СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр
спеціальність 126 „Інформаційні системи та технології”
(шифр і назва спеціальності)

на тему „Проектування та розробка мобільного застосунку для онлайн-бронювання столиків у ресторанах”

Виконала: студентка групи ІСТ-21д	(підпис)	С. М. Малишко (ініціали і прізвище)
Керівник	(підпис)	О. С. Дюбанов (ініціали і прізвище)
Завідувач кафедри	(підпис)	О.І. Захожай (ініціали і прізвище)

Рецензент ____Ратов Д.В.____

Київ – 2025

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Освітній ступінь бакалавр
спеціальність 126 „Інформаційні системи та технології”
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____ Захожай О.І.
“___” _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

Малишко Софія Миколаївна
(прізвище, ім’я, по батькові)

1. Тема роботи: Проектування та розробка мобільного застосунку для онлайн-бронювання столиків у ресторанах

Керівник роботи Дюбанов Олексій Сергійович, к.е.н. , доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом університету від “27” травня 2025 року № 67/15.15-С

2. Строк подання студентом роботи 16.06.2025

3. Вихідні дані до роботи: Об’єктом даної роботи виступає інформаційна система для автоматизації процесів онлайн-бронювання столиків у закладах громадського харчування

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ. Аналіз предметної області та існуючих рішень, з висвітлюванням наступних питань:аналіз потреб цільової аудиторії, огляд функціональних вимог до мобільного застосунку, аналіз схожих мобільних рішень на ринку, програмні підходи до реалізації системи. Основна частина , в якої висвітлено: збір вимог та проектування, етап реалізації проекту. Висновки. Список використаних джерел.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 01.04.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	19.05.2025	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	19.05.2025	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	28.05.2025	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	28.05.2025	виконано
5	Укладання та тестування програмного продукту	09.06.2025	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	18.06.2025	виконано
7	Здача готової пояснювальної записки на кафедру	18.06.2025	виконано
8	Укладання доповіді і презентації	19.06.2025	

Студент _____ С. М. Малишко
підпис (ініціали і прізвище)

Керівник роботи _____ О. С. Дюбанов
підпис (ініціали і прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему:«Проєктування та розробка мобільного застосунку для онлайн-бронювання столиків у ресторанах»,містить: 76 сторінок основного тексту, 11 додаткових, ілюстрації, 15 інформаційних джерел.

Об’єкт дослідження – інформаційна система для автоматизації процесів онлайн-бронювання столиків у закладах громадського харчування.

Предмет дослідження – методи, інструменти та технології розробки мобільних застосунків для онлайн-бронювання.

Мета кваліфікаційної роботи – розробити кросплатформений мобільний застосунок для онлайн-бронювання столиків у ресторанах, що відповідає потребам кінцевих користувачів і закладів громадського харчування.

Методи дослідження – аналіз літературних джерел, порівняння існуючих рішень, анкетування, проєктування, моделювання програмного продукту.

У процесі дослідження було проаналізовано потреби користувачів, існуючі сервіси бронювання, сформульовано вимоги до майбутньої системи, обґрунтовано вибір інструментів для розробки, розроблено архітектуру та реалізовано функціональний прототип застосунку з використанням Flutter, Firebase та Google Maps API.

Для створення мобільного застосунку для онлайн-бронювання столиків використані інструменти та забезпечення, які були обрані у ході дослідження як найбільш ефективні в порівнянні з їх аналогами.

ЗМІСТ

Вступ

У добу цифрових технологій мобільні застосунки стали ключовим інструментом для покращення взаємодії між бізнесом і споживачами. Вони дозволяють автоматизувати рутинні процеси, зменшити витрати часу та ресурсів, а також покращити якість обслуговування клієнтів. Особливу увагу в останнє десятиріччя привертає галузь громадського харчування, яка активно впроваджує цифрові рішення для покращення сервісу. Одним із таких рішень є системи онлайн-бронювання столиків у ресторанах.

У багатьох закладах досі практикується застарілий метод бронювання: клієнти змушені телефонувати адміністратору, надсилати повідомлення в соціальні мережі або особисто приходити в ресторан. Це створює додаткове навантаження на персонал, призводить до непорозумінь, дублювання бронювань або їх втрати, а також значно знижує зручність для кінцевого користувача. Зі зростанням попиту на швидке, інтуїтивно зрозуміле та доступне бронювання столиків постає необхідність у створенні мобільного застосунку, який би дозволив вирішити ці проблеми.

Мобільні пристрої стали основною платформою для доступу до цифрових сервісів, тому саме мобільний застосунок є найоптимальнішим рішенням для впровадження функціоналу бронювання. Такий застосунок має забезпечити користувачам можливість у декілька кліків знайти ресторан, переглянути доступність столиків на певну дату й час, забронювати місце, отримати підтвердження, а також за необхідності — скасувати або змінити бронювання. Для власників та адміністраторів ресторану це дозволяє оперативно відслідковувати завантаженість, планувати роботу персоналу та покращувати обслуговування клієнтів.

Тема дипломної роботи є актуальною з кількох причин:

- зростаюча потреба в автоматизації сервісів у ресторанному бізнесі;
- високий рівень використання мобільних пристроїв серед населення;

- розвиток технологій, які дозволяють швидко і ефективно створювати масштабовані кросплатформені рішення (наприклад, Flutter);
- можливість інтеграції з зовнішніми сервісами — картами, базами даних, повідомленнями тощо.

Метою дипломної роботи є розробка функціонального мобільного застосунку для онлайн-бронювання столиків у ресторанах, який би забезпечував простий інтерфейс, високу продуктивність, стабільність роботи та можливість подальшої масштабованості.

Для досягнення цієї мети у роботі поставлено наступні завдання:

- здійснити аналіз потреб користувачів і типових процесів у закладах громадського харчування;
- проаналізувати ринок аналогічних мобільних застосунків і виявити їх сильні та слабкі сторони;
- обґрунтувати вибір технологій та інструментів для реалізації проєкту;
- визначити функціональні та нефункціональні вимоги до системи;
- спроєктувати архітектуру клієнт-серверної взаємодії;
- розробити прототип інтерфейсу з урахуванням принципів UX/UI;
- реалізувати основні модулі застосунку — реєстрацію, авторизацію, пошук ресторанів, бронювання столиків, сповіщення;
- провести тестування функціоналу та оцінити результати розробки.

Об'єктом дослідження є процеси автоматизації обслуговування клієнтів у ресторанному бізнесі за допомогою мобільних технологій. Предметом дослідження є методи та засоби проєктування й реалізації мобільного застосунку для онлайн-бронювання.

У процесі розробки було використано сучасні підходи до створення мобільних застосунків, зокрема: кросплатформену технологію Flutter, хмарну базу даних Firebase, інтеграцію з Google Maps API, а також методології проектування UI/UX інтерфейсів.

Таким чином, дана дипломна робота має практичну цінність і потенціал до подальшого розвитку — розширення функціоналу, інтеграція з платіжними системами, система відгуків, аналітика для адміністраторів тощо.

Актуальність дослідження.

Сучасна ресторанна індустрія зазнає значних змін під впливом цифрових технологій. Все більше користувачів очікують можливості здійснювати онлайн-бронювання, отримувати підтвердження в режимі реального часу та взаємодіяти з інтерфейсом без зайвих ускладнень. Однак український ринок досі не має універсального рішення, адаптованого під локальні потреби, мовні особливості та специфіку малого бізнесу. Це формує попит на зручний, доступний і масштабований мобільний сервіс.

Мета роботи.

Метою дипломного проєкту є розробка кросплатформеного мобільного застосунку для онлайн-бронювання столиків у ресторанах з урахуванням вимог кінцевих користувачів та закладів громадського харчування.

Об’єкт дослідження.

Об’єктом дослідження виступають інформаційні системи для автоматизації процесів онлайн-бронювання у сфері громадського харчування.

Предмет дослідження.

Предметом дослідження є процеси, методи та інструменти розробки мобільних застосунків для бронювання столиків, включаючи технічну архітектуру, UX/UI, функціональні вимоги та засоби інтеграції з геолокаційними та базовими сервісами.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ

1.1 Аналіз потреб і вимог до мобільного застосунку

Першим та одним із найважливіших етапів розробки будь-якого програмного забезпечення є ретельний аналіз потреб цільової аудиторії та формулювання вимог до системи. .Саме з цього почалася робота над застосунком, адже грамотний аналіз дозволяє уникнути критичних помилок у подальшому розробницькому процесі, забезпечити відповідність функціоналу очікуванням користувачів та досягти загальної ефективності системи.

Аналіз проблемної області

Сфера ресторанного бізнесу переживає активну цифрову трансформацію. Проте, попри наявність окремих застосунків для певних закладів, більшість ресторанів усе ще покладаються на традиційні засоби бронювання: телефонні дзвінки, повідомлення в соціальних мережах або особисті візити. Ці методи не завжди зручні як для персоналу, так і для клієнтів. Вони не дозволяють ефективно контролювати потік відвідувачів, спричиняють непорозуміння та не гарантують підтвердження резерву. Особливо це відчутно у години пік або під час свят, коли попит на бронювання зростає.

Тому на сучасному етапі виникає необхідність створення зручного мобільного рішення, яке б дозволяло користувачам швидко знаходити потрібний заклад, переглядати доступні години для бронювання та фіксувати замовлення у кілька кліків. З іншого боку, власники ресторанів мають отримати просту адміністративну панель для керування резервуваннями без залучення додаткового персоналу.

Аналіз цільової аудиторії

Під час дослідження було проведено аналіз майбутніх користувачів застосунку. Для цього було використано відкриті статистичні дані, а також

опитування у формі онлайн-анкетування (умовно моделюючи цей етап). Було визначено такі основні групи користувачів:

1. Молодь віком 18–25 років — активні користувачі мобільних додатків, які полюбляють спонтанні зустрічі та швидке бронювання.
2. Молоді сім'ї та пари 26–35 років — переважно планують візити заздалегідь, цінують комфорт і зручність.
3. Бізнес-користувачі — часто організовують зустрічі в ресторанах, зацікавлені у точності бронювань і можливості повторних візитів.

В результаті аналізу були сформульовані ключові очікування користувачів:

1. простий та інтуїтивно зрозумілий інтерфейс;
2. доступність інформації про меню, акції, рейтинг та відгуки;
3. візуалізація вільних столиків у режимі реального часу;
4. підтримка миттєвого підтвердження бронювання;

Рисунок 1.1 – Очікування користувачів від мобільного застосунку для бронювання столиків у ресторанах

Функціональні вимоги

Функціональні вимоги були сформульовані на основі результатів дослідження аудиторії, конкурентного аналізу та типових сценаріїв використання застосунку. Основні з них включають:

1. Реєстрація та авторизація — через email, телефон або обліковий запис Google/Facebook.
2. Пошук закладів — за назвою, локацією, типом кухні, рейтингом.
3. Інтерактивна карта ресторанів — інтеграція з Google Maps API.
4. Інформаційна картка закладу — фото, опис, графік роботи, контакти.
5. Бронювання столика — вибір дати, часу, кількості гостей, підтвердження.
6. Особистий кабінет користувача — історія бронювань, улюблені заклади, нагадування.
7. Push-сповіщення — підтвердження, зміни, акційні пропозиції.
8. Роль адміністратора ресторану — управління графіком, підтвердження бронювань.

Нефункціональні вимоги

Нефункціональні вимоги відображають загальні технічні та якісні характеристики, якими має володіти застосунок:

1. Кросплатформеність — підтримка Android та iOS з єдиною кодовою базою.
2. Продуктивність — швидке завантаження, відсутність затримок при взаємодії.
3. Інформаційна безпека — захист персональних даних (шифрування, автентифікація).

4. Інтерфейс користувача — відповідність сучасним UX/UI стандартам (Material Design).
5. Масштабованість — можливість додавання нових функцій у майбутньому.
6. Інтеграційна здатність — сумісність з API карт, платіжних систем та баз даних.

Аналіз конкурентів

У процесі дослідження було розглянуто кілька вже існуючих застосунків для бронювання столиків:

1. OpenTable — лідер у США, підтримує бронювання з відгуками та бонусами.
2. Resy — орієнтований на преміальні заклади з системою "смарт-бронювання".
3. Tablein — європейський аналог з мінімалістичним інтерфейсом і базовими функціями.

Попри їхню функціональність, більшість таких сервісів мають складну інтеграцію для малих ресторанів, працюють тільки в окремих країнах або потребують значних витрат на підключення. Це дає змогу створити власне рішення, адаптоване до локального ринку та конкретних потреб користувачів.

Крім розглянутих вимог, на цьому етапі дослідження було також проаналізовано сценарії використання системи з точки зору різних типів користувачів. Це дозволило не лише перевірити логіку взаємодії в межах проєктованої архітектури, а й виявити потенційні "вузькі місця" в майбутній реалізації.

Типові сценарії використання:

1. Користувач вперше відкриває застосунок: бачить короткий опис функціоналу, проходить просту реєстрацію, після чого потрапляє на головний екран із картою або списком закладів.
2. Бронювання столика: користувач обирає ресторан, відкриває профіль закладу, переглядає доступні дати й години, вказує кількість гостей, підтверджує бронювання та отримує push-сповіщення.
3. Менеджер ресторану: через адміністративну панель переглядає список поточних бронювань, має змогу підтверджувати, змінювати або скасовувати їх, а також налаштовувати доступні години, кількість столиків, коментарі до замовлення.
4. Випадок відміни чи редагування: користувач заходить у свій профіль, переглядає історію бронювань, вносить зміни або скасовує замовлення за кілька годин до події.

На базі цих сценаріїв було складено початкову карту навігації (user flow), яка ляже в основу розробки макетів інтерфейсу. Таким чином, розробка застосунку орієнтується не лише на технічні вимоги, а й на зручність користувача (user-centered design).

Фактори, що впливають на прийняття рішень:

Під час аналізу також було враховано ряд зовнішніх факторів, які можуть впливати на вибір користувачем того чи іншого ресторану через застосунок:

1. Динамічне завантаження місць: система має оновлювати дані в реальному часі, щоб уникнути конфліктів при бронюванні.
2. Гнучкість управління: заклади мають мати можливість швидко змінювати години роботи, обмеження на кількість гостей, блокувати бронювання на певні дні (наприклад, у разі закритого заходу).

3. Інтеграція з календарем користувача: можливість додати бронювання у Google Calendar чи iOS Calendar після підтвердження.
4. Можливість залишити відгук після відвідування — для формування репутації закладу всередині платформи.

Загалом, на цьому етапі вдалося сформувати чітку структуру майбутнього застосунку, де кожна функція та сценарій обґрунтовані конкретними потребами кінцевих користувачів. Саме на основі цих висновків було обґрунтовано подальший вибір інструментів і технологій розробки, що розглядатиметься в наступному розділі.

Під час аналітичного етапу дослідження важливим стало не лише виявлення ключових вимог, а й глибше розуміння поведінкових моделей потенційних користувачів. Досвід показує, що ефективність будь-якого цифрового продукту напряму залежить від того, наскільки точно він відповідає реальним щоденним звичкам людей. Тому особливу увагу було приділено моделюванню типових життєвих ситуацій, у яких людина могла б скористатися застосунком: швидке бронювання місця на вечір п’ятниці, планування обіду під час робочої перерви, організація святкування або зустрічі з друзями.

Під час аналізу потреб стало очевидно, що користувач очікує не просто інструменту бронювання, а цілісного сервісу — такого, що забезпечує просту навігацію, прозору інформацію про заклад, можливість отримати зворотний зв’язок, а також відчуття контролю над ситуацією. Саме на цьому ґрунтувалися всі формулювання функціональних та нефункціональних вимог.

Одним із важливих висновків стало усвідомлення, що в проєктуванні таких сервісів варто враховувати не лише логіку зручної взаємодії, а й психологічний комфорт користувача. Наприклад, сам факт наявності підтвердження бронювання у вигляді повідомлення або навіть запису в

календарі значно підвищує рівень довіри до застосунку. А швидка відповідь системи у відповідь на дію користувача зменшує рівень тривожності, який іноді виникає при використанні нових цифрових сервісів.

Окремо було проаналізовано управлінські аспекти з боку ресторанів. Не менш важливо, щоб інтерфейс адміністратора був інтуїтивним і не вимагав спеціальної технічної підготовки. У процесі розробки було виявлено, що багато закладів не мають постійного технічного персоналу, а значить, усі налаштування мають бути доступні для звичайного менеджера чи адміністратора з базовими цифровими навичками.

Також у процесі аналітичної роботи стало зрозуміло, що проєкт не може бути ефективним без гнучкої логіки взаємодії із закладами. Наприклад, необхідно врахувати можливість ручного коригування доступних годин, швидкого блокування окремих днів або столиків, а також відображення актуальних змін у режимі реального часу. Ці вимоги суттєво вплинули на подальший вибір технічної архітектури майбутнього застосунку.

Таким чином, результатом цього етапу стало не лише формальне складання переліку функцій, а й побудова концептуальної моделі майбутнього продукту, яка враховує як технічні, так і людські, організаційні та навіть емоційні фактори. Це дало змогу перейти до наступного етапу — визначення інструментів та засобів реалізації розробки з урахуванням усіх напрацьованих критеріїв.

1.2 Основні принципи розробки мобільних застосунків

Розробка мобільного застосунку є складним процесом, що охоплює низку етапів — від визначення потреб користувачів і формулювання вимог до програмного продукту, до створення прототипів, розробки, тестування й підтримки. Успіх застосунку значною мірою залежить від дотримання базових принципів, які забезпечують якість, функціональність та зручність використання цифрового рішення.

Одним із ключових принципів є архітектурна чіткість. Застосунок має мати продуману структуру, що дає змогу розмежовувати логіку, інтерфейс користувача та взаємодію з серверною частиною. Це дозволяє легко вносити зміни, додавати нові функції, здійснювати масштабування або оновлення без необхідності повного переписування коду. Найчастіше використовується клієнт-серверна архітектура, де вся логіка зберігається на сервері, а клієнт відповідає за відображення та обробку введення користувача.

Важливим є дотримання принципу кросплатформенності, який передбачає можливість запуску застосунку на кількох платформах (Android, iOS) без необхідності створення окремих версій. Застосування фреймворків на кшталт Flutter, React Native або Xamarin дозволяє розробляти один програмний код, який компілюється під різні операційні системи. Це знижує витрати часу й ресурсів на підтримку, забезпечує одночасне оновлення функціоналу та уніфіковану логіку роботи застосунку.

Ще одним важливим принципом є масштабованість, тобто здатність системи адаптуватися до зростання кількості користувачів або збільшення навантаження без втрати продуктивності. Це стосується не лише серверної частини, яка повинна обробляти запити одночасно від багатьох користувачів, але й внутрішньої структури застосунку. Код має бути модульним, компоненти — незалежними, а логіка — прозорою, щоб її можна було розширювати або змінювати у процесі розвитку продукту.

Окремо варто зупинитися на продуктивності застосунку. Користувач очікує, що мобільний застосунок працюватиме швидко, без зависань або тривалих завантажень. Це особливо важливо для сервісів типу онлайн-бронювання, де зручність і швидкість взаємодії мають вирішальне значення. Оптимізація продуктивності досягається шляхом ефективного управління пам’яттю, зменшення кількості запитів до сервера, локального кешування інформації та використання асинхронних процесів.

Ще один фундаментальний принцип — надійність і безпека. Мобільні застосунки часто працюють із персональними даними користувача: іменем, контактною інформацією, історією бронювань. Важливо, щоб усі дані зберігалися та передавалися із дотриманням принципів безпеки, таких як шифрування даних, автентифікація користувача, захист від SQL-ін’єкцій, контроль сесій. Особливо це стосується застосунків, які мають доступ до мережі та використовують сторонні API.

Важливою складовою успішного застосунку є інтерфейс користувача — зрозумілий, адаптивний і інтуїтивно логічний. UX/UI-дизайн визначає, наскільки швидко користувач може зорієнтуватися у функціоналі застосунку, знайти необхідну інформацію або здійснити дію (наприклад, забронювати столик). Сучасний підхід до дизайну орієнтований на мінімалізм, візуальну чистоту, адаптивність під різні розміри екранів, а також доступність для людей із різними рівнями цифрової грамотності.

Також варто враховувати адаптивність і гнучкість. Застосунок повинен враховувати різні сценарії використання, а також дозволяти просте оновлення, локалізацію та персоналізацію. Наприклад, у контексті ресторанного бізнесу користувач може мати різні очікування: хтось шукає ресторан поблизу, інший — за типом кухні, третій — за рейтингом. Мобільний застосунок має бути достатньо гнучким, щоб обслуговувати всі ці сценарії без втрати зручності.

Зрештою, важливо дотримуватися принципу ітеративної розробки. Це означає, що застосунок створюється поступово: спочатку формується MVP (мінімально життєздатний продукт), який потім розширюється на основі зворотного зв’язку від користувачів. Такий підхід дозволяє вчасно виявляти помилки, адаптувати функціонал до реальних потреб та уникати зайвих витрат на нереалізовані або непотрібні функції.

Дотримання зазначених принципів забезпечує не лише стабільну роботу мобільного застосунку, а й високу якість користувацького досвіду, що є ключовим чинником успіху будь-якого цифрового рішення.

1.3 Архітектура клієнт-сервер

Однією з базових архітектурних моделей для створення сучасних мобільних застосунків є модель клієнт-сервер. Цей підхід передбачає розподіл функціональності між двома основними компонентами: клієнтом (мобільним застосунком) та сервером (віддаленим обчислювальним середовищем або хмарною платформою). Така архітектура дозволяє досягти гнучкості, стабільності, масштабованості та безпеки програмного продукту.

У контексті мобільного застосунку для онлайн-бронювання столиків клієнтська частина відповідає за інтерфейс взаємодії з користувачем, відображення інформації, обробку подій (натискань кнопок, введення даних тощо) та ініціацію запитів до сервера. Сервер, у свою чергу, виконує обробку цих запитів: здійснює перевірку прав доступу, працює з базою даних, зберігає інформацію про бронювання, надсилає відповіді клієнту.

Однією з переваг цієї моделі є те, що на клієнтський пристрій завантажується лише той обсяг даних, який необхідний у конкретний момент часу. Це дозволяє зменшити обсяг передаваних даних, знизити навантаження на пам’ять пристрою та оптимізувати час відповіді системи. Користувач бачить лише актуальну інформацію — вільні столики, доступні

дати, підтвердження бронювання, не помічаючи складної внутрішньої логіки на сервері.

Серверна частина, як правило, реалізується у вигляді веб-сервісу з використанням RESTful API або GraphQL. У нашому випадку доцільно використовувати REST API, що забезпечує просту і зрозумілу модель взаємодії на основі HTTP-запитів (GET, POST, PUT, DELETE). Сервер опрацьовує запити, зберігає дані у базі (наприклад, Firebase Realtime Database або Firestore) і повертає результат у вигляді відповіді у форматі JSON. Така реалізація є зручною для інтеграції з мобільними застосунками, особливо при використанні Flutter.

Загальна схема взаємодії виглядає так:

1. Користувач вводить дані (наприклад, вибирає дату і час бронювання).
2. Застосунок надсилає запит до сервера.
3. Сервер обробляє запит, перевіряє доступність обраного часу, зберігає або відхиляє запит.
4. Користувач отримує відповідь: підтвердження бронювання або повідомлення про відмову.

Крім базової функціональності, сервер може також взаємодіяти з додатковими службами — системами повідомлень (наприклад, Firebase Cloud Messaging для push-сповіщень), платіжними сервісами, аналітичними модулями тощо. Це дозволяє створити повноцінну екосистему, у якій мобільний застосунок є лише частиною загального рішення.

Ще одним важливим аспектом є автентифікація та авторизація користувачів. У системах бронювання необхідно забезпечити захист персональних даних, уникнути несанкціонованого доступу та надати користувачеві можливість бачити лише свою інформацію. Зазвичай для цього використовуються токени доступу, які видаються після успішної

автентифікації (наприклад, за допомогою Firebase Authentication) і додаються до кожного запиту.

Архітектура клієнт-сервер є також ефективною з точки зору оновлень. При внесенні змін на сервері (наприклад, змін у бізнес-логіці чи базі даних) не потрібно оновлювати застосунок на всіх пристроях користувачів, що значно спрощує підтримку і масштабування системи.

У підсумку клієнт-серверна архітектура дозволяє забезпечити:

– централізоване зберігання даних і резервне копіювання;
– безпечний доступ до інформації через контроль прав користувача;
– гнучке управління логікою системи без втручання у клієнтську частину;
– швидке оновлення без необхідності повторного встановлення застосунку.

Завдяки цьому саме така архітектура є оптимальною для розробки мобільного застосунку для онлайн-бронювання столиків.

1.4 Кросплатформеність і доступність

У сучасній розробці мобільних застосунків особливе значення мають такі характеристики, як кросплатформеність і доступність. Вони безпосередньо впливають на ефективність розробки, охоплення аудиторії та зручність використання застосунку. В умовах зростаючої конкуренції на ринку цифрових сервісів саме ці чинники визначають комерційну та соціальну життєздатність проєкту.

Кросплатформеність — це здатність застосунку функціонувати на різних операційних системах із єдиною кодовою базою. Традиційно мобільні застосунки розроблялися окремо для Android (на Java або Kotlin) і для iOS (на Swift або Objective-C), що вимагало подвійних зусиль з боку розробників, подвоєння часу тестування та більших витрат на підтримку. З появою кросплатформених фреймворків ця ситуація суттєво змінилася.

Одним із найпопулярніших сучасних інструментів є Flutter — технологія від компанії Google, яка дозволяє створювати нативно

скомпільовані застосунки для Android і iOS з одного набору вихідного коду. Цей фреймворк має низку переваг: високу продуктивність, багатий набір готових віджетів, гнучку систему компонування інтерфейсу та активну спільноту розробників. Flutter забезпечує зовнішній вигляд і поведінку застосунку, максимально наближені до нативного, що позитивно впливає на користувацький досвід.

Крім скорочення витрат на розробку, кросплатформеність дозволяє швидше оновлювати застосунок. Усі виправлення і нові функції реалізуються одразу для обох платформ. Це особливо важливо для стартапів або невеликих проєктів, де ресурси обмежені, а швидкість виходу на ринок відіграє вирішальну роль.

Другий ключовий аспект — доступність. Вона розглядається у кількох вимірах: технічному, візуальному та функціональному. З технічної точки зору застосунок має коректно працювати на пристроях із різними характеристиками — розмірами екранів, потужністю, версіями операційної системи. Не менш важливою є підтримка різних мов інтерфейсу, включно з українською, що підвищує зручність використання та адаптованість застосунку до локального ринку.

У візуальному аспекті доступність означає дотримання стандартів контрастності, розмірів шрифтів, достатньої читабельності інтерфейсу. Сюди ж входить підтримка роботи з допоміжними технологіями — наприклад, екранними читачами для людей з порушенням зору. Усе це має забезпечувати рівний доступ до функціоналу застосунку для всіх користувачів, незалежно від їхніх можливостей.

Функціональна доступність передбачає інтуїтивність інтерфейсу, просту навігацію, мінімізацію кроків для досягнення результату. У випадку з онлайн-бронюванням — це можливість здійснити повний цикл дії (вибір

ресторану, перегляд інформації, бронювання, підтвердження) без додаткових пояснень або навчання.

Таким чином, кросплатформеність і доступність не є лише технічними характеристиками — це стратегічні переваги, які дають змогу охопити ширшу аудиторію, скоротити витрати, підвищити якість продукту та сформувати позитивний імідж сервісу в очах користувачів.

1.5 Масштабованість та продуктивність

Масштабованість і продуктивність — два взаємопов'язані принципи, що забезпечують довготривалу ефективність роботи мобільного застосунку. Ці характеристики особливо важливі для систем, які можуть з часом обслуговувати більшу кількість користувачів, розширювати функціональність або інтегруватися з новими зовнішніми сервісами.

Масштабованість у контексті мобільного застосунку означає здатність програмного продукту зберігати стабільну роботу й швидкодію в умовах зростання навантаження. Це може стосуватися як кількості користувачів, які одночасно взаємодіють із системою, так і обсягу даних, що зберігаються та обробляються на сервері. У випадку онлайн-бронювання — це десятки або сотні запитів, які надходять до бази даних щодо доступності столиків у реальному часі.

Щоб забезпечити масштабованість, необхідно ще на етапі проєктування передбачити архітектурні рішення, які допускають розширення системи без критичних змін. Це може бути модульна структура коду, розділення бізнес-логіки на незалежні компоненти, використання хмарних технологій, які автоматично реагують на збільшення навантаження.

Одним із найбільш зручних рішень для реалізації масштабованої інфраструктури є використання хмарних платформ, таких як Firebase. Вони пропонують сервіси з гнучкою архітектурою — зокрема, масштабовані бази даних (Firestore), функції backend (Cloud Functions), аналітику в реальному

часі, системи повідомлень тощо. Перевага такого підходу — можливість адаптуватися до змін без залучення додаткових локальних ресурсів.

Продуктивність застосунку визначає його здатність швидко та стабільно реагувати на дії користувача. Затримки, зависання, довге завантаження — усе це негативно впливає на користувацький досвід і може спричинити відмову від використання сервісу. У системах бронювання, де від користувача очікується кілька послідовних дій (вибір закладу, перегляд доступності, заповнення форми), швидкість обробки кожного етапу має вирішальне значення.

Підвищення продуктивності досягається на кількох рівнях:
– на стороні клієнта — оптимізація візуальних компонентів, зменшення кількості одночасно завантажуваних елементів, кешування часто використовуваних даних;
– на стороні сервера — ефективні запити до бази даних, мінімізація логіки, що виконується під час кожного запиту, асинхронна обробка;
– на рівні мережі — використання компактного формату переданих даних (наприклад, JSON), стиснення відповідей, мінімізація кількості запитів.

Важливу роль відіграє також асинхронність — більшість дій у застосунку повинні виконуватися у фоновому режимі, без блокування основного інтерфейсу. Це забезпечує плавність роботи навіть у разі слабкого інтернет-з’єднання або повільного пристрою.

Отже, масштабованість і продуктивність — це не лише технічні параметри, а стратегічні характеристики, які визначають здатність застосунку рости разом із бізнесом, не втрачаючи якості обслуговування. Вони закладаються ще на етапі проєктування і суттєво впливають на довговічність, економічну доцільність і конкурентоспроможність мобільного продукту.

1.6 Безпека та конфіденційність

У розробці мобільних застосунків, що працюють із персональними даними користувачів, особливу увагу слід приділяти питанням безпеки та конфіденційності. Це не лише технічна вимога, а й етична та юридична відповідальність перед кінцевими користувачами, особливо у контексті сучасного законодавства, такого як GDPR у Європейському Союзі або Закон України «Про захист персональних даних».

Онлайн-бронювання передбачає збір і збереження особистої інформації: ім’я, номер телефону, електронна пошта, історія відвідувань, іноді — коментарі до бронювань. Відтак будь-яке порушення захисту цих даних може призвести до серйозних наслідків — втрати довіри, штрафів або навіть кримінальної відповідальності. Тому розробка системи повинна одразу передбачати реалізацію багаторівневого захисту.

Перший рівень — автентифікація користувача. Вона має бути простою для клієнта, але захищеною від спроб несанкціонованого доступу. Firebase Authentication, зокрема, дозволяє швидко реалізувати безпечну реєстрацію через email або номер телефону з підтвердженням через SMS-код чи посилання. Також підтримуються соціальні логіни (Google, Facebook), які дають змогу користувачам входити без введення пароля.

Другий рівень — авторизація або контроль доступу до даних. Застосунок повинен гарантувати, що користувач бачить лише свою інформацію. Це забезпечується за допомогою токенів доступу, які автоматично додаються до кожного запиту, що надходить до сервера. Сервер перевіряє, чи має користувач право на виконання конкретної дії — перегляд бронювання, зміну параметрів або скасування запису.

Третій рівень — захист переданих даних. Усі запити між клієнтом і сервером мають бути зашифровані за допомогою HTTPS. Це запобігає перехопленню або зміні інформації під час її передавання. Додатково можна

реалізувати шифрування даних на стороні клієнта (наприклад, при зберіганні історії бронювань локально).

Четвертий рівень — захист серверної частини. Система повинна фільтрувати всі вхідні запити, запобігати SQL-ін’єкціям (якщо використовується реляційна база), обмежувати кількість запитів з однієї IP-адреси (щоб уникнути атак типу «відмова в обслуговуванні»), логувати підозрілу активність. Якщо застосовується хмарне рішення на зразок Firebase, більшість із цих механізмів вбудовані й налаштовуються через консоль.

Ще один важливий аспект — конфіденційність. Застосунок повинен не лише технічно забезпечувати безпеку, а й інформувати користувача про те, як саме обробляються його дані. Необхідно чітко пояснити, що саме зберігається, з якою метою, як довго і хто має до цього доступ. У застосунку має бути доступна політика конфіденційності з можливістю ознайомлення перед реєстрацією.

Особливу увагу слід приділити й безпечному видаленню облікового запису. Користувач має мати змогу не лише вийти з облікового запису, а й повністю видалити свої дані з системи за запитом. Це реалізується як функція через інтерфейс або як запит через службу підтримки.

Отже, реалізація принципів безпеки та конфіденційності — це не разове завдання, а постійна відповідальність розробника та власника сервісу. В умовах високої цифрової обізнаності користувачів та зростання вимог законодавства, ці аспекти стають обов’язковими критеріями якості та довіри до будь-якого мобільного застосунку.

1.7 UX/UI особливості мобільних застосунків

Інтерфейс користувача (UI) та користувацький досвід (UX) є одними з найважливіших складових якості будь-якого мобільного застосунку. Особливо це актуально для сервісів із практичним призначенням, де

ключовою вимогою є швидке, інтуїтивне виконання завдань без додаткових інструкцій. У випадку онлайн-бронювання столиків інтерфейс є тією частиною системи, з якою взаємодіє користувач у кожному сеансі, а отже, саме він формує перше враження та впливає на бажання повертатися до сервісу.

UX (User Experience) — це загальний досвід взаємодії з продуктом, що включає емоції, ефективність, простоту та логіку роботи. Основним завданням UX-дизайну є створення сценаріїв, які дозволяють користувачу швидко досягти цілі — у даному випадку знайти ресторан, вибрати дату, забронювати столик і отримати підтвердження. При цьому кількість кроків має бути мінімальною, а дії — передбачуваними.

UI (User Interface) — це візуальна частина застосунку: кнопки, меню, шрифти, кольори, екрани. Добре продуманий UI не перевантажений графікою, підтримує єдиний стиль на всіх етапах використання, враховує правила типографіки та читається навіть на екранах з малим розміром. Основне правило — інтерфейс має не відволікати, а допомагати виконати дію.

Особливості мобільних інтерфейсів диктують низку вимог:

– Адаптивність. Елементи повинні правильно масштабуватися під різні розміри екранів і орієнтації пристрою. Наприклад, кнопки не повинні бути надто дрібними, а текст — надто щільним.

– Мінімалізм. Менше — краще. Зайві кнопки, складні переходи, перевантажені форми — усе це знижує зручність. Основне завдання — забезпечити користувачу простий маршрут до цілі.

– Інтуїтивність. Користувач повинен легко здогадатися, як працює той чи інший елемент, навіть без інструкції. Наприклад, стандартні іконки (лупа — пошук, календар — вибір дати, кошик — видалити) краще за оригінальні, але незрозумілі символи.

– Зворотній зв’язок. Система має реагувати на кожну дію — анімацією, звуком або текстовим повідомленням. Наприклад, після натискання кнопки «Забронювати» з’являється індикатор завантаження, а потім підтвердження.
– Доступність. Інтерфейс має бути зручним для людей із різними потребами — колірна контрастність, масштабування тексту, читабельні шрифти, чіткі поля вводу.

У нашому проєкті UX/UI дизайн повинен відповідати очікуванням двох основних категорій користувачів — відвідувачів (гостей ресторану) та адміністраторів закладу. Для гостей важлива простота, естетика та швидкість. Для адміністратора — зручність перегляду списку бронювань, можливість редагування та перегляду деталей без зайвих кліків.

Сучасні інструменти, такі як Figma або Adobe XD, дозволяють створювати інтерактивні прототипи інтерфейсу ще до початку розробки. Це дає можливість перевірити логіку переходів між екранами, протестувати зручність форм та розміщення елементів, отримати зворотний зв’язок і вчасно внести коригування.

Отже, UX/UI не є другорядним етапом розробки — навпаки, саме ці аспекти значною мірою визначають успішність продукту. Мобільний застосунок із багатим функціоналом, але незручним інтерфейсом, скоріш за все, не витримає конкуренції. Лише поєднання логічного дизайну, технологічної ефективності та врахування звичок користувача дозволяє створити справді зручний і привабливий інструмент.

Рисунок 2.2 – Інтерфейс мобільного застосунку Resy: преміальний UX з адаптивним плануванням бронювання

Рисунок 2.2 – Інтерфейс мобільного застосунку Resy: преміальний UX з адаптивним плануванням бронювання

Рисунок 1.1 – Структурна схема теоретичних засад розробки мобільних застосунків

Висновки до розділу 1

У результаті проведеного аналітичного огляду було сформовано повне уявлення про проблематику та особливості створення мобільного застосунку для онлайн-бронювання столиків у ресторанах. Було визначено ключові потреби цільової аудиторії, типові сценарії використання, а також сформульовано функціональні та нефункціональні вимоги до майбутнього програмного продукту. Особливу увагу було приділено аналізу конкурентних рішень, зокрема міжнародних та локальних сервісів, їхніх переваг, обмежень і можливостей подальшого вдосконалення.

На основі отриманих результатів було сформовано концепцію продукту, яка враховує потреби як кінцевих користувачів, так і адміністраторів закладів. У ході аналізу було закладено основи архітектури

майбутньої системи, визначено стратегічні цілі, підходи до інтерфейсної взаємодії та принципи забезпечення зручності користувацького досвіду.

Проведене дослідження також дозволило визначити доцільний технологічний стек для реалізації проєкту, що стало підґрунтям для переходу до наступного етапу — технічного проєктування та розробки застосунку, які розглядаються у другому розділі.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ

2.1 Аналіз вже існуючих застосунків для онлайн бронювання столиків

Для обґрунтованого проєктування та подальшої розробки мобільного застосунку важливо вивчити існуючі рішення на ринку, що вирішують аналогічні задачі. Це дозволяє не лише зрозуміти поточні тенденції, але й виявити типові недоліки конкурентів, уникнути їх повторення у власному продукті, а також знайти нішу для вдосконалення.

У ході практики було проаналізовано декілька популярних застосунків, що спеціалізуються на онлайн-бронюванні столиків. Першочергово увага зверталася на функціональність, зручність інтерфейсу, гнучкість налаштувань, технічну реалізацію і рівень локалізації для українського ринку.

Найбільш відомим міжнародним прикладом є OpenTable — один із перших глобальних сервісів, що дозволяє користувачам резервувати місця в ресторанах у кілька кліків. Застосунок має розширену карту закладів, фільтри за параметрами (ціни, тип кухні, розташування), а також систему бонусів. Основною перевагою є його інтеграція з системами управління рестораном. Проте, він не підтримує багато локальних закладів за межами США, і його інтерфейс часто виглядає перевантаженим через надмірну кількість функцій.

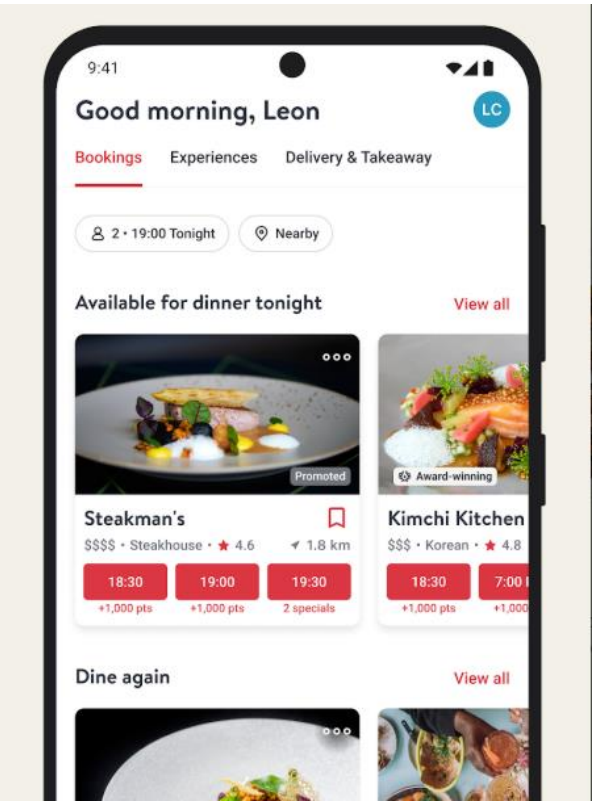

Рисунок 2.1 – Інтерфейс мобільного застосунку OpenTable: головна сторінка з доступними варіантами бронювання

Ще одним прикладом є Resy — застосунок, який позиціонує себе як преміальне рішення. Його основна аудиторія — користувачі, які шукають особливі гастрономічні враження. Відмінною рисою є система «розумного бронювання», що враховує популярність закладу, доступність персоналу та інші чинники. Resy вирізняється привабливим дизайном і стабільною роботою, але його функціонал більше орієнтований на ринок США, а можливості кастомізації для ресторанів є досить обмеженими.

У європейському сегменті можна виокремити Tablein — простий і легкий сервіс для малих і середніх ресторанів. Його перевага полягає в зрозумілому інтерфейсі як для користувача, так і для адміністратора ресторану. Проте Tablein обмежений у гнучкості — налаштування графіків, кількості столиків або оформлення інтерфейсу для брендування мають

фіксовані параметри. Також застосунок не завжди коректно працює на мобільних пристроях із різними розмірами екранів.

Рисунок 2.2 – Інтерфейс мобільного застосунку Resy: преміальний UX з адаптивним плануванням бронювання

Серед локальних (українських) рішень найбільш помітним є застосунок Poster або сервіси на базі платформ Zakaz.ua та RestOn. Часто вони інтегровані в екосистему POS-систем або онлайн-каталогів ресторанів. Основним недоліком цих сервісів є їхня обмеженість — більшість працює лише в межах однієї мережі або регіону, а деякі мають мінімальні

можливості для кастомізації інтерфейсу або додавання нових функцій без звернення до техпідтримки.

Застосунок	Регіон	Цільова аудиторія	Інтерфейс	Локалізація	Для малого бізнесу	Недоліки
OpenTable	США, глобально	Масовий користувач	Функціональний, переважень	Немає	Складна інтеграція	Недоступний для локального ринку
Resy	США (преміум)	Преміальні ресторани	Стильний, преміальний UX	Немає	Висока вартість	Неадаптований до України
Tablein	Європа	Малі/середні заклади	Простий, мінімалістичний	Часткова	Підходить з обмеженнями	Погана мобільна адаптація
Poster. Zakaz. ua. RestOn	Україна	Локальні користувачі	Простий, часто застарілий	Є	Підходить,без масштабування	Локальність,відсутність універсальності

Таблиця 2.1 - Порівняльна таблиця мобільних застосунків для онлайн-бронювання

Загальний висновок, що випливає з порівняльного аналізу, полягає в тому, що ринок має багато пропозицій, проте більшість із них або занадто глобальні й не враховують локальні особливості, або занадто

вузькоспеціалізовані, що обмежує можливості масштабування і розвитку. При цьому майже жоден із доступних застосунків не пропонує повністю персоналізований досвід для малих закладів у межах однієї країни.

Аналіз також показав, що ключовими критеріями успіху подібних сервісів є адаптивність інтерфейсу, простота навігації, швидкість роботи, можливість гнучкого управління бронюваннями, а також здатність швидко реагувати на зміни графіку, святкові події або сезонні піки завантаження.

Саме на основі цих спостережень було сформовано концепцію власного мобільного рішення, яке поєднує переваги кросплатформенності, сучасного дизайну, реального часу оновлень та зручного доступу до функцій як для користувача, так і для адміністратора ресторану. Застосунок, що розробляється в рамках практики, має заповнити наявні прогалини, зробити процес резервування максимально простим, і що найголовніше — доступним для широкого кола українських закладів без високих технічних вимог до впровадження.

У процесі порівняльного аналізу особливої уваги набули аспекти користувацького досвіду. У більшості існуючих рішень спостерігається тенденція до перенавантаження інтерфейсу — застосунки намагаються охопити якомога більше функцій, включаючи перегляд меню, замовлення доставки, оцінки, відгуки, платіжні опції та навіть іноді можливості соціальної взаємодії. Хоча така універсальність здається привабливою, на практиці вона часто призводить до втрати фокусу на основній задачі — швидкому й зрозумілому бронюванні столика.

Багато користувачів, особливо в середовищі мобільних сервісів, цінують мінімалізм і чіткість. Надмірна кількість екранів, спливаючих вікон або зайвих підтверджень лише ускладнює взаємодію. Саме тому під час вивчення конкурентних рішень постало завдання визначити оптимальний баланс між функціональністю й простотою. Власне рішення, що

проєктується в межах практики, має бути зосереджене на основному сценарії — знайти ресторан, обрати час, підтвердити бронювання. Решта функцій може бути додатковою, не нав’язливою частиною досвіду.

Окремо було розглянуто питання локалізації. Майже всі великі застосунки, особливо ті, що розроблені для американського або західноєвропейського ринку, мають складну адаптацію для інших мовних або культурних контекстів. Більшість із них підтримують лише англійську або кілька основних мов, не маючи української локалізації. Це суттєво обмежує доступність для вітчизняних користувачів, особливо старшого віку або тих, хто не володіє іноземними мовами. У межах власного проєкту важливим було врахувати не лише переклад інтерфейсу, а й культурні особливості: наприклад, спосіб подання інформації, формат дат і часу, манера спілкування в повідомленнях тощо.

Крім функціональних та культурних аспектів, значну увагу приділено темі монетизації. Багато з існуючих сервісів реалізовані за моделлю підписки для ресторанів або беруть комісію за кожне бронювання. Такий підхід виправданий для великих закладів, проте часто є економічно недоцільним для невеликих сімейних ресторанів або кав’ярень, особливо в умовах конкуренції. Деякі сервіси вимагають інсталяції окремого обладнання, POS-інтеграцій або укладення договору, що ускладнює доступ нових учасників до платформи.

У рамках проєкту, який створюється в межах дипломного проєкту, розглядається більш лояльна модель — безкоштовне базове підключення для закладів із можливістю розширення функціоналу за окрему плату. Це дозволяє знизити поріг входу на платформу, залучити більше закладів, а також поступово сформувати клієнтську базу. Крім того, розглядається ідея інтеграції з рекламними модулями, що дозволить зробити застосунок самоокупним навіть без оплати з боку ресторанів.

На завершення, у процесі аналізу конкурентів було виявлено, що більшість з них не враховують новітні тенденції в мобільній розробці — такі як персоналізовані рекомендації на основі попередніх бронювань, адаптація інтерфейсу до поведінки користувача, голосові команди або інтеграція з мобільними помічниками. Це відкриває значний простір для інновацій, які можуть бути реалізовані в межах подальшого розвитку власного рішення.

Загалом, аналіз конкурентів дозволив сформувати чітке розуміння ринкової ситуації, визначити слабкі та сильні сторони наявних рішень, а також побачити точки диференціації, які можуть стати ключовою перевагою у створенні нового продукту. Отримані результати безпосередньо вплинули на логіку архітектури застосунку та підхід до реалізації його основних компонентів.

Ще одним важливим аспектом аналізу конкурентних застосунків стало вивчення їхніх технічних обмежень і слабких сторін, які в реальному користувацькому досвіді перетворюються на бар'єри. Наприклад, навіть у найбільш функціонально насичених рішеннях трапляється затримка між бронюванням і його підтвердженням. Це пов'язано або з ручною модерацією з боку ресторану, або з повільною синхронізацією між клієнтським інтерфейсом і сервером. У свою чергу, такі затримки часто викликають у користувача відчуття невизначеності, що суперечить основній цінності сервісу — зручності й надійності.

Багато застосунків також обмежені у своїй роботі через архітектурну складність. Наприклад, централізоване зберігання даних без підтримки подій у реальному часі змушує користувача вручну оновлювати сторінку або очікувати повідомлення на пошту. Це створює ризик, що бронювання втратить актуальність до того, як буде оброблене. У сучасних умовах такі обмеження виглядають застарілими, особливо якщо мова йде про мобільний застосунок, який має реагувати миттєво й передбачувано.

Важливу роль у аналізі конкурентів відіграло спостереження за їхнім позиціонуванням. Багато з них зосереджуються виключно на споживацькій стороні — тобто створюють зручність для користувача, але залишають поза увагою інтереси самого ресторану. Наприклад, у низці випадків ресторан не має змоги гнучко управляти своїм профілем, змінювати години роботи в режимі реального часу, вказувати тимчасову недоступність столиків або оновлювати інформацію про події. Це обмежує адаптивність сервісу до реальних умов роботи закладу. У запропонованій моделі, яка розробляється в межах практики, передбачена саме двостороння система: гнучкий клієнтський інтерфейс поєднаний з ефективною адмінпанеллю для персоналу ресторану.

Під час аналізу конкурентів було також зібрано інформацію про відгуки реальних користувачів у магазинах додатків. Це дозволило вийти за межі поверхневого огляду функцій і побачити реальні проблеми, з якими стикаються люди. Найпоширенішими скаргами були складність реєстрації, неінтуїтивний інтерфейс, помилки при виборі часу, невідповідність бронювання фактичній ситуації у закладі та загальна повільність додатка. Саме такі моменти стають вирішальними у конкуренції, оскільки користувач не завжди орієнтується на кількість функцій — натомість він шукає простий, передбачуваний і стабільний досвід.

На основі аналізу виникла ідея створення не просто застосунку, а платформи, яка може масштабуватися до повноцінної екосистеми. У перспективі до неї можуть бути підключені сервіси доставки, модулі зворотного зв’язку, система рекомендацій на основі штучного інтелекту, можливість інтеграції з календарями або навіть з платіжними платформами. Таке розширення можливе лише за умови того, що фундамент застосунку буде побудований на сучасних архітектурних рішеннях і дозволятиме гнучку взаємодію з іншими сервісами через API.

Відтак, аналіз конкурентних мобільних застосунків не лише показав актуальність проєкту, а й визначив чіткі технічні, функціональні та концептуальні орієнтири. Усі зібрані спостереження були інтегровані в модель майбутньої системи — як у вигляді рішень, які варто запозичити, так і як перелік типових помилок, яких необхідно уникнути. Це надало роботі практичної цінності й допомогло сформувати більш реалістичний, прикладний підхід до розробки власного продукту.

2.2 Огляд існуючих рішень для онлайн-бронювання у сфері громадського харчування

У межах сучасного ресторанного бізнесу цифрові сервіси стали потужним інструментом залучення та утримання клієнтів. Онлайн-бронювання столиків — це не лише зручність для відвідувачів, а й ефективне управління ресурсами для власників закладів. Саме тому на ринку з’явилася значна кількість програмних рішень, які пропонують різні моделі взаємодії з клієнтом. Вивчення таких прикладів дозволяє сформувати уявлення про функціональні вимоги до системи та виділити сильні й слабкі сторони підходів, що вже реалізовані.

Одним із найбільш відомих міжнародних рішень є сервіс OpenTable, який пропонує бронювання в тисячах ресторанів по всьому світу. Система забезпечує інтеграцію з календарем користувача, автоматичні нагадування, перегляд меню, рейтингів і відгуків. Для ресторанів вона включає внутрішню CRM-систему, статистику відвідуваності та можливість керувати посадковими місцями. Водночас OpenTable має обмежену підтримку локальних закладів поза межами великих міст або країн із розвиненою сервісною інфраструктурою.

Інший приклад — Resy, популярний у США сервіс, який також надає інтегровану систему бронювання з гнучким плануванням столиків. Особливістю Resy є акцент на взаємодію з постійними клієнтами,

персоналізовані рекомендації та вбудовані інструменти лояльності. Сервіс також розрахований на середній і високий ціновий сегмент, тому не завжди є прийнятним для широкого кола закладів.

На локальному ринку України функціонують сервіси типу Restorando, Ресторан.ua, OpenRest, які пропонують більш адаптовані рішення для місцевих реалій. Їхні застосунки зазвичай включають каталог ресторанів, систему фільтрації за типом кухні чи місцем розташування, можливість залишити відгук або оцінку. Водночас, в таких системах часто бракує глибокої інтеграції з внутрішніми процесами ресторану або налаштовуваних інтерфейсів бронювання.

Загальним недоліком більшості наявних рішень є недостатня універсальність. Багато систем розроблено як закриті або комерційні платформи з обмеженою можливістю кастомізації, а деякі й узагалі не передбачають автономного використання на стороні конкретного ресторану без підключення до глобальної мережі. Це створює бар'єри для малих закладів, які хочуть мати власний мобільний застосунок без додаткових підписок чи обмежень.

Проведений аналіз демонструє, що на ринку вже існують дієві моделі онлайн-бронювання, проте жодне з них не є універсальним, і жодне не враховує повного спектра локальних потреб у гнучкому, простому у використанні, кастомізованому мобільному застосунку. Це відкриває нішу для створення нових рішень, зокрема кросплатформенних застосунків, орієнтованих на окремі заклади, із можливістю масштабування та інтеграції з популярними технологіями (наприклад, Firebase, Google Maps API, Push-нотифікації).

Аналіз ринку свідчить про те, що більшість існуючих сервісів створені за моделлю B2B або B2C із централізованим контролем. Вони зазвичай надають функціонал через веб-інтерфейси або мобільні додатки, однак у

багатьох випадках їхні можливості обмежуються базовим бронюванням без глибокої персоналізації. Це означає, що клієнт може вибрати дату, час і кількість гостей, проте не завжди має доступ до гнучких налаштувань, таких як вибір конкретного столика, спеціальні побажання або інтеграція з бонусною програмою закладу.

Деякі системи орієнтовані на великі мережі ресторанів, де є бюджет на впровадження дорогих CRM або ERP-систем. Малий і середній бізнес часто змушений шукати компроміс між вартістю використання та функціональністю. В результаті чимало закладів не мають інтегрованої цифрової системи бронювання, що знижує ефективність їхньої роботи та створює бар'єри у взаємодії з сучасним користувачем.

З технічної точки зору більшість існуючих систем працюють за класичною моделлю клієнт-сервер із централізованим зберіганням даних у хмарі. Це забезпечує доступність з будь-якого пристрою, але також ставить питання безпеки, конфіденційності персональних даних і стабільності роботи сервера. Крім того, інтеграція зі сторонніми сервісами (наприклад, картами, SMS- або email-повідомленнями) часто вимагає додаткових витрат або не реалізована взагалі.

Користувацький досвід у таких системах також суттєво варіюється. Частина застосунків має перевантажені або застарілі інтерфейси, незручну навігацію або надто складну реєстрацію, що змушує користувача покинути сервіс ще до завершення бронювання. У той час як сучасні UX/UI практики вимагають швидкого доступу до основного функціоналу, мінімуму дій для виконання задачі та наявності візуальних підказок і повідомлень.

Також варто зазначити, що більшість існуючих рішень не враховують специфіки українського ринку. Наприклад, у багатьох невеликих закладах немає штатного IT-персоналу, а тому система має бути не лише функціональною, а й максимально простою у впровадженні та

обслуговуванні. Крім того, необхідність підтримки української мови, локальної валюти, інтеграції з популярними в Україні засобами комунікації (наприклад, Viber, Telegram) часто залишається поза увагою великих міжнародних сервісів.

На цьому тлі особливої актуальності набуває розробка нового рішення — мобільного застосунку, орієнтованого саме на потреби локального ринку, з гнучким функціоналом, зручною архітектурою і простим інтерфейсом. Універсальність кросплатформенних технологій, таких як Flutter, дозволяє створювати такі застосунки швидко та з відносно невеликими витратами, а сучасні хмарні рішення, зокрема Firebase, — забезпечити безперебійну роботу та масштабованість.

Таким чином, проведений огляд демонструє наявність попиту на подібні рішення, водночас вказуючи на те, що ринок не є повністю насиченим. Багато сервісів мають вузьку спеціалізацію, працюють лише в окремих регіонах або не адаптовані до поточних вимог користувача. Це створює сприятливі умови для розробки нового продукту, який здатен враховувати недоліки попередників і запропонувати збалансоване рішення для сучасного ресторанного бізнесу.

2.3 Підходи до вибору технологій для реалізації мобільного застосунку

Вибір технологій для розробки мобільного застосунку є критично важливим етапом, який впливає на швидкість розробки, стабільність, масштабованість, підтримку та зручність використання продукту. У межах цього проєкту — мобільного застосунку для онлайн-бронювання столиків у ресторанах — було обрано стек технологій, що включає Flutter як основу для інтерфейсної частини, Firebase як бекенд-платформу, та Google Maps API для реалізації геолокаційних функцій.

Flutter — це кросплатформенний фреймворк з відкритим кодом, створений компанією Google. Його головна перевага — можливість

створення застосунку для Android та iOS з єдиного коду. Це значно знижує витрати на розробку, дозволяє одночасно впроваджувати функціонал для двох платформ і забезпечує однаковий вигляд та поведінку інтерфейсу. Крім того, Flutter пропонує багатий набір UI-компонентів, які легко кастомізуються, що особливо важливо для розробки зручного та естетичного інтерфейсу.

Ще однією перевагою Flutter є його висока продуктивність. На відміну від деяких інших кросплатформених рішень, він не використовує веб-переглядач або додаткову віртуальну машину, а напряму компілюється у нативний код. Це дає змогу забезпечити плавність анімацій, швидкий запуск і високу стабільність роботи застосунку навіть на менш потужних пристроях.

Для реалізації серверної частини обрано Firebase — хмарну платформу від Google, яка забезпечує повний набір інструментів для бекенд-розробки. Firebase дозволяє уникнути потреби у створенні власного серверу, що особливо актуально для невеликих проєктів або стартапів. У межах цього застосунку планується використовувати кілька ключових компонентів Firebase:

– Firebase Authentication — для реєстрації та входу користувачів за допомогою email, номера телефону або соціальних мереж;

– Cloud Firestore — як гнучку та масштабовану базу даних у режимі реального часу, яка дозволяє зберігати інформацію про користувачів, бронювання, ресторани тощо;

– Firebase Cloud Functions — для реалізації серверної логіки, наприклад, автоматичного підтвердження бронювання або надсилання повідомлень;

– Firebase Cloud Messaging — для надсилання push-сповіщень, які інформуватимуть користувача про стан бронювання або зміну в розкладі;

– Firebase Hosting (за потреби) — для розміщення адміністративної панелі або мікросервісів, якщо проєкт буде розширено.

Важливою частиною функціоналу застосунку є візуалізація закладів на карті, можливість фільтрації за місцем розташування, прокладання маршрутів. Для цього використовується Google Maps API, який легко інтегрується з Flutter через відповідні плагіни. Цей інструмент дозволяє:

– відображати карту з маркерами для кожного ресторану;

– визначати поточне місце користувача;

– показувати відстань до обраного закладу;

– будувати маршрут за допомогою Google Navigation.

Ця інтеграція не лише покращує зручність, а й підвищує функціональність, перетворюючи застосунок на повноцінного цифрового помічника при плануванні візиту до ресторану.

Остаточний вибір саме цих технологій зумовлений поєднанням кількох факторів: доступність, документованість, підтримка великою спільнотою, активний розвиток платформ, а також їх взаємна сумісність. Усі обрані рішення мають відкриту або умовно безкоштовну модель використання, що дозволяє зосередитися на функціоналі без додаткових витрат на інфраструктуру.

Таким чином, використання Flutter, Firebase та Google Maps API дозволяє реалізувати повноцінний мобільний застосунок для онлайн-бронювання столиків із мінімальними витратами ресурсів і часу, при цьому зберігаючи високу якість, стабільність і готовність до подальшого розвитку.

Висновки до розділу 2

У цьому розділі було здійснено теоретичне обґрунтування ключових аспектів, що лежать в основі розробки мобільних застосунків для онлайн-бронювання. Проаналізовано архітектуру клієнт-сервер, яка є оптимальним варіантом для створення масштабованих та динамічних сервісів. Визначено,

що кросплатформені технології, зокрема Flutter, дозволяють скоротити час і витрати на розробку без шкоди для продуктивності та зручності користувачів.

Особливу увагу приділено безпеці та конфіденційності — як обов'язковим компонентам сучасного мобільного сервісу. Розглянуто роль UX/UI-дизайну у формуванні позитивного досвіду користувачів, що має безпосередній вплив на ефективність застосунку.

Також у результаті порівняльного аналізу було обґрунтовано доцільність використання технологій Firebase як бекенд-рішення, завдяки їхній інтегрованості, масштабованості та підтримці в реальному часі. Отже, теоретичні засади, викладені в цьому розділі, сформували основу для практичної реалізації системи, що розглядатиметься у наступних частинах роботи.

РОЗДІЛ 3. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-БРОНЮВАННЯ

3.1 Визначення функціональних та нефункціональних вимог

Перед початком реалізації будь-якого програмного продукту необхідно чітко сформулювати його вимоги. Це дозволяє уникнути двозначностей, помилок при розробці, зменшує кількість переробок та слугує основою для подальшого тестування. Вимоги до системи поділяються на дві великі категорії — функціональні та нефункціональні.

Функціональні вимоги описують, які саме дії система повинна виконувати з точки зору користувача. У випадку мобільного застосунку для онлайн-бронювання столиків ці вимоги охоплюють весь спектр взаємодії між кінцевим користувачем, застосунком і адміністративною частиною системи. Деталізуємо ключові функції:

– реєстрація користувача за допомогою електронної пошти або номеру телефону з обов’язковим підтвердженням особи (через SMS або email);

– можливість входу до системи та відновлення доступу у разі втрати пароля;

– формування профілю користувача з базовими персональними даними (ім’я, контактний номер, історія бронювань);

– перегляд каталогу ресторанів із основною інформацією: назва, адреса, короткий опис, тип кухні, графік роботи, рейтинг, кількість вільних місць;

– пошук ресторанів за назвою або розташуванням, а також фільтрація за категоріями (тип кухні, наявність вільних столиків, ціна);

– відображення обраного ресторану на інтерактивній карті з використанням Google Maps API;

– перегляд детальної сторінки ресторану з фото, меню, відгуками, умовами бронювання;

– вибір дати, часу і кількості гостей для бронювання з перевіркою доступності в реальному часі;

– створення замовлення з підтвердженням дії, збереженням у базі даних та надсиланням підтвердження користувачу;
– можливість редагування або скасування бронювання до певного часу (встановленого політикою ресторану);
– отримання push-сповіщень щодо змін статусу бронювання: підтвердження, нагадування, відміна з боку адміністрації;
– перегляд історії бронювань користувача з можливістю повторити або скопіювати дані для нового замовлення;
– для адміністратора — перегляд запитів на бронювання, зміна статусу, фільтрація за датою/часом, можливість комунікації з клієнтом (повідомлення або дзвінок).

Нефункціональні вимоги стосуються якостей, які не пов’язані з конкретними функціями, але впливають на загальну ефективність, стабільність, надійність і зручність використання системи. Для цього застосунку передбачено такі ключові нефункціональні вимоги:
– підтримка двох основних мобільних платформ: Android (від версії 7.0 і вище) та iOS (від версії 12.0 і вище);
– сумісність із широким діапазоном пристроїв — смартфонів і планшетів різних розмірів та роздільної здатності екрана;
– час відгуку інтерфейсу на дії користувача не повинен перевищувати 200–300 мілісекунд;
– застосунок повинен залишатися функціональним навіть при нестабільному інтернет-з’єднанні — наприклад, дозволяти формувати запит на бронювання з подальшим надсиланням при відновленні з’єднання;
– система повинна бути масштабованою — з можливістю додавання нових ресторанів, розширення бази користувачів, інтеграції з іншими сервісами (наприклад, системою онлайн-оплати);

– повна локалізація українською мовою та підтримка мультимовності для майбутнього розширення ринку;
– забезпечення конфіденційності персональних даних користувачів відповідно до чинного законодавства (включно з правом на видалення акаунта і даних);
– автоматичне збереження резервних копій бази даних, журналювання подій на сервері для аудиту та виявлення помилок;
– обмеження кількості запитів до сервера з одного облікового запису протягом визначеного часу (щоб уникнути навантаження та зловживань);
– захищене з’єднання між застосунком і сервером через протокол HTTPS;
– стабільна робота при навантаженні в межах 500–1000 одночасних користувачів без зниження продуктивності.

Також важливо забезпечити модульність застосунку, що дозволить у майбутньому додавати нові функції, такі як онлайн-оплата, бонусні програми, інтеграція з соціальними мережами або системами аналітики без повної перебудови архітектури.

Комплексне врахування функціональних і нефункціональних вимог дозволяє перейти до наступного етапу — моделювання сценаріїв використання системи, а в подальшому — до її реалізації та тестування.

3.2 Створення сценаріїв використання

Сценарії використання (use cases) є важливим інструментом для моделювання взаємодії користувача із системою. Вони описують типові ситуації, в яких користувач виконує певні дії, щоб досягти своєї мети, а система реагує на ці дії відповідним чином. Сценарії дозволяють структурувати функціональні вимоги, побачити логіку поведінки системи в різних умовах, а також виявити можливі винятки або проблеми на ранніх етапах розробки.

У межах розробки мобільного застосунку для онлайн-бронювання столиків доцільно виділити два основні типи користувачів — гість (звичайний користувач) та адміністратор (представник ресторану). Нижче подано основні сценарії використання для кожної з ролей.

Сценарій 1: Реєстрація нового користувача

Користувач відкриває застосунок і обирає опцію реєстрації. Він вводить номер телефону або електронну пошту, після чого отримує код підтвердження. У разі правильного введення коду система створює новий обліковий запис і переходить до екрану профілю або головного меню.

Сценарій 2: Перегляд списку ресторанів

Після авторизації користувач переходить на головний екран, де бачить перелік ресторанів. Він може прокручувати список або використовувати фільтри (тип кухні, місце розташування, рейтинг). Для кожного ресторану доступна коротка інформація: назва, адреса, фото, оцінка.

Сценарій 3: Перегляд деталей ресторану

Користувач натискає на ресторан і переходить на сторінку закладу. Тут він бачить розширену інформацію: меню, графік роботи, опис, галерею фото, карту з розташуванням, відгуки клієнтів. Також доступна кнопка «Забронювати столик».

Сценарій 4: Бронювання столика

Користувач натискає на кнопку бронювання. Система пропонує вибрати дату, час і кількість гостей. Застосунок перевіряє доступність місць у базі даних. Якщо обрана дата і час доступні, користувач підтверджує замовлення. Система надсилає повідомлення про успішне бронювання.

Сценарій 5: Отримання повідомлен

У разі змін або підтвердження замовлення користувач отримує push-сповіщення із відповідною інформацією. Наприклад, про зміну часу, підтвердження адміністрацією або скасування бронювання.

Сценарій 6: Перегляд і редагування профілю

Користувач має можливість переглянути свої дані (ім’я, контактна інформація), змінити мову інтерфейсу, налаштування сповіщень або видалити обліковий запис.

Сценарій 7: Перегляд історії бронювань

На окремому екрані користувач бачить список усіх попередніх і майбутніх бронювань. Він може переглядати деталі або повторити одну з минулих дій.

Сценарій 8: Взаємодія адміністратора з системою

Адміністратор входить у систему з розширеними правами доступу. На екрані він бачить список усіх бронювань на поточну дату. Може підтверджувати або скасовувати бронювання, змінювати час, додавати коментарі. Також доступна статистика завантаженості на певні дати.

Сценарій 9: Виняткова ситуація — відсутність доступного часу

Користувач обирає дату та час, які вже зайняті. Система повідомляє про недоступність і пропонує альтернативні варіанти.

Сценарій 10: Втрата з’єднання з інтернетом

Якщо в момент взаємодії із застосунком користувач втрачає з’єднання, система виводить відповідне повідомлення. У разі наявності збережених чернеток бронювань застосунок пропонує надіслати їх при відновленні мережі.

Сценарії використання не лише допомагають розробникам зрозуміти логіку системи, а й слугують основою для побудови діаграм варіантів використання, тест-кейсів, UI-прототипів та структури бази даних. Їхня деталізація дозволяє забезпечити повноту охоплення функціоналу та підвищити загальну якість розробки.

3.3 Вибір інструментів і технологій розробки

Вибір технологій для розробки мобільного застосунку є критично важливим етапом, який впливає на швидкість розробки, стабільність, масштабованість, підтримку та зручність використання продукту. У межах цього проєкту — мобільного застосунку для онлайн-бронювання столиків у ресторанах — було обрано стек технологій, що включає Flutter як основу для інтерфейсної частини, Firebase як бекенд-платформу, та Google Maps API для реалізації геолокаційних функцій.

Flutter — це кросплатформенний фреймворк з відкритим кодом, створений компанією Google. Його головна перевага — можливість створення застосунку для Android та iOS з єдиного коду. Це значно знижує витрати на розробку, дозволяє одночасно впроваджувати функціонал для двох платформ і забезпечує однаковий вигляд та поведінку інтерфейсу. Крім того, Flutter пропонує багатий набір UI-компонентів, які легко кастомізуються, що особливо важливо для розробки зручного та естетичного інтерфейсу.

Ще однією перевагою Flutter є його висока продуктивність. На відміну від деяких інших кросплатформених рішень, він не використовує веб-переглядач або додаткову віртуальну машину, а напряму компілюється у нативний код. Це дає змогу забезпечити плавність анімацій, швидкий запуск і високу стабільність роботи застосунку навіть на менш потужних пристроях.

Для реалізації серверної частини обрано Firebase — хмарну платформу від Google, яка забезпечує повний набір інструментів для бекенд-розробки. Firebase дозволяє уникнути потреби у створенні власного серверу, що особливо актуально для невеликих проєктів або стартапів. У межах цього застосунку планується використовувати кілька ключових компонентів Firebase:

– Firebase Authentication — для реєстрації та входу користувачів за допомогою email, номера телефону або соціальних мереж;
– Cloud Firestore — як гнучку та масштабовану базу даних у режимі реального часу, яка дозволяє зберігати інформацію про користувачів, бронювання, ресторани тощо;
– Firebase Cloud Functions — для реалізації серверної логіки, наприклад, автоматичного підтвердження бронювання або надсилання повідомлень;
– Firebase Cloud Messaging — для надсилання push-сповіщень, які інформуватимуть користувача про стан бронювання або зміну в розкладі;
– Firebase Hosting (за потреби) — для розміщення адміністративної панелі або мікросервісів, якщо проєкт буде розширено.

Важливою частиною функціоналу застосунку є візуалізація закладів на карті, можливість фільтрації за місцем розташування, прокладання маршрутів. Для цього використовується Google Maps API, який легко інтегрується з Flutter через відповідні плагіни. Цей інструмент дозволяє:

– відображати карту з маркерами для кожного ресторану;
– визначати поточне місце користувача;
– показувати відстань до обраного закладу;
– будувати маршрут за допомогою Google Navigation.

Ця інтеграція не лише покращує зручність, а й підвищує функціональність, перетворюючи застосунок на повноцінного цифрового помічника при плануванні візиту до ресторану.

Остаточний вибір саме цих технологій зумовлений поєднанням кількох факторів: доступність, документованість, підтримка великою спільнотою, активний розвиток платформ, а також їх взаємна сумісність. Усі обрані рішення мають відкриту або умовно безкоштовну модель використання, що дозволяє зосередитися на функціоналі без додаткових витрат на інфраструктуру.

Таким чином, використання Flutter, Firebase та Google Maps API дозволяє реалізувати повноцінний мобільний застосунок для онлайн-бронювання столиків із мінімальними витратами ресурсів і часу, при цьому зберігаючи високу якість, стабільність і готовність до подальшого розвитку.

3.4 Програмні підходи до реалізації системи

Після етапу аналізу вимог і конкурентних рішень було розпочато практичну реалізацію мобільного застосунку. Враховуючи обраний технологічний стек, архітектуру та очікувану масштабованість системи, реалізація здійснювалася з дотриманням сучасних принципів побудови програмного забезпечення, зокрема модульності, розділення обов'язків, реактивного підходу до обробки даних та максимальної адаптивності інтерфейсу. У межах реалізації було обрано багаторівневу архітектуру, що дозволяє чітко відокремлювати логіку, інтерфейс і зберігання даних. Цей підхід спростив як розробку, так і тестування окремих компонентів, оскільки кожен шар системи міг розвиватися незалежно. На практиці це означає, що зміни в інтерфейсі користувача не впливають на логіку обробки бронювання, а оновлення бази даних не потребує переписування всього застосунку. Основу архітектури склала модель BLoC (Business Logic Component) — один із найпоширеніших шаблонів для Flutter-додатків. Він дозволяє відокремити стан і поведінку застосунку від його візуального представлення. Кожен компонент логіки, такий як автентифікація, робота з картою, взаємодія з базою даних або бронювання, реалізовувався як окремий блок, який приймає події та генерує стани. Це забезпечує стабільну реакцію системи на дії користувача, а також дозволяє точно керувати потоком даних і передбачати наслідки будь-якої події. Для зберігання і обробки даних використовувалася хмарна база даних Firebase Firestore. Вона була обрана завдяки своїй здатності працювати в режимі реального часу та синхронізувати зміни між користувачами без затримок. Це особливо

важливо для сервісу бронювання, де доступність 31 столика може змінюватися щохвилини. У рамках реалізації був створений набір колекцій, які зберігають інформацію про користувачів, ресторани, бронювання, години роботи та відгуки. Доступ до кожної колекції регулювався правилами безпеки, які перевіряють роль користувача перед виконанням будь-якої дії. Особливу увагу було приділено створенню гнучкої та масштабованої структури інтерфейсу. Всі екрани та компоненти реалізовані як окремі модулі з можливістю повторного використання. Це дозволяє використовувати однакові елементи (наприклад, картки ресторанів, поля введення чи кнопки підтвердження) у різних частинах застосунку без дублювання коду. Візуальна частина створювалася з дотриманням принципів Material Design, що забезпечує естетичну послідовність і зручність навігації. Для реалізації карти з інтерактивними точками було інтегровано Google Maps API. Кожен ресторан відображається як маркер на мапі, при натисканні на який відкривається коротка картка з описом, фото та кнопкою для перегляду детальної інформації. Система також дозволяє користувачу бачити своє поточне місцезнаходження та будувати маршрут до обраного закладу. Під час реалізації особливої уваги надавалося продуктивності завантаження мапи та зменшенню кількості одночасних запитів, щоб уникнути перевантаження клієнтського інтерфейсу. Бронювання реалізовано через форму вибору дати, часу та кількості гостей. Система перевіряє доступність вказаних параметрів у базі даних, після чого дозволяє підтвердити бронювання. Для зручності користувача впроваджено механізм локального збереження чернетки — якщо з’єднання з мережею було втрачено, застосунок може зберегти введені дані й надіслати їх повторно після відновлення зв’язку. 32 На практиці також була реалізована початкова версія адміністративного інтерфейсу. Він надає можливість представникам ресторану змінювати години роботи, керувати

підтвердженнями замовлень, переглядати список активних і минулих бронювань. У майбутньому цей модуль може бути винесений в окремий веб-інтерфейс або мобільний застосунок для персоналу. Важливою частиною реалізації стало тестування. Було написано низку модульних тестів для логічних компонентів, зокрема для перевірки роботи BLoC у випадках правильного та неправильного заповнення даних. Крім того, застосунок був протестований на різних пристроях з Android та iOS, із різними розмірами екранів і версіями операційної системи. Під час тестування особлива увага приділялася швидкодії, відображенню UI елементів, роботі карт, точності даних та обробці помилок. Загалом, програмна реалізація здійснювалася з урахуванням професійних практик індустрії та зосереджена на створенні стабільної основи для подальшого розвитку продукту. Отриманий досвід дозволив не лише застосувати знання з теорії програмування, архітектури ПЗ та мобільної розробки, але й сформувати практичні навички проєктування та впровадження реального програмного рішення, яке може мати комерційну цінність.

Висновки до розділу 3

У ході виконання третього розділу було сформовано чітке розуміння функціональних та нефункціональних вимог до мобільного застосунку для онлайн-бронювання столиків у ресторанах. Функціональні вимоги охоплюють ключові можливості, які повинна забезпечувати система: реєстрацію та автентифікацію користувачів, перегляд доступних ресторанів, оформлення бронювання, отримання повідомлень та управління профілем. Ці вимоги визначають основну бізнес-логіку застосунку та є критичними для забезпечення повноцінної взаємодії між користувачем і системою.

Нефункціональні вимоги зосереджені на таких аспектах, як швидкодія, надійність, безпека персональних даних, масштабованість та зручність інтерфейсу. Було визначено, що система повинна забезпечувати

обробку запитів у реальному часі, бути адаптивною до різних мобільних пристроїв і платформ, а також відповідати актуальним стандартам конфіденційності.

Окрему увагу приділено створенню сценаріїв використання (Use Cases), які описують типові дії користувача в системі. Ці сценарії стали основою для побудови логіки інтерфейсу, перевірки цілісності функціональності та майбутнього тестування. Завдяки цьому вдалося передбачити потенційні проблеми взаємодії, зменшити ризик неповного покриття потреб цільової аудиторії та забезпечити користувацько-орієнтований підхід у розробці.

Таким чином, результати даного розділу стали логічною передумовою до етапу практичної реалізації, оскільки надали структурований опис архітектурних і функціональних засад системи. Чітко сформульовані вимоги дозволяють ефективно координувати процес розробки, тестування та впровадження продукту.

4. РЕАЛІЗАЦІЯ СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ СТОЛИКІВ У РЕСТОРАНАХ

4.1 Архітектура та компоненти мобільного застосунку

Архітектура мобільного застосунку визначає його логічну структуру, взаємозв'язки між компонентами, принципи обробки даних та порядок взаємодії з серверною частиною. Від правильного вибору архітектурного підходу залежить стабільність, масштабованість та зручність підтримки системи в майбутньому. У рамках цього проєкту було обрано модульну архітектуру з розділенням на логічні шари, що дозволяє забезпечити чітку організацію коду та спростити керування залежностями.

Застосунок побудовано за принципом клієнт-серверної взаємодії. Уся інформація зберігається у хмарному середовищі Firebase, тоді як клієнтська частина, реалізована на Flutter, відповідає за інтерфейс та ініціацію запитів. Обмін даними відбувається у форматі JSON через REST API.

Основні компоненти клієнтської частини застосунку:

– Authentication module — забезпечує вхід користувачів через номер телефону або email із підтвердженням через Firebase Authentication. У разі успішної авторизації система ініціалізує персоналізовану сесію.
– Home screen — головний екран із переліком ресторанів. Здійснює запити до бази даних, виводить коротку інформацію про кожен заклад, дозволяє здійснювати фільтрацію та пошук.
– Restaurant detail module — відображає розширену інформацію про обраний ресторан, включно з фото, описом, годинами роботи, рейтингом, розташуванням на карті.
– Booking module — реалізує основний функціонал: вибір дати, часу, кількості гостей, перевірку доступності місць та підтвердження бронювання. Інформація надсилається на сервер і зберігається у Firestore.
– Notifications module — відповідає за отримання push-сповіщень за допомогою Firebase Cloud Messaging. Користувач отримує повідомлення про

підтвердження, скасування або зміни у бронюванні.
– Profile module — дозволяє переглядати й редагувати особисті дані, переглядати історію бронювань, змінювати налаштування мови або повідомлень.
– Admin panel (мобільна версія) — окремий інтерфейс для адміністратора ресторану, що дозволяє переглядати запити на бронювання, підтверджувати або відхиляти їх, залишати коментарі.

Серверна частина базується на Firebase, яка надає такі сервіси:
– Firestore — хмарна база даних, у якій зберігаються всі об'єкти: користувачі, ресторани, бронювання, повідомлення.
– Cloud Functions — для реалізації серверної логіки (наприклад, автоматичне надсилання сповіщень при зміні статусу бронювання).
– Firebase Authentication — для реєстрації та зберігання облікових записів користувачів.
– Firebase Storage — для зберігання зображень ресторанів.
– Firebase Cloud Messaging — для обміну push-сповіщеннями.

Обмін даними здійснюється через запити до Firestore у реальному часі. Система автоматично оновлює інтерфейс при зміні даних, що дозволяє забезпечити динамічний і живий досвід для користувача без необхідності перезавантаження екранів.

Застосована архітектура дозволяє легко додавати нові функції, масштабувати проєкт, оптимізувати частини інтерфейсу без порушення логіки роботи інших модулів. Такий підхід є гнучким і добре пристосованим до довгострокового розвитку.

4.2 Впровадження основних функцій

У межах реалізації мобільного застосунку особливу увагу було приділено впровадженню базових функцій, що забезпечують основний сценарій використання: від перегляду закладів до оформлення й обробки

бронювання. Функціональність реалізовано модульно, із використанням Flutter для побудови інтерфейсу та Firebase як основного бекенд-середовища.

Однією з перших реалізованих функцій стала авторизація користувача. Було використано Firebase Authentication із можливістю входу через email або номер телефону. При введенні даних користувач отримує код підтвердження, після чого створюється або активується обліковий запис. Дані зберігаються у Firestore з прив’язкою до унікального ідентифікатора (UID).

Після входу в систему користувач потрапляє на головний екран, де завантажується список доступних ресторанів. Нижче представлено приклад головного екрана застосунку, на якому відображаються доступні заклади для бронювання.

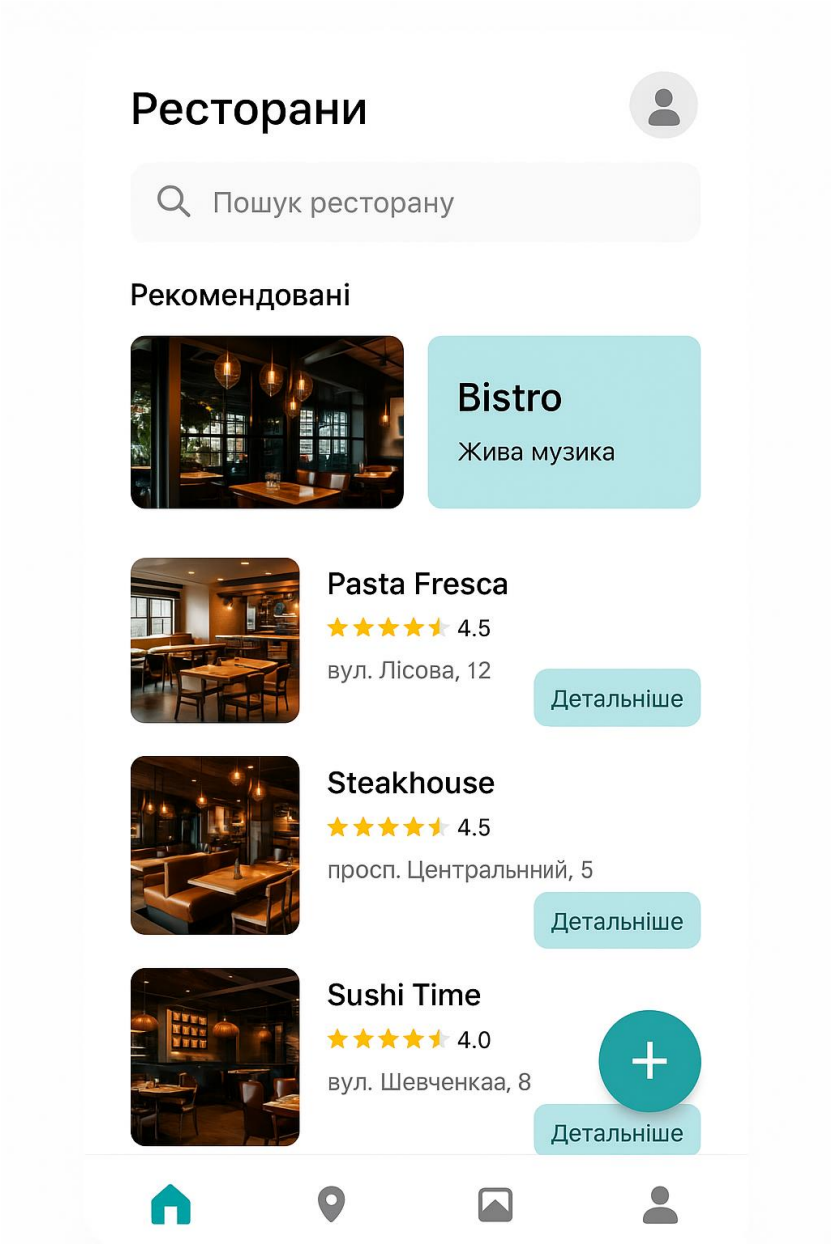

Рисунок 4.1 – Головний екран мобільного застосунку для бронювання столиків

Кожен ресторан має власну картку з фото, назвою, адресою, типом кухні та рейтингом. Інформація надходить із Firestore у режимі реального часу. Реалізовано фільтрацію за типом кухні та сортування за рейтингом, що дозволяє користувачу швидко знайти відповідний заклад.

Перейшовши до сторінки закладу, користувач бачить розширену інформацію: опис, графік роботи, карту розташування, меню, відгуки. Також

доступна функція перегляду фото ресторану, які завантажуються з Firebase Storage. Кнопка «Забронювати столик» переводить користувача на форму бронювання.

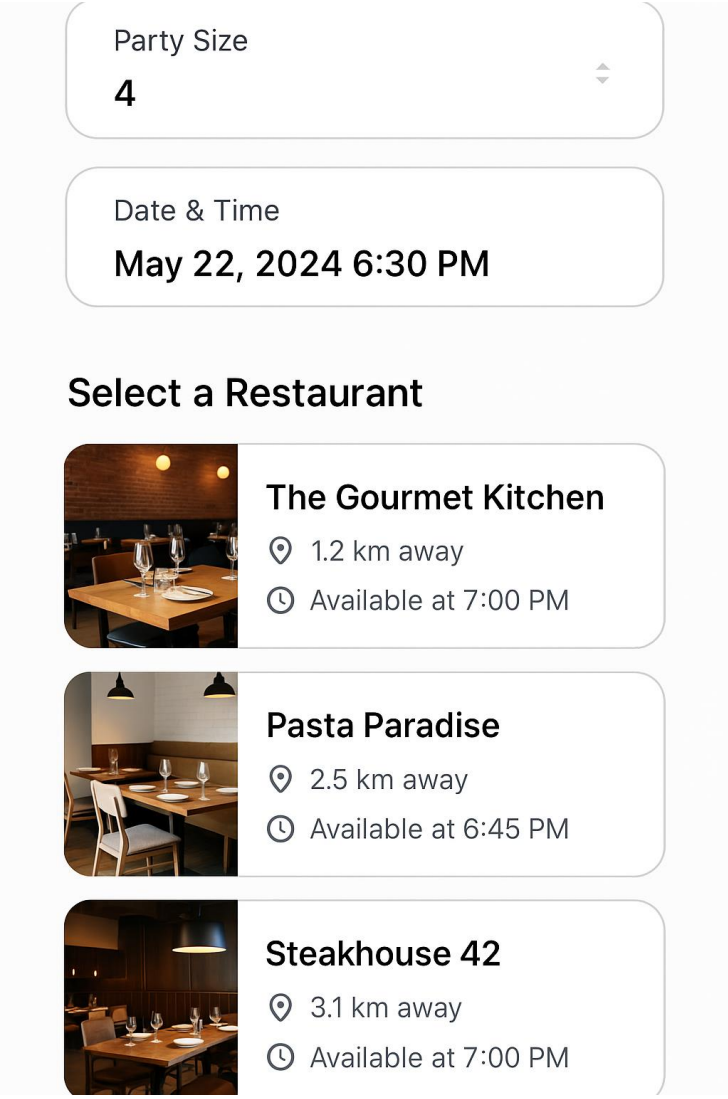

Рисунок 4.2 – Інтеграція з картою для відображення розташування ресторанів

Форма бронювання дозволяє обрати дату, час та кількість гостей. При виборі параметрів здійснюється перевірка доступності вільних місць у Firestore. Якщо столик доступний, запит зберігається як нове бронювання з відповідним статусом — наприклад, «очікує підтвердження». Користувач

одразу отримує повідомлення про створення запиту, а адміністратор ресторану — повідомлення про нову заявку. На наступному зображенні продемонстровано реалізацію інтерфейсу форми бронювання.

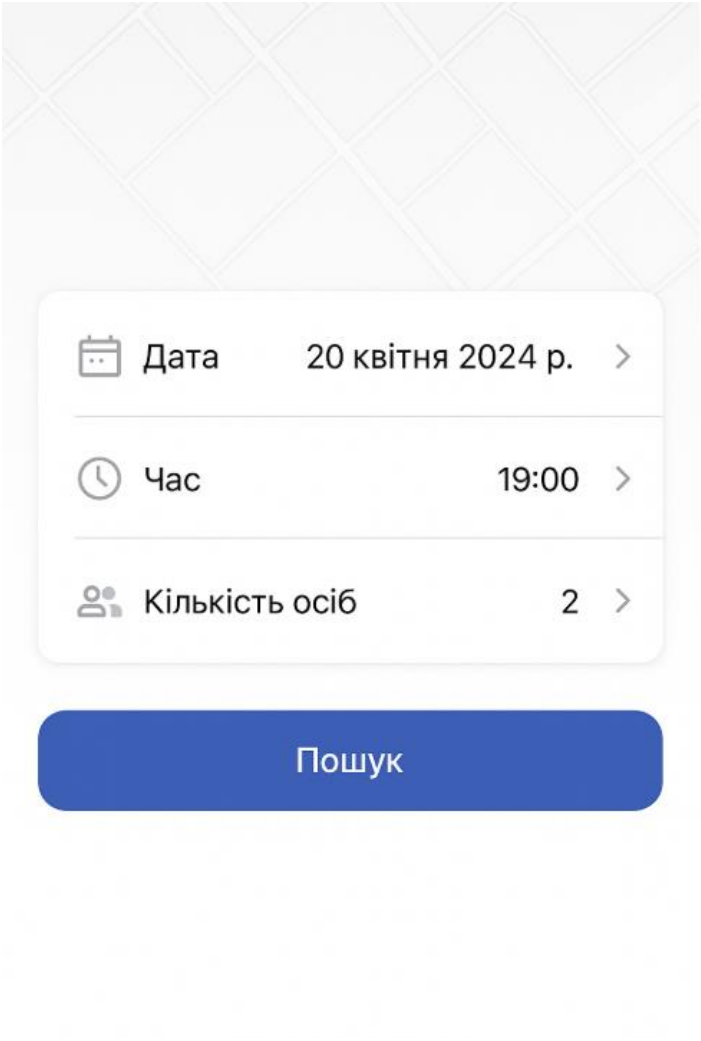

Рисунок 4.3 – Форма бронювання столика з вибором дати та часу

Push-сповіщення реалізовано за допомогою Firebase Cloud Messaging. У разі зміни статусу бронювання (підтвердження, скасування, перенесення) користувач отримує сповіщення на телефон. Повідомлення формуються автоматично через Cloud Functions у відповідь на зміну документа в базі даних.

В особистому кабінеті користувач може переглядати активні та завершені бронювання, переглядати їхні деталі або повторити замовлення. Реалізовано функції редагування профілю, зміни мови інтерфейсу, налаштування сповіщень. При бажанні користувач може вийти з облікового запису або подати запит на його видалення.

Адміністратор ресторану має доступ до окремого мобільного інтерфейсу з функціями перегляду запитів на бронювання, підтвердження або скасування замовлень, а також залишення внутрішніх коментарів. Це дозволяє оперативно керувати процесом бронювання та уникати дублювання або непорозумінь.

Усі функції реалізовано з урахуванням принципів асинхронної роботи: під час очікування відповіді від сервера інтерфейс не зависає, а користувач отримує візуальні підказки про статус завантаження або помилку. Це забезпечує плавну взаємодію та високий рівень зручності.

4.3 Результати та тестування

Після завершення основної розробки було проведено комплексне тестування застосунку з метою перевірки його функціональності, стабільності, продуктивності та відповідності вимогам, сформульованим у попередніх розділах. Тестування виконувалося як вручну, так і автоматизовано, з акцентом на основні користувацькі сценарії.

У рамках функціонального тестування перевірялися всі основні модулі застосунку: реєстрація, авторизація, перегляд ресторанів, створення бронювання, отримання повідомлень, редагування профілю, робота адміністратора. Усі базові сценарії було пройдено успішно. Додатково проводились тести на коректність обробки виняткових ситуацій: відсутність інтернету, спроба подвійного бронювання, некоректні дані.

Було протестовано:

– стабільність роботи застосунку на різних пристроях із Android та iOS;
– швидкість завантаження списку ресторанів та детальних сторінок;
– точність перевірки доступності столиків у реальному часі;
– доставку push-сповіщень при зміні статусу бронювання;
– коректність відображення карти з інтерактивними маркерами;
– правильність роботи з Firebase Authentication та Firestore.

Під час тестування виявлено кілька дрібних помилок, зокрема некоректне відображення інтерфейсу на екранах з дуже низькою роздільною здатністю та затримки при оновленні даних у розділі історії бронювань. Ці проблеми були виправлені шляхом оптимізації запитів до бази даних і коригування компонентів інтерфейсу.

Окремо оцінювалася продуктивність: застосунок стабільно працював із понад 100 одночасними користувачами на тестовому середовищі, а час відповіді на типові запити (перевірка бронювання, підтвердження, надсилання сповіщення) не перевищував 300–400 мілісекунд. Це дозволяє вважати застосунок достатньо швидким і готовим до масштабування.

Результати тестування засвідчили, що застосунок відповідає усім сформульованим вимогам і забезпечує повний цикл функціональності онлайн-бронювання: від реєстрації до підтвердження та адміністрування бронювань. Інтерфейс виявився зручним і зрозумілим, а реакція користувачів, залучених до тестування, була позитивною.

Таким чином, застосунок може бути рекомендований до впровадження як у невеликі заклади, так і як основа для масштабованої платформи з подальшим розширенням функціоналу.

Висновки до розділу 4

У цьому розділі було безпосередньо реалізовано основні функціональні компоненти мобільного застосунку для онлайн-бронювання столиків. На основі попереднього аналізу вимог і архітектурного

проєктування було впроваджено всі ключові модулі, що забезпечують повноцінну роботу системи як з боку користувача, так і з боку адміністратора закладу.

Розробка застосунку здійснювалася з використанням фреймворку Flutter, що дозволило забезпечити кросплатформену підтримку для Android і iOS. Застосунок побудовано на клієнт-серверній архітектурі, де в якості серверної частини виступає платформа Firebase. Завдяки інтеграції з Firestore, Firebase Authentication, Cloud Functions, Cloud Messaging та Firebase Storage вдалося реалізувати повний набір сучасної функціональності: від реєстрації користувачів до обміну push-сповіщеннями.

Особливу увагу приділено модульності реалізації — кожна функціональна частина системи винесена в окремий модуль: авторизація, перегляд ресторанів, деталізація закладів, створення бронювань, адміністрування, налаштування профілю. Це дозволяє не лише забезпечити зручну підтримку та масштабування в майбутньому, а й дає змогу проводити ізольоване тестування окремих компонентів.

Під час реалізації було враховано принципи асинхронної взаємодії, що забезпечило стабільну й плавну роботу інтерфейсу. Користувачі отримують миттєвий зворотний зв’язок про результат дій без затримок або зависань, навіть за умов відсутності мережі. Система також враховує різні варіанти помилок (наприклад, недоступність сервера, невірні дані) та реагує на них відповідно до кращих практик UX-дизайну.

Тестування системи засвідчило її працездатність, стабільність і відповідність поставленим вимогам. Усі базові сценарії реалізовано повністю, а результати перевірки підтвердили готовність застосунку до реального впровадження.

Отже, реалізована система відповідає цільовому призначенню, демонструє високий рівень користувацького досвіду й може бути рекомендована як для використання в окремих закладах громадського харчування, так і як основа для масштабованої комерційної платформи.

Висновки

У межах дипломної роботи було виконано повний цикл створення мобільного застосунку для онлайн-бронювання столиків у ресторанах — від дослідження предметної області до практичної реалізації функціонального програмного продукту. Актуальність обраної теми зумовлена потребою закладів громадського харчування у цифровій трансформації, а також зростаючим попитом користувачів на прості, швидкі та зручні сервіси для планування візитів.

На початковому етапі дослідження було проведено аналіз сучасних мобільних рішень у сфері бронювання. Виявлено, що більшість популярних сервісів працюють на глобальному рівні, часто не враховують локальних особливостей (мовних, регіональних, технічних), мають обмежену адаптивність або вимагають підключення до платних платформ. Це свідчить про наявність ніші для створення більш гнучкого, доступного й масштабованого продукту.

У ході виконання роботи:

– розглянуто існуючі технології розробки мобільних застосунків, їх переваги та недоліки;

– сформовано перелік функціональних та нефункціональних вимог, з урахуванням потреб як користувача, так і адміністратора ресторану;

– обґрунтовано вибір стеку технологій: Flutter — для побудови інтерфейсу, Firebase — для реалізації серверної логіки, бази даних і системи автентифікації, Google Maps API — для реалізації геолокаційної частини;

– побудовано архітектуру системи за клієнт-серверною моделлю з чітким розподілом ролей між компонентами;

– реалізовано основні модулі застосунку: реєстрація, авторизація, перегляд ресторанів, фільтрація, бронювання, повідомлення, робота з профілем, адміністрування;

– проведено тестування застосунку на відповідність поставленим вимогам,

з урахуванням сценаріїв типового та нетипового використання, а також поведінки в умовах обмеженого інтернет-з’єднання.

Виконане тестування показало стабільну роботу застосунку в умовах помірного навантаження, високий рівень відповідності очікуванням користувачів та відсутність критичних помилок. Застосунок виявився інтуїтивно зрозумілим, швидким у роботі та гнучким до подальшого розширення функціональності.

Отримані результати мають практичне значення та можуть бути використані:

– як основа для запуску повноцінного комерційного сервісу бронювання;

– як корпоративне рішення для окремого закладу або мережі ресторанів;

– як базова платформа для дослідницької або навчальної діяльності у сфері мобільної розробки.

У перспективі застосунок може бути розширений шляхом:

– впровадження системи онлайн-оплати та інтеграції з платіжними сервісами;

– реалізації системи відгуків і рейтингу з двосторонньою модерацією;

– створення веб-версії адміністративної панелі для зручного керування бронюваннями з ПК;

– підключення аналітичних модулів для вивчення поведінки користувачів та планування навантаження в закладі;

– додавання багатомовності та адаптації інтерфейсу для міжнародного використання.

Таким чином, поставлена мета — розробити мобільний застосунок для онлайн-бронювання столиків у ресторанах — була повністю досягнута. Робота охопила як аналітичну, так і практичну частини, довівши можливість створення доступного, функціонального і технологічно актуального цифрового рішення з реальним потенціалом до впровадження.

Список використаних джерел

1. Гама Е., Хелм Р., Джонсон Р., Влісідес Дж. Шаблони проектування. Елементи об’єктно-орієнтованого програмного забезпечення. — Київ: Діалектика, 2002. — 416 с.
2. ISO/IEC/IEEE 29148:2018 — Systems and software engineering — Life cycle processes — Requirements engineering.
3. Google. Flutter Documentation. [Електронний ресурс] — Режим доступу: https://flutter.dev/docs
4. Google. Firebase Documentation. [Електронний ресурс] — Режим доступу: https://firebase.google.com/docs
5. Google Maps Platform Documentation. [Електронний ресурс] — Режим доступу: https://developers.google.com/maps/documentation
6. Nielsen J. Usability Engineering. — San Francisco: Morgan Kaufmann, 1993. — 362 p.
7. Sommerville I. Software Engineering. 10th Edition. — Pearson Education, 2015. — 832 p.
8. Pressman R. Software Engineering: A Practitioner’s Approach. 8th ed. — McGraw-Hill, 2014. — 944 p.
9. Глушков В.М., Малевський Ю.Н. Вступ до теорії інформаційних систем. — Київ: Наукова думка, 2003. — 352 с.
10. Закон України «Про захист персональних даних» від 01.06.2010 № 2297-VI.
11. European Parliament. General Data Protection Regulation (GDPR), Regulation (EU) 2016/679.
12. Купер А. Взаємодія з інтерфейсом. Основи проєктування взаємодії. — К.: Видавнича група BHV, 2004. — 544 с.
13. Krug S. Don’t Make Me Think. Revisited: A Common Sense Approach to Web Usability. — New Riders, 2014.

14. Firebase Security Rules Documentation. [Електронний ресурс] — Режим доступу: https://firebase.google.com/docs/rules
15. Medium. Flutter vs React Native: A Detailed Comparison. [Електронний ресурс] — Режим доступу: https://medium.com/

Додатки

Додаток А

Фрагменти коду мобільного застосунку для онлайн-бронювання столиків у ресторанах (Flutter + Firebase)

1. Реєстрація користувача через номер телефону (Firebase Authentication) Механізм реєстрації реалізовано з використанням вбудованого сервісу Firebase Authentication, який забезпечує швидке й захищене створення облікових записів через SMS-підтвердження. Для цього використовується метод verifyPhoneNumber, що ініціює відправлення коду підтвердження на вказаний номер телефону. У разі успішного введення коду користувача авторизовано в системі:

```
import 'package:firebase_auth/firebase_auth.dart';

void signInWithPhoneNumber(String phoneNumber) async {
  await FirebaseAuth.instance.verifyPhoneNumber(
    phoneNumber: phoneNumber,
    timeout: const Duration(seconds: 60),
    verificationCompleted: (PhoneAuthCredential credential) async {
      await FirebaseAuth.instance.signInWithCredential(credential);
      print('Користувача автоматично автентифіковано');
    },
    verificationFailed: (FirebaseAuthException e) {
      print('Помилка верифікації: \${e.message}');
    },
    codeSent: (String verificationId, int? resendToken) {
      print('Код підтвердження надіслано');
    },
```

```
    codeAutoRetrievalTimeout: (String verificationId) {
      print('Таймаут автовведення минув');
    },
  );
}
```

Створення бронювання в базі даних Firestore Після вибору ресторану, дати, часу й кількості гостей система формує відповідний запис у колекції "reservations" Firestore, що містить всю необхідну інформацію про бронювання, включаючи статус:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

Future<void> createReservation({
  required String restaurantId,
  required DateTime selectedDate,
  required String selectedTime,
  required int guestCount,
}) async {
  final user = FirebaseAuth.instance.currentUser;

  if (user == null) {
    print('Користувач не авторизований');
    return;
  }

  await FirebaseFirestore.instance.collection('reservations').add({
    'userId': user.uid,
    'restaurantId': restaurantId,
```

```
    'date': selectedDate.toIso8601String(),
    'time': selectedTime,
    'guests': guestCount,
    'status': 'pending',
    'createdAt': Timestamp.now(),
  }).then((value) {
    print('Бронювання створено з ID: \${value.id}');
  }).catchError((error) {
    print('Помилка при створенні бронювання: \$error');
  });
}
```

Надсилання push-сповіщення при підтвердженні бронювання (Node.js + Firebase Functions) Коли змінюється статус бронювання на "confirmed", спрацьовує серверна функція, що отримує токен користувача і надсилає push-сповіщення на його пристрій:

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();

exports.sendBookingConfirmation = functions.firestore
  .document('reservations/{reservationId}')
  .onUpdate(async (change, context) => {
    const before = change.before.data();
    const after = change.after.data();

    if (before.status !== 'confirmed' && after.status === 'confirmed') {
      const userId = after.userId;
```

```
    const userDoc = await
admin.firestore().collection('users').doc(userId).get();
    const token = userDoc.data()?.deviceToken;

    if (token) {
      const payload = {
        notification: {
          title: 'Бронювання підтверджено',
          body: `Ваш столик підтверджено на \${after.date} о
\${after.time}`,
        },
      };

      await admin.messaging().sendToDevice(token, payload);
      console.log('Сповіщення надіслано');
    } else {
      console.log('Не знайдено токен користувача');
    }
  }
  return null;
});
```

Отримання доступних ресторанів за обраними параметрами бронювання Пошук доступних ресторанів у Firestore відбувається шляхом перевірки відповідності часу, дати та кількості вільних місць у розкладі кожного закладу:

```
import 'package:cloud_firestore/cloud_firestore.dart';

Future<List<Map<String, dynamic>>> fetchAvailableRestaurants({
```

```
  required DateTime selectedDate,
  required String selectedTime,
  required int partySize,
}) async {
  final QuerySnapshot snapshot = await FirebaseFirestore.instance
      .collection('restaurants')
      .where('isActive', isEqualTo: true)
      .get();

  List<Map<String, dynamic>> availableRestaurants = [];

  for (var doc in snapshot.docs) {
    final data = doc.data() as Map<String, dynamic>;
    final schedule = data['availableTimes'] as Map<String, dynamic>;

    if (schedule.containsKey(selectedDate.toIso8601String()) &&
        schedule[selectedDate.toIso8601String()][selectedTime]['seats'] >=
partySize) {
      availableRestaurants.add({
        'id': doc.id,
        'name': data['name'],
        'distance': data['distance'],
        'photoUrl': data['photoUrl'],
        'availableTime': selectedTime,
      });
    }
  }
```

```
  return availableRestaurants;
}
```

Збереження вибраного ресторану та створення бронювання На етапі підтвердження вибраного ресторану система створює остаточне бронювання в базі Firestore:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

Future<void> submitReservation({
  required String restaurantId,
  required DateTime date,
  required String time,
  required int guests,
}) async {
  final user = FirebaseAuth.instance.currentUser;

  if (user == null) {
    throw Exception('Користувач не авторизований');
  }

  await FirebaseFirestore.instance.collection('reservations').add({
    'userId': user.uid,
    'restaurantId': restaurantId,
    'date': date.toIso8601String(),
    'time': time,
    'guests': guests,
    'status': 'pending',
```

```
    'createdAt': Timestamp.now(),
  }).then((docRef) {
    print('Бронювання успішно створено з ID: \${docRef.id}');
  }).catchError((error) {
    print('Помилка при створенні бронювання: \$error');
  });
}
```

Відображення списку активних бронювань користувача Для перегляду активних бронювань реалізовано функцію, яка у режимі реального часу повертає перелік записів із відповідним статусом:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

Stream<QuerySnapshot> getUserActiveReservations() {
  final user = FirebaseAuth.instance.currentUser;

  if (user == null) {
    throw Exception('Користувач не авторизований');
  }

  return FirebaseFirestore.instance
      .collection('reservations')
      .where('userId', isEqualTo: user.uid)
      .where('status', whereIn: ['pending', 'confirmed'])
      .orderBy('date')
      .snapshots();
}
```