КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Інформаційних технологій

(назва випускової кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПОШУКУ ПОПУТНИКІВ

Стужука Ігоря Миколайовича

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації інформаційних технологій

(факультет)

Інформаційних технологій

(назва кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТ

Тетяна Гончаренко

_____

„___”_____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ
ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПОШУКУ ПОПУТНИКІВ

Виконав
Стужук Ігор Миколайович

Комп’ютерні науки
(спеціальність)
Інформаційні управляючі системи та технології
(освітня програма)
Групи КНс-21
Керівник к.т.н., доц. Гончаренко Т.А., PhD, Рябчун Ю.В.

Ідентичність підтверджую

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації інформаційних технологій

Випускова кафедра: Інформаційних технологій

Освітній ступінь: Бакалавр

Спеціальність: Комп'ютерні науки

Освітня програма: Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТ

Тетяна ГОНЧАРЕНКО

_____

„___" _____2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Стужуку Ігорю Миколайовичу

1. Тема роботи РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ПОШУКУ ПОПУТНИКІВ

затверджена наказом ректора КНУБА № 433/2 від «29» лютого 2024 року

2. Керівник роботи Гончаренко Тетяна Андріївна, к.т.н., доц.

3. Строк подання Здобувачем роботи до захисту_____

4. Зміст пояснювальної записки за розділами:

Р.1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Р.2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.4 ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

5. Графічний матеріал за розділами:

Р.1_____

Р.2_____

Р.3_____

Р.4_____

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	
Розділ 2	
Розділ 3	
Розділ 4	
Остаточне оформлення роботи	
Направлення роботи для перевірки на плагіат	
Попередній захист роботи на випусковій кафедрі	
Направлення роботи на рецензування	

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1			
Розділ 2			
Розділ 3			
Розділ 4			

8. Дата видачі завдання _____

Зав. кафедри	_____	Гончаренко Т.А.
	(підпис)	(прізвище та ініціали)
Керівники	_____	Гончаренко Т.А.
	(підпис)	(прізвище та ініціали)
	_____	Рябчун Ю.В.
	(підпис)	(прізвище та ініціали)
Здобувач	_____	Стужук І.М.
	(підпис)	(прізвище та ініціали)

АНОТАЦІЯ

Стужук І. М. Розробка мобільного додатку для пошуку попутників.

Атестаційна випускна робота бакалавра за спеціальністю 122 «Комп’ютерні науки», освітня програма «Інформаційні управляючі системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2024.

Дана дипломна робота присвячена розробці мобільного додатку для пошуку попутників, який автоматизує процес організації спільних поїздок. Робота включає практичну реалізацію розроблених функціональних можливостей додатку, таких як реєстрація користувачів, пошук попутників, створення та бронювання поїздок. Експериментальна частина оцінює зручність використання, ефективність пошуку попутників та загальну продуктивність додатку за допомогою тестування. На основі отриманих результатів формуються висновки та рекомендації для подальшого вдосконалення мобільного додатку, підвищення його функціональності та покращення користувацького досвіду.

Робота викладена на 110 аркушах, містить 3 додатки, 27 таблиць, 45 рисунків, список використаної літератури із 25 найменувань.

Ключові слова: ПЗ, БД, OS, UML, ERD, Java, SQLite, Android Studio.

SUMMARY

Stuzhuk I. M. Development of a mobile application for searching for road passengers. Bachelor's thesis in the specialty: 122 "Computer Science," specialization: "Information Management Systems and Technologies." – Kyiv National University of Construction and Architecture. – Kyiv, 2024.

This thesis is dedicated to the development of a mobile application for finding travel companions, which automates the process of organizing shared trips.

The work includes the practical implementation of the developed functionality of the application, such as user registration, search for fellow travelers, creation and booking of trips. The experimental part evaluates the usability, the effectiveness of searching for fellow travelers and the overall performance of the application through testing. Based on the obtained results, conclusions and recommendations are formed for

further improvement of the mobile application, increasing its functionality and improving the user experience.

The development of this application addresses a significant demand for efficient and user-friendly platforms that help users arrange shared travel. By automating the process, the application saves time and enhances convenience, allowing users to connect with potential travel partners easily. The integration of technologies like Java and SQLite ensures the application's robust performance and reliable data management, while Android Studio provides a seamless and intuitive development environment.

The work is presented on 110 pages, contains 3 appendices, 27 tables, 45 figures, and a list of references with 25 titles.

Keywords: SW, DB, OS, UML, ERD, Java, SQLite, Android Studio.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

БД – база даних;

OS (Operation System) – операційна система;

UML (Unified Modelling Language) – універсальна мова моделювання;

ERD (entity-relationship diagram, ER-діаграма) – діаграма сутність-зв'язок;

Java – об’єктно-орієнтована мова програмування;

SQLite – компактна вбудована реляційна база даних;

Android Studio – інтегроване середовище розробки (IDE) для платформи Android.

ВСТУП

У сучасному світі, з розвитком технологій і зростанням мобільності населення, питання ефективного використання транспортних засобів стає все більш актуальним. Багато країн рано чи пізно стикаються з транспортною проблемою, суть якої полягає в перевантаженості існуючої мережі доріг та пасажироперевезень. До того ж, існуюча мережа маршрутів громадського транспорту не завжди повністю покриває населений пункт, і багатьом пасажирам доводиться або тривалий час добиратися до зупинок, або відчутно довго чекати громадський транспорт. Основні способи вирішення проблеми – розвиток транспортної інфраструктури міста та збільшення кількості маршрутів громадського транспорту. Всі ці рішення мають досить істотні недоліки. Вони вимагають величезного вкладення фінансів, збільшують викиди CO2 в атмосферу, завдаючи шкоди навколишньому середовищу, довго окупаються, підвищують завантаженість доріг, що може спричинити часті затори і аварії, а також виникають проблеми з пошуком вільних паркувальних місць.

Однак, існує рішення, позбавлене цих недоліків – розвиток системи карпула. Карпул – це спільні поїздки на автомобілі. Переваги даної системи очевидні: на дорогах зменшується число транспортних засобів, завдяки чому зменшується число заторів і ймовірність аварій, істотно знижується викид парникових газів і зменшується кількість припаркованих автомобілів на вулицях міста. Для учасників карпула стає очевидним виграш у вартості і зручності поїздки: використовується тільки один автомобіль, отже, знижуються загальні витрати на паливо, ремонт і обслуговування. До того ж, виникає можливість додаткового спілкування людей.

Сучасні технології можуть значно спростити та покращити процес організації карпулінгу. Мобільні додатки та онлайн-платформи дозволяють користувачам легко знаходити попутників, домовлятися про деталі поїздок та стежити за рейтингами учасників. Такі додатки можуть включати функції інтеграції з картографічними сервісами для оптимізації маршрутів, системи

безпеки, засновані на відгуках та рейтингах, а також зручні способи обміну повідомленнями для координації поїздок.

Дана дипломна робота присвячена розробці мобільного додатку для пошуку попутників, який автоматизує процес організації спільних поїздок. Мобільний додаток надасть користувачам можливість легко і швидко знаходити попутників, координувати маршрути та домовлятися про деталі поїздки. Він забезпечить зручний інтерфейс для створення і пошуку поїздок, систему рейтингів і відгуків для підвищення довіри між користувачами, а також інтеграцію з картографічними сервісами для оптимізації маршрутів.

Метою даної роботи є розробка та впровадження програмного забезпечення, яке сприятиме зниженню транспортного навантаження на міські дороги, покращенню екологічної ситуації та підвищенню економічної ефективності поїздок. У роботі буде детально розглянуто процес проектування додатку, його функціональні можливості, технічні аспекти реалізації, а також проведено аналіз очікуваних результатів від впровадження даного рішення.

Завдання дослідження:

1. Аналіз існуючих систем карпула та вивчення їхніх переваг і недоліків.
2. Проектування архітектури мобільного додатку для пошуку попутників.
3. Розробка функціональних модулів мобільного додатку, включаючи інтерфейс користувача.
4. Впровадження та тестування мобільного додатку в реальних умовах.
5. Аналіз впливу впровадженого додатку на транспортне навантаження, екологічну ситуацію та економічну ефективність поїздок.

Об’єкт дослідження – процес організації спільних поїздок на автомобілях у міських умовах.

Предмет дослідження – мобільний додаток для пошуку попутників та його вплив на транспортне навантаження, екологію та економічну ефективність поїздок.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Пошук попутників стає все більш популярним у сучасному світі, де люди шукають співмандрівників для спільних подорожей, незалежно від їх масштабу та тривалості. Це може бути як випадок подорожі від місця проживання до роботи або навчального закладу, так і бажання спільно подорожувати в інші країни або відвідувати різні цікаві місця.

Попри існування великої кількості онлайн-сервісів та додатків для пошуку попутників, існує необхідність вдосконалення цього процесу. Часто існуючі рішення не забезпечують достатньої ефективності, зручності та безпеки для користувачів.

Аналіз предметної області допомагає глибше зрозуміти потреби та очікування користувачів. Це дозволяє визначити функціональні та нефункціональні вимоги до програмного забезпечення та забезпечити його відповідність потребам цільової аудиторії. Аналіз предметної області визначає напрямок подальшої розробки. Він допомагає зрозуміти, які функції та можливості програми будуть найбільш корисними для користувачів, і які аспекти системи потребують особливої уваги. Ретельний аналіз предметної області допомагає виявити потенційні проблеми та ризики ще на початковому етапі розробки. Це дозволяє вчасно приймати заходи для їх уникнення або зменшення впливу на подальшу роботу.

Постановка задачі допомагає чітко сформулювати проблеми, які потрібно вирішити, і визначити конкретні цілі, які необхідно досягти. Це створює основу для подальшої розробки та оцінки ефективності програмного забезпечення.

В даному розділі проведено аналіз предметної області, що стосується розробки мобільного додатку для пошуку попутників, аналіз готових рішень, після аналізу визначено основні проблеми та вимоги до програми, а також поставлено задачу, яку необхідно вирішити за допомогою розроблюваного програмного забезпечення. Всі ці етапи є ключовими у процесі розробки програмного

забезпечення, який визначає подальші кроки і спрямовує робочий процес на досягнення успішного результату.

1.1 Постановка та аналіз проблеми

У сучасному світі, де рух населення є невід'ємною частиною щоденного життя, існує потреба в ефективних засобах організації та спрощення спільних поїздок. Незважаючи на існуючі транспортні засоби та сервіси, відсутність зручного та надійного мобільного додатку для пошуку попутників ускладнює процес знаходження співмандрівників для подорожей.

Основною проблемою є відсутність централізованого та ефективного інструменту, який би забезпечував швидкий та зручний пошук попутників будь-яким транспортним засобом. Такий додаток повинен враховувати різноманітність критеріїв користувачів, забезпечуючи безпеку та зручність використання.

Основні аспекти постановки проблеми включають відсутність централізованого інструменту для пошуку попутників, оскільки існуючі сервіси роздріблені та не надають комплексного підходу, охоплюючи різні види транспорту та враховуючи потреби широкого спектру користувачів. Специфічні проблеми існуючих сервісів полягають у тому, що багато платформ обмежуються лише конкретним видом транспорту (автомобіль, автобус, літак), що ускладнює пошук попутників для комплексних маршрутів. Також спостерігається низька безпека та довіра до цих сервісів, оскільки більшість не надають належних засобів для перевірки ідентичності користувачів та забезпечення взаємодовіри між учасниками поїздок. Крім того, відсутність персоналізації означає, що багато платформ не забезпечують гнучкість у налаштуванні критеріїв пошуку, що призводить до невідповідності очікувань користувачів. Нарешті, існує відсутність інновацій та використання сучасних технологій, оскільки багато додатків не використовують потенціалу штучного інтелекту, геолокації та інших передових технологій, що обмежує їхню ефективність та конкурентоспроможність.

Отже, проблема полягає у відсутності інтегрованого, інноваційного та безпечного мобільного додатку для пошуку попутників, що обслуговує потреби

різних користувачів та сприяє зручності та ефективності організації спільних поїздок.

1.2 Дерево цілей

Дерево цілей — це інструмент для ієрархічної організації та визначення основних та підпорядкованих цілей для проекту або системи.

Основна ціль дипломної роботи – це розробка та впровадження мобільного додатку для пошуку попутників, що передбачає функціональність та зручність використання, безпеку та довіру, інновації та використання технологій, розширення користувальницької бази та конкурентоспроможність, а також стабільність та безперебійна робота. Для забезпечення функціональності та зручності використання додаток повинен мати швидкий та ефективний пошук попутників, інтуїтивно зрозумілий та легкий інтерфейс, а також функції персоналізації для кожного користувача. Безпека та довіра досягаються через механізми перевірки та підтвердження ідентичності користувачів, забезпечення конфіденційності особистої інформації, та систему взаємної оцінки і відгуків для створення довіри. Інновації та використання технологій включають застосування штучного інтелекту для персоналізованого аналізу та рекомендацій, використання геолокаційних технологій для точного визначення місцезнаходження користувачів та маршрутів, а також реалізацію сучасних засобів комунікації та обміну інформацією. Для розширення користувальницької бази та підвищення конкурентоспроможності, додаток повинен бути мультиплатформеним, активно просуватися для залучення нових користувачів, аналізувати конкурентне середовище та вдосконалювати функціонал. Забезпечення стабільності та безперебійної роботи включає високий рівень стабільності та продуктивності, системи резервного копіювання та відновлення для уникнення втрат даних, а також регулярні оновлення для виправлення помилок та вдосконалення функціоналу.

1.3 Вимоги та особливості проєктування системи

Вимоги та особливості проєктування системи є ключовим етапом в розробці будь-якого інформаційного проєктую. Ці вимоги та особливості визначають параметри та характеристики системи, спрямовані на задоволення потреб користувачів та вирішення конкретних завдань.

Вимоги до системи:

1.3.1 Функціональні вимоги

- Реєстрація та авторизація користувачів.
- Можливість створення та редагування профілю користувача.
- Пошук попутників за різними критеріями, такими як маршрут, час, вартість.
- Можливість ініціювати та долучатися до спільних подорожей.
- Система взаємної оцінки та відгуків для користувачів.
- Інтеграція з картографічними сервісами та системами геолокації.

1.3.2 Безпека та конфіденційність

- Захищена реєстрація та вхід в систему з використанням надійних методів аутентифікації.
- Захист особистих даних користувачів.
- Система взаємної перевірки користувачів для забезпечення безпеки спільних поїздок.

1.3.4 Інтерфейс

- Інтуїтивно зрозумілий та зручний інтерфейс користувача для мобільних пристроїв.
- Можливість персоналізації інтерфейсу для врахування індивідуальних потреб користувачів.

1.3.5 Інновації та технології

- Використання штучного інтелекту для рекомендацій та аналізу вподобань користувачів.
- Геолокаційні служби для точного визначення місцезнаходження та маршрутів.

− Інтеграція з сучасними технологіями комунікації та обміну інформацією.

Особливості проєктування системи:

1.3.6 Мультиверсійність

Розробка додатку, який буде доступний на основних мобільних версіях OS Android для максимального охоплення аудиторії.

1.3.7 Розширюваність

Проєктування системи з урахуванням можливості додавання нових функцій та розширення функціоналу в майбутньому.

1.3.8 Адаптивність

Забезпечення адаптивного дизайну для зручного використання на різних розмірах екранів мобільних пристроїв.

1.3.9 Висока продуктивність

Оптимізація додатку для високої продуктивності та швидкодії, навіть при великій кількості користувачів.

1.3.10 Підтримка міжнародних користувачів

Забезпечення можливості використання додатку для організації спільних поїздок у різних країнах та містах.

1.3.11 Моніторинг та підтримка

Реалізація системи моніторингу та підтримки для оперативного виявлення та вирішення проблем користувачів.

1.3.12 Регулярні оновлення

Розробка плану регулярних оновлень для виправлення помилок, покращення безпеки та впровадження нових функцій відповідно до змін потреб користувачів і технологічних трендів.

1.4 Аналіз готових рішень

Для успішної розробки додатку важливо провести глибокий аналіз існуючих готових рішень на ринку. Цей процес дозволяє розібрати сильні та слабкі сторони конкурентів, виявити можливі ніші і можливості для вдосконалення власного

продукту. Аналізуючи успіхи і недоліки інших рішень, можна зрозуміти, що приваблює користувачів у даній галузі: чи це зручний інтерфейс, швидкість обробки замовлень, або широкий вибір послуг. Також важливо виявити, де інші рішення мають слабкі сторони, які можна використати для покращення власного продукту. Цей аналіз допоможе розробити унікальні конкурентні переваги і створити продукт, який буде приваблювати користувачів і відповідати їхнім потребам більш ефективно, ніж існуючі рішення на ринку.

1.4.1 Blablacar

Blablacar (Рис.1.1) – це онлайн-платформа для спільного використання автомобілів, яка дозволяє користувачам знаходити попутників для спільних поїздок. Створений у 2006 році, Blablacar став одним з найбільш популярних сервісів карпулінгу в світі.

Рисунок 1.1 – Blablacar

Головна мета Blablacar полягає у зменшенні транспортних витрат, екологічних викидів та зменшенні транспортного трафіку шляхом забезпечення зручного та ефективного співвідношення між водіями та пасажирами, які мають спільні маршрути.

На платформі Blablacar користувачі можуть створювати облікові записи, вказуючи деталі своїх поїздок (наприклад, дату, час і маршрут), і шукати попутників, які подорожують тим же напрямком. Водії можуть запропонувати

вільні місця в своєму автомобілі, а пасажири можуть забронювати місце на відповідному маршруті.

Основні переваги Blablacar включають широке покриття (сервіс доступний у більшості країн), зручний інтерфейс користувача, велику базу активних користувачів та можливість економії на транспортних витратах.

Незважаючи на певні обмеження, такі як обмежена інтеграція з іншими видами транспорту та фокус на автомобільних поїздках, Blablacar залишається популярним і надійним вибором для тих, хто шукає ефективний та соціально відповідальний спосіб подорожей.

Сильні сторони:

- Широке покриття та популярність сервісу в різних країнах.
- Зручний інтерфейс та велика база користувачів.

Слабкі сторони:

- Орієнтований переважно на автомобільні подорожі, що може обмежити аудиторію.
- Відсутність інтеграції з іншими видами транспорту.

1.4.2 Uber, Lyft

Uber та Lyft - це дві найбільші компанії у сфері ридшерінгу (спільного використання транспортних засобів) в США та багатьох інших країнах. Обидві компанії пропонують послуги з перевезення пасажирів за допомогою мобільних додатків, які дозволяють користувачам швидко і зручно замовляти поїздки.

Uber (Рис 1.2) був заснований у 2009 році і швидко став глобальним лідером у сфері ридшерінгу. Компанія надає широкий спектр послуг, від стандартних поїздок на автомобілях до преміум-сервісів, таких як UberBLACK, а також спільних поїздок через UberPOOL. Uber також експериментує з іншими видами транспорту, включаючи електровелосипеди та самокати (через сервіс JUMP), а також доставку їжі (UberEats).

Рисунок 1.2 - Uber

Сильні сторони Uber:

- Uber доступний у більш ніж 900 містах по всьому світу.
- Користувачі можуть легко замовити поїздку через мобільний додаток, а також отримувати розрахунки вартості та відстежувати автомобіль у режимі реального часу.
- Різноманітність послуг, від економ-класу до преміум-послуг та спільних поїздок.

Слабкі сторони Uber:

- Вартість поїздок може бути високою, особливо під час пікових годин.
- Uber часто стикається з регуляторними проблемами та протестами з боку традиційних таксі-служб у різних країнах.

Lyft (Рис 1.3) був заснований у 2012 році і є основним конкурентом Uber в Північній Америці. Lyft також пропонує широкий спектр послуг, включаючи стандартні поїздки, преміум-сервіси (Lyft Lux) і спільні поїздки (Lyft Shared). Компанія зосереджується на створенні дружньої та безпечної платформи для користувачів та водіїв.

Рисунок 1.3 - Lyft

Сильні сторони Lyft:

- Сильна клієнтоорієнтованість. Lyft часто отримує позитивні відгуки за підтримку клієнтів та відношення до водіїв.
- Інноваційність. Компанія активно впроваджує нові технології та послуги, такі як велосипеди та самокати.
- Часто пропонує знижки та акції для нових та постійних клієнтів.

Слабкі сторони Lyft:

- Lyft здебільшого зосереджується на ринку США та Канади, на відміну від Uber, який має ширше міжнародне покриття.
- Lyft постійно змагається з Uber за частку ринку, що може обмежувати його можливості для розширення та інновацій.

1.4.3 BlaBlaBus, FlixBus

BlaBlaBus та FlixBus - це два популярних автобусних оператори, які пропонують міжміські та міжнародні перевезення, забезпечуючи користувачів доступним і зручним способом подорожувати на великі відстані. Обидві компанії активно розвиваються на європейському ринку, пропонуючи широкий спектр послуг та маршрути в багатьох країнах.

BlaBlaBus (Рис 1.4) є дочірньою компанією BlaBlaCar, відомої платформи для пошуку попутників. Запущена у 2019 році, BlaBlaBus швидко завоювала популярність завдяки своїм доступним цінам і зручним маршрутам.

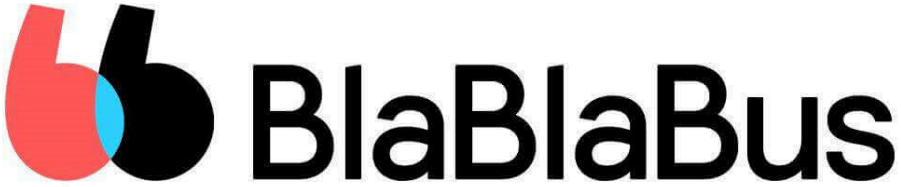

Рисунок 1.4 - BlaBlaBus

Сильні сторони:

- Компанія пропонує конкурентоспроможні ціни на квитки, що робить подорожі доступними для широкого кола користувачів.
- BlaBlaBus забезпечує покриття багатьох напрямків у Європі, що дозволяє користувачам легко планувати подорожі між великими містами.
- Спільне використання платформи з BlaBlaCar дозволяє користувачам комбінувати різні види транспорту для досягнення максимальної гнучкості та зручності.

Слабкі сторони BlaBlaBus:

- Як і інші автобусні сервіси, BlaBlaBus має фіксовані маршрути та розклади, що може не завжди відповідати індивідуальним потребам пасажирів.
- На відміну від BlaBlaCar, BlaBlaBus не пропонує можливості для користувачів знайти попутників для зниження витрат на поїздку.

FlixBus (Рис 1.5) був заснований у 2013 році і швидко став одним з провідних автобусних перевізників у Європі. Компанія відома своїми доступними цінами, зручним бронюванням через мобільний додаток і широким спектром послуг.

Рисунок 1.5 - FlixBus

Сильні сторони FlixBus:

- FlixBus пропонує маршрути у понад 30 країнах Європи та Північної Америки, забезпечуючи велике покриття і зручні пересадки між різними містами.
- Мобільний додаток FlixBus дозволяє легко бронювати квитки, переглядати розклади і відстежувати автобуси в режимі реального часу.
- Компанія відома своїми доступними цінами, що робить її привабливим вибором для бюджетних мандрівників.

Слабкі сторони FlixBus:

- Як і BlaBlaBus, FlixBus має фіксовані маршрути та розклади, що може бути незручним для деяких користувачів.
- FlixBus стикається з конкуренцією з боку залізничного транспорту та авіаліній, які можуть запропонувати швидші подорожі на великі відстані.

Висновки: існуючі готові рішення вже визнані на ринку та мають свої переваги та обмеження. На їхніх прикладах можна вивчити кращі практики та уникнути недоліків при розробці мобільного додатку. Важливо враховувати потреби користувачів та створювати продукт, який буде конкурентоспроможним та забезпечує новаторські функції.

1.5 Постановка задачі

Потрібно розробити додаток для OS Android, де користувач матиме змогу шукати, створювати і бронювати поїздки на потрібну дату та час.

Вхідні дані:

- Інформація для реєстрації;
- Інформація для авторизації;
- Інформація про користувача (профіль);
- Інформація для створення поїздки (водій);
- Інформація для пошуку поїздки;
- Інформація бронювання поїздки (попутник);
- Інформація фільтру;
- Інформація відгуку.

Вихідні дані:

- Дані реєстрації;
- Дані авторизації;
- Список створених поїздок;
- Поїздки, відібрані за вказаними параметрами;
- Список заброньованих поїздок;
- Список користувачів;
- Список відгуків;
- Список автомобілів.

Функції системи:

- Реєстрація;
- Авторизація;
- Створення поїздок водієм;
- Відбір поїздок за параметрами (дата, пункт відправлення/призначення, ціна, кількість вільних місць);
- Перегляд заброньованих поїздок;
- Перегляд деталей поїздки;
- Бронювання поїздок;

- Перегляд профілів;
- Редагування профілю;
- Створення відгуків;
- Перегляд відгуків;
- Налаштування ПЗ.

Більш детальний опис програми представлено в додатку А.

Розділ 2. ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для проектування програмного забезпечення було використано об’єктно-орієнтовану методологію.

Об'єктно-орієнтоване проектування - це методологія проектування, що поєднує процес об'єктної декомпозиції і прийоми подання логічної і фізичної, а також статичної і динамічної моделей проектованої системи. Використання об'єктно-орієнтованої методології зараз нерозривно пов'язано з використанням мови UML (Unified Modeling Language - уніфікована мова моделювання), що являє собою систему позначень, яка базується на діаграмах і призначена для моделювання систем на основі об'єктно-орієнтованого підходу.

Об'єктно-орієнтоване проектування (ООП) відіграє ключову роль у сучасній розробці програмного забезпечення, принесши із собою значні зрушення у способі мислення про створення програм. Запропоноване в 1960-х роках, ООП швидко стало домінуючим підходом у світі програмування.

Основна ідея ООП полягає в тому, щоб розглядати програму як сукупність об'єктів, які мають властивості та можуть взаємодіяти між собою. Кожен об'єкт є екземпляром певного класу і має власний стан та поведінку. ООП дозволяє моделювати реальний світ у програмі, що спрощує розробку та робить код більш зрозумілим і повторно використовуваним.

Основні принципи ООП - інкапсуляція, наслідування та поліморфізм - на яких ґрунтується об'єктно-орієнтований підхід. Інкапсуляція дозволяє об'єднати дані та методи, які їх обробляють, в один об'єкт, наслідування дозволяє створювати нові класи на основі існуючих і поліморфізм дозволяє об'єктам реагувати на методи відповідним чином залежно від контексту виклику. Вивчення та розуміння цих принципів дозволяє розробникам писати більш читабельний, ефективний та підтримуваний код, що є ключем до успішної розробки програмного забезпечення.

Використання мови моделювання UML є однією з ключових складових об'єктно-орієнтованого проектування та сприяє вдосконаленню процесу розробки програмного забезпечення.

Мова моделювання є потужним інструментом для візуалізації, структурування та опису системного проектування в об'єктно-орієнтованій парадигмі. Використання UML дозволяє розробникам чітко уявити архітектуру та взаємозв'язки між компонентами системи, а також детально описати її функціональність та поведінку. Завдяки різноманітним типам діаграм UML, таким як діаграми класів, послідовностей та варіантів використання, розробники можуть ефективно спілкуватися між собою та з зацікавленими сторонами, покращуючи розуміння, документацію та якість програмного забезпечення в цілому.

Приклади практичного застосування об'єктно-орієнтованого проектування широко представлені в різних галузях розробки програмного забезпечення такі як:

- Банківська система

У банківській сфері ООП використовується для розробки систем управління клієнтськими рахунками, обробки транзакцій та ведення баз даних клієнтів. Кожен клієнт може бути представлений як об'єкт класу «Клієнт», що має свої власні атрибути та методи

- Системи управління проектами

У програмах для управління проектами, таких як Jira або Trello, ООП дозволяє створювати об'єкти для представлення завдань, проектів, користувачів та команд. Кожен об'єкт може мати свої властивості та методи для управління ним.

- Інтернет-магазини

У веб-розробці для створення інтернет-магазинів використовується ООП для моделювання товарів, замовлень, користувачів та корзини покупок. Кожен товар може бути представлений як об'єкт класу «Товар», а кожен замовлення - як об'єкт класу «Замовлення».

- Графічні редактори

У графічних редакторах, таких як Adobe Photoshop або GIMP, ООП використовується для моделювання об'єктів, таких як шари, фігури та тексти.

Кожен об'єкт на полотні може бути представлений як окремий об'єкт класу з відповідними методами для редагування та маніпулювання.

- Моделювання транспортних систем

У програмах для моделювання транспортних систем, таких як Simulink або AnyLogic, ООП дозволяє створювати об'єкти для представлення транспортних засобів, маршрутів та пасажирів. Кожен засіб транспорту може бути представлений як об'єкт класу «Транспорт», а кожен пасажир - як об'єкт класу «Пасажир».

Ці приклади ілюструють широкий спектр застосувань об'єктно-орієнтованого проектування у різних сферах індустрії та розробки програмного забезпечення.

2.1 **Ескізний проект**

Ескізний проект – це етап, з якого починається проектування будь-якого програмного забезпечення, у якому представляються результати зовнішнього проектування програмного забезпечення. Його мета - це створення загального плану або концепції програми, яка буде розроблена. На цьому етапі зазвичай не деталізуються технічні деталі або реалізаційні аспекти програми. Замість цього увага приділяється визначенню функціональності, інтерфейсів користувача та основних характеристик програми.

Результати ескізного проекту включають такі елементи:

- **Опис функціональності** – визначення основних функцій та можливостей програми. Він включає список основних функцій, які програма повинна виконувати, та короткий опис кожної з них.
- **Прототип інтерфейсу користувача** – створення простого прототипу інтерфейсу користувача, який демонструє, як програма буде взаємодіяти з користувачем. Це може бути малюнок або макет екранів програми.
- **Визначення основних модулів** – ідентифікація основних компонентів або модулів, які будуть потрібні для реалізації функціональності програми. Це допомагає зрозуміти, як програма буде організована і які будуть основні

компоненти.

В цілому, ескізний проект надає загальне уявлення про те, яким буде програмне забезпечення, і визначає основні вимоги до нього. Це важливий крок у забезпеченні того, що розробка програми рухається у правильному напрямку від самого початку.

При проектуванні даного програмного забезпечення було обрано мову моделювання UML.

2.1.1 Контекстна діаграма

Контекстна діаграма - це початкова діаграма моделі, що характеризує зв’язки системи з навколишнім середовищем.

Контекстна діаграма для програмної системи представлена на рисунку 2.1.

Рисунок 2.1 – Контекстна діаграма програмної системи

2.1.2 Діаграма варіантів використання

Діаграма варіантів використання – діаграма, на якій зображено відношення між акторами та прецедентами в системі. Діаграма варіантів використання для програмної системи представлена на рисунку 2.2.

Рисунок 2.2 – Діаграма варіантів використання для програмної системи

Розглянемо більш детально кожний варіант використання, надаючи опис потоків подій. Специфікації варіантів використання представлено в таблицях 2.1-2.14.

Таблиця 2.1 – Специфікація прецеденту «Реєстрація»

Реєстрація	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу зареєструватися у системі
Передумови	Не визначені
Основний потік подій	Користувач ініціює відкриття інтерфейсу реєстрації. Система відкриває інтерфейс реєстрації. Користувач заповнює необхідні поля інтерфейсу, система перевіряє валідність введених даних і створює новий акаунт. Якщо дані невалідні – виконується альтернативний потік подій.
Альтернативний потік подій	Система повідомляє користувачу, про невалідність введених даних. Дійовій особі пропонується можливість повторити введення або завершити варіант використання.
Постумови	Не визначені

Таблиця 2.2 – Специфікація прецеденту «Авторизація»

Авторизація	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу авторизуватися в системі
Передумови	Виконано прецедент «Реєстрація»
Основний потік подій	Користувач ініціює відкриття інтерфейсу для авторизації, система відкриває інтерфейс. Користувач вводить адресу електронної пошти та пароль, система перевіряє вірність введених даних та допускає до акаунту. Якшо дані невірні – виконується альтернативний потік подій.
Альтернативний потік подій	Дійовій особі пропонується можливість повторити введення або завершити варіант використання
Постумови	Не визначені

Таблиця 2.3 – Специфікація прецеденту «Редагування профілю»

Редагування профілю	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу редагувати свій профіль
Передумови	Виконано прецедент «Авторизація»
Основний потік подій	Користувач ініціює відкриття інтерфейсу для редагування профілю, система відкриває інтерфейс. Користувач вводить нові дані в поля, які необхідно редагувати. Система перевіряє

	валідність введених даних та оновлює редаговану інформацію. Якщо дані невірні - виконується альтернативний потік подій.
Альтернативний потік подій	Дійовій особі пропонується можливість повторити введення або завершити варіант використання
Постумови	Не визначені

Таблиця 2.4 – Специфікація прецеденту «Пошук поїздок»

Пошук поїздок	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу шукати доступні поїздки
Передумови	Виконано прецедент «Авторизація».
Основний потік подій	Користувач ініціює відкриття інтерфейсу пошуку поїздки. Система відкриває відповідний інтерфейс. Користувач обирає потрібний час, місце відправлення та місце прибуття. Система відбирає дані. Результат відбору відображається на інтерфейсі.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.5 – Специфікація прецеденту «Перегляд деталей поїздки»

Перегляд деталей поїздки	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє переглядати користувачу деталі обраної поїздки
Передумови	Виконано прецедент «Пошук поїздок»
Основний потік подій	Користувач ініціює відкриття інтерфейсу перегляду деталей поїздки. Система відкриває відповідний інтерфейс
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.6 – Специфікація прецеденту «Перегляд профілей»

Перегляд профілей	
Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу переглядати профілі інших користувачів
Передумови	Виконано прецедент «Перегляд деталей поїздки»
Основний потік подій	Користувач ініціює відкриття інтерфейсу перегляду профілю іншого користувача. Система відкриває інтерфейс перегляду профілю.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.7 – Специфікація прецеденту «Сортування поїздок»

Сортування поїздок	
Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу сортувати поїздки
Передумови	Виконано прецедент «Пошук поїздок».
Основний потік подій	Користувач ініціює сортування поїздок за датою, типом транспорту або ціною. Система відображає відсортовані поїздки.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.8 – Специфікація прецеденту «Бронювання поїздок»

Бронювання поїздок	
Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу бронювати поїздки
Передумови	Виконання прецеденту «Перегляд деталей поїздки»
Основний потік подій	Користувач ініціює бронювання обраної поїздки. Система додає поїздку у список заброньованих поїздок.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.9 – Специфікація прецеденту «Перегляд заброньованих поїздок»

Перегляд заброньованих поїздок	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати список заброньованих поїздок
Передумови	Виконання прецеденту «Бронювання поїздок»
Основний потік подій	Користувач ініціює відкриття інтерфейсу перегляду списку заброньованих ним поїздок. У разі наявності заброньованих користувачем поїздок система відображає їх. Якщо користувач не забронював жодної поїздки виконується альтернативний потік подій.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.10 – Специфікація прецеденту «Додати авто»

Додати авто	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу додати автомобіль
Передумови	Виконання прецеденту «Редагування профілю» або «Створення поїздки»

Основний потік подій	Користувач ініціює відкриття інтерфейсу для додавання автомобілю у своєму профілі. Система перевіряє вік користувача. Якщо вік 18 і більше, то система відкриває інтерфейс для додавання. В протилежному випадку виконується альтернативний потік подій.
Альтернативний потік подій	Система інформує користувача про те, що він не має прав на керування авто
Постумови	Не визначені

Таблиця 2.11 – Специфікація прецеденту «Створення поїздки»

Створення поїздки	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу створювати поїздки
Передумови	Виконаний прецедент «Додати авто»
Основний потік подій	Користувач ініціює відкриття інтерфейсу створення поїздки. Система відкриває інтерфейс створення поїздки. Користувач заповнює всі необхідні поля для створення поїздки. Система створює нову поїздку.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.12 – Специфікація прецеденту «Створення відгуків»

Створення відгуків	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу створювати відгуки іншим користувачам
Передумови	Виконання прецеденту «Перегляд профілей»
Основний потік подій	Користувач ініціює відкриття інтерфейсу створення відгуку у профілі іншого користувача. Система відкриває інтерфейс створення відгуку. Користувач ставить оцінку и пише коментар. Система додає залишений відгук до списку відгуків.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.13 – Специфікація прецеденту «Перегляд відгуків»

Перегляд відгуків	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу переглядати відгуки у своєму профілі та у профілях інших користувачів
Передумови	Виконання прецеденту «Перегляд профілей» або «Редагування профілю»

Основний потік подій	Користувач ініціює відкриття інтерфейсу перегляду відгуків. Система відображає відкритий інтерфейс.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

Таблиця 2.14 – Специфікація прецеденту «Налаштування»

Налаштування	
Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє користувачу виконувати налаштування для додатку
Передумови	Виконання прецеденту «Авторизація»
Основний потік подій	Користувач ініціює відкриття інтерфейсу налаштувань системи. Система відкриває інтерфейс налаштувань. Користувач виконує потрібні йому налаштування в додатку. Система зберігає виконані налаштування.
Альтернативний потік подій	Не визначено
Постумови	Не визначені

2.1.3 Концептуальна модель

Концептуальна модель — модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

ER-діаграма для представлення структури даних представлена на рисунку 2.3.

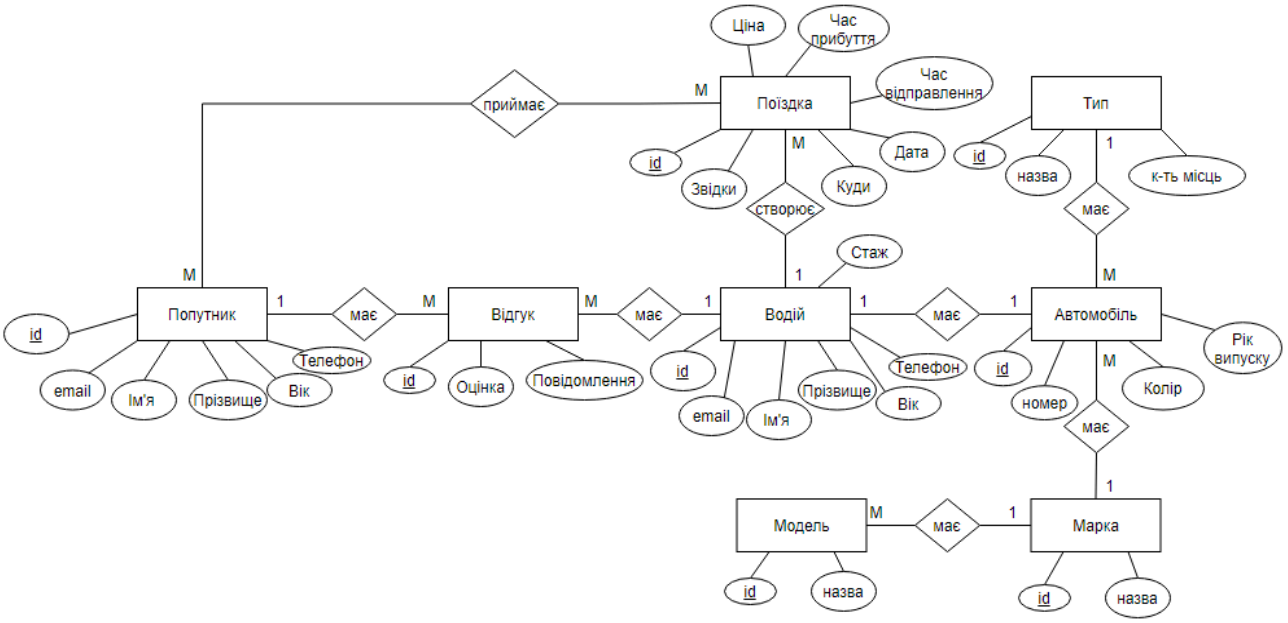

Рисунок 2.3 - ER-діаграма для представлення структури даних

Опис зв’язків між сутностями концептуальної моделі для програмної системи показаний у таблиці 2.15.

Таблиця 2.15 – Таблиця зв’язки між сутностями

Сутності	Тип зв’язку	Зміст зв’язку
1	2	3
Попутник Відгук	1:M	Один попутник може мати декілька відгуків
Попутник Поїздка	M:M	Багато попутників можуть приймати декілька поїздок
Водій Відгук	1:M	Один водій може мати декілька відгуків
1	2	3
Водій Поїздка	1:M	Один водій може створювати декілька поїздок
Водій Автомобіль	1:1	Один водій має один автомобіль
Тип Автомобіль	1:M	Один тип має декілька автомобілів
Марка Автомобіль	1:M	Одна марка має декілька автомобілів
Марка Модель	1:M	Одна марка має декілька моделей

2.1.4 Діаграма станів

Діаграма станів – це діаграма, що показує, як об'єкт переходить з одного стану в інший. Діаграма станів служить для моделювання динамічних аспектів системи, також корисна при моделюванні життєвого циклу об'єкта. Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни станів об'єкта, поведінка якого характеризується його реакцією на зовнішні події.

Перехід між підсистемами здійснюватиметься через головне вікно, частина якого буде активна незалежно від обраної підсистеми і інших об’єктів конфігурації. Діаграма станів для програмної системи представлена на рисунку 2.4.

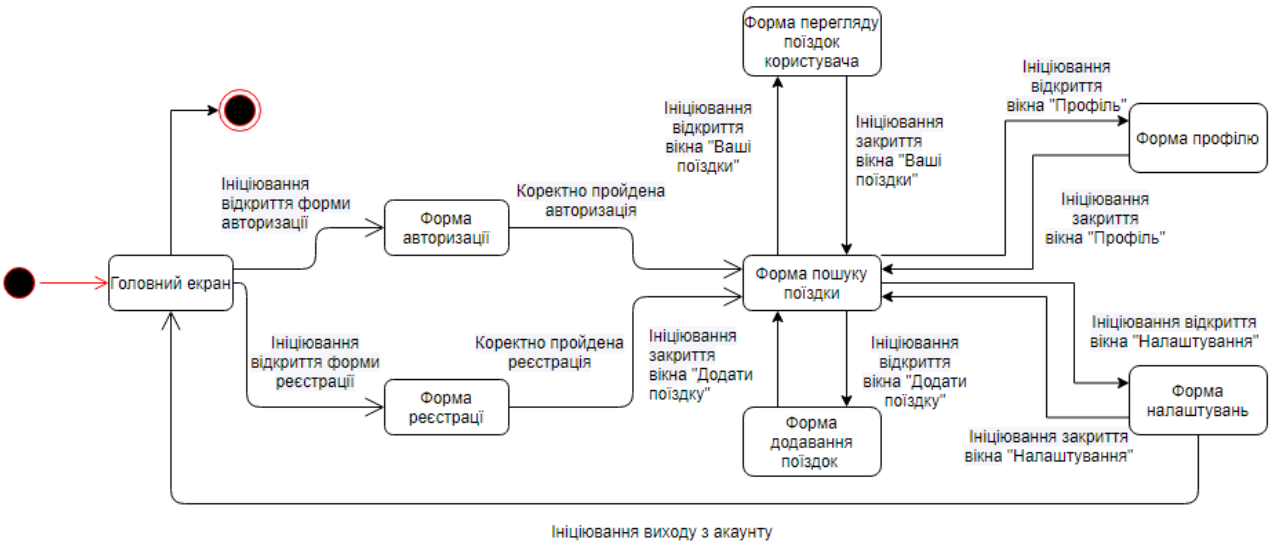

Рисунок 2.4 - Діаграма станів для переходу між формами

2.1.5 Проектування інтерфейсу

Інтерфейс користувача – засіб зручної взаємодії користувача з інформаційною системою; сукупність засобів для обробки та відображення інформації, максимально застосованих для зручності користувача.

Макет головного вікна представлений на рисунку 2.5.

Рисунок 2.5 – Макет головного вікна

Макет вікна пошуку поїздки представлений на рисунку 2.6.

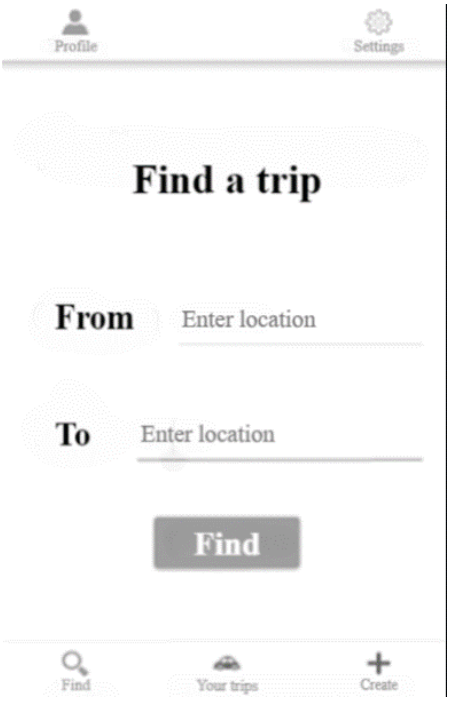

Рисунок 2.6 – Макет вікна пошуку поїздки

Макет вікна профілю користувача представлений на рисунку 2.7.

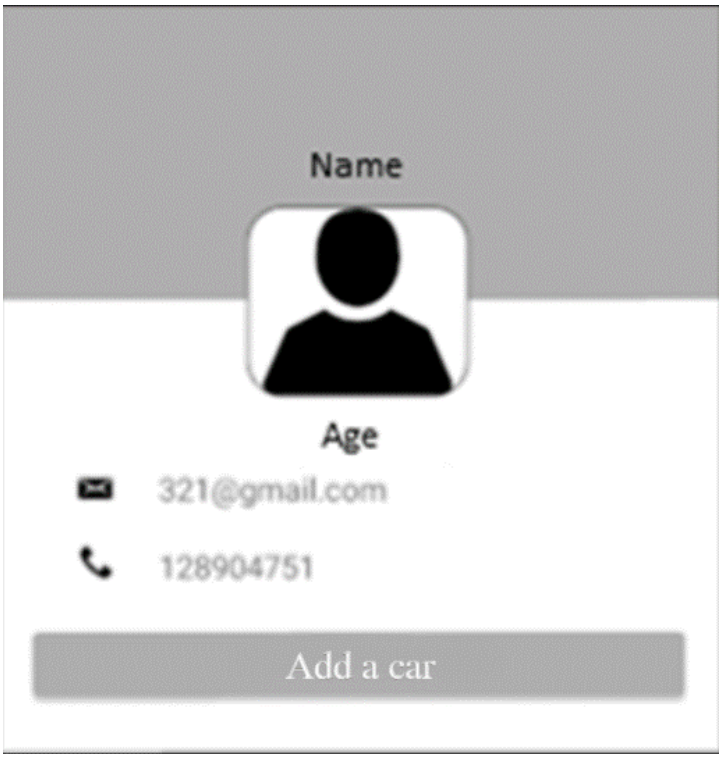

Рисунок 2.7 – Макет вікна профілю користувача

Макет вікна авторизації представлений на рисунку 2.8.

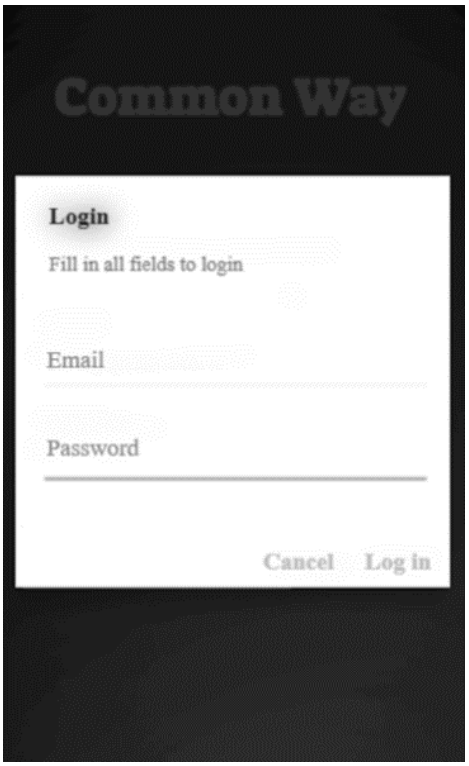

Рисунок 2.8 - Макет вікна авторизації

Макет вікна реєстрації представлений на рисунку 2.9.

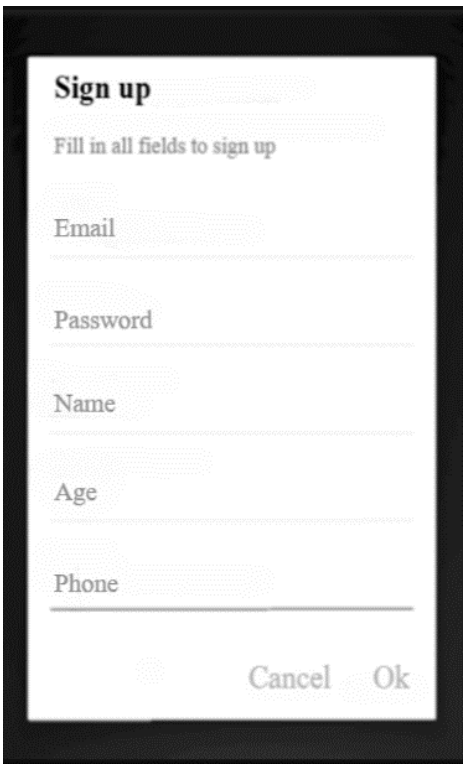

Рисунок 2.9 - Макет вікна реєстрації

Макет вікна заброньованих поїздок користувача представлений на рисунку 2.10.

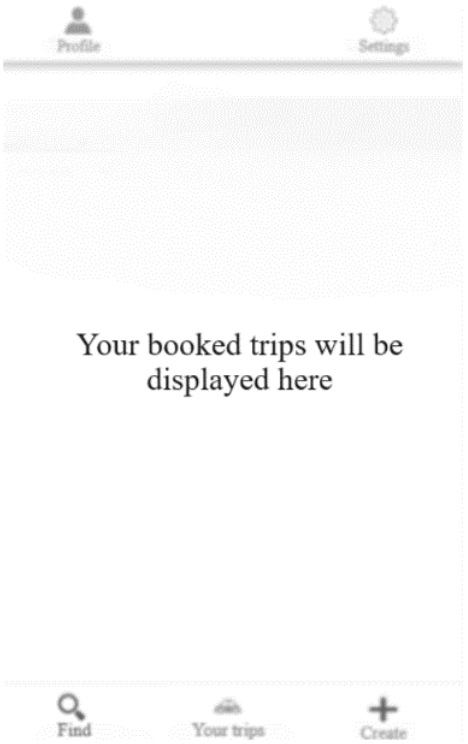

Рисунок 2.10 - Макет вікна заброньованих поїздок

Макет вікна створення поїздок представлений на рисунку 2.11.

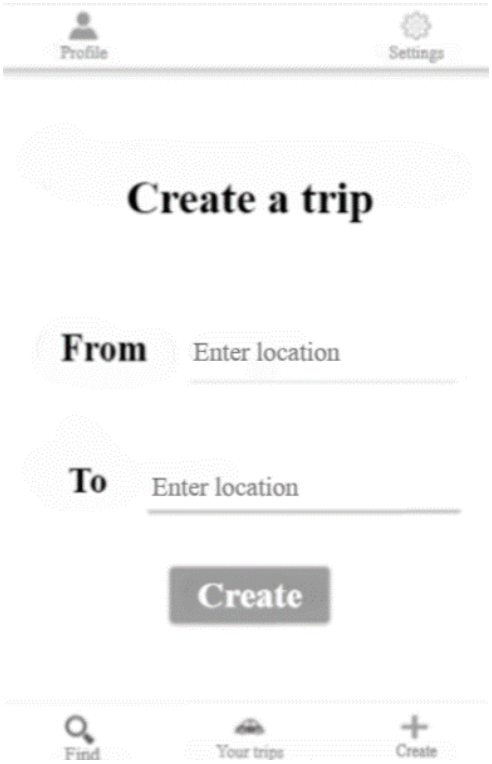

Рисунок 2.11 - Макет вікна створення поїздок

Макет вікна налаштувань представлений на рисунку 2.12.

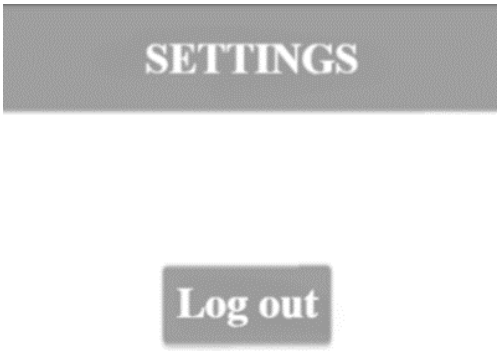

Рисунок 2.12 - Макет вікна налаштувань

2.2 Технічний проект

Технічний проект програмної системи – це етап, який детально описує: виконувані функції і варіанти їх використання, структури оброблюваних баз даних, взаємозв'язки даних та алгоритми їх обробки.

Технічний проект є наступним етапом у процесі розробки програмного забезпечення, після ескізного проекту. На цьому етапі подробно описуються всі технічні аспекти програми, включаючи її функціональність, структури даних, бази даних та алгоритми обробки даних.

Технічний проект детально визначає всі технічні аспекти програми та її реалізацію, що допомагає розробникам зрозуміти, як програму слід буде створити та як вона буде працювати в реальному середовищі.

2.2.1 Логічна модель

Фаза логічного проектування полягає в перетворенні концептуальної моделі даних у логічну. Для розробки програмного забезпечення було обрано реляційну модель. Логічна модель відображає взаємозв'язки між реляційними таблицями.

Логічна модель структури даних представлена на рисунку 2.13.

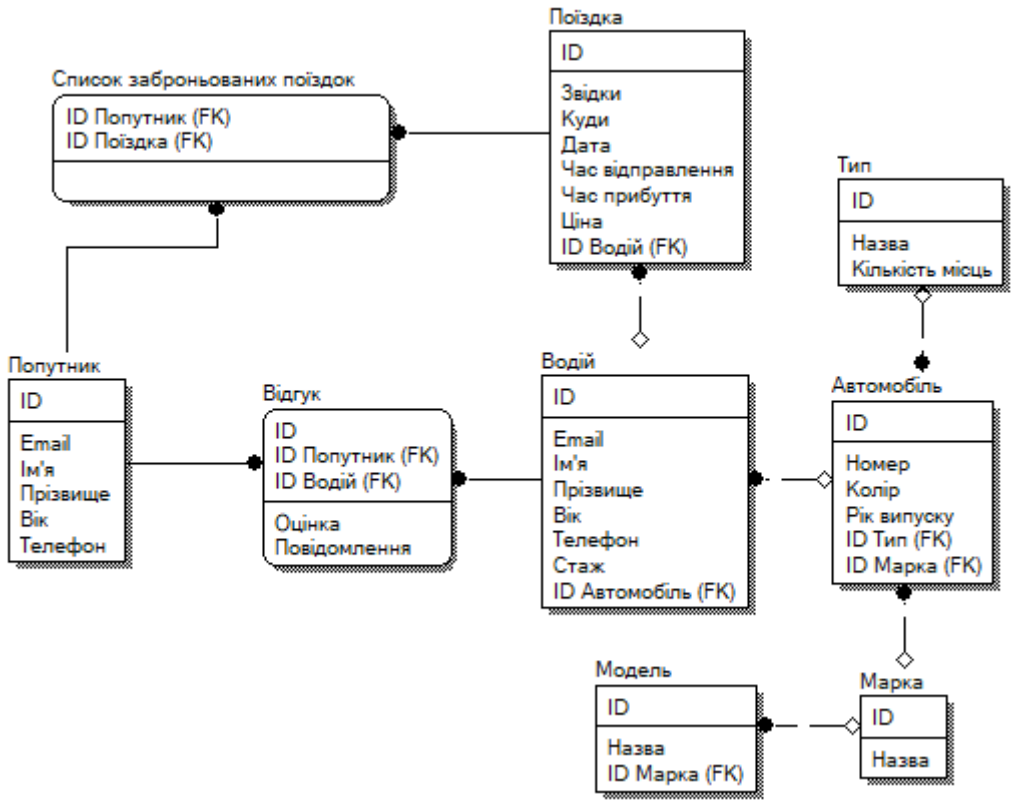

Рисунок 2.13 – Логічна модель даних

Для функціоналу авторизації та реєстрації використовується документоорієнтована БД Firebase, яка призначена для зберігання ієрархічних структур даних (документів) і зазвичай реалізована за допомогою підходу NoSQL. В основі документоорієнтованих СУБД лежать документні сховища, що мають структуру дерева.

2.2.2 Діаграми послідовностей

Діаграма послідовності відображає взаємодію об'єктів впорядкованих за часом, зокрема, відображає задіяні об'єкти та послідовність відправлених повідомлень.

Діаграма послідовності для основного потоку подій прецеденту «Реєстрація» представлена на рисунку 2.13.

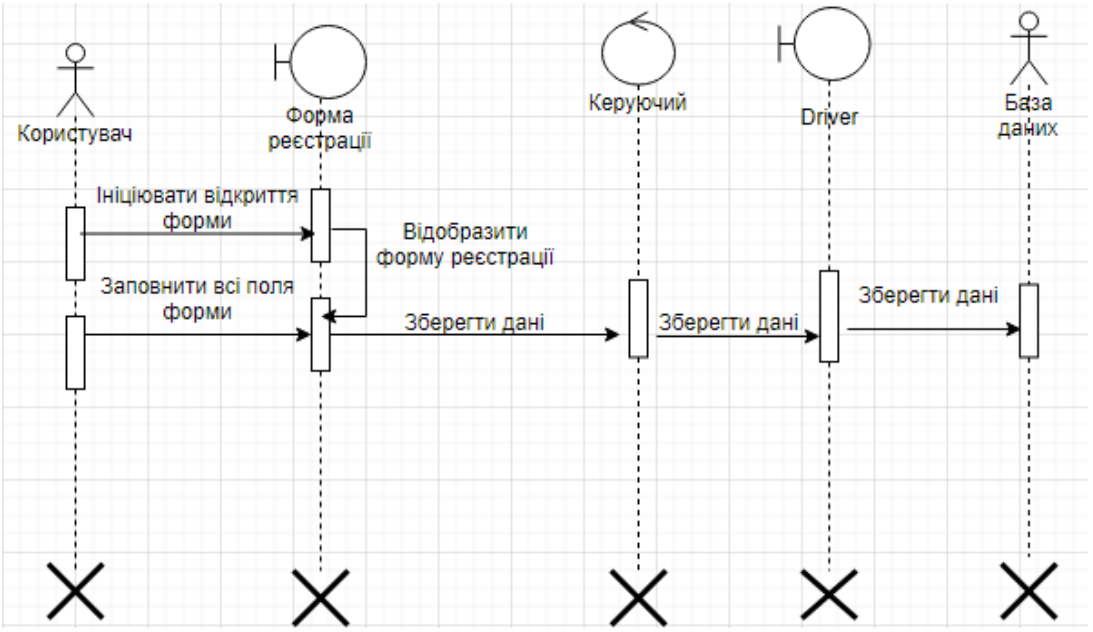

Рисунок 2.13 – Діаграма послідовності для прецеденту «Реєстрація»

Діаграма послідовності для основного потоку подій прецеденту «Пошук поїздки» представлена на рисунку 2.14.

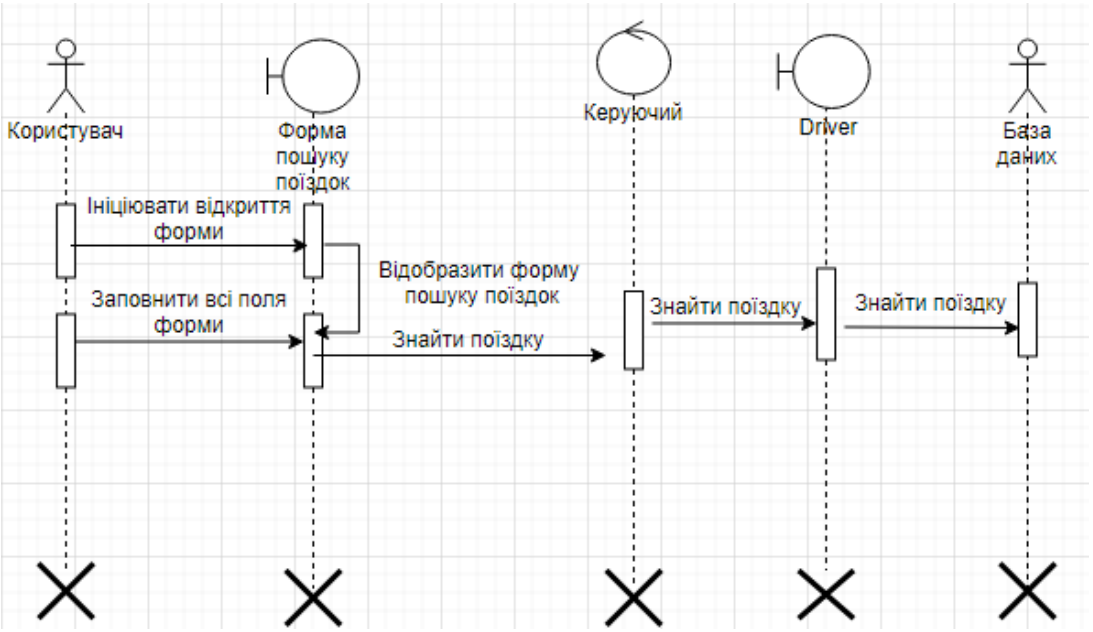

Рисунок 2.14 – Діаграма послідовності для прецеденту «Пошук поїздки»

Діаграма послідовності для основного потоку подій прецеденту «Додати авто» представлена на рисунку 2.15.

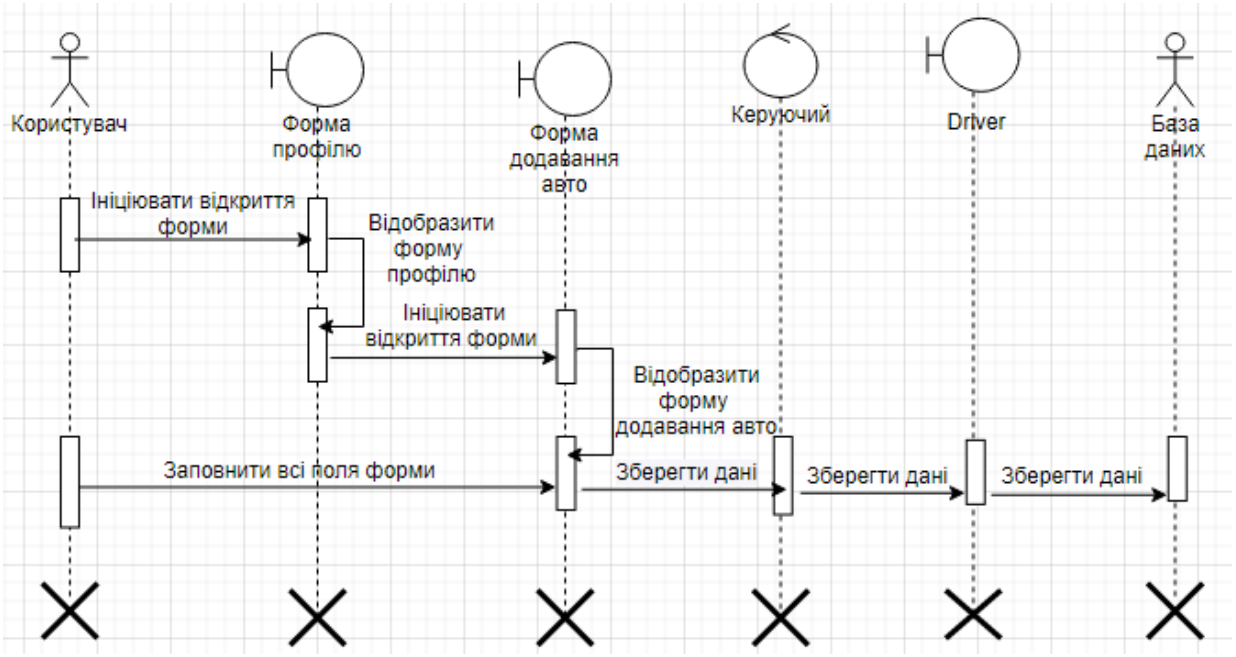

Рисунок 2.15 – Діаграма послідовності для прецеденту «Додати авто»

2.2.3 Діаграма класів

Діаграма класів – діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв'язок між ними. Класи-сутності з атрибутами та методами представлені на рисунку 2.16.

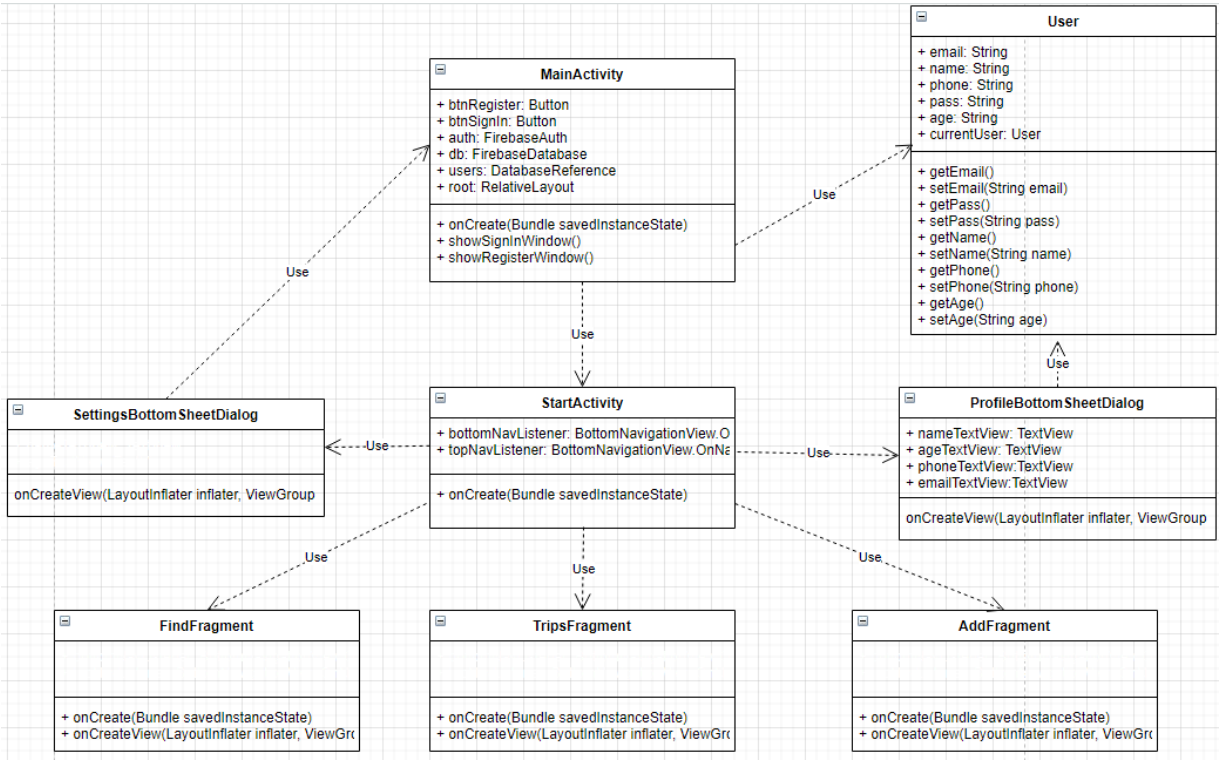

Рисунок 2.16 – Діаграма класів програмної системи

2.2.4 Діаграма діяльності

Діаграма діяльності відображає логіку або послідовність переходу від однієї діяльності до іншої, при цьому увага фіксується на результаті діяльності. Сам же результат може привести до зміни стану системи або повернення деякого значення. Діаграма діяльності для доступу до програмної системи представлена на рисунку 2.17.

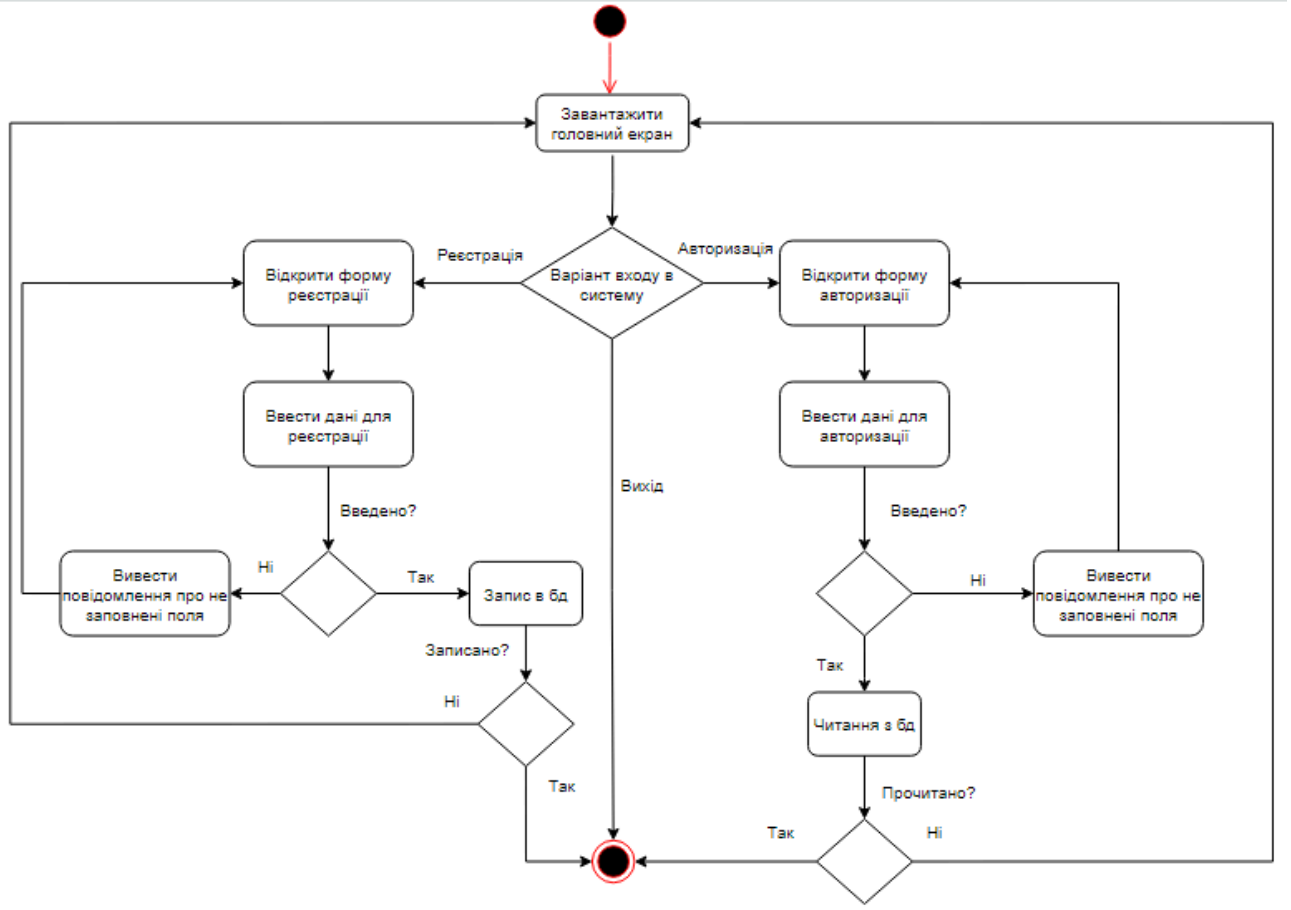

Рисунок 2.17 - Діаграма діяльності для доступу до програмної системи

2.3 Робочий проект

2.3.1 Вибір засобів розробки

В якості засобу розробки дипломного проекту було обрано середовище Android Studio (Рисунок 2.18.). Це інтегроване середовище розробки (IDE) для роботи з платформою Android. IDE перебувала у вільному доступі. Android Studio, заснована на програмному забезпеченні IntelliJ IDEA від компанії JetBrains, - офіційний засіб розробки Android-додатків. Дане середовище розробки доступне

для Windows, Mac OS та Linux. Офіційними мовами програмування для платформи Android є Java (Рисунок 2.19), C ++ та Kotlin.

Рисунок 2.18 – Android Studio

Рисунок 2.19 – Java

Середовище розробки адаптоване для виконання типових завдань, що вирішуються в процесі розробки застосунків для платформи Android. У тому числі у середовище включені засоби для спрощення тестування програм на сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Крім можливостей, присутніх в IntelliJ IDEA, в AndroidStudio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Для прискорення розробки застосунків представлена колекція типових елементів інтерфейсу і візуальний редактор для їхнього компонування, що надає зручний попередній перегляд різних станів інтерфейсу застосунку (наприклад,

можна подивитися як інтерфейс буде виглядати для різних версій Android і для різних розмірів екрану). Для створення нестандартних інтерфейсів присутній майстер створення власних елементів оформлення, що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub.

До складу також включені пристосовані під особливості платформи Android розширені інструменти рефакторингу, перевірки сумісності з минулими випусками, виявлення проблем з продуктивністю, моніторингу споживання пам'яті та оцінки зручності використання. У редактор доданий режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API. Інтегрована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрових підписів. Надано інтерфейс для управління перекладами на інші мови.

Java - це високорівнева, об'єктно-орієнтована мова програмування, яка була розроблена компанією Sun Microsystems у 1995 році. Однією з ключових особливостей Java є її платформонезалежність. Код, написаний на Java, компілюється в байт-код, який виконується на віртуальній машині Java (JVM). Це дозволяє запускати Java-програми на будь-якій платформі, яка має встановлену відповідну JVM, без потреби змінювати вихідний код.

Ще однією важливою особливістю Java є її безпека. Java використовує механізми безпеки, такі як контроль доступу до класів та віртуальна машина для контролю доступу до ресурсів. Це дозволяє запобігти багатьом типам атак, таких як витоки пам'яті або переповнення буфера. Також вона має багатофункціональну стандартну бібліотеку, яка включає в себе інструменти для роботи з мережами, базами даних, роботи з файлами, графікою та інші. Це дозволяє розробникам швидко створювати різноманітні програми без необхідності використання сторонніх бібліотек або інструментів.

Крім того, Java є однією з найпопулярніших мов програмування в світі. Вона використовується для розробки різноманітних програм, включаючи мобільні додатки для Android, веб-додатки, корпоративні системи, ігри та багато іншого.

Java має велику спільноту розробників та активно розвивається, що робить її привабливим вибором для багатьох проектів різного масштабу і складності.

Кожна програма для Android може створювати і використовувати БД.

Firebase - ц е інтегрована платформа від Google, яка надає розробникам зручні інструменти для роботи з різними аспектами розробки мобільних та веб-додатків. Одним із ключових компонентів Firebase є його хмарна база даних, яка є відмінним рішенням для зберігання та синхронізації даних у реальному часі. (Рисунок 2.20).

Рисунок 2.20 – Firebase

Основна перевага Firebase полягає в його простоті використання та масштабованості. Розробники можуть легко зберігати та отримувати дані за допомогою зручного API, що значно спрощує роботу з базою даних. Крім того, Firebase надає можливості автоматичної синхронізації даних між різними пристроями, що дозволяє користувачам отримувати доступ до оновлених даних в реальному часі.

Додатково, Firebase пропонує розширені можливості, такі як аутентифікація користувачів, зберігання медіафайлів за допомогою Cloud Storage, аналітика взаємодії з додатком та багато іншого. Ці функції дозволяють розробникам створювати додатки, які не лише забезпечують ефективну роботу з даними, але й надають користувачам різноманітні можливості та функціонал.

Firebase має велику перевагу у вигляді миттєвої синхронізації даних. Будь-які зміни, внесені в базу даних на одному пристрої, негайно відображаються на всіх інших пристроях. Це забезпечує однорідний та актуальний стан даних для всіх користувачів додатку в реальному часі. Такий підхід дозволяє створювати додатки, які динамічно реагують на зміни та забезпечують зручний та надійний досвід для користувачів

SQLite (Рис.2.21) — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92.

Рисунок 2.21 – SQLite

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

Room - це сучасний спосіб зберігання даних у додатках для Android, який є частиною нової архітектури Android, розробленої компанією Google. Ця архітектура включає групу бібліотек, що підтримують модульний і

структурований підхід до розробки додатків, сприяючи кращій організації коду та зручності його підтримки.

Room виступає як високорівневий інтерфейс для роботи з низькорівневими прив'язками SQLite, які вбудовані в Android. Це означає, що Room забезпечує більш простий та зручний спосіб взаємодії з базою даних SQLite, абстрагуючи складність прямої роботи з API SQLite. Використовуючи Room, розробникам більше не потрібно безпосередньо працювати з Cursor або ContentResolver, оскільки Room автоматизує багато завдань, пов'язаних з управлінням базою даних.

Однією з ключових переваг Room є те, що він виконує більшість своєї роботи під час компіляції, створюючи API поверх вбудованого SQLite API. Це не лише спрощує процес розробки, але й зменшує кількість помилок, пов'язаних з написанням SQL-запитів вручну. Room забезпечує всі необхідні можливості для виконання складних запитів та управління базою даних, зменшуючи необхідність писати SQL-запити самостійно.

Таким чином, Room пропонує розробникам ефективний і простий спосіб роботи з базами даних в Android-додатках, забезпечуючи при цьому високу продуктивність та надійність. Він є відмінною альтернативою іншим ORM-рішенням, таким як Realm, ORMLite або GreenDao, поєднуючи простоту використання з потужними можливостями для управління даними.

2.3.2 Обґрунтування вибору інструментарію

Успішна розробка програмного забезпечення потребує ретельного вибору інструментарію, що включає середовище програмування, сховище даних та мову програмування. Розглянемо критерії вибору кожного з цих компонентів.

Критеріями вибору середовища програмування є:

- Інтеграція з іншими інструментами

Середовище програмування повинне добре інтегруватися з іншими інструментами та бібліотеками, які використовуються в проекті. Наприклад, Android Studio забезпечує відмінну інтеграцію з Gradle, Firebase, Git та іншими інструментами.

- Зручність використання

Важливо, щоб середовище програмування було зручним та інтуїтивно зрозумілим для розробників. Android Studio надає багатий набір інструментів для розробки, відлагодження та тестування додатків, що значно спрощує роботу розробників.

- Підтримка сучасних технологій

Середовище повинне підтримувати сучасні технології та стандарти програмування. Android Studio регулярно оновлюється та забезпечує підтримку нових версій Android SDK, що дозволяє використовувати останні можливості платформи.

- Документація та спільнота

Велика спільнота розробників та наявність якісної документації дозволяють швидко знаходити відповіді на питання та вирішувати проблеми, що виникають під час розробки.

- Можливість візуального програмування

Android Studio надає розробникам інструменти для візуального програмування, такі як Layout Editor, який дозволяє створювати інтерфейси користувача шляхом перетягування компонентів на екран. Це значно спрощує процес розробки та тестування UI, дозволяючи розробникам швидко бачити результати своїх змін.

- Використання системи контролю версіями Git

Android Studio має вбудовану підтримку системи контролю версій Git, що дозволяє розробникам легко керувати змінами у вихідному коді, співпрацювати з іншими розробниками та відслідковувати історію змін. Це забезпечує зручність в управлінні проектом та допомагає запобігати втраті даних.

- Відкритий код

Android Studio є відкритим проектом з відкритим вихідним кодом, що забезпечує прозорість та можливість внесення змін і покращень спільнотою

розробників. Відкритість коду також дозволяє розробникам краще розуміти внутрішню роботу середовища та налаштовувати його під свої потреби.

- Безкоштовність

Android Studio є безкоштовним інструментом, що робить його доступним для широкого кола розробників, включаючи студентів, незалежних розробників та малі

- Зручність інтерфейсу

Android Studio має інтуїтивно зрозумілий та зручний інтерфейс, що спрощує навігацію та використання різних інструментів. Висока ступінь налаштовуваності дозволяє розробникам налаштовувати середовище під свої потреби та уподобання, що підвищує ефективність роботи.

Критерії вибору сховища даних:

- Тип даних та обсяг

Firebase підходить для зберігання структурованих даних в реальному часі та забезпечує масштабованість для великих обсягів даних. SQLite є оптимальним вибором для локального зберігання невеликих обсягів даних, таких як налаштування додатка або кешовані дані.

- Масштабованість

Firebase забезпечує високу масштабованість завдяки хмарній архітектурі, що дозволяє обробляти великі обсяги даних та підтримувати велику кількість користувачів. SQLite, хоча і не є масштабованим у хмарі, забезпечує ефективну роботу з локальними даними на пристрої.

- Продуктивність

Firebase оптимізований для роботи з хмарними даними в реальному часі, забезпечуючи високу швидкість доступу та синхронізації даних між клієнтами. SQLite забезпечує швидкий доступ до локально збережених даних на пристрої, що робить його відмінним вибором для роботи з даними, які не потребують постійної синхронізації.

- Безпека

Firebase пропонує вбудовані механізми аутентифікації та авторизації, що допомагають забезпечити безпеку даних користувачів. SQLite, з іншого боку, залежить від механізмів безпеки пристрою, на якому працює додаток, але може бути використаний разом з іншими інструментами безпеки для захисту даних.

- Безкоштовність

Firebase та SQLite є безкоштовними для використання в багатьох випадках. Firebase пропонує безкоштовний тарифний план, який забезпечує достатні ресурси для розробки та тестування додатків, а також платні плани для великих проектів. SQLite також є безкоштовним і відкритим програмним забезпеченням, що робить його доступним для всіх розробників без додаткових витрат

Критерії обрання мови програмування:

- Кроссплатформеність

Java є платформонезалежною мовою програмування, що дозволяє запускати додатки на будь-якій платформі, яка підтримує JVM. Це забезпечує широкі можливості для розробки кросплатформених додатків.

- Продуктивність

Java забезпечує високу продуктивність виконання коду завдяки компіляції в байт-код та виконанню на JVM. Це дозволяє ефективно використовувати ресурси системи та забезпечувати високу швидкість роботи додатків.

- Безпека

Java має вбудовані засоби безпеки, такі як управління пам'яттю та контроль доступу до класів, що допомагає запобігати поширеним вразливостям. Це робить Java безпечною мовою програмування для розробки різноманітних додатків.

- Підтримка спільноти та ресурси

Java має велику спільноту розробників та багатий набір бібліотек і фреймворків, що спрощує процес розробки та дозволяє швидко знайти рішення для більшості завдань. Доступність великої кількості ресурсів, включаючи документацію, навчальні матеріали та приклади коду, робить Java популярним вибором серед розробників.

- Власні знання мови.

2.3.3 Розробка фізичної моделі

Фізичне проектування бази даних – процес створення опису конкретної реалізації бази даних, розташовуваної у вторинній пам'яті.

Фізичне проектування бази даних є критичним етапом у процесі розробки програмного забезпечення, оскільки воно визначає, як дані будуть зберігатися та оброблятися у реальній системі. Фізична модель бази даних описує структуру збереження даних і методи доступу, що забезпечують найбільш ефективний доступ до інформації.

Фізична модель даних представлена на рисунку 2.23.

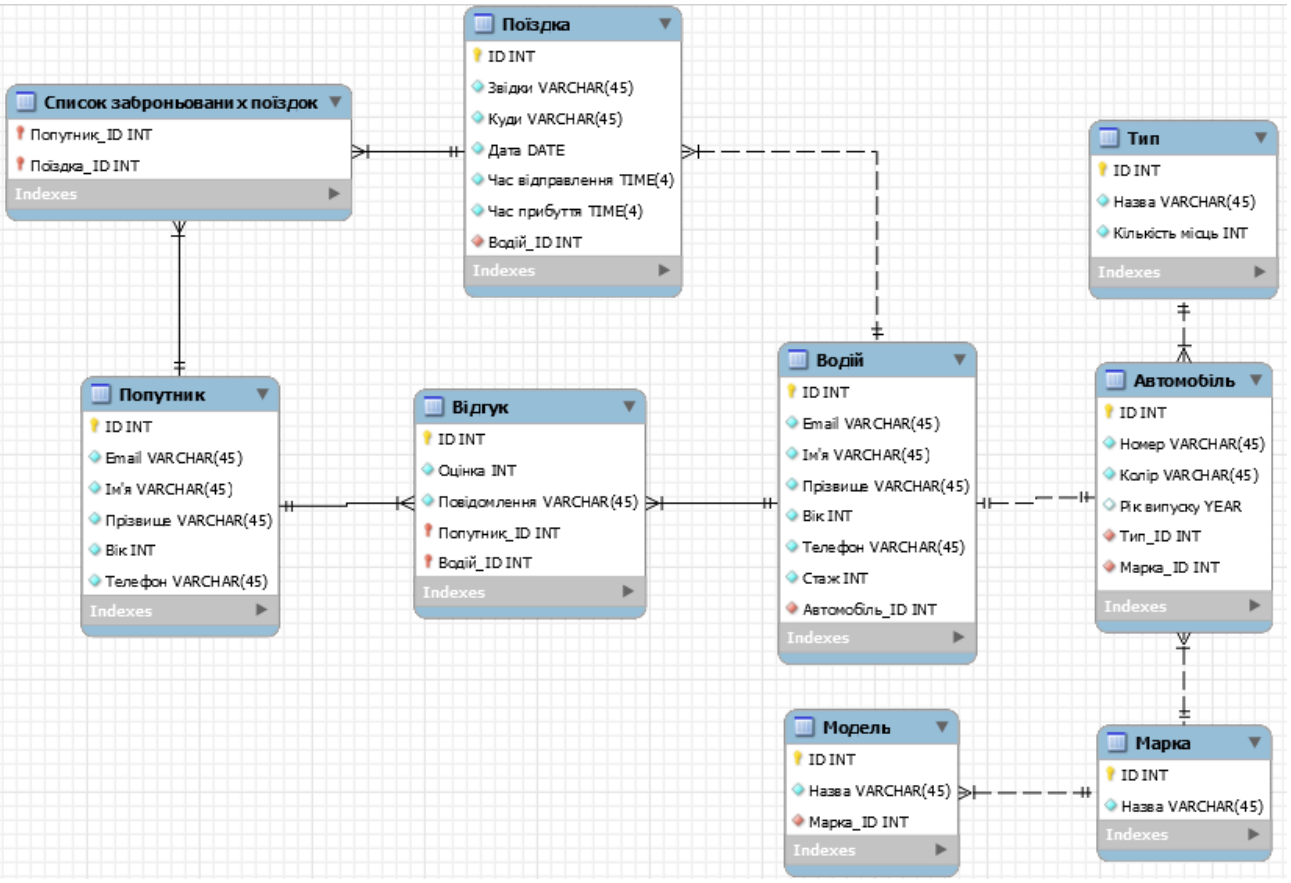

Рисунок 2.23 – Фізична модель даних

В таблицях 2.16 – 2.24 представлено опис структур таблиць, які відображають назву поля, тип даних та обмеження на це поле.

Таблиця 2.16 – Структура таблиці «Companion»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Email		VARCHAR(45)	NOT NULL
Name		VARCHAR(45)	NOT NULL
Surname		VARCHAR(45)	NOT NULL
Age		INTEGER	NOT NULL
Phone		INT	NOT NULL

Таблиця 2.17 – Структура таблиці «Driver»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Email		VARCHAR(45)	NOT NULL
Name		VARCHAR(45)	NOT NULL
Surname		VARCHAR(45)	NOT NULL
Age		INTEGER	NOT NULL
Phone		INT	NOT NULL
Experience		INT	NOT NULL
Car_ID	FK	INT	NOT NULL

Таблиця 2.18 – Структура таблиці «Feedback»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження

1	2	3	4
ID	PK	INTEGER	NOT NULL
Rating		INTEGER	NOT NULL
Message		VARCHAR(45)	NOT NULL
Companion_ID	FK	INTEGER	NOT NULL
Driver_ID	FK	INTEGER	NOT NULL

Таблиця 2.19 – Структура таблиці «Trip»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Whence		VARCHAR(45)	NOT NULL
Whither		VARCHAR(45)	NOT NULL
Date		DATE	NOT NULL
Departure time		TIME(4)	NOT NULL
Arrival time		TIME(4)	NOT NULL
Driver_ID	FK	INTEGER	NOT NULL

Таблиця 2.20 – Структура таблиці «Booked trips»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
Companion_ID	FK	INTEGER	NOT NULL
Trip_ID	FK	INTEGER	NOT NULL

Таблиця 2.21 – Структура таблиці «Car»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Number		VARCHAR(45)	NOT NULL
Color		VARCHAR(45)	NOT NULL
ManufactureYear		YEAR	NOT NULL

Type_ID	FK	INTEGER	NOT NULL
Brand_ID	FK	INTEGER	NOT NULL

Таблиця 2.22 – Структура таблиці «CarType»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Name		VARCHAR(45)	NOT NULL

Таблиця 2.23 – Структура таблиці «CarBrand»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Name		VARCHAR(45)	NOT NULL

Таблиця 2.24 – Структура таблиці «CarModel»

Ідентифікатор поля	Ознака ключа	Тип даних	Обмеження
1	2	3	4
ID	PK	INTEGER	NOT NULL
Name		VARCHAR(45)	NOT NULL
Brand_ID	FK	INTEGER	NOT NULL

Розділ 3. РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Результатом розробки дипломного проекту є мобільний додаток для пошуку попутників. Дане програмне забезпечення являє собою додаток, який було розроблено для OS Android з мінімальною версією 5.0 (Lollipop). Для досягнення поставленої цілі виконано аналіз предметної області, в якому було досліджено роботу інших додатків.

В ескізному проекті побудовано контекстну діаграму, діаграму варіантів використання, концептуальну модель, діаграми переходів станів, розроблено ескізи користувацького інтерфейсу.

В технічному проекті побудовано логічну модель бази даних, розроблено діаграму послідовності, пакетів та класів для варіантів використання програмного забезпечення.

В робочому проекті виконано та аргументовано вибір інструментів, а також побудовано фізичну модель даної бази даних.

Під час роботи над даним додатком було застосовано декілька бібліотек (com.android.support:design:25.0.0, androidx.constraintlayout:constraintlayout:2.0.4, uk.co.chrisjenx:calligraphy:2.3.0), які дали можливість створити зручний та інтуїтивно зрозумілий дизайн.

В результаті розробки було реалізовано функціонал реєстрації та авторизації, клієнтську частину яка складається з таких вікон:

- «Головне вікно»;
- «Реєстрація»;
- «Авторизація»;
- «Пошук поїздки»;
- «Створення поїздки»;
- «Поїздки користувача»;
- «Профіль»;
- «Додавання автомобіля»;

- «Налаштування».

Головний екран (Рис 3.1.) – цей екран є вітальним екраном, який надає користувачам можливість увійти в систему або зареєструвати новий обліковий запис.

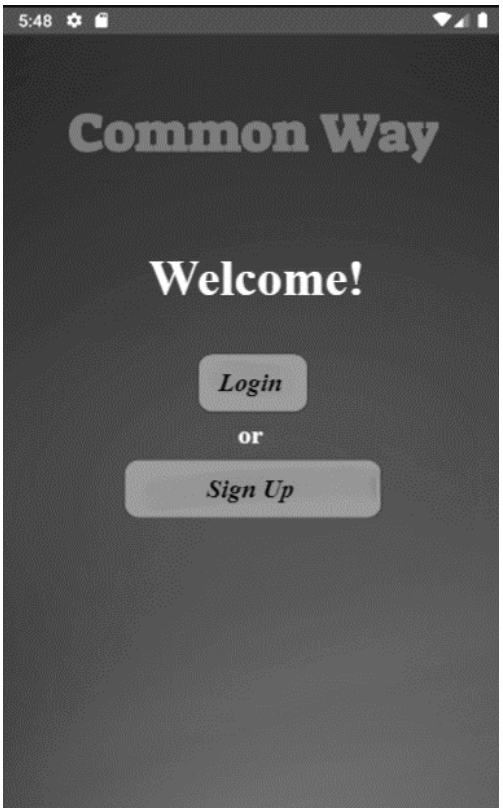

Рисунок 3.1 - Головний екран

Головний екран є початковою точкою взаємодії користувачів з додатком і має інтуїтивно зрозумілий дизайн, що сприяє зручному входу або реєстрації та добре виконує свою функцію, забезпечуючи простий та зручний доступ до основних можливостей додатку.

Екран має дві кнопки:

- **Login –** кнопка для входу в систему.
- **Sign Up –** кнопка для реєстрації нового облікового запису.

Форма авторизації (Рис 3.2) – наступний важливий елемент додатку, форма є частиною процесу автентифікації користувача, коли користувачі повинні ввести свою електронну адресу та пароль, щоб увійти в додаток. Дизайн форми простий і

зосереджений на основних елементах, необхідних для виконання функції входу. Форма складається з двох полів і двох кнопок:

- **Email** – поле для вводу електронної адреси користувача.
- **Password** – поле для вводу паролю користувача.
- **Log in** – кнопка для підтвердження введених даних і входу у додаток.
- **Cancel** – кнопка для скасування авторизації та повернення до головного вікна.

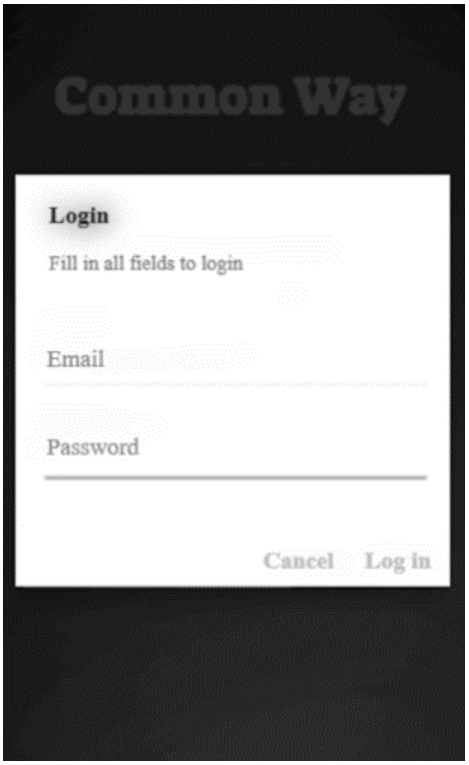

Рисунок 3.2 – Форма авторизації

Форма реєстрації (Рис 3.3) - ця форма створена для реєстрації нового користувача, і вимагає заповнення всіх зазначених полів для створення облікового запису. Форма складається з наступних елементів:

- **Email** – текстове поле для введення електронної пошти користувача.
- **Password** – текстове поле для введення пароля користувача.
- **Name** – текстове поле для введення імені користувача.
- **Age** – текстове поле для введення віку користувача.
- **Phone** – текстове поле для введення номера телефону.

- **Cancel** – кнопка для скасування реєстрації, закриття форми та повернення на головний екран мобільного застосунку.
- **Ok** – кнопка для підтвердження введених даних та завершення реєстрації.

Після завершення валідації всіх введених даних система створить обліковий запис для користувача та надасть доступ до основних функцій додатку.

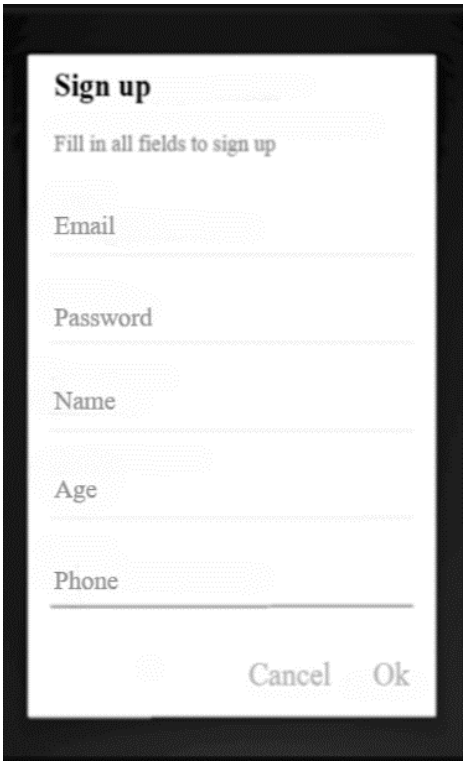

Рисунок 3.3 – Форма реєстрації

Вікно пошуку поїздки (Рис.3.4) – призначене для легкого та швидкого пошуку подорожей, забезпечуючи інтуїтивно зрозумілу навігацію і чіткі вказівки для користувача. У верхній частині вікна знаходиться верхня навігаційна панель на якій розташовані іконки «Profile» та «Settings». Вони дозволяють користувачу переходити до налаштувань профілю та загальних налаштувань додатку.

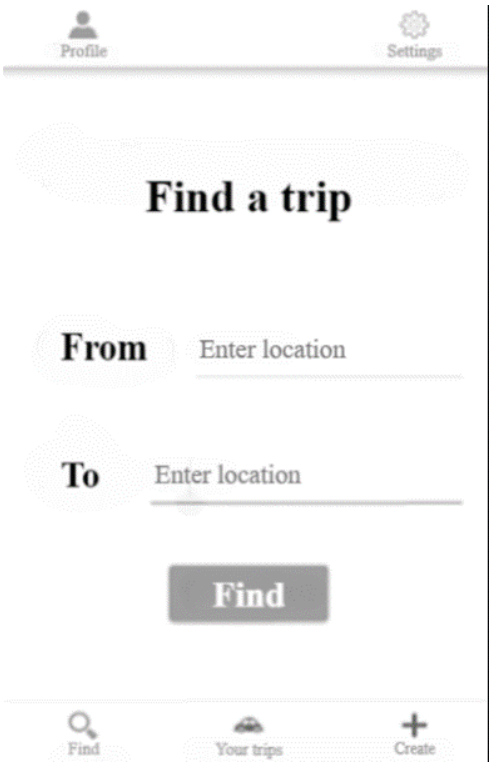

Рисунок 3.4 - Вікно пошуку поїздки

Саме ж вікно складається з:

- Заголовку «Find a trip» – розташований по центру, великим шрифтом, що ясно вказує на основну функцію вікна — пошук поїздки
- Текстового поля «From», де користувач може ввести початкову точку своєї поїздки.
- Текстового поля «To», де користувач може ввести кінцеву точку своєї поїздки.
- Велика кнопка «Find» – розташована під полями вводу. Натискання на цю кнопку запускає процес пошуку поїздки між введеними пунктами.

У нижній частині екрана розташована нижня панель навігації яка складається з трьох кнопок:

- **Find** – переходить до форми пошуку (поточний екран).
- **Your trips** – показує список поїздок користувача.
- **Create** – дозволяє створити нову поїздку.

Вікно заброньованих поїздок користувача (Рис.3.5) призначене для перегляду запланованих поїздок користувача, забезпечуючи легкий доступ до інформації про дату, час, місце та ціну поїздки.

Рисунок 3.5 – Вікно поїздок користувача

Вікно поїздок складається з наступних елементів:

- **Заголовок** – заголовок «Today» розташований по центру, великим шрифтом, що вказує на те, що нижче знаходиться список поїздок, запланованих на сьогодні (або іншу дату).
- **Список поїздок**

  У кожній поїздці зазначені:

  - **Час відправлення** – 23:00.
  - **Місце відправлення** – Kyiv.
  - **Час прибуття** – 05:50.
  - **Місце прибуття** – Odesa.
  - **Ціна** – 300 ₴.

У разі якщо користувач не має заброньованих поїздок, йому буде показано вікно з повідомленням про те, що тут будуть відображатися його заброньовані поїздки (Рис.3.6)

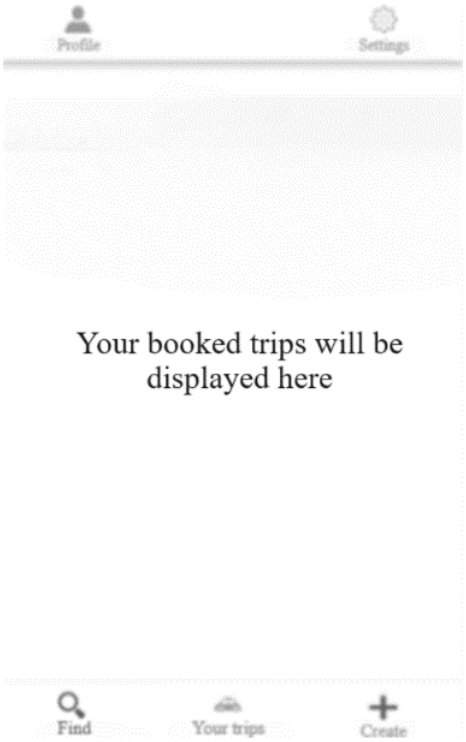

Рисунок 3.6 – Пусте вікно поїздок користувача

Вікно створення поїздок (Рис 3.7) – призначене для легкого та швидкого пошуку подорожей, забезпечуючи інтуїтивно зрозумілу навігацію і чіткі вказівки для користувача.

Вікно створення поїздок складається з наступних елементів:

- Заголовок «Create a trip» – розташований по центру, великим шрифтом, що ясно вказує на основну функцію вікна — створення поїздки
- Текстове поле «From», де користувач може ввести початкову точку своєї поїздки.
- Текстове поле «To», де користувач може ввести кінцеву точку своєї поїздки.
- Велика кнопка «Create» – розташована під полями вводу. Натискання на цю кнопку запускає процес створення поїздки між введеними пунктами.

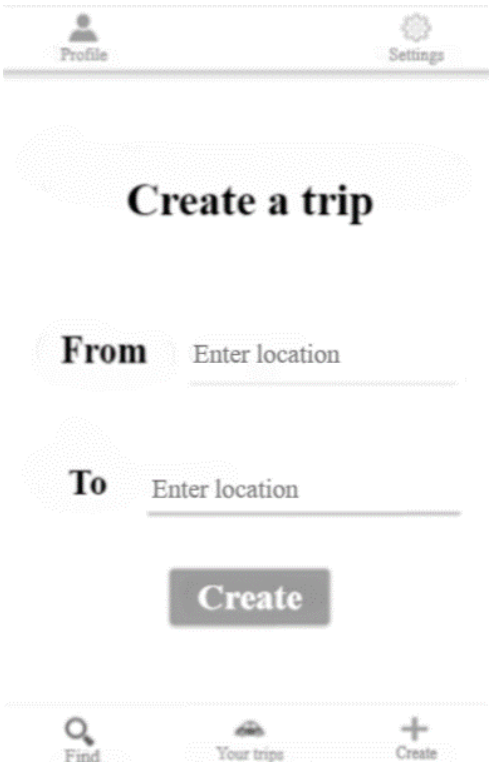

Рисунок 3.7 – Вікно створення поїздок

Вікно профіля користувача (Рис. 3.8) дозволяє користувачу легко переглядати свої особисті дані, а також додавати новий автомобіль до свого профілю.

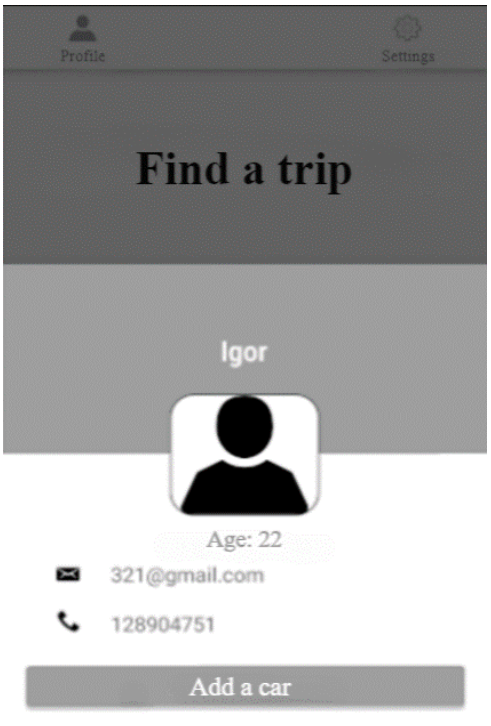

Рисунок 3.8 – Вікно профіля

Вікно складається з інформації про користувача, а саме:

- Ім’я
- Іконка профілю
- Вік
- Email
- Телефон

і кнопки «Add a car» розташованої під інформацією користувача. Натискання на цю кнопку відкриває форму для додавання нового автомобіля до профілю користувача.

Екран додавання автомобіля (Рис 3.9) – цей екран є важливою частиною процесу налаштування профілю для водіїв, які хочуть запропонувати спільні поїздки.

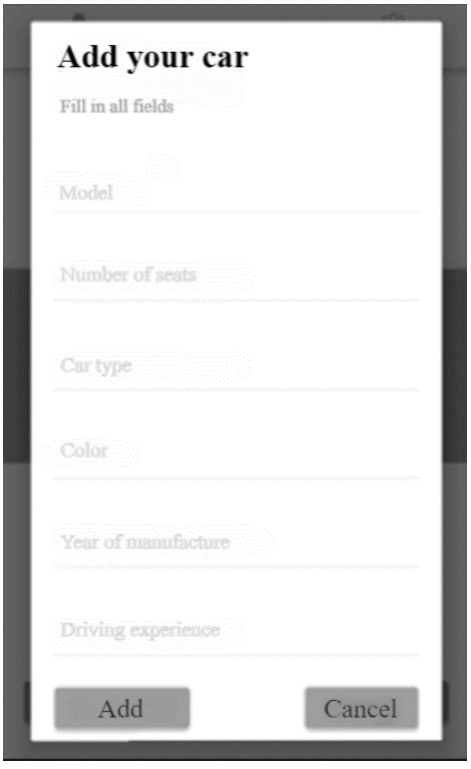

Рисунок 3.9 – Вікно додавання автомобіля

Екран додавання автомобіля представляє собою форму для введення даних, яка складається з наступних полів:

- Model – поле для введення моделі автомобіля.
- Number of seats – поле для введення кількості місць в автомобілі.
- Car type – поле для введення типу автомобіля (наприклад, седан, хетчбек тощо).
- Color – поле для введення кольору автомобіля.
- Year of manufacture – поле для введення року випуску автомобіля.
- Driving experience – поле для введення водійського стажу користувача.

та двох кнопок:

- Add – кнопка для підтвердження введеної інформації та додавання автомобіля до профілю.
- Cancel – кнопка для скасування дії та повернення до попереднього екрану без збереження змін.

Цей екран забезпечує користувачам простий та інтуїтивно зрозумілий спосіб додавання інформації про свій автомобіль, що є важливою частиною процесу налаштування профілю в додатку

Вікно налаштувань (Рис.3.10) – це вікно дозволяє користувачу виходити з облікового запису за допомогою зручної кнопки «Log out».

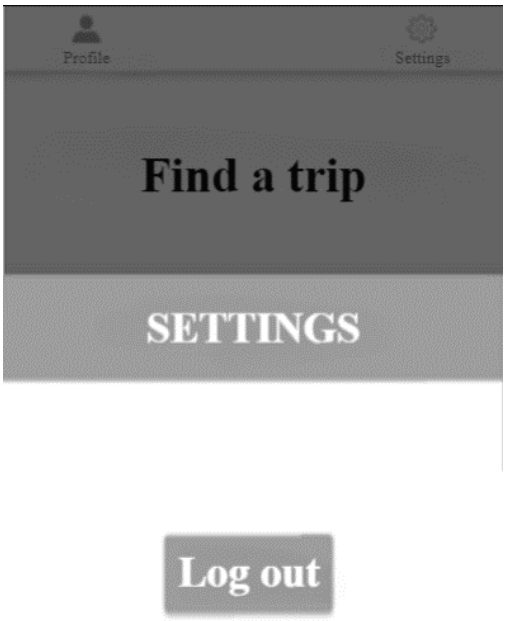

Рисунок 3.10 – Вікно налаштувань

Детальніше результат роботи представлений в додатку Б.

Для розробки додатку було використано наступні інструменти:

- Java – об’єктно-орієнтована мова програмування.
- SQLite – вільна система для керування реляційними БД.
- AndroidStudio – це інтегроване середовище розробки (IDE) для роботи з платформою Android.
- FireBase – це хмарна база даних, дані в якій зберігаються миттєво.
- Room – це високорівневий інтерфейс для низькорівневих прив'язок SQLite, вбудованих в Android.

Розділ 4. ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

4.1 Вимоги до програмного забезпечення та основні підходи до його проектування з погляду користувача

Розробка мобільного додатку для пошуку попутників повинна базуватися на ключових вимогах та підходах, що забезпечують зручність, доступність, продуктивність, безпеку і мультиплатформенність. Інтерфейс додатку має бути інтуїтивно зрозумілим, що дозволить користувачам легко знаходити необхідну інформацію та виконувати потрібні дії. Логічна структура навігації повинна забезпечувати швидкий доступ до всіх функцій. Додаток має відповідати принципам універсального дизайну, щоб бути доступним для всіх користувачів, включаючи людей з обмеженими можливостями, і працювати на різних платформах, зокрема iOS та Android, з забезпеченням консистентного користувацького досвіду. Він повинен швидко завантажуватися та реагувати на дії користувача, ефективно використовуючи ресурси пристрою. Захист персональних даних користувачів є пріоритетом, тому необхідно застосовувати сучасні методи шифрування та аутентифікації. Таким чином, основні підходи до проектування включають інтуїтивність, зручність, доступність, продуктивність і безпеку, що сприяє створенню високоякісного та ефективного програмного забезпечення для користувачів.

4.2 Ергономічні цілі і показники якості програмного продукту

Ергономічні цілі при розробці мобільного додатку спрямовані на зниження когнітивного навантаження та забезпечення комфортності використання. Основні показники якості програмного продукту включають інтуїтивність інтерфейсу, який дозволяє користувачам легко орієнтуватися і виконувати завдання з мінімальними зусиллями. Комфортність використання досягається через відповідність розміру елементів інтерфейсу фізичним можливостям користувачів, зручність для різних

сценаріїв використання (наприклад, одноручне використання або робота в різних орієнтаціях екрана), а також надання зрозумілого зворотного зв'язку на дії користувачів. Додаток повинен бути адаптивним до різних розмірів екранів і орієнтацій пристроїв, забезпечуючи ефективність виконання основних завдань без зайвих зусиль з боку користувача. Успішна реалізація цих цілей і показників якості сприяє створенню зручного, ефективного та задовольняючого користувачів програмного продукту.

4.3 Основні характеристики, що враховуються при розробці інтерфейсу користувача

При розробці інтерфейсу користувача важливо враховувати основні характеристики, такі як консистентність дизайну, інтуїтивність, естетичний вигляд, ефективність та доступність для всіх користувачів. Вимоги до зручності та комфортності полягають у створенні інтерфейсу, який легко сприймається і використовується користувачами, з мінімальним когнітивним навантаженням та позитивним емоційним враженням. Прототип інтерфейсу розробляється з урахуванням таких проблем, як некоректне розташування елементів чи недосконаліть функціоналу, і вимагає тестування з метою виявлення і виправлення недоліків. Принципи реалізації призначеного для користувача інтерфейсу полягають у використанні зрозумілої мови, зручних елементів управління та наданні зворотного зв'язку. Вимоги до процесів інтерфейсу та проектування компонентів включають розробку детальних макетів, тестування їх на користувачах та вдосконалення з урахуванням отриманих відгуків. У свою чергу, проектування і реалізація компонентів інтерфейсу передбачає створення окремих елементів, таких як кнопки, форми, списки, з використанням принципів консистентності та ефективності, та їх інтеграцію у загальний дизайн з урахуванням потреб користувачів.

4.4 Техніко-економічне обґрунтування розробки

Для успішної реалізації проекту з розробки мобільного додатку для пошуку попутників необхідне чітке бачення розвитку потенціалу обраної ідеї та детальне планування кожного етапу. Даний підрозділ містить докладний опис проекту, починаючи від резюме і закінчуючи оцінкою ризиків та страхуванням. У ньому визначені ключові аспекти, які допоможуть оцінити ринковий потенціал додатку, зрозуміти його конкурентні переваги, розробити ефективну маркетингову стратегію та забезпечити фінансову стабільність проекту.

4.4.1 Резюме проекту

Назва: CommonWay

Місце розташування: Мобільний додаток буде доступний для завантаження в основних мобільних магазинах додатків (Google Play, App Store) та орієнтований на користувачів у великих містах по всьому світу.

Мета: Основною метою проекту є зниження транспортного навантаження на міські дороги, покращення екологічної ситуації в містах та підвищення економічної ефективності поїздок для користувачів. Це буде досягнуто шляхом впровадження зручного та надійного сервісу для організації спільних поїздок на автомобілях (карпулінгу).

Суть проекту: Проект передбачає розробку мобільного додатку CommonWay, який автоматизує процес організації спільних поїздок, надаючи користувачам можливість легко знаходити попутників, координувати маршрути та домовлятися про деталі поїздки. Додаток пропонує широкий спектр функціональних можливостей для забезпечення зручності та безпеки користувачів:

1. Користувачі можуть створювати нові поїздки або шукати вже створені поїздки інших користувачів на основі їхнього маршруту та часу поїздки.
2. Користувачі можуть залишати відгуки та рейтинги про водіїв і пасажирів, що підвищує довіру та безпеку спільних поїздок.

3. Оптимізація маршрутів за допомогою інтеграції з популярними картографічними сервісами (Google Maps, Apple Maps), що дозволяє вибирати найбільш зручні та швидкі маршрути.
4. Вбудована система обміну повідомленнями для координації деталей поїздки між водіями та пасажирами.
5. Впровадження сучасних технологій захисту даних для забезпечення конфіденційності та безпеки особистої інформації користувачів.

Переваги CommonWay:

- Зменшення кількості транспортних засобів на дорогах, що призводить до зменшення заторів та покращення трафіку.
- Зниження викидів CO2 та інших шкідливих речовин в атмосферу завдяки зменшенню кількості автомобілів.
- Зниження витрат на паливо, ремонт та обслуговування автомобілів завдяки спільному використанню транспортних засобів.
- Можливість знайомства та спілкування з новими людьми під час поїздок.

Цільова аудиторія - міські жителі віком від 18 до 45 років, які регулярно користуються автомобілями для пересування містом, включаючи студентів, офісних працівників та інших активних громадян, які зацікавлені в економічно вигідних та екологічно чистих способах пересування.

Очікувані результати:

- Зменшення заторів на дорогах міста.
- Зниження рівня забруднення повітря.
- Зменшення витрат на транспорт для користувачів.
- Покращення соціальної взаємодії серед мешканців міста.

Резюмуючи все вище сказане, цей проект має великий потенціал для створення значного позитивного впливу на міські транспортні системи, екологію та економіку, а також для підвищення якості життя міських жителів.

5.4.2 Опис проектованого продукту

Проектований продукт: Мобільний додаток Common Way

Common Way – це інноваційний мобільний додаток для організації спільних поїздок на автомобілях (карпулінг). Його основною метою є надання користувачам зручного інструменту для пошуку попутників, координації маршрутів та спільного використання транспортних засобів. Додаток призначений для міських жителів, які хочуть зменшити витрати на поїздки, знизити навантаження на дороги та покращити екологічну ситуацію в місті.

Основні функції та можливості Common Way:

- Створення та пошук поїздок:

Користувачі можуть створювати нові поїздки, вказуючи пункт відправлення, пункт призначення, дату і час поїздки, кількість вільних місць, а також додаткову інформацію (наприклад, чи дозволено брати з собою домашніх тварин або багаж). Користувачі можуть шукати поїздки, які відповідають їхнім потребам, використовуючи фільтри за маршрутом, часом поїздки, рейтингом водія та іншими параметрами.

- Система рейтингів та відгуків:

Після завершення поїздки користувачі можуть оцінити водія та пасажирів за п'ятибальною шкалою, що допомагає підвищити рівень довіри та безпеки. Користувачі можуть залишати текстові відгуки, що дають змогу ділитися враженнями про поїздку та поведінку інших учасників.

- Інтеграція з картографічними сервісами:

Додаток автоматично інтегрується з популярними картографічними сервісами (Google Maps, Apple Maps), що дозволяє водіям планувати оптимальні маршрути з урахуванням дорожньої ситуації та об'їздів. Користувачі можуть в режимі реального часу відстежувати маршрут поїздки на карті та отримувати інформацію про затори.

- Обмін повідомленнями:

Вбудована система обміну повідомленнями дозволяє водіям та пасажирам легко координувати деталі поїздки, обговорювати місце зустрічі, уточнювати час

відправлення тощо. Додаток надсилає сповіщення про нові повідомлення, заплановані поїздки та інші важливі події.

- Безпека та захист даних:

CommonWay використовує сучасні методи шифрування даних, щоб забезпечити конфіденційність особистої інформації користувачів. Для підвищення безпеки додаток може включати функції верифікації користувачів, наприклад, за допомогою електронної пошти, номеру телефону або соціальних мереж.

- Додаткові функції:

Інтеграція з популярними платіжними системами для безготівкових розрахунків за спільні поїздки. Зручний календар для планування регулярних поїздок, наприклад, поїздок на роботу чи навчання. Цілодобова служба підтримки користувачів, яка допомагає вирішувати будь-які питання та проблеми.

Користувацький інтерфейс

Головний екран – цей екран є початковою точкою взаємодії користувачів з додатком і має інтуїтивно зрозумілий дизайн, що сприяє зручному входу або реєстрації.

Екран пошуку поїздки – пошукова форма для швидкого знаходження необхідних маршрутів.

Екран поїздок користувача – відображає список заброньованих поїздок користувачем

Екран створення поїздки – інтуїтивно зрозуміла форма для введення деталей нової поїздки.

Екран профілю користувача – включає інформацію про користувача, його рейтинг, відгуки та історію поїздок.

Екран додавання автомобіля користувача – цей екран є важливою частиною процесу налаштування профілю для водіїв, які хочуть запропонувати спільні поїздки.

Чат – зручний інтерфейс для обміну повідомленнями з іншими користувачами.

Common Way пропонує комплексне рішення для організації спільних поїздок, яке не лише спрощує процес пошуку попутників, але й підвищує економічну та екологічну ефективність поїздок, сприяючи створенню більш комфортних умов життя в містах.

4.4.2 Оцінка ринку збуту

Оцінка ринку збуту є важливим етапом при розробці та впровадженні мобільного додатку для карпулінгу, такого як "Common Way". Це дозволяє зрозуміти потенційний обсяг ринку, конкурентне середовище, цільову аудиторію та можливі виклики. Нижче представлена детальна оцінка ринку збуту для "Common Way".

Аналіз цільової аудиторії

Аналіз цільової аудиторії приведено в таблиці 4.1.

Таблиця 4.1 – Аналіз цільової аудиторії

Вік	18-45 років. Це активні користувачі, які регулярно користуються смартфонами та мають потребу в щоденних поїздках.
Місце проживання	Великі міста з високою щільністю населення, де проблема заторів і транспортного навантаження є особливо актуальною.
Професійний статус	Студенти, офісні працівники, фрілансери та інші активні користувачі, які мають регулярні маршрути та шукають економічно вигідні способи пересування.
Мотиви	Економія коштів на поїздках, зменшення часу в заторах, екологічна свідомість, соціальна взаємодія

Географічний аналіз ринку

Основні регіони для запуску:

- Європа

Міста як Київ, Львів, Лондон, Париж, Берлін, Амстердам, де транспортні проблеми є надзвичайно актуальними.

- Північна Америка

Міста як Нью-Йорк, Лос-Анджелес, Сан-Франциско, Чикаго з високим рівнем автомобілізації та заторами.

- Азія

Мегаполіси як Токіо, Сеул, Шанхай, де населення активно користується технологіями для оптимізації пересування.

Розширення ринку

- Середні та малі міста

Після успішного запуску в мегаполісах можна розглядати вихід на ринки менших міст, де транспортні проблеми також є актуальними.

- Інтернаціоналізація

Підтримка кількох мов для залучення користувачів з різних країн.

Аналіз попиту

У сучасному світі все більше людей стають екологічно свідомими. Це проявляється у збільшенні попиту на екологічно чисті технології та способи пересування. Люди прагнуть зменшити свій вуглецевий слід, сприяючи зниженню рівня забруднення повітря. Карпулінг якраз відповідає цим потребам, оскільки дозволяє зменшити кількість автомобілів на дорогах, що, в свою чергу, зменшує викиди парникових газів. Вибираючи карпулінг, користувачі роблять свій внесок у захист навколишнього середовища.

Іншим важливим фактором який впливає на актуальність ринку – це економічний фактор. Вартість володіння автомобілем постійно зростає. Це включає не лише ціну пального, яка є дуже мінливою і часто зростає, але й витрати на обслуговування, страхування та паркування. У великих містах, де проблема паркування особливо актуальна, вартість паркувальних місць може бути дуже високою. Карпулінг дозволяє розділити ці витрати між кількома пасажирами, що робить поїздки більш економічно вигідними для кожного учасника.

Сучасні технології значно спрощують організацію повсякденних активностей. Мобільні додатки стали невід'ємною частиною життя багатьох

людей, надаючи можливість швидко та зручно виконувати різноманітні завдання – від замовлення їжі до організації подорожей. Мобільні додатки для карпулінгу, такі як "Common Way", пропонують інтуїтивно зрозумілий інтерфейс, швидкий пошук попутників, оптимізацію маршрутів та системи рейтингу для підвищення довіри між користувачами. Це робить карпулінг не лише економічно вигідним, але й зручним способом пересування, що підходить для сучасного темпу життя.

Загалом, аналіз попиту показує, що карпулінг має великий потенціал для зростання завдяки зростаючому попиту на екологічні рішення, підвищенню вартості володіння автомобілем та зручності використання сучасних технологій.

Висновки: оцінка ринку збуту показує, що "Common Way" має великий потенціал для успішного запуску та зростання. Зростаючий попит на економічно вигідні та екологічні способи пересування у великих містах, разом із зручністю використання мобільних додатків, створює сприятливі умови для розвитку сервісу. Правильний маркетинговий підхід, адаптація до локальних умов та постійне вдосконалення функціоналу допоможуть "Common Way" успішно конкурувати на ринку та задовольнити потреби своєї цільової аудиторії.

4.4.3 Стратегія маркетингу

Стратегія маркетингу є ключовим аспектом успішного запуску та розвитку будь-якого програмного продукту, включаючи мобільний додаток для карпулінгу "Common Way". В умовах зростаючої конкуренції та швидко змінюваних ринкових тенденцій, ефективна маркетингова стратегія дозволяє не лише залучити нових користувачів, але й утримувати їх, забезпечуючи сталий розвиток сервісу.

Цей підрозділ присвячений детальному аналізу методів просування додатку на ринку. Розглянемо основні аспекти, які сприятимуть збільшенню впізнаваності бренду, залученню цільової аудиторії та підвищенню лояльності користувачів.

Методи просування

- **Соціальні мережі (Facebook, Instagram, Twitter)**

Для просування в соцільаних мережах використовують три основні стратегії: цілеспрямована реклама, контент-маркетинг та інтерактивні кампанії.

Цілеспрямована реклама включає використання таргетованої реклами для охоплення цільової аудиторії. Така реклама базується на інтересах, демографічних даних та поведінці користувачів у соціальних мережах, що дозволяє ефективно залучати потенційних користувачів додатку.

Контент-маркетинг передбачає створення цікавого та корисного контенту про переваги карпулінгу, екологічні та економічні аспекти. Це можуть бути статті, інфографіка, відео, історії успіху та поради щодо безпечного спільного використання автомобілів. Такий контент допоможе підвищити обізнаність користувачів про додаток і його переваги.

Інтерактивні кампанії включають проведення конкурсів, опитувань та акцій, які сприяють залученню користувачів та підвищенню їхньої активності. Це дозволить створити спільноту навколо додатку та залучити більше користувачів через активну взаємодію та позитивні відгуки.

- **Партнерства з університетами та великими компаніями**

Співпраця з навчальними закладами для просування додатку серед студентів та викладачів. Це може включати проведення презентацій, семінарів, надання знижок та акцій. Така співпраця сприятиме підвищенню обізнаності про додаток серед молоді, яка активно користується мобільними додатками та є відкритою до нових технологій і рішень для пересування.

Партнерство з великими компаніями для впровадження карпулінгу серед їхніх співробітників. Це може бути реалізовано через корпоративні програми, заохочення до використання додатку та організацію спільних поїздок на роботу. Такий підхід дозволить зменшити транспортне навантаження на міські дороги, знизити витрати на паркування та паливо, а також сприяти підвищенню екологічної відповідальності компаній.

- **Рекламні кампанії в громадському транспорт**

Плакати та банери у автобусах, трамваях, метро та на зупинках громадського транспорту. Це дозволить охопити широку аудиторію, яка активно користується міським транспортом. Завдяки яскравим та інформативним плакатам і банерам, пасажири дізнаватимуться про додаток та його переваги, що сприятиме збільшенню кількості завантажень.

Аудіо та відеореклама у громадському транспорті, що інформуватиме пасажирів про переваги карпулінгу та закликатимуть завантажити додаток. Аудіо- та відеореклама дозволить ефективно привернути увагу пасажирів, особливо в часи пік, коли громадський транспорт є найбільш завантаженим. Це допоможе досягти широкого охоплення та підвищити обізнаність про додаток серед населення.

- **Підтримка та участь у місцевих екологічних ініціативах**

Участь у місцевих екологічних проектах, фестивалях та акціях підвищить обізнаність про додаток серед екологічно свідомих громадян.

Спонсорування екологічних подій та заходів привертатиме увагу до бренду та його екологічних цінностей.

Організація спільних ініціатив з місцевими екологічними організаціями для просування ідей сталого розвитку та використання карпулінгу.

Висновки: стратегія маркетингу для "Common Way" орієнтована на залучення активних користувачів мобільних додатків у міських районах. Використання соціальних мереж, партнерство з університетами та великими компаніями, рекламні кампанії в громадському транспорті та участь у місцевих екологічних ініціативах допоможуть ефективно просувати додаток та збільшувати кількість користувачів. Ці методи сприятимуть підвищенню обізнаності про карпулінг, його економічні та екологічні переваги, а також забезпечать довгостроковий успіх проекту.

4.4.4 План виробництва додатку

Етап 1: Аналіз вимог та проектування архітектури додатку

Початковий етап включає детальний аналіз потреб користувачів і визначення основних функцій додатку. Після збору вимог проводиться проектування архітектури, вибір технологій і визначення структури бази даних. Цей етап закладає основи для подальшої розробки, забезпечуючи ясність і консистентність у всіх наступних кроках.

Етап 2: Розробка основних функціональних модулів

На даному етапі реалізуються основні функції додатку відповідно до визначених вимог. Це включає розробку інтерфейсу користувача, бізнес-логіки та інтеграцію з іншими системами. Кожен модуль тестується для впевненості у його функціональності та сумісності з іншими частинами системи.

Етап 3: Інтеграція з картографічними сервісами

Оскільки додаток має включати картографічні можливості, цей етап фокусується на інтеграції з відповідними картографічними сервісами. Це може включати вибір та інтеграцію зі сторонніми API для отримання геоданих, реалізацію функцій відображення мап і взаємодію з ними.

Етап 4: Тестування та відладка додатку

На цьому етапі проводяться різні види тестування: одиниць, інтеграційне тестування та системне тестування. Основна мета - виявлення та виправлення будь-яких помилок у додатку, щоб забезпечити його надійність і стабільність перед релізом.

Етап 5: Запуск бета-версії та збирання відгуків користувачів

Після успішного тестування випускається бета-версія додатку для обмеженої аудиторії користувачів. Збираються відгуки і фідбек від користувачів щодо функціональності, інтерфейсу та загального враження від використання додатку.

Етап 6: Випуск стабільної версії та постійне оновлення

Після успішного завершення всіх попередніх етапів і враховуючи отриманий фідбек випускається стабільна версія додатку. Після релізу продовжується

підтримка і регулярні оновлення, включаючи нові функції, виправлення помилок і підтримку сумісності з оновленнями платформ.

Цей план виробництва дозволяє систематично та ефективно розвивати додаток, забезпечуючи його високу якість, відповідність вимогам користувачів та успішний запуск на ринку.

4.4.5 Організаційний та юридичний плани

Організаційний план

Команда проекту складається з кількох ключових ролей, які спільно працюють для досягнення цілей проекту:

- Керівник проекту

Основна роль керівника проекту полягає в організації робочих процесів, координації дій команди та взаємодії з замовниками. Він відповідає за вчасне виконання завдань, контролює бюджет та вирішує конфліктні ситуації.

- Розробники програмного забезпечення

Ця група включає в себе розробників, які займаються написанням програмного коду додатку. Вони враховують вимоги користувачів і спільно працюють для створення функціональних модулів додатку.

- Дизайнери інтерфейсу користувача

Дизайнери відповідають за створення зручного та естетичного інтерфейсу користувача. Вони розробляють макети, проводять тестування UX/UI та забезпечують відповідність дизайну бренду компанії.

- Маркетологи

Ці спеціалісти розробляють стратегії маркетингу для просування додатку на ринку. Вони визначають цільову аудиторію, розробляють промоційні матеріали, ведуть комунікацію з потенційними користувачами і здійснюють аналіз конкурентів.

- Спеціалісти з підтримки користувачів

Ця група забезпечує підтримку користувачів, відповідає на їх запитання через платформи підтримки, електронну пошту або чат. Вони вирішують технічні проблеми та надають інструкції щодо використання додатку.

Юридичний план

Юридичний план включає в себе ряд важливих аспектів для забезпечення правової відповідності та захисту даних користувачів:

- Реєстрація компанії та отримання необхідних ліцензій

Це перший крок до офіційного створення компанії. Він включає реєстрацію юридичної особи та отримання всіх необхідних дозвільних документів для роботи у відповідній сфері.

- Забезпечення відповідності додатку нормам захисту даних користувачів

Важливий аспект, особливо в контексті GDPR (Загального регламенту про захист персональних даних) та інших регуляційних вимог щодо захисту приватності користувачів. Це включає розробку політики конфіденційності та впровадження технічних заходів для захисту даних.

- Розробка користувацької угоди та політики конфіденційності

Ці документи встановлюють правила використання додатку користувачами та обов'язки компанії щодо захисту персональних даних. Вони є важливими для юридичної захищеності компанії і підвищення довіри користувачів.

Цей план забезпечує не лише ефективну розробку та впровадження додатку, але й враховує всі аспекти управління командою, юридичній відповідності та захисту даних користувачів. Він спрямований на досягнення успіху проекту і забезпечення позитивного досвіду для кінцевих користувачів.

4.4.6 Стратегія фінансування

Стратегія фінансування є ключовим компонентом для забезпечення успішної реалізації проекту розробки додатку. Є декілька стратегій фінансування:

- Власні кошти

Використання власних коштів засновників або інвесторів є одним з найпростіших і швидких способів забезпечення фінансування. Це можуть бути особисті заощадження засновників, персональні кредити або інші ресурси. Використання власних коштів може дати більший контроль над проектом і уникнути залучення зовнішніх інвесторів, однак це також несе ризик втрати особистих фінансових ресурсів у разі невдачі проекту.

- Приватне фінансування

Залучення приватних інвесторів або венчурних капіталістів є популярним варіантом для стартапів та інноваційних проектів. Це мож бути індивідуальні інвестори, ангельські інвестори або венчурні фонди, які зацікавлені в новаторських технологіях. Приватні інвестори часто можуть також вносити свій досвід та контакти, що додає додаткову цінність поза фінансовою підтримкою.

- Гранти та субсидії

Отримання грантів або субсидій від державних або міжнародних організацій може бути важливим джерелом фінансування для інноваційних проектів у галузі програмного забезпечення та технологій. Ці ресурси часто надаються з метою підтримки інновацій та технологічного розвитку, і можуть мати специфічні вимоги щодо презентації та використання фінансування.

- Краудфандинг

Використання платформ краудфандингу, таких як Kickstarter, Indiegogo або GoFundMe, може допомогти залучити фінансування від широкої громадськості. Це може бути особливо корисним для створення зацікавленості та підтримки серед потенційних користувачів і прихильників продукту.

- Банківське фінансування

Отримання кредитів від банків або інших фінансових установ може бути варіантом для покриття витрат на розробку та запуск проекту. Цей варіант може вимагати демонстрацію фінансової стійкості та спроможності повернення кредиту у майбутньому, відповідно до доходів від продажу продукту або від підписок.

Висновки: стратегія фінансування є важливим інструментом для забезпечення стабільного функціонування проекту розробки додатку. Вона дозволяє залучити необхідні ресурси, уникнути фінансових проблем і забезпечити успішний випуск та масштабування продукту на ринку.

4.4.7 Оцінка ризику та страхування

Оцінка ризиків дозволяє ідентифікувати потенційні загрози та проблеми, які можуть виникнути протягом розробки та експлуатації додатку. Це можуть бути технічні проблеми, зміни вимог користувачів, або навіть зміни у законодавстві.

На основі оцінки ризиків розробляються стратегії управління ризиками, які включають у себе уникнення, зменшення або прийняття ризиків. Це дозволяє команді проекту заздалегідь приготуватися до можливих негативних сценаріїв і реагувати на них ефективно.

Через оцінку ризиків можна зменшити невизначеність і несподіваність у процесі розробки. Це сприяє стабільності та прогностованій успішності проекту.

Види ризиків:

- **Технічні ризики** такі як:

Проблеми з сумісністю - різноманітні мобільні пристрої та версії операційних систем можуть вплинути на сумісність додатку.

Безпека даних – ризики пов'язані з витоками персональної інформації користувачів, що можуть виникнути внаслідок кібератак або несанкціонованого доступу до бази даних.

- **Бізнес-ризики**

Конкуренція, змагання з іншими аналогічними додатками може ускладнити привертання користувачів.

Монетизація, непрогнозовані складнощі у виборі оптимальної моделі монетизації (наприклад, підписка, реклама, платіжні комісії).

- **Організаційні ризики**

Кадрові перешкоди – проблеми зі складом команди розробників, що можуть вплинути на терміни виконання проекту.

Маркетинг і просування – ризики, пов'язані з недооцінкою маркетингових витрат та стратегій для залучення користувачів.

Страхування

Високотехнологічні проекти, такі як розробка мобільних додатків, можуть потрапити під вимоги за порушення авторських прав, порушення конфіденційності або недоліки в продукті. Страхування від відповідальності може покривати витрати на юридичні послуги, відшкодування шкоди та інші витрати, пов'язані з цими справами.

Мобільні додатки збирають та обробляють значну кількість персональних даних користувачів. Витоки даних, віруси та інші кіберзагрози можуть викликати серйозні фінансові наслідки. Страхування може надати фінансову підтримку для відшкодування збитків, відновлення даних та відновлення репутації після подій такого роду.

Страхування від відповідальності та кіберзагроз може значно знизити фінансові ризики, пов'язані з можливими судовими справами або витоками даних. Це особливо актуально у високотехнологічних проектах, де безпека даних є пріоритетом.

Охорона інфраструктури, такої як серверне обладнання, хостингові послуги та інші важливі активи, є критично важливою для неперебірної роботи мобільного додатку. Страхування забезпечує захист від ризиків фізичних пошкоджень, природних катастроф, крадіжок та інших небезпечних подій.

Наявність адекватного страхування підвищує довіру від користувачів та інвесторів, показуючи, що команда проекту готова до вирішення можливих проблем. Це демонструє відповідальність і готовність виконувати вимоги безпеки та захисту даних.

Висновки: ефективне управління ризиками та належне страхування можуть значно зменшити вплив негативних подій на проект і забезпечити стабільність

його розвитку. Враховуючи високу динаміку ринку програмного забезпечення та його зв'язок з технологіями, ці аспекти стають критичними для успіху проекту.

4.4.8 Розрахунок витрат на створення програми

Під час виконання дипломного проекту необхідно було розробити додаток пошуку попутників для ОС Android. Розрахунок економічної ефективності дозволить обґрунтувати доцільність розробки, розрахувати собівартість додатку та оцінити час його окупності.

Розрахуємо вартість розробки програмного забезпечення.

Середня заробітна плата Android розробників (Junior) складає 32000 грн/міс, а вартість сучасного ПК складає 30000 грн. Вартість кіловат-години електроенергії складає 4,32 грн.

Витрати на допоміжні матеріали приведені в таблиці 4.1.

Таблиця 4.1 – Розрахунок витрат на «Матеріали і комплектуючі вироби»

Пункти витрат	Сума, грн./міс
1	2
Інтернет	250,00
Всього матеріали	250,00

Вартість програми розраховується по формулі:

$$С_{пр} = (З_{зп} + З_{сз} + З_{зв} + З_{е} + З_{м} + А_{об})Т, \text{ де} \quad (4.1)$$

Т – тривалість розробки, міс;

$З_{зп}$ – основна і додаткова заробітна плата персоналу, грн.;

$З_{сз}$ – відрахування на соціальні заходи (30% від основної і додаткової заробітної плати, грн.);

$З_{зв}$ – загальногосподарські витрати (10% від основної заробітної плати, грн.);

$З_{е}$ – витрати на електроенергію;

$З_{м}$ – витрати на основні і допоміжні матеріали;

$А_{об}$ – амортизаційні відрахування на устаткування;

$З_{е}$ при споживанні потужності 240 Вт, тривалості роботи на місяць складає 22*8 = 176 годин, вартість кіловат-години електроенергії 4,32 грн, з цього слідує:

$$З_е = 176 * 4,32 * 0,24 = 178 \text{ грн.}$$

Амортизаційні відрахування на устаткування розраховуються за формулою:

$$А_{об} = \frac{\text{Вартість ПК}}{\text{Срок використання (міс)}} \quad (4.2)$$

$$А_{об} = \frac{30000}{24} = 1250 \text{ грн.}$$

Витрати на розробку програмного забезпечення приведені в таблиці 4.2.

Таблиця 4.2 – Витрати на розробку програмного забезпечення

	Найменування витрат	Одиниця вимірювання	Кількість
	1	2	3
	Термін розробки	міс	1,5
	Основна та допоміжна заробітна плата	грн.	32000,00
	Відрахування на соціальні заходи	грн.	9600,00
	Загально-господарчі витрати	грн.	3200,00
	Витрати на головні та допоміжні матеріали	грн.	250,00
	Витрати на електроенергію	грн.	178,00
	Амортизаційні відрахування	грн.	1250,00

$$С_{пр} = (32000 + 9600 + 3200 + 178 + 250 + 1250) * 1,5 = 69642 \text{ грн.}$$

4.4.9 Розрахунок економічної ефективності

Розрахунок економічної ефективності мобільного додатку включає оцінку витрат на розробку, підтримку та маркетинг порівняно з очікуваними доходами від продажів та інших джерел, що допомагає зробити обгрунтоване рішення щодо вкладення ресурсів та повернення інвестицій.

Розрахунок економічної ефективності також враховує прогнозовану кількість користувачів та їхню здатність генерувати доход через рекламу, спонсорські угоди або платні підписки. Крім того, аналізується динаміка ринку мобільних додатків, конкурентна обстановка та можливі ризики, що можуть вплинути на успіх додатку. Такий комплексний підхід допомагає зробити

обґрунтоване рішення щодо інвестицій та стратегії розвитку додатку з точки зору економічної доцільності.

Якщо враховувати, що в місяць як мінімум 1 клієнт буде купляти додаток, тоді щомісячний мінімальний дохід становитиме 500 грн. Термін окупності програмного забезпечення розраховується за наступною формулою:

$$T = \frac{З_{пр}}{I}, \text{ де} \quad (4.3)$$

$З_{сэ}$ – витрати на створення програми;

I – місячний дохід від продаж.

Тобто, максимальний термін окупності програмного забезпечення:

$$T = \frac{69642}{500} = 139 \text{ міс.}$$

Якщо кількість клієнтів в місяць збільшиться до 2, то термін окупності скоротиться вдвічі.

Висновки: ергономіка грає критичну роль у забезпеченні зручності і ефективності використання програмного продукту користувачами. Врахування принципів ергономіки дозволяє знизити час на навчання і покращити загальне враження від взаємодії з додатком. Щодо техніко-економічного обґрунтування, воно дозволяє ретельно оцінити всі витрати та прогнозувати доходи, що є ключовими аспектами перед початком розробки. Ці аспекти разом сприяють успішному впровадженню додатку на ринок і забезпечують його фінансову стійкість і конкурентоспроможність.

ВИСНОВКИ

В результаті виконання дипломного проекту було розроблено мобільний додаток для пошуку попутників на операційній системі Android. Цей додаток дозволяє користувачам шукати, створювати і бронювати поїздки на потрібну дату та час.

Під час роботи над дипломним проектом були пройдені такі етапи:

- Детальне дослідження предметної області, аналіз готових рішень та постановка задачі

Було проведено ретельний аналіз предметної області для визначення потреб та вимог користувачів, проаналізовано існуючі рішення на ринку та визначено їхні недоліки, що підкреслило актуальність створення нового додатку та сформульовано чітке завдання для розробки додатку, що відповідає вимогам користувачів.

- Проектування

Проектування додатку складалося з ескізного, технічного та робочого проектів. Для проектування була обрана об'єктно-орієнтована методологія, що дозволило створити гнучку та масштабовану архітектуру додатку. У ході ескізного проекту були розроблені прототипи інтерфейсу та основні функціональні вимоги. Технічний проект включав детальний опис функцій, структури бази даних, алгоритмів обробки даних. Робочий проект охоплював вибір інструментарію для розробки, обґрунтування цього вибору та розробку фізичної моделі додатку.

- Розробки і тестування функціональних модулів мобільного додатку

Було розроблено та протестовано основні функціональні модулі додатку, включаючи модулі пошуку, створення і бронювання поїздок. Створено інтерфейс користувача, який є зручним та інтуїтивно зрозумілим, забезпечуючи приємний користувацький досвід. Проведено тестування додатку для виявлення та виправлення помилок, а також для забезпечення стабільної та надійної роботи.

- Опрацювання техніко-економічної частини та підрахунок собівартості програмного забезпечення.

Проведено аналіз витрат на розробку програмного забезпечення, включаючи вартість інструментів, витрати на оплату праці розробників та інші ресурси. Підготовлено техніко-економічне обґрунтування проекту, що демонструє його економічну доцільність та потенційну прибутковість. Підготовлено документацію, що описує всі аспекти розробки, функціональності та використання додатку.

Під час виконання проекту самостійно були опрацьовані основні компоненти додатків під операційною системою Android, життєвий цикл роботи кожного типу збереження даних, створення екрану в редакторі, робота з файлами та зберігання даних. Це дозволило створити повнофункціональний додаток, що відповідає сучасним вимогам та забезпечує користувачам можливість зручно та ефективно шукати попутників для поїздок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розрахунок амортизації обладнання. URL: https://lico-ocenka.ru/kak-rasschitat-amortizatsiyu-oborudovaniya/#i-5 (дата звернення 19.05.2024).
2. Шаллоуей А. Шаблони проектування. Новий підхід до об'єктно аналізу і проектування / А. Шаллоуей, Д. Тротт. - М.: Вид. будинок «Вільямс», 2002. – 288 с.
3. Буч Г. UML. Класика CS / Г. Буч, А. Якобсон, Дж. Рамбо: пер. з англ. С. Орлова. - Вид. 2-е. - СПб.: Вид-во «Пітер», 2006. - 736 с.
4. Офіційний сайт Firebase. URL: https://firebase.google.com/?hl=ru (дата звернення: 11.03.2024).
5. Офіційний сайт Andoid Studio. URL: https://developer.android.com/studio (дата звернення: 22.02.2024).
6. Хорстманн К.С., Корнелл Г. Бібліотека професіонала. Java 2. Том 1. Харків: Видавницький дім «Вільямс», 2003. 848 с.
7. Android Developers. Офіційна документація Android. URL: https://developer.android.com/develop (дата звернення: 23.04.2024).
8. Nielsen J. Usability Engineering. – Academic Press, 1993. – 362 с.
9. ISO 9241-11:2018. Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. – International Organization for Standardization, 2018. – 40 с.
10. Wroblewski L. Mobile First. – A Book Apart, 2011. – 120 с.
11. Коваленко О.І. Основи мобільної розробки: Навчальний посібник. – Видавництво Київського університету, 2020. – 320 с
12. Турбан Е., Шардан Р., Вітер М. Інформаційні технології для менеджменту. – Видавництво Харківського університету, 2015. – 560 с.
13. Stack Overflow. Питання та відповіді розробників. URL: https://stackoverflow.com (дата звернення: 01.06.2024).
14. Johnson J. Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules. – Morgan Kaufmann, 2010. – 250 с.

15. Morville P., Rosenfeld L. Information Architecture for the World Wide Web: Designing Large-Scale Web Sites. O'Reilly Media, 2006. 528 с.
16. Norman D.A. The Design of Everyday Things: Revised and Expanded Edition. Basic Books, 2013. 384 с.
17. Benyon D. Designing User Experience: A Guide to HCI, UX and Interaction Design. Pearson, 2019. 640 с.
18. Nielsen J. Heuristic Evaluation // Usability Inspection Methods / Ed. by J. Nielsen & R.L. Mack. John Wiley & Sons, 1994. С. 25-62.
19. Pressman, R.S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
20. Kurose, J.F., & Ross, K.W. (2017). Computer Networking: A Top-Down Approach. Pearson.
21. Гриценко В.І., Михайленко І.П. Програмування на Java. Київ: Видавництво "Освіта", 2018. 300 с.
22. Бондаренко М.І., Петренко В.Ю. Розробка мобільних додатків: Теорія та практика. Львів: Видавництво "Львівська Політехніка", 2021. 400 с.
23. Воронцов Д.С. Технології розробки мобільних застосунків. Дніпро: Видавництво "Дніпровський національний університет", 2022. 350 с.
24. Шаронов В.П. Основи проектування користувацьких інтерфейсів. Київ: Видавництво "Наукова думка", 2019. 280 с.
25. Фаулер М., Райс Д., Фаулер Дж., Міллер Д. Сервіси, архітектури та бізнес-логіка. Мікросервіси з практикою: Будівництво гнучких систем. Київ: Видавництво "К.І.С.", 2019. 464 с.

ДОДАТОК А. Опис програми

А.1 Загальні відомості

Позначення та найменування програми: мобільний додаток для пошуку попутників. Найменування програми «Common Way».

Програмне забезпечення, необхідне для функціонування програми: для функціонування програми необхідно мати смартфон на базі OS Android з мінімальною версією 5.0.

Мови програмування, на яких написана програма: програма написана у середовищі програмування Android Studio на мові Java.

А.2 Функціональне призначення

Додаток призначений для користувачів смартфонів на базі OS Android, які мають змогу шукати, створювати і бронювати поїздки на потрібну дату та час.

А.3 Структура програми з описом функцій складових частин і зв'язки між ними

1. Реєстрація користувача:

   Вхідні дані: email, пароль, ім’я, вік, телефон.

   Вихідні дані: акаунт користувача.

2. Авторизація користувача:

   Вхідні дані: логін, пароль.

   Вихідні дані: аккаунт клієнта.

3. Додавання автомобіля:

   Вхідні дані: номер, колір, марка, модель, рік випуску.

   Вихідні дані: доданий автомобіль

4. Пошук поїздки:

   Вхідні дані: місце відправлення, місце призначення, дата.

   Вихідні дані: знайдені поїздки.

5. Створення поїздки:

   Вхідні дані: місце відправлення, місце призначення, дата відправлення, час відправлення, ціна.

   Вихідні дані: створена поїздка.

6. Створення відгуків;

Вхідні дані: обраний профіль, оцінка, повідомлення.

Вихідні дані: створений відгук.

А.4 Використовувані технічні засоби

Технічним засобом може бути смартфон будь-якого виробника з OS Android і мінімальною версією OS – 5.0.

А.5 Виклик і завантаження: виклик заздалегідь встановленої на смартфон програми виконується за допомогою іконки на робочому столі смартфона.

ДОДАТОК Б. Керівництво користувача

Для інсталяції розробленого ПЗ «Розробка мобільного додатку пошуку попутників» на OS Android повинен бути встановлений даний додаток «Common Way». Користувачем системи буде власник телефону, на якому встановлений даний додаток.

Запуск програми здійснюється за допомогою ярлика на робочому столі телефону.

Головне вікно представлено на рисунку Б.1.

Після запуску додатку відкривається головне вікно системи. Система надає можливість обрати функцію авторизації або реєстрації.

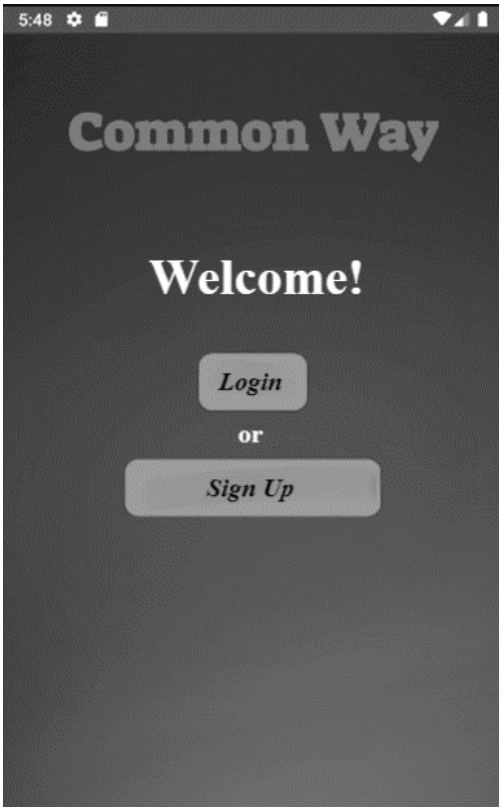

Рисунок Б.1 - Головне вікно

Форма авторизації представлена на рисунку Б.2.

Після натискання на кнопку «Увійти» відкривається форма авторизації. Система надає можливість ввести адресу електронної пошти та пароль, або відмінити вхід.

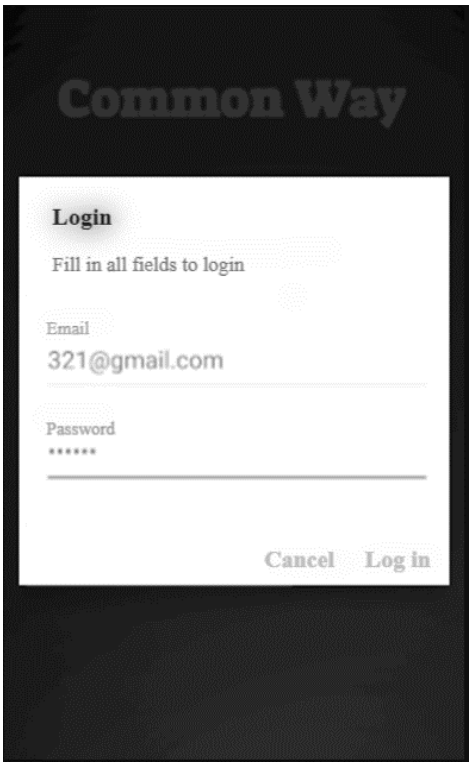

Рисунок Б.2 – Форма авторизації

Форма реєстрації представлена на рисунку Б.3.

Після натискання на кнопку «Зареєструватися» відкривається форма реєстрації. Система надає можливість ввести дані у всі необхідні поля, або відмінити вхід.

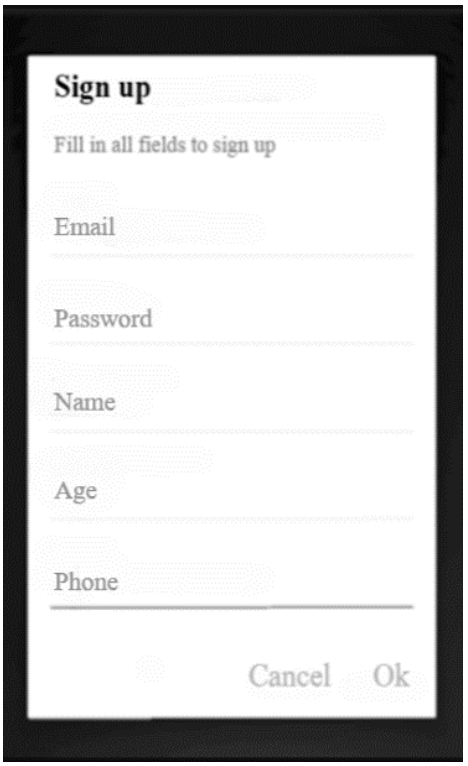

Рисунок Б.3 – Форма реєстрації

Вікно пошуку поїздки представлено на рисунку Б.4.

Після успішної авторизації або реєстрації користувач потрапляє на вікно пошуку поїздки, на якому знаходяться поля для вводу початкового і кінцевого населеного пункту та кнопка «Знайти».

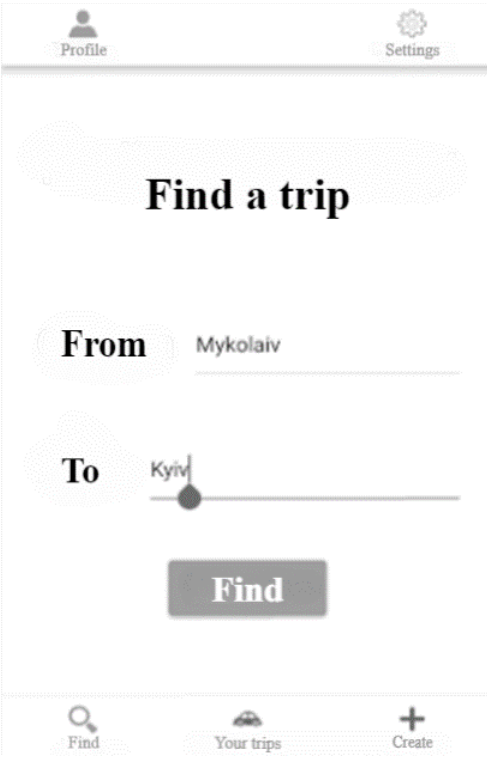

Рисунок Б.4 - Вікно пошуку поїздки

Вікно заброньованих поїздок користувача представлено на рисунку 3.5.

Користувач може переміщуватися по вікнах за допомогою нижнього меню.

Після переходу на вікно «Ваші поїздки» користувач потрапляє на вікно, у якому відображаються всі заброньовані ним поїздки.

Вікно створення поїздок представлено на рисунку Б.6.

Користувач може переміщуватися по вікнах за допомогою нижнього меню.

Після переходу на вікно «Створити» користувач потрапляє на вікно створення поїздки, на якому , на якому знаходяться поля для вводу початкового і кінцевого населеного пункту та кнопка «Створити».

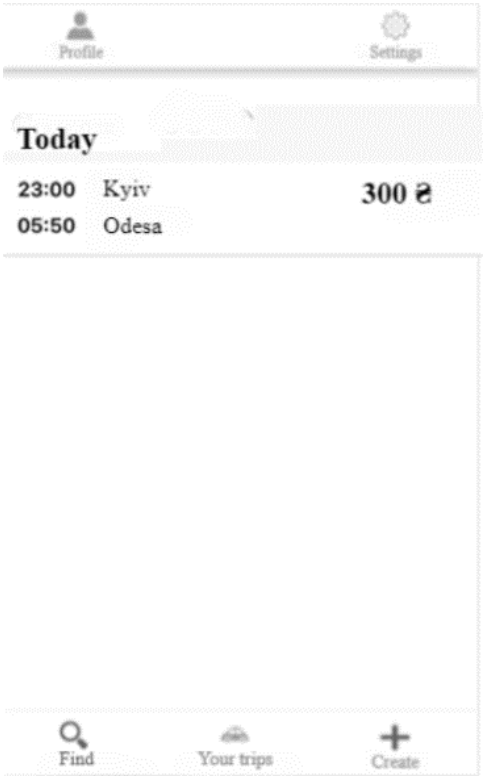

Рисунок Б.5 – Вікно поїздок користувача

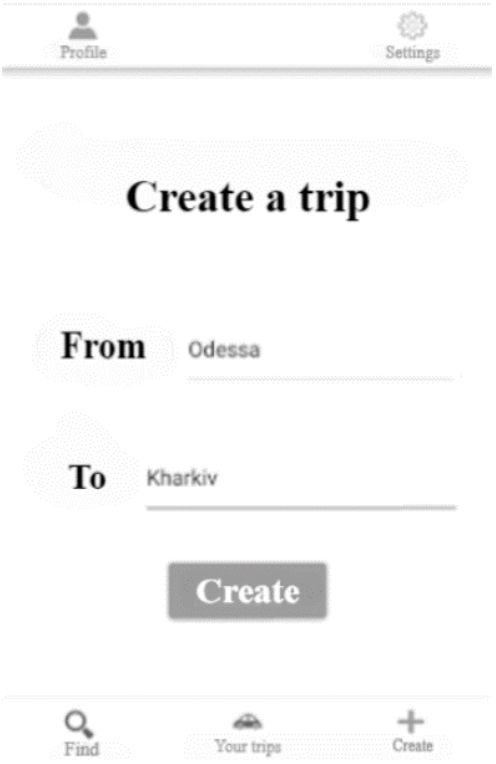

Рисунок Б.6 – Вікно створення поїздок

Вікно профілю користувача представлено на рисунку Б.7.

Після натискання на іконку профілю відображається спливаюче вікно профілю, на якому відображається вся інформація, яку користувач вводив при

реєстрації. Також є кнопка «Додати авто», яка дозволяє користувачу додавати автомобіль для подальшого створення поїздок.

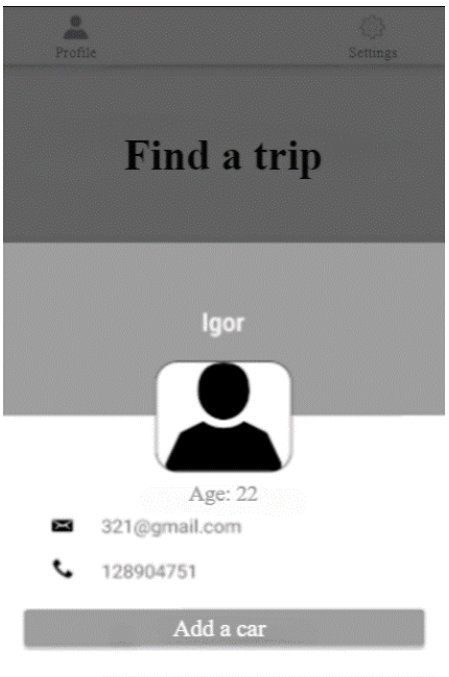

Рисунок Б.7 – Вікно профіля

Вікно налаштувань представлено на рисунку Б.8.

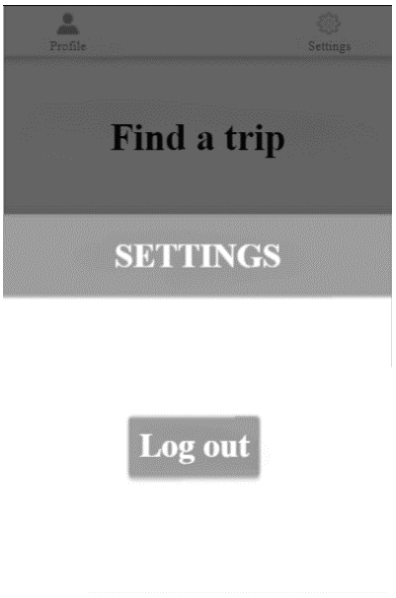

Рисунок Б.8 – Вікно налаштувань

Після натискання на іконку налаштувань відображається спливаюче вікно налаштувань, на якому відображається кнопка «Вийти», яка дозволяє користувачу вийти зі свого акаунту.

ДОДАТОК В. Код розробки

Лістинг В.1 – Клас «MainActivity»

```
package com.example.commonway;


import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.room.Room;


import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.Button;

import android.widget.RelativeLayout;


import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.android.material.snackbar.Snackbar;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.ValueEventListener;
```

```
import com.rengwuxian.materialedittext.MaterialEditText;


import java.util.Objects;
public class MainActivity extends AppCompatActivity {
    Button btnRegister, btnSignIn;
    FirebaseAuth auth;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference users;
    RelativeLayout root;
    AppDatabase db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        db = App.getInstance().getDatabase();


        btnSignIn = findViewById(R.id.btnSignIn);
        btnRegister = findViewById(R.id.btnRegister);
        root = findViewById(R.id.root_element);


        auth = FirebaseAuth.getInstance();
        firebaseDatabase = FirebaseDatabase.getInstance();
        users = firebaseDatabase.getReference("Users");


        btnRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
                showRegisterWindow();
            }
        });
        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showSignInWindow();
            }
        });
    }

    private void showSignInWindow() {
        AlertDialog.Builder dialog = new AlertDialog.Builder(this);
        dialog.setTitle("Login");
        dialog.setMessage("Fill in all fields to login");
        LayoutInflater inflater = LayoutInflater.from(this);
        View sing_in_window = inflater.inflate(R.layout.sing_in_window,null);
        dialog.setView(sing_in_window);

                                        final MaterialEditText email =
sing_in_window.findViewById(R.id.email_field);
                                        final MaterialEditText password =
sing_in_window.findViewById(R.id.pass_field);

        dialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
```

```
        });

        dialog.setPositiveButton("Login", new DialogInterface.OnClickListener() {

            @Override

            public void onClick(DialogInterface dialogInterface, int i) {

                if(email.getText() != null &&
TextUtils.isEmpty(email.getText().toString())){

                    Snackbar.make(root,"Enter your email",
Snackbar.LENGTH_SHORT).show();

                    return;

                }



                if(password.getText() != null &&
password.getText().toString().length() <= 5){

                    Snackbar.make(root,"The password should be over 5 symbols",
Snackbar.LENGTH_SHORT).show();

                    return;

                }


auth.signInWithEmailAndPassword(email.getText().toString(),password.getText().toStri
ng())

                        .addOnSuccessListener(new OnSuccessListener<AuthResult>() {

                            @Override

                            public void onSuccess(AuthResult authResult) {


FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getInstance(
)


.getCurrentUser().getUid()).addValueEventListener(new ValueEventListener() {

                                    @Override

                                    public void onDataChange(@NonNull DataSnapshot
snapshot) {

                                        User.currentUser =
snapshot.getValue(User.class);

                                        startActivity(new Intent(MainActivity.this,
StartActivity.class));
```

```
                                        finish();
                                    }


                                    @Override
                                    public void onCancelled(@NonNull DatabaseError
error) {


                                    }
                                });
                            }
                        }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Snackbar.make(root,"Error of authentication" +
e.getMessage(),Snackbar.LENGTH_SHORT).show();
                    }
                });
            }
        });


        dialog.show();
    }


    private void showRegisterWindow() {
        AlertDialog.Builder dialog = new AlertDialog.Builder(this);
        dialog.setTitle("Sign Up");
        dialog.setMessage("Fill in all fields to sign up");
        LayoutInflater inflater = LayoutInflater.from(this);
        View registerWindow = inflater.inflate(R.layout.register_window,null);
```

```
        dialog.setView(registerWindow);


        MaterialEditText email = registerWindow.findViewById(R.id.email_field);
        MaterialEditText password = registerWindow.findViewById(R.id.pass_field);
        MaterialEditText name = registerWindow.findViewById(R.id.name_field);
        MaterialEditText phone = registerWindow.findViewById(R.id.phone_field);
        MaterialEditText age = registerWindow.findViewById(R.id.age_field);


        dialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        dialog.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                if(email.getText() != null &&
TextUtils.isEmpty(email.getText().toString())){
                    Snackbar.make(root,"Enter your email",
Snackbar.LENGTH_SHORT).show();
                    return;
                }
                if(name.getText() != null &&
TextUtils.isEmpty(name.getText().toString())){
                    Snackbar.make(root,"Enter your name",
Snackbar.LENGTH_SHORT).show();
                    return;
                }
                if(age.getText() != null &&
TextUtils.isEmpty(age.getText().toString())){
```

```
                    Snackbar.make(root,"Enter your age",
Snackbar.LENGTH_SHORT).show();

                    return;

                }

                if(phone.getText() != null &&
TextUtils.isEmpty(phone.getText().toString())){

                    Snackbar.make(root,"Enter your phone",
Snackbar.LENGTH_SHORT).show();

                    return;

                }

                if(password.getText() != null &&
password.getText().toString().length() <= 5){

                    Snackbar.make(root,"Password should be over 5 symbols",
Snackbar.LENGTH_SHORT).show();

                    return;

                }

            // Registration

            auth.createUserWithEmailAndPassword(email.getText().toString(),

                    password.getText().toString())

                    .addOnSuccessListener(new OnSuccessListener<AuthResult>() {

                        @Override

                        public void onSuccess(AuthResult authResult) {

                            User user = new User();

                            user.setEmail(email.getText().toString());


user.setName(Objects.requireNonNull(name.getText()).toString());

                            user.setPass(password.getText().toString());


user.setPhone(Objects.requireNonNull(phone.getText()).toString());

                            user.setAge(age.getText().toString());


users.child(Objects.requireNonNull(FirebaseAuth.getInstance().getCurrentUser()).getU
id())
```

```
                                    .setValue(user)
                                                        .addOnSuccessListener(new
OnSuccessListener<Void>() {
                                @Override
                                public void onSuccess(Void aVoid) {
                                        Snackbar.make(root,"Registration is finished
successfully", Snackbar.LENGTH_SHORT);
                                    User.currentUser = user;
                                    UserDao userDao = db.userDao();
                                    userDao.insert(user);
                                        startActivity(new Intent(MainActivity.this,
StartActivity.class));
                                    finish();
                                }
                            });
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                                        Snackbar.make(root,"Error of sign up" +
e.getMessage(),Snackbar.LENGTH_SHORT).show();
                }
            });
            }
        });
        dialog.show();
    }
}
```

Лістинг В.2 – Клас «StartActivity»

```
package com.example.commonway;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.fragment.app.Fragment;
import android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class StartActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_start);

        AppDatabase db = App.getInstance().getDatabase();
        UserDao userDao = db.userDao();
        List<User> userList = userDao.getAll();

        BottomNavigationView bottomNavigationView =
findViewById(R.id.bottom_navigation);
        BottomNavigationView topNavigationView = findViewById(R.id.top_navigation);

        bottomNavigationView.setOnNavigationItemSelectedListener(bottomNavListener);
        topNavigationView.setOnNavigationItemSelectedListener(topNavListener);


getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                new FindFragment()).commit();
    }

    private BottomNavigationView.OnNavigationItemSelectedListener bottomNavListener
=
            new BottomNavigationView.OnNavigationItemSelectedListener() {
                @SuppressLint("NonConstantResourceId")
                @Override
                public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                    Fragment selectedFragment = null;
                    switch (item.getItemId()) {
                        case R.id.addFragment:
                            selectedFragment = new AddFragment();
                            break;
                        case R.id.findFragment:
                            selectedFragment = new FindFragment();
                            break;
                        case R.id.tripsFragment:
                            selectedFragment = new TripsFragment();
                            break;
                    }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                            selectedFragment).commit();
                    return true;
                }
            };
    private BottomNavigationView.OnNavigationItemSelectedListener topNavListener =
            new BottomNavigationView.OnNavigationItemSelectedListener() {
                @SuppressLint("NonConstantResourceId")
                @Override
                public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                    switch (item.getItemId()) {
```

```
                        case R.id.profileFragment:
                            ProfileBottomSheetDialog profileBottomSheet = new
ProfileBottomSheetDialog();

profileBottomSheet.show(getSupportFragmentManager(),"profileBottomSheet");
                            break;
                        case R.id.settingsFragment:
                            SettingsBottomSheetDialog settingsBottomSheet = new
SettingsBottomSheetDialog();

settingsBottomSheet.show(getSupportFragmentManager(),"settingsBottomSheet");
                            break;
                    }
                    return true;
                }
            };
}
```

Лістинг В.3 – Клас «User»

```
package com.example.commonway;

import androidx.room.Entity;
import androidx.room.Ignore;
import androidx.room.PrimaryKey;

@Entity
public class User {
    @PrimaryKey (autoGenerate = true)
    public long id;
    public String email, name, phone, age;
    @Ignore
    public String pass;
    @Ignore
    public static User currentUser;

    public User() {
    }

    public User(String email, String name, String phone, String pass, String age) {
        this.email = email;
        this.name = name;
        this.phone = phone;
        this.pass = pass;
        this.age = age;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPass() {
        return pass;
    }

    public void setPass(String pass) {
        this.pass = pass;
    }
}
```

Лістинг В.4 – Клас «AddFragment»

```
package com.example.commonway;

import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class AddFragment extends Fragment {

    public AddFragment() {}

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_add, container, false);
    }
}
```

Лістинг В.5 – Клас «FindFragment»

```
package com.example.commonway;


import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;
```

```
import android.view.ViewGroup;

public class FindFragment extends Fragment {

    public FindFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_find, container, false);
    }
}
```

Лістинг В.6 – Клас «TripsFragment»

```
package com.example.commonway;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class FindFragment extends Fragment {

    public FindFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_find, container, false);
    }
```

```
}package com.example.commonway;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TripsFragment extends Fragment {

    public TripsFragment() {
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_trips, container, false);
    }
}
```

Лістинг В.7 – Клас « ProfileBottomSheetDialog»

```
package com.example.commonway;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;

import android.text.TextUtils;
import android.util.Log;
import android.view.ContextThemeWrapper;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.example.commonway.aboutCar.Brand;
import com.example.commonway.aboutCar.BrandDao;
import com.example.commonway.aboutCar.Car;
import com.example.commonway.aboutCar.CarDao;
import com.example.commonway.aboutCar.Model;
```

```
import com.example.commonway.aboutCar.ModelDao;
import com.example.commonway.aboutCar.Type;
import com.example.commonway.aboutCar.TypeDao;
import com.google.android.material.bottomsheet.BottomSheetDialogFragment;
import com.google.android.material.dialog.MaterialDialogs;
import com.google.android.material.snackbar.Snackbar;
import com.rengwuxian.materialedittext.MaterialEditText;

import java.util.List;

public class ProfileBottomSheetDialog extends BottomSheetDialogFragment {

    private TextView nameTextView, ageTextView, phoneTextView, emailTextView;
    private Button addCar, btnAddCar, cancelAddCar;
    private MaterialEditText carNumber, carBrand, carModel, carColor, carType,
carPlacesCount, manufactureYear;
    RelativeLayout root;
    AppDatabase db;
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        db = App.getInstance().getDatabase();
    }

    @SuppressLint("SetTextI18n")
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.bottom_sheet_profile, container, false);

        nameTextView = v.findViewById(R.id.name_textview);
        ageTextView = v.findViewById(R.id.age_textView);
        phoneTextView = v.findViewById(R.id.phone_textview);
        emailTextView = v.findViewById(R.id.email_textview);
        addCar = v.findViewById(R.id.add_car_button);

        addCar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showAddCarWindow();
            }
        });
        nameTextView.setText(User.currentUser.getName());
        emailTextView.setText(User.currentUser.getEmail());
        phoneTextView.setText(User.currentUser.getPhone());
        ageTextView.setText("Age: " + User.currentUser.getAge());
        return v;
    }

    @SuppressLint("ResourceType")
    private void showAddCarWindow() {
        AlertDialog.Builder dialog = new AlertDialog.Builder(getActivity());
        dialog.setTitle("Add a car");
        LayoutInflater inflater = LayoutInflater.from(getActivity());
        View addCarWindow = inflater.inflate(R.layout.add_car_window,null);
        dialog.setView(addCarWindow);

        carNumber = addCarWindow.findViewById(R.id.car_number_field);
        carBrand = addCarWindow.findViewById(R.id.brand_field);
        carModel = addCarWindow.findViewById(R.id.model_field);
        carPlacesCount = addCarWindow.findViewById(R.id.places_count_field);
        carType = addCarWindow.findViewById(R.id.car_type_field);
        carColor = addCarWindow.findViewById(R.id.car_color_field);
```

```
        manufactureYear = addCarWindow.findViewById(R.id.manufacture_year_field);


        dialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        dialog.setPositiveButton("Add", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                if(carNumber.getText() != null &&
TextUtils.isEmpty(carNumber.getText().toString())||
                (carBrand.getText() != null &&
TextUtils.isEmpty(carBrand.getText().toString()))||
                (carModel.getText() != null &&
TextUtils.isEmpty(carModel.getText().toString()))||
                (carPlacesCount.getText() != null &&
TextUtils.isEmpty(carPlacesCount.getText().toString()))||
                (carType.getText() != null &&
TextUtils.isEmpty(carType.getText().toString()))||
                (carColor.getText() != null &&
TextUtils.isEmpty(carColor.getText().toString()))||
                (manufactureYear.getText() != null &&
TextUtils.isEmpty(carColor.getText().toString()))){
                    Snackbar.make(root,"Fill in all fields!",
Snackbar.LENGTH_SHORT).show();
                    return;
                }
                Type type = new
Type(carType.getText().toString(),Integer.parseInt(carPlacesCount.getText().toString
()));
                TypeDao typeDao = db.typeDao();
                typeDao.insert(type);

                Brand brand = new Brand(carBrand.getText().toString());
                BrandDao brandDao = db.brandDao();
                brandDao.insert(brand);

                Model model = new
Model(carModel.getText().toString(),brandDao.getId(brand.getName()));
                ModelDao modelDao = db.modelDao();
                modelDao.insert(model);

                Car car = new Car();
                car.brandId = brandDao.getId(brand.getName());
                car.typeId = typeDao.getId(type.getName());
                car.setNumber(carNumber.getText().toString());
                car.setColor(carColor.getText().toString());

car.setManufactureYear(Integer.parseInt(manufactureYear.getText().toString()));
                CarDao carDao = db.carDao();
                carDao.insert(car);

            }
        });
        dialog.show();
    }
}
```

Лістинг В.8 – Клас « SettingsBottomSheetDialog»

```
package com.example.commonway;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import com.google.android.material.bottomsheet.BottomSheetDialogFragment;

public class SettingsBottomSheetDialog extends BottomSheetDialogFragment {

    public SettingsBottomSheetDialog() {

    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.bottom_sheet_settings, container, false);
        Button exitButton = v.findViewById(R.id.exit_button);

        exitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(getContext(), MainActivity.class));
            }
        });

        return v;
    }
}
```