_Outlook (Дата звернення 27.09.2023)

30. Lotus Organizer, Wikipedia, 2016р. [Електронний ресурс] – URL: https://ru.wikipedia.org/wiki/Lotus_Organizer (Дата звернення 02.10.2023)
31. Google Calendar, Wikipedia, 2023р. [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Google_Calendar (Дата звернення 02.10.2023)
32. iCloud Calendar, Apple Inc. [Електронний ресурс] – URL: https://www.icloud.com/calendar (Дата звернення 02.10.2023)
33. Evernote, Evernote Comp. [Електронний ресурс] – URL: https://evernote.com/ (Дата звернення 03.10.2023)
34. Todoist, , Doist Inc. [Електронний ресурс] – URL: https://todoist.com/ (Дата звернення 03.10.2023)
35. Asana, Asana, Inc. [Електронний ресурс] – URL: https://asana.com/ (Дата звернення 11.10.2023)
36. Trello, Atlassian [Електронний ресурс] – URL: https://trello.com/uk (Дата звернення 11.10.2023)
37. Штучний інтелект, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82 (Дата звернення 11.10.2023)
38. Any.do, Any.do Team [Електронний ресурс] – URL: https://www.any.do/en (Дата звернення 11.10.2023)
39. Sectograph, Laboratory 27 [Електронний ресурс] – URL: https://sectograph.com/ (Дата звернення 11.10.2023)
40. To Do List, Dmyrto Tatarenko [Електронний ресурс] – URL: https://apps.apple.com/ua/app/todo-list-tasks/id1659401476?l=uk (Дата звернення 11.10.2023)
41. Як зробити аналіз цільової аудиторії?, Shapoval Agency, 2021р. [Електронний ресурс] – URL:

https://cases.media/article/yak_zrobiti_analiz_cilovoyi_auditoriyi (Дата звернення 15.10.2023)

42. Android, Google LLC [Електронний ресурс] – URL: https://www.android.com (Дата звернення 15.10.2023)
43. iOS, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/IOS (Дата звернення 20.10.2023)
44. What is Backup?, Acronis International GmbH., 2023р. [Електронний ресурс] – URL: https://www.acronis.com/en-us/blog/posts/data-backup/ (Дата звернення 20.10.2023)
45. Google Play, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Google_Play (Дата звернення 20.10.2023)
46. App Store, Wikipedia, 2022р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/App_Store (Дата звернення 20.10.2023)
47. What is Friendly Interface, Rodrigo Augusto Peres Velozo, 2017
48. СУБД, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85 (Дата звернення 20.10.2023)
49. Microsoft Windows, Wikipedia [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Microsoft_Windows, 2023р. (Дата звернення 21.10.2023)
50. Linux, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Linux (Дата звернення 21.10.2023)
51. macOS, Wikipedia [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/MacOS, 2023р. (Дата звернення 21.10.2023)

52. Android Architecture Components, Ihor Kucherenko, 2017р. [Електронний ресурс] – URL: https://proandroiddev.com/android-architecture-components-cb1ea88d3835 (Дата звернення 21.10.2023)
53. Data access object(DAO), Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/Data_access_object (Дата звернення 21.10.2023)
54. Save data in a local database using Room, Google For Developers, 2023р. [Електронний ресурс] – URL: https://developer.android.com/training/data-storage/room#kts (Дата звернення 26.10.2023)
55. XML, Wikipedia, 2023р. [Електронний ресурс] – URL: https://uk.wikipedia.org/wiki/XML (Дата звернення 26.10.2023)

ДОДАТКИ

Додаток А – Протокол кваліфікаційної роботи

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 95,7% Схожість 4,3%

Аналіз звіту подібності (відмітити потрібне)

ᵛ 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 розгляд експертної комісії кафедри.
□ 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 _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Владислав ПОДОЛЯНИН
(підпис) (прізвище, ініціали)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (прізвище, ініціали)

Додаток Б – Технічне завдання

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о.Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
" 06 " 10 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних.»

08-33.МКР.08.00.000 ТЗ

Студент групи 2АКІТ-22М

_____ Владислав ПОДОЛЯНИН
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н., доцент

_____ Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних.

1.2. Галузь застосування – Планування і облік завдань.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ №247 від “18” вересня 2023 року.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення ефективності планування та виконання завдань.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. "How To Use Graphic Design To Sell Things- автор Michael Bierut

2. "Database Systems" – автори: Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom

3. "SQL QuickStart Guide" - автор Walter Shields.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Запис завдання;
- Редагування завдання;
- Відмітки виконання завдання.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10/11;
- Середовище для розробки дизайну;

- Середовище для розмітки Android Studio;
- СУБД SQLite.

5.2.2. Умови експлуатації системи:

- робота на мобільних пристроях;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів планування та обліку завдань 03.10.2023 р.
2. Розробка дизайну мобільного додатку 06.10.2023 р.
3. Робота над розміткою зовнішнього вигляду 12.11.2023 р.
4. Розробка баз даних 22.11.2023 р.

6.2 Графічні матеріали:

1. Розробка моделі системи 21.11.2023 р.
2. Розробка моделі бази даних системи 24.11.2023 р.
3. Тестування програмного забезпечення 26.11.2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до 28.11.2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до 11.12.2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до 20.12.2023 р.

Додаток Д – Ілюстративна частина

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка автоматизованої системи планування і обліку завдань. Частина 1.

Розробка дизайну мобільного додатку та бази даних

1. Вступ
2. Мета, предмет та практична цінність дослідження;
3. Аналіз аналогів;
4. Any.do
5. Sectograph
6. ToDo List & Tasks
7. Вибір операційної системи для додатку
8. Інструмент для розробки дизайну програми Figma;
9. Стадії розробки дизайну;
10. Стадії розробки дизайну;
11. Стадії розробки дизайну;
12. Стадії розробки дизайну;
13. Середовище розмітки зовнішнього вигляду програми в Android Studio;
14. Приклад розмітки на екрані налаштувань;
15. Структура бази даних;
16. Приклад створення методу DAO в Room Database;
17. Вигляд конвертованої функції deleteCategory()
18. Висновки

Виконав: студент 2-го курсу, групи
2АКІТ-22м

Владислав ПОДОЛЯНИН

Керівник: Керівник к.т.н., доцент
Олег КОВАЛЮК

«01» 12 2023 р.

Опонент: к.т.н., доцент каф. АІІТ
Роман МАСЛІЙ

«06» 12 2023 р.

Розробка автоматизованої системи планування і обліку завдань. Частина 1. Розробка дизайну мобільного додатку та бази даних

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
СТУДЕНТ ГРУПИ 2АКІТ-22М
ПОДОЛЯНИН ВЛАДИСЛАВ
КЕРІВНИК РОБОТИ
ДОЦЕНТ КАФ. КСУ
КОВАЛЮК О. О.

МЕТА, ПРЕДМЕТ ТА ПРАКТИЧНА ЦІННІСТЬ

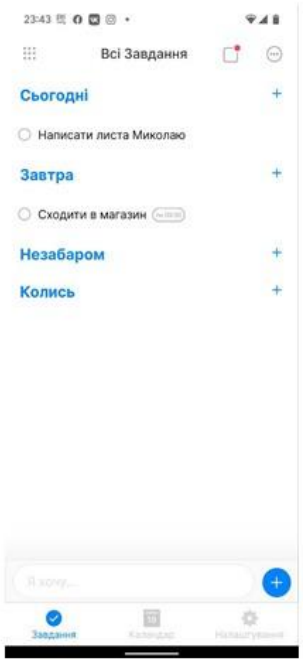
Мета дослідження. Основною метою дослідження є покращити можливості та комфорт користувачів у плануванні завдань.

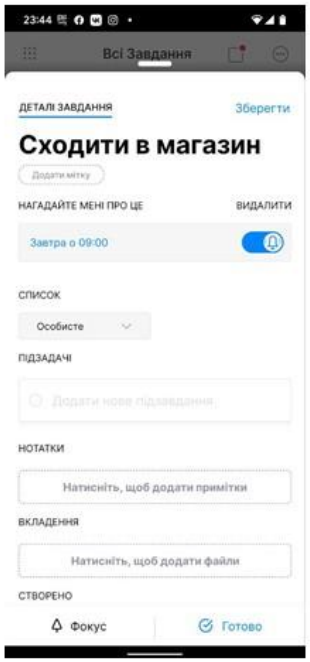
Предмет дослідження. Предметом дослідження є розробка дизайну програми та бази даних для автоматизованої системи, спрямованої на зручне та ефективне планування та відстеження завдань.

Практична цінність. Розроблена система буде корисною для широкого кола користувачів, які прагнуть до підвищення своєї продуктивності та ефективності управління завданнями, особливо в умовах активного використання смартфонів.

АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

Any.do

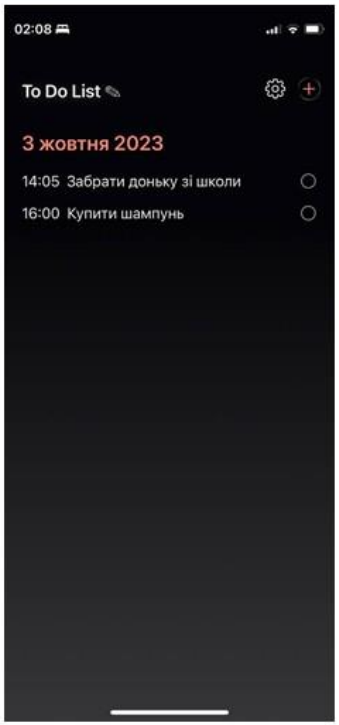

23:43
Всі Завдання
Сьогодні
Написати листа Миколаю
Завтра
Сходити в магазин
Незабаром
Колись
Я хочу...
Завдання
Календар
Налаштування

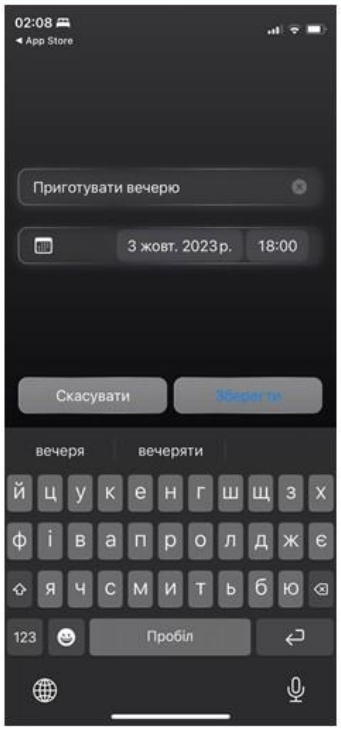

23:44
Всі Завдання
ДЕТАЛІ ЗАВДАННЯ
Зберегти
Сходити в магазин
Додати мітку
НАГАДАЙТЕ МЕНІ ПРО ЦЕ
ВИДАЛИТИ
Завтра о 09:00
СПИСОК
Особисте
ПІДЗАДАЧІ
Додати нове підзавдання
НОТАТКИ
Натисніть, щоб додати примітки
ВКЛАДЕННЯ
Натисніть, щоб додати файли
СТВОРЕНО
Фокус
Готово

SECTOGRAPH


23:47
Sectograph
Груд. 18
неділя
16:08
нд 18
18 Груд. 2022
неділя
Забрати посилку
1h. 16:30 – 17:30
Купити апельсина
1h. 22:00 – 23:00

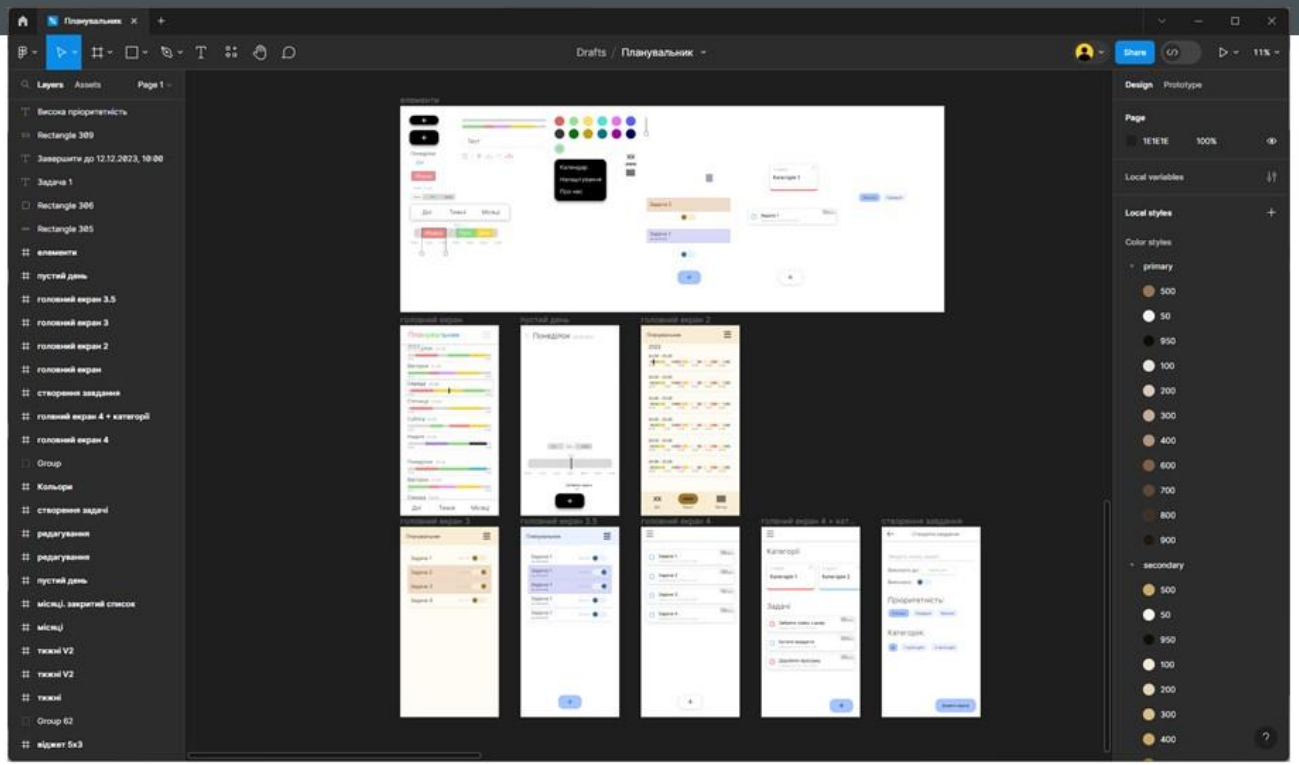
ToDo List & Tasks



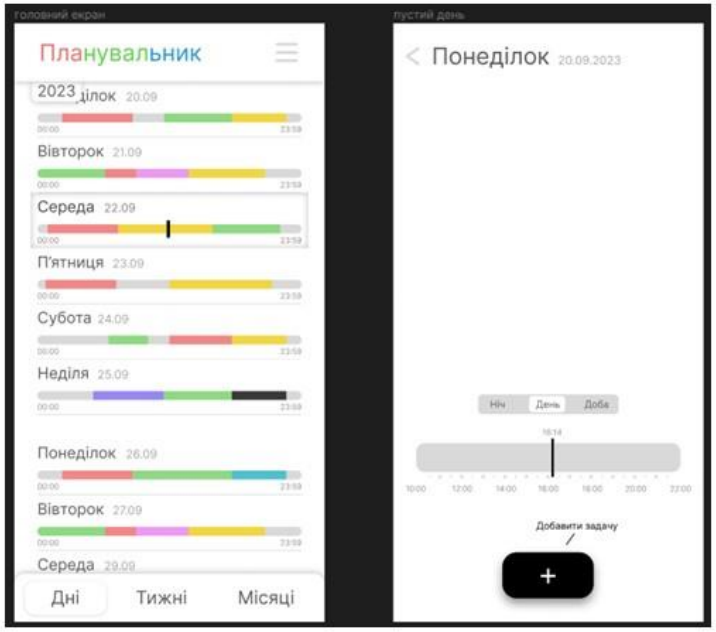


ВИБІР ОПЕРАЦІЙНОЇ СИСТЕМИ ДЛЯ ДОДАТКУ

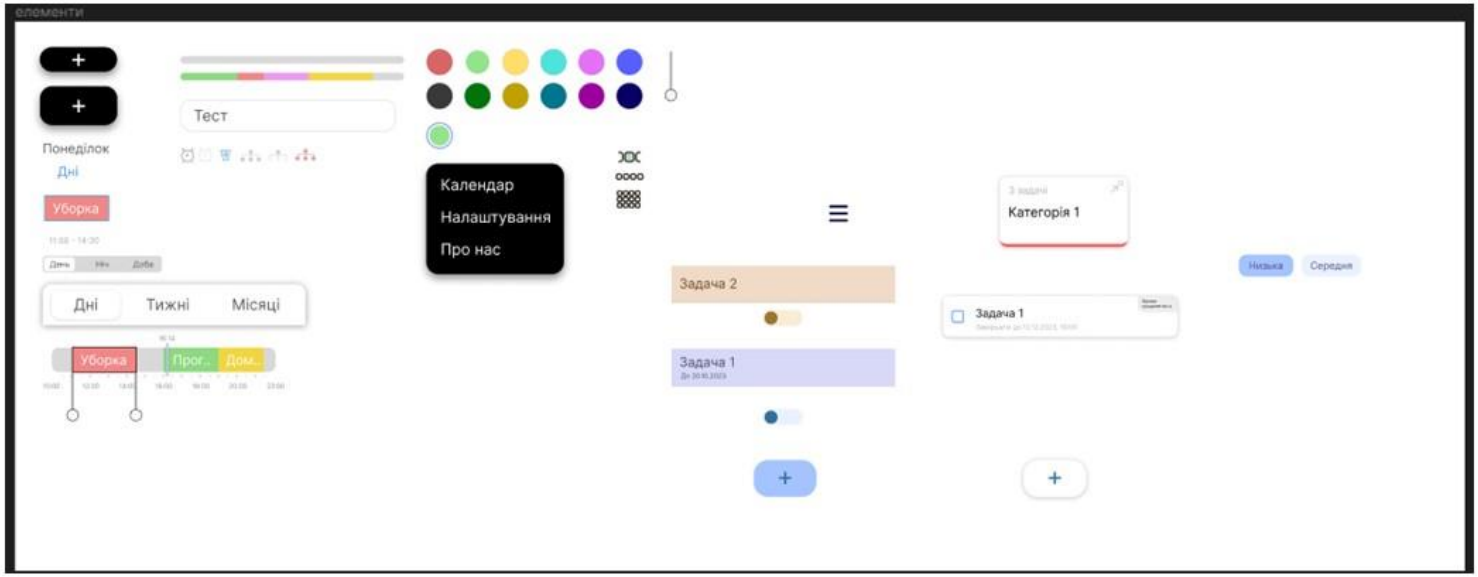
ІНСТРУМЕНТ ДЛЯ РОЗРОБКИ ДИЗАЙНУ ПРОГРАМИ FIGMA

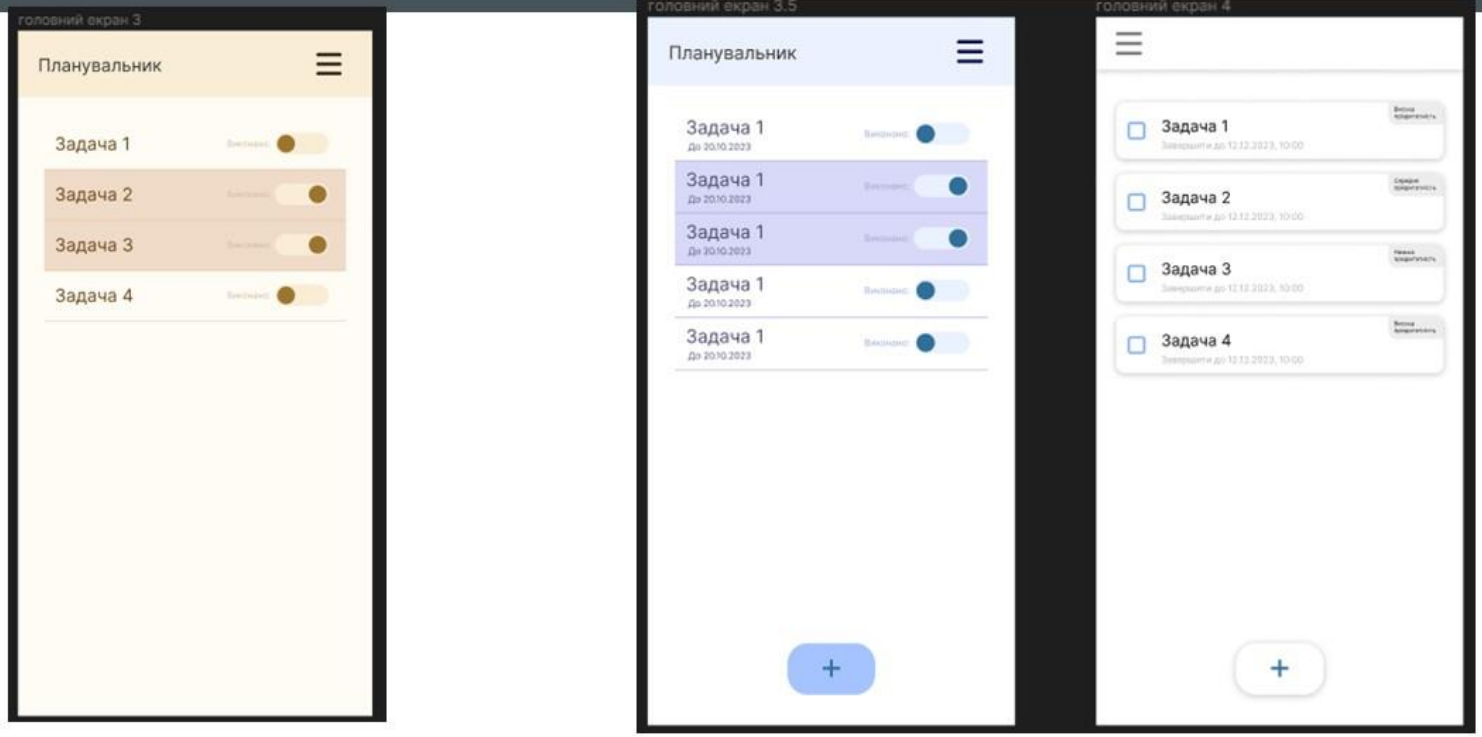


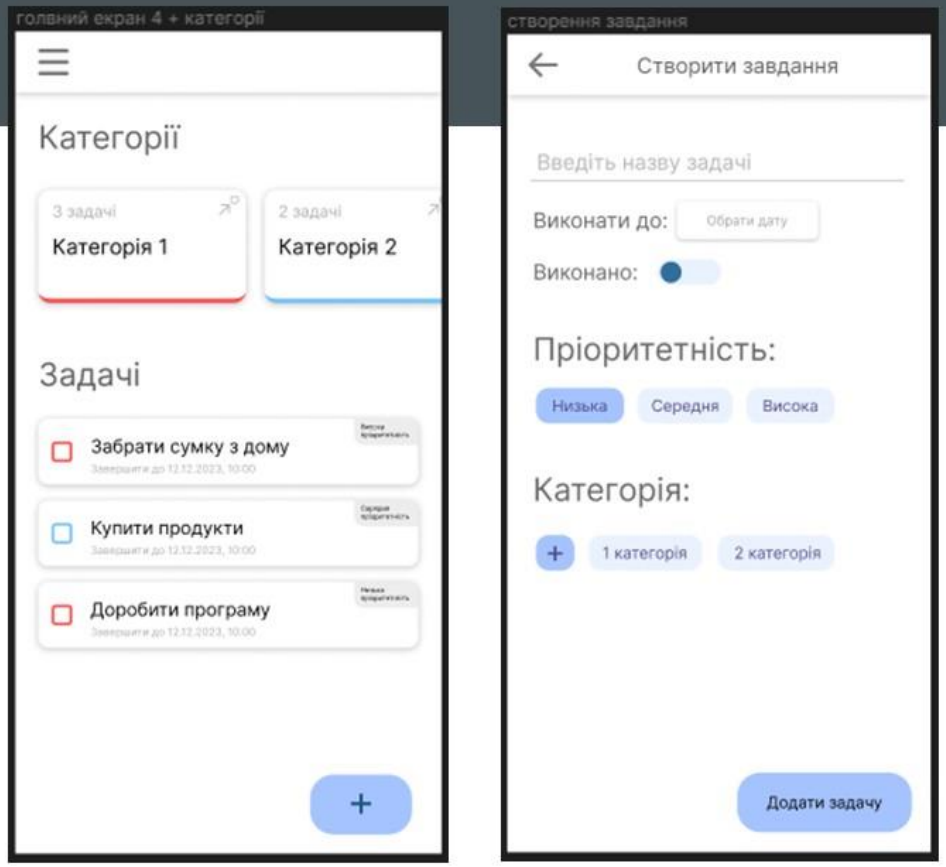
СТАДІЇ РОЗРОБКИ ДИЗАЙНУ



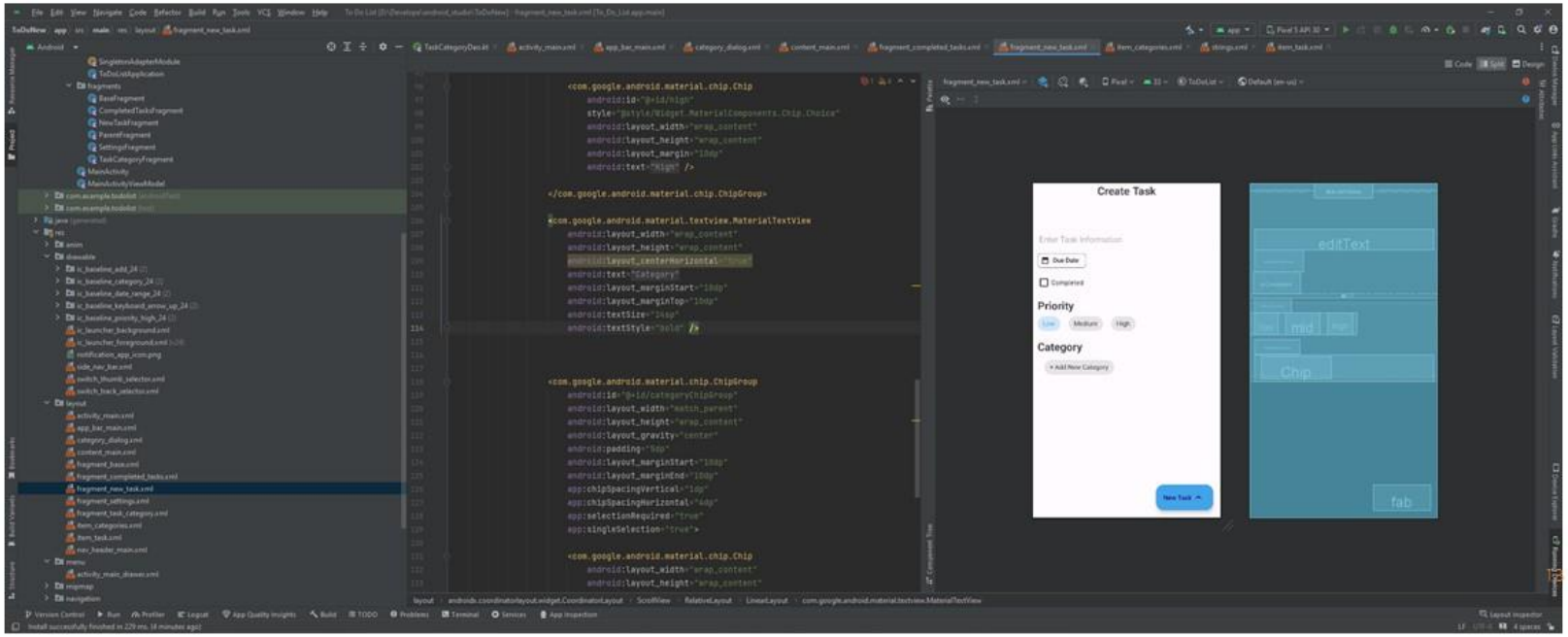
9


Елементи
Тест
Понеділок
Дні
Уборка
Календар
Налаштування
Про нас
Дні
Тижні
Місяці
Задача 2
Задача 1
Категорія 1
Задача 1
Низька
Середня

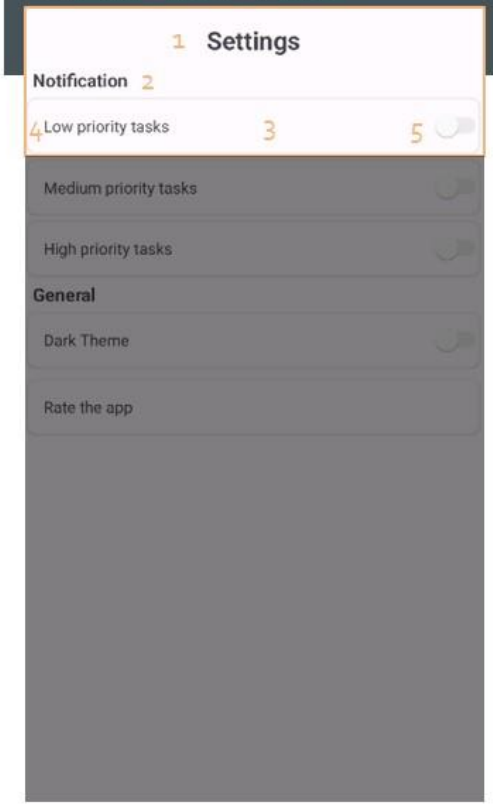

головний екран 3
Планувальник
Задача 1
Задача 2
Задача 3
Задача 4
головний екран 3.5
Планувальник
Задача 1
До 20.10.2023
Задача 1
До 20.10.2023
Задача 1
До 20.10.2023
Задача 1
До 20.10.2023
Задача 1
До 20.10.2023
головний екран 4
Задача 1
Задача 2
Задача 3
Задача 4


головний екран 4 + категорії
Категорії
3 задачі
Категорія 1
2 задачі
Категорія 2
Задачі
Забрати сумку з дому
Купити продукти
Доробити програму
створення завдання
Створити завдання
Введіть назву задачі
Виконати до:
Обрати дату
Виконано:
Пріоритетність:
Низька
Середня
Висока
Категорія:
1 категорія
2 категорія
Додати задачу

СЕРЕДОВИЩЕ РОЗМІТКИ ЗОВНІШНЬОГО ВИГЛЯДУ ПРОГРАМИ В ANDROID STUDIO



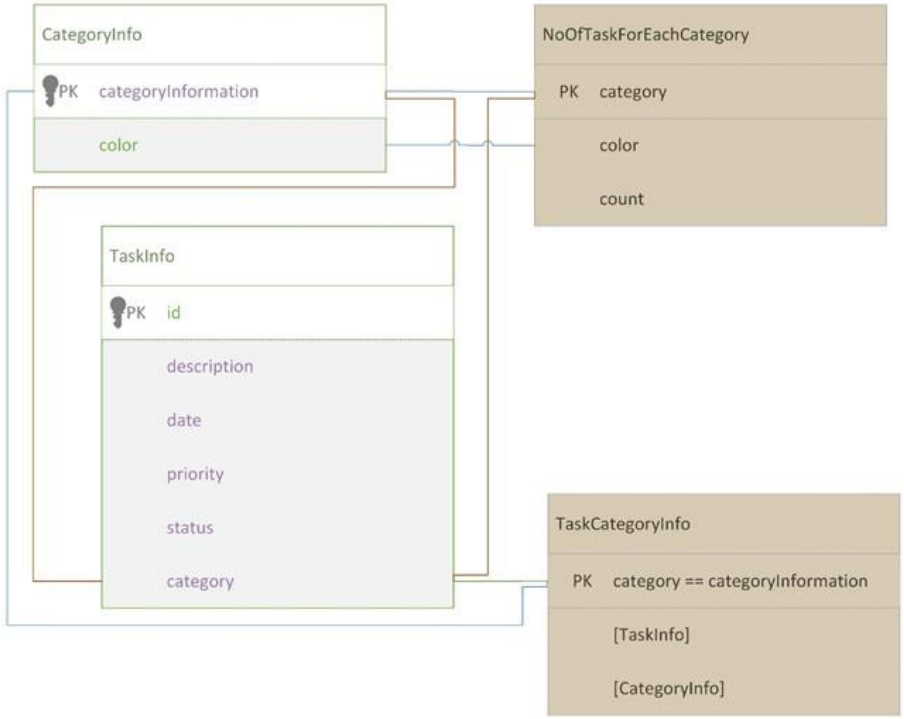
ПРИКЛАД РОЗМІТКИ НА ЕКРАНІ НАЛАШТУВАНЬ




```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            tools:context=".presentation.fragments.SettingsFragment">

            1 <com.google.android.material.textview.MaterialTextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginTop="15dp"
                android:layout_marginBottom="10dp"
                android:text="@string/settings"
                android:textSize="24sp"
                android:textStyle="bold" />

            2 <com.google.android.material.textview.MaterialTextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:text="@string/notification"
                android:textSize="16sp"
                android:textStyle="bold" />

            3 <com.google.android.material.card.MaterialCardView
                style="@style/Widget.App.CardView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:elevation="8dp"
                app:strokeColor="@color/light_grey"
                app:strokeWidth="1dp">

                4 <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:padding="18dp"
                    android:text="@string/low_priority_tasks" />

                5 <com.google.android.material.switchmaterial.SwitchMaterial
                    android:id="@+id/low_tasks"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="end"
                    app:thumbTint="@drawable/switch_thumb_selector"
                    app:trackTint="@drawable/switch_track_selector"/>

            </com.google.android.material.card.MaterialCardView>
```

android:id →

СТРУКТУРА БАЗИ ДАНИХ



ПРИКЛАД СТВОРЕННЯ МЕТОДУ DAO В ROOM DATABASE

```
    @Query("SELECT * " +
        "FROM taskInfo " +
        "WHERE status = 1 " +
        "ORDER BY date")
fun getCompletedTask():
LiveData<List<TaskCategoryInfo>>
```

```
    @Delete
suspend fun deleteCategory(categoryInfo: CategoryInfo)
```

ВИГЛЯД КОНВЕРТОВАНОЇ ФУНКЦІЇ DELETECATEGORY()

```
      Override
public Object deleteCategory(final CategoryInfo categoryInfo,
    final Continuation<? super Unit> continuation) {
  return CoroutinesRoom.execute(__db, true, new Callable<Unit>() {
    @Override
    public Unit call() throws Exception {
      __db.beginTransaction();
      try {
        __deletionAdapterOfCategoryInfo.handle(categoryInfo);
        __db.setTransactionSuccessful();
        return Unit.INSTANCE;
      } finally {
        __db.endTransaction();
      }
    }
  }, continuation);
}
```

ВИСНОВКИ

- Аналіз систем планування, розгляд додатків та вибір API стали кроком до нової розробки. Дослідження та аналіз аналогів підкреслили важливість розробки. Перелік аспектів для розробки мобільного планувальника охоплює функціональність та зручності. Вибір методів, таких як Figma для дизайну, SQLite та Room Database для бази даних, базується на вартості та функціональності. Робота з дизайном через Figma забезпечує ефективність. Вибір SQLite та Room Database для бази даних обумовлений офіційним статусом та ефективністю управління завданнями. Детальний аналіз в Android Studio підкреслив важливість роботи з розміткою та вибір оптимальних інструментів. Робота з базою даних через SQLite та Room Database спростила організацію та взаємодію. Вибір інструментів відображає важливість етапів розробки для створення функціональних та зручних додатків.
- Загалом, розроблена модель додатку повинна виправдати очікування та стати новим компаньйоном для користувача.