МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО КРОСПЛАТФОРМНОГО ЗАСТОСУНКУ «ПЛАНЕР» »**

Виконав: студент 2 курсу, групи 8.1228
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Д.О.Кубрак
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " травня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кубраку Данііlу Олеговичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка мобільного кросплатформного застосунку «Планер»

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 29.11.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3.Програмна реалізація системи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Програмна реалізація системи
4. Розробка структур окремих підсистем

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	05.05..23	
2.	Збір вихідних даних.	10.05.2023	
3.	Обробка методичних та теоретичних	05.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	15.07.2023	
5.	Розробка третього розділу.	19.08.2023	
6.	Оформлення та нормоконтроль	09.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	13.12.2023	

Студент _____ Д.О.Кубрак
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В.Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка мобільного кросплатформного застосунку «Планер»»: Обсяг 90 сторінок. Містить 35 рисунки, 1 додаток та 15 бібліографічних посилань.

ІНТЕРФЕЙС, ПЛАНЕР, МОБІЛЬНИЙ ДОДАТОК, FLUTTER, MVC

Об’єкт дослідження – мобільні кросплатформні застосунки

Мета роботи: Мета цієї роботи полягає в створенні мобільного додатку, що включатиме необхідні інструменти для оптимізації планування та організації особистої діяльності користувачів.

Використаний інструментарій: Flutter – фреймворк для розробки мобільних додатків, який може бути використаний для створення інтерфейсу мобільного додатку; MVC (Model-View-Controller) – архітектурний підхід, який розділяє логіку програми на три складові: модель (дані), представлення (інтерфейс) та контролер (логіка додатку). MVC може бути використаний для організації коду мобільного додатку для його кращого управління та підтримки.

Метод дослідження – опис, аналіз, синтез, пояснення .

В ході роботи було досліджено мобільні застосунки, які містять необхідні інструменти для поліпшення планування та організації діяльності. В роботі запропоновано мобільний застосунок – «Планер» для організації та планування діяльності користувача.

Розроблений мобільний застосунок буде корисним для здобувачів освіти під час організації навчальної діяльності, а також для всіх зацікавлених користувачів.

SUMMARY

Master's qualification work "Development of the mobile cross-platform application "Planer": Volume of 90 pages. Contains 35 figures, 1 appendix and 15 bibliographic references.

INTERFACE, PLANNER, MOBILE APP, FLUTTER, MVC

The object of the research is mobile cross-platform applications

Purpose of work: The purpose of this work is to create a mobile application that will include the necessary tools to optimize the planning and organization of personal activities of users.

What tools were used:

Flutter: This is a mobile application development framework that can be used to create a mobile application interface.

MVC (Model-View-Controller): This is an architectural approach that divides the application logic into three components - the model (data), the view (interface) and the controller (application logic). MVC can be used to organize the code of a mobile application for better management and support.

Research method – description, analysis, synthesis, explanation.

In the course of the work, mobile applications were investigated, which contained the necessary tools for improving the planning and organization of activities.

A certain number of mobile applications and examples of work with gliders of various simulations were investigated.

ЗМІСТ

ВСТУП

Кожна людина повсякденному житті, під час середнього пережитку, стикається з питаннями чи проблемами, на які важко знайти відповідь. Наприклад, студент-першокурсник приїжджає в Студмістечко в карантинну реальність, толком не знаючи, де знаходиться корпус і де самі гуртожитки. Студенти, які тривалий час навчаються, знають, що система електронного щоденника не працює належним чином, а також дещо бракує мобільної версії, через що неможливо легко перевірити свої результати, перебуваючи поза комп’ютером. Варто також відзначити, що не завжди раціонально заходити на різні сайти для пошуку інформації, тому що швидкість Інтернету іноді може забирати багато дорогоцінного часу.

Для пошуку та отримання необхідної інформації людина витрачає багато ресурсів, тому що у нього немає такої можливості знайти все в одному місці та через зручний інтерфейс. Для вирішення аналогічних проблем потрібно розробити систему, яка вміщатиме основний функціонал. Наприклад, вона буде включати графік розкладу, довідник для вирішення питань і планер, що допомагатиме учням не забувати про важливі справи та не перенавантажувати свою пам'ять. Для реалізації цього проекту потрібно виконати такі умови:

Сьогодні ця тема є дуже актуальною та набуває популярності серед громадськості внаслідок стрімкого технологічного прогресу та розвитку інтернет-індустрії. Система обліку оцінок, презентація новин та публікація розкладу потребує поліпшень, як мінімум у формі оновлення.

З урахуванням сучасних реалій та тенденцій, такі розробки мають сенс і стають популярними, якщо вони належним чином реалізовані, мають сервісну підтримку та своєчасні оновлювання. Це залежатиме не лише від розробників, але й від команди програмістів, які співпрацюють з адміністрацією навчального закладу. Без їхньої допомоги буде складно

отримати доступ до даних або реалізувати систему максимально безпечною та ефективною.

Мета роботи: створення мобільного додатку, що включатиме необхідні функціональні можливості для оптимізації процесів планування та організації особистої діяльності користувача.

В ході роботи були використані такі інструменти: Flutter – фреймворк для розробки мобільних додатків; MVC (Model-View-Controller) – архітектурний підхід,

В ході роботи було досліджено мобільні застосунки, які містили необхідні інструменти для поліпшення планування та організації діяльності.

1 ПОСТАНОВКА ЗАДАЧІ

Мета проекту – розробити мобільний додаток, який міститиме інструменти для покращення планування та організації діяльності. Для досягнення цієї мети потрібно вирішити наступні завдання:

а) провести огляд існуючих аналогів та проаналізувати їхні переваги та недоліки;

б) дослідити наявні рішення для записів, нотаток та нагадувань. Описати переваги та недоліки різних програмних продуктів і обґрунтувати потребу в створенні нового додатку;

в) вибрати засоби для розробки, такі як мова програмування та інші інструменти. Обґрунтувати вибір і навести альтернативи;

г) розробити архітектурні рішення для структури проекту та описати інтерфейс користувача з мобільним за стосунком;

д) тестувати мобільний додаток та надати приклади використання для демонстрації можливостей застосунку;

е) надати опис інтерфейсу користувача, зокрема календаря-планера. Система повинна бути легкою у користуванні та мати зручний інтерфейс;

Додаток повинен бути мобільним, зручним для користувача та мати можливість перемикання між різними його функціями, такими як планування та калькулятор.

Життя нині важко уявити без цифрових інструментів. З урахуванням сучасних реалій, користування цифровими технологіями стає нормою. Навіть під час світової пандемії COVID-19, що призвела до переходу до онлайн-навчання та роботи через Інтернет, цифрові технології стали невід'ємною частиною нашого життя.

Наразі існує багато додатків та програм, але жоден не є універсальним. Це й визначило ідею дипломного проекту – створення мобільного додатку

планера. Проект включає аналіз предметної області, створення бази даних, зручний інтерфейс користувача та різноманітні можливості.

Мета полягає у створенні мобільного додатку на платформі Android з використанням мови програмування Dart. Завдання включає розробку інтерфейсу, моделювання системи та синхронізацію з базою даних. Під час розробки мобільного додатку акцент зроблено на застосування його здобувачами освіти для покращення організації навчальної діяльності.

2 АНАЛІЗ МОБІЛЬНИХ ПЛАНЕРІВ-ЩОДЕННИКІВ

Навчальний процес є ключовим у закладі освіти, оскільки через нього реалізується виконання навчального плану. Цей процес включає різні елементи та об’єкти, без яких він став би неможливим. Навчальний план визначає розподіл залікових кредитів між різними предметами, розклад навчального процесу та план навчання на кожен семестр. Цей план визначає перелік та обсяг вивчення навчальних предметів, форми проведення навчальних занять та їх обсяг, а також види поточного та підсумкового контролю, а також процедури державної атестації.

2.1 Системи організації навчальної діяльності

Існує велика кількість програм для університетів, які відображають розклад, ведуть електронний щоденник, надають новини та створюють нотатки. Кожна з них має свої переваги і недоліки.

Сайт moodle.znu.edu.ua використовується для відображення розкладу пар та сесій для всіх груп та викладачів в ЗНУ. Це відкрита система, тому інформація не є конфіденційною. Однак, відкритий доступ може бути не найкращим рішенням.

Крім того, у ЗНУ існує платформа "Moodle" − це електронний кабінет. Вона має наступні переваги:

- вимагає авторизації для входу;
- надає інформацію про людину;
- містить останні новини;
- має щоденник людини;
- надає інформацію про заліки, екзамени тощо.

Проте, незважаючи на потрібний функціонал, є деякі недоліки:

- неотримання очікуваних оновлень (застарілий інтерфейс);

- неусувані труднощі у співпраці викладачів з системою;

- відсутність мобільної версії чи додатку.

Це всього лише декілька прикладів популярних програм університетів, кожна з яких має свої плюси та мінуси.

На рисунку 2.1 можна оцінити інтерфейс веб-сторінки:

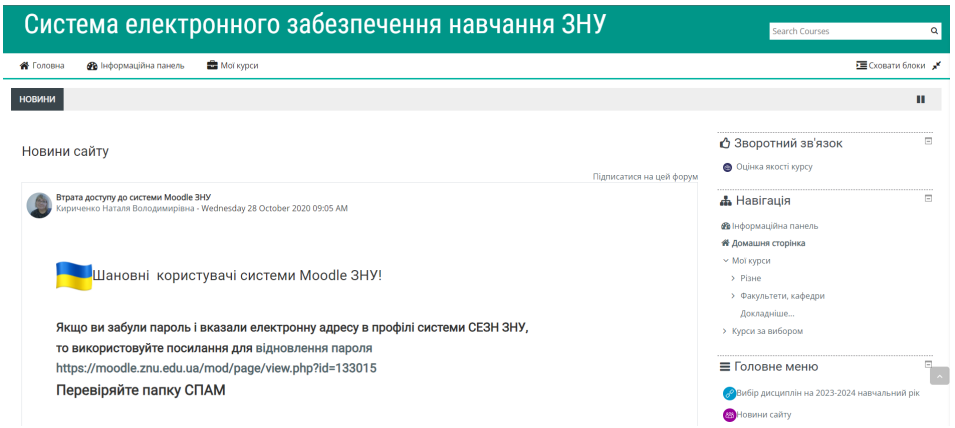


Рисунок 2.1 – СЕЗН Moodle

Blackboard Learn

Це додаток для інтерактивного викладання, навчання, створення спільнот та обміну знаннями. Використовуючи Blackboard Learn можно розробляти будь-яку теорію або модель для свого онлайн-курсу, оскільки – це відкрита та гнучка система, націлена на покращення успішності учнів.

2.1.1 Класифікація додатків для мобільного пристрою

Мобільні додатки класифікуються за певним призначенням:

а) корпоративні мобільні додатки:

Їх призначення – спрощення роботи в компанії та швидка передача корпоративних даних серед працівників. Ці додатки адресовані спочатку працівникам компанії, а також реальним та потенційним клієнтам і партнерам. Вони мають більш спрощений процес розробки та

використовують кольори та елементи корпоративного стилю компанії для дизайну.

б) контентні мобільні додатки:

Основна мета − надання різноманітної інформації у текстовому, відео чи аудіо форматі. Ці додатки часто призначені для засобів масової інформації, радіо, телевізійних каналів чи порталів.

в) сервісні мобільні додатки:

Їх мета − надання певних сервісів, виконання завдань у реальному часі, відповідно до запитів користувачів. Ці додатки більш складні в розробці, оскільки повинні працювати надійно, від калькулятора до програм для роботи з великими обсягами тексту чи графіки.

г) ігрові мобільні додатки:

Основне завдання − забезпечення розваг, хоча це не єдине їх призначення. Ці додатки ускладнюються необхідністю вдало впровадити контекстну чи пряму рекламу.

Окрім цього, важливо класифікувати їх за структурою:

а) вебдодатки: Мобільна версія сайту з розширеним інтерактивом, що працює через браузер. Вони є недорогими та швидко впроваджуваними.

б) гібридні додатки або генератори мобільних додатків: Між нативними та вебдодатками, встановлюються через офіційні магазини, мають обмежений доступ до апаратних можливостей.

в) Нативні додатки: Написані під конкретну платформу, вбудовані в операційну систему, швидкі та ефективні, але вимагають більше часу для розробки та високі витрати.

2.1.2 Інтерфейс мобільних додатків

У роботі з аналізу та вибору оптимального дизайну для інтерфейсу користувача, було відібрано кілька прикладів робіт для подальшого аналізу їх

переваг та недоліків з метою поліпшення зовнішнього вигляду та функціональності програмного забезпечення.

При оцінці дизайну було звернуто увагу на сприйняття та зручність інтерфейсу, оскільки це один з ключових аспектів розробки. Також проведено порівняння з існуючими мобільними додатками, їхнім інтерфейсом та враховано переваги та недоліки в порівнянні з іншими розробками (Рисунок 2.2 − Рисунок 2.4).

Зазначений дизайн інтерфейсу виглядає досить зручним для операцій реєстрації та авторизації користувача. Він відзначається відсутністю зайвих деталей та повністю задовольняє поставлені вимоги щодо стилістики. Також важливою перевагою є можливість входу через облікові записи на платформах Facebook та Google.

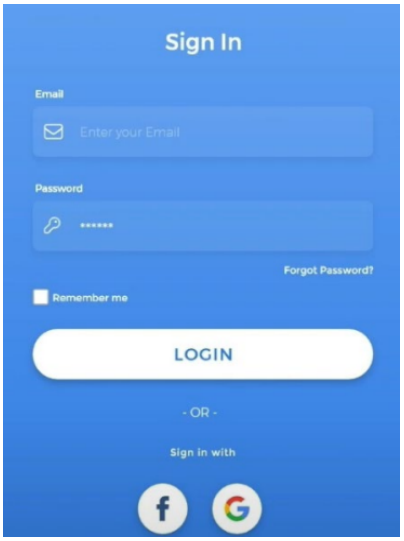
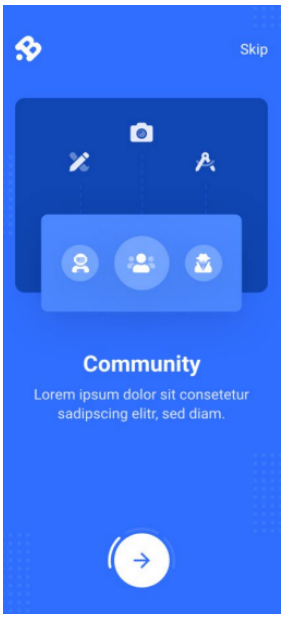


Рисунок 2.2 − UI для авторизації користувача



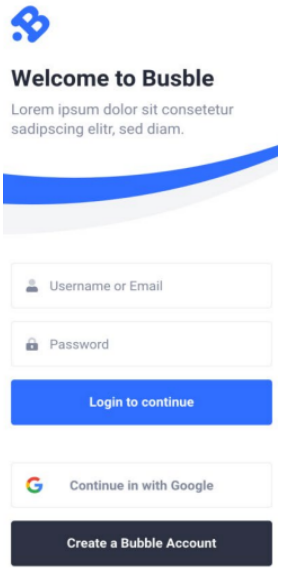


Рисунок 2.3 − UI приклад авторизації Рисунок 2.4 − UI глибока авторизація

Наразі основною метою є поліпшення структури для більшої функціональності та зручності.

Судячи з вказаного опису, можна підкреслити, що основним елементом повинні бути поля для введення даних та кнопка підтвердження. При цьому, заголовок та підзаголовок можуть займати забагато місця та виглядати непотрібно на даній сторінці, несучи лише вторинну інформацію. Також, кнопки для авторизації через соціальні мережі можуть бути розміщені менш вигідно, якщо вони займають багато місця та виглядають неестетично для користувача.

Вирішення цих питань може полягати у оптимізації простору на сторінці, зменшенні кількості вторинних елементів та підвищенні зручності для користувача.

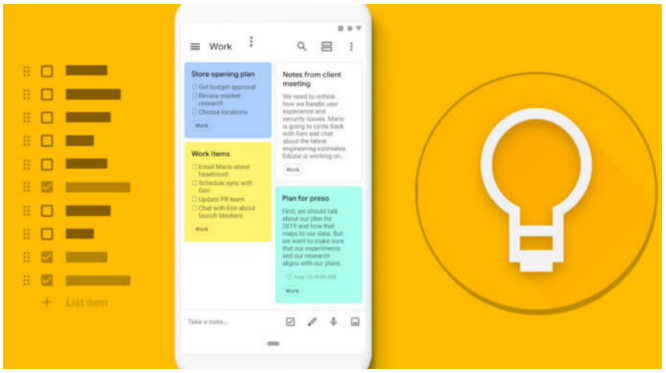


Рисунок 2.4 – Приклад загальної авторизації

2.1.4 Популярні сервіси

Популярність сервісів планування та записів, таких як наприклад Samsung Notes, можна пояснити рядом факторів. Ці сервіси стали невід'ємною частиною повсякденного життя через свою зручність та багатий функціонал.

Користувачі цінують такі сервіси за їхню універсальність, здатність швидко внести записи в будь-якому вигляді, зручний доступ до них з різних пристроїв та можливість організації матеріалу згідно з особистими потребами. Такі сервіси стали не лише інструментом планування, а й зручним способом ведення щоденних записів, вивчення, створення заміток та відстеження різних проектів.

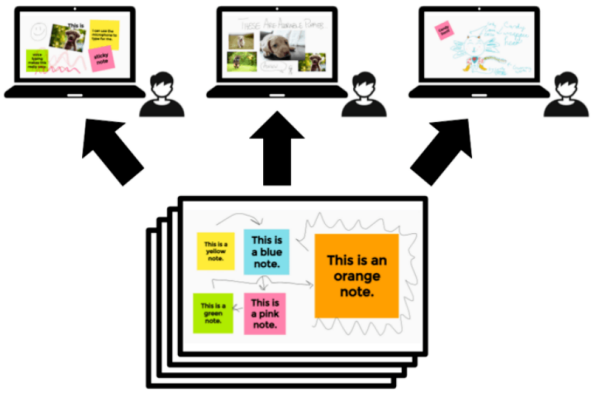


Рисунок 2.5 – Особливість нотатків в популярних сервісах

Це чудово, що популярні сервіси надають швидкий пошук, що полегшують знаходження необхідних нотаток. Опція сортування за змістом або спільним доступом додатково сприяє легкості у використанні.

Функція закріплення нотаток за календарем для створення нагадувань є дуже зручною. Це не лише спрощує організацію справ, але й додає безпеки, особливо у випадку конфіденційних даних, які можуть бути в нотатках.

В порівнянні з інтерфейсом Samsung Notes, розробляємий мною додаток виглядає більш привабливим та більш функціональним. Він надає зручні функції, такі як прив'язка до календаря і можливість додавати замітки у формі малюнків або шляхом вставлення зображень та фотографій, що є надзвичайно корисним для користувача.

Додаток Samsung Notes також надає можливість створювати різноманітні замітки: писати, малювати, робити позначки різними стилів та використовувати різні кольорові підкреслення за допомогою S Pen. Можна додавати фотографії чи голосові нотатки, а також маркувати нотатки за допомогою пошукових тегів.

Цей додаток дає можливість одночасно переглядати відео та робити нотатки. Просто натисніть кнопку на пері S Pen та двічі торкніться екрана, щоб відкрити спливаюче вікно блокнота. Ви можете налаштувати його розмір і прозорість так, як зручно користувачу, і матимете зручний робочий інструмент для занотовування.

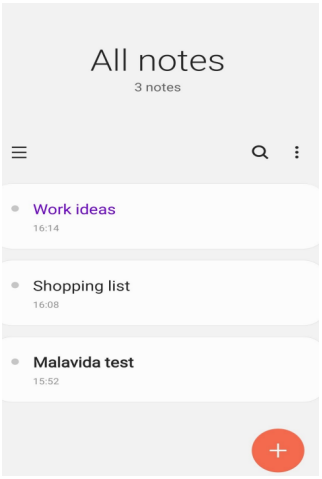


Рисунок 2.6 − Приклад нотатків

Підсумковий висновок щодо цього додатку – ідеї цього додатку варто використати у запропонованій роботі, оскільки це один із найкращих прикладів дизайну інтерфейсу.

3 ЗАСОБИ РОЗРОБКИ

Для розробки додатку було використано такі засоби:

– Android Studio;

– Dart − мова програмування;

– Sqlite та Firestore − СУБД;

– MVC (технологія, значення якої розкриється потім).

3.1 Середовище для розробки

Було спроектовано та розроблено мобільний додаток, що надає можливість перегляду навчальної інформації, створення нотаток та корегування інформації для налаштування під потреби користувача.

Android Studio (Рисунок 3.1) прийшло на заміну плагіну ADT для платформи Eclipse. Це середовище побудоване на основі вихідного коду продукту IntelliJ IDEA Community Edition, що розробляється компанією JetBrains. Це середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки додатків для платформи Android. Воно включає засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування додатків, що працюють на пристроях з різною роздільністю екранів (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, які присутні в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування та розгортання додатків, заснована на складальному інструментарії Gradle і підтримка засобів безперервної інтеграції.

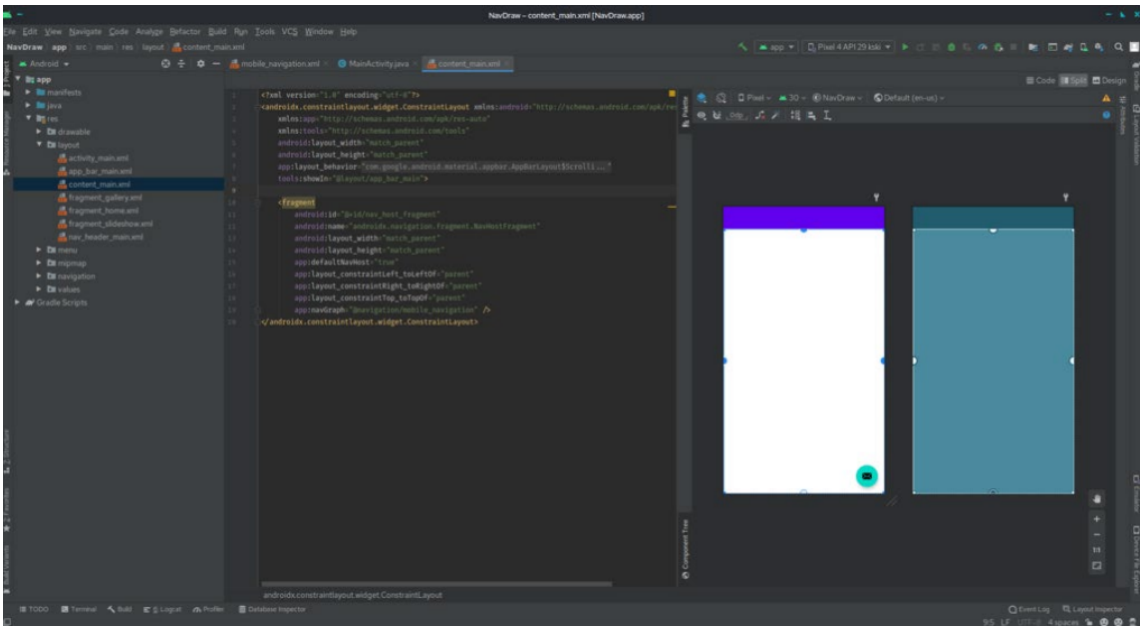


Рисунок 3.1 −Інтерфейс Android Studio

Середовище Android Studio має такі нові можливості та оновлення:

- живі макети (layout): редактор WYSIWYG − живе кодування — дозволяє відображати (render) програму в реальному часі;

- консоль розробника: надає підказки по оптимізації, допомагає з перекладом, відстежує напрямок, агітації та акції − включає метрики Google аналітики;

- бета-релізи та покрокові виходи;

- засноване на Gradle.

3.2 Мова програмування

Обрання мови Flutter для розробки мобільного застосунку було обумовлено її контролем компанією Google, перспективністю та швидким розвитком. Навіть при наявності інших конкуруючих мов, таких як React Native, Flutter є привабливим вибором, оскільки використовує мову програмування – Dart.

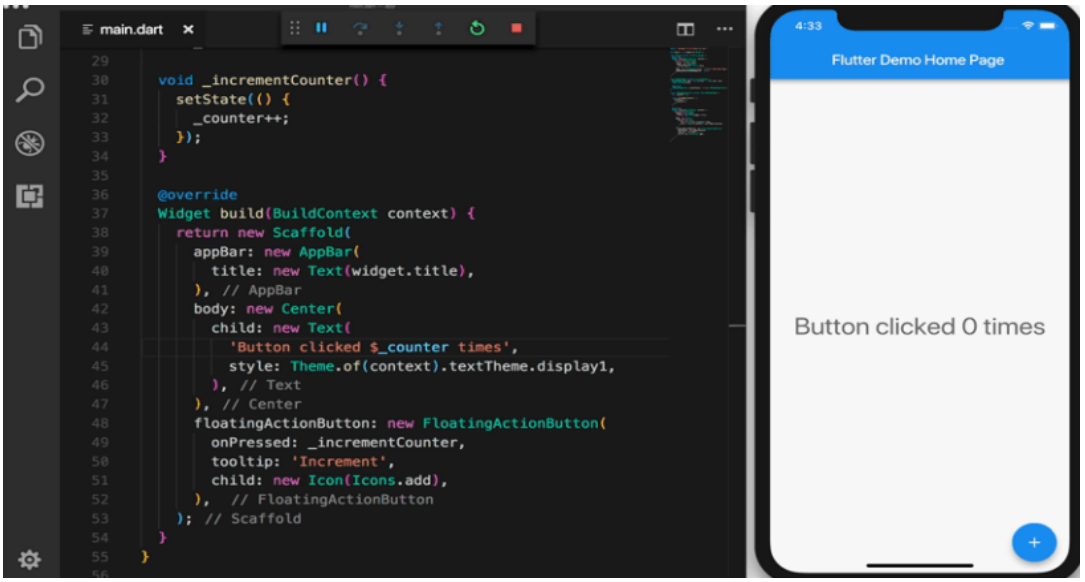


Рисунок 3.2 −Приклад BuildContext

Dart, розроблений компанією Google, є мовою програмування, яку позиціонують як структурну мову для веброзробки. Ця мова призначена для довгострокового використання та має потенційно стати альтернативою JavaScript. JavaScript стикається з певними проблемами у сучасній розробці, такими як обмеженість розширюваності, продуктивності та підтримки складних додатків. Dart має синтаксис, схожий на Java, не вимагає явного визначення типів та є придатною як для створення серверних, так і для клієнтських додатків.

3.3 СУБД

У додатку для збереження даних користувачів та планування використовуються дві бази даних: одна для зберігання логінів та паролів, інша для планування. Можна об'єднати ці дві бази, але для розробки вибрано використання обох одночасно.

SQLite − це легка система управління реляційними базами даних, реалізована у вигляді бібліотеки, яка має багато функцій стандарту SQL-92. SQLite поширюється як суспільне надбання та доступний для використання

без обмежень та безкоштовно для будь-яких цілей. Фінансову підтримку розробки SQLite забезпечує спеціальний консорціум, до якого входять компанії, такі як Adobe, Oracle, Mozilla, Nokia, Bentley та Bloomberg. З 2018 року SQLite, разом з JSON та CSV, є рекомендованим форматом зберігання структурованих даних для Бібліотеки Конгресу США.

Firebase – це платформа для розробки мобільних та веб-застосунків. Компанія Firebase Inc. розробляла цю платформу з 2011 року, а в 2014 році її придбала компанія Google:

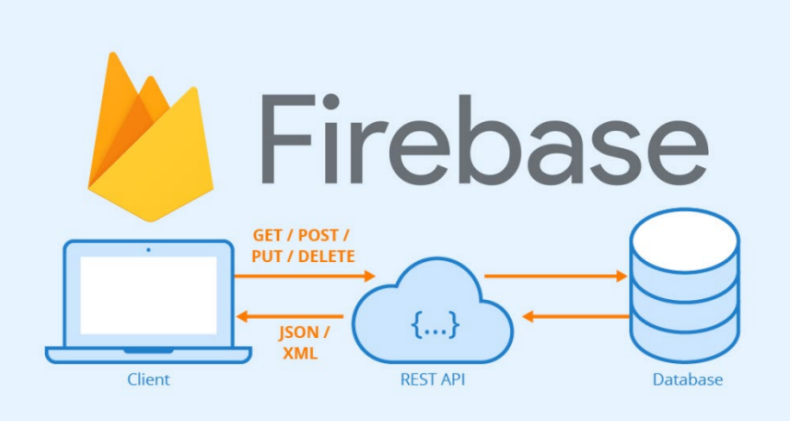


Рисунок 3.3 – Схема БД Firebase

В рамках платформи Firebase виокремлюються кілька ключових компонентів, кожен з яких виконує свою важливу роль:

а) Firebase Analytics: Безкоштовний інструмент для оцінки та моніторингу застосунків, який дозволяє розробникам отримувати відомості про використання своїх застосунків і залучення користувачів.

б) Firebase Cloud Messaging (FCM): Крос-платформний інструмент для надсилання повідомлень і сповіщень на Android, iOS та веб-застосунки, який надається безкоштовно.

в) Firebase Auth: Служба аутентифікації користувачів, яка підтримує соціальні логін-провайдери, такі як Facebook, GitHub, Twitter та Google. Також містить систему управління користувачами та підтримує аутентифікацію за електронною поштою та паролем.

г) Realtime Database: База даних, яка працює в режимі реального часу та забезпечує розробникам API для синхронізації даних між клієнтами та зберігання їх в хмарі Firebase.

д) Firebase Storage: Сервіс для надійного завантаження та вивантаження різних файлів для застосунків Firebase, який підтримує зберігання зображень, аудіо-, відео- та іншого контенту, створеного користувачами. Firebase Storage використовує Google Cloud Storage.

Що стосується порівняння із першою СУБД SQLite, база даних Cloud Firestore виглядає більш розгалуженою та функціональною. Однак, на даному етапі, для проекту вибрано використання саме SQLite. Це рішення може бути переглянуте в майбутньому, особливо після отримання API від певних ресурсів. Firebase надає більше можливостей та широкий набір інструментів, які можуть стати важливими для подальшого розвитку проекту.

3.4 Контроль версій набору утиліт Git

Git являє собою набір утиліт командного рядка, що працюють з параметрами. Всі налаштування зберігаються у текстових файлах конфігурації. Це рішення робить Git таким, що легко переноситься на будь-яку платформу та спрощує його інтеграцію з іншими системами (включаючи можливість створення графічних git-клієнтів з різними інтерфейсами).

Репозиторій Git представляє собою каталог файлової системи, що містить файли конфігурації сховища, файли журналів, які зберігають операції, виконані над репозиторієм, індекс, що описує розташування файлів, та сховище, де фактично зберігаються файли. Структура файлів у сховищі не відображає реальної структури, що міститься в файловому дереві репозиторію (див. рис.). Це орієнтоване на підвищення швидкості виконання операцій з репозиторієм.

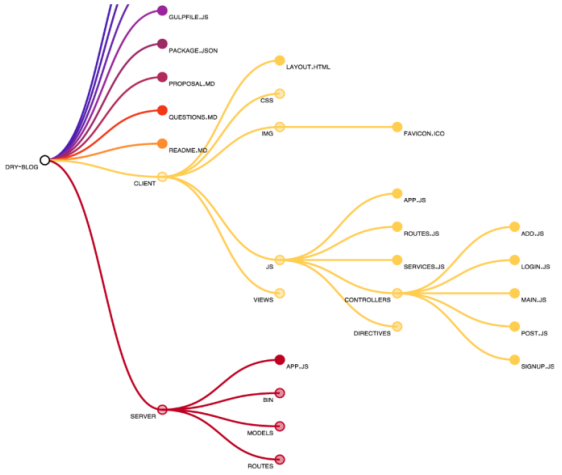


Рисунок 3.4 − Інтерфейс Android Studio

Під час виконання команд на зміни у системі контролю версій (наприклад, локальні зміни або отримані патчі від іншого вузла), ядро створює нові файли в сховищі, які відображають нові стани вже змінених файлів. Важливо зауважити, що жодні операції не вносять змін до вже існуючих файлів у сховищі. Зазвичай репозиторій Git зберігається в підкаталозі з назвою ".git" у кореневому каталозі робочої копії файлової системи, що зберігається в репозиторії. Будь-яке дерево файлів у системі може бути перетворено в репозиторій Git, використовуючи команду створення сховища з кореневого каталогу цього дерева (або вказавши кореневий каталог як параметр програми).

Репозиторій можна імпортувати з іншого вузла, доступного через мережу. Під час імпорту нового сховища автоматично створюється робоча копія, яка відображає останній зафіксований стан імпортованого сховища (тобто, зміни в робочій копії вихідного вузла не копіюються, якщо на цьому вузлі не було виконано команду commit).

3.5 Технологія MVC

MVC є архітектурним шаблоном, використовуваним у проектуванні та розробці програмного забезпечення. Його мета полягає в створенні гнучкого дизайну програми, який полегшує розробку та роботу з програмою, а також надає можливість повторного використання окремих компонентів програми. Цей шаблон спрямований на поділ системи на три ключові компоненти: модель даних, вигляд (інтерфейс користувача) та контролер.

Модель (Model) відповідає за зберігання даних та їхню структуру. Вона представляє інформацію та правила застосунку, незалежні від інтерфейсу користувача.

Вигляд (View) відображає дані користувачеві, тобто інтерфейс програми. Він представляє інформацію, що зберігається в моделі та взаємодіє з користувачем.

Контролер (Controller) виконує роль посередника між моделлю та виглядом, керуючи взаємодією між ними. Він отримує вхідні дані, перетворює їх на команди для моделі чи вигляду та відповідає за реакцію на дії користувача.

MVC дозволяє створити програму з високим рівнем узагальнення, забезпечуючи структуру, що розділяє логіку застосунку, його інтерфейс та ділить відповідальності між різними компонентами.

В рамках шаблону MVC, контролер функціонує як проміжний шар, що обробляє вхідні дані та взаємодіє як із моделлю, так і з виглядом. Він приймає вхідні дані від користувача чи з інших джерел, обробляє їх та ініціює відповідні дії у моделі чи вигляді.

Модель відповідає за обробку даних та забезпечення логіки програми. Вона інкапсулює основні дані та функції, що відповідають за їхню обробку. Модель не залежить від процесу вводу чи виводу даних, що дозволяє використовувати її в різних контекстах, навіть якщо методи виводу чи введення даних змінюються.

Вигляд відповідає за представлення інформації для користувача. Це можуть бути різні області, такі як таблиці, поля форм чи інші візуальні компоненти, які показують дані користувачеві. Вигляд відображає інформацію, що зберігається в моделі та дозволяє користувачу взаємодіяти з цими даними.

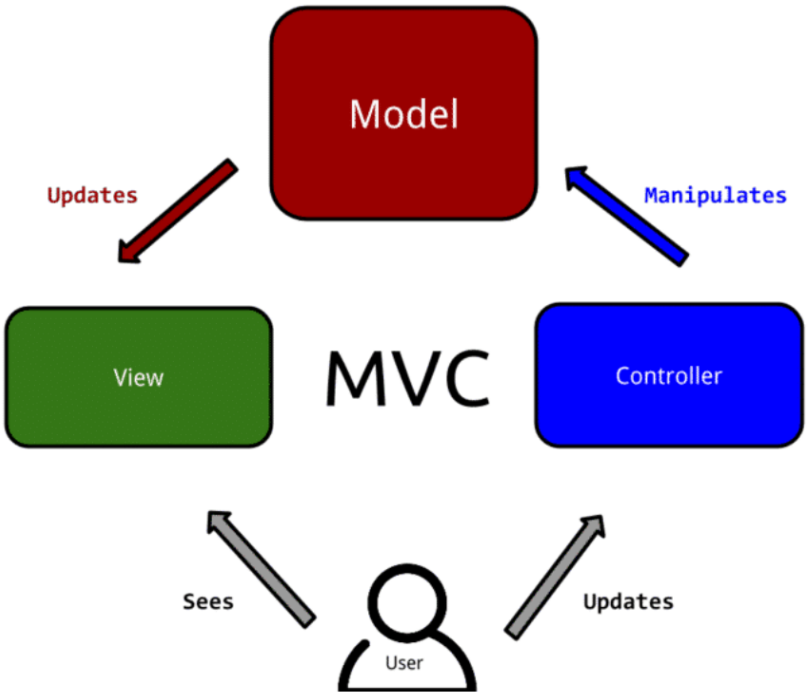


Рисунок 3.5 – Взаємодія користувача з даними

4. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

Згідно третього пункта, для розробки системи використовувались Android Studio та методологія MVC. У проекті використовувалась база даних sqlite, яку можна було замінити на більш розширену та широко використовувану платформу Firebase. Мобільний додаток було розроблено на мові Dart з використанням платформи Flutter в середовищі Android Studio.

4.1 Взаємодія із застосунком

Процес підтримки життєвого циклу додатку може розпочинати одна людина в простій конфігурації за замовчуванням. Проте з часом можуть бути внесені оновлення до додатку, розширена його функціональність і здійснена інтеграція з системою університету для реалізації працездатності системи авторизації за допомогою логінів та паролів кампусу. Проект має відкритий вихідний код, який було завантажено за допомогою інструменту контролю версій Git у новий репозиторій системи Github. Файли проекту можуть бути завантажені та використані, і для уникнення помилок при запуску мобільного додатку рекомендується зберігати зміни в окремих папках.

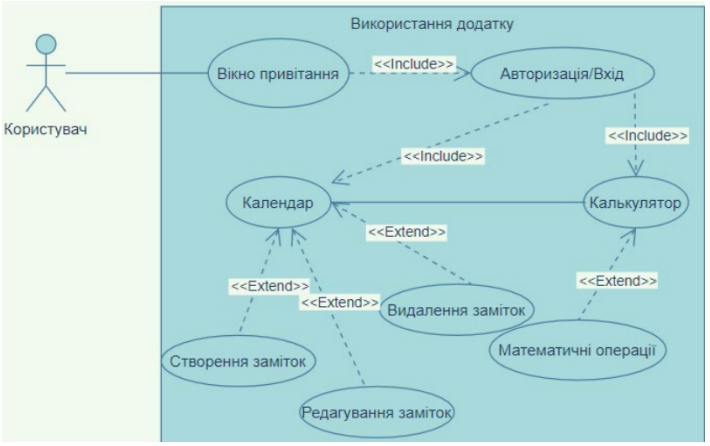


Рисунок 4.1 – Схема взаємодії користувача з додатком

4.2 MVC

В представленому проекті, для кращої візуалізації стилів, дизайну та працездатності, ми застосовуємо підхід, заснований на архітектурному шаблоні MVC (Model-View-Controller). Модель (Models) розташована в окремій папці. На рисунку 4.1 наведено приклад моделі для реалізації нейморфічного калькулятора.

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

const kBlackColorText = Colors.black;
const kOrangeColorText = Colors.deepOrange;
const kOperationTextStyle = TextStyle(
  fontSize: 30,
  color: Colors.blue,
  fontFamily: "MontserratLight ",
  fontWeight: FontWeight.w100,
);
const kResultTextStyle = TextStyle(
  fontSize: 70,
  color: Colors.orange,
  fontFamily: "MontserratLight ",
  fontWeight: FontWeight.w100,
);
```

Рисунок 4.1 – Модель нейроморфічного калькулятора

У представленому проекті створено необхідні контролери. Контролер отримує введення дані від користувача, обробляє їх і надсилає результат обробки, як правило, назад у вигляді представлення.

Використовуючи контролери, дотримуємось певних умов. Згідно з домовленостями про іменування, назви контролерів повинні закінчуватися на суфікс "Controller".

На рисунку 4.2 показано приклад одного з контролерів.

```dart
class _HomePageState extends State<HomePage> with T

  AnimationController _scaleController;
  AnimationController _scale2Controller;
  AnimationController _widthController;
  AnimationController _positionController;

  Animation<double> _scaleAnimation;
  Animation<double> _scale2Animation;
  Animation<double> _widthAnimation;
  Animation<double> _positionAnimation;
```

Рисунок 4.2 – Контролер

```dart
class Todo {
  int id;
  String title;
  String description;
  String category;
  String todoDate;
  int isFinished;

  todoMap() {
    var mapping = Map<String, dynamic>();
    mapping['id'] = id;
    mapping['title'] = title;
    mapping['description'] = description;
    mapping['category'] = category;
    mapping['todoDate'] = todoDate;
    mapping['isFinished'] = isFinished;
```

Рисунок 4.3 - Праця з моделлю даних

Останнім етапом проектування є створення екранів для відображення інформації, що означає створення інтерфейсу. На рисунку 4.4 можна спостерігати застосування контролерів. Для структури визначені візуальні стилі, як для основного вікна, так і для додаткових вікон (наприклад, видалення, редагування, оновлення, зміни тощо).

```
getAllCategories() async {
  _categoryList = List<Category>();
  var categories = await _categoryService.readCategories();
  categories.forEach((category) {
    setState(() {
      var categoryModel = Category();
      categoryModel.name = category['name'];
      categoryModel.description = category['description'];
      categoryModel.id = category['id'];
      _categoryList.add(categoryModel);
    });
  });
}

_editCategory(BuildContext context, categoryId) async {
  category = await _categoryService.readCategoryById(categoryId);
  setState(() {
    _editCategoryNameController.text = category[0]['name'] ?? 'No Name';
    _editCategoryDescriptionController.text =
        category[0]['description'] ?? 'No Description';
  });
  _editFormDialog(context);
}

_showFormDialog(BuildContext context) {
  return showDialog(
      context: context,
      barrierDismissible: true,
      builder: (param) {
        return AlertDialog(
          actions: <Widget>[
```

Рисунок 4.4 - Візуальні стилі різних вікон

Це типовий та ефективний підхід у розробці проекту, оскільки він дозволяє систематизувати роботу та сприяє створенню послідовного каркасу проекту. Застосування такого підходу сприяє оптимізації часу, витрат та ресурсів, що в свою чергу може позитивно позначитися на продуктивності в розробці програмного забезпечення.

4.3 Діаграма класів та архітектурна побудова

Діаграма класів є однією з основних діаграм, які використовуються в мові UML. Вона відображає структуру системи, яка включає в себе класи, їх

властивості, методи та зв'язки між ними. Класи відображаються у вигляді прямокутників, в яких зазвичай вказані назва класу, його атрибути та методи. Ця діаграма може містити зв'язки, такі як асоціації між класами, зв'язки узагальнення або залежності.

Асоціація відображається лінією між класами, що показує, як один клас пов'язаний з іншим. На цій лінії можуть бути вказані інші важливі деталі, такі як назви асоціацій та їхні типи.

Узагальнення використовується для визначення відношень між більш загальними класами та їх конкретними підкласами. Воно відображається у вигляді стрілки, яка вказує на більш загальний клас. Це відношення використовується для створення ієрархії класів.

Залежності вказують на те, як один клас використовує інший. Це може бути через параметри методів чи інші типи взаємодії. Вони відображаються пунктирними лініями.

Діаграма класів має важливе значення під час проектування програм та дає можливість розуміти структуру системи, її класи та взаємозв'язки між ними.

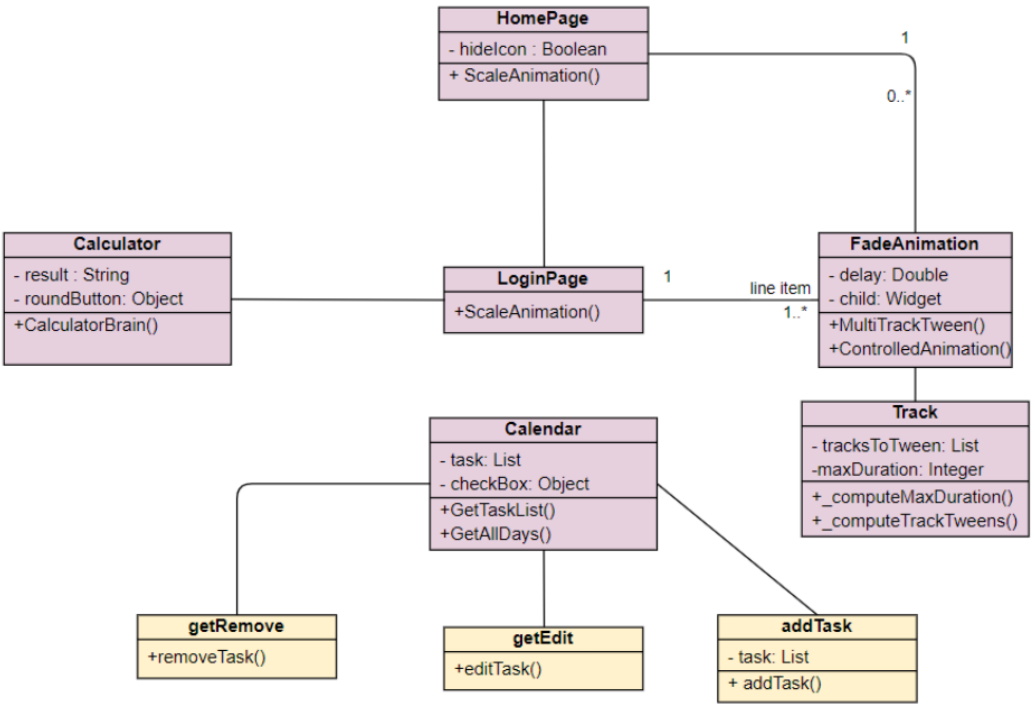


Рисунок 4.5 – Модель даних

Звичайна архітектура додатку була посилена додатковими класами, призначеними для поліпшення зрозуміння коду. Деякі класи були виділені для створення об'єктів та визначення констант. Наприклад, класи, такі як `FadeAnimation()`, `Track()`, `RoundButton()` - вони були використані для уніфікації стилю та створення єдиної структури інтерфейсу користувача по всьому додатку.

Додатково, для розробки мобільного додатку була розроблена діаграма використання (use case), яка описує взаємодію користувача з системою. Це один з компонентів моделі прецедентів, яка спрощує уявлення про систему на концептуальному рівні.

Прецедент описує можливості, які надає система (функціональність) і дозволяє користувачеві досягти конкретних результатів. Це може бути послуга або функція, яку пропонує система, описуючи типовий спосіб взаємодії користувача з системою. Варіанти використання використовуються для визначення зовнішніх вимог до системи та її функціоналу.

Ці діаграми важливі для визначення функціональності та поведінки системи. Вони дозволяють клієнтам, кінцевим користувачам та розробникам обговорювати систему, визначати пріоритети та очікувані функції. У разі моделювання системи діаграми прецедентів допомагають аналітикам:

- чітко відокремлювати систему від оточення;
- визначати, які користувачі та системи взаємодіють та їх очікувані дії;
- чітко визначати поняття, які стосуються детального опису функціоналу системи (прецедентів) в глосарії предметної області.

4.4 Адаптивність застосунку

Вирішення адаптивного дизайну стає важливим, особливо для розробників, які спрямовані на підтримку різних платформ з використанням

єдиної кодової бази. Це особливо відноситься до розробки на Flutter, оскільки цей інструмент орієнтований на основні платформи.

У Flutter існує безліч способів створення адаптивного дизайну. Один з методів полягає у використанні віджета MediaQuery для отримання інформації про поточний екран:

```
dart
Size screenSize = MediaQuery.of(context).size;
Orientation orientation = MediaQuery.of(context).orientation;
```

Далі розробник може використовувати цю інформацію для створення власних віджетів, заснованих на розмірах екрану чи його орієнтації. Крім цього, можна скористатися пакетами, які спрощують процес розробки адаптивного інтерфейсу. Зазвичай такі пакети надають стандартний інтерфейс, що полегшує налаштування різних варіантів дизайну в залежності від розмірів чи орієнтації екрану.

```
ResponsiveBuilder( builder: (context, sizingInformation)
{
if (sizingInformation.deviceScreenType == DeviceScreenType.desktop)
{ return Container(color:Colors.blue); }
if (sizingInformation.deviceScreenType == DeviceScreenType.tablet)
{ return Container(color:Colors.red); }
if (sizingInformation.deviceScreenType == DeviceScreenType.watch)
{
return Container(color:Colors.yellow); } return Container(color:Colors.purple);
},
},
);
}
```

Під час написання адаптивного коду важливо прагнути до гнучкості, яка дозволяє вирішувати різні сценарії адаптації без повторення умовних операцій (наприклад, використання операторів if / switch). Інколи адаптивний код варто робити гнучким лише в одному напрямку, наприклад, зміна цілочисельного значення (такого як crossAxisCount в GridView). Важливо забезпечити можливість створювати адаптивний код, що може бути цілочисельним значенням або віджетом, не навантажуючи код зайвими умовними операціями. Крім того, деякі розробники мають різні підходи до визначення контрольних точок у коді. Вбудовані компоненти Material, наприклад, як кнопка ElevatedButton, динамічно змінюють свій зовнішній вигляд відповідно до внутрішнього стану компонента, наприклад, коли кнопка натиснута, обрана або отримує фокус.

Такий же підхід можна застосувати для адаптивного інтерфейсу, де станом може бути обрана конфігурація UX, що відображається у ScreenScope:

```
class ScreenScope {
final double minWidth;
final double maxWidth;
final double minHeight;
final double maxHeight;
final Orientation?
orientation;
...
}
```

4.5 Реалізація анімацій

Анімація грає важливу роль у враженні користувача від вашої програми. Застосунок, який використовує безліч швидких та практично непомітних анімацій, виглядає більш вишукано та професійно порівняно з тим, де анімації відсутні. На переповненому ринку, як, наприклад, Google Play, це може вирішувати питання успіху чи невдачі вашого додатку.

Flutter пропонує один із найсучасніших двигунів для створення гібридних додатків на сьогодні. Ви можете створювати складні анімації, які відтворюються стабільно з частотою 60 кадрів в секунду.

Анімація типу "tween" є одним з найпопулярніших типів анімацій, який можна створити в Flutter. Для цього необхідно задати початкове та кінцеве значення. Фреймворк автоматично генерує проміжні значення між початковим і кінцевим, щоб плавно змінювати властивість вашого віджета.

Припустимо, що ми хочемо створити просту tween-анімацію, яка переміщує віджет від нижнього лівого кута екрану до верхнього правого кута. Наша мета - анімувати властивість left нашого віджета.

Для управління анімацією потрібні об'єкти Animation та AnimationController. Їх необхідно додати як змінні до вашого стану (state):
dart
Animation<double> animation;
AnimationController controller;

Щоб ініціалізувати обидва об'єкти, перевизначте метод `initState()` вашого класу. У цьому методі, викличте конструктор класу `AnimationController`, щоб ініціалізувати контролер. Також встановіть властивість `duration`, щоб визначити тривалість анімації.

dart
@0ver ride

```
void iniiState() {
  super.initState();
  controller = AnimationController(
    duration: Duration(seconds: 4),
    vsync: this,
  );
}
```

Цей код создає контролер для анімації - тривалістю всього чотири сек:

```
@override
void initState() {
   super.initState();
   controller = new AnimationController(vsync: this,
      duration: new Duration(seconds: 4));

   // More code here
}
```

Створюємо об'єкт Tween, визначаємо початок і кінець значення анімації.

```
Tween tween = new Tween<double>(begin: 10.0, end: 180.0);
```

Для зв'язку Tween з AnimationController,

```
animation = tween.animate(controller);
```

Подію анімації на кожен тик тікер, які повинні бути оброблені, щоб анімація спрацювала. У наступному коді показано як:

```
animation.addListener(() {
   setState(() {
   });
});
```

Так, метод `setState()` не потрібно викликати при роботі з анімацією, якщо ви не змінюєте стан інших змінних. Метод `forward()` з контролером анімації:

dart
controller.forward();

Це запустить анімацію. Створіть віджет, встановіть значення об'єкта `Animation` з властивостю `left`.

dart
Positioned(
  left: animation.value,
  child: Material(
    child: Icon(Icons.add),
  ),
),

Це застосує анімацію до віджету `Positioned`, який відображатиме `Material Icon`.

```
@override
Widget build(BuildContext context) {
  return new Container(
    color: Colors.white,
    child: new Stack(
     children: <Widget>[
      new Positioned(
        child: new Material(
          child: new Icon(Icons.airport_shuttle,
            textDirection: TextDirection.ltr,
            size: 81.0
          )
        ),
       left: animation.value, // Animated value
       top: 30.0  // Fixed value
      )
     ],

          textDirection: TextDirection.ltr,)
        );
    }
```

4.6 Налаштування певних залежностей

Dart вимагає налагодження залежностей. `pubspec.yaml` - це ключовий файл для розуміння середовища коду, написаного в цій мові.

У проектах Flutter кожен включає файл `pubspec.yaml` (див. рис), який зазвичай відомий як "pubspec". Цей файл містить важливу інформацію про ваш проект: назву, версію, залежності на пакети, ресурси, інструкції збірки та інше.

Що є ключовим для цього файлу? Наочно:

- name: Назва вашого проекту.

- description: Опис вашого проекту.

- version: Версія вашого проекту.

А далі йдуть секції, які визначають залежності вашого проекту, такі як пакети Flutter, використані пакети (з відповідними версіями), тести, середовище та інші конфігураційні параметри.

`pubspec.yaml` - це як земля для вашого проекту, де вказуються всі матеріали, які потрібні для його побудови та функціонування.

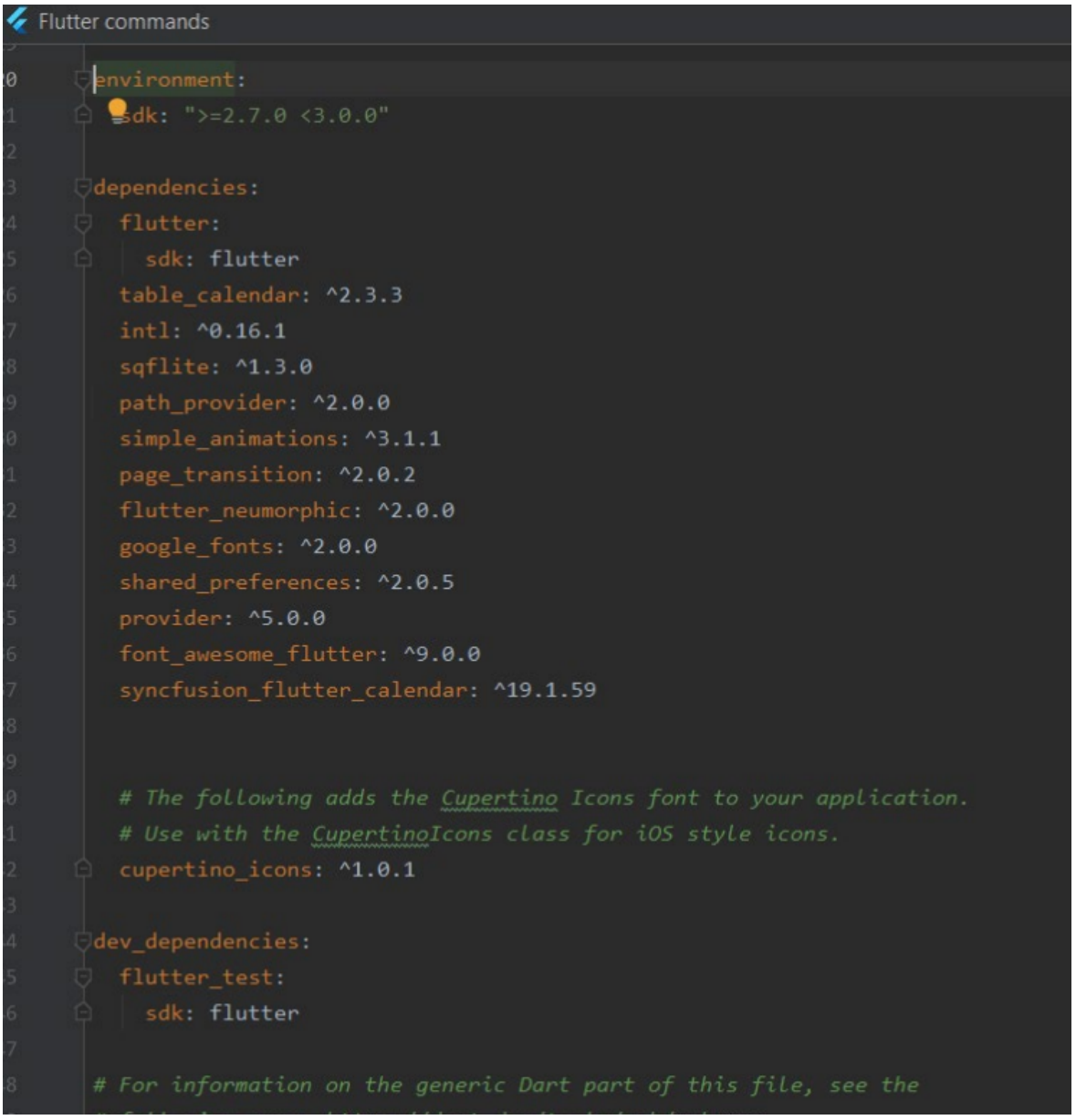


Рисунок 4.7 – Конфігураційні параметри

Базовий шаблон `pubspec` автоматично генерується під час створення нового проекту Flutter. Цей файл знаходиться у верхній частині каталогу проекту та містить метадані, які інструменти Dart і Flutter використовують

для роботи. Форматування в `pubspec` використовує YAML, придатний для читання людиною. Будьте уважні, оскільки пробіли (відступи) в цьому форматі мають значення.

Для використання певних шрифтів, фотографій, або зображень та інших ресурсів, їх потрібно вказати та оголосити в цьому файлі. Це підтвержується прикладом з рисунку, який є частиною дипломної роботи.

```
# To add custom fonts to your application, add a fonts section here,
# in this "flutter" section. Each entry in this list should have a
# "family" key with the font family name, and a "fonts" key with a
# list giving the asset and other descriptors for the font. For
# example:
# fonts:
#   - family: Schyler
#     fonts:
#       - asset: fonts/Schyler-Regular.ttf
#       - asset: fonts/Schyler-Italic.ttf
#         style: italic
#   - family: Trajan Pro
#     fonts:
#       - asset: fonts/TrajanPro.ttf
#       - asset: fonts/TrajanPro_Bold.ttf
#         weight: 700
#
# For details regarding fonts from package dependencies,
# see https://flutter.dev/custom-fonts/#from-packages
```

Рисунок 4.8 - Оголошення певних параметрів в файлі залежності

Flutter використовує підтримку різноманітних загальних пакетів, які розроблені спільнотою для використання в екосистемах Flutter та Dart. Дозволяє розробникам швидко створювати додатки без необхідності відтворювати всі функції з нуля.

Усі пакети мають номери версій, вказані в файлі `pubspec.yaml` кожного пакету. Там відображається версія пакету та список попередніх версій, які були випущені.

При додаванні пакетів до `pubspec.yaml` файлу використовуються наступні формати:

1. Обмеження діапазону: вказує мінімальну та максимальну версії, наприклад:

yaml

dependencies:

url_launcher: '>=5.4.0 <6.0.0'

2. Обмеження діапазону зі синтаксисом курсора, аналогічно звичайному діапазону:

yaml

dependencies:

collection: '^5.4.0'

За для оновлення Flutter SDK ви можете використати команду `flutter upgrade` з кореневого каталогу вашого застосунку (там, де знаходиться файл `pubspec.yaml`). Це допоможе оновити SDK Flutter і пакети до їхніх останніх доступних версій.

Рисунок 4.9 – Оновлення flutter

Так, команда `flutter upgrade` спочатку отримує найновішу версію Flutter SDK, яка доступна на обраному вами каналі Flutter. Після цього вона оновлює кожний пакет, від якого залежить ваш додаток, до останньої сумісної версії.

Якщо вам потрібна ще більш оновлена версія Flutter SDK, ви можете переключитись на менш стабільний канал Flutter, наприклад, `beta` або `dev`, а потім запустити команду `flutter upgrade`, щоб отримати оновлення з цього менш стабільного каналу. Варто пам'ятати, що версії на менш стабільних

каналах можуть містити нові функції та виправлення помилок, але в той же час бути менш стабільними, ніж версії на стабільних каналах.

5.РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розглянуто систему організації завдань, що вимагає встановлення архівного файлу для доступу до неї на будь-якому мобільному пристрої. Після завантаження та встановлення додатка на смартфон користувач може почати користуватися мобільним застосунком.

Додаток націлений на надання різноманітних інструментів для організації та виконання завдань, функціонуючи як органайзер, що забезпечує широкі можливості у керуванні різними завданнями та їх виконанні. Зараз він включає деякі функції, проте може бути покращений або розширений у майбутньому.

При першому запуску додатка користувач потрапляє до екрану вітання.

На етапі далі відбувається плавний анімаційний перехід до екрану авторизації. Ця сторінка була створена як приклад візуального розгляду процесу витягування даних, проте важливо підкреслити, що ці дані коректно захищені і без відповідного дозволу адміністрації не можуть бути використані. На рисунку 5.1 відображено поле для введення даних авторизації.

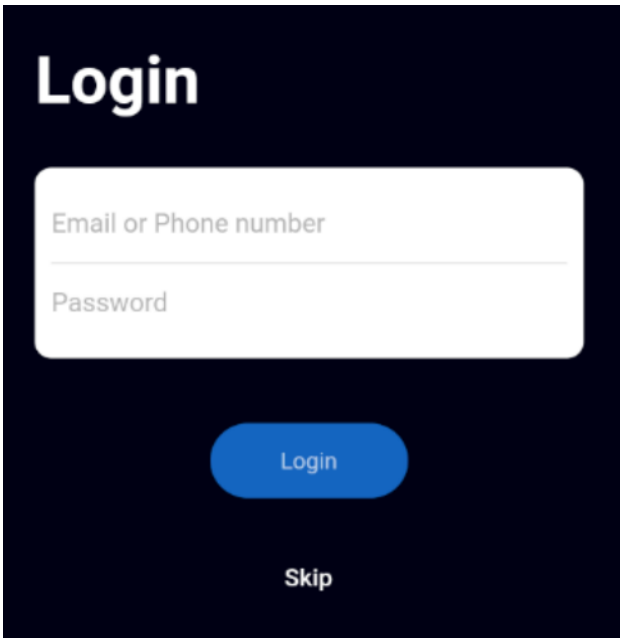


Рисунок 5.1 – Авторизація користувача

Потрапляємо на головну сторінку календаря, на якому відображаються справи та нотатки.

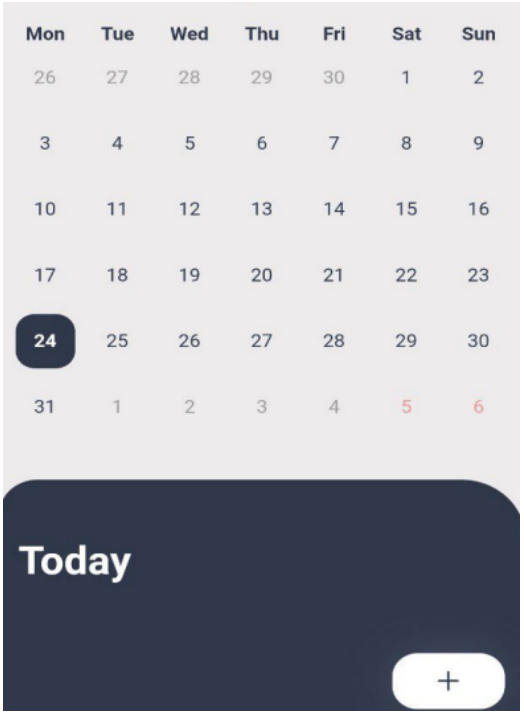


Рисунок 5.2 – Головна сторінка додатку

Рисунок 5.3 – Головне меню

Далі у користувача є можливість створювати записи, складати плани та виконувати прості математичні обчислення, якщо це необхідно.

На сторінці завдань доступний функціонал, який представлений на рисунку . Тут користувач має змогу створити нове завдання або нагадування. Запис складається шляхом введення назви та опису. У випадку, якщо одне з

полів залишиться порожнім, запис все одно буде створено. Але для більшої зручності користувача краще заповнити обидва поля.

Створені записи відображаються у вигляді списку нижче. Для доступу до цього списку необхідно обрати дату, з якою пов’язані створені записи.

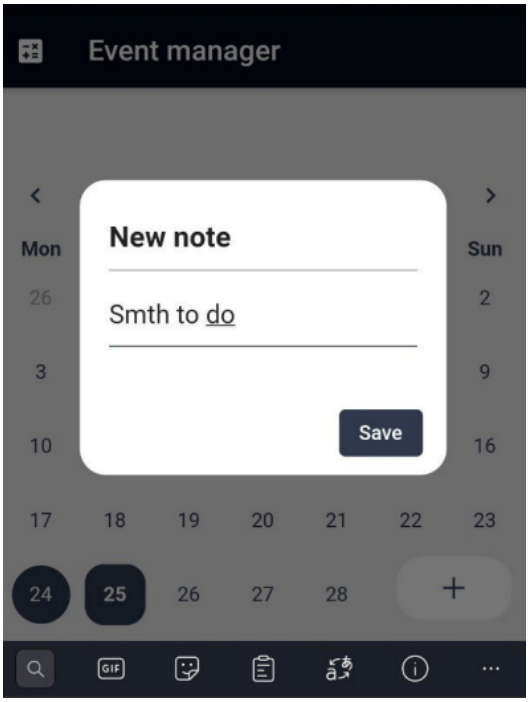


Рисунок 5.4 – Створення нотатку

Якщо користувач має намір відредагувати створену замітку, він може виконати свайп вправо. Після цього з’явиться вікно редагування, де можна змінити назву та детальний опис завдання. Крім цього, є можливість позначити завдання як виконане чи завершене.

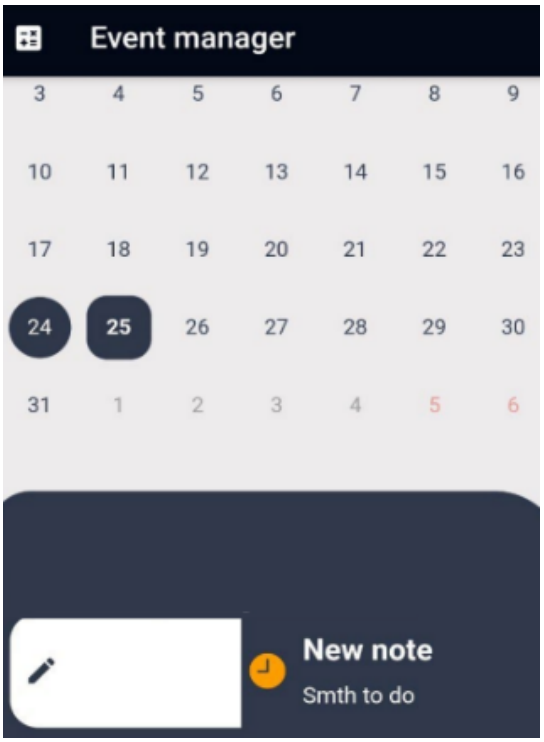


Рисунок 5.5 – Позначення статусу завдання

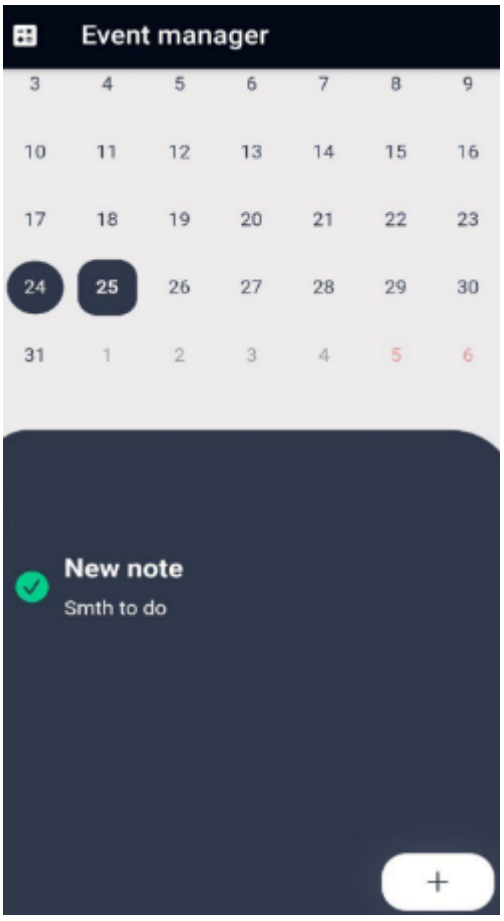


Рисунок 5.6 – Завершення справи

Користувач, у разі виникнення неактуальності справи чи відсутності необхідності редагування, має можливість видалити нотатку зі списку шляхом виконання свайпу вліво. Цей дійсний спосіб дозволяє автоматично видалити зазначену замітку, забезпечуючи ефективну систему управління списком нотаток та їх актуальністю :

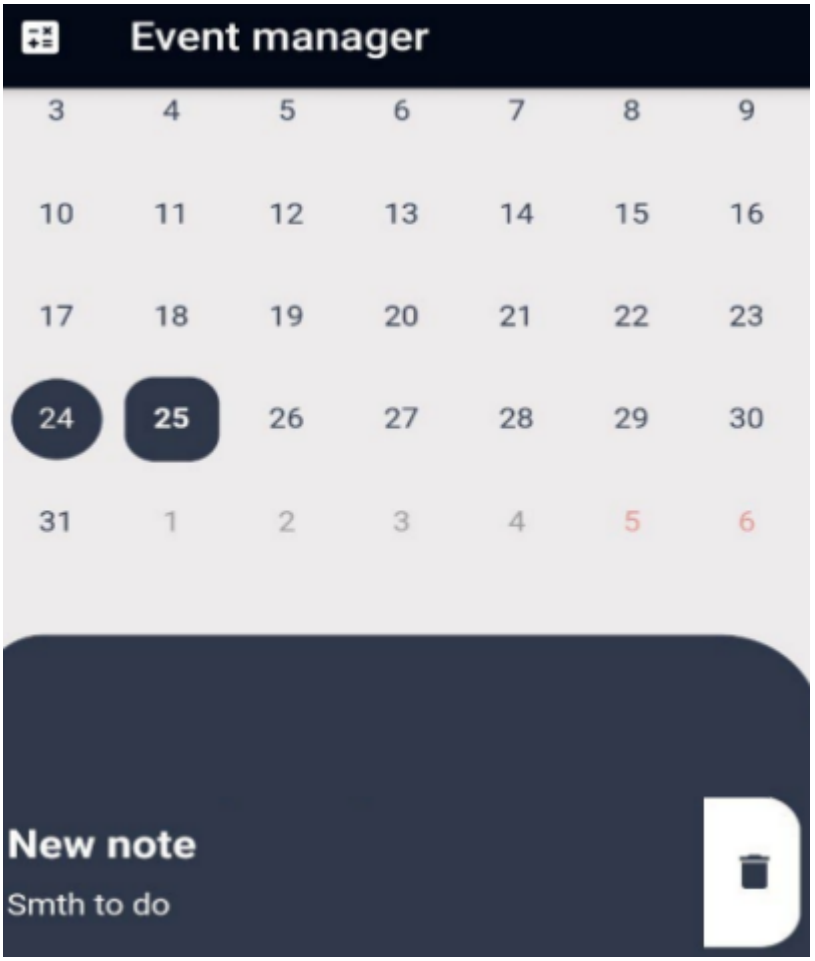


Рисунок 5.7 – Видалення нотатку

Отже, програмне забезпечення для мобільних пристроїв дозволяє створювати, налагоджувати та планувати різноманітні завдання, вносити зміни до свого розкладу, а також зберігати всі важливі справи та нагадування у смартфоні для ефективнішої та організованішої роботи.

ВИСНОВКИ

Отже, мобільний додаток "Планер" представляє собою комплексний інструмент, спрямований на полегшення організації роботи. Мобільний додаток дозволяє користувачам зручно планувати розклад, робити нотатки, вносити зміни у плани, відмічати завдання як виконані або видаляти їх.

Основні результати включають наступне:

1. Огляд та вибір технологій: Визначено найоптимальніші інструменти для створення застосунку, включаючи Android Studio та Dart на Flutter. Також SQLite для роботи з даними, враховуючи його швидкість.

2. Архітектура та тестування: Було роведено тестування та наведено приклади функцій, що підтверджують працездатність додатку та його відповідність поставленим вимогам.

3. Інтерфейс користувача: Розроблено зручний та легкий у використанні інтерфейс, який включає різноманітні функції, такі як калькулятор, планер, календар та інші.

4. Функціональні можливості: Додаток дозволяє створювати замітки, редагувати їх, позначати як виконані чи видаляти. Також присутній широкий функціонал, який включає калькулятор, планер, календар та карту містечка.

5. Можливість оновлення та подальші дослідження: Результати дослідження можуть бути використані університетськими працівниками та студентами, або іншими людьми. Додаток відкритий для оновлень та розширення функціоналу.

Зазначені результати роботи мобільного додатку є перспективними для подальших досліджень та можуть бути використані для поліпшення та розвитку програмного продукту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Рижкова, І.О. Огляд крос-платформених технологій розробки мобільних додатків. Наукові записки УНДІЗ. 2021. No1(53). С. 58-69.

2. Патлань В.М., Коротун О.С. Порівняльний аналіз кросплатформених фреймворків для створення мобільних додатків. Вісник НТУ "ХПІ". 2018. No 19(1295). С. 134-140.

3. Мисак Н.С., Древецький В.В. Flutter vs React Native: порівняльний аналіз для розробки крос-платформних мобільних додатків. Матеріали XVI науково-практичної конференції "Інформаційні технології: наука, техніка, технологія, освіта, здоров’я". Харків, 2020. С. 277.

4. Шахов С.В. Огляд особливостей розробки мобільних кросплатформених додатків на Xamarin. Вісник Черкаського державного технологічного університету. 2020. No 4. С.100-107.

5. Моргун А.С. Кросплатформні технології розробки мобільних додатків: переваги та недоліки. ScienceRise. 2015. No5/3(10). С. 43-48.

6. Гуляницький Л.Ф. Основи крос-платформної розробки мобільних додатків. Вісник Херсонського національного технічного університету. 2014. No 4(51). С. 76-81.

7. Жуковін О.О., Бука Т.В. Порівняльний аналіз платформ кросплатформеної розробки мобільних додатків. ScienceRise. 2016. No5/2(22). С. 25-31.

8. Левікін В.М. Підходи до розробки крос-платформенних мобільних додатків. Інформаційні технології та комп’ютерне моделювання. Івано-Франківськ, 2016. Випуск 3(2). С. 50-60.

9. Пилипенко О.О. Огляд сучасних крос-платформених фреймворків розробки мобільних додатків. Молодий вчений. 2021. No 1(93). С. 62-65.

10. Шинкаренко О.М. Вибір оптимальної кросплатформної технології для створення мобільного додатку. Молодий вчений. 2018. No4.1 (56.1). С. 90-93.

11. SQL Pocket Guide: A Guide to SQL Usage. 4th Edition. O'Reilly Media, 2021.C.354.

12. SQL for Data Analysis: Advanced Techniques for Transforming Data into Insights. 2021.C.357.

13. SQL in 10 Minutes a Day, Sams Teach Yourself. 2019.C.256.

14. SQL: 3 books 1 - The Ultimate Beginner, Intermediate & Expert Guides To Master SQL Programming Quickly with Practical Exercises. 2022.C.346.

15. SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. (Learn Coding Fast with Hands-On Project Book 5) Kindle Edition.2018.C.166.

Додаток А

Мобільний додаток

Лістинг програми

main.dart

```
import 'package:flutter/material.dart';
import 'package:todo_app/screens/welcome_screen.dart';
void main() {
 runApp(MyApp());
}
class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
 return MaterialApp(
 title: 'Flutter Demo',
 debugShowCheckedModeBanner: false,
 theme: ThemeData(
 primarySwatch: Colors.blue,
 ),
 home: HomePage(),
 );
 }
}
```

welcome.dart

```
import 'package:todo_app/animations/FadeAnimation.dart';
import 'package:todo_app/screens/login_screen.dart';
import 'package:flutter/material.dart';
import 'package:page_transition/page_transition.dart';
class Home extends Statef Widget {
 @override
 _HomePageState createState() => _HomePageState();
}
```

```
class _HomePageState extends State<HomePage> with
TickerProviderStateMixin{
AnimationController _scaleController;
AnimationController _scale2Controller;
AnimationController _widthController;
AnimationController _positionController;
Animation<double> _scaleAnimation;
Animation<double> _scale2Animation;
Animation<double> _widthAnimation;
Animation<double> _positionAnimation;
bool hideIcon = false;
);
_scaleAnimation = Tween<double>(
begin: 1.0, end: 0.8
).animate(_scaleController)..addStatusListener((status) {
if (status == AnimationStatus.completed) {
_widthController.forward();
}
});
_widthController = AnimationController(
vsync: this,
duration: Duration(milliseconds: 600)
);
).animate(_widthController)..addStatusListener((status) {
if (status == AnimationStatus.completed) {
_positionController.forward();
}
});
_positionController = AnimationController(
vsync: this,
```

```
duration: Duration(milliseconds: 1000)
);
_positionAnimation = Tween<double>(
begin: 0.0,
end: 215.0
).animate(_positionController)..addStatusListener((status) {
if (status == AnimationStatus.completed) {
setState(() {
hideIcon = true;
});
_scale2Controller.forward();
}
});
_scale2Controller = AnimationController(
vsync: this,
duration: Duration(milliseconds: 300)
);
_scale2Animation = Tween<double>(
begin: 1.0,
end: 32.0
).animate(_scale2Controller)..addStatusListener((status) {
if (status == AnimationStatus.completed) {
                    Navigator.push(context,                PageTransition(type:
PageTransitionType.fade, child:
      LoginPage()));
}
});
}
@override
Widget build(BuildContext context) {
```

```
final double width = MediaQuery.of(context).size.width;
return Scaffold(
backgroundColor: Color.fromRGBO(3, 9, 23, 1),
body: Container(
width: double.infinity,
child: Stack(
children: <Widget>[
Positioned(
top: -50,
decoration: BoxDecoration(
image: DecorationImage(
75
image: AssetImage('assets/images/one.png'),
fit: BoxFit.cover
)
),
)),
),
Positioned(
top: -100,
left: 0,
child: FadeAnimation(1.3, Container(
width: width,
height: 400,
decoration: BoxDecoration(
image: DecorationImage(
image: AssetImage('assets/images/one.png'),
fit: BoxFit.cover
)
),
```

```
)),
),
decoration: BoxDecoration(
image: DecorationImage(
image: AssetImage('assets/images/one.png'),
fit: BoxFit.cover
)
),
)),
),
Container(
padding: EdgeInsets.all(20.0),
child: Column(
mainAxisAlignment: MainAxisAlignment.end,
crossAxisAlignment: CrossAxisAlignment.start,
children: <Widget>[
  style: TextStyle(color: Colors.white.withOpacity(.7), height: 1.4,
fontSize:
19),)),
SizedBox(height: 180,),
builder: (context, child) => Container(
width: _widthAnimation.value,
height: 80,
padding: EdgeInsets.all(10),
decoration: BoxDecoration(
borderRadius: BorderRadius.circular(50),
color: Colors.blue.withOpacity(.4)
),
child: InkWell(
onTap: () {
```

```
_scaleController.forward();
},
child: Stack(
children: <Widget> [
AnimatedBuilder(
animation: _positionController,
builder: (context, child) => Positioned(
left: _positionAnimation.value,
child: AnimatedBuilder(
animation: _scale2Controller,
scale: _scale2Animation.value,
child: Container(
width: 60,
child: hideIcon == false ? Icon(Icons.arrow_forward, color:
Colors.white,) : Container(),
)
),
77
),
),
),
]
),
),
),
),
)),
)),
SizedBox(height: 60,),
],
```

```
),
)
],
),
),
);
}
}
```

login.dart

```
import 'package:todo_app/animations/FadeAnimation.dart';
import 'package:flutter/material.dart';
import 'package:todo_app/screens/home_page.dart';
Center(
 child: Container(
 padding: EdgeInsets.all(30),
 child: Column(
 crossAxisAlignment: CrossAxisAlignment.start,
 mainAxisAlignment: MainAxisAlignment.center,
 children: <Widget>[
 FadeAnimation(1.2, Text("Login",

 FadeAnimation(1.5, Container(
 padding: EdgeInsets.all(10),
78
 decoration: BoxDecoration(
 borderRadius: BorderRadius.circular(10),
 color: Colors.white
 ),
 child: Column(
 children: <Widget>[
```

```
Container(
decoration: BoxDecoration(
border:Colors.grey[300]))
),
child: TextField(

),
),
),
Container(
decoration: BoxDecoration(
),
child: TextField(
decoration: InputDecoration(
border: InputBorder.none,
hintStyle: TextStyle(color: Colors.grey.withOpacity(.8)),
hintText: "Password"
),
),
),
],
),
)),
SizedBox(height: 40,),
FadeAnimation(1.8, Center(
child: Container(
width: 120,
padding: EdgeInsets.all(15),
decoration: BoxDecoration(
borderRadius: BorderRadius.circular(50),
```

```
color: Colors.blue[800]
 Center(
child: Container(
margin: EdgeInsets.all(25),
// ignore: deprecated_member_use
child: OutlineButton(
  child: Text("Skip", style: TextStyle(fontSize: 15.0, color: Colors.white),),
 .push(MaterialPageRoute(builder: (context) => HomePage())),
),
),
),
// Center(
// child: Container(
// margin: EdgeInsets.all(25),
// child: FlatButton(
// child: Text('Skip', style: TextStyle(fontSize: 15.0),),
// shape: RoundedRectangleBorder(
// borderRadius: BorderRadius.circular(15)),
// color: Colors.blue[800],
// textColor: Colors.white.withOpacity(.7),
// onPressed: () {},
// ),
// ),
// ),
],
),
),
),
);
```

```
 }
}
80
home_page.dart
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:intl/intl.dart';
import 'package:table_calendar/table_calendar.dart';
import 'package:todo_app/database_helper.dart';
import 'package:todo_app/models/task.dart';
import 'package:todo_app/calculate_page.dart';
 @override
 _HomePageState createState() => _HomePageState();
}
class _HomePageState extends State<HomePage> {
 Color _themeColor=Color(0xff30384c);
 CalendarController calController;
 TextEditingController tfTitleController=TextEditingController();
 TextEditingController tfDecController=TextEditingController();
 GlobalKey<FormState> key=GlobalKey();

 super.initState();
 calController=CalendarController();
 getAllTasks();
 }
 Future<void> getAllTasks()async {
 List<Map<String,dynamic>> queryRows=await
DataBaseHelper.instance.onDateTasks(dateFormat.format(_selectedD
ate));
 queryRows.forEach((taskMap) {
```

```
Task toBeAdd=new Task();
toBeAdd.id=taskMap[DataBaseHelper.columnId];
toBeAdd.title=taskMap[DataBaseHelper.columnTitle];
toBeAdd.description=taskMap[DataBaseHelper.columnDescription];

toBeAdd.completed=taskMap[DataBaseHelper.columnCompleted]==1;
      81

toBeAdd.date=dateFormat.parse(taskMap[DataBaseHelper.columnDate]);
taskList.add(toBeAdd);
});
setState(() {
});
}

getEditBg() {
return Container(
padding: EdgeInsets.only(left: 10),
alignment: Alignment.centerLeft,
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.circular(20),
),
child: Icon(
Icons.edit,
color: _themeColor,
size:25,
),
);
}
```

```
getDeleteBg() {
return Container(
alignment: Alignment.centerRight,
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.circular(20),
),
padding: EdgeInsets.only(right: 10),
),
);
}
removeTask(index) async{
 int i=await DataBaseHelper.instance.deleteTask({
DataBaseHelper.columnId:tobeRemoved.id
});
setState(() {
print('row deleted $i');
});
}
}
Scaffold.of(context).showSnackBar(SnackBar(
duration: Duration(seconds: 1),
content: Text(
'${task.title} task deleted',
style: TextStyle(
fontSize: 14,
),
),
));
}
```

```
/ Future<void> showTaskDialog(BuildContext context,Task task,int taskIndex) async {
bool isCompleted=task!=null?task.completed:false;
tfTitleController.text=task!=null?task.title:tfTitleController.text;
83
tfDecController.text=task!=null?task.description:tfDecController.text;
return StatefulBuilder(
builder: (context,setState){
return AlertDialog(
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(20),
),
 content: Container(
child: Form(
key: key,
child: Column(
mainAxisSize: MainAxisSize.min,
children: [
decoration: InputDecoration(
hintText: "title",
hintStyle: TextStyle(
color: Colors.grey,
fontSize: 20,
),
focusedBorder: UnderlineInputBorder(
borderSide: BorderSide(
color: _themeColor,
)
)
),
```

```
),
SizedBox(height: 10,),
Container( constraints: BoxConstraints(
maxHeight: 50,
),
child: TextFormField(
controller: tfDecController,
maxLines: null,
style: TextStyle(
fontSize: 18,
),
decoration: InputDecoration(
hintText: "Description",
hintStyle: TextStyle(
color: Colors.grey,
fontSize: 18,
),
focusedBorder: UnderlineInputBorder(
borderSide: BorderSide(
color: _themeColor,
)
)
),
),
),
SizedBox(height: 10,),
task==null?SizedBox():Row(
mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: [
Text(
```

```
"Completed",
style: TextStyle(
color: _themeColor,
fontSize: 18,
),
),
Checkbox(
});
},
activeColor: _themeColor,
checkColor: Colors.white,
)
85
],
)
],
),
),
) ,
actions: [
Container(
padding: EdgeInsets.only(right: 10),
child: FlatButton(
color: _themeColor,
shape: RoundedRectangleBorder(
borderRadius: BorderRadius.circular(5)
),
 if(key.currentState.validate()) {
Task newTask=Task(
title: tfTitleController.text,
```

```
description: tfDecController.text,
 date: _selectedDate,);
tfTitleController.text='';
tfDecController.text='';
if(task==null) {
newTask.id=await DataBaseHelper.instance.insertTask(
{
DataBaseHelper.columnTitle:newTask.title,
DataBaseHelper.columnDescription:newTask.description,
DataBaseHelper.columnCompleted:newTask.completed?1:0,

DataBaseHelper.columnDate:dateFormat.format(newTask.date).toStri
ng(),
}
);
taskList.add(newTask);
} else {
newTask.id=task.id;
86
int result=await DataBaseHelper.instance.updateTask(
{
DataBaseHelper.columnId:newTask.id,
DataBaseHelper.columnTitle:newTask.title,
DataBaseHelper.columnDescription:newTask.description,
DataBaseHelper.columnCompleted:newTask.completed?1:0,

DataBaseHelper.columnDate:dateFormat.format(newTask.date).toStri
ng(),
}
);
```

```
print('row updated $result');
taskList.insert(taskIndex,newTask);
}
Navigator.of(context).pop();
}
},
),
)
],
);
}
);
});
}
 removeTask(index);
showSnackBar(context, task, index);

}
else {
int taskIndex=taskList.indexOf(task);
removeTaskFromList(index);
await showTaskDialog(context,task,taskIndex);
setState(() {
});

}
},
secondaryBackground: getDeleteBg(),
background: getEditBg(),
child: Card(
```

```
color: _themeColor,
elevation: 0,
shadowColor: Colors.white,
child: Row(
children: [
taskList[index].completed?
Icon(
CupertinoIcons.check_mark_circled_solid,
color: Color(0xff00cf8d),
size: 30,):
Icon(
CupertinoIcons.clock_solid,
color: Color(0xffff9e00),
size: 30,
),
SizedBox(width: 10,),
Container(
padding: EdgeInsets.only(left: 0,right: 0,top: 10),
width: MediaQuery.of(context).size.width*0.75,
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
'${taskList[index].title}',
style: TextStyle(
fontSize: 20,
 color: Colors.white
),
),
SizedBox(height: 10),
```

```
Text(
taskList[index].description,
style: TextStyle(
fontSize: 15,
),
),
SizedBox(height: 10),
],
),
)
],
),
),
),
),
),
);
}
 backgroundColor: Color(0xFFEDEBEC),
appBar: AppBar(
backgroundColor: Color.fromRGBO(3, 9, 23, 1),
leading: RaisedButton(
onPressed: () => Navigator.of(context)
.push(MaterialPageRoute(builder: (context) => CalculatorPage())),
elevation: 0.0,,
),
body: SingleChildScrollView(
child: Container(
padding: EdgeInsets.only(top: 50),
child: Column(
```

```
children: [
TableCalendar(
startingDayOfWeek: StartingDayOfWeek.monday,
calendarStyle: CalendarStyle(
weekdayStyle: TextStyle(color: _themeColor,fontWeight:
FontWeight.normal,fontSize: 14),
weekendStyle: TextStyle(color: _themeColor,fontWeight:
FontWeight.normal,fontSize: 14),
selectedColor: _themeColor,
todayColor: _themeColor,
todayStyle: TextStyle(
fontSize: 14,
color: Colors.white
)
),
onDaySelected: (date,events,holiday)=>{
setState(() {
_selectedDate=date;
taskList=List();
getAllTasks();
}),
},
builders: CalendarBuilders(
selectedDayBuilder: (context,date,events)=>
Container(
margin: EdgeInsets.all(5.0),
alignment: Alignment.center,
decoration: BoxDecoration(
color: _themeColor,
borderRadius: BorderRadius.circular(15),
```

```
),
child: Text(
date.day.toString(),
style: TextStyle(
90
color: Colors.white,
fontWeight: FontWeight.bold
),),
)
),
daysOfWeekStyle: DaysOfWeekStyle(
weekendStyle: TextStyle(
color: _themeColor,
fontWeight: FontWeight.bold
),
weekdayStyle: TextStyle(
color: _themeColor,
fontWeight: FontWeight.bold
)
),
headerStyle: HeaderStyle(
formatButtonVisible: false,
centerHeaderTitle: true,
titleTextStyle: TextStyle(
color: _themeColor,
fontWeight: FontWeight.bold,
fontSize: 20,
),
leftChevronIcon: Icon(
Icons.chevron_left,
```

```
color: _themeColor,
),
rightChevronIcon: Icon(
Icons.chevron_right,
color: _themeColor,
)
),
calendarController: calController,
),
SizedBox(height: 30,),
Container(
padding: EdgeInsets.only(left: 15,right: 10),
height: MediaQuery.of(context).size.height,
width: MediaQuery.of(context).size.width,
decoration: BoxDecoration(
color: _themeColor,
borderRadius: BorderRadius.only(topRight:
Radius.circular(50),topLeft:
Radius.circular(30))
91
),
child: Container(
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
SizedBox(height: 50,),
Text(
getDiff(),
style: TextStyle(
color: Colors.white,
```

```
),
),
getTaskList(),
],
),

)

)
],
),
),
),
floatingActionButton: Container(
padding: EdgeInsets.all(0),
decoration: BoxDecoration(
color: Colors.white,
borderRadius: BorderRadius.all(Radius.circular(20)),
boxShadow: [
BoxShadow(
color: Colors.white12,
blurRadius:30,
),
],
),
child: RawMaterialButton(
highlightColor: Colors.transparent,
92
onPressed: () async{
await showTaskDialog(context,null,null);
```

```
setState(() {
});
},
child: Icon(
CupertinoIcons.add,
color: _themeColor,
),
),
)
);
}
}
calculator_brain.dart
class CalculatorBrain {
String output = '0'; //store results to be printed
String _output = ''; // store calculation result
dynamic num1 =
0; // store output value when the arithmetic operator is pressed
dynamic num2 = 0; // store output value when the equal operator is pressed
String operator = ''; //store operator when operator button is pressed
String resultOperationText = ''; //store operation result below output text
bool isPressedPercentageButton = true; //determine percentage button state
String {
if (buttonText == 'AC') {
_output = '';
num1 = 0;
num2 = 0;
```

```
 operator = '';
 output = '0';
 resultOperationText = '';
 isPressedPercentageButton = true;
  _output = _output + '';
93
 output = _output;
 resultOperationText = output;
 } else {
 _output = '
-' + _output;
 output = _output;
 resultOperationText = output;

}
 return output;
 } else if (buttonText == '%') {
 if (isPressedPercentageButton) {
 if (output.contains('.')) {
 num1 = double.parse(output);
 } else {
 num1 = int.parse(output);

}
 _output = (num1 / 100).toString();
 output = _output;
 _output = '';
 num1 = 0;
 resultOperationText = output;
 return output;
```

```
}
 } else if (buttonText == "
+ "  | |
 buttonText == "
– "  | |
 buttonText == "÷" ||
 buttonText == "x") {
 if (output.contains('.')) {
 num1 = double.parse(output);
 } else {
 num1 = int.parse(output);

}
 operator = buttonText;
 resultOperationText = operator;
 isPressedPercentageButton = false;
 print(operator);
 _output = "";
 } else if (buttonText == ".") {
 if (_output.contains(".")) {
 print("Already conatains a decimals");
 _output = _output + ";
 output = _output;
 resultOperationText = output;
 } else {
94
 _output = _output + buttonText;
 output = _output;
 resultOperationText = output;
```

```
}
 } else if (buttonText == "
 = ") {
 isPressedPercentageButton = true;
 if (output.contains('.')) {
 num2 = double.parse(output);
 } else {
 num2 = int.parse(output);

}
 if (operator == "
 + ") {
 _output = (num1 + num2).toString();

}
 if (operator == "
 − ") {
 _output = (num1
- num2).toString();

}
 if (operator == "x") {
 _output = (num1 * num2).toString();

}
 if (operator == "÷") {
 _output = (num1 / num2).toString();

}
```

```
num1 = 0;
num2 = 0;
operator = "";
if (_output.contains('.')) {
output = double.parse(_output).toStringAsFixed(2);
} else {
output = _output;

}
resultOperationText = '';
_output = '';
} else {
_output = _output + buttonText;
output = _output;
resultOperationText = resultOperationText + buttonText;

}
print(_output);
return output;

}
}
```