

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ
ЗАМОВЛЕННЯМИ У ЗАКЛАДАХ ОБСЛУГОВУВАННЯ НА ОСНОВІ
FLUTTER ТА FIREBASE**

**DEVELOPMENT OF A MOBILE APPLICATION FOR MANAGING
ORDERS IN SERVICE ESTABLISHMENTS BASED ON FLUTTER AND
FIREBASE**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
Групи ІПЗ-42
Голюк Денис Олегович

Керівник:
Самчук Людмила Михайлівна

Кваліфікаційну роботу
допущено до захисту
«__» _____ 2026 р.
к.т.н., доцент
Гарант освітньої програми:
К.т.н., доцент
Ліщина Наталія Миколаївна

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти: бакалавр

Галузь знань: 12 Інформаційні технології

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Голюку Денису Олеговичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: «Розробка мобільного застосунку для управління замовленнями у закладах обслуговування на основі Flutter та Firebase»

Керівник роботи: к.т.н., доцент, Самчук Людмила Михайлівна

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи «29» травня 2026р.

3. Вихідні дані до роботи: предметна область автоматизації процесів управління замовленнями у закладах обслуговування, сучасні технології розробки мобільних застосунків (Flutter, Dart), хмарні сервіси та інструменти роботи з даними (Firebase Firestore, Firebase Authentication), методи проектування програмного забезпечення (UML), підходи до реалізації клієнт-серверної архітектури, вимоги до інформаційних систем, а також принципи забезпечення безпеки та цілісності даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.

5. Перелік графічного матеріалу: 13 рисунків, лістингів 3, таблиць 1

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Самчук Л. М.</i>		
Специфікації вимог до розробленої системи	<i>Самчук Л. М.</i>		
Розробка об'єкта проектування	<i>Самчук Л. М.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	_____ %		
Академічна доброчесність	<i>Самчук Л. М.</i>		

7. Дата видачі завдання «03» лютого 2026р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Денис ГОЛЮК

Керівник кваліфікаційної роботи _____

Людмила САМЧУК

АНОТАЦІЯ

Голюк Д. О. Розробка мобільного застосунку для управління замовленнями у закладах обслуговування на основі Flutter та Firebase. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено аналіз предметної області, пов'язаної з автоматизацією процесу обслуговування клієнтів у закладах харчування, розглянуто існуючі програмні рішення та визначено їхні переваги й недоліки, а також обґрунтовано доцільність розробки власного мобільного застосунку. У другому розділі сформовано функціональні та нефункціональні вимоги до системи, виконано проектування програмного забезпечення із застосуванням UML-нотації, а також обґрунтовано вибір технологій, методів і засобів реалізації. У третьому розділі описано процес практичної реалізації застосунку, розроблено структуру бази даних Firebase Firestore, реалізовано механізми автентифікації та обробки замовлень, а також наведено результати тестування, оцінки ефективності та техніко-економічного обґрунтування системи. У висновках підсумовано результати дослідження, підтверджено досягнення поставленої мети та виконання всіх завдань.

Ключові слова: мобільний застосунок, система замовлень, QR-код, Firebase, Flutter, база даних, автентифікація, реальний час, інформаційна система.

ABSTRACT

Holiuk D. O. development of a Mobile Application for Managing Orders in Service Establishments Based on Flutter and Firebase. Manuscript. Bachelor's qualification work in Software Engineering, specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, and a list of references.

In the first chapter, the problem domain related to the automation of customer service processes in catering establishments was analyzed. Existing software solutions were reviewed, their advantages and disadvantages were identified, and the feasibility of developing a custom mobile application was substantiated. In the second chapter, functional and non-functional requirements were defined, the system was designed using UML notation, and the choice of technologies, methods, and tools was justified. The third chapter describes the implementation of the application, including the development of the Firebase Firestore database structure, implementation of authentication mechanisms and order processing, as well as testing results, efficiency evaluation, and economic justification. The conclusions summarize the obtained results, confirm the achievement of the research objective, and verify the fulfillment of all tasks.

Keywords: mobile application, order management system, QR code, Firebase, Flutter, database, authentication, real-time system, information system.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У ЗАКЛАДАХ ОБСЛУГОВУВАННЯ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ	10
1.1 Аналіз сучасного стану проблеми	10
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	16
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ ..	17
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення	17
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	19
РОЗДІЛ 3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У ЗАКЛАДАХ ОБСЛУГОВУВАННЯ НА ОСНОВІ FLUTTER ТА FIREBASE	25
3.1 Практична реалізація об'єкта проектування	25
3.2 Тестування та налагодження інформаційно-комп'ютерної системи.....	30
3.3 Розробка бази даних.....	33
3.4 Техніко-економічне обґрунтування	37
ВИСНОВКИ.....	40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	42

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активним упровадженням цифрових рішень у сферу обслуговування, зокрема в діяльність закладів громадського харчування та сервісних підприємств. Зростання кількості мобільних пристроїв, поширення безконтактних технологій і підвищення вимог споживачів до швидкості та якості обслуговування зумовлюють необхідність автоматизації ключових бізнес-процесів, серед яких особливе місце займає процес приймання та обробки замовлень.

Традиційна модель обслуговування клієнтів у закладах обслуговування передбачає значну кількість ручних операцій, що пов'язані з прийманням замовлень, їх передаванням на кухню, контролем виконання та розрахунками. Такий підхід часто призводить до затримок, помилок у замовленнях, перевантаження персоналу в години пікового попиту та зниження загального рівня задоволеності клієнтів. Особливо актуальною ця проблема стає для закладів із великою кількістю столів і високою інтенсивністю обслуговування.

Одним із перспективних напрямів вирішення зазначених проблем є використання мобільних застосунків, що дозволяють автоматизувати процеси замовлення та забезпечити оперативну взаємодію між клієнтами й персоналом. Використання QR-кодів для доступу до електронного меню та формування замовлень безпосередньо зі смартфона клієнта дає змогу зменшити кількість проміжних операцій, скоротити час обслуговування та знизити ймовірність помилок. Водночас для персоналу закладу важливо мати інструмент, який дозволяє централізовано керувати замовленнями, відстежувати їхній стан у режимі реального часу та розподіляти відповідальність між співробітниками.

У контексті розробки таких систем особливу увагу привертають кросплатформні технології створення мобільних застосунків, зокрема

фреймворк Flutter, а також хмарні платформи, що забезпечують зберігання даних і синхронізацію в реальному часі. Використання платформи Firebase у поєднанні з Cloud Firestore дозволяє реалізувати масштабовану, надійну та відносно просту в супроводі інформаційну систему без необхідності розгортання власної серверної інфраструктури.

З огляду на вищезазначене, розробка мобільного застосунку для управління замовленнями у закладах обслуговування є актуальним завданням, що поєднує теоретичні аспекти проєктування інформаційних систем і практичні навички створення сучасного програмного забезпечення.

Метою кваліфікаційної роботи бакалавра є розробка мобільного застосунку для управління замовленнями у закладах обслуговування на основі фреймворку Flutter та хмарної платформи Firebase, який забезпечує автоматизацію процесу приймання, обробки та контролю замовлень, а також покращує взаємодію між клієнтами та персоналом закладу.

Об'єктом кваліфікаційної роботи бакалавра є процеси обслуговування клієнтів у закладах громадського харчування та сервісних підприємствах, зокрема процеси формування, передавання та виконання замовлень.

Предметом кваліфікаційної роботи бакалавра є методи, моделі та програмні засоби розробки мобільних інформаційних систем для управління замовленнями, а також способи зберігання, синхронізації та обробки даних у хмарних середовищах.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

- проаналізувати сучасний стан проблеми автоматизації процесів замовлення у закладах обслуговування та існуючі програмні рішення в цій сфері;
- визначити функціональні та нефункціональні вимоги до мобільного застосунку управління замовленнями;
- обґрунтувати вибір технологій і інструментальних засобів для реалізації системи, зокрема фреймворку Flutter та платформи Firebase;

- спроектувати архітектуру програмного забезпечення, структуру бази даних і логіку взаємодії основних компонентів системи;
- розробити клієнтський і службовий інтерфейси мобільного застосунку з урахуванням ролей користувачів;
- реалізувати механізми зберігання та синхронізації замовлень у режимі реального часу;
- забезпечити базовий рівень захисту даних і розмежування прав доступу;
- протестувати розроблений програмний продукт та оцінити його працездатність.

Практичне значення отриманих результатів полягає у розробці мобільного застосунку для автоматизації процесу управління замовленнями у закладах обслуговування, який може бути використаний для підвищення ефективності роботи персоналу та оптимізації взаємодії з клієнтами. Реалізована система забезпечує можливість швидкого формування й обробки замовлень, підтримує синхронізацію даних у режимі реального часу та дозволяє зменшити кількість помилок, пов'язаних із ручною обробкою інформації. Використання сучасних технологій Flutter та Firebase забезпечує масштабованість, гнучкість та можливість подальшого розширення функціональних можливостей програмного продукту відповідно до потреб користувачів.

РОЗДІЛ 1

АНАЛІЗ ЗАСТОСУНКІВ ДЛЯ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У ЗАКЛАДАХ ОБСЛУГОВУВАННЯ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Сфера закладів обслуговування, зокрема підприємств громадського харчування, характеризується високою динамічністю процесів та значною залежністю якості сервісу від швидкості й точності обробки замовлень. Традиційні підходи до приймання замовлень, що ґрунтуються на усній комунікації або ручному введенні даних, часто призводять до помилок, затримок у передачі інформації між клієнтом і персоналом, а також підвищеного навантаження на працівників у години пікового попиту. Це негативно впливає як на задоволеність клієнтів, так і на загальну ефективність функціонування закладу.

Активний розвиток мобільних технологій та кросплатформних фреймворків сприяв появі цифрових рішень, орієнтованих на автоматизацію процесів обслуговування клієнтів, зокрема приймання та обробки замовлень у режимі реального часу. Одним із поширених підходів є використання QR-кодів для доступу до електронного меню та ініціювання процесу формування замовлення, що є складовою більш комплексних систем управління замовленнями. Наукові дослідження підтверджують, що QR-меню можуть позитивно впливати на сприйняту якість сервісу та задоволеність клієнтів, однак ефективність таких рішень значною мірою залежить від рівня довіри до технології та зручності користувацького інтерфейсу [1].

У науковій літературі також розглядаються QR-орієнтовані системи замовлення у форматі dine-in, у яких QR-код використовується як ідентифікатор конкретного столу або локації. Після сканування коду клієнт отримує доступ до цифрового меню та може сформувати замовлення, яке автоматично передається до системи обробки закладу. Такий підхід дозволяє

скоротити кількість проміжних операцій і зменшити ймовірність помилок при передачі інформації [2].

Окрема група робіт присвячена поєднанню процесів замовлення та оплати в межах єдиного цифрового сценарію. Використання QR-кодів для ініціації замовлення та подальшої оплати дає змогу скоротити час обслуговування та оптимізувати роботу персоналу, особливо в умовах високого навантаження [3]. Водночас такі системи потребують надійної серверної інфраструктури та безпечної обробки даних користувачів.

Важливим аспектом є також прийняття мобільних технологій користувачами. Дослідження, засновані на розширених моделях прийняття технологій, показують, що ключовими чинниками є очікувана користь, простота використання та довіра до системи [4]. Це означає, що мобільні застосунки для управління замовленнями повинні забезпечувати інтуїтивно зрозумілий інтерфейс, прозорі статуси замовлень і мінімальну кількість дій з боку користувача.

Патентний аналіз свідчить про активну розробку технічних рішень у сфері QR-орієнтованих систем замовлення для ресторанів та інших сервісних об'єктів. У відповідних патентах (рис. 1.1) описуються методи ідентифікації столу або клієнта за допомогою QR-коду, динамічне відображення меню на мобільному пристрої та передача замовлення на сервер закладу [5, 7]. При цьому більшість патентних рішень зосереджується на окремих аспектах процесу замовлення і не враховує комплексно операційні потреби персоналу. Сучасні QR-орієнтовані системи замовлення дедалі частіше інтегруються з хмарними сервісами та мобільними платформами, що забезпечує підтримку синхронізації даних у режимі реального часу, централізоване зберігання інформації та оперативне оновлення статусів замовлень. Це особливо важливо для закладів обслуговування з високою інтенсивністю роботи, де швидкість передачі інформації між клієнтом, персоналом та кухнею безпосередньо впливає на якість сервісу.

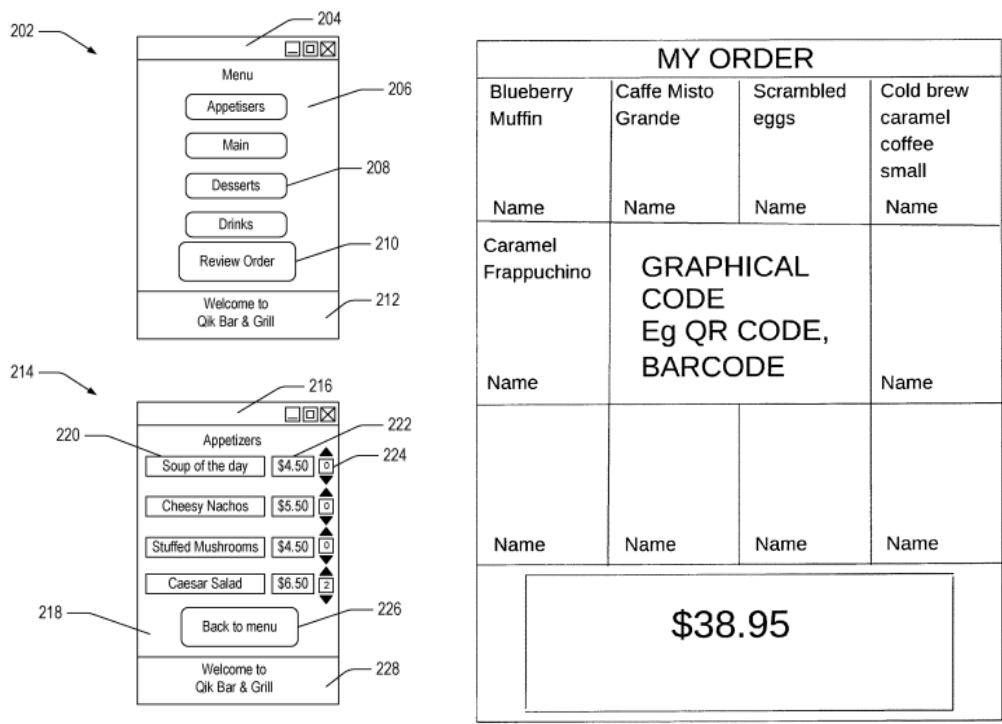


Рисунок 1.1 – Приклади динамічного відображення меню на мобільному пристрої [5, 7]

На практиці широкого поширення набули комерційні рішення класу Order at Table та Order & Pay. Наприклад, платформа Innovorder (рис. 1.2) надає можливість клієнтам оформлювати замовлення за столом шляхом сканування QR-коду та передавати їх безпосередньо до внутрішніх систем закладу [8].

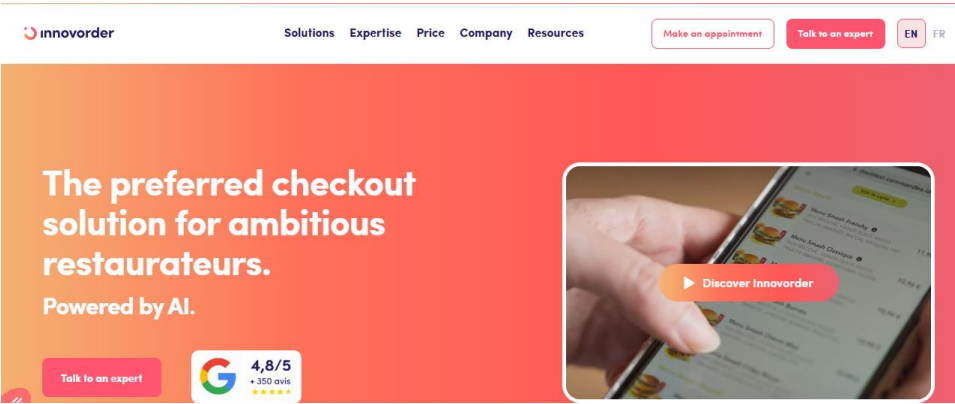


Рисунок 1.2 – Платформа Innovorder [8]

Аналогічний підхід реалізовано в системі компанії Access Group, яка поєднує процеси замовлення та оплати за столом у межах одного цифрового рішення [9]. Водночас аналітичні огляди підкреслюють, що QR-системи можуть мати обмеження, пов'язані з інклюзивністю та залежністю від стабільного інтернет-з'єднання, що необхідно враховувати під час проєктування таких рішень [10].

Серед сучасних цифрових рішень для автоматизації приймання замовлень у закладах обслуговування виділяються сервіси, які дозволяють клієнтам самостійно сканувати QR-код і переглядати меню з подальшим оформленням замовлення та оплатою. Одним із таких рішень є OrderLina – система, що генерує унікальні QR-коди для столів, які гості можуть сканувати, щоб оформити замовлення без участі персоналу (рис. 1.3) [11].



Рисунок 1.3 – OrderLina [11]

Аналогічно, ChoiceQR (рис. 1.4) пропонує комплекс функцій, включаючи онлайн-меню, спосіб бронювання столів та CRM-інструменти, що дозволяє не лише замовляти їжу, але й збирати відгуки та управляти базою клієнтів [12].



Рисунок 1.4 – ChoiceQR [12]

Також варто зазначити рішення Yoordi Direct (рис. 1.5), яке дозволяє клієнтам розміщувати та оплачувати замовлення через QR-інтерфейс без встановлення окремого застосунку [13].

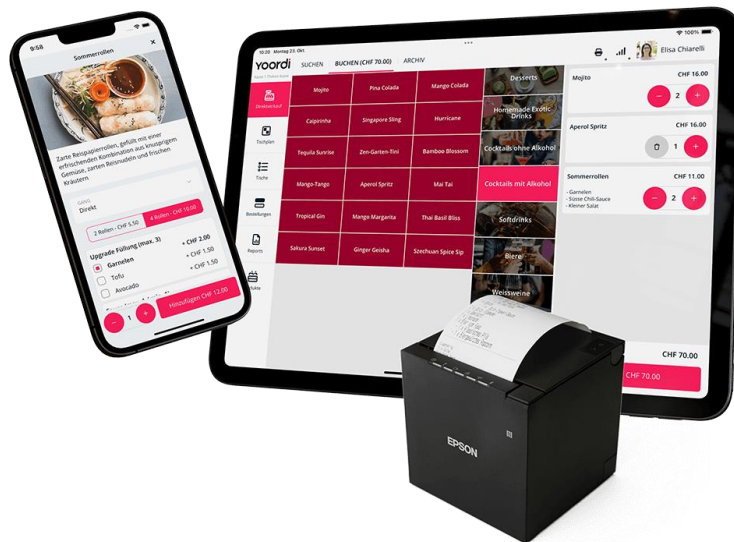


Рисунок 1.5 – Yoordi Direct [13].

Інші приклади включають MyDigiMenu і MENU TIGER, що розширюють стандартне QR-меню функціями управління позиціями та інтеграцією з POS-системами. Окремо слід відзначити рішення Square QR Code Ordering, яке інтегрує QR-замовлення з повноцінною системою POS та оплатою.

Таким чином, аналіз реальних наукових джерел, патентних рішень і комерційних систем показує, що попри наявність різноманітних QR-орієнтованих рішень, залишається потреба у створенні комплексного мобільного застосунку для управління замовленнями у закладах обслуговування. Такий застосунок має поєднувати зручний клієнтський сценарій із функціональним інтерфейсом для персоналу та забезпечувати надійну обробку даних у режимі реального часу. Використання Flutter як кросплатформного фреймворку та Firebase як хмарної платформи створює передумови для реалізації масштабованої та ефективної системи, що відповідає сучасним вимогам сфери обслуговування.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

У процесі проєктування та розробки програмного забезпечення для управління замовленнями у закладах обслуговування важливим етапом є чітке формулювання мети та завдань роботи. Саме визначення завдань дозволяє структурувати подальші етапи дослідження, окреслити межі предметної області та забезпечити послідовність виконання теоретичних і практичних частин роботи. У цьому підрозділі наведено перелік основних завдань, виконання яких є необхідним для досягнення поставленої мети:

- проаналізувати сучасний стан проблеми автоматизації процесів замовлення у закладах обслуговування та існуючі програмні рішення в цій сфері;
- визначити функціональні та нефункціональні вимоги до мобільного застосунку управління замовленнями;
- обґрунтувати вибір технологій і інструментальних засобів для реалізації системи, зокрема фреймворку Flutter та платформи Firebase;
- спроектувати архітектуру програмного забезпечення, структуру бази даних і логіку взаємодії основних компонентів системи;
- розробити інтерфейс мобільного застосунку з урахуванням ролей користувачів;
- реалізувати механізми зберігання та синхронізації замовлень у режимі реального часу;
- забезпечити базовий рівень захисту даних і розмежування прав доступу;
- протестувати розроблений програмний продукт та оцінити його працездатність.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Проектування програмного забезпечення для управління замовленнями у закладах обслуговування потребує системного підходу, який включає аналіз предметної області, формування цілей і задач системи, визначення вимог користувачів, а також побудову моделей програмної системи.

З огляду на характер завдання та можливість поетапного розширення функціональності, доцільним є використання ітеративної (інкрементальної) моделі розробки. Такий підхід дозволяє поступово уточнювати вимоги, тестувати окремі модулі системи та вносити корективи на ранніх етапах.

Для моделювання програмної системи та її елементів застосовуються формалізовані методи UML, які дозволяють наочно описати структуру системи, взаємодію між компонентами та сценарії використання. UML-діаграми забезпечують єдине розуміння архітектури системи як для розробника, так і для потенційних користувачів або замовників.

Вимоги до системи формуються на основі аналізу предметної області. Для їх виявлення доцільно використовувати методи аналізу сценаріїв використання, а також підхід «подія–відгук», який дозволяє чітко визначити реакцію системи на дії користувачів.

Архітектура розроблюваного програмного забезпечення передбачає клієнт–серверну модель, у якій мобільний застосунок, створений за допомогою Flutter, виступає клієнтською частиною, а платформа Firebase забезпечує серверну інфраструктуру. Firebase використовується для зберігання даних у хмарній базі, реалізації автентифікації користувачів та синхронізації даних у реальному часі.

Діаграма прецедентів (Use Case Diagram) відображає основні сценарії використання системи та взаємодію ролей: «Клієнт», «Працівник», «Адміністратор» (рис. 2.1).

Для «Клієнт» визначено прецеденти сканування QR-коду столу, перегляду меню, формування та надсилання замовлення, а також перегляду статусу. Для «Працівник» – перегляд черги замовлень, приймання/відхилення замовлення, оновлення статусів і перегляд історії. Для «Адміністратор» – керування меню, столами/QR-кодами та ролями користувачів. Також відображено залежності типу «include»: відправлення замовлення включає його формування, а формування – перегляд меню, який включає сканування QR-коду.

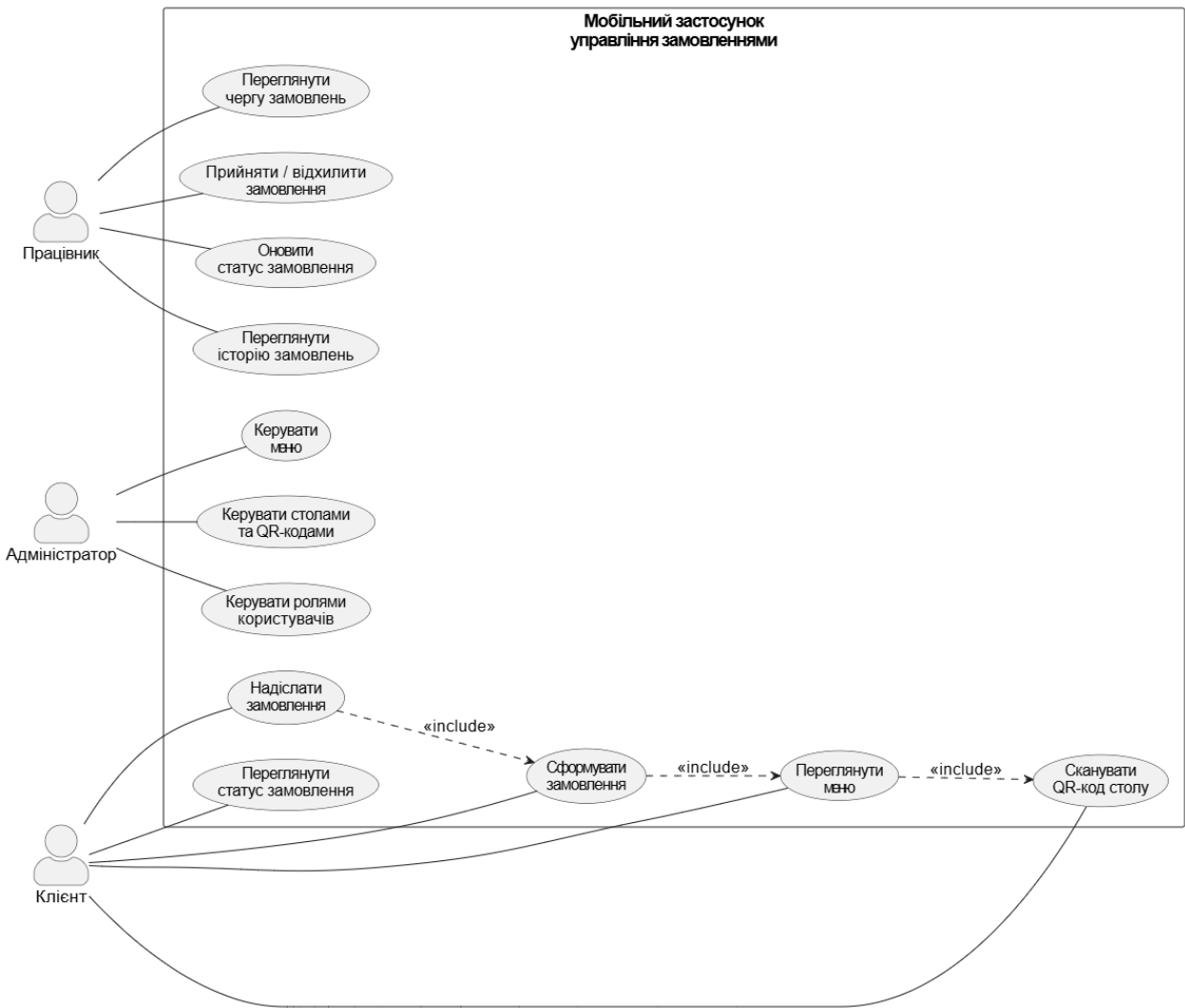


Рисунок 2.1 – Діаграма прецедентів

UX/UI-дизайн мобільного застосунку має забезпечувати простоту та зрозумілість взаємодії користувачів із системою. Інтерфейс клієнта орієнтований на швидкий доступ до меню та мінімальну кількість дій для формування замовлення, тоді як інтерфейс персоналу передбачає зручний перегляд списку замовлень, зміну їх статусів та контроль процесу обслуговування.

Інформаційні потоки в системі організовані таким чином, щоб дані про замовлення передавалися від клієнта до серверної частини, а далі – до персоналу закладу в режимі реального часу. Застосування Firebase дозволяє реалізувати ефективну технологію інформаційного обслуговування та задовольнити інформаційні потреби користувачів без необхідності складної серверної конфігурації.

Побудовані моделі програмного забезпечення дозволяють зробити висновок, що обрана архітектура та методи проєктування є доцільними для реалізації поставлених задач. Запропоноване рішення забезпечує автоматизацію процесів управління замовленнями, підвищує прозорість взаємодії між клієнтом і персоналом та створює основу для подальшого розширення функціональності системи.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка мобільного застосунку для управління замовленнями у закладах обслуговування вимагає використання сучасних програмних засобів, які забезпечують кросплатформність, масштабованість, підтримку роботи в режимі реального часу та надійну обробку даних. У межах цього розділу здійснюється аналіз існуючих інструментів розробки програмного забезпечення, методів та алгоритмів, які можуть бути застосовані для вирішення поставленого завдання, а також обґрунтовується вибір конкретних технологій.

На сучасному етапі розвитку програмного забезпечення для створення мобільних застосунків широко використовуються такі підходи: нативна розробка (Android, iOS), гібридні рішення та кросплатформні фреймворки. Нативна розробка забезпечує високу продуктивність, однак вимагає створення окремих застосунків для кожної платформи, що ускладнює супровід та збільшує витрати часу і ресурсів. Гібридні підходи спрощують розробку, але часто мають обмеження щодо продуктивності та доступу до системних ресурсів.

Кросплатформні фреймворки дозволяють створювати єдиний код для декількох платформ, що є доцільним для інформаційних систем у сфері обслуговування. Серед таких фреймворків найбільш поширеними є React Native, Xamarin та Flutter. У межах даної роботи обрано Flutter, оскільки він забезпечує високу продуктивність, використовує єдину кодову базу, підтримує створення адаптивного інтерфейсу та має розвинену екосистему бібліотек. Flutter дозволяє швидко реалізувати сучасний UX/UI-дизайн і ефективно працювати з асинхронними даними, що є важливим для систем управління замовленнями.

У роботі використано платформу Firebase, зокрема сервіси Firebase Authentication та Cloud Firestore (NoSQL документна БД).

У процесі реалізації серверної частини програмної системи використано хмарну платформу Firebase, що забезпечує комплексну підтримку функціонування застосунку. Зокрема, для зберігання даних застосовано базу даних Cloud Firestore, яка дозволяє організувати структуру даних у вигляді колекцій і документів та забезпечує швидкий доступ до інформації незалежно від кількості користувачів.

Автентифікація користувачів у системі реалізована за допомогою сервісу Firebase Authentication, що забезпечує безпечний механізм входу та ідентифікації користувачів. Додатково реалізовано рольову модель доступу, яка базується на збереженні ролі користувача в базі даних Cloud Firestore та використанні відповідних правил доступу. Такий підхід дозволяє

розмежувати функціональні можливості між різними категоріями користувачів, зокрема клієнтами, працівниками та адміністраторами.

Важливою особливістю обраного підходу є підтримка синхронізації даних у режимі реального часу, що забезпечується можливостями Firebase Firestore. Це дозволяє оперативно оновлювати інформацію про замовлення, зміну їх статусу та інші події без необхідності перезавантаження застосунку.

Крім того, використання хмарної інфраструктури забезпечує масштабованість системи без необхідності ручного адміністрування серверів. Firebase автоматично обробляє навантаження та забезпечує стабільну роботу системи навіть при збільшенні кількості користувачів.

Інтеграція з мобільним застосунком, розробленим із використанням фреймворку Flutter, здійснюється через відповідні програмні інтерфейси Firebase SDK. Це дозволяє забезпечити ефективну взаємодію між клієнтською та серверною частинами системи, а також спрощує процес розробки та підтримки програмного продукту.

Firestore як NoSQL-база даних дозволяє зберігати інформацію про замовлення, меню, користувачів і статуси у вигляді документів та колекцій, що добре відповідає структурі предметної області.

У розроблюваному мобільному застосунку використовуються алгоритми прикладного характеру, орієнтовані на подієву обробку даних і зміну станів об'єктів системи. Ключовим є алгоритм управління замовленнями, який охоплює повний життєвий цикл замовлення – від моменту його формування клієнтом до завершення обслуговування та архівації.

Алгоритм формування замовлення клієнтом, представлений на рисунку 2.2, починається зі сканування QR-коду, що дозволяє системі визначити ідентифікатор закладу та конкретного столу. Після цього здійснюється завантаження актуального меню з бази даних, клієнт обирає необхідні позиції та підтверджує замовлення. Далі клієнт може робити замовлення без акаунта, тільки через QR, після чого сформоване замовлення

зберігається у базі даних Firestore. Завершальним етапом є надсилання сповіщення персоналу закладу про надходження нового замовлення.

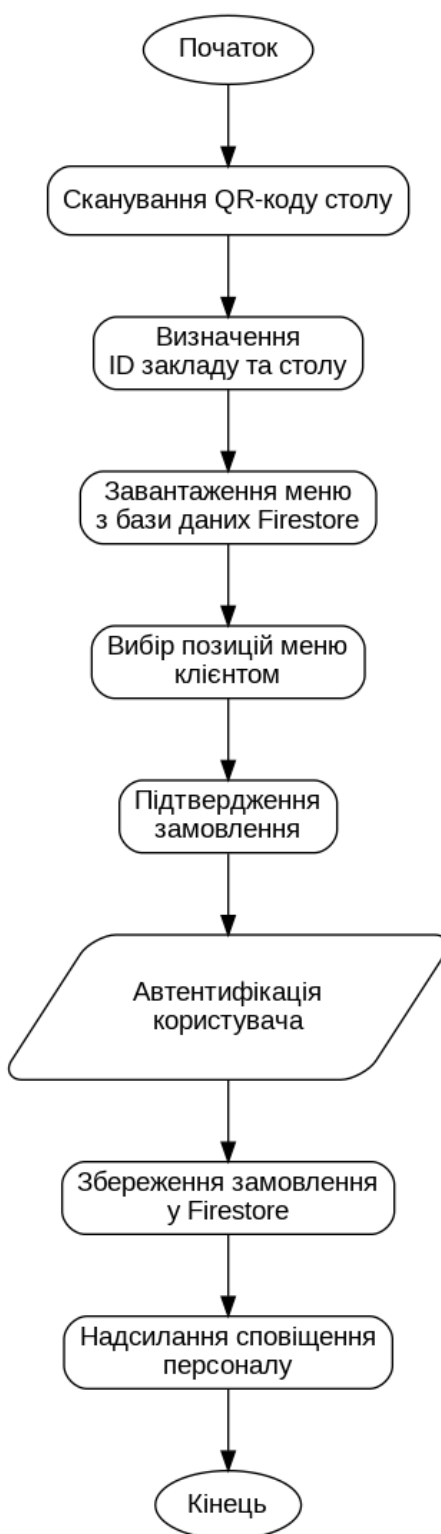


Рисунок 2.2 – Блок-схема алгоритму формування замовлення клієнтом

Алгоритм обробки замовлення персоналом, наведений на рисунку 2.3, відображає послідовність дій працівника закладу після отримання інформації про нове замовлення. Персонал переглядає список активних замовлень та приймає рішення щодо їх обробки. У разі підтвердження замовлення система послідовно змінює його статуси відповідно до етапів виконання: «Прийнято», «Готується», «Готово» та «Видано». Після завершення обслуговування замовлення переноситься до архіву, що забезпечує збереження історії виконаних замовлень і можливість подальшого аналізу.

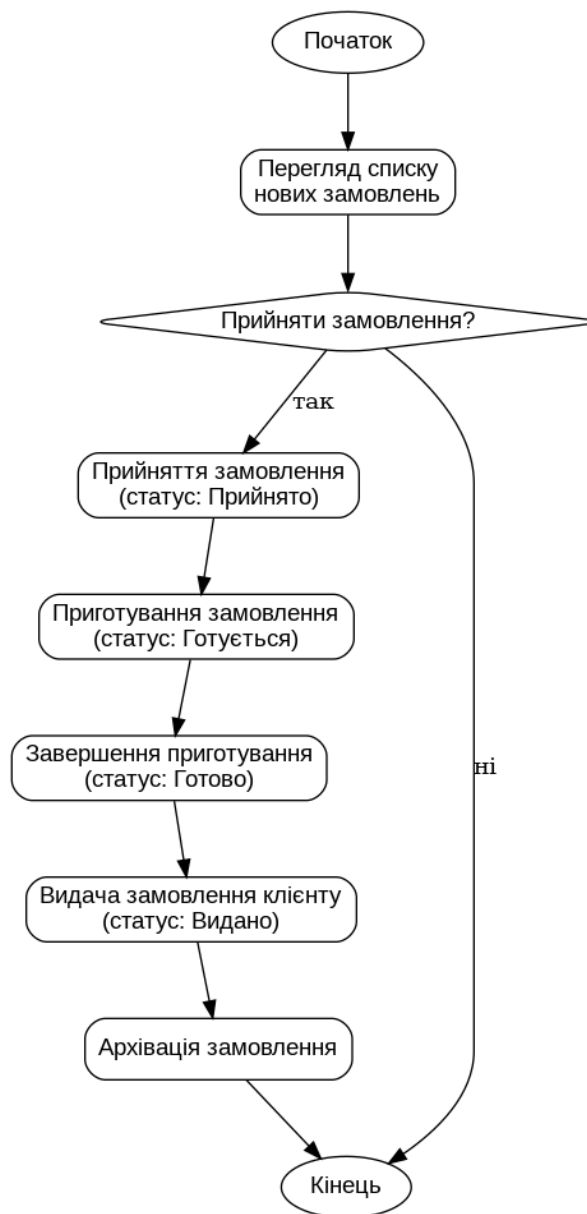


Рисунок 2.5 – Блок-схема алгоритму обробки замовлення персоналом

Для управління життєвим циклом замовлення використовується модель станів, у якій замовлення може перебувати в одному з визначених станів (створено, прийнято, готується, готово, видано). Перехід між станами здійснюється за подіями, ініційованими діями персоналу або системи.

Таким чином, аналіз існуючих засобів і методів розробки програмного забезпечення дозволив обґрунтовано обрати кросплатформний фреймворк Flutter та хмарну платформу Firebase як основні інструменти для реалізації мобільного застосунку управління замовленнями. Обрані методи проєктування та алгоритми забезпечують ефективне вирішення задач предметної області, масштабованість системи та зручність її використання як для клієнтів, так і для персоналу закладів обслуговування.

РОЗДІЛ 3

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ У ЗАКЛАДАХ ОБСЛУГОВУВАННЯ НА ОСНОВІ FLUTTER ТА FIREBASE

3.1 Практична реалізація об'єкта проектування

Ідея розроблюваного мобільного застосунку полягає у створенні кросплатформного рішення на основі фреймворку Flutter, призначеного для автоматизації процесу замовлення у закладах обслуговування. Застосунок орієнтований на спрощення взаємодії між клієнтами та персоналом і підвищення ефективності обслуговування.

Після запуску застосунку користувач може обрати один із сценаріїв роботи: автентифікацію або реєстрацію для співробітників закладу, а також використання QR-сканера для клієнтів. Клієнти сканують QR-код столу, отримують доступ до електронного меню та формують замовлення, яке зберігається в централізованій базі даних.

Для забезпечення контролю внутрішніх процесів передбачено обов'язкову реєстрацію співробітників. Після входу в систему співробітник визначає кількість столів у закладі та обирає зону відповідальності, що дозволяє відображати лише релевантні замовлення. Нові замовлення відображаються в режимі реального часу з відповідним сповіщенням, а після завершення обслуговування можуть бути видалені з системи.

Реєстрація та вхід співробітників. Після запуску мобільного застосунку відображається стартовий екран, під час якого виконується ініціалізація клієнтської частини та встановлення з'єднання з сервісами Firebase. Після завершення підготовчих операцій користувачеві надається початкове меню з можливістю виконати вхід або реєстрацію (для співробітників), а також перейти до сканування QR-коду (для клієнтів). Такий поділ сценаріїв використання відповідає вимогам предметної області, оскільки клієнтський режим не потребує доступу до службових функцій, тоді як персонал працює

із замовленнями та статусами і має бути ідентифікований у системі. Клієнтський сценарій починається зі сканування QR-коду. Вхідними даними QR-коду виступає посилання або службова інформація, що дозволяє ідентифікувати заклад/стіл та відкрити інтерфейс оформлення замовлення. Реалізація такого сканера в Flutter зазвичай базується на сторонньому пакеті, який надає віджет камери та callback для обробки результату (onDetect). При цьому важливо враховувати ризик застарівання неофіційних пакетів і залежність від оновлень Flutter.

Для запобігання несанкціонованому доступу до інтерфейсу персоналу застосунок використовує механізми автентифікації платформи Firebase. Застосування Firebase Authentication дозволяє централізовано керувати обліковими записами, забезпечувати базову перевірку даних та підтримувати подальше розмежування прав доступу (співробітник/адміністратор) на рівні правил доступу до даних. У межах даної реалізації використано автентифікацію за допомогою електронної пошти та пароля як найбільш універсальний спосіб для корпоративного використання.

Під час реєстрації співробітник вводить адресу електронної пошти, пароль і його підтвердження, а також службові дані, необхідні для роботи в закладі (наприклад, короткий ідентифікатор/ініціали для відображення в замовленнях). Додатково обирається роль або посада зі списку, що використовується для коректного призначення доступу та подальшої прив'язки до конкретного закладу обслуговування. Після успішного створення облікового запису застосунок може ініціювати підтвердження електронної пошти, що підвищує надійність ідентифікації користувача. Усе це представлено на рисунку 3.1. Функція registerStaff() реалізує перевірку введених даних, створення облікового запису співробітника у Firebase Authentication та формування службового профілю у Cloud Firestore. Така реалізація дозволяє розмежовувати публічний клієнтський сценарій та службовий режим роботи персоналу, що є необхідним для систем управління

замовленнями (ліст. 3.1). Після реєстрації користувачі можуть увійти в застосунок.

DO Staff DO

Реєстрація, вхід і перехід до сценарію роботи

Екран демонструє службову реєстрацію працівника, авторизацію за електронною поштою та окремий сценарій для клієнта через QR-код столу.

Реєстрація персоналу
Створення службового акаунта з роллю, ініціалами та прив'язкою до закладу.

Безпечний вхід
Авторизація через Firebase Authentication для доступу до внутрішніх функцій.

Електронна пошта:

Ініціали:

Пароль:

Підтвердження пароля:

Роль:

Заклад:

Зареєструвати співробітника **Увійти**

Сканувати QR-код

Рисунок 3.1 – Етапи реєстрації, сканування, входу у застосунок

Лістинг 3.1 – Функція реєстрації співробітника (Flutter + Firebase)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
class StaffAuthService {
  final FirebaseAuth _auth;
  final FirebaseFirestore _db;
  StaffAuthService({
    FirebaseAuth? auth,
    FirebaseFirestore? db,
  }) : _auth = auth ?? FirebaseAuth.instance,
       _db = db ?? FirebaseFirestore.instance;
  /// Реєстрація співробітника та створення службового профілю.
  Future<UserCredential> registerStaff({
    required String email,
    required String password,
    required String confirmPassword,
    required String initials,
    required String role, // напр.: "офіціант", "адміністратор"
    required String servicePlaceId // ID закладу
  }) {
```

```

}) async {
  // 1) Валідація введених даних
  final normalizedEmail = email.trim();
  final normalizedInitials = initials.trim();
  if (normalizedEmail.isEmpty || !normalizedEmail.contains('@')) {
    throw ArgumentError('Некоректна адреса електронної пошти.');
```

```

  }
  if (password.length < 7) {
    throw ArgumentError('Пароль має містити щонайменше 6 символів.');
```

```

  }
  if (password != confirmPassword) {
    throw ArgumentError('Паролі не співпадають.');
```

```

  }
  if (normalizedInitials.isEmpty) {
    throw ArgumentError('Ініціали є обов'язковими.');
```

```

  }
  if (role.trim().isEmpty) {
    throw ArgumentError('Необхідно обрати роль/посаду.');
```

```

  }
  if (servicePlaceId.trim().isEmpty) {
    throw ArgumentError('Необхідно вказати заклад обслуговування.');
```

```

  }
  // 2) Створення користувача в Firebase Authentication
  final credential = await _auth.createUserWithEmailAndPassword(
    email: normalizedEmail,
    password: password,
  );
  final user = credential.user;
  if (user == null) {
    throw StateError('Не вдалося створити обліковий запис.');
```

```

  }
  // 3) Збереження службового профілю у Firestore
  await _db.collection('staff').doc(user.uid).set({
    'email': normalizedEmail,
    'initials': normalizedInitials,
    'role': role.trim(),
    'servicePlaceId': servicePlaceId.trim(),
    'createdAt': FieldValue.serverTimestamp(),
    'isActive': true,
  });
  // 4) Надсилання листа підтвердження email (опційно)
  if (!user.emailVerified) {
    await user.sendEmailVerification();
  }
  return credential;
}
}

```

Налаштування параметрів обслуговування співробітником. Після успішної автентифікації співробітник переходить до етапу налаштування параметрів роботи закладу обслуговування. На цьому етапі необхідно задати загальну кількість столів у закладі, що використовується системою для коректної обробки та фільтрації замовлень.

Введення кількості столів реалізовано двома способами: шляхом безпосереднього введення числового значення у текстове поле або за допомогою повзунка. Такий підхід підвищує зручність користування та дозволяє швидко задати необхідне значення.

Для прямого введення числа використовується текстове поле з обмеженням на введення лише числових символів. У разі введення значення, що виходить за допустимі межі, користувачеві відображається відповідне повідомлення. При коректному введенні значення відбувається оновлення стану застосунку та збереження актуальних параметрів (ліст. 3.2).

Лістинг 3.2 – Введення кількості столів через текстове поле

```

TextField(
  controller: tablesController,
  keyboardType: TextInputType.number,
  inputFormatters: [
    FilteringTextInputFormatter.digitsOnly,
  ],
  onChanged: (value) {
    final count = int.tryParse(value);
    if (count == null || count < 1 || count > 25) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Кількість столів має бути в межах 1-25'),
        ),
      );
    } else {
      setState(() {
        totalTables = count;
      });
    }
  },
)

```

Кінець лістингу 3.2

Після завдання загальної кількості столів співробітник обирає номери столів, за які він відповідає. Для цього використовується діалоговий віджет множинного вибору, який формує список доступних столів у межах заданого діапазону та зберігає вибрані значення у стані застосунку. Такий механізм дозволяє відображати співробітнику лише ті замовлення, що належать до його зони відповідальності.

Рисунок 3.2 – Приклад введення кількості столів

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

У процесі розробки мобільного застосунку для управління замовленнями у закладах обслуговування на основі Flutter та Firebase було виконано тестування та налагодження програмного продукту з метою перевірки коректності роботи функціональних модулів, стабільності взаємодії з базою даних, а також зручності користувацького інтерфейсу.

Тестування проводилось на етапі реалізації та після завершення основних функціональних компонентів застосунку. Основна увага була приділена перевірці сценаріїв взаємодії користувачів із системою, а також обробці помилкових і граничних ситуацій.

Функціональне тестування передбачало перевірку відповідності реалізованих функцій вимогам, визначеним на етапі проєктування. Було протестовано основні сценарії роботи клієнтів та співробітників, зокрема:

- сканування QR-коду столу та перехід до меню закладу;
- формування та збереження замовлення клієнтом;
- реєстрація та вхід співробітників із використанням Firebase Authentication;
- відображення черги замовлень для персоналу;
- зміна статусу замовлення та його архівація після завершення обслуговування.

Під час тестування встановлено, що обробка QR-коду та отримання ідентифікатора столу відбувається менш ніж за 0,5 секунди. Нові замовлення з'являються в інтерфейсі співробітника практично миттєво (до 1 секунди) завдяки використанню механізмів реального часу Firestore (рис. 3.4).

Інтеграційне тестування було спрямоване на перевірку взаємодії між клієнтською частиною застосунку, сервісами Firebase Authentication та хмарною базою даних Firestore. Особливу увагу приділено коректності збереження та синхронізації даних замовлень між різними користувачами системи.

Було перевірено:

- створення облікового запису співробітника та збереження службового профілю в Firestore;
- запис замовлень у колекцію orders;
- фільтрацію замовлень відповідно до закладу та номерів столів, закріплених за співробітником.

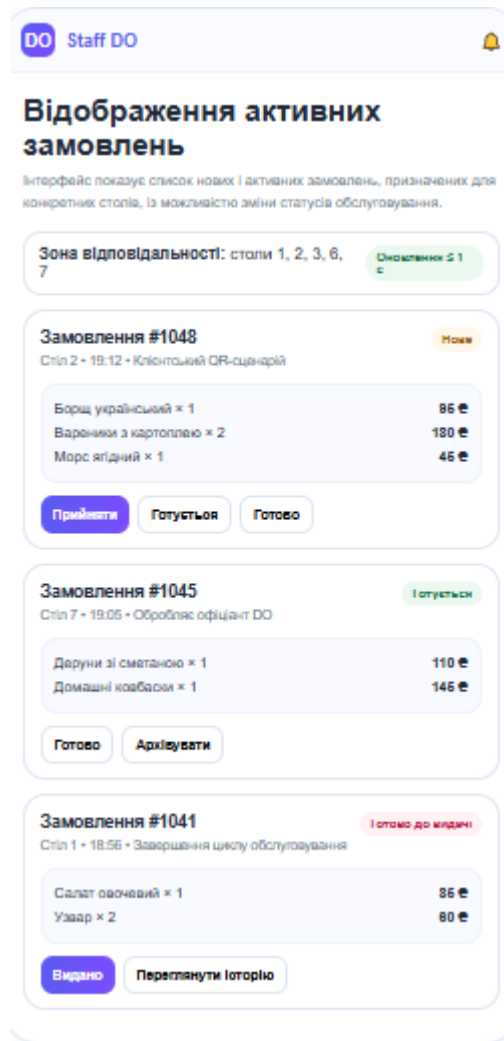


Рисунок 3.4 – Відображення активних замовлень у додатку

Результати тестування підтвердили коректну роботу інтерфейсу та відповідність принципам зрозумілості й доступності для користувачів.

У процесі налагодження застосунку використовувалися стандартні засоби логування Flutter та обробки винятків. Для запобігання аварійному завершенню роботи програми всі критичні операції взаємодії з базою даних обгорнуті у блоки try-catch.

Приклад обробки помилки під час збереження замовлення наведено нижче: у лістингу 3.3.

Лістинг 3.3 – Обробка помилки під час збереження замовлення

```
try {
  await FirebaseFirestore.instance
```

```

.collection('orders')
.add(orderData);
} catch (e) {
  debugPrint('Помилка збереження замовлення: $e');
}

```

Кінець лістингу 3.3

Такий підхід дозволяє своєчасно виявляти помилки під час роботи з мережею або базою даних та підвищує стабільність роботи застосунку. Основні результати тестування узагальнено в таблиці 3.1.

Таблиця 3.1 – Результати тестування програмного продукту

Тип тестування	Об'єкт перевірки	Очікуваний результат	Фактичний результат
Функціональне	Реєстрація співробітника	Створення акаунта	Успішно
Інтеграційне	Збереження замовлення	Запис у Firestore	Успішно
Інтеграційне	Синхронізація даних	Оновлення в реальному часі	Успішно

Проведене тестування та налагодження підтвердили коректність роботи розробленого мобільного застосунку, стабільність взаємодії з сервісами Firebase та відповідність реалізованих функцій поставленим вимогам. Система готова до подальшого використання та розширення функціональних можливостей.

3.3 Розробка бази даних

У застосунку база даних (БД) є ключовим компонентом, що забезпечує зберігання, обробку та синхронізацію даних між клієнтською частиною застосунку та серверною інфраструктурою. Основними завданнями БД у межах цієї роботи є збереження довідникової інформації (дані про заклад обслуговування, співробітників, меню), а також оперативне збереження й актуалізація замовлень у режимі реального часу.

Особливістю предметної області є необхідність миттєвого доступу до змін у стані замовлень для всіх зацікавлених користувачів, зокрема персоналу закладу. Тому БД повинна забезпечувати швидку вибірку даних, їх сортування за часовими ознаками та автоматичну синхронізацію між пристроями без потреби ручного оновлення. Такий підхід є типовим для сучасних мобільних сервісних застосунків, орієнтованих на роботу з подіями та зміну станів у реальному часі.

У розроблюваній системі використано хмарну платформу Firebase, яка добре інтегрується з фреймворком Flutter і широко використовується у мобільних інформаційних системах та складі якої для зберігання та синхронізації даних застосовано документно-орієнтовану базу даних Cloud Firestore.

Доцільність використання Firebase обумовлена кількома чинниками.

По-перше, Firebase надає готовий механізм автентифікації користувачів (Firebase Authentication (ліст. 3.1), що дозволяє реалізувати безпечну реєстрацію та вхід співробітників за допомогою електронної пошти та пароля без необхідності розробки власної серверної логіки. Це знижує складність реалізації та мінімізує ризики помилок у механізмах безпеки.

По-друге, платформа підтримує документно-орієнтовану БД Cloud Firestore, яка є гнучкою щодо структури даних і добре підходить для зберігання ієрархічних об'єктів, таких як меню закладів, категорії товарів та списки замовлень. На відміну від реляційних баз даних, Firestore не потребує жорстко фіксованої схеми, що дозволяє легко розширювати структуру даних у процесі розвитку застосунку.

По-третє, Firebase забезпечує автоматичну синхронізацію даних у режимі реального часу. Це означає, що будь-які зміни у замовленнях (створення нового замовлення, зміна його статусу, видалення) миттєво відображаються на пристроях усіх співробітників, відповідальних за обслуговування відповідних столів. Така властивість є критично важливою для систем управління замовленнями у закладах обслуговування.

Отже, вибір Firebase є обґрунтованим з погляду функціональності, масштабованості та відповідності вимогам предметної області.

При проєктуванні структури БД було враховано необхідність чіткого розмежування сутностей предметної області та забезпечення зручної навігації між ними. Основними логічними сутностями системи є заклад обслуговування, співробітники, меню та замовлення.

У розроблюваній інформаційній системі для управління замовленнями у закладах обслуговування структура меню реалізована з використанням документно-орієнтованої бази даних Cloud Firestore. Меню організовано у вигляді окремої колекції Menu, у якій кожен документ даному закладу обслуговування, кафе з українською тематикою. Такий підхід дозволяє логічно розділяти дані різних закладів та забезпечує масштабованість системи. Додатковою перевагою використання Cloud Firestore є підтримка ієрархічної структури даних та механізмів синхронізації у режимі реального часу. Завдяки цьому при зміні меню адміністратором закладу оновлення автоматично відображаються у мобільному застосунку без необхідності повторного завантаження даних або ручного оновлення інтерфейсу. Це забезпечує актуальність інформації для користувачів та підвищує ефективність роботи персоналу закладу обслуговування. Крім того, використання документно-орієнтованого підходу спрощує розширення структури меню у майбутньому, зокрема шляхом додавання нових категорій, описів страв, цін, фотографій або інформації про наявність позицій без необхідності зміни загальної архітектури бази даних.

На рисунку 3.5 наведено приклад структури зберігання меню у базі даних Firestore. Документи колекції Menu мають назви, «Козак», «Село», «Вечорниці», «Левада». Усередині документа закладу зберігаються поля, які представляють категорії меню, зокрема «Закуски», «Супи», «Гарячі страви» та «Напої». Значеннями цих полів є масиви рядків, кожен з яких відповідає окремій позиції меню, наприклад «Сало з часником», «Вареники з картоплею», «Оселедець під шубою», «Домашні ковбаски», «Борщ».

menu	Menu	Ceno
Start collection	+ Add document	+ Start collection
Menu >	Козак Село > Вечорниці Левада + Start collection ▶ Село ▶ Вечорниці ▶ Левада	+ Add field ▼ Закуси ▶ [Сало з часником", ▶ [Вареники з картоплею", ▶ [Оселедець під шубою", ▶ [Домашні ковбаски", ▶ Борщ" > Супи > Гарячі страви > Напої

Рисунок 3.5 – Приклад структури зберігання меню у БД

Така структура є доцільною для мобільного застосунку, оскільки дозволяє отримувати повний перелік категорій і позицій меню одного закладу в межах одного запиту до бази даних. Крім того, вона спрощує подальше відображення даних у клієнтському інтерфейсі та не потребує складних зв'язків між таблицями, характерних для реляційних баз даних.

Взаємодія клієнтської частини застосунку з даними меню реалізована через послідовну ієрархію компонентів інтерфейсу. На рисунку 3.6 представлено логічну схему компонентів, що відповідають за відображення меню в застосунку. Головний компонент «Додаток» ініціює роботу системи та передає керування компоненту «Вибір», який відповідає за вибір закладу обслуговування. Після цього відбувається завантаження компонента «КатегоріяМеню», який формує список доступних категорій і позицій меню для вибраного закладу. Кінцевим елементом цієї ієрархії є компонент «Товар», який відповідає за відображення окремої позиції меню та її додавання до замовлення.



Рисунок 3.6 – Логічна схема компонентів

Зв'язок між структурою бази даних і компонентами інтерфейсу є прямим: дані з документа відповідного закладу в колекції Menu використовуються для побудови списку категорій, а значення масивів категорій – для формування переліку товарів. Такий підхід забезпечує узгодженість між логічною моделлю даних і структурою клієнтського застосунку, а також спрощує реалізацію та подальший супровід системи.

Запропонована організація даних і компонентів є достатньо гнучкою та може бути розширена в майбутньому, наприклад шляхом заміни масивів рядків на складні об'єкти з додатковими атрибутами (ціна, доступність, опис), без необхідності суттєвої зміни архітектури застосунку.

3.4 Техніко-економічне обґрунтування

У процесі розробки мобільного застосунку для управління замовленнями у закладах обслуговування важливим етапом є оцінка економічної доцільності створення та впровадження програмного продукту. Техніко-економічне обґрунтування дозволяє визначити витрати на розробку

системи, оцінити ефективність її використання та обґрунтувати доцільність впровадження.

Основними складовими витрат є витрати на оплату праці розробника та експлуатаційні витрати. Витрати на оплату праці визначаються за формулою (3.1):

$$C_{dev}=T \cdot R, \quad (3.1)$$

де T – загальний час розробки (год),

R – погодинна ставка розробника.

Приймаючи $T=160$ годин та $R=150$ грн/год, отримаємо:

$$C_{dev}=160 \cdot 150=24000 \text{ грн.}$$

Додатково враховуються витрати на електроенергію, які визначаються за формулою (3.2):

$$C_{el}=P \cdot T \cdot C_{kWh}, \quad (3.2)$$

де P – потужність обладнання (кВт),

C_{kWh} – вартість електроенергії.

Підставивши $P=0,2$ кВт, $T=160$ год та $C_{kWh}=4$ грн/кВт·год, отримуємо:

$$C_{el}=0,2 \cdot 160 \cdot 4=128 \text{ грн.}$$

Загальні витрати на розробку визначаються як сума складових відповідно до формули (3.3):

$$C_{total}=C_{dev}+C_{el}, \quad (3.3)$$

звідки:

$$C_{\text{total}} = 24000 + 128 = 24128 \text{ грн.}$$

Економічний ефект від впровадження системи визначається за рахунок підвищення швидкості обробки замовлень та зменшення навантаження на персонал. Термін окупності визначається за формулою (3.4):

$$T_{\text{payback}} = C_{\text{total}} / E_{\text{month}} \quad (3.4)$$

де E_{month} – щомісячний економічний ефект.

За умови $E_{\text{month}} = 600$ грн отримуємо:

$$T_{\text{payback}} = 24128 / 6000 \approx 4 \text{ місяці.}$$

Як видно з формул (3.1)-(3.4), розробка програмного продукту є економічно доцільною, оскільки забезпечує відносно невеликі витрати на реалізацію та швидкий термін окупності. Використання сучасних хмарних технологій дозволяє додатково зменшити витрати на інфраструктуру та забезпечити масштабованість системи.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра досягнуто поставленої мети – розроблено мобільний застосунок для управління замовленнями у закладах обслуговування на основі Flutter та Firebase, який забезпечує автоматизацію процесів приймання, обробки та контролю замовлень у режимі реального часу.

У межах першого завдання проведено аналіз сучасного стану проблеми та існуючих рішень у сфері автоматизації замовлень. Зокрема, розглянуто наукові підходи до використання QR-кодів у сервісних системах, а також проаналізовано конкретні комерційні платформи, серед яких Innovorder, Access Group Order & Pay, OrderLina, ChoiceQR, Yoordi Direct, MyDigiMenu, MENU TIGER та Square QR Code Ordering. Аналіз показав, що більшість рішень орієнтовані переважно на клієнтський сценарій замовлення або інтеграцію з POS-системами, але не забезпечують повноцінного управління замовленнями з боку персоналу, що обґрунтовує необхідність створення комплексного застосунку.

Визначено функціональні та нефункціональні вимоги до системи. До основних функціональних можливостей віднесено сканування QR-коду столу, перегляд електронного меню, формування замовлення клієнтом, відображення та обробку замовлень персоналом, зміну статусів («прийнято», «готується», «готово», «видано»), а також керування ролями користувачів. Нефункціональні вимоги включають забезпечення швидкодії (обробка запитів до 1 секунди), роботи в режимі реального часу, масштабованості, зручності інтерфейсу та базового рівня безпеки.

Обґрунтовано вибір технологій – для реалізації мобільного застосунку обрано Flutter як кросплатформний фреймворк, що забезпечує єдину кодову базу та високий рівень продуктивності. Для серверної частини використано Firebase, зокрема Firebase Authentication для автентифікації користувачів і

Cloud Firestore як документно-орієнтовану базу даних із підтримкою синхронізації в режимі реального часу.

Розроблено структуру бази даних, що включає основні колекції: staff (співробітники), orders (замовлення), menu (меню закладів). Забезпечено логіку взаємодії компонентів системи, яка передбачає передачу замовлень від клієнта до персоналу з використанням хмарної інфраструктури.

Реалізовано клієнтський і службовий інтерфейси застосунку. Клієнт має можливість сканувати QR-код, переглядати меню та формувати замовлення без авторизації, тоді як співробітники проходять автентифікацію, отримують доступ до списку замовлень і можуть змінювати їх статуси відповідно до етапів обслуговування.

Реалізовано механізми зберігання та синхронізації даних у режимі реального часу за допомогою Cloud Firestore. Це забезпечує миттєве відображення нових замовлень та змін їх станів у застосунках персоналу.

Забезпечено базовий рівень захисту даних шляхом використання Firebase Authentication, перевірки коректності введених даних та розмежування доступу між клієнтами і співробітниками.

Проведено тестування програмного продукту. Результати показали, що система коректно обробляє основні сценарії використання: час обробки QR-коду становить менше 0,5 секунди, а оновлення замовлень у системі відбувається в межах 1 секунди, що підтверджує ефективність реалізованих рішень.

Отже, усі поставлені завдання виконано, а розроблений мобільний застосунок відповідає вимогам предметної області, є працездатним, масштабованим і може бути використаний у реальних умовах функціонування закладів обслуговування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yiğitoğlu V., Şahin E., Güneri B., Demir M.Ö. The Impact of Sustainable QR Menus on Service Quality and Customer Satisfaction. URL: <https://www.mdpi.com/2071-1050/17/5/2323> (дата звернення: 06.02.2026).
2. Patil P. et al. Smart QR-Based Restaurant Dine-In System with Sales Analysis. URL: <https://www.sciencedirect.com/science/article/pii/S1877042816305936> (дата звернення: 12.02.2026).
3. Gawande A., Patil N., Deshpande S. QR Code-Based Smart Food Ordering and Payment System. URL: <https://www.scitepress.org/Papers/2019/77232/77232.pdf> (дата звернення: 12.02.2026).
4. Liébana-Cabanillas F. et al. Antecedents of the adoption of QR code mobile payment systems. URL: <https://www.sciencedirect.com/science/article/pii/S0040162519304830> (дата звернення: 18.03.2026).
5. Ordering method and system for restaurants (WO2013110942A1). URL: <https://patents.google.com/patent/WO2013110942A1> (дата звернення: 16.03.2026).
6. Order system using QR code (KR20200140453A). URL: <https://patents.google.com/patent/KR20200140453A> (дата звернення: 06.02.2026).
7. QR Code Ordering Method and System for Restaurants (US20190325508A1). URL: <https://patents.google.com/patent/US20190325508A1> (дата звернення: 14.02.2026).
8. Innovorder. Order at Table – Digital Ordering Solution for Restaurants. URL: <https://www.innovorder.com/en/order-at-table> (дата звернення: 18.04.2026).
9. Access Group. Order & Pay at Table System. URL: <https://www.theaccessgroup.com/en-gb/hospitality/order-and-pay/> (дата звернення: 17.05.2026).

10. Tabesto. QR Code Ordering and Payment: Advantages and Limitations. URL: <https://www.tabesto.com/en/qr-code-ordering> (дата звернення: 16.05.2026).
11. OrderLina. URL: <https://www.restolabs.com/blog/best-qr-menu-software-restaurants> (дата звернення: 18.05.2026).
12. Choice. URL: <https://choiceqr.com/> (дата звернення: 22.05.2026).
13. Yoordi Direct. URL: <https://yoordi.com/yoordi-pos> (дата звернення: 06.02.2026).