МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ХАРЧЕНКО ЕРСАН КЕНАНОВИЧ

Допускається до захисту:

в.о. завідувача кафедри інформаційних технологій

канд. техн. наук, доцент

_____ О. В. Зелінська

«____»_____ 2024 р.

СТВОРЕННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ РАЙДШЕРИНГУ ЗАСОБАМИ FLUTTERFLOW

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:
Ю. В. Поремський, ст. викладач
кафедри інформаційних технологій,
к. т. н., ст. викладач

_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

Вінниця – 2024

АНОТАЦІЯ

Харченко Е. К. Створення та реалізація мобільного додатку райдшерингу засобами FlutterFlow. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (бакалаврській) роботі досліджено процес створення та реалізації мобільного додатку для райдшерингу з використанням FlutterFlow. Показано вплив інтеграції платформи Firebase на функціональність та безпеку додатку. Встановлено ефективність використання FlutterFlow для швидкого та зручного розроблення інтерфейсу користувача.

Ключові слова: FlutterFlow, Flutter, Firebase, розробка, шифрування, інтерфейс, база даних, додаток, райдшеринг, інтеграція, мобільний, безпека, карта, геолокація, поїздка, водій, пасажир, маршрут, спільна поїздка.

ABSTRACT

Kharchenko E. K. Creation and implementation of a ridesharing mobile application using FlutterFlow. Specialty 122 «Computer Science» educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia, 2024.

The process of creating and implementing a mobile application for ridesharing using FlutterFlow was investigated in the qualification (bachelor's) thesis. The influence of the integration of the Firebase platform on the functionality and security of the application is shown. Established the effectiveness of using FlutterFlow for fast and convenient user interface development.

Keywords: FlutterFlow, Flutter, Firebase, development, encryption, interface, database, app, ridesharing, integration, mobile, security, map, geolocation, ride, driver, passenger, route, rideshare.

ЗМІСТ

ВСТУП

У сучасному світі, де мобільні технології набувають все більшого значення, створення та вдосконалення мобільних додатків для різних сфер життя стає актуальним завданням. Зокрема, в рамках сучасного транспортного сектору, виникає потреба у зручних та ефективних рішеннях для організації поїздок та спільних подорожей.

Райдшеринг, або спільний користування транспортом, стає все більш популярним способом переміщення великої кількості людей. Цей підхід не лише сприяє оптимізації використання автомобілів, але й забезпечує економічну та екологічну ефективність.

У цьому контексті, важливо розробити мобільний додаток для райдшерингу, який би поєднував в собі зручність використання, безпеку користувачів, ефективність управління поїздками та якісну підтримку. Використання інструментів, таких як FlutterFlow для візуальної розробки та Firebase для забезпечення функціональності та безпеки, дозволяє швидко та ефективно створювати та впроваджувати такі додатки на ринку.

Тому, тема "Створення та реалізація мобільного додатку райдшерингу засобами FlutterFlow" є дуже актуальною і важливою з точки зору подальшого розвитку транспортного сектору та покращення якості життя користувачів мобільних додатків.

Мета цієї дипломної роботи полягає в створенні інноваційного мобільного додатку для райдшерингу, який буде забезпечувати користувачам зручність, безпеку та ефективність у спільних поїздках. Цей додаток має на меті полегшити процес знаходження попутників та організації поїздок, зменшити транспортні навантаження та сприяти збереженню навколишнього середовища шляхом оптимізації використання автомобілів.

Крім того, дослідження такого важливого та актуального питання сприятиме розширенню знань у галузі мобільної розробки, інтеграції хмарних сервісів та покращенню якості транспортних послуг для користувачів.

Завдання дослідження:

1. Ознайомитись з основами роботи у середовищі FlutterFlow.
2. Дослідити можливості інтеграції Firebase з FlutterFlow для реалізації аутентифікації користувачів, зберігання даних, безпеки та інших функціональних можливостей.
3. Спроектувати та розробити додаток для райдшерингу з використанням FlutterFlow.

Об'єктом дослідження дипломної роботи є мобільний додаток для райдшерингу, розроблений з використанням FlutterFlow та інтегрований з Firebase. Досліджується процес створення та реалізації додатку, його функціональність, ефективність і безпека для забезпечення зручного та надійного інструменту для організації спільних поїздок та управління ними.

Предметом дослідження дипломної роботи є розробка мобільного додатку для райдшерингу за допомогою FlutterFlow та інтеграція його з Firebase для забезпечення зручності, безпеки та ефективності в організації спільних поїздок та управлінні ними.

Теоритичне значення одержаних результатів:

1. Визначення ефективних стратегій розробки мобільних додатків для райдшерингу з використанням інструментів FlutterFlow та інтеграції з Firebase.
2. Розроблений інтерфейс користувача, який відповідає сучасним стандартам та вимогам зручності використання.
3. Вивчення та впровадження найкращих практик забезпечення безпеки та конфіденційності даних користувачів в мобільних додатках.
4. Досліджено можливості використання хмарних сервісів для підтримки та оптимізації функціоналу додатку.
5. Оцінено ефективність та задоволення користувачів використанням розробленого додатку у рамках райдшерингу.

Практичне значення одержаних результатів:

1. Покращення зручності та ефективності користування: Розроблений додаток дозволяє користувачам швидко та зручно знаходити попутників для

спільних поїздок, що сприяє зменшенню транспортних заторів та оптимізації використання автотранспорту.

2. Забезпечення безпеки та конфіденційності: Інтеграція з Firebase дозволяє забезпечити високий рівень безпеки та конфіденційності даних користувачів, що є критичним аспектом для додатку райдшерингу.

3. Екологічний внесок: Зменшення кількості особистих автомобілів на дорогах завдяки райдшерингу сприяє зменшенню викидів шкідливих речовин у атмосферу та сприяє збереженню навколишнього середовища.

4. Підтримка спільноти та соціальна взаємодія: Додаток сприяє зближенню та співпраці між користувачами, допомагаючи їм знаходити спільні інтереси та спільно організовувати поїздки.

5. Послуги для бізнесу: Додаток може також мати практичне значення для комерційних організацій, які надають послуги транспортування, дозволяючи їм ефективно керувати та оптимізувати використання свого автопарку.

6. Дослідження та підтримка мобільних технологій: Результати роботи над додатком можуть бути корисні для подальших досліджень та розвитку мобільних додатків у сфері транспортування та спільних подорожей.

Апробація результатів дослідження:

Досліджені результати та розроблений додаток райдшерингу можуть бути апробовані та визнані через наступні кроки:

1. Представлення науковій громадськості: Участь у наукових конференціях та семінарах дозволить представити отримані результати перед вченими та спеціалістами галузі для отримання їхньої оцінки та зворотного зв'язку.

2. Публікація в наукових журналах: Опублікування статей наукових досліджень у відомих наукових журналах дозволяє отримати визнання та підтвердження наукового статусу отриманих результатів.

3. Демонстрація практикуючим спеціалістам: Організація презентацій та виставок для професіоналів та практиків галузі дозволяє показати практичну вартість дослідження та розробленого додатку.

4. Взаємодія зі спільнотою користувачів: Залучення користувачів для тестування та використання розробленого додатку, отримання їхніх відгуків та пропозицій для подальшого вдосконалення.

Ці кроки дозволять підтвердити важливість та практичне значення отриманих результатів дослідження у реальному використанні та науковому співтоваристві.

Структура кваліфікаційної (бакалаврської) роботи скаладається з 3 розділів, 2 з яких теоретичних та 1 практичний, з висновків та додатків.

РОЗДІЛ 1
ВИКОРИСТАННЯ FLUTTERFLOW ДЛЯ РОЗРОБЛЕННЯ МОБІЛЬНИХ ДОДАТКІВ

1.1. Огляд інструменту FlutterFlow: підходи до швидкої розробки мобільних додатків

FlutterFlow - це інструмент, який забезпечує можливість створення мобільних додатків на платформі Flutter без необхідності написання коду з нуля. Однією з його ключових переваг є візуальний редактор, що дозволяє розробникам просто перетягувати та налаштовувати компоненти для створення інтерфейсів додатків.

Підходи до швидкої розробки мобільних додатків з використанням FlutterFlow можуть включати в себе використання готових шаблонів, які прискорюють процес створення основних елементів додатку. Також важливо відзначити можливість налаштовувати анімації, переходи між екранами та інші ефекти, що дозволяють створювати більш привабливий інтерфейс. [1]

З використанням FlutterFlow можна ефективно керувати структурою додатку та його компонентами, що дозволяє розробникам швидко впроваджувати зміни та додавати новий функціонал без зайвих зусиль.

Для команд розробників важливою перевагою є можливість спільної роботи над проектом, обмін даними та швидка інтеграція з різними сервісами та інструментами, що дозволяє прискорити процес розробки та покращити його якість.

Заснований на Flutter, FlutterFlow дозволяє розробникам створювати мобільні додатки за допомогою візуального редактора, що значно спрощує роботу над проектом та зменшує час на його виконання. [2]

Одним із переваг FlutterFlow є гнучкість налаштування компонентів і інтерфейсу, що дозволяє створювати унікальні дизайни, враховуючи потреби користувачів та сучасні тренди в дизайні мобільних додатків.

Ще однією перевагою є підтримка FlutterFlow широкого спектру функціональності, такої як реактивність, анімації, робота з базами даних та API, що дозволяє розробникам створювати повнофункціональні та ефективні додатки для різних цілей.

Також варто відзначити, що FlutterFlow активно оновлюється та розширюється, додаючи нові функції та покращуючи інтерфейс користувача. Це забезпечує розробникам доступ до останніх технологій та можливостей у сфері розробки мобільних додатків.

Інтуїтивний інтерфейс FlutterFlow дозволяє навіть новачкам у мобільній розробці швидко засвоїти основи та почати створювати власні додатки. Це дозволяє швидше втілювати ідеї та прототипи без значного витрачання часу на вивчення складних технічних аспектів. [3]

FlutterFlow підтримує різні види екранів і сторінок, що дозволяє створювати додатки з різноманітним функціоналом та навіть складними інтерфейсами. Розробники можуть легко налаштовувати навігацію між екранами та взаємодіювати з різними компонентами.

Один з важливих аспектів FlutterFlow - це можливість перевіряти додаток у реальному часі без необхідності компіляції коду. Це дозволяє швидко виявляти помилки та вносити зміни на льоту, що значно спрощує процес тестування та вдосконалення додатку.

Крім того, FlutterFlow підтримує різні типи додаткового функціоналу, такі як вбудовані картки, календарі, форми введення та багато іншого. Це дозволяє створювати більш різноманітні та цікаві додатки, які задовольняють потреби користувачів. [5]

У підсумку, FlutterFlow є інноваційним та потужним інструментом для розробки мобільних додатків, який поєднує в собі зручність використання, багатий функціонал та можливість розширення за допомогою інтеграції з Firebase та іншими сервісами.

Низькодоковий підхід: FlutterFlow базується на концепції низькодокового програмування, що дозволяє створювати мобільні додатки шляхом візуального

моделювання. Це дозволяє швидко створювати прототипи та основний функціонал додатків без потреби писати велику кількість коду з нуля. Підходи низькодокового програмування, на яких базується FlutterFlow, є ключовими для швидкої та ефективної розробки мобільних додатків. Цей підхід передбачає використання графічного інтерфейсу та візуального моделювання, що значно спрощує процес створення додатків. [4]

Основна перевага низькодокового підходу полягає у тому, що розробники не залежать від великої кількості коду та не потребують глибоких знань у програмуванні. Використання графічного інтерфейсу дозволяє створювати елементи додатків за допомогою перетягування та налаштування властивостей, що значно прискорює процес розробки.

Одним з головних переваг низькодокового підходу є швидкість розробки. Розробники можуть швидко створювати прототипи додатків та експериментувати з різними ідеями без необхідності витрачати багато часу на написання коду з нуля. Це дозволяє більш оперативно відгукатися на зміни вимог користувачів.

Іншою важливою характеристикою низькодокового підходу є зниження порогу входу для новачків у сфері мобільної розробки. Замість глибокого розуміння програмування, користувачі можуть використовувати інтерфейс FlutterFlow для створення складних функціональних елементів, що раніше були доступні тільки програмістам.

Таким чином, низькодоковий підхід, на якому ґрунтується FlutterFlow, є важливим інструментом для розробки мобільних додатків, що дозволяє швидко втілювати ідеї у життя, зберігаючи при цьому високу якість та ефективність процесу розробки.

Готові компоненти та шаблони: FlutterFlow має вбудовані готові компоненти та шаблони, які можна використовувати для швидкої реалізації різноманітних елементів інтерфейсу та функціональності додатку. Це дозволяє значно економити час та зусилля розробників. [6]

Готові компоненти та шаблони в FlutterFlow відкривають широкі можливості для швидкої та ефективної розробки мобільних додатків. Цей

інструмент надає розробникам доступ до великого набору готових елементів інтерфейсу, які можна легко і швидко інтегрувати в проект.

Однією з головних переваг є можливість використання готових компонентів для створення складних інтерактивних елементів, таких як кнопки, панелі меню, списки, картки і багато інших. Це дозволяє розробникам уникнути витрат часу на написання коду з нуля і зосередитися на інших аспектах розробки додатку.

Використання готових компонентів і шаблонів також сприяє стандартизації дизайну та інтерфейсу додатку. Це робить додаток більш зручним для користувачів, оскільки вони можуть легко розуміти та навігувати в додатку, використовуючи вже відомі елементи.

Наявність готових компонентів та шаблонів у FlutterFlow дозволяє розробникам швидко та ефективно створювати мобільні додатки, зосереджуючись на творчості та функціональності, а не на вирішенні технічних деталей з нуля. [7]

Використання готових компонентів і шаблонів дозволяє розробникам економити час на тестування та відлагодження. Оскільки ці компоненти вже пройшли перевірку на відповідність стандартам та оптимізацію, вони зазвичай працюють без помилок і вимагають менше зусиль для їх налагодження.

Мультиплатформність: FlutterFlow підтримує мультиплатформну розробку, що означає, що один і той же код можна використовувати для створення додатків для різних платформ, таких як Android та iOS. Це дозволяє прискорити процес розробки та знизити витрати на обслуговування кількох кодових баз.

Мультиплатформність в FlutterFlow становить важливу перевагу для розробників мобільних додатків, оскільки вона дозволяє використовувати один і той же код для створення додатків для різних платформ, таких як Android та iOS. Це спрощує процес розробки та знижує витрати на обслуговування кількох кодових баз. [8]

Завдяки мультиплатформності розробники можуть швидко реагувати на зміни на ринку мобільних технологій та адаптувати свої додатки під різні

платформи без втрати якості. Це дозволяє їм ефективно використовувати свої ресурси та швидше виводити додатки на ринок.

Зараз мультиплатформність стає все більш важливою для розробки мобільних додатків, оскільки користувачі використовують різні пристрої та платформи. FlutterFlow забезпечує зручний спосіб створення додатків, які працюють на різних ОС без потреби писати окремий код для кожної платформи.

Одним із ключових переваг мультиплатформності є зменшення часу та зусиль, необхідних для розробки та тестування додатків. Розробники можуть сконцентруватися на вдосконаленні функціоналу та дизайну додатку, а не на вирішенні проблем, пов'язаних із сумісністю з різними платформами.

Отже, використання мультиплатформності у FlutterFlow дозволяє розробникам ефективно створювати додатки, які працюють на різних платформах, забезпечуючи єдинообразний досвід користувача та збільшуючи потенціал успіху на ринку мобільних додатків. [9]

Інтерактивне моделювання: За допомогою FlutterFlow можна візуально моделювати та налаштовувати різні елементи додатку, такі як екрани, кнопки, поля введення тощо, та негайно бачити результати в реальному часі. Це дозволяє розробникам швидко та ефективно тестувати та вдосконалювати додаток.

Інтерактивне моделювання, яке надає FlutterFlow, виявляється дуже корисним для розробників мобільних додатків. Завдяки цій можливості, розробники можуть візуально створювати та налаштовувати різні елементи інтерфейсу додатку без необхідності написання коду вручну. Це дозволяє їм швидко прототипувати та тестувати нові ідеї, а також вдосконалювати вже існуючі елементи.

Один з головних плюсів цього підходу полягає у тому, що розробники можуть негайно бачити результати своєї роботи в реальному часі. Це дозволяє ефективно виявляти та виправляти помилки, а також швидко адаптувати додаток до змінних вимог користувачів чи ринкових умов.

Інтерактивне моделювання також сприяє співпраці між розробниками та дизайнерами, оскільки вони можуть спільно працювати над виглядом та

функціоналом додатку, візуалізуючи всі зміни безпосередньо в середовищі FlutterFlow. Це дозволяє створювати додатки, які не лише ефективно працюють, а й мають привабливий та користувацьки-орієнтований дизайн.

Спільна робота та обмін ресурсами: FlutterFlow підтримує спільну роботу над проектами, що дозволяє командам розробників ефективно співпрацювати, ділитися ресурсами та швидше досягати поставлених цілей у розробці додатків.

Інструмент FlutterFlow дозволяє розробникам спільно працювати над проектами та обмінюватися ресурсами для ефективного створення мобільних додатків. Розглянемо це ширше:

Колаборація в FlutterFlow дозволяє розробникам з різних команд спільно працювати над проектами, ділитися ідеями та взаємодіяти над створенням якісних додатків. Це створює сприятливе середовище для обміну досвідом, швидшого виявлення проблем та швидкої реалізації вдосконалень.

Команди в FlutterFlow можуть спільно тестувати додатки безпосередньо в середовищі розробки, що дозволяє ефективно виявляти та виправляти помилки, а також вдосконалювати функціонал на підставі спільних тестів. [10]

Завдяки можливості обміну ідеями та кращими практиками, розробники в FlutterFlow можуть швидше розвивати свої навички та знання, що сприяє підвищенню продуктивності та якості роботи.

Узагальнюючи, спільна робота та обмін ресурсами у FlutterFlow сприяють ефективному вирішенню завдань розробки, розвитку якісних додатків та підвищенню продуктивності розробничих команд.

1.2. Принципи побудови інтерфейсу в FlutterFlow: компоненти, структура та патерни дизайну

В FlutterFlow компоненти є основними будівельними блоками інтерфейсу. Це можуть бути кнопки, тексти, картинки, поля введення тощо. Компоненти мають свої властивості, які можна налаштовувати, такі як розмір, кольори, шрифти тощо. Гнучка система компонентів дозволяє швидко створювати та налаштовувати різноманітні елементи інтерфейсу. [11]

Компоненти в FlutterFlow представляють собою різноманітні елементи інтерфейсу, які дозволяють будувати функціональні та естетично приємні додатки. Основні типи компонентів включають в себе такі елементи, як кнопки, тексти, картинки, поля введення, списки, вкладки, меню, табличні дані, анімації та багато інших.

Почнемо з кнопок. Кнопки використовуються для створення інтерактивних елементів, що реагують на натискання користувача. Вони можуть мати різний стиль, кольори, розміри та іконки, що дозволяє створювати різноманітні кнопки для різних дій та ефектів. [12]

Тексти використовуються для відображення інформації користувачеві. FlutterFlow надає можливість налаштовувати розмір шрифту, колір тексту, стиль (жирний, курсивний), вирівнювання та інші властивості текстового елемента.

Картинки використовуються для відображення графічних зображень. Їх можна імпортувати з різних джерел або використовувати готові ресурси. Картинки також можна налаштовувати за розміром, обрізати, додавати ефекти та анімацію.

Поля введення дозволяють користувачам вводити текстові дані. Вони можуть бути однорядковими або багаторядковими, мати обмеження на введення деяких символів або формату даних, таких як електронна пошта або номер телефону.

Списки використовуються для відображення набору елементів, які можуть бути прокручуваними. Вони можуть бути однорядковими або містити багаторядкові елементи, інтерактивні елементи (наприклад, відмітки чекбоксів) та багато іншого. [13]

Додатково, FlutterFlow дозволяє створювати вкладки для організації контенту на декілька сторінок, меню для навігації між різними екранами, табличні дані для відображення структурованої інформації у вигляді таблиць, анімації для надання динамічності та живості інтерфейсу та багато інших функціональних та дизайнерських елементів.

Усі ці компоненти мають свої унікальні властивості, які дозволяють налаштовувати їх зовнішній вигляд та поведінку відповідно до потреб додатку. Це забезпечує гнучкість та можливість створювати різноманітні та інноваційні інтерфейси для мобільних додатків на FlutterFlow.

Далі слід розглянути структуру додатку в FlutterFlow. Додаток складається з екранів, які відповідають за відображення різних частин функціоналу. Кожен екран містить компоненти, що утворюють його інтерфейс. Структура додатку визначається логікою та потребами користувачів, що дозволяє створювати зручні та логічно організовані додатки. [14]

Структура додатку в FlutterFlow визначається його функціональністю та взаємодією з користувачем. Основним елементом структури є екрани, які відображають різні частини додатку та відповідають за відображення відповідного контенту.

Кожен екран може мати свою унікальну структуру та складатися з різних компонентів. Наприклад, у додатку райдшерингу можуть бути такі екрани як "Головний екран з пошуком поїздок", "Екран вибору маршруту", "Профіль користувача" тощо. Кожен з цих екранів має свої унікальні компоненти, які відображають відповідний контент та функціонал.

Крім екранів, важливою частиною структури додатку є навігація між екранами. FlutterFlow дозволяє легко налаштовувати різні види навігації, такі як таби, вкладки, бокове меню, що дозволяє користувачам зручно переміщатися між різними частинами додатку. [15]

Під час створення структури додатку в FlutterFlow важливо враховувати не лише функціональність, а й естетичний вигляд та зручність для користувачів. Наприклад, розташування компонентів на екрані може бути оптимізоване для зручного взаємодії, забезпечуючи достатній розмір та відступи між елементами.

Крім того, важливо забезпечити правильну організацію робочих процесів в додатку, наприклад, через використання правильних патернів дизайну та архітектури. Це допомагає зберігати код додатку чистим, організованим і легко змінюваним у майбутньому.

Структура додатку в FlutterFlow повинна бути добре продуманою та забезпечувати зручну роботу розробників та користувачів, ефективне управління ресурсами та можливість масштабування для майбутніх розвитку.

Патерни дизайну є важливою складовою процесу розробки в FlutterFlow. Вони визначають стандартні підходи до організації коду та структурування додатку.

У FlutterFlow, як і в будь-якому іншому фреймворку, можна використовувати різні патерни дизайну в залежності від потреб проекту. Розглянемо деякі з найпоширеніших патернів:

1. MVC (Model-View-Controller):

Модель (Model): Відповідає за дані та бізнес-логіку додатку.

Вид (View): Відображає інтерфейс користувача та реагує на події від користувача.

Контролер (Controller): Керує взаємодією між Моделлю та Видом, обробляючи запити від користувача та оновлення даних.

Модель (Model) у паттерні MVC відповідає за представлення даних та бізнес-логіку додатку. Це означає, що вона визначає, які дані потрібні для додатку, як вони будуть зберігатися та які операції можна здійснювати з цими даними. Наприклад, якщо створюється додаток для управління завданнями, то модель може включати класи для представлення завдань, їх властивостей (наприклад, назва, дата завершення, пріоритет тощо) і методів для додавання, видалення та редагування завдань. [16]

Вид (View) в MVC відображає інтерфейс користувача та відображає дані, представлені моделлю. Він відповідає за те, щоб користувач міг бачити і взаємодіяти з даними додатку. Наприклад, для додатку управління завданнями, вид може включати різні екрани для списку завдань, деталей конкретного завдання, форми для додавання нових завдань тощо. Вид також реагує на події користувача, такі як натискання кнопок або введення даних, і відправляє ці події до контролера для обробки.

Контролер (Controller) управляє взаємодією між моделлю та видом. Він отримує події від виду, наприклад, коли користувач натискає кнопку для збереження завдання, і виконує відповідні дії з моделлю, такі як збереження нового завдання в базі даних. Контролер також може отримувати дані від моделі, які потрібні для відображення у вигляді у виді. Він реалізує логіку додатку, включаючи обробку запитів від користувача та оновлення даних у моделі. [17]

2. MVVM (Model-View-ViewModel):

Модель (Model): Аналогічно до MVC, відповідає за дані та бізнес-логіку.

Вид (View): Відображає інтерфейс та реагує на дії користувача.

ViewModel: Містить логіку, яка не входить або пов'язана з моделлю та видом, допомагає розділити логіку інтерфейсу від даних.

У паттерні MVVM (Model-View-ViewModel), як і в MVC, модель (Model) відповідає за представлення даних та бізнес-логіку додатку. Вона визначає структуру та операції з даними, а також взаємодію з джерелами даних, такими як база даних або веб-сервіси.

Вид (View) в MVVM відображає інтерфейс користувача та реагує на дії користувача. Він відображає дані, отримані від ViewModel, та передає дії користувача до ViewModel для обробки. Відмінність MVVM від MVC полягає в тому, що вид не взаємодіє безпосередньо з моделлю, а отримує дані через ViewModel.

ViewModel в MVVM містить логіку, яка не входить або пов'язана з Моделлю та Видом. Його основна функція - це відокремлення логіки інтерфейсу від даних. ViewModel обробляє запити від виду, взаємодіє з моделлю для отримання та оновлення даних, та підготовлює дані для відображення у виді.

Основна перевага MVVM полягає у відокремленні логіки представлення та бізнес-логіки, що полегшує тестування та підтримку коду. Крім того, цей підхід дозволяє використовувати та перевикористовувати компоненти інтерфейсу та логіку у різних частинах додатку. [18]

3. Singleton (Одинак):

Цей паттерн дозволяє створювати клас, який має тільки один екземпляр і надає глобальний доступ до цього екземпляра.

Паттерн Singleton (Одинак) використовується для створення класу, який має лише один екземпляр у всій програмі та забезпечує глобальний доступ до цього екземпляра. Основна мета цього паттерна - забезпечити однаковий доступ до об'єкта з будь-якої точки програми та уникнути множинного створення однакових об'єктів, що може призвести до непотрібних витрат ресурсів.

Щоб реалізувати Singleton, необхідно визначити приватний конструктор класу, який перешкоджає зовнішньому створенню екземплярів класу. Клас також має мати статичний метод або поле для отримання єдиного екземпляра. При цьому перший виклик методу чи поля створює новий екземпляр класу, а всі подальші виклики повертають вже існуючий екземпляр. [19]

Переваги використання паттерна Singleton включають уніфікований доступ до ресурсів, ефективне використання пам'яті та ресурсів, а також можливість управління глобальними об'єктами у програмі. Однак варто враховувати, що надмірне використання Singleton може призвести до збільшення залежностей та складнощів у тестуванні. Тому його слід використовувати розсудливо та у випадках, коли є справжня потреба у глобальному доступі до об'єкта.

4. Factory (Фабрика):

Використовується для створення об'єктів без вказання конкретного класу цих об'єктів. Це дозволяє динамічно вибирати тип створюваних об'єктів.

Паттерн Factory (Фабрика) використовується для створення об'єктів без прив'язки до конкретного класу цих об'єктів. Основна мета цього паттерна - дозволити динамічно вибирати тип створюваних об'єктів залежно від умов програми чи зовнішніх параметрів.

Ідея паттерна Factory полягає в тому, що існує абстрактний клас або інтерфейс, який визначає метод для створення об'єктів, а конкретні класи-фабрики реалізують цей метод та відповідають за створення конкретних об'єктів.

Таким чином, клієнтський код може викликати фабричний метод для отримання необхідного об'єкта без необхідності знати, який саме клас створюється.

Переваги використання паттерна Factory включають узагальнення процесу створення об'єктів, можливість зміни типу створюваних об'єктів без змін в клієнтському коді, а також спрощення тестування та управління ресурсами.

Наприклад, у мобільному додатку для створення різних типів користувачів (зареєстрованих, гостей тощо) може бути використаний паттерн Factory. Клас-фабрика може мати методи для створення різних типів користувачів на основі переданих параметрів чи умов програми. [20]

5. Observer (Спостерігач):

Цей паттерн використовується для реалізації механізму підписки-відписки, коли один об'єкт надсилає сповіщення про зміни свого стану іншим об'єктам, які підписані на нього.

Паттерн Observer (Спостерігач) використовується для реалізації механізму підписки-відписки, коли один об'єкт (відомий як суб'єкт або відправник) надсилає сповіщення про зміни свого стану іншим об'єктам (відомим як спостерігачі), які підписані на нього. Цей паттерн дозволяє забезпечити зв'язок між об'єктами, при цьому зменшуючи залежності між ними.

У паттерні Observer є два головних учасника: суб'єкт (Subject) і спостерігач (Observer). Суб'єкт володіє даними, до яких спостерігачі можуть підписатися, тобто цікавитися змінами в цих даних. Кожен спостерігач має метод, який викликається при отриманні сповіщення від суб'єкта про зміну стану. Саме ці методи спостерігачів використовуються для реагування на зміни даних в суб'єкті.

Переваги використання паттерна Observer полягають у тому, що він дозволяє створювати слабосполучені системи, де суб'єкти та спостерігачі можуть змінюватися незалежно один від одного. Цей підхід сприяє покращенню модульності програми та розділенню обов'язків між об'єктами.

Наприклад, у мобільному додатку паттерн Observer може бути використаний для оновлення інформації на екрані користувача при зміні даних в базі даних або відправленні повідомлень. Саме спостерігачі, які підписані на

відповідні події, будуть отримувати сповіщення та оновлювати відповідні елементи інтерфейсу. [21]

Ці патерни та інші допомагають покращити структуру та підтримуваність додатків, забезпечуючи ефективнішу розробку та зменшення кількості помилок. Кожен паттерн має свої особливості та використання, тому вибір паттерну залежить від конкретних потреб та завдань проекту.

1.3. Методики організації функціональності в FlutterFlow: управління станом, обробка подій та навігація

Методики організації функціональності в FlutterFlow включають управління станом, обробку подій та навігацію. Кожен з цих аспектів важливий для створення добре структурованого та ефективного мобільного додатку.

Управління станом є однією з ключових концепцій при розробці додатків. Стан визначає, які дані в даний момент відображаються на екрані та які дії можна виконати. У FlutterFlow існують різні підходи до управління станом, такі як робота зі станом компонентів, використання стейтфул віджетів або використання сторонніх бібліотек для керування глобальним станом додатку. Важливо правильно організувати стан для забезпечення ефективності та зручності розробки.

Стан компонентів: Один з підходів до управління станом в FlutterFlow - це робота зі станом окремих компонентів. Кожен віджет може мати свій власний стан, який визначає, які дані відображаються та які дії можна виконати в цьому конкретному віджеті. Це дозволяє більш точно контролювати відображення даних та взаємодію в окремих частинах додатку.

Управління станом компонентів у FlutterFlow є важливою стратегією, оскільки вона дозволяє розробникам більш точно контролювати відображення даних та взаємодію в окремих частинах додатку. Кожен віджет має свій власний стан, що означає, що зміни, які відбуваються в одному віджеті, не впливають на інші компоненти додатку.

Стейтфул віджети: У FlutterFlow використовуються стейтфул віджети, які зберігають свій власний стан та перерендерюються при зміні цього стану. Це дозволяє створювати динамічні інтерфейси, які реагують на зміни даних та події в додатку. Наприклад, при натисканні кнопки може змінюватися стан віджета, що призводить до зміни відображення на екрані.

Стейтфул віджети в FlutterFlow є потужним засобом для створення динамічних інтерфейсів, які реагують на зміни стану та події в додатку. Основна відмінність між стейтфул та стейтлесс віджетами полягає в тому, що стейтфул віджети можуть зберігати свій власний стан, який перерендерюється при зміні цього стану. [22]

Такий підхід дозволяє створювати динамічні інтерфейси та взаємодіювати з користувачем. Наприклад, ви можете використовувати стейтфул віджет для створення форми з полями введення, де дані автоматично оновлюються при введенні користувачем нової інформації. Це робить додаток більш відзивчивим та зручним у використанні.

Сторонні бібліотеки для стану: У деяких випадках може виникнути потреба у глобальному керуванні станом додатку. Для цього можна використовувати сторонні бібліотеки, такі як Provider або Bloc, які дозволяють ефективно керувати станом всього додатку та передавати його між різними віджетами та екранами.

В деяких сценаріях, особливо коли додаток стає складнішим і потребує глобального керування станом, може знадобитися використання сторонніх бібліотек для ефективного управління станом. Однією з таких бібліотек є Provider, яка забезпечує простий та потужний механізм для передачі та оновлення стану в додатку.

Використання сторонніх бібліотек для керування станом дозволяє підтримувати чистоту та організацію коду, спрощуючи розробку та підтримку додатку у майбутньому. Однак важливо обирати ту бібліотеку, яка найкраще відповідає потребам конкретного проекту та забезпечує необхідний рівень функціональності та гнучкості.

Ефективність та зручність: Правильне організування стану допомагає забезпечити ефективність та зручність розробки. Це означає, що розробник може легко контролювати стан та взаємодію в додатку, що в свою чергу сприяє швидкому виправленню помилок та вдосконаленню функціоналу.

Правильне управління станом в додатку дійсно має значний вплив на ефективність та зручність розробки. Коли стан організовано логічно та систематично, це спрощує процес розробки та управління кодом.

Крім того, зручність розробки також забезпечується гнучкістю управління станом. Коли стан додатку легко масштабувати та модифікувати, це дозволяє швидше реагувати на зміни вимог або додавати нові функції без значних зусиль.

Отже, ефективне управління станом в додатку є ключовим аспектом для досягнення високої продуктивності та зручності розробки. Його правильна організація сприяє швидкому розвитку, зменшенню часу на виправлення помилок та покращенню функціоналу додатку. [23]

Навігація визначає спосіб переміщення користувача між різними екранами та частинами додатку. У FlutterFlow навігацію можна організувати за допомогою різних методів, таких як стекова навігація, табова навігація, бокове меню та інші. Важливо вибрати той метод навігації, який найкраще відповідає функціональності та потребам додатку.

Навігація в додатку - це не лише спосіб переміщення користувача між екранами, але й ключовий елемент для створення зручного та інтуїтивного досвіду використання додатку. У FlutterFlow вибір методу навігації залежить від конкретної функціональності додатку та його інтерфейсу.

Один з найпоширеніших методів навігації - це стекова навігація. У цьому методі, кожен екран, який відкривається, додається до стеку, і користувач може повернутися назад до попереднього екрану, натиснувши кнопку "Назад". Цей підхід добре підходить для ієрархічних структур додатків, де користувач переходить від одного рівня до іншого.

Бокове меню (Drawer або Sidebar) - це ще один популярний спосіб навігації, де головне меню знаходиться збоку екрану і відкривається або зсувається, коли

користувач торкнеться або проведе пальцем по екрану. Це дозволяє швидко доступатися до основних функцій додатку, навіть якщо він має велику кількість розділів.

Навігація має бути логічною, інтуїтивно зрозумілою та допомагати користувачеві швидко знаходити необхідну інформацію чи функцію.

РОЗДІЛ 2
ІНТЕГРАЦІЯ FIREBASE ДЛЯ МОБІЛЬНИХ ДОДАТКІВ: ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНАЛЬНОСТІ ТА БЕЗПЕКИ

2.1 Аналіз можливостей Firebase для мобільних додатків: база даних, аутентифікація, зберігання файлів

Firebase - це платформа від Google, яка надає різноманітні інструменти для розробки мобільних додатків, включаючи базу даних, аутентифікацію користувачів та зберігання файлів. Розглянемо кожен з цих аспектів детальніше.

База даних Firebase - це один з ключових компонентів платформи, який дозволяє зберігати та управляти даними додатку в хмарі. Firebase пропонує дві основні бази даних: Realtime Database та Cloud Firestore. Realtime Database - це NoSQL база даних, яка працює в режимі реального часу, дозволяючи миттєво синхронізувати дані між користувачами. [24]

Realtime Database в Firebase - це NoSQL база даних, яка працює в режимі реального часу. Основна особливість цієї бази даних полягає в тому, що вона надає миттєву синхронізацію даних між користувачами. Це означає, що будь-яка зміна, яку ви вносите до бази даних, негайно відображається на всіх пристроях, які підключені до цієї бази даних. Це особливо корисно для розробки додатків, які потребують миттєвої реакції на зміни даних, наприклад, чати, реально-часні оновлення тощо

Cloud Firestore - це розширена версія бази даних, яка надає більше можливостей управління даними, такі як підтримка пакетних операцій та більш гнучкий запит даних. Cloud Firestore, з іншого боку, є більш розширеною версією бази даних Firebase. Вона також працює в реальному часі, але надає додаткові можливості управління даними. Наприклад, Cloud Firestore підтримує більш гнучкі операції читання та запису даних, а також підтримує транзакції та пакетні операції. Ви також можете використовувати багаторівневі запити для вибору даних, що дозволяє ефективно працювати з великими обсягами інформації.

Обидві бази даних в Firebase добре інтегруються з іншими сервісами платформи, такими як аутентифікація користувачів, зберігання файлів, аналітика тощо. Це робить Firebase потужним інструментом для розробки мобільних додатків, які потребують надійного та ефективного управління даними.

Аутентифікація користувачів - ще один важливий аспект Firebase для мобільних додатків. Firebase Authentication надає різні методи аутентифікації, такі як електронна пошта та пароль, номер телефону, соціальні мережі (Google, Facebook, Twitter, GitHub) та інші. Це дозволяє зручно та безпечно впроваджувати механізми входу та реєстрації користувачів у додаток.

Firebase Authentication - це інструмент в Firebase, який дозволяє забезпечити безпеку та зручність аутентифікації користувачів у мобільних додатках. Однією з основних переваг цієї функції є різноманітні методи аутентифікації, які дозволяють користувачам входити в додаток з використанням різних ідентифікаційних даних. [25]

Наприклад, метод електронної пошти та пароля дозволяє користувачам створювати акаунт за допомогою своєї електронної адреси та обраного пароля. Це є стандартним методом аутентифікації, який використовується в багатьох додатках.

Крім того, Firebase Authentication підтримує аутентифікацію через соціальні мережі, такі як Google, Facebook, Twitter, GitHub та інші. Це дозволяє користувачам входити в додаток за допомогою свого існуючого облікового запису на популярних платформах, що забезпечує зручність та швидкість входу.

Однією з переваг Firebase Authentication є його безпека. Всі дані передаються зашифровано, що забезпечує конфіденційність інформації користувачів під час аутентифікації.

Зберігання файлів - Firebase також надає можливості для зберігання та управління файлами в хмарі. Firebase Storage дозволяє зберігати зображення, відео, аудіо та інші типи файлів у безпечному та надійному хмарному сховищі. Це дозволяє зменшити навантаження на сервер додатку та забезпечує легкий доступ до файлів для користувачів з будь-якого пристрою.

Firebase Storage це частина платформи Firebase, яка забезпечує зберігання та управління різними типами файлів в хмарі. Це включає зображення, відео, аудіо та інші медіафайли. Одна з головних переваг Firebase Storage - це безпечність і надійність зберігання. Дані зберігаються в захищеному хмарному середовищі, що дозволяє уникнути втрати файлів чи несанкціонованого доступу до них.

2.2 Засоби забезпечення безпеки даних в мобільних додатках з використанням Firebase

Забезпечення безпеки даних є критичним завданням для мобільних додатків, особливо з огляду на зростаючі вимоги до конфіденційності та захисту особистих даних користувачів. Firebase надає різні засоби та механізми для забезпечення безпеки даних у мобільних додатках.

1. Шифрування даних: Firebase дозволяє шифрувати дані, які зберігаються в базі даних та хмарному сховищі. Шифрування даних забезпечує захист від несанкціонованого доступу та забезпечує конфіденційність даних користувачів. [26]

Шифрування даних у Firebase є важливим механізмом захисту конфіденційності та безпеки інформації користувачів. Цей процес полягає в перетворенні звичайного тексту (даних) у незрозумілу форму (шифр), яка доступна лише за допомогою спеціального ключа, що розшифровує дані назад у звичайний вигляд. Основна мета шифрування - запобігти несанкціонованому доступу до даних та збереження конфіденційності інформації.

Шифрування даних є важливим елементом забезпечення безпеки і приватності у мобільних додатках. Воно гарантує, що навіть у випадку можливого витоку даних, вони залишаються захищеними та непридатними для несанкціонованого використання.

2. Аутентифікація та авторизація: Firebase Authentication дозволяє впроваджувати різні механізми аутентифікації, включаючи двофакторну

аутентифікацію, а також забезпечує контроль доступу до функцій додатка за допомогою механізмів авторизації.

Firebase Authentication - це сервіс Firebase, який забезпечує механізми аутентифікації користувачів у мобільних додатках. Основною метою аутентифікації є підтвердження ідентичності користувача та надання доступу до функціоналу додатку лише авторизованим особам. Firebase Authentication підтримує різні методи аутентифікації, що включають:

- Електронна пошта та пароль: Користувачі можуть створювати облікові записи за допомогою електронної пошти та паролю. Цей метод є одним з найпоширеніших у веб-додатках і мобільних додатках.
- Номер телефону: Користувачі можуть підтверджувати свою ідентичність за допомогою SMS-коду, надісланого на їх мобільний телефон. Цей метод є зручним та швидким для користувачів, оскільки вони можуть уникнути введення складних паролів.
- Соціальні мережі: Firebase підтримує автентифікацію через різні соціальні мережі, такі як Google, Facebook, Twitter, GitHub та інші. Користувачі можуть увійти до додатку, використовуючи свої існуючі облікові записи в цих мережах.
- Анонімна аутентифікація: Firebase також дозволяє анонімну аутентифікацію, коли користувачі можуть використовувати додаток без створення облікового запису. Це може бути корисним для тимчасового доступу або анонімного використання функціоналу.

Після успішної аутентифікації користувача Firebase Authentication надає можливість реалізувати механізми авторизації, які визначають права доступу користувача до конкретних функцій та даних додатку. Це дозволяє контролювати безпеку додатку та забезпечити лише авторизованим користувачам доступ до привілейованого функціоналу.

3. Моніторинг безпеки: Firebase має вбудовані інструменти моніторингу безпеки, які дозволяють виявляти та вирішувати потенційні загрози

безпеці даних, такі як небезпечні активності, спроби несанкціонованого доступу та інші вразливості.

Система моніторингу безпеки в Firebase дозволяє автоматично виявляти та аналізувати потенційні вразливості, такі як атаки типу Cross-Site Scripting (XSS), витоки даних, недостатньо захищені API, а також небезпечні активності користувачів, які можуть вказувати на спроби несанкціонованого доступу.

За допомогою моніторингу безпеки в Firebase можна отримати інформацію про потенційні загрози та вразливості, що дозволяє оперативно реагувати та вживати заходів для їх вирішення. Такий підхід допомагає забезпечити високий рівень безпеки для додатків та даних користувачів, що є надзвичайно важливим у сучасному цифровому середовищі. [27]

4. Бекапи та відновлення: Firebase забезпечує можливість регулярних бекапів даних, що дозволяє відновлювати дані в разі втрати або пошкодження інформації.

Система бекапів та відновлення в Firebase гарантує безпеку ваших даних шляхом регулярних створень копій інформації. Цей процес дозволяє вам відновлювати дані у випадку їх втрати або пошкодження внаслідок непередбачених ситуацій, таких як технічні проблеми, кібератаки або випадкове видалення.

Бекапи даних - це копії важливої інформації, які забезпечують збереження даних навіть у випадку їх втрати або пошкодження. У Firebase бекапи даних відбуваються автоматично з регулярними інтервалами, щоб гарантувати, що ви маєте найсвіжішу копію даних у випадку потреби відновлення.

Цей процес бекапів може бути налаштований згідно з вашими потребами та вимогами безпеки. Наприклад, ви можете визначити, як часто створювати бекапи, де зберігати ці копії, і чи потрібно вам зберігати історію раніше створених бекапів.

Однією з важливих переваг системи бекапів Firebase є те, що цей процес автоматизований і інтегрований безпосередньо з платформою. Це означає, що ви

можете не турбуватися про ручне створення бекапів та відновлення даних, адже це вже передбачено та працює надійно.

Додатково, бекапи в Firebase можуть забезпечувати не лише збереження даних, але і можуть бути корисними для відладки та аналізу даних. Ви можете використовувати ці бекапи для відновлення інформації на окремому тестовому середовищі або для аналізу змін в даних протягом часу.

У підсумку, система бекапів та відновлення в Firebase забезпечує вам спокійність та впевненість у безпеці ваших даних, а також надійність та гнучкість управління процесом бекапів та відновлення. Проведення регулярних бекапів важливе для забезпечення безпеки даних та надійності вашого додатку. Завдяки системі бекапів Firebase ви можете мати певність, що ваші дані будуть захищені і готові до відновлення у випадку будь-яких непередбачених обставин.

5. SSL шифрування: Firebase використовує SSL шифрування для захисту передачі даних між додатком та сервером, забезпечуючи безпеку під час комунікації.

SSL шифрування (Secure Sockets Layer) є важливим механізмом для забезпечення безпеки під час передачі даних через мережу, і Firebase активно використовує цей протокол для захисту комунікації між додатком та сервером. Давайте розглянемо, як SSL шифрування працює в контексті Firebase. [28]

Коли ваш мобільний додаток взаємодіє з Firebase, дані, що передаються між додатком та сервером Firebase, захищені SSL шифруванням. Це означає, що дані шифруються перед відправкою з додатку і розшифровуються при прийомі на сервері, забезпечуючи конфіденційність та цілісність даних під час їх передачі.

Основні переваги SSL шифрування включають:

1. Конфіденційність даних: Інформація, яка передається між додатком і сервером Firebase, залишається конфіденційною та захищеною від небажаних переглядів третіх осіб.

2. Цілісність даних: SSL шифрування дозволяє перевірити, чи були дані змінені або підроблені під час передачі, що допомагає уникнути маніпуляцій з даними під час транспорту.

3. Достовірність сервера: SSL сертифікати дозволяють перевіряти автентичність сервера, з яким взаємодіє додаток, що запобігає можливості атак типу "людина посередництвом" (man-in-the-middle).

Загалом, використання SSL шифрування в Firebase забезпечує високий рівень безпеки під час передачі даних, що є важливим аспектом для захисту конфіденційності та цілісності інформації в мобільних додатках.

6. Доступ до аудиту: Firebase надає можливість перегляду та моніторингу логів доступу до даних, що дозволяє виявляти та відслідковувати можливі порушення безпеки.

Firebase надає зручний інтерфейс для перегляду та моніторингу логів доступу до даних, що є важливим аспектом з точки зору забезпечення безпеки. Цей інструмент дозволяє адміністраторам та розробникам виявляти та відслідковувати різноманітні можливі порушення безпеки в додатках.

Крім того, логи дозволяють встановлювати права доступу до даних та функцій додатка. Вони дають можливість перевіряти, які користувачі мають доступ до конкретних даних, що є важливим для контролю прав доступу та запобігання несанкціонованим діям. [29]

Іншим важливим аспектом є моніторинг активності користувачів. Інструменти аудиту в Firebase дозволяють виявляти та аналізувати активність користувачів, включаючи зміни даних, вхід в систему та інші важливі події. Це допомагає вчасно виявляти можливі загрози та аномалії, що дозволяє приймати необхідні заходи для забезпечення безпеки даних.

Таким чином, інструменти моніторингу та аудиту в Firebase грають ключову роль у забезпеченні безпеки даних у мобільних додатках, допомагаючи виявляти, відстежувати та реагувати на потенційні загрози безпеки.

Ці засоби забезпечують комплексний підхід до безпеки даних у мобільних додатках, що дозволяє розробникам забезпечити високий рівень захисту для користувачів та їх особистих даних.

Аутентифікація та авторизація, які реалізовані у Firebase Authentication, гарантують, що тільки правомірні користувачі матимуть доступ до додатка та

його функціональності. Використання двофакторної аутентифікації та інших методів підтвердження особи значно ускладнює доступ для потенційних зловмисників, зменшуючи ризик несанкціонованого доступу.

Моніторинг безпеки та інструменти аудиту Firebase дозволяють розробникам ефективно виявляти, відстежувати та реагувати на потенційні загрози безпеки даних. Шляхом аналізу логів доступу та моніторингу активності користувачів можна оперативно виявляти аномалії та ризикові ситуації, що дозволяє вчасно реагувати та захищати дані від можливих атак.

Нарешті, засоби бекапів та відновлення у Firebase дозволяють забезпечити безпеку даних у випадку втрати чи пошкодження інформації. Регулярні бекапи та можливість швидкого відновлення даних дозволяють мінімізувати втрати та забезпечувати безперервну роботу додатків.

Усі ці заходи разом створюють надійний та ефективний механізм захисту даних у мобільних додатках, що дозволяє користувачам відчувати впевненість у безпеці своїх особистих даних під час користування додатками, розробленими на базі Firebase.

2.3 Вплив інтеграції Firebase на функціональність та продуктивність мобільного додатку

Firebase дозволяє миттєво синхронізувати дані між користувачами та додатками завдяки Realtime Database та Cloud Firestore. Це покращує функціональність додатка, дозволяючи користувачам бачити зміни в реальному часі та ефективно спілкуватися. [29]

Реально-часна синхронізація даних, яку надає Firebase через Realtime Database та Cloud Firestore, має значний вплив на функціональність мобільного додатка. Перше, що варто відзначити, це спрощення спілкування та спільної роботи між користувачами, оскільки зміни в даних негайно відображаються для всіх учасників системи. Це робить взаємодію більш надійною та зручною, зменшуючи можливість конфліктів та неузгодженостей у даних.

Крім того, реально-часна синхронізація дозволяє створювати додатки з динамічним вмістом, такими як чати, онлайн-ігри, колективна робота над документами тощо. Користувачі можуть бачити зміни в даних миттєво, що створює більш іммерсивне та захоплююче досвід використання додатка.

Додатково, завдяки реально-часній синхронізації, розробники можуть легко і ефективно реалізувати специфічні функції, такі як сповіщення користувачів про зміни, автоматичне оновлення даних, підтримка реактивного інтерфейсу тощо. Це відкриває широкі можливості для створення інноваційних та високофункціональних додатків.

За допомогою реально-часної синхронізації в Firebase, розробники можуть також забезпечити безпеку даних під час обміну ними між користувачами та сервером. Технологія Realtime Database та Cloud Firestore використовує захищене з'єднання, що гарантує конфіденційність та цілісність інформації під час передачі через мережу. Такий підхід сприяє високому рівню безпеки та довіри у використанні додатку. [30]

Firebase Authentication надає швидкий та безпечний механізм аутентифікації користувачів через електронну пошту, номер телефону, а також соціальні мережі. Це поліпшує функціональність додатка, дозволяючи користувачам легко та безпечно увійти до системи.

Швидка аутентифікація користувачів через Firebase Authentication дійсно має значний вплив на функціональність та продуктивність мобільного додатка. Перше, що слід відзначити, це зменшення часу, необхідного для користувачів для входу в додаток. Замість довгих форм реєстрації чи вводу облікових даних, вони можуть скористатися швидкими методами аутентифікації, такими як електронна пошта, номер телефону чи профілі у соціальних мережах. Це покращує користувацький досвід та збільшує імовірність того, що користувачі залишаться в додатку і використовуватимуть його регулярно.

Додатково, швидка аутентифікація користувачів дозволяє забезпечити безпеку даних та облікових записів. Firebase Authentication використовує потужні механізми шифрування та аутентифікації, що знижує ризик несанкціонованого

доступу та забезпечує конфіденційність особистої інформації користувачів. Це сприяє підвищенню рівня довіри до додатку серед користувачів і допомагає залучати нових клієнтів.

Також важливою перевагою швидкої аутентифікації є підтримка багатоплатформеності. Firebase Authentication дозволяє користувачам входити в додаток через різні пристрої та платформи без необхідності повторного введення даних. Це зроблює досвід використання додатка більш безпечним, зручним та універсальним для користувачів.

Узагальнюючи, швидка аутентифікація користувачів через Firebase Authentication покращує функціональність додатка, забезпечує безпеку та зручність для користувачів, а також сприяє збільшенню активності і задоволеності аудиторії. [31]

Firebase Storage дозволяє ефективно зберігати та управляти файлами, такими як зображення, відео, аудіо тощо. Це покращує функціональність додатка, забезпечуючи користувачам легкий доступ до мультимедійних ресурсів.

Ефективне зберігання та управління файлами через Firebase Storage є важливим аспектом для функціональності та продуктивності мобільного додатка. Однією з основних переваг є можливість зберігання різноманітних типів файлів, таких як зображення, відео, аудіо, документи тощо, в безпечному та надійному хмарному сховищі. Це дозволяє ефективно управляти мультимедійним контентом додатка та забезпечує легкий доступ до цих файлів для користувачів з будь-якого пристрою.

Крім того, Firebase Storage надає швидкий доступ до файлів завдяки оптимізації процесів зберігання та передачі даних. Це допомагає підвищити продуктивність додатка, зменшити час завантаження великих мультимедійних файлів та поліпшити користувацький досвід.

Окрім цього, Firebase Storage вбудований у Firebase Console, що дозволяє зручно керувати та моніторити завантажені файли, забезпечуючи адміністраторам та розробникам легкий доступ до управління файловими ресурсами додатка. Це сприяє підтримці та оптимізації файлової інфраструктури

додатка, що є важливим для забезпечення його функціональності та продуктивності.

Firebase забезпечує шифрування даних під час передачі за допомогою SSL, а також забезпечує контроль доступу до даних через механізми авторизації. Це покращує функціональність додатка, забезпечуючи захист від несанкціонованого доступу до інформації користувачів.

Підвищена безпека даних завдяки інтеграції з Firebase є ключовим фактором для забезпечення функціональності та продуктивності мобільного додатка. Однією з головних переваг є використання SSL шифрування для захисту передачі даних між додатком та сервером Firebase. Це гарантує конфіденційність та безпеку інформації під час її передачі через мережу, що є критичним аспектом для будь-якого додатка, особливо якщо він обробляє чутливі дані користувачів, такі як особисті дані, фінансова інформація тощо.

Також, Firebase надає механізми авторизації та контролю доступу, що дозволяють обмежувати доступ до різних частин додатка для різних користувачів. Це включає в себе можливість встановлення рівнів доступу, визначення прав користувачів, а також використання двофакторної аутентифікації для підвищення рівня безпеки облікових записів.

Такий підхід до безпеки даних допомагає забезпечити захист від несанкціонованого доступу до інформації, уникнення втрати чутливих даних та забезпечення довіри користувачів до додатка. Це в свою чергу сприяє поліпшенню функціональності та продуктивності додатка, оскільки користувачі мають можливість використовувати його з впевненістю в безпеці своїх особистих даних. [32]

Firebase надає інструменти моніторингу безпеки, які дозволяють виявляти та вирішувати потенційні загрози безпеці даних. Це покращує функціональність додатка, забезпечуючи безпеку та конфіденційність даних користувачів.

Моніторинг та аналіз безпеки, доступний завдяки інтеграції з Firebase, є важливою складовою для забезпечення функціональності та продуктивності мобільного додатка. За допомогою вбудованих інструментів моніторингу безпеки

Firebase, розробники мають змогу виявляти та вирішувати потенційні загрози безпеці даних в реальному часі. Це включає в себе виявлення небезпечних активностей, спроб несанкціонованого доступу та інших потенційних вразливостей, які можуть погрожувати безпеці користувачів та їх даних.

Інструменти моніторингу безпеки Firebase надають можливість в реальному часі відслідковувати події та дії в додатку, що дозволяє оперативно реагувати на потенційні загрози та вживати необхідні заходи для їх вирішення. Це покращує функціональність додатка, забезпечуючи безпеку та конфіденційність даних користувачів, а також сприяє підвищенню рівня довіри користувачів до додатку.

Ще, можливість аналізувати та моніторити безпеку даних дозволяє постійно вдосконалювати стратегії безпеки та покращувати архітектуру додатка з метою забезпечення найвищого рівня захисту.

Так, вплив інтеграції Firebase на функціональність та продуктивність мобільного додатку суттєвий і багатогранний. В першу чергу, реально-часна синхронізація даних дозволяє користувачам бачити зміни в реальному часі та ефективно спілкуватися, що покращує їхній досвід взаємодії з додатком. Швидка аутентифікація користувачів робить процес увіходу до системи швидким і безпечним, що позитивно впливає на юзабіліті додатку та збільшує його привабливість для користувачів.

Ефективне зберігання та управління файлами, разом з підвищеною безпекою даних через шифрування та контроль доступу, гарантує надійність та конфіденційність інформації. Моніторинг та аналіз безпеки забезпечують постійний контроль за можливими загрозами, що допомагає зберігати безпеку даних на високому рівні.

Усі ці аспекти роблять мобільний додаток, інтегрований з Firebase, більш ефективним, безпечним та зручним для користувачів. Це покращує якість обслуговування та сприяє збільшенню задоволення користувачів від використання додатку.

РОЗДІЛ 3
РОЗРОБКА ДОДАТКУ У FLUTTERFLOW: БІЗНЕС АНАЛІЗ ТА ІНТЕГРАЦІЯ З FIREBASE

3.1 Аналіз предметної області та визначення функціональних вимог мобільного додатку

Почнемо з вимог до додатку райдшерингу.

Реєстрація користувачів: Додаток повинен надавати користувачам можливість швидко та безпечно реєструватися та входити в систему. Це може бути через електронну пошту, номер телефону або соціальні мережі.

Оплата: Додаток повинен підтримувати різні методи оплати: картки, мобільні гаманці, готівка. Платежі мають бути безпечними та шифруватися для захисту фінансових даних користувачів. Користувачі мають доступ до історії своїх платежів у додатку.

Відгуки та оцінки: Користувачі повинні мати можливість залишати відгуки про свої поїздки, водіїв і пасажирів. Користувачі можуть оцінювати інших користувачів, що дозволяє створити систему рейтингу для підвищення якості послуг. Водії та пасажири можуть переглядати свої відгуки, що допомагає їм покращувати свій досвід. [33]

Далі поговоримо про технічні аспекти.

FlutterFlow є візуальним конструктором додатків на базі Flutter, що дозволяє розробникам створювати мобільні додатки швидко та інтуїтивно. Завдяки функціям перетягування та візуальним блокам коду розробка додатку стає швидшою та простішою. Додаток, створений за допомогою FlutterFlow, може працювати на обох платформах (iOS та Android) без додаткової роботи. Необхідно вибрати базу даних, яка забезпечить швидкість та ефективність обробки даних (наприклад, Firebase). Зберігання даних: База даних має забезпечити безпечне зберігання інформації про користувачів, поїздки, маршрути та інші дані.

Також торкнемось розробка інтерфейсу та потенційних проблем та ризиків. Використання FlutterFlow для створення інтуїтивного дизайну інтерфейсу, що забезпечує зручність користувачів. Інтерфейс має відповідати стилю бренду додатку та бути впізнаваним. Інтерфейс має швидко реагувати на дії користувачів, що забезпечує якісний користувацький досвід. Інтеграція платіжної системи повинна забезпечити безперебійний та безпечний процес оплати. Впровадження стандартів безпеки для захисту фінансових даних користувачів.

Необхідно забезпечити безпеку особистої інформації користувачів, включаючи їхні імена, електронні адреси, номери телефонів та платіжні дані. Використання сучасних методів шифрування для передачі та зберігання даних, щоб забезпечити їхню конфіденційність та захист від несанкціонованого доступу. [34]

Обмеження доступу до даних тільки тим особам та системам, які його потребують, а також регулярне оновлення паролів та облікових даних.

Дотримання норм захисту даних, таких як GDPR або інші місцеві закони про конфіденційність даних. Додаток має бути здатним відстежувати зміни в маршрутах і оновлювати їх у режимі реального часу для забезпечення актуальності даних. Забезпечення швидкої обробки та оновлення даних щодо маршрутів, пунктів посадки та висадки, і часу прибуття. Додаток повинен пропонувати унікальні або вдосконалені функції, щоб виділитися на фоні конкурентів і привабити більше користувачів. Забезпечення зручності використання та високого рівня обслуговування для користувачів, що підвищує їхню лояльність.

Наступним поговоримо про функціональні вимоги:

Реєстрація нових користувачів: Користувачі (пасажири та водії) мають можливість обрати свою роль (пасажир або водій) під час реєстрації в додатку. Користувачі повинні вказати основну інформацію, таку як ім'я, прізвище, електронну адресу, пароль.

Вхід у систему для зареєстрованих користувачів: Пасажир та водій вводять свою електронну адресу разом з паролем для входу в систему. Система проводить аутентифікацію, перевіряючи введені дані користувача з інформацією, що зберігається в базі даних користувачів. Якщо введені дані відповідають запису в базі даних, користувач має доступ до свого облікового запису та функціоналу додатку.

Редагування профілю користувача: Користувач має мати можливість редагувати свій профіль в додатку. Ця функція включає наступні можливості: Користувач може завантажити нове фото для свого профілю через інтерфейс додатку. [35]

Перегляд інформації про водія та поїздку: Пасажир має можливість переглядати повну інформацію про водія, включаючи його ім'я, фото, рейтинг та відгуки інших користувачів. Пасажир отримує деталі про дату та час відправлення, напрямок поїздки, вартість та інші умови користування.

Контакт з водієм: Пасажир має можливість зв'язатися з водієм за допомогою номера телефону, який вказаний у додатку.

Залишення відгуку: Після завершення поїздки пасажир може залишити відгук про водія та оцінити його роботу. Це допомагає іншим користувачам зробити кращий вибір при виборі водія для майбутніх поїздок.

Перегляд рейтингу та відгуків: Пасажир може переглядати рейтинг водія та читати відгуки інших користувачів для оцінки роботи водія та комфорту поїздки.

Підтримка користувачів: Пасажир може звернутися до служби підтримки для вирішення будь-яких питань або проблем, пов'язаних з поїздкою.

Додавання інформації про поїздку: Водій має можливість вказати дату, час відправлення та прибуття, точки початку і кінця маршруту, вартість подорожі, та інші деталі, щоб пасажири могли бачити повну інформацію перед бронюванням.

Введення особистої інформації: Водій може ввести своє ім'я, контактний номер телефону та інші особисті дані, які допоможуть пасажирам зв'язатися з ним та впізнати його як водія.

Завантаження фотографії: Водій може завантажити своє фото, щоб пасажири мали можливість побачити його обличчя та були впевнені, з ким вони їдуть. [36]

Вказання ціни за поїздку: Водій має змогу встановити ціну за поїздку або скористатися системою автоматичного розрахунку ціни відповідно до встановлених параметрів та умов.

Взаємодія з пасажирами: Водій може залишати контактний номер телефону для швидкого зв'язку з пасажирами, вирішення питань по маршруту або для надання додаткової інформації про подорож.

Далі поговоримо про створення use-case діаграм. Use case diagram є одним з ключових інструментів в аналізі та проектуванні програмного забезпечення. Це графічний засіб для моделювання функціональних вимог до системи, який дозволяє показати, як користувачі (актори) взаємодіють з системою для виконання певних дій (use cases). Основні складові use case diagram:

Актори (Actors): Актори представляють ролі, які взаємодіють з системою. Це можуть бути користувачі, зовнішні системи або інші системи, що взаємодіють з системою.

Use Cases (Дії): Use cases описують функціональні можливості системи або дії, які можуть виконувати актори в системі.

Взаємозв'язки між акторами та діями: Стрілки вказують взаємозв'язки між акторами та use cases, показуючи, як актори взаємодіють з системою через певні дії.

Включення (Include) та Розширення (Extend):

Include показує, як одна дія включає в себе іншу дію.

Extend показує, як одна дія може розширювати іншу дію за певних умов.

Система: Це прямокутник, що відображає межі системи, в якій відбуваються всі дії та взаємодії. [37]

Use cases diagram допомагає розуміти функціональні вимоги до системи, ідентифікувати основні функції та взаємозв'язки між ними, а також визначати, як користувачі взаємодіють з системою для досягнення своїх цілей.

Основні кроки розробки Use cases для мобільного застосунку райдшерингу:

Почнемо з ідентифікації ролей або акторів, які будуть взаємодіяти з системою. Спочатку я визначив акторів як користувача, пасажирів, водіїв. Це допомогло мені зрозуміти, хто буде користуватися моєю системою та які функції вони будуть мати. [38]

Потім визначивчши основні можливості кожного актора або use cases. Наприклад, було розроблено use cases для користувачів "Авторизація/Реєстрація", "Редагування профілю", "Додавання номеру телефону", "Додавання дати народження". Для пасажирів, такі як "Пошук поїздки", "Зв’язок з службою підтримки", "Перегляд доступних поїздок", "Перегляд деталей поїздки", "Оплата поїздки", "Оцінка водія", "Зв’язок з водієм", "Зв’язок з службою підтримки". Для водіїв я створив "Перегляд поїздки", "Створення поїздки", "Контакт з підтримкою", "Взаємодія з пасажирами".

Далі я показав взаємозв'язки між акторами та відповідними діями на моїй діаграмі. Наприклад, показав, що водій створює поїздку. Пасажир може замовити цю поїздку.

Далі я визначив включення, які стосуються користувача, а саме: “Редагування профілю” адже коли користувач редагує профіль він має змогу додавати або змінювати своє фото, додавати номер телефону, додавати дату народження. Та розширення по відношенню до пасажира, а саме: “Перегляд деталей поїздок” це буде відбуватись коли пасажир буде переглядати доступні поїздки він може натиснути та переглянути деталі.

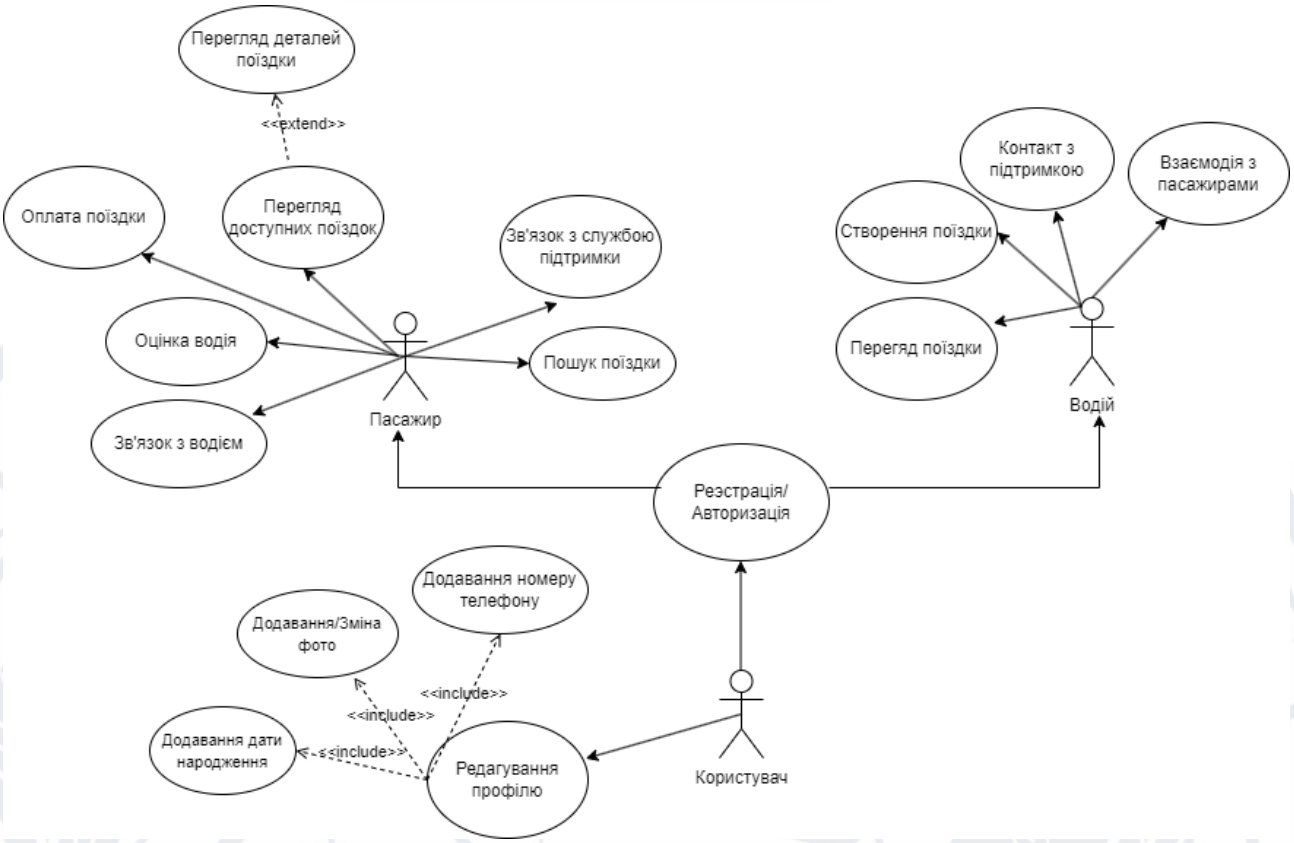


Рисунок 3.1 – Діаграма використання в додатку райдшерингу

3.2 Інтеграція з Firebase

Підключення Firebase до FlutterFlow передбачає інтеграцію популярної платформи розробки мобільних додатків з можливостями хмарного сервісу Firebase. Це надає розробникам багато переваг, таких як доступ до баз даних у реальному часі, аутентифікації користувачів, хмарних функцій та багато іншого.

Перш за все, вам потрібно зайти у Firebase Console та увійти в свій обліковий запис Google. У консолі ви повинні створити новий проект, обравши кнопку "Створити проект" або "Додати проект". Це дозволить вам дати своєму проекту унікальну назву та налаштувати його відповідно до ваших потреб. Додатково ви можете налаштувати проект під свій смак, наприклад, обравши, чи хочете ви використовувати Google Analytics для моніторингу та аналізу активності вашого додатку. [39]

Після створення нового проекту ви можете додати мобільний додаток до свого проекту у Firebase. Це можуть бути додатки для iOS та Android. Вам

потрібно ввести важливі дані про додаток, наприклад, ім'я пакету для Android або пакетний ідентифікатор для iOS. Після додавання додатку, вам буде надано конфігураційні файли, такі як google-services.json для Android та GoogleService-Info.plist для iOS, які необхідно завантажити.

У FlutterFlow, вам потрібно відкрити свій проєкт та перейти до розділу "Settings" або "Налаштування", де ви зможете налаштувати різні аспекти вашого проєкту.

У розділі "Firebase Integration" або подібному, ви повинні додати конфігураційні файли, які ви отримали з Firebase Console (google-services.json та GoogleService-Info.plist). Це допоможе налаштувати інтеграцію вашого додатку з Firebase.

Використовуючи можливості FlutterFlow, ви можете працювати з базами даних у Firebase, такими як Firestore. Це дає вам можливість створювати, читати, оновлювати та видаляти дані з бази даних безпосередньо в середовищі FlutterFlow.

Інтеграція з Firebase забезпечує швидке налаштування функцій аутентифікації користувачів, наприклад, реєстрацію, вхід та вихід користувачів. Ви також можете додати функції для відновлення паролю, щоб підвищити зручність для користувачів. [40]

Firebase дозволяє використовувати хмарні функції для виконання різних дій на сервері, наприклад, обробки даних. Це може бути корисно для реалізації складніших функцій вашого додатку, які потребують потужностей сервера.

Firestore організований у вигляді колекцій та документів. Колекції — це основні групи, що об'єднують пов'язані документи. Вони дозволяють структурувати дані в логічні блоки. Наприклад, ви можете створити колекцію "Користувачі" для зберігання документів з інформацією про користувачів.

Документи — це окремі записи в колекціях, які містять різні поля з даними. Кожен документ має унікальний ідентифікатор, що дозволяє легко отримувати доступ до нього або здійснювати маніпуляції з даними.

Колекції та документи у Firestore можуть бути організовані в складну ієрархічну структуру, яка дозволяє мати підколекції всередині документів. Це забезпечує гнучкість у структурі даних, дозволяючи вам зберігати пов'язані дані разом у межах однієї колекції або документа. Ця можливість організації даних дозволяє вам легко моделювати складні взаємозв'язки між даними, наприклад, взаємозв'язок між користувачами та їх замовленнями, зберігаючи їх у відповідних колекціях і підколекціях. Це також сприяє ефективній роботі з даними, оскільки дозволяє вам гнучко керувати доступом до різних рівнів даних і здійснювати запити на дані в різних контекстах.

Робота з Firestore у FlutterFlow починається зі створення колекцій та документів, а потім переходить до читання даних з бази даних за допомогою запитів. Цей процес є основою роботи з даними у Firestore та дозволяє ефективно керувати даними у вашому додатку.

У Firestore ви можете створювати колекції, які є основними контейнерами для схожих документів. Колекції допомагають організувати дані у зручну структуру, наприклад, створення колекції "Користувачі" для зберігання даних про користувачів. Це надає можливість структурувати базу даних відповідно до потреб додатку, що дозволяє легше працювати з даними та підтримувати їх в актуальному стані. Кожна колекція у Firestore може містити документи, що зберігають різні дані.

Кожен документ має унікальний ідентифікатор, що дозволяє легко знайти його або виконати операції з ним. Документи складаються з полів з даними, які можуть бути різних типів: рядки, числа, масиви, об'єкти, дати тощо. Ви можете додавати документи до колекцій за допомогою різних інструментів у FlutterFlow, що дозволяють зберігати необхідну інформацію у базі даних.

Читання даних з Firestore у FlutterFlow виконується за допомогою запитів або звернення до конкретних документів чи колекцій. Це дозволяє отримати дані для їх подальшого використання у додатку, наприклад, для відображення у користувацькому інтерфейсі. Запити у Firestore дозволяють отримати вибірку документів, що відповідають певним критеріям, наприклад, пошук документів у

колекції за певним полем або значенням. Це дуже корисно для пошуку та фільтрації даних, щоб отримати конкретну інформацію з бази даних.

Після отримання даних з Firestore, ви можете обробляти їх у вашому додатку відповідно до своїх потреб. Це може включати відображення даних у користувацькому інтерфейсі, їх перетворення або використання для прийняття рішень у додатку.

Обробка даних також може включати маніпулювання ними, наприклад, обчислення, фільтрацію або сортування.

Окрім цього, Firestore забезпечує можливість синхронізації даних у реальному часі, що дозволяє додатку залишатися актуальним і реагувати на зміни миттєво.

Розробка бази даних та таблиць у Firestore є важливим етапом проєктування додатку, оскільки структура бази даних значно впливає на ефективність та масштабованість додатку. Firestore надає можливості для створення гнучких структур даних та встановлення реляційних відносин між колекціями та документами. Давайте розглянемо кожен аспект детальніше.

Проєктування бази даних починається з визначення необхідних колекцій, документів та полів. Документи в кожній колекції визначаються як окремі записи, що містять поля з даними, такими як ім'я користувача, дата замовлення, вартість товарів тощо.

Після того, як визначені колекції та документи, їх можна створити у Firestore. Для кожного документа можна визначити поля, які містять дані різних типів: рядки, числа, дати, масиви, вкладені об'єкти тощо. Залежно від потреб додатку, базу даних можна організувати у складну ієрархічну структуру з підколекціями та документами всередині інших документів.

Хоч Firestore не має класичної табличної структури, як у реляційних базах даних, вона дозволяє створювати структури, які можна вважати аналогом таблиць. Колекції можна розглядати як таблиці, а документи як рядки у цих таблицях, причому поля документів виступають як стовпці. Розробка таблиць у

Firestore полягає у правильному проєктуванні колекцій та документів, які зберігають дані у структурованому форматі.

Використання посилань дозволяє створити різні типи відносин: один до одного, один до багатьох, багато до багатьох. Це допомагає моделювати складні взаємозв'язки між даними та забезпечує ефективне керування базою даних.\

У базі даних для додатку райдшерінгу є чотири основні колекції (таблиці): "users" (користувачі), "rides" (поїздки), "payment" (оплати), та "review" (відгуки). Ці колекції взаємопов'язані через різні типи зв'язків, що забезпечує інтерактивність та взаємодію між різними аспектами додатку.

Колекція "users" представляє користувачів, включаючи пасажирів та водіїв. Користувачі можуть бути пов'язані з поїздками через зв'язок багато до багатьох, де поїздки вказують на користувачів у ролі пасажирів або водіїв. Кожен користувач також може здійснювати кілька оплат, утворюючи зв'язок багато до одного з колекцією "payment". Крім того, користувачі можуть залишати відгуки про свої поїздки або інші користувачі, створюючи зв'язок багато до одного з колекцією "review".

Колекція "rides" представляє поїздки, в яких беруть участь користувачі. Вона пов'язана з користувачами через зв'язок багато до багатьох, вказуючи на пасажирів та водіїв поїздок. Крім того, кожна поїздка може мати пов'язану оплату, утворюючи зв'язок один до одного з колекцією "payment". Поїздки також можуть отримувати відгуки, створюючи зв'язок один до багатьох з колекцією "review".

Колекція "payment" відповідає за зберігання інформації про оплати за поїздки. Вона пов'язана з поїздками через зв'язок один до одного, вказуючи на поїздки, за які була здійснена оплата. Також кожна оплата пов'язана з користувачем, який її здійснив, утворюючи зв'язок багато до одного з колекцією "users".

Колекція "review" містить відгуки користувачів про поїздки та інших користувачів. Відгуки пов'язані з поїздками через зв'язок один до багатьох, вказуючи на поїздки, які були оцінені. Також відгуки пов'язані з користувачами,

які їх залишили, та користувачами, про яких вони були залишені, утворюючи зв'язок багато до одного з колекцією "users".

Такі зв'язки між колекціями дозволяють ефективно керувати взаємодією між користувачами, поїздками, оплатами та відгуками, забезпечуючи надійний та інтерактивний досвід у додатку райдшерінгу.

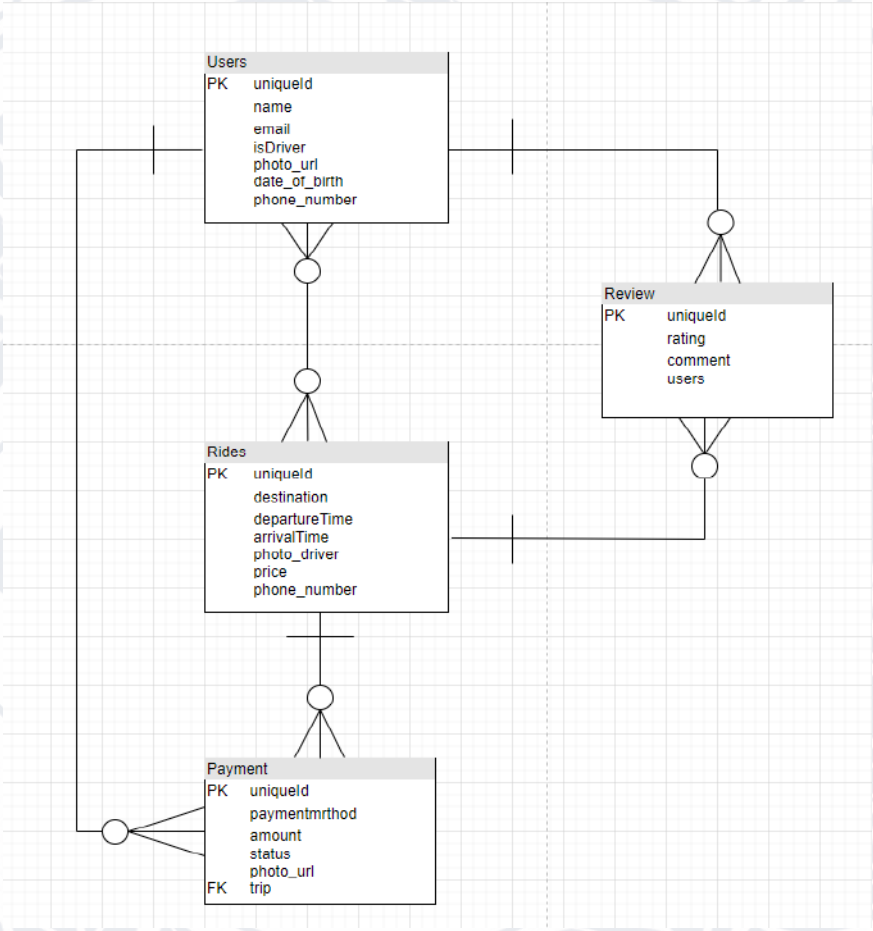


Рисунок 3.2 – Діаграма бази даних

3.3 Розробка додатку засобами FlutterFlow

Розпочавши проект додатку для райдшерингу, спершу зосередились на стадії планування та дизайну інтерфейсу. Для цього було проведено аналіз існуючих рішень на ринку та визначено основні функціональні вимоги, що повинні були включатися до додатку. Підходячи до цього з наукового погляду, ми керувалися такими методами:

Здійснено огляд та аналіз існуючих додатків райдшерингу, визначено їхні переваги та недоліки, а також здійснено аналіз потреб цільової аудиторії.

Проведено опитування та інтерв'ю з потенційними користувачами для визначення їхніх потреб та вимог до функціоналу додатку.

Визначено основні сценарії взаємодії користувачів з додатком, включаючи реєстрацію, пошук поїздок та оплату послуг.

Після планування функціоналу додатку перейшли до розробки його інтерфейсу з використанням FlutterFlow. FlutterFlow надав зручний інструментарій для швидкої реалізації дизайну без необхідності написання коду вручну. Ми розробили основні екрани додатку, такі як екран входу, реєстрації, пошуку поїздок, деталізації поїздки та екран оплати.

Екран входу є першим контактом користувача з додатком. Його головна мета - дозволити існуючим користувачам авторизуватися у системі. Основні елементи цього екрану включають:

Форма входу:

Поле для введення електронної адреси або номера телефону: Це текстове поле, де користувач може ввести свою електронну адресу або номер телефону, залежно від того, якими обліковими даними він користується для входу в додаток.

Також текстове поле, призначене для введення паролю користувача.

Посилання на відновлення паролю - це посилання, зазвичай під формою входу, яке дозволяє користувачам відновити свій пароль у випадку, якщо вони його забули або втратили. Після введення вірних облікових даних користувач натискає на кнопку увійти, щоб увійти в свій обліковий запис у системі.

Ця кнопка може бути активною або неактивною залежно від того, чи заповнені обов'язкові поля вводу (наприклад, електронна адреса і пароль).

Крім того, для поліпшення користувацького досвіду та забезпечення безпеки даних, екран входу також містити такі елементи:

Опція "Запам'ятати мене", яка дозволяє користувачам зручно увійти в систему без повторного введення своїх облікових даних при кожному вході.

Та кнопки авторизації через соціальні мережі наприклад, кнопки "Увійти через Facebook" або "Увійти через Google", які дозволяють користувачам

авторизуватися за допомогою своїх облікових записів у відповідних соціальних мережах.

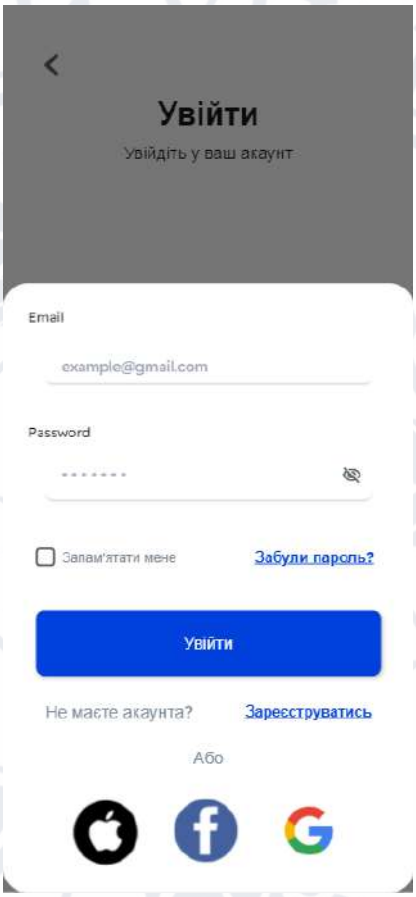


Рисунок 3.3 – Сторінка “Увійти” в додатку Share Wheels

Екран реєстрації потрібен для нових користувачів, які ще не мають облікового запису в системі. Основні елементи цього екрану включають:

Форма реєстрації: Ця форма містить поля для введення основної інформації, необхідної для створення облікового запису. Зазвичай вона включає такі поля, як: ім'я та прізвище, електронна адреса, пароль який, забезпечує безпеку облікового запису шляхом обмеження доступу до нього тільки за допомогою введення вірного пароля.

Вибір ролі: У додатку райдшерингу користувачі повинні мати можливість вибрати роль, яка відповідає їхнім потребам - водій або пасажир. Це важливий етап реєстрації, оскільки від вибору ролі залежатиме функціонал, доступний користувачу після реєстрації.

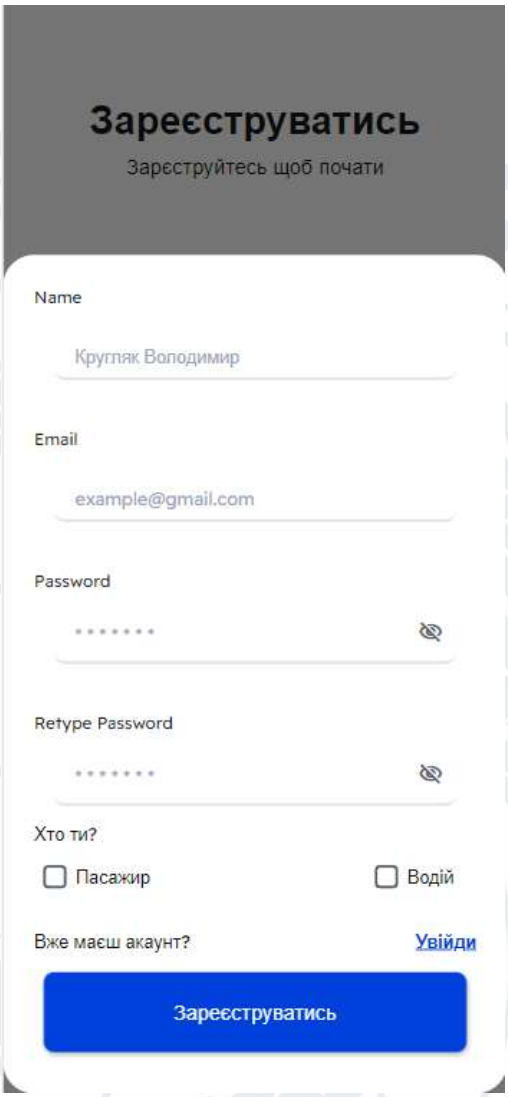


Рисунок 3.4 – Сторінка “Реєстрації” у додатку ShareWheels

Уявимо що реєстрація пройшла зі сторони пасажира. На головній сторінці пасажира буде відображений список доступних поїздок або маршрутів. Кожна поїздка буде відображена у вигляді окремого елементу, який містить інформацію про маршрут, ім’я, фото водія, маршрут та вартість. Користувач може використовувати спеціальне поле для введення міста, щоб знайти маршрут до пункту призначення. Після введення, додаток відобразить доступні поїздки для цього маршруту.

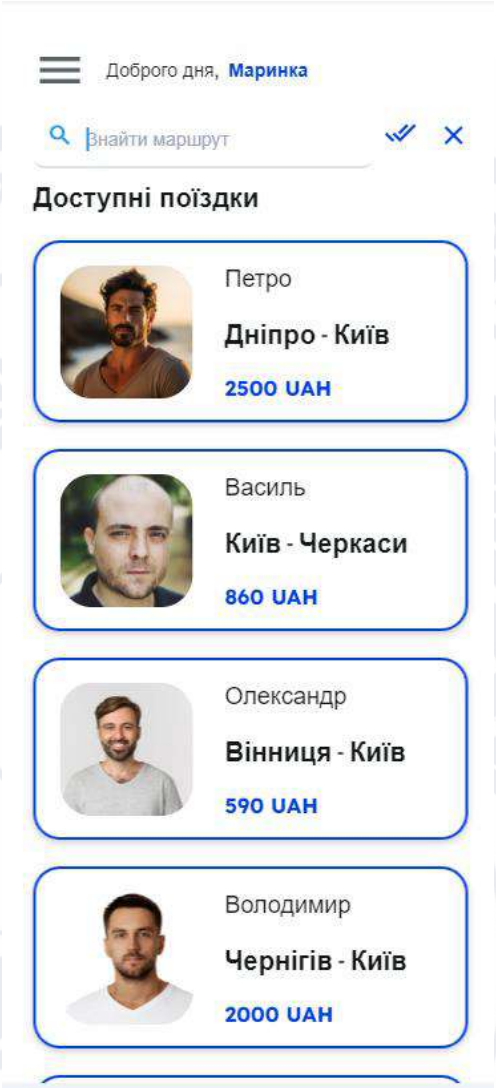


Рисунок 3.5 - Сторінка “Головна сторінка пасажира” в додатку ShareWheels.

Користувач може натискати на будь-яку поїздку у списку, щоб отримати більше інформації про неї. Після натискання на поїздку відкривається сторінка з детальною інформацією, де користувач може переглянути інформацію про маршрут, автомобіль, вартість, номер телефону водія для зв’язку тощо.

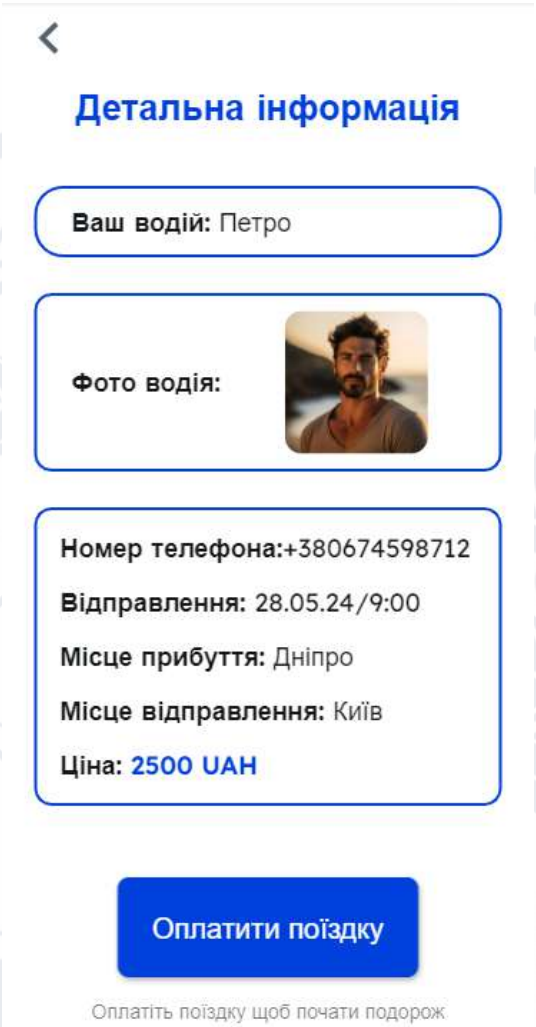


Рисунок 3.6 - Сторінка “Детальна інформація” в додатку ShareWheels

Користувач може вибрати зручний для себе метод оплати, такий як кредитна або дебетова карта, платіжна система або готівка. Користувач повинен ввести необхідні дані для здійснення оплати, такі як номер картки, термін дії, CVV-код та інші дані, які можуть знадобитися для проведення операції.

Після введення необхідної інформації користувач повинен натиснути кнопку "Оплатити". Після цього додаток здійснить оплату, а користувач отримає підтвердження про успішну операцію.

Рисунок 3.7 - Сторінка “Оплати” в додатку ShareWheels

Після успішної оплати користувач отримає повідомлення про це у додатку, яке підтвердить, що оплата була успішно проведена.

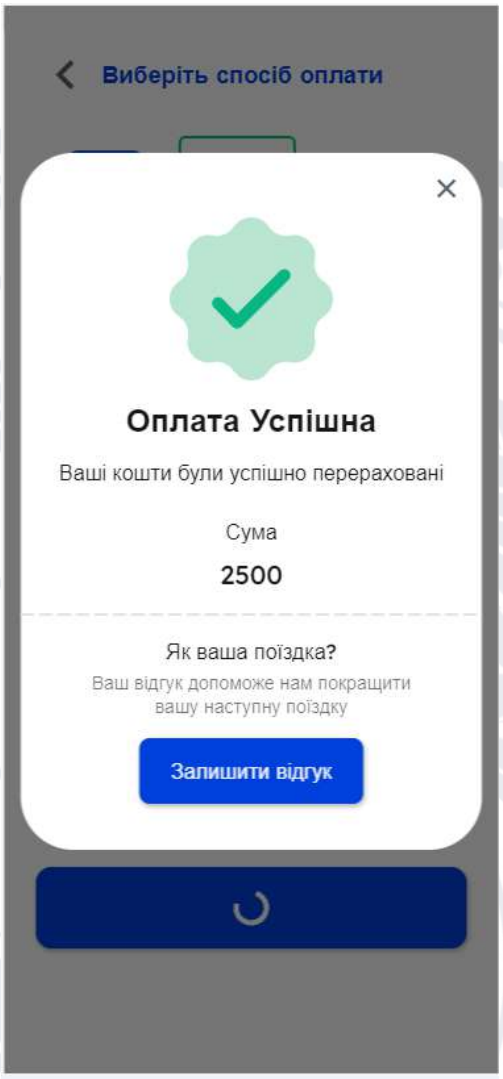


Рисунок 3.8 - Сторінка “Успішна оплата” в додатку Share Wheels

Після успішної оплати користувач може бачити форму для залишення відгуку про водія. Ця форма може містити поля для введення текстового відгуку та встановлення рейтингу водія (наприклад, за допомогою зірочок).

Користувач може встановити рейтинг водія за його роботу під час поїздки. Це може бути важливою інформацією для інших користувачів, які шукають водіїв у майбутньому. Користувач може залишити додаткові коментарі або відгуки про водія, які можуть бути корисні для інших користувачів. Текстовий відгук може містити позитивні або негативні коментарі про сервіс, стиль водіння, комунікацію тощо.

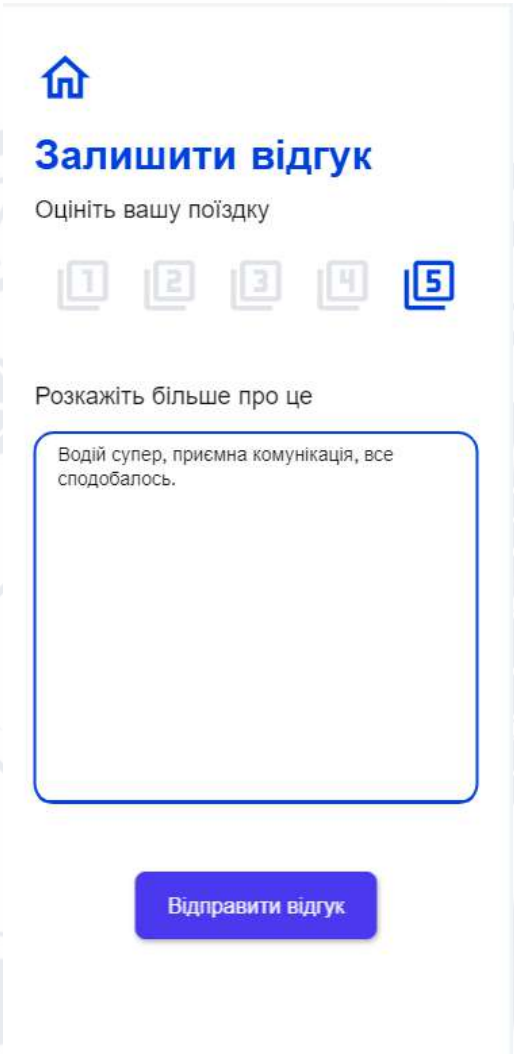


Рисунок 3.9 - Сторінка “Залишити відгук” в додатку ShareWheels

Головною частиною головної сторінки буде список доступних поїздок, які створили інші водії. Цей список буде включати інформацію про маршрут, вартість та інші важливі деталі.

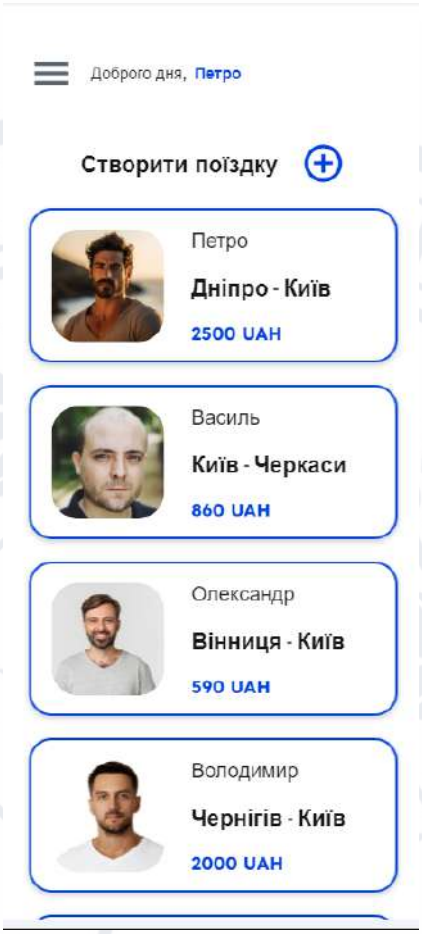


Рисунок 3.10 - Сторінка “Головна сторінка водія” в додатку ShareWheels

Водій може створити нову поїздку, натиснувши на відповідну кнопку або елемент інтерфейсу. Після цього він може вказати деталі маршруту, час відправлення, вартість та інші необхідні параметри.

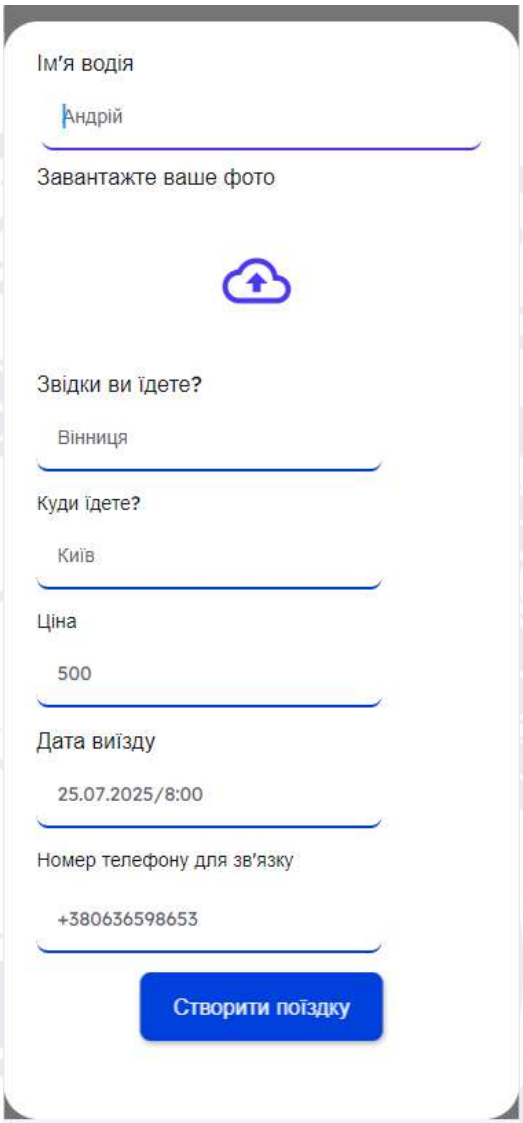


Рисунок 3.11 - Сторінка “Створити поїздку” в додатку ShareWheels

Кожен користувач може редагувати основну інформацію в своєму профілі, таку як ім'я, номер телефону та дату народження. Це дозволяє користувачам оновлювати свої особисті дані у разі зміни. Користувачі можуть додавати або змінювати своє фото профілю, щоб персоналізувати свій обліковий запис та зробити його більш впізнаваним для інших користувачів.

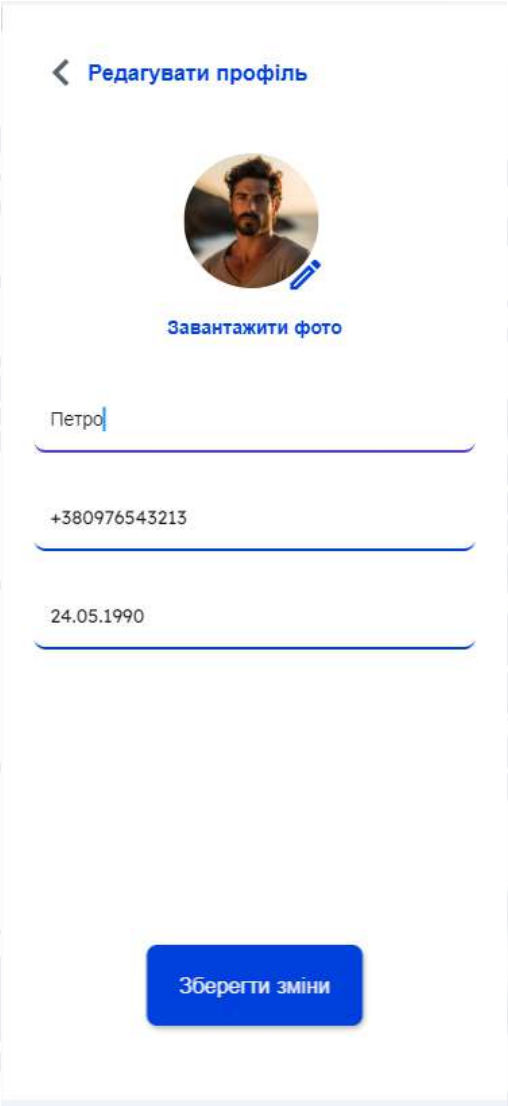


Рисунок 3.12 - Сторінка “Редагувати профіль” в додатку ShareWheels

Сторінка налаштувань у додатку райдшерингу дозволяє користувачам налаштовувати різні аспекти їхнього облікового запису та досвіду користування додатком. Центр допомоги, де користувачі можуть знайти відповіді на свої питання щодо користування додатком, вирішення проблем або неполадок.

Розділ "Про нас", який містить загальну інформацію про компанію або сервіс, контактна інформація тощо. Можливість відновлення паролю, щоб користувачі могли відновити свій пароль у разі забуття або втрати.

Розділ з політикою конфіденційності, який визначає, які особисті дані збираються та обробляються додатком, а також правила щодо їхньої захисту та конфіденційності.

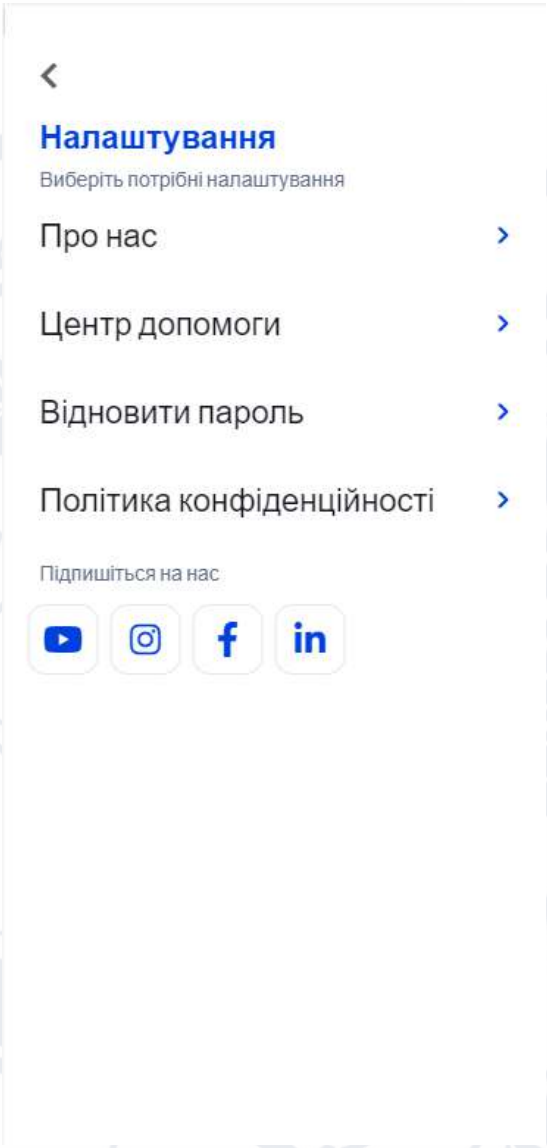


Рисунок 3.13 - Сторінка “Налаштування” в додатку ShareWheels

ВИСНОВКИ

У ході даної дипломної роботи було проведено дослідження та розробка мобільного додатку для райдшерингу, заснованого на технологіях FlutterFlow та Firebase. Цей процес включав аналіз потреб користувачів, проектування та реалізацію функціональних модулів, тестування та оптимізацію продукту.

Під час досліджень було встановлено, що використання FlutterFlow дозволило значно зменшити час розробки і спростити процес створення інтерфейсу користувача. Крім того, інтеграція Firebase забезпечила надійність зберігання даних та безпеку механізмів аутентифікації користувачів.

Основні функціональні можливості додатку, такі як створення та виконання поїздок, відстеження геолокації та оплата послуг, були успішно реалізовані. Користувачі відзначили зручність та простоту використання додатку.

Оптимізація та тестування додатку на різних пристроях сприяли підвищенню продуктивності та відповідності сучасним стандартам мобільних додатків.

Під час розробки додатку було використано різні методики та патерни дизайну, такі як MVC та MVVM, що дозволило ефективно організувати структуру проекту та розділити логіку додатку на окремі компоненти. Це сприяло збереженню чистоти коду, його повторному використанню та забезпечило легкість підтримки і розширення функціоналу.

Також важливим етапом у розробці була реалізація use case diagram та системної архітектури додатку. Вони допомогли чітко визначити основні функціональність та взаємозв'язки між різними компонентами системи, що сприяло зрозумінню та управлінню розробкою проекту.

У підсумку, результати цієї роботи підтверджують ефективність використання технологій FlutterFlow та Firebase у розробці мобільних додатків, а також підкреслюють їхню важливість у створенні високоякісних та функціональних продуктів для користувачів.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. lessandro Biessek. "Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2". Packt Publishing, 2019. 562 p.
2. Carmine Zaccagnino. "Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games". Packt Publishing, 2019. 398 p.
3. Eric Windmill. "Flutter in Action". Manning Publications, 2020. 384 p.
4. Marco L. Napoli. "Beginning Flutter: A Hands-On Guide to App Development". Wiley, 2019. 528 p.
5. Frank Zammetti. "Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK". Apress, 2019. 416 p.
6. David DeRemer, Kevin D. Moore. "Flutter Recipes: Mobile Development Solutions for iOS and Android". Apress, 2020. 552 p.
7. Thomas Bailey. "Flutter for Dummies". Wiley, 2020. 480 p.
8. Carmine Zaccagnino. "Flutter Cookbook: Over 100 Proven Techniques and Solutions for App Development with Flutter 2.2 and Dart". Packt Publishing, 2021. 674 p.
9. Michael Katz. "Building Cross-Platform Apps with Flutter and Dart". Apress, 2021. 356 p.
10. Chris Buckett. "Dart in Action". Manning Publications, 2013. 424 p.
11. Олександр Гладкий. "Програмування на Dart для початківців". Київ: Видавництво "Факт", 2020. 312 с.
12. Олександр Литвиненко. "Flutter: Розробка мобільних додатків". Львів: Видавництво "Самміт-Книга", 2021. 420 с.
13. Ігор Мазур. "Основи Flutter: Створення мобільних додатків". Київ: Діафра-К, 2020. 384 с.
14. Андрій Ковальчук. "Практичний курс по Flutter". Київ: Наш Формат, 2021. 448 с.

15. Владислав Галка. "Програмування на Dart". Київ: Видавництво "Світ книги", 2019. 368 с.
16. Юрій Литвин. "Створення кросплатформених додатків на Flutter". Київ: Видавництво "Будинок книги", 2020. 400 с.
17. Іван Мельник. "Мобільна розробка на Flutter для початківців". Львів: Видавництво Самміт-Книга, 2021. 328 с.
18. Володимир Коваль. "Розробка мобільних додатків з використанням Firebase". Київ: Діафра-К, 2020. 352 с.
19. Марія Петрів. "Програмування мобільних додатків на Dart і Flutter". Львів: Видавництво "Світ", 2020. 300 с.
20. Андрій Вовк. "Інтеграція Firebase з мобільними додатками". Київ: Видавництво "Факт", 2020. 320 с.
21. Flutter Flow Pros and Cons in App Production. URL: https://scaleupally.io/blog/flutterflow-pros-and-cons-in-production/
22. What is FlutterFlow? Full guide, review, and exploration of alternatives. URL: https://www.lowcode.agency/blog/flutterflow
23. What is Flutter Flow, and why is it used?. URL: https://taglineinfotech.com/what-is-flutter-flow/#:~:text=Flutter%20Flow%20was%20a%20visual,apps%20without%20writing%20extensive%20code
24. What is Flutter Flow, and why is it used?. URL: https://taglineinfotech.com/what-is-flutter-flow/#What_is_Flutter_Flow
25. Live Tracking in Google Maps Using FlutterFlow and Firebase - Rider App. URL: https://blog.flutterflow.io/live-tracking-google-maps-rider/
26. Live Tracking in Google Maps Using FlutterFlow and Firebase - Driver App. URL: https://blog.flutterflow.io/live-tracking-google-maps-driver/
27. Laurence Moroney. "The Definitive Guide to Firebase". Apress, 2017. 354 p.
28. K. Scott Allen. "Modern Authentication with Azure Active Directory for Web Applications". O'Reilly Media, 2015. 130 p.

29. Claus Matzinger. "Hands-On Mobile Development with .NET Core". Packt Publishing, 2019. 450 p.
30. Waleed Arshad. "Practical Flutter: Build Beautiful, Interactive Apps". Apress, 2021. 400 p.
31. Neil Smyth. "Flutter Essentials: Second Edition". Payload Media, 2020. 750 p.
32. Aki Iskandar. "Flutter for Job Seekers: A guide to get a job using Flutter and Dart". Independently published, 2020. 310 p.
33. Souvik Paul. "Firebase Cookbook". Packt Publishing, 2019. 296 p.
34. Priyanka Tyagi. "Firebase Essentials - Android Edition". Packt Publishing, 2017. 176 p.
35. Олексій Тарасов. "Програмування на Dart: від початківця до професіонала". Львів: Видавництво "Самміт-Книга", 2020. 380 с.
36. Дмитро Павленко. "Створення мобільних додатків з Flutter і Firebase". Київ: Видавництво "Світ", 2020. 296 с.
37. Сергій Гончаренко. "Кросплатформенний розвиток на Flutter". Київ: Видавництво "Будинок книги", 2021. 360 с.
38. Андрій Бондар. "Flutter і Dart для початківців". Львів: Видавництво "Самміт-Книга", 2020. 340 с.
39. Олена Яворська. "Інтеграція Firebase в мобільні додатки". Київ: Видавництво "Факт", 2021. 290 с.
40. Віталій Петров. "Створення додатків на Flutter з використанням Firebase". Київ: Видавництво "Діафра-К", 2020. 308 с.

ДОДАТКИ

ДОДАТОК А

Опис структури бази даних

Опис структури таблиці “Users”

Таблиця А.1

Ключ	Ім’я поля	Тип даних	Розмір поля	Опис
PK	uniqueID	Числовий	Ціле	Ідентифікатор
	name	Символьний	100	Ім’я
	email	Символьний	100	Електронна адреса
	isDriver	Boolean		Чи користувач є водієм
	date_of_birth	Дата/час		Дата народження
	created_at	Дата/час		Час створення
	phone_number	Символьний	50	Номер телефону

Опис структури таблиці “Review”

Таблиця А.2

Ключ	Ім’я поля	Тип даних	Розмір поля	Опис
PK	uniqueID	Числовий	Ціле	Ідентифікатор
	rating	Числовий	Ціле	Рейтинг
	comment	Символьний	255	Детальний відгук
FK	userID	Числовий	Ціле	Ідентифікатор користувача

Опис структури таблиці “Rides”

Таблиця А.3

Ключ	Ім’я поля	Тип даних	Розмір поля	Опис
PK	uniqueID	Числовий	Ціле	Ідентифікатор
	destination	Символьний	100	Напрямок
	departure_time	Дата/Час		Дата відправки
	arrival_time	Дата/Час		Дата прибуття
	photo_driver	Image_path		Фото водія
	price	Числовий	З плаваючою комою	Ціна

	phone_number	Символьний	50	Номер телефону

Опис структури таблиці “Payment”

Таблиця А.4

Ключ	**Ім’я поля**	**Тип даних**	**Розмір поля**	**Опис**
PK	uniqueID	Числовий	Ціле	Ідентифікатор
	payment_method	Enum		Matercard, visa, cash, paypal
	amount	Числовий	З плаваючою комою	Сума
	status	Enum		Оплачено, не оплачено, в очікуванні
FK	trip_id	Числовий	Ціле	Ідентифікатор поїздки

ДОДАТОК Б

Data-Flow diagram додатку ShareWheels

Рисунок Б.1

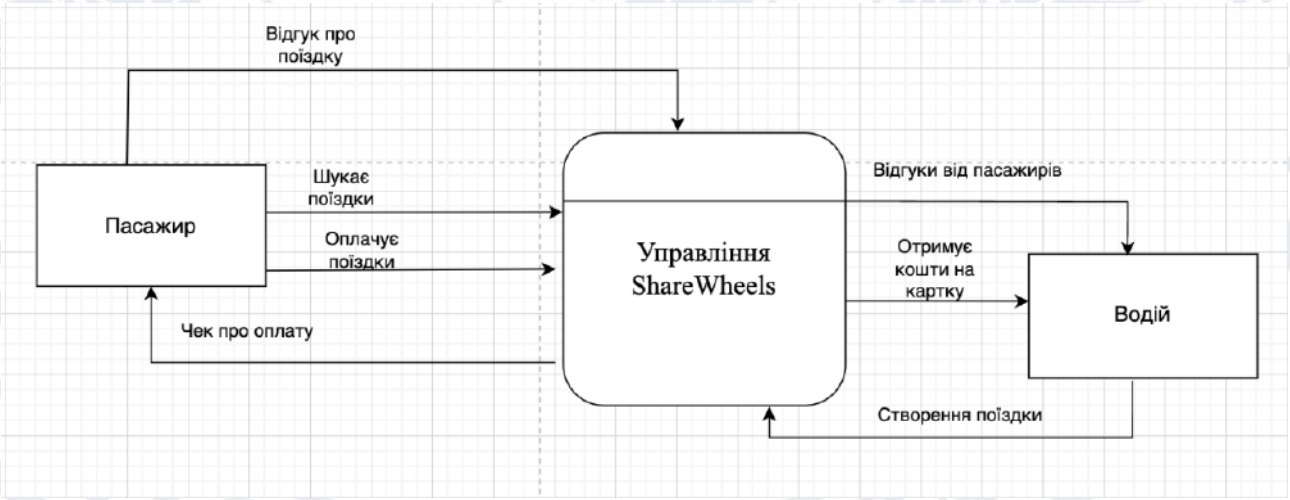
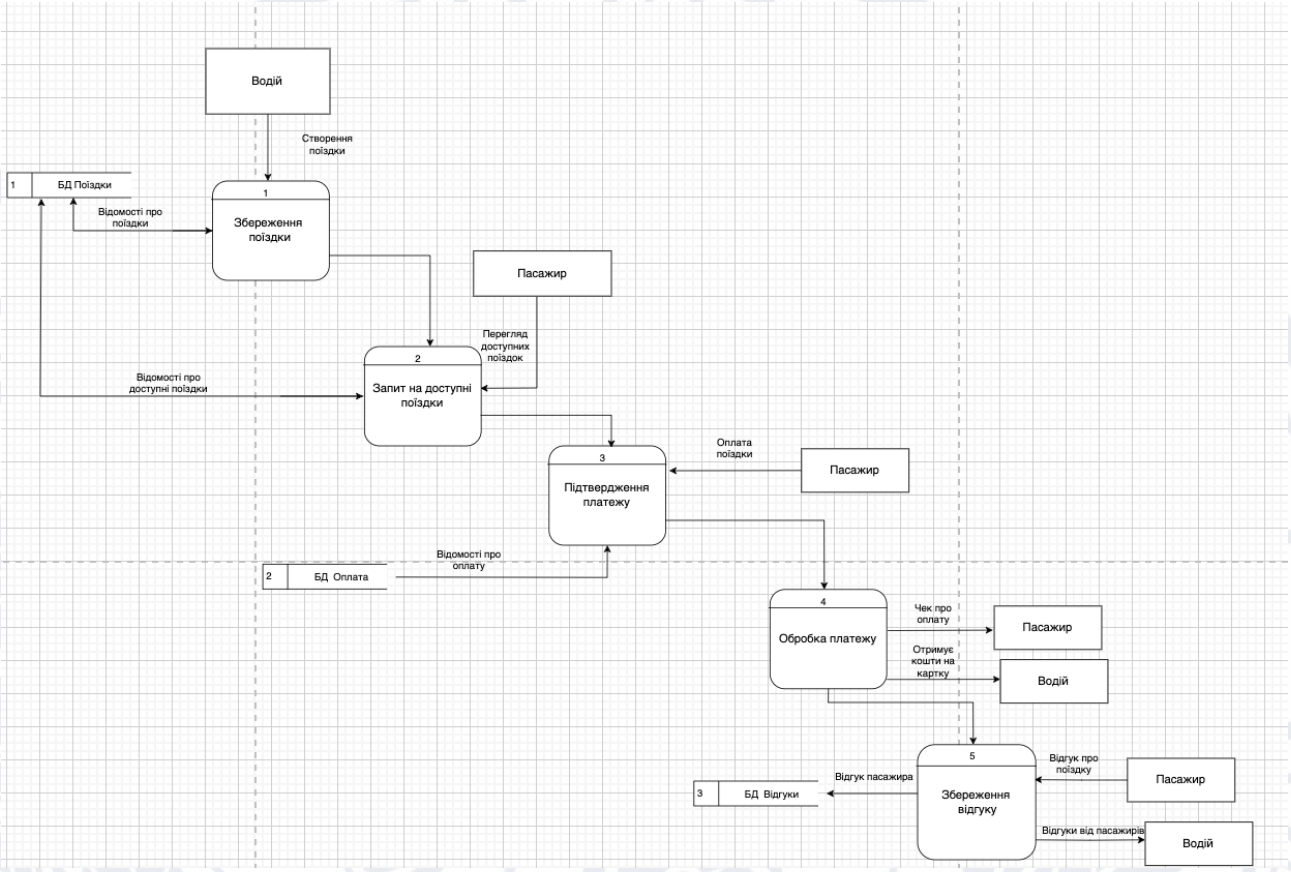


Рисунок Б.2


Водій
Створення поїздки
1
БД Поїздки
Відомості про поїздки
1
Збереження поїздки
Пасажир
Перегляд доступних поїздок
2
Запит на доступні поїздки
Відомості про доступні поїздки
Оплата поїздки
Пасажир
3
Підтвердження платежу
2
БД Оплата
Відомості про оплату
4
Обробка платежу
Чек про оплату
Пасажир
Отримує кошти на картку
Водій
5
Збереження відгуку
Відгук про поїздку
Пасажир
Відгук пасажира
3
БД Відгуки
Відгуки від пасажирів
Водій

Харченко Ерсан Кенанович

Факультет інформаційні та прикладні технології

122 Комп'ютерні науки

Освітня програма

ДЕКЛАРАЦІЯ
про дотримання академічної доброчесності

Я, _____
_____
____

Повністю вказується ПІБ та статус (посада для працівників, освітня (освітньо-наукова) програма – для здобувачів вищої освіти)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»; що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації; що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів; що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)