Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка мобільного додатку для пошуку попутників з використанням Google AppSheet

Виконав(ла): студент(ка) 4 курсу, групи СП-43 спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Черевайко С. О.
(прізвище та ініціали)

Керівник (підпис) Бревус В. М.
(прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М.
(прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р.
(прізвище та ініціали)

Рецензент (підпис) Стадник М.А.
(прізвище та ініціали)

Тернопіль
2025

АНОТАЦІЯ

Кваліфікаційна робота на здобуття освітнього ступеню «бакалавр» за спеціальністю 121 Інженерія програмного забезпечення. Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група: СП-43, 2025 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеня «бакалавр» містить: 62 с., 12 рис.

Тема: Розробка мобільного додатку для пошуку попутників з використанням Google AppSheet.

Об'єктом дослідження є процес створення програмного забезпечення для організації поїздок попутників, з урахуванням сучасних підходів до безкодової (no-code) розробки.

Метод дослідження – порівняльний аналіз існуючих платформ розробки мобільних додатків, розробка логіки, архітектури та інтерфейсу додатку на основі Google AppSheet, моделювання основних бізнес-процесів і поведінки користувача.

Завданням проєкту є створення прототипу мобільного застосунку для пошуку та пропонування поїздок, з фільтрацією маршрутів, переглядом профілів, обміном інформацією та автоматизацією дій без програмування.

У результаті реалізовано структуру додатку, протестовано функціонал, обґрунтовано вибір Google AppSheet як оптимальної платформи для швидкої розробки MVP.

Наукова новизна – полягає в інтеграції no-code підходу з функціональністю сервісу попутників, що спрощує створення мобільних додатків без традиційного коду.

Практичне значення – прототип може бути основою для реального впровадження подібних сервісів або використання у навчальному процесі.

Ключові слова: AppSheet, мобільний додаток, пошук попутників, no-code, Google Sheets, цифровізація, автоматизація.

ABSTRACT

Qualification paper for obtaining the bachelor's degree in specialty 121 Software Engineering. Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-43, 2025. The explanatory note to the bachelor’s qualification paper contains: 62 pages, 12 figures.

Title: Development of a Mobile Application for Ride-Sharing Using Google AppSheet.

Object of research is the process of creating software for organizing shared trips using modern no-code development tools.

Research method – comparative analysis of existing mobile application development platforms, development of application logic, architecture, and user interface using Google AppSheet, as well as modeling of key business processes and user scenarios.

The purpose of the project is to develop a mobile application prototype for finding and offering rides, filtering routes, viewing profiles, exchanging information, and automating processes without traditional coding.

As a result, the application structure was implemented, core functionality tested, and the choice of Google AppSheet justified as an optimal platform for fast MVP development.

Scientific novelty – lies in the integration of the no-code approach with ride-sharing functionality, which simplifies mobile application development without traditional programming.

Practical significance – the prototype can serve as a basis for real-world implementation of similar services or be used as a learning example in educational settings.

Keywords: AppSheet, mobile application, ride-sharing, no-code, Google Sheets, digitalization, automation.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій мобільні додатки стають невіддільною частиною повсякденного життя. Вони суттєво спрощують виконання різноманітних завдань – від банківських операцій до організації спільних поїздок. Розвиток інфраструктури мобільних сервісів сприяє зростанню попиту на зручні, інтуїтивно зрозумілі та ефективні інструменти для вирішення побутових проблем. Однією з актуальних потреб суспільства є оптимізація перевезень за принципом «carpooling» – пошуку попутників для спільних поїздок. Це дозволяє не лише зекономити ресурси, але й зменшити навантаження на транспортну систему та навколишнє середовище. У цьому контексті набуває особливої актуальності створення доступного мобільного сервісу для пошуку попутників із використанням сучасних no-code платформ, таких як Google AppSheet.

Актуальність теми зумовлена потребою у швидких та економічних рішеннях, які дозволяють ефективно розробляти мобільні додатки без залучення значних фінансових і технічних ресурсів. Використання платформ без програмування відкриває нові можливості для розробників-початківців, студентів та малого бізнесу. Зокрема, Google AppSheet, як одна з провідних no-code платформ, дозволяє створювати повноцінні мобільні додатки, інтегровані з Google-сервісами, без потреби у традиційній розробці на мовах програмування. Це значно скорочує час розробки, знижує поріг входу та забезпечує ширший доступ до інноваційних технологій. Цей підхід особливо корисний в умовах обмежених ресурсів, коли необхідно швидко протестувати ідею або створити MVP. Крім того, такі платформи сприяють діджиталізації процесів у сферах, де раніше це було технічно складно.

Метою дослідження є теоретичне обґрунтування, проектування та моделювання мобільного додатку для пошуку попутників на базі Google AppSheet, а також аналіз його переваг порівняно з іншими сучасними інструментами розробки.

У межах досягнення мети передбачено виконання таких завдань:

- проаналізувати існуючі сервіси пошуку попутників;
- визначити вимоги до мобільного додатку такого типу;
- обгрунтувати вибір Google AppSheet як платформи розробки;
- розробити архітектурну модель додатку; описати інтерфейс та логіку його роботи;
- провести порівняльний аналіз з іншими інструментами no-code;
- сформулювати висновки щодо ефективності обраного підходу.

Об’єкт дослідження: процес створення мобільних додатків засобами платформ без кодування.

Предмет дослідження: технологічні, архітектурні та функціональні особливості розробки додатку для пошуку попутників із використанням Google AppSheet.

Методологічну основу роботи становить сукупність загальнонаукових методів, таких як аналіз, синтез, моделювання, порівняння та систематизація. Також у роботі застосовано методи структурного та функціонального проектування, моделювання бізнес-процесів та дослідження інтерфейсних рішень. Окрему роль відіграє порівняльний аналіз Google AppSheet та альтернативних платформ, що дозволяє обґрунтовано оцінити їхні можливості в контексті поставлених задач.

Структура роботи: робота складається зі вступу, чотирьох розділів, висновків і списку використаних джерел. У першому розділі розглядаються існуючі сервіси та підходи до створення подібних додатків. Другий розділ присвячено проектуванню архітектури майбутнього мобільного додатку на базі Google AppSheet. У третьому розділі описано теоретичний процес реалізації та тестування функціоналу, а також наведено діаграми для візуалізації логіки та структури додатку. Четвертий розділ містить питання безпеки життєдіяльності та охорони праці при роботі з цифровими технологіями. Загальний обсяг роботи – 62 сторінка.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд сервісів пошуку попутників

У сучасному світі дедалі більше уваги приділяється розвитку сервісів спільного користування транспортом. Це пов’язано як із прагненням до економії коштів на поїздках, так і з бажанням знизити екологічне навантаження на навколишнє середовище. Одним із таких рішень є платформи для пошуку попутників, що дозволяють об’єднувати водіїв та пасажирів із подібними маршрутами. Для того щоб краще зрозуміти особливості та потенціал розробки мобільного додатку такого типу, доцільно розглянути функціональні можливості вже існуючих рішень, зокрема таких як BlaBlaCar, Liftshare та Poparide. Кожен із цих сервісів займає важливу нішу в сегменті carpooling-додатків та має свої унікальні риси, що визначають його зручність, надійність і популярність серед користувачів.

BlaBlaCar є одним із найвідоміших і найбільш масштабних сервісів для пошуку попутників у Європі. Основна функціональність платформи полягає у можливості створення водієм маршруту із зазначенням дати, часу відправлення, кількості вільних місць і вартості поїздки. Пасажири можуть у зручному форматі здійснювати пошук маршрутів за допомогою фільтрів, а також переглядати профілі водіїв, що містять оцінки та відгуки інших користувачів. Окрему увагу BlaBlaCar приділяє безпеці: сервіс перевіряє особу користувача, а також надає функцію страхування поїздки у деяких країнах. Додаток має зручний мобільний інтерфейс, який підтримує багатомовність, GPS-навігацію, сповіщення, а також інтеграцію з платіжними системами, що дозволяє автоматизувати розрахунки між учасниками поїздки [5]. Інтерфейс додатку BlaBlaCar проілюстровано на наступній сторінці на рис. 1.1:

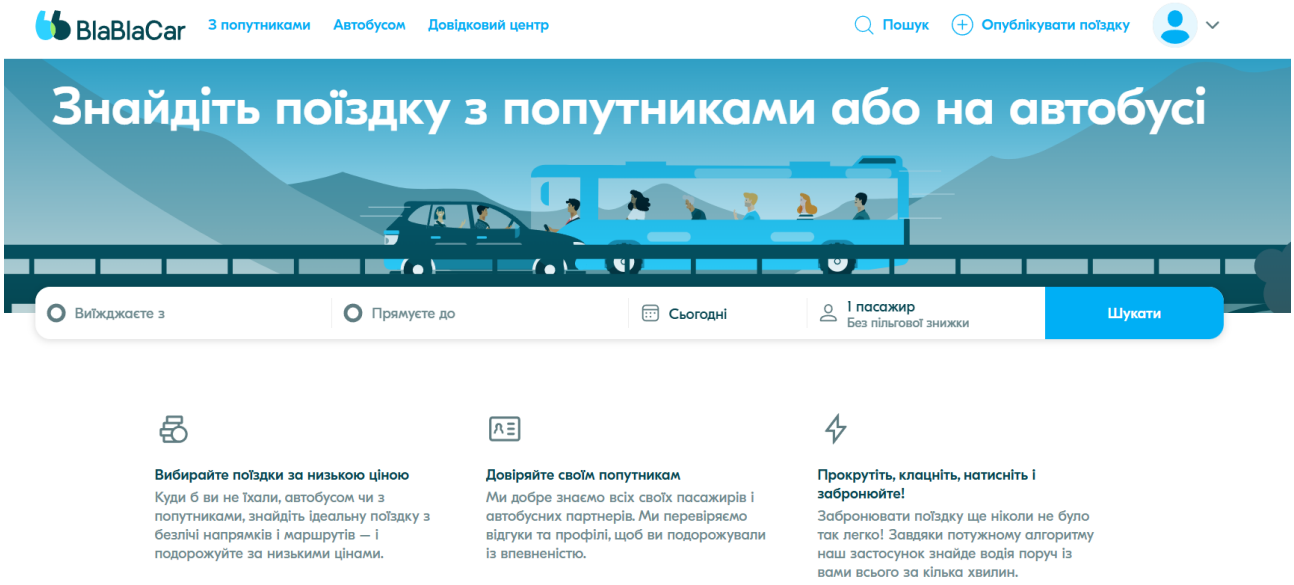

Рис. 1.1 – Інтерфейс додатку BlaBlaCar

Liftshare – це британський сервіс, орієнтований на організацію спільних поїздок як між приватними особами, так і в межах корпоративних програм. На відміну від BlaBlaCar, Liftshare активно працює з компаніями, організовуючи спільні поїздки співробітників до офісу або на підприємство. Основними функціями платформи є реєстрація маршрутів, фільтрація пасажирів або водіїв за географічним принципом, можливість узгодження часу виїзду, обговорення деталей поїздки в особистих повідомленнях. Liftshare, також, акцентує увагу на соціальному впливі – користувачам пропонуються звіти про зекономлені ресурси: зменшення кількості викидів CO_2, витрат на паливо тощо.

Poparide – канадський сервіс, що функціонує переважно у Північній Америці. Його відмінною рисою є орієнтація на довгі маршрути та міжміські поїздки. Попри схожість із BlaBlaCar у плані інтерфейсу та базових можливостей, Poparide пропонує інтуїтивно зрозумілу систему бронювання місць, а також сувору політику скасування поїздок, що сприяє підвищенню відповідальності користувачів. Додаток надає можливість перегляду детальної інформації про транспортний засіб, профіль водія з оцінками, а також підтримує електронні платежі з миттєвим підтвердженням бронювання (див. рис. 1.2) [6].

Інформація про поїздки зберігається в особистому кабінеті, що полегшує повторні подорожі тим самим маршрутом. Особливою перевагою Poparide є можливість аналізу історії поїздок та рівня задоволеності користувачів.

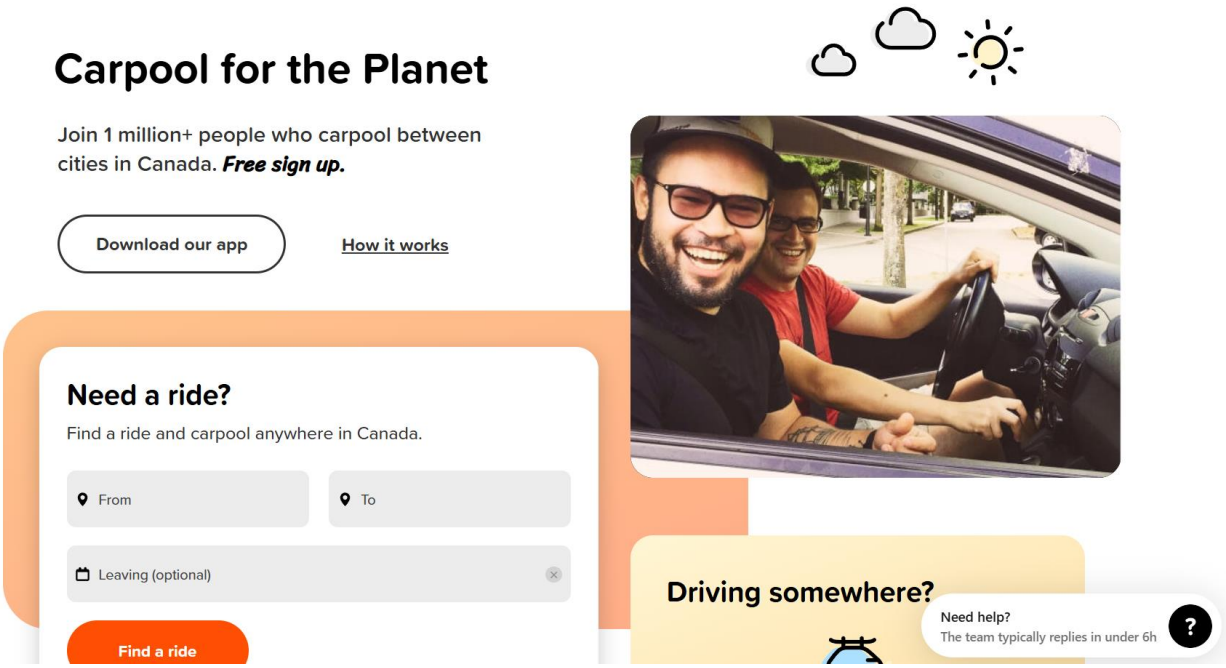

Рис. 1.2 – Інтерфейс додатку Poparide

Загалом, аналіз зазначених сервісів демонструє ключові функціональні компоненти, які повинні бути реалізовані в мобільному додатку для пошуку попутників. Це – зручна система реєстрації та авторизації, пошук за маршрутом, перегляд профілю користувача, рейтингова система, чат між водієм і пасажиром, інтеграція з платіжними сервісами та сповіщення. Існують додаткові можливості, як-от планування регулярних поїздок, оцінка впливу на довкілля, історія поїздок.

Особливої уваги заслуговує користувацький досвід – саме він значною мірою визначає успішність додатка на ринку. Простота інтерфейсу, швидкість реагування, інтуїтивність навігації та загальна надійність системи відіграють критичну роль у формуванні довіри з боку користувачів. Крім того, в умовах зростання конкуренції важливо не лише копіювати функції лідерів ринку, а й пропонувати унікальні рішення, які справді вирішують проблеми користувачів або додають цінності до їхнього досвіду [2].

Але, варто не лише розглядати їхній функціонал, але й оцінити ключові переваги та недоліки, що безпосередньо впливають на ефективність, зручність і конкурентоспроможність подібних рішень. Такий підхід дозволяє виявити як найкращі практики, які можна інтегрувати в новий додаток, так і типові проблеми, яких слід уникати під час проєктування власної системи. Зіставлення сильних і слабких сторін популярних платформ, таких як BlaBlaCar, Liftshare та Poparide, формує цілісне уявлення про очікування користувачів та прогалини в поточному ринку.

Однією з основних переваг розглянутих сервісів є високий рівень оптимізації користувацького досвіду. Зокрема, такі функції, як інтуїтивно зрозумілий інтерфейс, швидкий пошук маршрутів, система рейтингу користувачів і безпечні електронні платежі, значною мірою підвищують довіру до платформи та сприяють її популярності. Велике значення має також мобільна доступність сервісів – всі три додатки мають повнофункціональні мобільні версії, що дозволяє здійснювати бронювання в режимі реального часу та оперативно реагувати на зміни [3]. Додатковими плюсами є прозора система ціноутворення, наявність зворотного зв’язку, можливість планування як одноразових, так і регулярних поїздок, а також певний рівень персоналізації взаємодії (наприклад, відображення уподобань водія чи пасажира щодо музики, спілкування під час поїздки тощо).

Водночас існує низка недоліків, які актуалізують потребу у вдосконаленні подібних платформ. По-перше, глобальні сервіси не завжди враховують локальні особливості – зокрема, у сільських або малонаселених районах пошук попутників може бути утрудненим через обмежену кількість активних користувачів. По-друге, деякі сервіси вимагають обов’язкової верифікації через складні процедури, що може відлякувати частину потенційної аудиторії. Також важливим мінусом є залежність від постійного інтернет-з’єднання та складність користування для літніх або технологічно необізнаних осіб. Ще однією проблемою є іноді недостатній рівень перевірки безпеки поїздок або поверхнева модерація контенту профілів, що створює ризики для учасників подорожей.

1.2 Вимоги до мобільного додатку

У процесі розробки будь-якого мобільного додатку надзвичайно важливим етапом є формування чітких функціональних вимог, які відображають основні сценарії взаємодії користувача з системою. Саме ці вимоги визначають логіку роботи додатку, його базову архітектуру та обсяг функціоналу, що реалізується на першому етапі розробки. У випадку сервісу для пошуку попутників, побудованого на платформі Google AppSheet, функціональні вимоги повинні охоплювати всі ключові елементи, що забезпечують ефективне з'єднання водіїв та пасажирів, інтуїтивну зручність, а також оперативність пошуку та обміну інформацією.

Однією з головних функцій додатку є реєстрація та авторизація користувачів. Цей компонент має забезпечувати як простоту, так і безпечність входу до системи. Залежно від можливостей AppSheet, реєстрація може бути реалізована через інтеграцію з Google-акаунтом або за допомогою введення електронної пошти з підтвердженням особи. Успішна авторизація відкриває доступ до персонального кабінету, в якому користувач може створювати поїздки або шукати доступні маршрути.

Ключовим функціональним блоком є пошук маршрутів, що забезпечує можливість швидко знайти попутників за визначеним напрямком. Система повинна дозволяти введення місця відправлення, пункту призначення, дати та часу поїздки, з подальшою фільтрацією результатів. Інтерфейс пошуку повинен бути зрозумілим і компактним, а результати – відображатися у вигляді списку з коротким описом поїздки, контактами водія або пасажира.

Ще одним важливим компонентом є створення та редагування оголошень про поїздки. Зареєстровані водії мають змогу вносити інформацію про плановані поїздки, зокрема вказувати маршрут, час відправлення, кількість вільних місць, вартість проїзду та особливі умови (наприклад, наявність багажу, уподобання щодо спілкування під час поїздки тощо). З іншого боку, пасажири можуть публікувати запити на поїздки, на які можуть відгукуватися водії.

Система сповіщень відіграє важливу роль у підтриманні актуальності інформації. В AppSheet можлива реалізація повідомлень через електронну пошту або push-нотифікації на мобільному пристрої. Вони інформують користувача про нові запити, підтвердження бронювання, зміни в маршруті або повідомлення від інших учасників.

Окрім цього, важливо реалізувати систему рейтингу та відгуків, яка дозволяє користувачам оцінювати досвід співпраці після завершення поїздки. Таке рішення не лише сприяє підвищенню рівня довіри до сервісу, але й виконує функцію своєрідної модерації, адже користувачі з низьким рейтингом автоматично отримують менше шансів бути обраними.

Функціональні вимоги, теж, мають враховувати можливість перегляду історії поїздок, збереження обраних маршрутів, фільтрацію оголошень за критеріями (ціна, дата, тривалість), підтримку декількох мов інтерфейсу (у разі масштабування) та простий механізм звернення до служби підтримки або адміністратора платформи.

Не менш важливим аспектом є нефункціональні вимоги, які визначають загальні характеристики системи та якість її роботи. Саме нефункціональні параметри формують користувацьке враження про сервіс, впливають на рівень довіри до нього, а також безпосередньо позначаються на ефективності використання платформи в умовах реального часу. До таких вимог належать масштабованість, стабільність, швидкодія, безпека даних та доступність сервісу на різних пристроях.

Однією з ключових нефункціональних вимог є зручність інтерфейсу користувача. Враховуючи, що платформа AppSheet орієнтована на створення мобільних додатків із мінімальним кодом або взагалі без нього, особлива увага приділяється логічній структурі екранів, зрозумілому розміщенню елементів управління та інтуїтивній навігації. Інтерфейс має бути адаптований для різних типів користувачів – як водіїв, так і пасажирів – а також підтримувати коректне відображення на різних пристроях (смартфонах, планшетах). Мінімалізм, візуальна чистота та швидкий доступ до основних функцій спрощують користування.

Не менш важливою вимогою є швидкодія додатку. Незважаючи на те, що AppSheet не надає повного контролю над серверною частиною або базами даних, оптимізація структури таблиць, застосування ефективних фільтрів даних та грамотне використання кешування можуть значно покращити швидкість відгуку системи. Особливо це актуально для таких дій, як пошук маршрутів або обробка запитів на поїздки, де затримки навіть у кілька секунд можуть призвести до зниження задоволеності користувача. Крім того, варто враховувати обмеження мобільного інтернету, адже у багатьох користувачів може бути нестабільне або повільне з'єднання. У такому випадку критично важливо, щоб додаток залишався функціональним і не викликав фрустрації навіть при обмежених ресурсах.

Ще одним визначальним аспектом є забезпечення безпеки даних і конфіденційності користувачів. У межах AppSheet доступна інтеграція з базами даних Google (Google Sheets, Google Cloud SQL), де можна реалізувати контроль доступу, обмеження прав користувачів та автентифікацію через обліковий запис (див. рис. 1.3) [7].

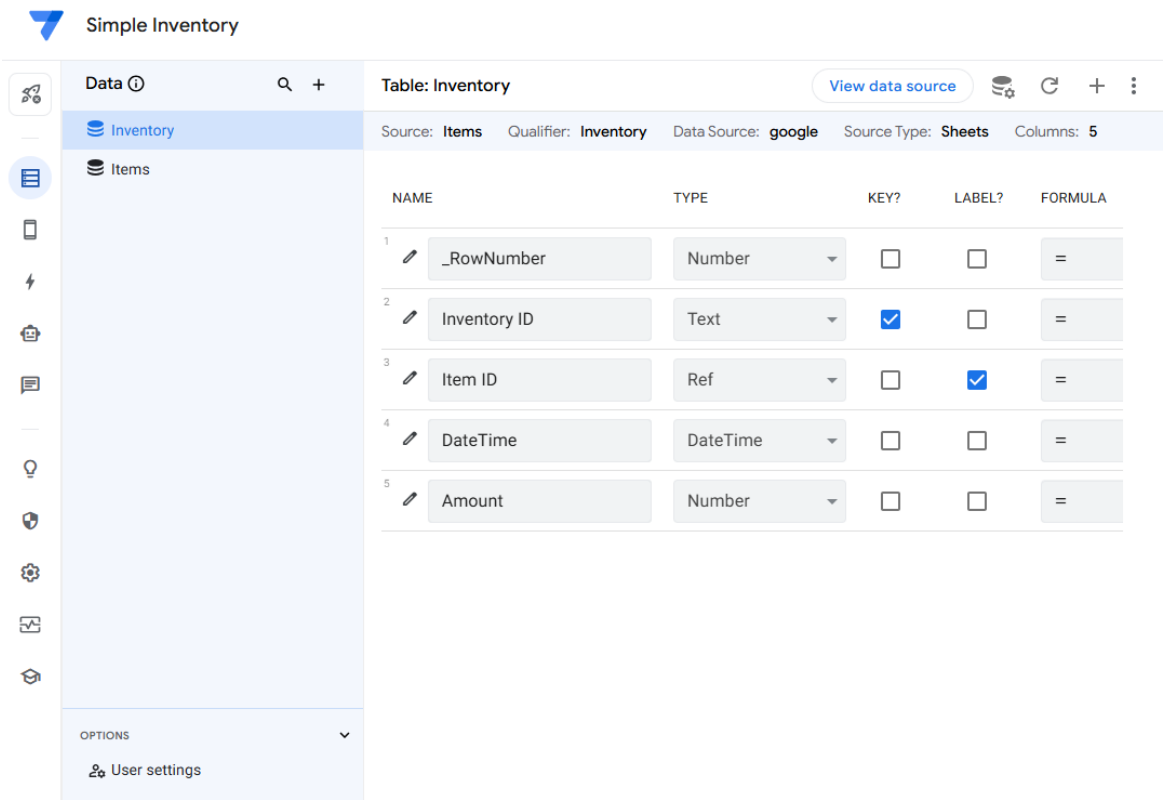

Рис. 1.3 – Приклад інтеграції Google AppSheet з базою даних типу SQL

Окрім цього, платформа підтримує SSL-шифрування під час обміну даними, що гарантує захист персональної інформації (імен, контактів, маршрутів). Система також має забезпечити розмежування ролей користувачів: наприклад, доступ до адміністративних функцій має бути відкритим лише для модераторів або розробників.

До основних вимог слід враховувати й стійкість до помилок і відмов, можливість швидкого відновлення після збоїв, а також масштабованість рішення – у разі збільшення кількості користувачів або географії сервісу. Незважаючи на певні обмеження AppSheet порівняно з нативною розробкою, грамотно сформульовані нефункціональні вимоги дозволяють створити на її основі надійний, зручний і конкурентоспроможний продукт, орієнтований на масове використання в реальних умовах [9].

1.3 Вибір методології розробки ПЗ

В процесі створення мобільного додатку для пошуку попутників важливо не лише визначити функціональні та нефункціональні вимоги, а й обрати відповідну методологію розробки програмного забезпечення, яка б найкраще відповідала технічним умовам, ресурсним можливостям та особливостям платформи, що використовується. Підбір правильної методології безпосередньо впливає на ефективність реалізації проєкту, швидкість розробки, гнучкість у внесенні змін, а також якість кінцевого продукту. У сучасній практиці найчастіше використовуються підходи Waterfall, Agile, а також концепції безкодової та малокодової розробки (No-Code/Low-Code), які особливо актуальні у контексті платформи Google AppSheet [11].

Waterfall, або каскадна модель розробки, передбачає послідовне проходження всіх етапів створення програмного забезпечення – від аналізу вимог до тестування і впровадження. Цей підхід є лінійним, а отже, кожен етап має бути

завершений до переходу на наступний. Waterfall відзначається високим рівнем формалізації, чіткістю документації та передбачуваністю результатів. Але, у випадку з динамічними або швидкозмінними проєктами, до яких належать додатки для мобільних пристроїв, ця модель виявляється недостатньо гнучкою. Будь-які зміни вимог на пізніх етапах ведуть до значного перевантаження розробки, що робить Waterfall менш придатним для реалізації проекту через AppSheet, де швидке реагування на зворотній зв’язок є ключовим.

Натомість Agile (гнучка методологія) передбачає ітеративну, поступову розробку з частими перевірками, активною участю замовника та постійним вдосконаленням продукту. Вона базується на принципах адаптивності, гнучкості й колективної відповідальності. У межах Agile розробка ведеться короткими циклами (спринтами), що дозволяє на кожному етапі тестувати функціональність, отримувати фідбек та коригувати подальші дії. Такий підхід добре підходить для додатків, функціональність яких може змінюватися залежно від потреб користувачів або ринкових умов.

Особливе місце серед сучасних підходів займають No-Code та Low-Code технології, до яких належить Google AppSheet. Ці моделі передбачають розробку програмних продуктів без необхідності традиційного програмування (No-Code) або з мінімальним залученням коду (Low-Code). Такий підхід значно скорочує час на розробку, знижує вартість впровадження та робить створення додатків доступним навіть для користувачів без глибоких технічних знань. В основі AppSheet лежить саме філософія No-Code, яка дозволяє швидко моделювати інтерфейс, структуру даних і логіку додатку на основі Google Sheets, Excel, SQL або інших джерел даних.

У сучасному цифровому середовищі, де швидкість розробки, адаптивність до змін і доступність інструментів є вирішальними факторами успішності проєкту, вибір відповідної технологічної платформи має стратегічне значення. Ураховуючи характер дослідження, функціональні потреби майбутнього додатку для пошуку попутників, а також аналіз різних підходів до розробки, логічним і обґрунтованим рішенням стало застосування Google AppSheet як основної платформи для проєктування і створення цифрового продукту.

AppSheet – це інноваційна хмарна платформа, орієнтована на безкодову розробку мобільних і вебзастосунків, яка дозволяє створювати повноцінні продукти без необхідності програмування традиційним способом. Її інтеграція з екосистемою Google (зокрема Google Sheets, Google Drive, BigQuery тощо) відкриває широкі можливості для ефективної роботи з даними, що особливо актуально для сервісів, пов’язаних із пошуком, обробкою маршрутів і взаємодією з користувачами. Простий інтерфейс, гнучка логіка налаштувань, підтримка різноманітних джерел даних і наявність вбудованих шаблонів значно спрощують процес створення додатку, знижуючи поріг входу навіть для користувачів без технічного бекграунду.

Одна з ключових переваг AppSheet полягає у його здатності підтримувати швидке прототипування: розробник одразу бачить результат змін у логіці чи інтерфейсі, що сприяє динамічному вдосконаленню продукту відповідно до зворотного зв’язку або змін у вимогах. Це повністю узгоджується з принципами гнучкої методології (Agile), що була раніше визнана найбільш доцільною для реалізації проєкту. Крім того, платформа пропонує потужні інструменти для автоматизації процесів, створення умовних дій, формул, логіки сповіщень, геолокаційного позиціювання та інших механізмів, що особливо важливо для додатку, орієнтованого на взаємодію з географічною інформацією та користувацькими запитами [8].

Також, варто підкреслити, що AppSheet дозволяє масштабувати рішення без істотного переписування логіки або повторного програмування, що робить цю платформу придатною як для локальних, так і для регіональних або навіть національних ініціатив. Важливо й те, що AppSheet має вбудовані інструменти для безпеки та керування доступом до даних, що дозволяє дотримуватися вимог конфіденційності, захисту персональної інформації та стабільності роботи додатку.

З огляду на усі зазначені характеристики, Google AppSheet виявляється найкращим вибором у контексті завдань цього дослідження. Поєднання простоти та гнучкості розгортання дає змогу реалізувати проєкт не лише ефективно й економічно, але й у відповідності до сучасних технологічних вимог.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОДАТКУ НА БАЗІ GOOGLE APPSHEET

2.1 Архітектура рішення

Побудова архітектури мобільного додатку на базі Google AppSheet розпочинається з формування логічної структури даних, яка лежить в основі всієї системи. У даному випадку додаток для пошуку попутників оперує кількома ключовими сутностями: користувачами, поїздками, маршрутами, заявками на участь у поїздці та повідомленнями. Ці сутності реалізовано у вигляді взаємопов’язаних таблиць, створених у Google Sheets – основному джерелі даних для AppSheet.

Таблиця користувачів містить поля з основною інформацією про кожного зареєстрованого учасника системи, зокрема ім’я, електронну пошту, роль (водій або пасажир), а також рейтинг, що формується на основі зворотного зв’язку. Таблиця поїздок включає інформацію про маршрут, час відправлення, кількість вільних місць, тип транспорту, контакт водія та умови поїздки. Важливим елементом структури є таблиця маршрутів, яка визначає пункти відправлення і призначення, а також дозволяє реалізувати функцію пошуку та фільтрації відповідно до географічних параметрів.

Між цими таблицями можна налаштувати логічні зв’язки типу «один до багатьох» або «багато до одного». Для прикладу, один користувач може створити багато поїздок, а кожна поїздка може мати багато заявок від пасажирів. Ці зв’язки реалізуються за допомогою AppSheet-функціоналу «Ref», який дозволяє зв’язати таблиці через унікальні ідентифікатори та автоматично підключати пов’язані записи в інтерфейсі.

Крім основної структури таблиць, у додатку реалізовано логіку фільтрації, відображення та взаємодії з даними. Наприклад, при створенні нової поїздки система автоматично призначає її автором поточного користувача, а також перевіряє, чи не заповнено всі необхідні поля.

Додатково впроваджено формули для розрахунку кількості доступних місць у поїздці в реальному часі, умовні правила для керування доступом до дій (наприклад, редагування або скасування поїздки доступне лише її автору), а також правила валідації введених даних. Вся логічна структура додатку побудована на принципах цілісності, взаємозв'язаності та автоматизації.

Наступним етапом у побудові архітектури буде планування інтеграції з зовнішніми сервісами, зокрема продуктами Google. Завдяки тому, що Google AppSheet побудований на інфраструктурі Google Cloud, розробка додатку дозволяє легко організувати взаємодію з такими сервісами, як Google Sheets, Google Maps та Google Firebase. Ми плануємо використати цю можливість для створення гнучкої, зручної та масштабованої платформи для пошуку попутників.

В першу чергу, джерелом даних у додатку виступатимуть таблиці Google Sheets. Вони використовуватимуться як база для зберігання інформації про поїздки, користувачів, маршрути та повідомлення. Завдяки прямому зв'язку AppSheet із Google Sheets, усі зміни в додатку автоматично синхронізуватимуться з таблицями в реальному часі. Це забезпечить прозорість структури, простоту резервного копіювання та можливість швидкого доступу до даних за межами AppSheet, якщо це буде потрібно.

Додатково планується інтеграція з Google Maps для реалізації зручного візуального представлення маршрутів. Це дозволить користувачам бачити графічне зображення початкового та кінцевого пункту подорожі, а також отримувати оцінку маршруту за відстанню. Водії зможуть зазначати координати точок відправлення та прибуття, які автоматично будуть відображатися на мапі, що значно покращить навігаційний досвід користувачів.

Ще один перспективний напрям – використання Google Firebase для реалізації сповіщень (Push Notifications) та підвищення рівня безпеки через систему автентифікації. У майбутньому планується поєднання AppSheet із Firebase Authentication через API-інтеграцію або зовнішні скрипти Google Apps Script, що дозволить здійснювати авторизацію через Google-акаунти або номер телефону, зберігаючи при цьому централізований контроль над сесіями користувачів.

Всі ці сервіси будуть взаємодіяти як окремі компоненти, що формують єдину інфраструктуру додатку. Їх інтеграція не вимагатиме створення складного коду завдяки no-code функціоналу AppSheet, але забезпечить високий рівень ефективності та функціональності. Діаграму класів додатку наведено на рис. 2.1:

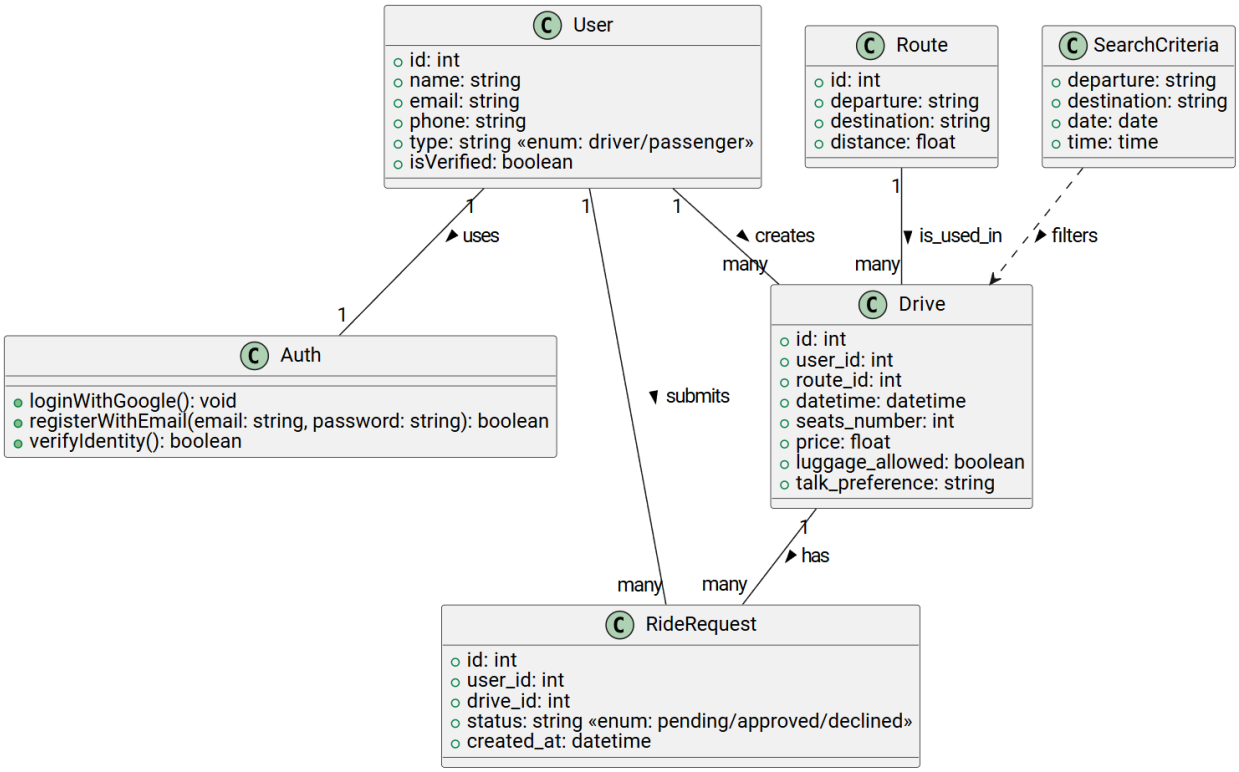

Рис. 2.1 – UML-діаграма класів проєкту

Ця діаграма класів відображає основні сутності додатку та їхні взаємозв’язки. Вона описує логічну структуру системи мобільного додатку для пошуку попутників, акцентуючи увагу на ключових об'єктах, їхніх атрибутах і зв'язках між ними.

Центральним елементом моделі виступає клас User, який репрезентує всіх користувачів системи – як водіїв, так і пасажирів. Користувачі мають базові персональні дані, включаючи ім’я, контактну інформацію та тип (водій або пасажир), статус-поле стосовно верифікації користувача у системі. Залежно від типу, вони можуть створювати оголошення про поїздки або залишати запити на бронювання.

Для забезпечення доступу до функціональності додатку передбачено клас Auth, який реалізує механізми реєстрації та авторизації. Це може бути або вхід через Google-акаунт, або реєстрація за допомогою електронної пошти з подальшим підтвердженням особи. Після успішної авторизації користувач отримує доступ до персонального кабінету.

Ключовим елементом логіки є клас Drive, який описує конкретну поїздку, створену водієм. Кожна поїздка пов’язана з конкретним маршрутом (Route) – структурою, що зберігає інформацію про пункт відправлення, пункт призначення та відстань. У межах поїздки можна задати дату й час, кількість вільних місць, вартість, а також додаткові умови (наявність багажу, уподобання щодо спілкування тощо). Пасажири можуть залишати запити на участь у поїздці, що відображено у вигляді класу RideRequest. Заявка включає статус (наприклад, очікує підтвердження, схвалено або відхилено), дату створення, а також посилання на поїздку та користувача, який її подав.

Для підтримки функції пошуку використовується допоміжний клас SearchCriteria, який описує критерії фільтрації поїздок за такими параметрами, як пункт відправлення, призначення, дата й час. Цей клас дозволяє пасажирам швидко знаходити релевантні оголошення та надсилати запити на участь.

Уся модель побудована так, щоб забезпечити просту, розширювану й логічно цілісну структуру додатку. Завдяки чітко визначеним зв'язкам між класами можна ефективно організувати взаємодію між водіями й пасажирами, реалізувати гнучкий механізм керування поїздками та забезпечити зручний користувацький досвід. У майбутньому до цієї моделі можуть бути додані додаткові компоненти, зокрема чат, рейтингова система, сповіщення та платіжна інтеграція, що дозволить створити повноцінну, конкурентоспроможну платформу. Такий підхід також сприяє легшому тестуванню та підтримці системи, оскільки логіка розділена на окремі, чітко визначені блоки. У разі потреби в розширенні функціоналу достатньо додати нові класи або модифікувати наявні без ризику порушення цілісності архітектури [32].

2.2 Інтерфейс користувача

Наступним кроком у розробці стає формування інтерфейсу користувача, який буде зрозумілим, адаптивним і зручним для різних категорій користувачів. Для реалізації цього етапу ми плануємо використати вбудований конструктор інтерфейсу, який надає платформа Google AppSheet. Цей інструмент дозволяє створювати повноцінний UX без потреби у програмуванні, забезпечуючи візуальне налаштування логіки відображення елементів, їхньої послідовності, стилю та поведінки.

Конструктор інтерфейсу в AppSheet працює на основі логіки, закладеної в структурах таблиць, зв’язках між даними та обраних шаблонах відображення. На практиці це виглядає як вибір типів представлення даних: наприклад, список поїздок можна оформити у вигляді таблиці, галереї, картки або мапи, залежно від зручності користувача. Для додатку з пошуку попутників планується використати такі базові UX-модулі: Deck View для відображення списків маршрутів і поїздок, Form View для створення або редагування поїздок, Detail View для перегляду конкретної поїздки, а також Map View для візуалізації маршрутів.

Шаблони, запропоновані AppSheet, дозволяють миттєво створювати типові екрани: головне меню, сторінку профілю користувача, форму подання заявки або форму пошуку маршруту. Ці шаблони надалі можна гнучко адаптувати – змінювати кольори, порядок елементів, тексти, значки, поведінку кнопок. Для прикладу, при відкритті додатку користувач одразу потраплятиме на стартову сторінку, де буде перелік доступних поїздок, відфільтрований за найближчими датами. Через меню внизу екрана можна буде перейти до профілю, створити власну поїздку або переглянути вже подані заявки. AppSheet також підтримує умовну логіку для відображення інтерфейсу, тому, наприклад, водії бачитимуть лише кнопки для створення поїздок, а пасажири – лише функціонал пошуку та подачі заявок. Це дає змогу реалізувати рольову модель використання без складної ручної розробки.

Окрему увагу буде приділено оптимізації інтерфейсу для мобільних пристроїв. AppSheet автоматично підлаштовує елементи під розмір екрана, однак додатково планується ручне налаштування деяких компонентів, щоб уникнути перевантаження або незручного перегортання сторінок. Наприклад, у формі створення нової поїздки поля будуть згруповані логічно: спочатку місце і час, потім кількість місць, а вже після цього – додаткові опції або коментарі.

Для ілюстрації процедур використання додатка користувачами різних ролей розроблено діаграму варіантів використання, яку наведено на наступній сторінці на рис. 2.2:

Рис. 2.2 – Діаграма варіантів використання

Ця діаграма демонструє основні точки входу для кожного типу користувача, їхні дії та послідовність взаємодій. Вона дозволяє структурувати майбутній інтерфейс з урахуванням ролей, забезпечуючи гнучкість та зрозумілий процес користування.

2.3 Моделювання логіки бізнес-процесів

Після завершення налаштування інтерфейсу користувача особливу увагу варто приділити моделюванню бізнес-процесів, які будуть реалізовані у додатку. В рамках платформи Google AppSheet передбачені широкі можливості для автоматизації та логічного управління поведінкою додатку, що дозволяє суттєво спростити користувацький досвід і забезпечити коректну роботу всіх функціональних блоків.

У майбутньому планується налаштувати умови, дії та автоматизації, що регулюватимуть різні етапи взаємодії користувачів із додатком. Зокрема, можна буде створювати правила, які автоматично активуються при певних подіях, наприклад, надсилання повідомлень про підтвердження участі, оновлення статусу поїздки чи відправлення сповіщень про нові заявки. Для цього в AppSheet використовується система «Workflow» і «Bots», які дозволяють створювати ланцюжки дій, залежно від умов, що визначаються користувачем або адміністратором.

Важливо також передбачити логіку валідації введених даних – перевірку коректності маршрутів, наявності вільних місць, відповідності ролей користувачів. Для цього будуть реалізовані умовні вирази, які унеможливлять виконання певних дій без виконання відповідних вимог. Наприклад, пасажир не зможе надіслати запит на участь у поїздці, якщо його профіль не підтверджено, або якщо на вибраний маршрут уже немає доступних місць.

Особливу увагу буде приділено синхронізації та оновленню даних у реальному часі – завдяки інтеграції з Google Sheets і Firebase, зміни, внесені одним користувачем, миттєво відобразяться у додатку інших користувачів. Це важливо для забезпечення актуальності інформації і уникнення конфліктів, наприклад, при одночасному бронюванні місць. Крім того, AppSheet дозволяє налаштувати механізми автоматичного оновлення даних та періодичну синхронізацію у фоновому режимі, що додатково підвищує стабільність і швидкість взаємодії.

Для кращої візуалізації структури та послідовності бізнес-процесів передбачено створення Діаграми діяльності (Activity Diagram), яка відображатиме покрокову логіку виконання основних процесів у додатку – від реєстрації користувача до підтвердження поїздки. Цю діаграму наведено на рис. 2.3:

Рис. 2.3 – UML-діаграма діяльності проєкту

Ця діаграма демонструє ключові етапи логіки взаємодії користувачів із додатком, ілюструючи, які умови впливають на перебіг процесу, та які дії будуть автоматично виконуватись системою. Завдяки такому підходу можна не лише оптимізувати роботу додатку, а й забезпечити чітку структуру бізнес-логіки надійного та зручного сервісу пошуку попутників.

Після визначення бізнес-логіки та налаштування основних процесів важливо зосередитись на механізмах обробки подій, які безпосередньо впливають на взаємодію користувача з додатком і роблять роботу сервісу більш інтерактивною та оперативною. У Google AppSheet ця функціональність реалізується за допомогою тригерів, push-сповіщень і фільтрації даних, що дозволяє автоматично реагувати на зміни в системі та підтримувати користувачів у курсі важливої інформації.

У майбутньому планується налаштувати тригери, які будуть запускати певні дії при виникненні визначених подій, таких як створення нового запису, оновлення існуючого маршруту або зміна статусу поїздки. Наприклад, при додаванні нового користувача або запиту на участь у поїздці система автоматично ініціюватиме перевірку даних і сповіщатиме відповідальних осіб.

Ключовим елементом інтерактивності стануть push-сповіщення, що інформуватимуть користувачів про важливі події: підтвердження участі, відмова водія, зміни у розкладі або наявність нових маршрутів, які відповідають їхнім уподобанням. Це дозволить забезпечити швидкий зворотній зв’язок і покращити якість користувацького досвіду. Push-сповіщення будуть адаптовані до типу користувача – водій або пасажир – щоб надавати лише релевантну інформацію. Крім того, користувач зможе самостійно налаштовувати пріоритети сповіщень.

Окрім цього, буде реалізовано фільтрацію даних, що дасть змогу користувачам ефективно сортувати і відбирати маршрути за різними критеріями – наприклад, за датою, відстанню, наявністю вільних місць або рейтингом водія. Передбачено також можливість збереження комбінацій фільтрів, щоб користувачі могли швидко повертатися до найбільш актуальних для них параметрів. Додатково, фільтрація працюватиме в реальному часі, автоматично оновлюючи результати при зміні умов пошуку, що покращить динамічність взаємодії з додатком.

Для наочності та кращого розуміння роботи системи обробки подій буде створена Діаграма бізнес-процесів, яка демонструє, як тригери взаємодіють із різними модулями додатку, як відбувається ініціація push-сповіщень і яким чином працює фільтрація інформації. Таку діаграму зображено на рис. 2.4.

Рис. 2.4 – UML-діаграма бізнес-процесів проєкту

Діаграма бізнес-процесів чітко ілюструє послідовність дій, які виконуються після виникнення певної події у додатку. Вона демонструє, як тригери відслідковують зміни даних і активують відповідні процеси, що дозволяє автоматизувати ключові функції сервісу – від оновлення інформації до інформування користувачів через push-сповіщення. Завдяки ефективній фільтрації даних користувачі отримують швидкий доступ до релевантних маршрутів, що підвищує зручність та швидкість пошуку попутників. Окрім цього, діаграма допомагає виявити потенційні вузькі місця в логіці процесів і спростити майбутню оптимізацію додатку. Такий підхід дозволяє не лише автоматизувати рутинні операції, а й забезпечити масштабованість системи без втрати продуктивності.

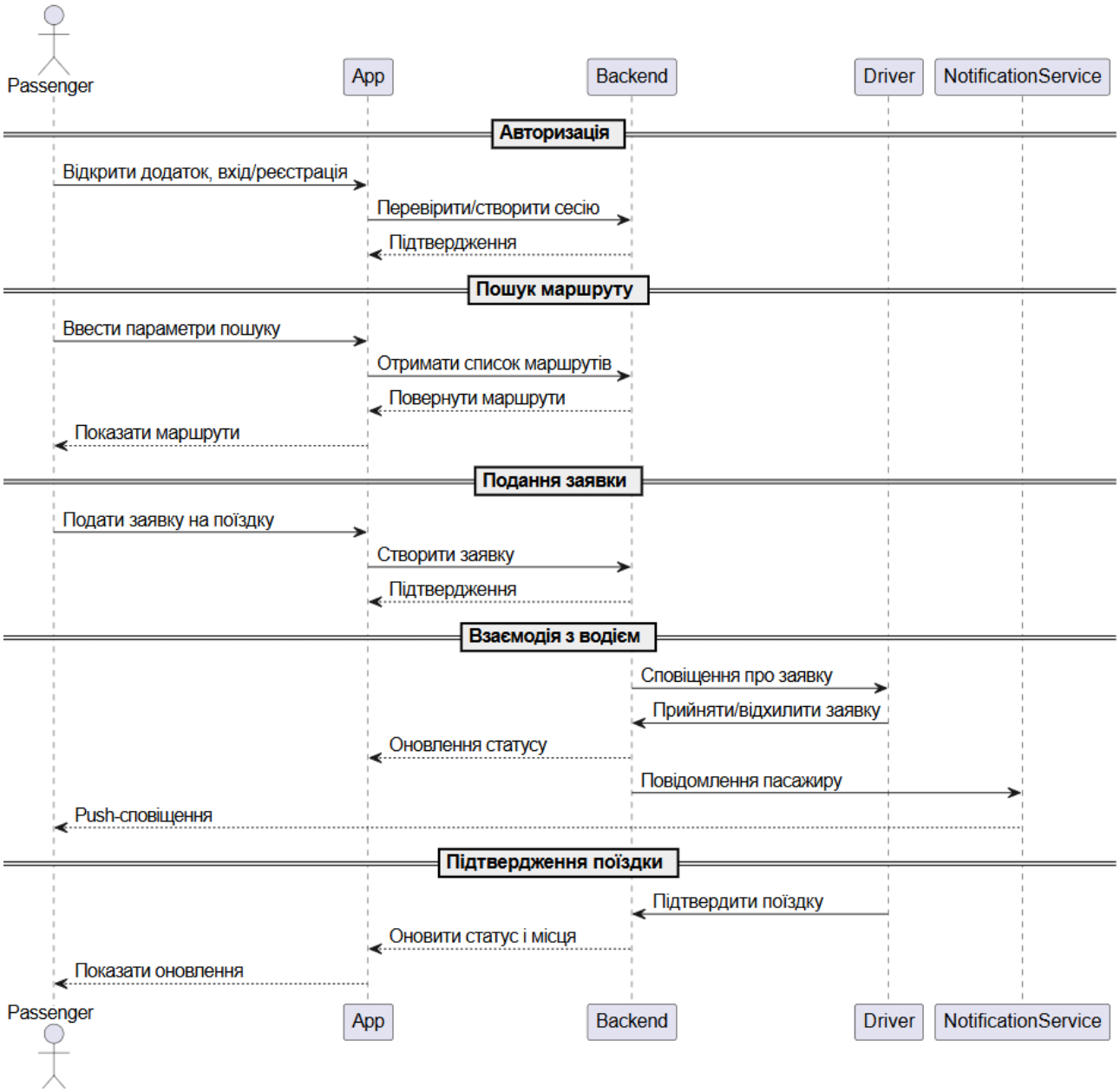

Рис. 2.5 – UML-діаграма послідовностей процесів проєкту

У даній послідовності основна увага зосереджена на ключових сценаріях взаємодії пасажира і водія з системою, що забезпечує пошук попутників і організацію поїздок.

Спершу користувач, у ролі пасажира, запускає мобільний додаток і здійснює авторизацію або реєстрацію. Цей етап є фундаментальним для безпечного доступу до системи, адже без підтвердження особи подальша робота недоступна. Запит на авторизацію спрямовується від клієнта (додатку) до бекенд-сервера, який відповідає за перевірку наявності користувача у базі.

Наступним кроком пасажир використовує інтерфейс додатку для пошуку маршруту. Він вводить необхідні параметри: пункт відправлення, пункт призначення, дату та час поїздки. Ці дані надсилаються до бекенду, який проводить пошук у базі даних активних маршрутів, фільтрує їх за заданими критеріями та формує відфільтрований список, що надсилається назад у додаток. Для користувача відображається перелік доступних поїздок з основною інформацією: маршрутом, датою, кількістю вільних місць, ціною та контактами водія. Такий підхід забезпечує швидкий і зручний пошук, що є ключовим для користувацького досвіду.

Коли пасажир обирає відповідний варіант, він подає заявку на участь у поїздці через додаток. Система створює відповідний запис у базі даних і відправляє підтвердження користувачу. Ця заявка надалі надсилається водієві, який є організатором поїздки, у вигляді сповіщення. Водій має можливість переглянути профіль пасажира, оцінити деталі заявки і прийняти рішення про підтвердження або відхилення.

У разі прийняття заявки, бекенд оновлює статус поїздки, додає пасажира до списку учасників і коригує кількість вільних місць. Одночасно система ініціює відправку push-сповіщення пасажиру через сервіс сповіщень, щоб він був проінформований про успішне бронювання. Це підвищує рівень комунікації та дозволяє уникнути непорозумінь.

Підтвердження поїздки з боку водія є заключним кроком у формуванні остаточного статусу бронювання. Після цього оновлена інформація одразу ж відображається в додатку пасажира – це забезпечує прозорість і актуальність даних у режимі реального часу. Крім того, можливе автоматичне генерування сповіщень для всіх учасників поїздки щодо деталей маршруту, часу виїзду або змін у планах.

Архітектурно ця послідовність демонструє чітке розмежування відповідальностей між клієнтською частиною (інтерфейсом користувача), серверною логікою (бекендом) та сервісом сповіщень. Такий підхід полегшує масштабування системи, дозволяє швидко адаптуватися до змін у вимогах та впроваджувати нові функції без руйнації існуючої логіки. Зокрема, використання

окремого компоненту для обробки повідомлень дозволяє розвантажити основний сервер та забезпечити стабільну роботу навіть при великій кількості одночасних користувачів.

Останнім кроком в процесі розробки стане впровадження механізмів контролю доступу та управління правами користувачів. Це необхідно для того, щоб кожен користувач мав доступ лише до тих функцій і даних, які відповідають його ролі та рівню повноважень, що значно підвищує безпеку системи і захищає персональну інформацію. Важливою складовою такого підходу є чітке розмежування обов'язків, що допомагає мінімізувати потенційні ризики несанкціонованих дій або витоку даних. З технічної точки зору, система контролю доступу має бути гнучкою і масштабованою, аби в майбутньому можна було легко додавати нові ролі або змінювати права без необхідності кардинального переписування коду.

У рамках платформи Google AppSheet планується налаштувати різні рівні доступу, які будуть контролювати, які дані користувач може переглядати, редагувати чи додавати. Для цього використовуватимуться вбудовані механізми автентифікації, інтеграція з Google-акаунтами, а також налаштування ролей безпосередньо у конфігурації додатку. Архітектурно це означає, що логіка авторизації буде частково винесена на рівень платформи, що суттєво спрощує реалізацію і підтримку системи. Водночас, додаток повинен мати механізми валідації запитів і на серверному рівні, щоб запобігти обхідній маніпуляції правами через клієнтський інтерфейс.

Наприклад, зареєстровані водії отримають розширені права для створення і редагування маршрутів, тоді як пасажири матимуть можливість лише переглядати доступні поїздки та подавати заявки на участь. Такий поділ функціоналу потребує чіткої реалізації ролей у базі даних та логіці бізнес-процесів, де кожна дія перевірятиметься на відповідність дозволам користувача [23]. Це також накладає вимоги до дизайну інтерфейсу: елементи, які не доступні для певної ролі, мають бути приховані або недоступні для взаємодії, що знижує ймовірність помилок і підвищує комфорт користування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ФУНКЦІОНАЛУ

3.1 Розробка ключових функцій додатку

У процесі розробки мобільного додатку для пошуку попутників однією з ключових функціональних складових стане реалізація системи фільтрації маршрутів. Це необхідно для того, щоб кожен користувач міг швидко й ефективно знайти релевантні поїздки відповідно до своїх потреб – наприклад, за пунктом призначення, датою, часом відправлення або кількістю вільних місць.

Фільтрація маршрутів в AppSheet реалізовується через формули, що динамічно обробляють дані з пов’язаної таблиці (наприклад, Google Sheets). Але в разі гіпотетичного використання JavaScript для реалізації тієї ж логіки, яка застосовується при фільтрації, можна було б уявити приблизний алгоритм наступним чином.

Припустимо, у нас є масив об’єктів, кожен із яких представляє окремий маршрут. Кожен маршрут містить інформацію про відправну точку, пункт призначення, дату, час, кількість доступних місць та інші супровідні дані. Завданням фільтрації є відібрати лише ті маршрути, які відповідають критеріям, заданим користувачем через інтерфейс (наприклад, за допомогою форм або випадаючих списків) (див. лістинг 3.1).

```
// Приклад: масив маршрутів
const routes = [
    {
      from: 'Львів',
      to: 'Київ',
      date: '2025-06-15',
      time: '09:00',
      seatsAvailable: 3
    },
    {
      from: 'Львів',
      to: 'Одеса',
      date: '2025-06-15',
      time: '11:00',
```

```
    seatsAvailable: 0
  },
  {
    from: 'Тернопіль',
    to: 'Київ',
    date: '2025-06-16',
    time: '08:30',
    seatsAvailable: 2
  }
];

// Вхідні параметри від користувача
const filter = {
  from: 'Львів',
  to: 'Київ',
  date: '2025-06-15'
};

// Функція фільтрації
const filteredRoutes = routes.filter(route => {
  return (
    route.from === filter.from &&
    route.to === filter.to &&
    route.date === filter.date &&
    route.seatsAvailable > 0
  );
});

console.log(filteredRoutes);
```

Лістинг 3.1 – Функція фільтрації маршрутів

У цьому прикладі система виконує фільтрацію за трьома параметрами: місце відправлення, місце призначення та дата. Додатково застосовується перевірка на наявність вільних місць, аби не пропонувати користувачеві поїздки, які вже недоступні. Цю логіку легко адаптувати для AppSheet за допомогою виразів типу SELECT(...), FILTER(...) чи LOOKUP(...), де кожен з параметрів підставляється в залежності від введених користувачем значень.

В AppSheet подібна фільтрація може бути реалізована у вигляді віртуальної колонки, яка повертає відфільтрований список на основі динамічних умов. Для прикладу, поле типу EnumList з динамічними значеннями буде оновлюватися автоматично після кожної зміни критеріїв. Це дозволяє вбудувати інтелектуальну систему підбору маршрутів без використання стороннього коду, серверної логіки.

Наступний крок – створення та налаштування системи профілів користувачів, яка дозволить чітко розділити ролі водіїв і пасажирів, зберігати важливу інформацію про кожного учасника сервісу, а також забезпечити персоналізацію та базову довіру в межах платформи.

У рамках мобільного додатку, розробленого на платформі AppSheet, ми плануємо реалізувати структуру, у якій кожен користувач після реєстрації зможе мати власний профіль. У цьому профілі зберігатимуться як базові дані (ПІБ, контактний номер, електронна пошта), так і додаткові відомості, залежно від ролі. Якщо користувач обирає роль водія, у його профілі буде відображено інформацію про наявні транспортні засоби, їхні характеристики (модель, кількість місць, тип пального), середню оцінку за поїздки, а також кількість завершених маршрутів. Для пасажира передбачено відображення історії поїздок, оцінок, отриманих від водіїв, та статусу підтвердження профілю (наприклад, верифікація за телефоном або електронною поштою).

Розподіл на ролі буде здійснюватися через окреме поле в таблиці користувачів, де роль вибирається при першому вході або змінюється згодом через параметри профілю. Для реалізації цього функціоналу в AppSheet буде створено окрему таблицю Users, яка пов’язується з таблицями Drivers і Passengers. Ця структура дозволяє використовувати умовну логіку (наприклад, показувати різні форми і доступ до різних функцій залежно від ролі користувача), що суттєво підвищує гнучкість застосунку.

Для демонстрації логіки на прикладі коду JavaScript можна побачити процес створення профілю наступним чином на лістингу 3.2:

```
function createUserProfile(data) {
    const profile = {
      id: generateUserId(),
      fullName: data.fullName,
      email: data.email,
      phone: data.phone,
      role: data.role, // 'driver' або 'passenger'
      createdAt: new Date().toISOString()
    };
```

```
  if (data.role === 'driver') {
    profile.car = {
      model: data.carModel,
      seats: data.carSeats,
      licensePlate: data.licensePlate
    };
    profile.rating = 5.0;
    profile.tripsCompleted = 0;
  }

  if (data.role === 'passenger') {
    profile.tripsCompleted = 0;
    profile.rating = 5.0;
  }

  saveToDatabase(profile); // умовна функція збереження
}
```

Лістинг 3.2 – Функція створення профілю користувача

У середовищі AppSheet така логіка реалізується безпосередньо через динамічні форми введення, де поля з’являються лише тоді, коли користувач обирає відповідну роль. Наприклад, якщо користувач позначив себе як водій, система автоматично розгортає додаткові поля для введення інформації про транспорт.

Після впровадження системи профілів для водіїв і пасажирів важливим функціональним розширенням стане розробка рейтингової системи та механізму зворотного зв’язку. Це не лише логічне продовження персоналізації, а й критично важливий елемент довіри між користувачами платформи, що особливо актуально для сервісів, побудованих на моделі спільних поїздок.

У майбутньому ми плануємо реалізувати систему, яка дозволить кожному користувачу залишати оцінки та текстові відгуки після завершення поїздки. Ці оцінки будуть впливати на середній рейтинг профілю – як водія, так і пасажира. Після завершення маршруту в додатку з’являтиметься форма, що пропонує оцінити поїздку за п’ятибальною шкалою, а також додати короткий коментар. Водночас система не дозволятиме залишати відгуки, якщо маршрут не було підтверджено як завершений, що виключає випадки фальсифікацій.

Для збереження зворотного зв’язку буде створено окрему таблицю, умовно назвемо її Ratings, що міститиме такі поля, як ID поїздки, ID користувача, ID того,

кого оцінюють, оцінка, текст відгуку, а також дата створення. За допомогою відносин між таблицями Trips, Users та Ratings, ми зможемо відображати агреговану інформацію у вигляді середнього балу в профілі кожного учасника. Наприклад, у профілі водія буде видно середню оцінку на основі всіх завершених маршрутів, а також останні коментарі пасажирів.

Код, що демонструє обробку зворотного зв'язку, наведено на лістингу 3.3:

```
function submitRating(tripId, reviewerId, targetUserId, score, comment) {
    const rating = {
      id: generateUniqueId(),
      tripId: tripId,
      reviewerId: reviewerId,
      targetUserId: targetUserId,
      score: score,
      comment: comment,
      createdAt: new Date().toISOString()
    };

    saveToDatabase(rating); // умовне збереження в базу

    // Після збереження — оновлення середнього рейтингу
    updateUserAverageRating(targetUserId);
  }
```

Лістинг 3.3 – Функція надсилання відгуку до поїздки

Функція updateUserAverageRating викликатиме підрахунок усіх оцінок для певного користувача та оновлюватиме значення середнього рейтингу у його профілі. У середовищі AppSheet це реалізується автоматично за допомогою формул типу AVERAGE(SELECT(Ratings[score], [targetUserId] = [_THISROW].[UserId])).

Також, передбачається механізм модерації – додаток не відображатиме образливі або порожні коментарі, що не несуть інформаційної цінності. Для цього ми плануємо впровадити мінімальний обсяг символів у полі коментаря або перевірку на вміст заборонених слів за допомогою правил перевірки даних.

У результаті система рейтингів стане не лише важливим інструментом соціального підтвердження, а й дієвим способом формування відповідальної спільноти. Завдяки ній користувачі зможуть орієнтуватися на досвід інших.

3.2 Перевірка працездатності

Після реалізації ключових функцій додатку – таких як фільтрація маршрутів, система профілів, рейтингування та зворотний зв’язок – необхідно буде провести перевірку працездатності проєкту. Першим кроком у цьому напрямку є теоретичне моделювання сценаріїв використання, що дозволяє оцінити передбачену логіку взаємодії між користувачем та додатком без залучення реальних даних або зовнішніх тестувальників. Такий підхід особливо корисний у початковій фазі валідації рішення, коли воно ще не впроваджене в реальному середовищі.

Ми плануємо побудувати низку типових ситуацій, які імітують реальне користування додатком. Наприклад, одним із базових сценаріїв стане процес реєстрації нового користувача: ми уявляємо, як користувач відкриває додаток, вводить свої дані, проходить перевірку електронної пошти та створює профіль. Далі моделюється ситуація, в якій цей користувач, як пасажир, вводить потрібний маршрут – наприклад, зі Львова до Тернополя на визначену дату. Система, згідно з логікою фільтрації, відображає всі активні поїздки з подібними параметрами. Користувач переглядає деталі профілю водія, бачить рейтинг та відгуки, а після цього надсилає запит на поїздку.

В іншому сценарії розглядається взаємодія з боку водія: він створює маршрут, додає кількість вільних місць, час виїзду та вартість. Потім до нього надходить запит від пасажира. Водій переглядає профіль потенційного попутника і має змогу прийняти або відхилити запит. Після підтвердження система автоматично формує статус поїздки, додає пасажира до списку, оновлює кількість доступних місць і запускає відповідні сповіщення. Водночас, ця послідовність дій дає змогу підтримувати актуальний стан поїздок у реальному часі та забезпечує гнучке управління ресурсами платформи.

Також, моделюємо завершення маршруту. Після того як обидва учасники підтверджують завершення поїздки, з’являється можливість залишити оцінку та коментар. Це активує збереження даних у таблиці Ratings.

Крім того, розглядаються й виняткові ситуації: наприклад, що відбувається, якщо водій скасував поїздку за 2 години до виїзду, або коли пасажир не з’явився. У таких випадках перевіряється, чи коректно змінюється статус поїздки, чи оновлюються відповідні поля, чи надсилаються автоматичні повідомлення іншим сторонам. Далі, постає потреба в системному аналізі потенційних помилок, що можуть виникати під час виконання бізнес-логіки, обробки даних або взаємодії між модулями. Завдання цього етапу полягає не лише в ідентифікації ймовірних точок збою, а й у розробці стратегій, які унеможливлять або мінімізують вплив таких ситуацій на користувацький досвід і загальну стабільність роботи системи.

Аналіз починається з перегляду кожного етапу взаємодії користувача з додатком. Зокрема, особливу увагу буде приділено полям, обов’язковим для заповнення під час створення маршруту, реєстрації або підтвердження участі у поїздці. У разі, якщо користувач пропустить поле або введе некоректні дані, система має автоматично виводити повідомлення про помилку у зрозумілому форматі. Для цього передбачається застосування валідаційних механізмів на рівні таблиць AppSheet, де кожне поле буде супроводжене умовами допустимих значень.

Інший критичний аспект – запобігання логічним конфліктам, наприклад, коли один і той самий пасажир намагається забронювати місце в кількох поїздках, що відбуваються одночасно. Такі ситуації планується контролювати за допомогою виразів типу FILTER() та COUNTIF() у формульній логіці AppSheet, які перевірятимуть наявність активних бронювань у поточному часовому проміжку.

Ще одним поширеним джерелом помилок є взаємодія з зовнішніми сервісами – такими як Google Maps для побудови маршрутів або Firebase для зберігання зображень профілів. Якщо доступ до сервісу буде тимчасово втрачено або перевищено ліміт запитів, у додатку передбачатиметься резервна логіка: наприклад, сповіщення про технічні труднощі з відповідною рекомендацією повторити дію пізніше або звернутися до служби підтримки.

Тестування додатків у AppSheet дуже зручне і доступне навіть без складних налаштувань. Після створення або редагування додатку на порталі AppSheet користувач отримує можливість миттєво переглянути прев’ю своєї роботи. Це

прев'ю автоматично адаптується під різні типи пристроїв – смартфони, планшети, десктопи – що дозволяє оцінити, як інтерфейс виглядатиме і працюватиме на реальних гаджетах користувачів. Такий підхід суттєво полегшує перевірку дизайну, взаємодії та логіки додатку без потреби в окремому емуляторі чи фізичному пристрої.

Крім того, портал надає різноманітні опції для тестування функціональності: можна змінювати вхідні дані, імітувати дії користувача, перевіряти реакцію тригерів, логіку фільтрів і правила доступу. AppSheet також дозволяє переглядати журнал змін і помилок, що допомагає швидко виявити і виправити недоліки ще на ранніх етапах розробки. Приклади тестування додатку з використанням інструментів у AppSheets наведено на рисунках 3.1-3.4:

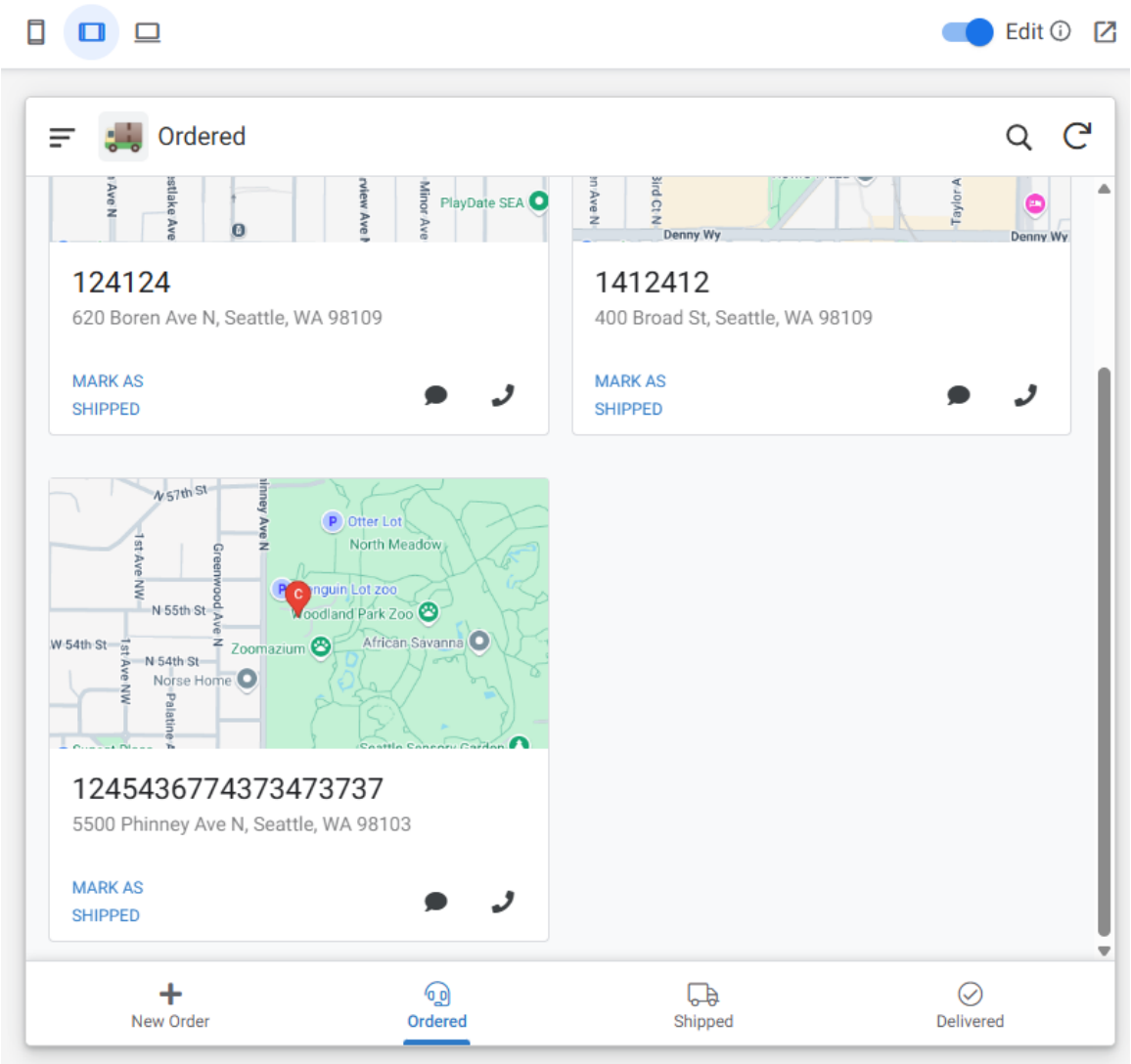

Рис. 3.1 – Тестування замовлень проєкту у AppSheets у форматі планшета

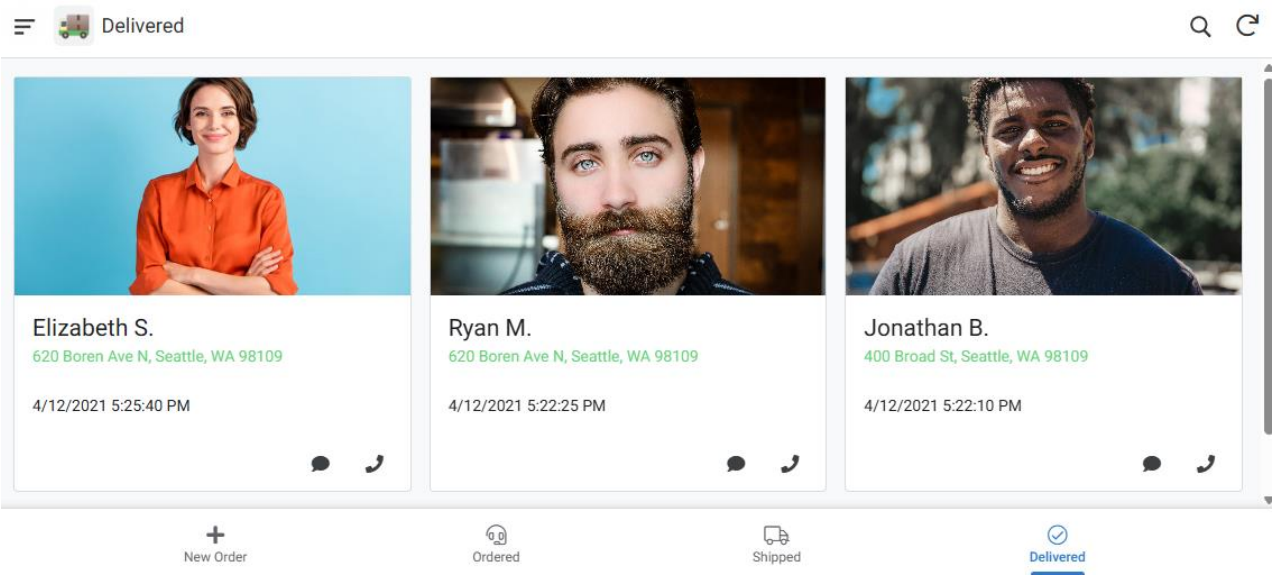

Рис. 3.2 – Тестування закінчених поїздок проєкту у форматі десктопу

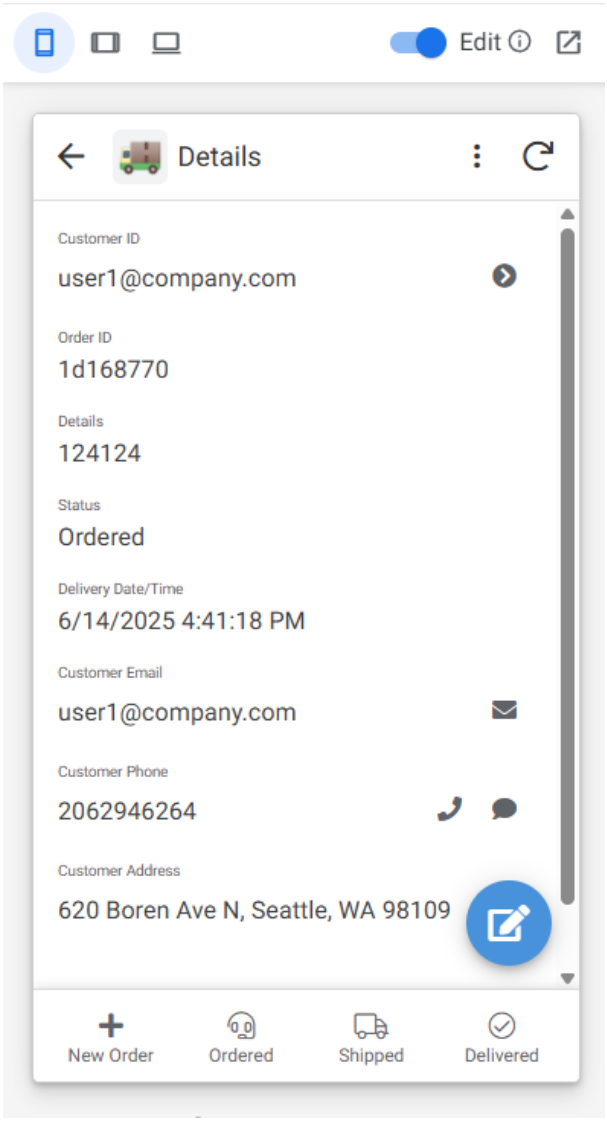

Рис. 3.3 – Тестування сторінки деталей поїздки проєкту у мобільному форматі

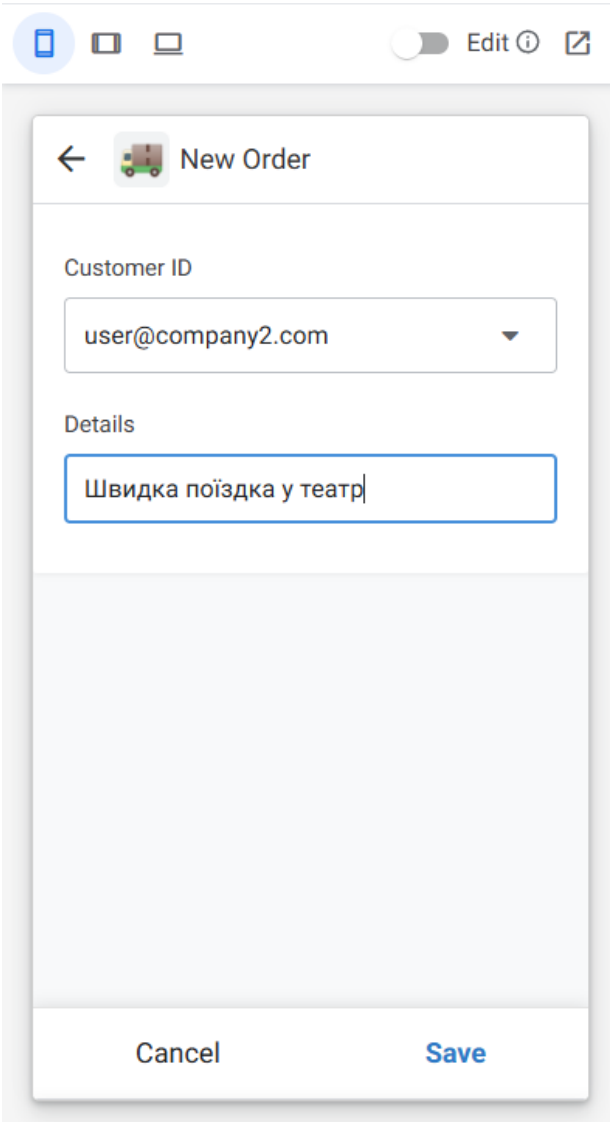

Рис. 3.4 – Тестування модального вікна створення поїздки проєкту у мобільному форматі

Також, враховується ймовірність внутрішніх помилок у роботі автоматизацій – наприклад, коли не спрацьовує надсилання push-сповіщення. У таких випадках буде реалізовано журнал подій, в якому фіксуватиметься статус виконання кожної дії. Завдяки цьому адміністратор або розробник зможе швидко виявити причину збою та виправити її без потреби повного перезапуску додатку.

Особливу увагу буде приділено некоректним ролям і доступам: якщо користувач якимось чином отримав доступ до функцій, які не відповідають його ролі (наприклад, пасажир – до створення маршрутів), система має заблокувати виконання дії із повідомленням про недостатні права.

Щоб вчасно виявити недопрацювання у логіці, помилки при обробці даних або некоректну поведінку елементів інтерфейсу, доцільно використати вбудовані інструменти платформи AppSheet, які дозволяють здійснювати перевірку застосунку без необхідності публікації чи розгортання фінальної версії.

Основним інструментом, який буде використовуватися для внутрішнього тестування, є Editor Preview – інтерактивне середовище в самому редакторі AppSheet, де розробник може симулювати дії користувача: перегляд списків маршрутів, додавання нових записів, виконання фільтрації, бронювання поїздок, залишення відгуків тощо. У режимі попереднього перегляду є можливість швидко перемикатися між типами пристроїв (смартфон, планшет, веббраузер), що дозволяє перевірити адаптивність інтерфейсу та логіку навігації між екранами.

Окрему увагу буде приділено перевірці умов видимості та доступності елементів. Для прикладу, якщо певне поле має бути доступним лише для користувачів із роллю «водій», або кнопка підтвердження поїздки повинна з’являтися лише після заповнення всіх необхідних полів — усе це тестується безпосередньо у режимі Preview, з використанням функцій перемикання ролей та тестових облікових записів. AppSheet дозволяє вказати конкретну email-адресу, під якою виконується тестування, щоб відстежити доступ і реакцію системи на дії користувача з відповідним рівнем прав.

Також, буде використано Test Formulas – спеціальний механізм, який дозволяє перевірити правильність функціональних виразів (наприклад, IF(), FILTER(), LOOKUP()) ще до їх інтеграції у логіку додатку. Таким чином, можна переконатися, що логіка відображення кнопок, запуску дій або відправлення повідомлень працює як очікується, без потреби створювати повноцінні сценарії вручну.

Ще один корисний інструмент – Audit History, який забезпечує журнал усіх змін і подій, що відбулися в додатку під час тестування. Цей журнал дозволяє простежити, яка дія була ініційована, ким саме, і які дані при цьому були змінені. Завдяки цьому можна оперативно виявити логічні помилки або некоректні записи, перевірити, чи правильно працюють обмеження доступу та автоматизовані дії.

На завершення тестування планується провести імітацію типових сценаріїв використання (бронювання маршруту, оцінка водія, фільтрація результатів, тощо) за участі кількох тестових акаунтів з різними ролями. Це дозволить зібрати первинну статистику щодо взаємодії користувачів із додатком, перевірити поведінку в умовах навантаження.

3.3 Порівняльний аналіз AppSheet та альтернативних інструментів

В процесі розробки мобільного додатку надзвичайно важливим було ретельно підібрати інструмент, який би забезпечив баланс між швидкістю розробки, можливостями інтеграції з популярними сервісами, а також гнучкістю інтерфейсу й масштабованістю проєкту [4]. Після вивчення низки no-code/low-code платформ, для практичного застосування було обрано, неодноразово згаданий, Google AppSheet. Щоб обгрунтувати цей вибір, доречно провести реальне порівняння з одним із найбільш популярних конкурентів – Microsoft PowerApps.

Крім того, варто звернути увагу на специфіку ліцензування та цінову політику обох платформ, що може істотно впливати на загальний бюджет проєкту, особливо на початкових етапах. Google AppSheet пропонує гнучкі тарифні плани, які дозволяють починати з мінімальними витратами та масштабувати рішення у міру зростання потреб, що є важливим для стартапів і невеликих команд. У той же час, Microsoft PowerApps, хоч і має широку корпоративну підтримку та інтеграцію з продуктами Microsoft 365, часто потребує більш складної налаштувальної роботи і може виявитися дорожчим для малого бізнесу. Google AppSheet в свою чергу відзначається інтуїтивно зрозумілим інтерфейсом і невеликим порогом входження.

Порівняння охоплює два ключові аспекти, важливі для реалізації поставленого проєкту: простота інтеграції з зовнішніми сервісами та підтримка різних платформ і пристроїв (див. табл. 3.1).

Таблиця 3.1 – Порівн. аналіз Google AppSheet та Microsoft PowerApps

Критерій	Google AppSheet	Microsoft PowerApps
Інтеграція з таблицями даних	Пряма й нативна інтеграція з Google Sheets, швидке підключення через Google Drive. Налаштування займає кілька хвилин.	Інтеграція з Excel Online (через OneDrive) або SQL-таблицями. Налаштування складніше, особливо поза екосистемою Microsoft.
Інтеграція з картографією	Google Maps інтегрується безпосередньо, доступ до геолокацій та маршрутів реалізується вбудованими елементами.	Використовується Bing Maps, можливе підключення до Google Maps, але через API з додатковим кодуванням.
Push-сповіщення	Нативна підтримка Email- і SMS-сповіщень, інтеграція з Firebase Cloud Messaging потребує зовнішніх сценаріїв.	Краще реалізовано через Power Automate, проте потребує окремої передплати.
Інтерфейс налаштувань	Інтуїтивний інтерфейс, UX-модулі легко перетягуються, можливість швидкої адаптації форм та переглядів.	Інтерфейс більше орієнтований на корпоративних користувачів, складніший у навігації для початківців.
Підтримка платформ	Працює на Android, iOS та у веббраузері без встановлення окремих додатків. Автоматична адаптація до екранів.	Потребує встановлення PowerApps Player, є залежність від Microsoft 365. Мобільна версія працює стабільно, але обмежено.
Швидкість запуску MVP	Можна зібрати базовий додаток за 1–2 дні, всі зміни миттєво відображаються.	Старт вимагає більше конфігурацій, налаштувань прав і підключень.
Вартість	Безкоштовна розробка в базовій версії. Оплата залежить від кількості активних користувачів.	Входить у пакет Microsoft 365, але для повного функціоналу потрібна платна ліцензія.
Спільнота та підтримка	Активна спільнота Google AppSheet Community, велика кількість шаблонів і гайдів.	Потужна підтримка серед розробників Microsoft, але більше орієнтована на бізнес-рішення.

На основі наведеного аналізу можна зробити висновок, що AppSheet є більш гнучким і простим у використанні рішенням для невеликих і середніх проєктів, особливо в контексті використання Google-екосистеми, яка вже застосовується в більшості навчальних закладів, малих підприємств і команд. Простота інтеграції з Google Sheets, Maps і Drive дозволяє швидко розгорнути MVP та протестувати ключові функції додатку без глибоких знань програмування.

У той час як PowerApps надає потужний арсенал засобів для інтеграції в корпоративну екосистему Microsoft, цей підхід виправданий лише за умови вже існуючої інфраструктури Microsoft 365 і наявності фахівців, які мають досвід з Power Platform.

Але, для повної картини важливо розглянути ще одну групу конкурентів – це візуальні інструменти з акцентом на гнучкий дизайн та розширені можливості кастомізації, зокрема Adalo та FlutterFlow. Обидва ці сервіси активно застосовуються для створення мобільних застосунків із більш складною логікою та гнучкістю UI, і вони суттєво відрізняються від AppSheet саме за глибиною кастомізації (див. табл. 3.2).

Таблиця 3.2 – Порівняльний аналіз AppSheet, Adalo та FlutterFlow за рівнем кастомізації

Критерій	Google AppSheet	Adalo	FlutterFlow
Кастомізація інтерфейсу	Обмежена готовими шаблонами та стандартними модулями: зміна кольору, іконок.	Повна свобода дизайну: позиціонування компонентів вручну, побудова кастомних UI-екранів.	Найбільш детальна кастомізація: пряме налаштування layout, шрифтів, анімацій, responsive-дизайну.

Продовження таблиці 3.2

Налаштування стилів	Мінімальні. Користувач не може змінити CSS або додати власні стилі. Вибір тем обмежений.	Можливість редагування кольорів, шрифтів, відступів, тіней, форм.	Повна підтримка кастомного дизайну через параметри стилів, з можливістю CSS-імітації.
Анімація та переходи	Відсутні. Зміна між переглядами – миттєва, без візуального ефекту.	Простий набір анімацій для переходів та елементів (fade, slide).	Складні анімації, побудова взаємодій між елементами, таймери, кастомні транзишени.
Кастомна логіка	Обмежена логіка у вигляді умов, правил відображення, дій (без коду). Розширення через Webhook/API.	Сценарії користувача (Actions, Triggers), але без складного коду.	Підтримка кастомного коду на Dart. Можна реалізовувати повноцінні алгоритми.
Гнучкість навігації	Готові сценарії: перегляд/додавання/редагування. Налаштування навігації через меню та типи подань.	Можна створювати будь-яку структуру навігації вручну.	Повна свобода в архітектурі навігації, включаючи стекові та вкладені переходи.
Підключення зовнішніх компонентів	Обмежене. Без можливості вставки кастомних компонентів.	Можна створювати компоненти вручну або з базовими діями.	Дозволяє створення або імпорт кастомних віджетів та підключення сторонніх бібліотек.

Як видно з аналізу, AppSheet значно програє як Adalo, так і FlutterFlow у гнучкості кастомізації, особливо коли мова йде про інтерфейс та індивідуальну поведінку компонентів. Adalo є добрим вибором для дизайнерів, які прагнуть повного контролю над виглядом додатку, зберігаючи no-code підхід. FlutterFlow, у свою чергу, виходить за межі no-code і стає інструментом для low-code розробки з підтримкою Dart-коду, що робить його придатним для високофункціональних застосунків з індивідуальним стилем та логікою.

У випадку з AppSheet, розробник працює в межах заданої системи: логіка додається у вигляді умов і дій, а інтерфейс – на основі шаблонів із фіксованим дизайном. Це істотно спрощує процес, але водночас обмежує можливості реалізації унікального візуального досвіду або нестандартної логіки.

Проте для завдань, де пріоритетом є швидкість реалізації, інтеграція з Google-середовищем і формування простого, функціонального інтерфейсу без потреби у складному дизайні, AppSheet залишається цілком раціональним вибором.

В контексті освітніх проєктів, AppSheet особливо зручний для викладачів, студентів і учасників хакатонів, які не мають глибоких технічних знань, але мають потребу швидко створити функціональний прототип додатку для демонстрації ідеї, управління даними або автоматизації певних процесів. Його візуальний конструктор, можливість роботи напряму з Google Sheets, а також наявність шаблонів значно знижують поріг входу, роблячи створення додатків доступним навіть для новачків без досвіду програмування.

Для стартапів, які знаходяться на ранніх стадіях розвитку, AppSheet пропонує ідеальне середовище для побудови тестових версій сервісів, аналізу поведінки користувачів і швидкого коригування ідеї згідно з фідбеком. Його головна перевага – у мінімізації часу між концептом і першим прототипом, без потреби інвестувати значні ресурси в команду розробників на цьому етапі. Це дозволяє сфокусуватися не на технічній реалізації, а на перевірці гіпотез і цінності продукту для цільової аудиторії. Крім того, AppSheet забезпечує достатній рівень гнучкості для швидкого внесення змін у логіку додатку та інтерфейс, що особливо важливо на ранніх етапах, коли бізнес-модель може активно змінюватися. Завдяки інтеграції з

популярними сервісами, такими як Google Sheets та Firebase, стартапи отримують можливість легко збирати й аналізувати дані без необхідності створювати складну інфраструктуру.

У випадку з MVP-проєктами (Minimum Viable Product), де основне завдання – протестувати функціональність, зібрати зворотний зв'язок і виявити ринковий потенціал, AppSheet забезпечує ідеальний баланс між функціональністю, швидкістю реалізації і витратами. Його інтеграція з Firebase, можливість використання сканування, GPS, push-сповіщень, дій по тригерах і доступу до камери робить платформу достатньо гнучкою для більшості типових MVP-функцій без залучення спеціалістів з бекенду чи мобільної розробки [15].

Завдяки моделі швидкого оновлення та публікації, AppSheet також дозволяє оперативно вносити зміни, реагуючи на перші результати тестування, що особливо важливо для динамічних проєктів. І, нарешті, оскільки AppSheet працює повністю у хмарі, немає потреби налаштовувати серверну інфраструктуру, що є додатковою перевагою для команд з обмеженим технічним ресурсом. Порівняння з традиційними підходами до розробки мобільних додатків демонструє, що використання AppSheet значно скорочує час виходу на ринок і знижує початкові інвестиції. Це особливо актуально для стартапів, які прагнуть швидко протестувати свої ідеї, не втрачаючи при цьому гнучкості в подальшому розвитку [20]. Такий підхід дозволяє сфокусуватися на аналізі користувацької поведінки і вдосконаленні продукту на основі реальних даних, що суттєво підвищує шанси на успіх проєкту.

Отже, у сферах, де важливі швидкість, простота, доступність і адаптивність, AppSheet не лише залишається релевантним, а й часто є найбільш ефективним інструментом для старту проєкту, особливо коли йдеться про навчання, експерименти або перевірку життєздатності ідеї без надлишкових витрат. Водночас, варто пам'ятати, що платформа має певні обмеження у масштабуванні та кастомізації, які можуть стати помітними на етапах росту проєкту або при потребі складнішої логіки. Тому, коли MVP підтверджує свою цінність і збільшується кількість користувачів, часто виникає необхідність переходу на більш гнучкі рішення з повноцінною серверною частиною.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ушкодженнях м’яких тканин, суглобів і кісток

У разі травмування м’яких тканин, суглобів або кісток своєчасне надання домедичної допомоги (долікарської) має вирішальне значення для запобігання тяжким ускладненням, інфікуванню та поглибленню травм. Основні дії мають бути організовані відповідно до Наказу МОЗ України від 29.03.2022 № 356 «Про затвердження порядків надання домедичної допомоги особам при невідкладних станах» [40]. Надання допомоги дозволяє зменшити ризик ускладнень, зупинити розвиток шоку, обмежити кровотечу, уникнути інфікування рани та забезпечити стабільний стан потерпілого до моменту прибуття професійної медичної допомоги. Воно повинно відбуватись чітко, спокійно та в логічній послідовності, з урахуванням стану потерпілого та характеру ушкодження.

У випадку ушкоджень м’яких тканин, таких як садна, порізи, подряпини, рвані або колоті рани, першим кроком є оцінка характеру кровотечі. Якщо спостерігається капілярна або венозна кровотеча, її слід зупинити шляхом натискання на ділянку рани чистою серветкою або марлею. Потім місце ушкодження промивають чистою проточною водою або, за наявності, обробляють антисептичними засобами (наприклад, хлоргексидином, перекисом водню). Після очищення на рану накладають стерильну пов’язку, що щільно фіксується, але не перетискає тканини. У разі сильного забруднення (наприклад, при травмі на відкритій місцевості) або якщо у рані є сторонні предмети (осколки, пісок, уламки), їх не слід намагатись вийняти самостійно – варто лише закрити рану стерильною тканиною і якнайшвидше транспортувати людину до медичного закладу.

Надання допомоги повинно відбуватись чітко, спокійно та у визначеній послідовності з урахуванням стану потерпілого. При цьому необхідно суворо дотримуватись принципів безпеки, як для рятувальника, так і для самого потерпілого.

Загальний алгоритм надання долікарської допомоги:

1. Оцінка безпеки місця події.
2. Оцінка стану постраждалого:
   - перевірка свідомості, дихання, кровотечі;
   - виявлення відкритих травм або деформацій.
3. Виклик екстреної допомоги (103 або 112).
4. Надання допомоги відповідно до виявлених ушкоджень.
5. Контроль стану потерпілого до приїзду медиків.

Надання допомоги при травмах м’яких тканин:

У випадку ушкоджень м’яких тканин, таких як садна, порізи, подряпини, рвані або колоті рани:

- Оцінити характер кровотечі.
- При капілярній або венозній кровотечі зупинка здійснюється натисканням стерильної серветки.
- Промити рану водою або антисептиком (хлоргексидин, перекис водню).
- Накласти стерильну пов’язку, що щільно фіксується, але не перетискає тканини.
- У разі сторонніх предметів у рані – не виймати! Тільки захистити від інфекції та транспортувати.
- Надання допомоги при ушкодженнях суглобів (вивихи, розтягнення):
- Заборонено намагатися самостійно вправляти суглоб.
- Забезпечити повну нерухомість ураженої кінцівки, зафіксувавши її в положенні, в якому вона перебуває.
- Накласти холод (лід, загорнутий у тканину) на 10–15 хв для зменшення болю і набряку.
- Уникати різких рухів та навантажень.
- При необхідності – транспортувати потерпілого до медичного закладу.
- Оцінити загальний стан, за потреби — покласти, підняти ноги, зігріти.
- Підтримувати спокій потерпілого, пояснювати дії.

Надання допомоги при переломах:

- Закритий перелом:
  - Накласти імпровізовану шину (рейки, картон, парасоля).
  - Шина повинна фіксувати суглоби вище та нижче перелому.
  - Надійно, але без надмірного тиску, зафіксувати пов’язками.
- Відкритий перелом:
  - Спочатку – стерильна пов’язка на рану.
  - Потім – накласти шину, не натискаючи на рану.
  - При сильній кровотечі накласти джгут вище місця пошкодження з обов’язковим записом часу (на аркуші або прямо на шкірі потерпілого).

Важливо пам’ятати:

- Не можна давати їжу або рідину при підозрі на важкі травми.
- Заборонено переміщувати постраждалого без потреби.
- Усі дії виконуються акуратно, щоб не погіршити стан потерпілого.

Після надання першої допомоги будь-якого рівня тяжкості, особу потрібно якнайшвидше доставити до медичного закладу. Навіть при незначних симптомах травми без відповідної діагностики можливі ускладнення: інфекції, порушення кровопостачання, неправильне зрощення кісток, обмеження рухомості суглобів тощо.

Лише кваліфікований медичний працівник може точно встановити характер ушкодження та призначити ефективне лікування. Він проведе необхідні обстеження, зокрема рентген, УЗД, аналізи крові або інші діагностичні процедури, що дозволяють виявити приховані ушкодження, які не завжди можна розпізнати відразу після травми. Зволікання або самолікування може призвести до хронічного болю, деформацій, втрати працездатності, а в окремих випадках – до інвалідності.

Кожен крок базується на знаннях, логіці та оцінці ситуації. Надання допомоги – це не просто набір дій, а частина культури безпеки, що рятує життя [40].

4.2 Економічне значення заходів щодо покращення умов та охорони праці

Охорона праці – це не лише частина соціальної відповідальності підприємства, а й вимога Конституції та законодавства України. Згідно зі статтею 43 Конституції України, кожен громадянин має право на належні, безпечні й здорові умови праці. Це право гарантується державою і реалізується через відповідні закони, зокрема Закон України «Про охорону праці» № 2694-XII, Кодекс законів про працю (КЗпП), а також численні нормативні документи і постанови Кабінету Міністрів України [37, 38, 39, 41].

Охорона праці – це не лише витрати, а довгострокова інвестиція в безперервність і стабільність виробництва. Вона формує культуру відповідальності та знижує ризики нещасних випадків. Крім того, дотримання вимог безпеки позитивно впливає на мотивацію працівників та їхню продуктивність. Безпечне робоче середовище дозволяє зменшити витрати на:

- компенсацію за нещасні випадки,
- лікування та тимчасову непрацездатність працівників,
- плинність кадрів і повторне навчання,
- простої виробництва,
- юридичні процеси та відшкодування шкоди.

Фінансова та адміністративна відповідальність за порушення законодавства про охорону праці:

Відповідальність роботодавця визначається відповідно до кількох рівнів:

- адміністративна – штрафи за недотримання законодавства;
- матеріальна – відшкодування шкоди потерпілому;
- цивільна – компенсація моральної шкоди;
- кримінальна – за факти загибелі або тяжкого ушкодження працівника [38, 39, 41].

Найпоширеніші штрафи (дані Держпраці України, станом на 2024 рік):

- Відповідно до частини 5 статті 41 Кодексу України про адміністративні правопорушення (КУпАП), порушення вимог законодавчих та інших нормативних актів про охорону праці тягне за собою накладення штрафу на працівників в розмірі від 4 до 10 неоподаткованих мінімумів доходів громадян [42, ст. 41].
- Згідно зі статтею 265 Кодексу законів про працю України, неоформлені трудові відносини тягнуть за собою штраф у розмірі десяти мінімальних заробітних плат за кожного працівника, щодо якого виявлено порушення [39, ст. 265].
- Недопуск інспектора Держпраці до перевірки тягне за собою штраф у розмірі трьох мінімальних заробітних плат, згідно з частиною 2 статті 265 КЗпП [39, ст. 265].
- Порушення правил техніки безпеки, зафіксоване контролюючими органами, але таке, що не призвело до нещасного випадку, тягне за собою адміністративну відповідальність у вигляді штрафу. Розмір штрафу, згідно зі статтею 41 КУпАП, може сягати від 20 до 40 неоподатковуваних мінімумів доходів громадян для посадових осіб підприємств, установ та організацій, а також для громадян-суб'єктів підприємницької діяльності [41, ст. 41]. Для працівників штраф може бути від 4 до 10 неоподатковуваних мінімумів доходів громадян.
- Відсутність проведення обов'язкових інструктажів та навчання працівників з охорони праці карається штрафом. Відповідно до статті 271 Кримінального кодексу України, порушення вимог законодавства про охорону праці, що включає відсутність навчання та інструктажів, може призвести до штрафу від 1000 до 3000 НМДГ або виправних робіт на строк до двох років, або пробаційного нагляду на строк до двох років, або обмеження волі на той самий строк. Крім того, частина 5 статті 41 Кодексу України про адміністративні правопорушення передбачає штраф від 4 до 10 неоподатковуваних мінімумів доходів громадян за порушення вимог законодавчих та інших нормативних актів про охорону праці.

Економічна доцільність заходів з охорони праці:

Впровадження ефективної системи охорони праці дає підприємству низку переваг:

- зменшення витрат на лікарняні, штрафи, відшкодування;
- підвищення продуктивності праці за рахунок створення комфортних умов;
- зниження рівня травматизму та аварій;
- зміцнення ділової репутації та підвищення лояльності персоналу;
- уникнення простоїв та непередбачуваних втрат.

Працівники, які працюють у безпечних і контрольованих умовах, демонструють вищий рівень мотивації, дисципліни та продуктивності. Це позитивно позначається не лише на внутрішньому мікрокліматі підприємства, а й на економічних показниках загалом.

З макроекономічної точки зору, системний підхід до охорони праці в масштабах регіону або галузі:

- сприяє підвищенню добробуту населення;
- знижує тиск на систему охорони здоров’я;
- формує позитивне інвестиційне середовище;
- сприяє євроінтеграційним прагненням України.

Висновок:

Ефективна система охорони праці – це не лише спосіб дотримання норм, а стратегічний ресурс, що гарантує:

- збереження здоров’я працівників;
- зниження витрат;
- захист підприємства від юридичних та репутаційних ризиків.

ВИСНОВКИ

У процесі проведеного дослідження було комплексно проаналізовано, спроєктовано та змодельовано концепцію мобільного додатку для пошуку попутників на основі платформи Google AppSheet. Робота охопила теоретичне вивчення наявних рішень на ринку, зокрема популярних сервісів BlaBlaCar, Liftshare та Poparide, визначення їхніх сильних і слабких сторін, формулювання вимог до нового цифрового продукту, розробку архітектури системи, моделювання логіки та інтерфейсу, а також порівняльний аналіз із альтернативними інструментами.

Зокрема, було з’ясовано, що сучасні сервіси пошуку попутників мають розвинений функціонал, але не завжди враховують локальні особливості, потреби малих спільнот або обмеженість технічних ресурсів користувачів. Це створює передумови для появи більш гнучких, доступних і локалізованих рішень на основі no-code технологій, до яких належить AppSheet.

Обґрунтування вибору Google AppSheet як основного інструменту реалізації базувалося на кількох ключових факторах. По-перше, платформа надає можливість створення повноцінних мобільних додатків без потреби у традиційному програмуванні, що істотно знижує поріг входу для розробників-початківців. По-друге, тісна інтеграція з екосистемою Google (Sheets, Maps, Firebase) дозволяє легко працювати з даними, реалізовувати геолокаційні функції, надсилати сповіщення та керувати автентифікацією користувачів. По-третє, AppSheet забезпечує високу швидкість прототипування, що дає змогу швидко перевіряти гіпотези, вносити зміни та адаптувати логіку до нових умов без перезапуску системи.

У ході моделювання було реалізовано логічну структуру даних, побудовано систему взаємозв’язків між таблицями користувачів, маршрутів, поїздок і відгуків, а також розроблено елементи інтерфейсу користувача з урахуванням ролей (водій/пасажир), бізнес-процесів, системи рейтингу й доступу. Побудовано низку

UML-діаграм (діаграма класів, варіантів використання, бізнес-процесів і діяльності), що візуалізують ключові аспекти логіки роботи сервісу.

Проведений порівняльний аналіз AppSheet з іншими платформами – такими як Microsoft PowerApps, Adalo, FlutterFlow – підтвердив переваги обраного рішення саме в контексті швидкості, доступності, простоти інтеграції та низької вартості впровадження. Особливо яскраво ці переваги проявляються в освітніх, стартапових і MVP-проєктах, де обмежений бюджет і необхідність швидкого тестування ідеї мають критичне значення.

У підсумку можна стверджувати, що використання Google AppSheet у рамках даного проєкту є технічно доцільним, економічно виправданим і методологічно обґрунтованим вибором, що дозволяє досягти поставлених цілей із мінімальними витратами, зберігаючи при цьому високу якість реалізації.

Щодо перспектив подальшого розвитку проєкту, доцільно виділити кілька ключових напрямів. По-перше, у разі успішної реалізації базової версії додатку, можна розширити функціонал за рахунок впровадження системи платіжних модулів (інтеграція з Google Pay), а також створення більш розвиненого чату між учасниками поїздки. По-друге, можлива адаптація додатку під потреби конкретних локальних громад (наприклад, для студентів, працівників одного підприємства, жителів сільської місцевості). По-третє, перспективним напрямом є побудова аналітичного модуля для збору статистики про маршрути, рівень задоволеності користувачів та завантаження системи. Також, потенційно цікавим є запуск API-доступу для підключення сторонніх сервісів або вбудовування функціоналу в інші додатки.

Загалом, проєкт демонструє реальний потенціал застосування безкодових технологій у вирішенні прикладних задач сучасного суспільства та формує платформу для подальших досліджень, масштабування та впровадження інновацій у сферу мобільних сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. McCloud. Як Google AppSheet допоможе оптимізувати роботу вашого бізнесу. URL: https://mccloud.global/ua/

2. AppSheet – лідер ринку платформ із низьким кодом. URL: https://wiseit.com.ua/appsheet-market-leader/

3. AppSheet – огляд і застосування для оптимізації бізнес-процесів. URL: https://saas-distri.com/appsheet/

4. Сервіс для пошуку автомобільних попутників. URL: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/2367/1/402 Бочков Максим Володимирович.pdf

5. BlaBlaCar. URL: https://www.blablacar.com.ua/

6. Poparide. URL: https://www.poparide.com/

7. Оптимізація роботи з AppSheet. URL: https://cloudfresh.com/ua/cloud-blog/optymizatsiya-roboty-z-appsheet/

8. Як використовувати AppSheet для автоматизації процесів? URL: https://tsystem.com.ua/yak-vykorystovuvaty-appsheet-dlya-avtomatyzacziyi-proczesiv/

9. Огляд Google Cloud AppSheet. URL: https://bizmag.com.ua/ohliad-google-cloud-appsheet/

10. Автоматизація за допомогою AppSheet Automation. URL: https://cloudfresh.com/ua/cloud-blog/avtomatyzatsiya-za-dopomogoyu-appsheet-automation/

11. No-code розробка: яке майбутнє чекає сферу додатків. URL: https://uaspectr.com/2021/01/28/rozrobka-bez-kodu/

12. Low-code і no-code платформи: загроза для розробників чи нові можливості. URL: https://ain.ua/2025/05/26/low-code-ta-no-code-platformi/

13. Як Google AppSheet допоможе оптимізувати роботу вашого бізнесу. URL: https://mccloud.global/ua/yak-google-appsheet-dopomozhe-optymizuvaty-robotu-vashogo-biznesu/

14. Биков В. Ю. Відкриті web-орієнтовані системи моніторингу впровадження результатів науково-педагогічних досліджень / Биков В. Ю., Спірін О. М., Лупаренко Л. А.. // Теорія і практика управління соціальними системами. – 2014. – № 1. – С. 3–25.

15. Биков В.Ю. Мобільний простір і мобільно орієнтоване середовище Інтернет-користувача: особливості модельного подання та освітнього застосування [Електронний ресурс] / Биков В. Ю. / Інформаційні технології і засоби навчання: електронне наукове фахове видання //Інформаційні технології в освіті. – 2013. - № 17. - ст. 09-37.

16. Ігнатенко, М. В. Google AppSheet як інструмент no-code розробки мобільних застосунків у навчальному процесі / М. В. Ігнатенко // Вісник Кременчуцького нац. ун-ту імені М. Остроградського. – 2023. – № 2(143). – С. 56–62.

17. Марченко О. В. Доцільність використання no-code технологій для швидкого розвитку цифрових продуктів / О. В. Марченко. – Київ : КНУ, 2022.

18. Онопрієнко, А. В. Цифрова трансформація освіти: виклики, інструменти, перспективи / А. В. Онопрієнко // Освіта та розвиток обдарованої особистості. – 2021. – № 6. – С. 4–9.

19. Петрів, І. І. Економічні аспекти та особливості охорони праці на підприємствах харчової промисловості. 2024.

20. Розробка стартап-проектів: Конспект лекцій: навч. посіб. для студ. спеціальностей 151 – «Автоматизація та комп’ютерно-інтегровані технології» та 152 – «Метрологія та інформаційно-вимірювальна техніка»/ О.А. Гавриш, К.О. Бояринова, К.О. Копішинська; КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2019. – 188 с.

21. Романів, Л. В., Бабух, І. Б. Охорона праці в Україні: проблеми, досвід, перспективи [Електронний ресурс] / Л. В. Романів, І. Б. Бабух // Соціально-економічні проблеми сучасного періоду України. – 2014. – Вип. 4(108).

22. Чаплінська І. В. Цифрова грамотність в освітньому процесі: практичне застосування Google-сервісів // Інформаційні технології і засоби навчання. – 2022. – № 2(90). – С. 123–130.

23. Юрковський С. С. Автоматизація бізнес процесів у малому та середньому бізнесі за допомогою no-code та low-code платформ: кваліфікаційна робота за спеціальністю 123 Комп'ютерна інженерія / С. С. Юрковський; наук. кер. Д. В. Стаценко. – Київ : КНУТД, 2024. – 75 с.

24. A. Abran, Software Metrics and Software Metrology: Wiley-IEEE Computer Society Press, 2010.

25. Alamin M. A., Malakar S. et al. An Empirical Study of Developer Discussions on Low-Code Software Development Challenges // arXiv. – 2021.

26. C.Y Laporte, A.April, Software Quality Assurance, IEEE Computer Society Press, 1st ed., 2018.

27. D. Budgen, Software Design, 3rd ed., CRC Press, 2021.

28. D.C. Montgomery and G.C. Runger, Applied Statistics and Probability for Engineers, 7th ed. Hoboken, NJ: Wiley, 2018.

29. E.A. Commission, "Criteria for Accrediting Engineering Programs, 2022-2023," ABET, 2021.

30. Elisa Yumi Nakagawa, Pablo Oliveira Antonio, Frank Schnicke, Thomas Kuhn, Peter Liggesmeyer, Continuous Systems and Software Engineering for Industry 4.0: A disruptive view, Elsevier, Volume 135, July 2021.

31. E.W. Cheney and D.R. Kincaid, Numerical Mathematics and Computing, 6th ed. Belmont, CA: Brooks/Cole, 2007.

32. ISO/IEC/IEEE 12207-2017 Systems and Software Engineering — Software Life Cycle Processes, 2017.

33. Luo Y., Liang P., Wang C. et al. Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective // arXiv. – 2021.

34. Meho L.I., Yang K. Impact of data sources on citation counts and rankings of LIS faculty: Web of Science versus Scopus and Google Scholar // J. Am. Soc. Inf. Sci. – 2007. – V. 58, N 13.

35. Yogesh R. Python: Simple though an Important Programming language / R. Yogesh // International Research Journal of Engineering and Technology, 2019. – Vol. 06. – P. 1856-1858.

36. Конституція України : Закон України від 28 черв. 1996 р. № 254к/96-ВР // Відомості Верховної Ради України. – 1996. – № 30. – Ст. 141.

37. Про охорону праці : Закон України від 14 жовт. 1992 р. № 2694-XII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12

38. Кодекс законів про працю України : Закон України від 10 груд. 1971 р. № 322-VIII [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08

39. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах : Наказ МОЗ України від 29 берез. 2022 р. № 356 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0356-22

40. Про затвердження Порядку накладення штрафів за порушення законодавства про охорону праці : Постанова Кабінету Міністрів України від 17 лип. 2013 р. № 1107 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1107-2013-п

41. Про адміністративні правопорушення : Закон України від 7 грудня 1984 р. №4316-IX [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/80731-10

42. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329

43. Методичні вказівки до виконання дипломної роботи освітнього рівня - бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення/ Укладачі : Петрик М.Р.,Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

ДОДАТКИ

Додаток А – Диск з роботою