Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту

Завідувач кафедри комп'ютерних наук, кандидат технічних наук, доцент,

_____ Ірина МАШКІНА

(підпис)

« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр» Спеціальність 122 Комп'ютерні науки Освітня програма 122.00.01 Інформатика

Тема:

Розробка мобільного додатку для допомоги у пошуку та оцінюванні закладів громадського харчування

Виконав
студент групи ІНб-2-21-4.0д

Колбін Владислав Тарасович

_____

Науковий керівник
кандидат технічних наук, доцент
_____ Ірина МЕЛЬНИК

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту
Завідувач кафедри комп'ютерних наук, кандидат технічних наук, доцент,
_____ Ірина МАШКІНА
(підпис)
« ___ » _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА

«Розробка мобільного додатку для допомоги у пошуку та оцінюванні закладів громадського харчування»

Виконавець – студент групи ІНб-2-21-4.0д

Колбін Владислав Тарасович

Вихідні дані: Наукові статті, курси освіти до необхідних тем.

Основні завдання: Здійснити огляд публікацій за темою роботи, обґрунтувати мету, об'єкт, предмет та наукові основи дослідження, розробити завдання, структуру та зміст бакалаврської роботи, обрати та обґрунтувати методи і засоби дослідження, підготувати матеріали до розділів роботи відповідно до індивідуального завдання, здійснити практичну розробку проекту зв'язаного з пошуком закладів громадського харчування

Пояснювальна записка: Обсяг – до 55 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.

1. Графічні матеріали: Презентація.
2. Додатки: Рис. 1-5, додатки А-Б
3. Строк подання роботи на кафедру: «5» червня 2025 р.

Науковий керівник:
кандидат технічних наук
доцент
_____ Мельник І.Ю.
дата

Виконавець:
_____
дата

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи бакалавра	25.12.2024	Створення теми для бакалаврської дипломної роботи
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	10.01.2025	Вивчення рекомендацій для створення додатку дипломної роботи
3	Складання змісту та календарного плану роботи	20.01.2025	Створення струкутури
4	Підбір літератури, складання завдання	31.01.2025	Пошук літератури, швидке проходження онлайн-курсів по
5	Написання реферату та вступу	07.02.2025	Створення короткого опису проекту
6	Написання першого розділу	20.02.2025	Написання теоретичної частини роботи
7	Написання другого розділу	6.03.2025	Теоретичний аналіз сфери створюванного

			додатку, його аналоги, їх недоліки та переваги
8	Написання третього розділу	27.03.2025	Показ практичної частини проекту, коду клієнтської та серверної частин додатку
9	Написання висновків	04.04.2025	Опис результату роботи та які навички були набуті
10	Виправлення зауважень	14.04.2025	Форматування дипломної роботи за вказівками керівника
11	Створення презентації	15.05.2025	
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	05.06.2025	

Здобувач_____

Керівник роботи _____

РЕФЕРАТ

Кваліфікаційна робота: 40 с., 5 рис., 2 табл., 23 посилання, додатки А-Б.

1. Актуальність. Сьогоднішній темп життя диктує нові умови та виклики, пов'язані з організацією повсякденного побуту, зокрема з вибором місць для харчування. З розвитком інформаційних технологій змінюється і підхід до прийняття рішень користувачами: замість випадкового вибору все більше людей покладаються на цифрові інструменти, які допомагають їм орієнтуватися у великій кількості доступних опцій. Зокрема, мобільні додатки стали незамінною частиною сучасного життя, дозволяючи швидко та зручно отримувати необхідну інформацію у будь-який момент часу.

Однією з актуальних проблем, з якою стикаються користувачі великих і середніх міст, є необхідність знаходити якісні та надійні заклади громадського харчування. Часто люди витрачають багато часу на пошук місця для обіду, вечері чи зустрічі з друзями, при цьому не завжди отримують задоволення від сервісу чи якості їжі. Саме тому виникає потреба в існуванні зручного цифрового інструменту, який би об’єднував не лише базу таких закладів, а й давав можливість порівнювати їх, читати відгуки інших користувачів та залишати власну оцінку.

Не менш важливим є аспект локалізації: більшість популярних додатків, як-от Google Maps чи TripAdvisor, хоч і пропонують широкий функціонал, але не завжди адаптовані до потреб локального користувача — з урахуванням мови, локального контенту, специфіки регіону тощо. Тому створення додатку, орієнтованого саме на українського користувача, має велике практичне значення, особливо в умовах зростання попиту на вітчизняні IT-продукти.

Крім того, варто зазначити, що цифровізація сфери обслуговування позитивно впливає не лише на користувачів, а й на самі заклади, які за допомогою таких сервісів можуть краще розуміти потреби своїх клієнтів,

удосконалювати сервіс, а також залучати нову аудиторію. Таким чином, розробка мобільного додатку для пошуку та оцінювання закладів громадського харчування є не просто актуальною — вона відповідає загальносвітовим трендам розвитку розумних міст і цифрового суспільства.

Об’єкт дослідження: мобільні додатки для пошуку та оцінювання закладів громадського харчування.

Предмет дослідження: методи та технології розробки кросплатформених мобільних додатків із використанням React Native, Java та бази даних MySQL.

Мета дослідження: дослідити сучасні підходи до розробки мобільних додатків для індустрії громадського харчування, визначити переваги кросплатформенних рішень, а також оцінити можливості використання React Native у поєднанні з Java-бекендом для створення функціонального та масштабованого сервісу.

Завдання дослідження: визначити ключові поняття та роль мобільних додатків у сфері громадського харчування, проаналізувати особливості нативної розробки, а також порівняти їх ефективність, розглянути основні функції мобільних додатків для пошуку та оцінювання закладів, дослідити інтеграцію фронтенд-рішення на React Native з Java-бекендом та MySQL, оцінити переваги, недоліки та можливості таких рішень у контексті реальних потреб користувачів.

Основні результати: визначено ключові функціональні компоненти мобільного додатку для сфери громадського харчування, зокрема пошук закладів, відображення їх на мапі, перегляд детальної інформації, залишення оцінок і відгуків, проведено аналіз сучасних підходів до кросплатформеної розробки, а також технічних можливостей інтеграції React Native з серверною частиною на Java, сформульовано основні вимоги до архітектури додатку, які забезпечують масштабованість, гнучкість і доступність на Android та iOS-платформах.

Практичне значення дослідження: результати дослідження мають практичне застосування у створенні цифрових рішень для туристів, мешканців міст та власників закладів, розроблений додаток може використовуватись для зручного пошуку кафе, ресторанів, кав’ярень, перегляду відгуків і рейтингу, що сприяє підвищенню прозорості у виборі місця для харчування, використання React Native забезпечує ефективну кросплатформенну розробку, тоді як Java та MySQL дозволяють створити стабільну серверну частину з можливістю обробки великих обсягів даних.

Ключові слова: заклади харчування, мобільний додаток, React Native, Java, MySQL, кросплатформенність, відгуки, геолокація, рейтинг, інтерфейс користувача.

ЗМІСТ

ВСТУП

У сучасному динамічному світі мобільні технології відіграють ключову роль у повсякденному житті людей, допомагаючи спростити безліч завдань — від спілкування до планування дозвілля. Однією з важливих щоденних

потреб є пошук якісних та зручних місць для харчування. Незважаючи на велику кількість закладів громадського харчування, користувачі часто стикаються з проблемами вибору — через нестачу достовірної інформації, відсутність зручного сервісу або надлишок неструктурованих відгуків в інтернеті. Саме ця проблема надихнула мене на створення мобільного додатку, який допомагатиме знаходити, оцінювати та обирати заклади харчування швидко, зручно і ефективно.

Метою дипломної роботи стало створення сучасного мобільного додатку для допомоги у пошуку та оцінюванні закладів громадського харчування. У процесі розробки я зосередився на тому, щоб інтерфейс був максимально інтуїтивно зрозумілим, а функціонал — справді корисним для кінцевого користувача. Для реалізації проєкту я обрав кросплатформену технологію **React Native**, що дало змогу забезпечити роботу додатку як на Android, так і на iOS пристроях, не витрачаючи ресурси на розробку окремих нативних рішень.

З боку серверної частини було використано **Java**, а зберігання інформації про заклади, рейтинги та користувачів реалізовано на базі **MySQL**. Це дозволило побудувати надійне, масштабоване та гнучке рішення, яке в майбутньому можна легко доповнювати новими модулями та функціями. Також у проєкті реалізовано базову систему авторизації, відображення закладів на карті, перегляд детальної інформації, а також залишення оцінок і відгуків користувачами.

Розроблений додаток орієнтований на широке коло користувачів — як місцевих жителів, так і туристів, які хочуть швидко знайти перевірене місце для обіду чи вечері. Він поєднує в собі функціональність, простоту у використанні та актуальність представленої інформації, що робить його цінним інструментом у сфері громадського харчування.

Актуальність цієї роботи полягає в прагненні запропонувати цифрове рішення, що відповідає сучасним потребам користувачів — отримувати

якісну, структуровану та корисну інформацію в зручному форматі. Надалі функціональність додатку може бути розширена новими можливостями: персоналізованими рекомендаціями, інтеграцією з навігаційними сервісами, системою бронювання столиків тощо.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1 Основні поняття та визначення мобільного додатку

Сучасний світ неможливо уявити без мобільних додатків. Вони стали невід'ємною частиною нашого повсякденного життя, змінюючи способи взаємодії з інформаційними технологіями. Мобільні додатки — це спеціалізовані програми, розроблені для роботи на смартфонах, планшетах і інших мобільних пристроях. Вони надають користувачам широкий спектр функціональних можливостей, допомагаючи вирішувати найрізноманітніші завдання.

Основні характеристики мобільних додатків:

1. Платформозалежність та кросплатформеність**:** Розробка мобільних додатків може бути двох основних типів: нативні додатки для конкретних операційних систем або кросплатформені технології, що дозволяють створювати додатки, які працюють на кількох платформах одночасно. Використання таких технологій, як React Native, Flutter чи Xamarin, дозволяє значно зменшити витрати на розробку та обслуговування додатків.

2. Функціональність та інтерфейс**:** Мобільні додатки можуть виконувати різноманітні функції — від простих утиліт, як калькулятори або записники, до складних систем для управління фінансами чи проектами. Ключовим моментом є інтуїтивно зрозумілий інтерфейс, що забезпечує легкий доступ до основних функцій.

Класифікація мобільних додатків:

- Нативні додатки**:** Ці додатки розроблені для конкретних операційних систем, надають максимальну продуктивність і повний доступ до апаратних можливостей пристрою.
- Веб-додатки: Ці додатки працюють через браузер, не потребують інсталяції, і мають високу кросплатформність.
- Кросплатформені додатки**:** Єдина кодова база, що дозволяє запускати додатки на кількох платформах одночасно.

Соціально-економічний контекст мобільних додатків**:**

Мобільні додатки змінили багато аспектів нашого життя, від електронної комерції та фінансових послуг до освіти, охорони здоров'я та розваг.

Особливості сучасних мобільних додатків:

- Інтеграція з різними сервісами:Сучасні додатки активно використовують API для інтеграції з соціальними мережами, хмарними сервісами, платіжними системами.
- Переваги мобільних додатків**:**
    - Цілодобова доступність
    - Персоналізація під конкретного користувача
    - Швидкий доступ до функцій
    - Легкість у використанні

1.2 Порівняння нативної та кросплатформеної розробки

Вибір між нативною та кросплатформеною розробкою є одним з найважливіших рішень при створенні мобільного додатку. Кожен з підходів має свої переваги та недоліки, які впливають на кінцеву якість продукту.

- **Нативна розробка:** Це традиційний підхід, коли програмне забезпечення створюється для конкретної операційної системи за допомогою рідних інструментів і мов програмування. Цей підхід забезпечує високу продуктивність і максимальний доступ до апаратних можливостей пристрою. Він є оптимальним для додатків з високими вимогами до швидкодії і графіки.
- **Кросплатформена розробка:** Цей підхід дозволяє створювати додатки за допомогою єдиного коду, який працює на кількох платформах. Приклади таких технологій — це React Native, Flutter та інші. Кросплатформені технології значно знижують витрати на розробку і дозволяють швидше вивести продукт на ринок.

Переваги кросплатформеної розробки:

- Зменшення витрат на розробку
- Можливість одночасного випуску додатків для кількох платформ
- Спрощене обслуговування та оновлення додатків
- Уніфікований підхід до розробки

Недоліки:

- Нижча продуктивність порівняно з нативними додатками
- Обмеження в доступі до деяких апаратних функцій
- Складнощі з реалізацією складних анімацій та інтерактивних елементів

1.3 Переваги та недоліки використання React Native

React Native став популярним фреймворком для кросплатформеного розвитку мобільних додатків завдяки своїм численним перевагам. Він дозволяє використовувати JavaScript і React для створення високопродуктивних додатків, що надають майже нативну продуктивність.

- **Архітектура React Native:**Архітектура базується на компонентному підході, що дозволяє створювати адаптивні компоненти для різних платформ. Однією з головних особливостей є механізм "міст" (bridge), який забезпечує комунікацію між JavaScript і нативними компонентами.
- **Життєвий цикл компонентів:***Як і у веб-розробці, життєвий цикл компонентів в React Native дозволяє ефективно керувати станом та оновленням UI.

Переваги React Native:

- Швидка розробка завдяки єдиному коду
- Легкість у підтримці та оновленні
- Висока продуктивність завдяки нативним модулям

Недоліки:

- Іноді складно досягти ідеального результату у всіх платформах
- Обмежений доступ до деяких апаратних функцій пристроїв

РОЗДІЛ 2. АНАЛІЗ РИНКУ ТА ВИМОГ ДОДАТКУ

2.1 Огляд мобільних додатків для пошуку закладів громадського харчування: функціонал, інтерфейс, відгуки користувачів

Google Maps. є одним з найбільш популярних додатків для пошуку закладів громадського харчування.

- **Функціонал**: пошук закладів за різними критеріями, навігація до обраних місць, відгуки користувачів, можливість переглядати меню та фотографії закладів.
- **Інтерфейс**: зручний, з можливістю швидкого фільтрування за категоріями (кафе, ресторани, бари) та типами кухні.
- **Відгуки**: користувачі відзначають точність інформації, але інколи зустрічаються неточності в даних про години роботи та наявність столиків.

TripAdvisor Популярний додаток для пошуку та оцінки ресторанів і кафе.

- **Функціонал**: інтеграція з картою для пошуку закладів, фільтри за типом кухні, ціною та рейтингами, можливість залишати відгуки та переглядати меню.
- **Інтерфейс**: простий та інтуїтивно зрозумілий, з чітким поділом на категорії.
- **Відгуки**: додаток високо оцінюють за детальність відгуків, хоча деякі користувачі відзначають надмірну кількість реклами.

Yelp надає можливість знайти місця для харчування та переглядати відгуки про них.

- **Функціонал**: пошук закладів за категоріями, перегляд фотографій, відгуків, меню, а також можливість замовляти столики або доставку.
- **Інтерфейс**: легкий у використанні, з розширеними можливостями фільтрації.
- **Відгуки**: користувачі вважають Yelp дуже корисним для пошуку маловідомих, але хороших закладів, хоча іноді відгуки можуть бути суб'єктивними.

Zomato Додаток для пошуку ресторанів, кафе та інших закладів громадського харчування, з можливістю перегляду меню та фото.

- **Функціонал**: інтерфейс для пошуку закладів, фільтрація за кухнею, ціною, розташуванням, а також можливість замовлення їжі онлайн.
- **Інтерфейс**: простий, але з великою кількістю даних, що потребують часу для вивчення.
- **Відгуки**: користувачі цінують функцію замовлення через додаток, але іноді наявність старих відгуків впливає на загальний рейтинг.

Untappd є спеціалізованим додатком для пошуку закладів, що пропонують різноманітні види пива.

- **Функціонал**: пошук барів, пабів і ресторанів за типами пива, перегляд меню з сортами пива, можливість додавати свої відгуки та оцінки. Додаток також дозволяє слідкувати за новими випусками пива та подіями, пов'язаними з пивом.
- **Інтерфейс**: стильний та простий у використанні, з можливістю швидкого пошуку та фільтрації.
- **Відгуки**: користувачі люблять можливість додавати фотографії пива, оцінювати його смакові якості і ділитися відгуками з іншими

користувачами. Однією з найбільших переваг є інтеграція з соціальними мережами для обміну відгуками.

-

2.2 Виявлення основних потреб користувачів і вимог до функціональності

Виявлення основних потреб користувачів та визначення вимог до функціональності мобільного додатку для пошуку закладів є критичним етапом розробки. Ось основні потреби користувачів:

- **Простота пошуку** Користувачі хочуть швидко знайти заклад, що відповідає їх вимогам (тип кухні, ціна, рейтинг). Для цього важливо мати зручні фільтри та можливість швидкого доступу до результатів пошуку.
- **Реальні відгуки та оцінки** Корисними є чесні та об'єктивні відгуки інших користувачів. Чим більше відгуків, тим легше зробити вибір.
- **Чітка інформація про заклад** Користувачі потребують актуальну інформацію про години роботи, меню, можливість забронювати столик або замовити доставку.
- **Інтерактивні карти та навігація** Мобільний додаток має містити інтерактивну карту для зручної навігації до вибраного закладу.
- **Локалізація та персоналізація** Залежно від місця розташування користувача, додаток має пропонувати найближчі заклади з можливістю налаштування мови та валюти.

2.3 Вимоги до функціональності

1. **Пошук закладів** Пошук за категоріями (ресторани, кафе, бари тощо), можливість фільтрувати за ціною, типом кухні, рейтингом.

2. **Інтерактивна карта** Можливість перегляду результатів пошуку на карті та отримання маршруту до вибраного закладу.
3. **Відгуки користувачів** Відображення реальних відгуків користувачів з можливістю залишати власні оцінки та коментарі.
4. **Інформація про заклад** Відображення години роботи, меню, ціни, доступність столиків, наявність онлайн-замовлення чи доставки.
5. **Персоналізація** Можливість збереження улюблених місць, налаштування повідомлень про акції або зміни в розкладі.

РОЗДІЛ 3. ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКА НА REACT NATIVE

3.1 Визначення основних функцій додатку пошуку та оцінки закладів

Сучасний розвиток цифрових технологій активно впливає на спосіб, яким ми шукаємо, оцінюємо та взаємодіємо з різними закладами громадського харчування. Розробка мобільного додатку для пошуку, оцінки та взаємодії з ресторанними закладами має на меті спрощення цього процесу та забезпечення користувачів ефективними інструментами для пошуку та вибору закладів за різними критеріями. У контексті досліджуваного проекту розробка мобільного додатку передбачає створення інтуїтивно зрозумілого інтерфейсу для пошуку, відображення результатів на карті, а також систему оцінок і відгуків для користувачів.

Основні функції додатку включають:

1. **Пошук закладів**: Можливість швидко знаходити заклади на карті за допомогою пошукових запитів. Для цього використовуються дані з Google Places API, які надають точну інформацію про розташування, назви та категорії закладів.
2. **Фільтрація результатів**: Додаток надає користувачам можливість фільтрувати результати пошуку за різними критеріями, такими як тип закладу, відстань, рейтинг і ціна. Це дозволяє користувачам знайти найбільш відповідні варіанти за допомогою простого і швидкого інтерфейсу.
3. **Відображення закладів на карті**: Завдяки інтеграції з Google Maps, користувачі можуть бачити всі заклади на інтерактивній карті з

можливістю вибору та детального перегляду інформації про конкретні місця.

4. **Оцінки та відгуки**: Користувачі можуть оцінювати заклади за різними параметрами (якість обслуговування, атмосфера, їжа тощо) та залишати відгуки, що допомагає іншим користувачам робити обґрунтовані вибори.
5. **Підтримка профілю користувача**: Користувачі можуть створювати та редагувати свої профілі, зберігати улюблені заклади та історію своїх відвідин для зручності при подальших пошуках.

Завдяки таким функціям додаток забезпечує користувачам повний контроль над пошуком та вибором закладів, а також дозволяє на основі зібраної аналітики покращувати досвід використання.

Технологічне рішення для розробки цього додатку засноване на використанні **React Native Expo**, що забезпечує крос-платформенну сумісність між iOS та Android, а також інтеграції з **MySQL** для зберігання даних користувачів та історії їх відгуків.

3.2 Проєктування архітектури системи та розробка структури даних: моделювання закладів, категорій, відгуків користувачів

У даному розділі описано архітектурні рішення для створення мобільного додатку на платформі **React Native Expo**, а також модель даних, необхідну для забезпечення ефективної роботи додатку з пошуку та оцінки закладів громадського харчування.

Мобільний додаток використовує трирівневу архітектуру:

1. **Клієнтський рівень (Frontend)**:

- Використовується **React Native** (Expo) для розробки крос-платформеного мобільного додатку. Для забезпечення взаємодії з користувачем використовуються бібліотеки **react-navigation**, **expo-linear-gradient**, **react-native-animatable**, а також підтримка асинхронного зберігання даних через **@react-native-async-storage/async-storage**.
- **Google Places API** інтегрується для отримання інформації про заклади, що дозволяє здійснювати пошук закладів, отримувати їх точне місцезнаходження та іншу важливу інформацію.

2. **Рівень сервісів (Backend)**:

- Для управління даними користувачів та їх аутентифікацією використовується **MySQL** база даних, де зберігаються інформація про користувачів, їх профілі та відгуки.
- Запити до бази даних здійснюються через **Java** сервер, що забезпечує швидкість обробки запитів і високу надійність.

3. **Рівень даних**:

- Структура бази даних включає таблиці для зберігання інформації про **заклади** (назва, категорія, місцезнаходження, рейтинг), **відгуки користувачів** (підпис, текст відгуку, оцінка) та **користувачів** (ім’я, email, історія відгуків).
- Для зберігання структури даних використовується **MySQL**, що дозволяє швидко обробляти запити і підтримувати високу стабільність при великій кількості одночасних користувачів.

База даних організована у вигляді зв’язаних таблиць. Основні таблиці:

Таблиця Users:

Поле	Тип	Опис
id	INT (PK)	Унікальний ідентифікатор
email	VARCHAR	Адреса електронної пошти
password_hash	VARCHAR	Хеш паролю
name	VARCHAR	Ім’я користувача

Таблиця 1.1

Таблиця Places:

Поле	Тип	Опис
id	INT (PK)	Унікальний ID закладу
name	VARCHAR	Назва закладу
description	TEXT	Опис
address	VARCHAR	Адреса
latitude	DOUBLE	Географічна широта

Таблиця 1.2

Таблиця favorites:

Поле	Тип	Опис
id	INT (PK)	Унікальний ID
user_id	INT (FK)	Користувач, який додав до улюбленого
place_id	INT (FK)	Заклад, доданий до обраного

Таблиця 1.3

3.3 Інтерфейс користувача, прототипи екранів додатку.

1. Екран входу в обліковий запис (Login)Це початковий екран авторизації, де користувачеві пропонується ввести ім’я користувача та пароль. Центрована назва “RestSearcher” зверху одразу показує, як називається застосунок. Під полями вводу розташована кнопка “Login” для входу до системи, а також посилання на реєстрацію для нових користувачів. Екран реалізовано максимально просто та зрозуміло, що дозволяє користувачу без зусиль увійти до свого облікового запису. Такий підхід зменшує бар’єри для початку використання застосунку, що важливо для користувацького досвіду.

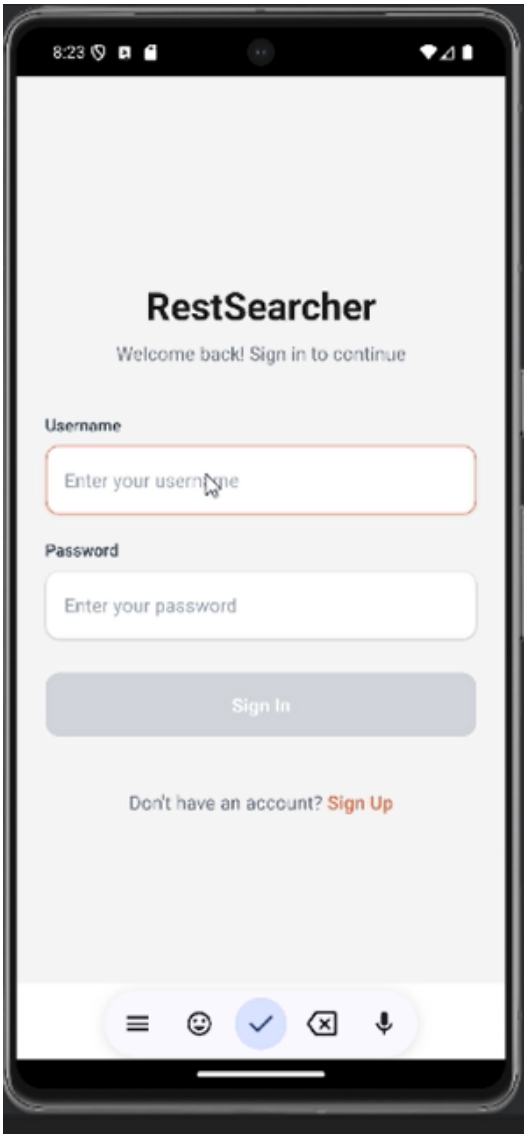

Рис.1 Login Screen

2. **Карта з закладами (Map Screen)** На цьому екрані відображається інтерактивна карта з великою кількістю маркерів, які позначають розташування різних закладів громадського харчування. Карта охоплює територію Києва, і дає змогу візуально оцінити кількість доступних місць поблизу. У нижній частині екрана розміщена кнопка “Go to favorites”, що забезпечує швидкий перехід до списку обраного.

Цей екран є основним інструментом пошуку нових закладів. Користувач може натискати на маркери для перегляду детальнішої інформації (якщо реалізовано), що робить навігацію інтуїтивно зрозумілою. Також внизу присутня панель навігації з вкладками, яка дозволяє перемикатись між основними розділами застосунку.[2]

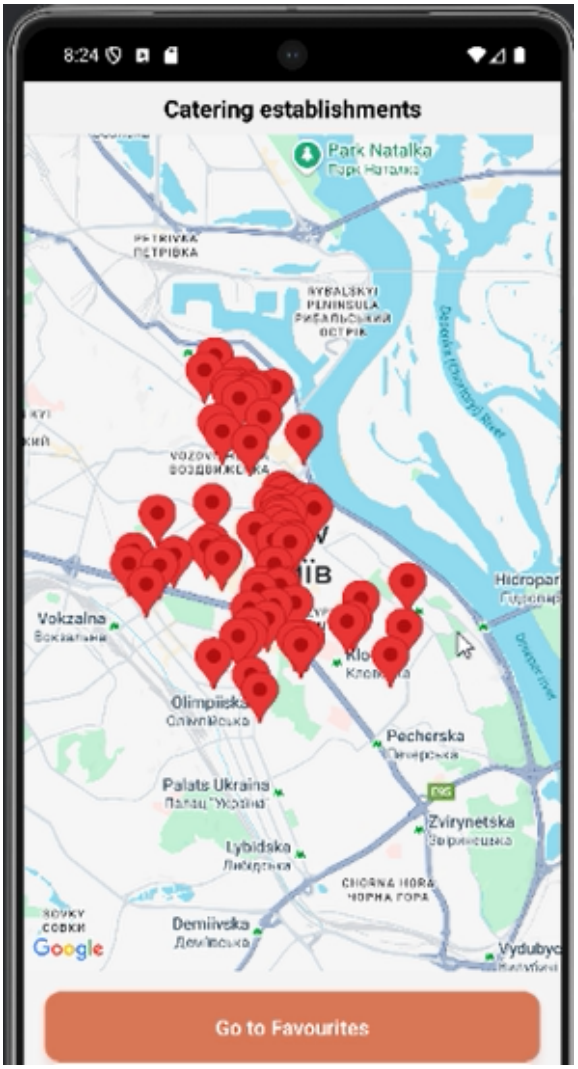

Рис.2 Map Screen

3. **Екран обраних закладів (Favorites)**Цей екран показує список закладів, які користувач додав до обраного. Кожен елемент у списку містить назву закладу та його адресу, а також кнопку “Remove”, яка дозволяє видалити заклад із обраного. Візуально блоки обраних закладів мають чітке відокремлення одне від одного, що дозволяє легко орієнтуватися у списку навіть при великій кількості об'єктів.

Цей розділ додатку покликаний забезпечити швидкий доступ до улюблених місць без потреби повторного пошуку на карті. Завдяки функції видалення, користувачі можуть легко оновлювати список згідно зі своїми вподобаннями.

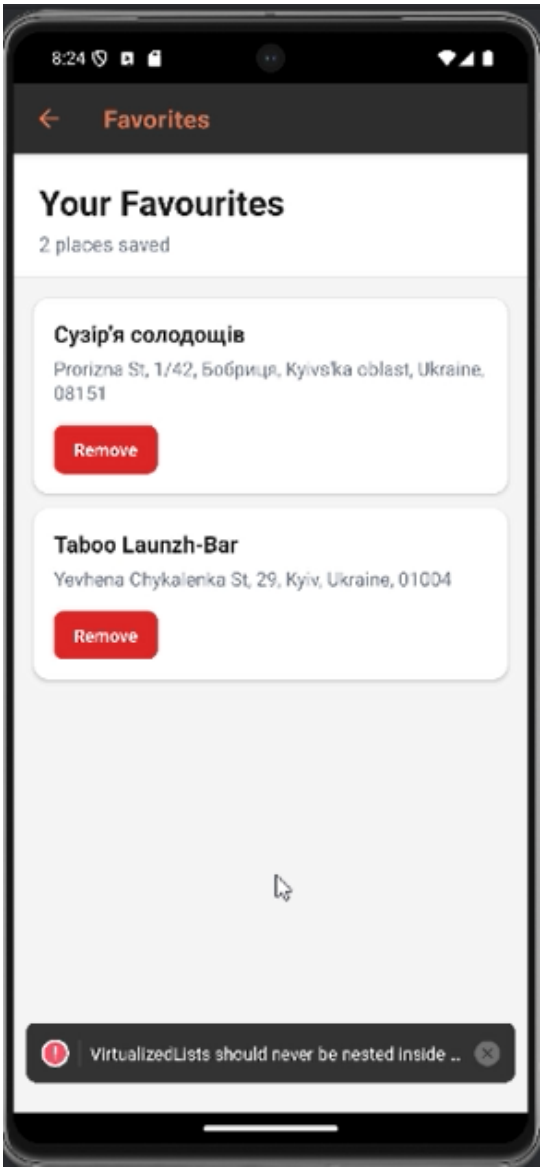

Рис.3 Favorites Screen

4. Екран профілю користувача (Profile)Тут користувач може переглядати й редагувати свої особисті дані. У верхній частині відображено ім'я користувача, а нижче — блок "Personal Information", який включає поля для імені, електронної пошти, номера телефону та біографії. Також доступні кнопки "Edit Profile" для редагування інформації та "Logout" для виходу з облікового запису. Інтерфейс виконано в темних тонах, що додає сучасного вигляду.

Цей екран дозволяє користувачу підтримувати актуальність своїх даних, а також забезпечує безпеку та контроль за доступом до облікового запису через функцію виходу.

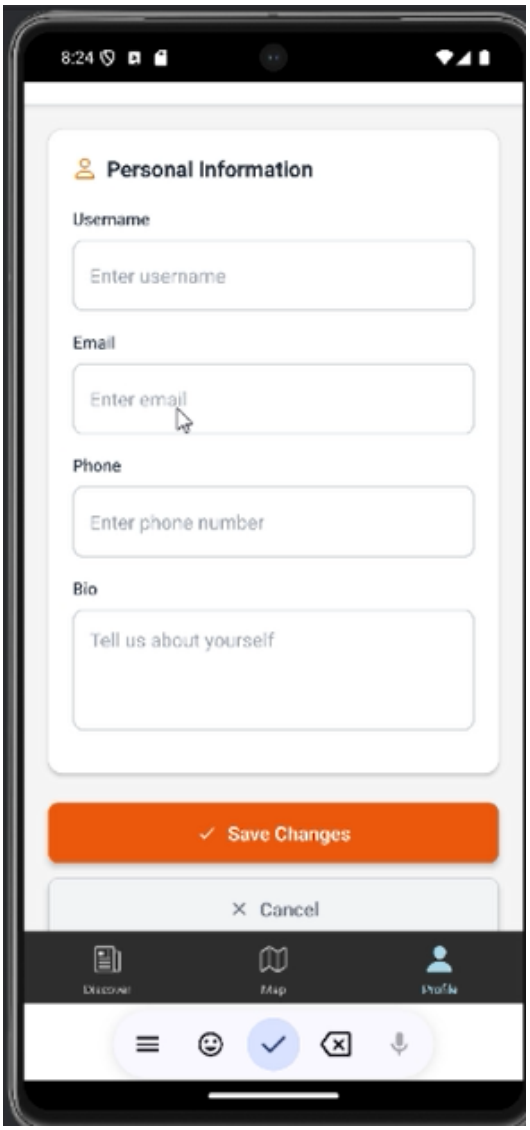

Рис.4 Profile Screen

5. Екран інформації о закладах (Place Details Screen)

Цей екран показує основну інформацію обраного закладу.

Його рейтинг на основі деякої кількості відгуків. Точну адресу ресторану, кафе, тощо. Години роботи, якщо вони вказані власником. Контактний номер для зв’язку для бронювання столиків або уточнення інформації у

закладі.

А також посилання на сайт, якщо є.

В нижній частині панелі є кнопка «Додати до обраного», яка надає змогу додати заклад в улюблені місця громадського харчування.

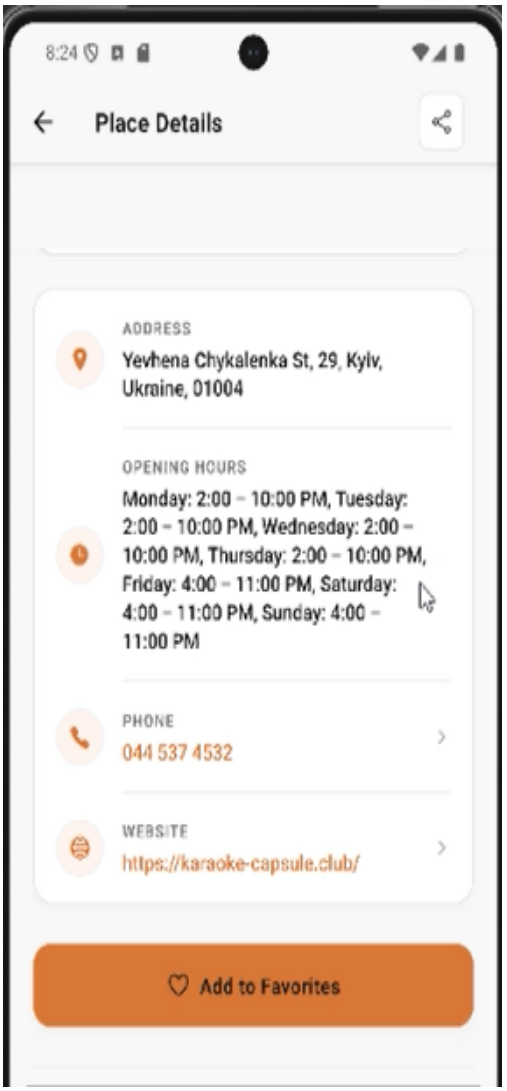

Рис.5 Place Details

3.4 Код серверної частини додатку

1.GoogleMapsService.java

Цей код взаємодіє з Google Place Api тим самим отримує інформацію та список закладів у вказаних координатах(на даний момент там вказано

координати центру міста Київ та RADIUS = 3000 задля меншої нагрузки на мою комп’ютерну систему.

Функція getPlaceDetailsFromGoogleMaps(String id) забезпечує отримання детальної інформації про місце за його унікальним place_id. Вона створює URL з потрібними параметрами, виконує GET-запит та отримує JSON-відповідь. Результат передається до функції mapToPlaceDto, яка перетворює JSON у структурований об'єкт PlaceDto для подальшого використання.

Допоміжна функція mapToPlaceDto обробляє дані, отримані від Google API. Вона аналізує наявність різних елементів: назви закладу, категорії, оцінки, кількості відгуків, адреси, географічних координат, годин роботи, контактного номера, веб-сторінки та зображень. Наприклад, при наявності рейтингу створюється опис формату "Рейтинг 4.3 на основі 120 відгуків". Для фотографій функція формує посилання через photo_reference за допомогою додаткового методу buildPhotoUrl.

Програмний код наведений у додатку А.

2. UserServiceImpl.java

Клас UserServiceImpl є сервісом, який реалізує логіку, пов’язану з роботою користувачів у системі. Він помічений анотацією @Service, що означає його приналежність до сервісного шару в архітектурі додатку. У цьому класі відбувається керування реєстрацією, аутентифікацією, а також додаванням і видаленням улюблених закладів користувача.

У класі використовується механізм автоматичної ін’єкції залежностей за допомогою анотації @Autowired. Це дозволяє автоматично підключати потрібні сервіси. Зокрема, використовується UserRepository для взаємодії з базою даних користувачів, PasswordEncoder для безпечного збереження паролів, JwtProvider для генерації JWT-токенів після успішної аутентифікації,

а також GoogleMapService, який звертається до Google Maps API для перевірки правильності ідентифікатора закладу.

Метод registerUser використовується для реєстрації нового користувача. Перед створенням облікового запису перевіряється, чи не зайнятий логін — якщо так, кидається виняток UsernameAlreadyInUseException. Якщо логін вільний, створюється новий об’єкт користувача, його пароль хешується, і користувач зберігається в базу даних.

Метод authenticate відповідає за вхід користувача в систему. Спочатку здійснюється пошук користувача в базі даних за логіном. Якщо його не знайдено, кидається виняток NoUserFoundException. Якщо користувач існує, перевіряється пароль. У разі помилки в паролі викидається CredentialException. Якщо всі дані правильні, генерується JWT-токен, який повертається клієнту.

Метод getFavouritePlaces повертає список улюблених місць для поточного користувача. Щоб дізнатися, хто саме авторизований, використовується метод getUserFromSecurityContext, який витягує користувача з контексту безпеки. Після цього повертається список закладів, що збережені як улюблені.

Метод addFavouritePlaceById додає нове улюблене місце за його ідентифікатором. Спочатку перевіряється, чи цей placeId дійсний за допомогою виклику до Google Maps API. Якщо місце не має назви або адреси, викидається виняток InvalidPlaceIdException. У разі валідного ідентифікатора місце додається до списку користувача, і зміни зберігаються в базі даних.

Аналогічним чином працює метод deletePlaceById, який видаляє улюблене місце зі списку. Перед видаленням також перевіряється валідність placeId, а потім з поточного користувача видаляється відповідний елемент із переліку улюблених.

Метод isPlaceIdValid є допоміжним і перевіряє, чи існує місце з таким placeId у Google Maps. Якщо API не повертає ні назви, ні адреси, такий ідентифікатор вважається некоректним, і викидається відповідний виняток.

Метод getUserFromSecurityContext відповідає за отримання поточного авторизованого користувача з контексту Spring Security. З нього береться ім’я користувача, після чого через репозиторій витягується об'єкт користувача. Якщо такого користувача не знайдено, викидається помилка.

Загалом клас UserServiceImpl виконує всі ключові дії, пов’язані з життєвим циклом користувача: реєстрацію, вхід у систему, збереження улюблених закладів і їх видалення. Він тісно інтегрується з механізмом безпеки, з JWT-токенами, а також із зовнішнім API Google Maps для перевірки валідності закладів.

Програмний код наведений у додатку Б.

ВИСНОВКИ:

У ході виконання роботи були вирішені наступні завдання:

По-перше, визначено ключові поняття та роль мобільних додатків у сфері громадського харчування. Проведено аналіз літературних джерел та сучасних сервісів, що дало змогу сформувати уявлення про потреби користувачів і актуальність подібних рішень. Показано, що такі додатки сприяють підвищенню якості обслуговування, покращують комунікацію з клієнтами та забезпечують зручний доступ до інформації про заклади.

По-друге, проаналізовано особливості нативної та кросплатформеної розробки, порівняно їх ефективність. Розглянуто сильні та слабкі сторони кожного підходу. Обґрунтовано вибір React Native як оптимального рішення для реалізації даного проєкту, оскільки він забезпечує одночасну підтримку Android та iOS, зменшує витрати часу і ресурсів на розробку.

По-третє, розглянуто основні функції мобільних додатків для пошуку та оцінювання закладів. На основі аналізу популярних сервісів, таких як Google Maps та Untappd, було визначено ключові функціональні компоненти: пошук, відображення на мапі, перегляд детальної інформації, залишення відгуків і оцінок, а також збереження улюблених місць. Ці функції реалізовано у створеному додатку.

По-четверте, досліджено інтеграцію фронтенд-рішення на React Native з Java-бекендом та базою даних MySQL. Створено надійну архітектуру клієнт-серверної взаємодії, реалізовано REST API на Spring Boot, авторизацію з JWT, обробку запитів до бази даних, пошук закладів, додавання відгуків та рейтингів.

По-п’яте, оцінено переваги, недоліки та можливості створеного рішення у контексті реальних потреб користувачів. Згідно з результатами тестування, додаток є зручним у використанні, підтримує усі основні

сценарії, має інтуїтивно зрозумілий інтерфейс і може бути корисним як туристам, так і місцевим мешканцям. Також визначено напрями подальшого розвитку, зокрема впровадження системи рекомендацій, розширення мовної підтримки та адаптація для планшетів.

Таким чином, усі поставлені завдання дослідження було виконано повністю. Розроблений додаток є практичним інструментом, що має прикладне значення та потенціал для подальшого вдосконалення. Отримані результати можуть бути використані як основа для майбутніх досліджень у сфері мобільної розробки, інтеграції картографічних сервісів та покращення взаємодії з користувачем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. React Native Documentation URL: https://reactnative.dev/docs/getting-started

2. Як зробити мобільний додаток за допомогою JS. Шлях React Native URL: https://dou.ua/lenta/articles/how-to-learn-react-native/

3. React Native Expo Documentation URL: https://docs.expo.dev/get-started/introduction/

4. Інтенсивне занурення у JavaScript URL: https://www.udemy.com/course/intensive-js/

5. Complete Java Programming Tutorial: From Beginner to Advance URL: https://www.udemy.com/course/complete-java-programming-tutorial-from-beginner-to-advance/

6.Android Studio Introduction URL: https://developer.android.com/develop

7.Android Studio Documentation URL: https://source.android.com/docs

8.MySQL Documentation URL: https://dev.mysql.com/doc/

9.IntelliJ Idea Documentation URL: https://www.jetbrains.com/help/idea/getting-started.html

10. Basic concepts of Android development : Build Your First App URL: https://www.udemy.com/course/the-beginners-path-to-android-build-your-first-app/

11. Introduction to Databases and SQL Querying URL: https://www.udemy.com/course/introduction-to-databases-and-sql-querying/

12. Advanced Databases and SQL Querying URL; https://www.udemy.com/course/advanced-tsql-querying-using-sql-2014/

13. React Native Basic URL: https://www.udemy.com/course/react-native-basic/

14. React Native Course For React Developers With Projects(2023) https://www.udemy.com/course/react-native-full-course/

15. Java Documentation URL: https://docs.oracle.com/en/java/

16. React Navigation Documentation URL: https://reactnavigation.org/docs/getting-started/

17. React Native Expo Create Documentation URL: https://docs.expo.dev/more/create-expo/

18. React Native Async Storage Documentation URL: https://reactnative.dev/docs/asyncstorage

19. JSON web tokens documentation URL: https://auth0.com/docs/secure/tokens/json-web-tokens

20. Spring Data JPA. Spring Data JPA. URL: https://spring.io/projects/spring-data-jpa

21.В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.

22.Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.

23.Яскевич, Владислав Олександрович (2023) *JavaScriptбібліотека React Навчальний посібник для студентів спеціальності Комп'ютерні науки* Київський університет імені Бориса Грінченка, Україна, Київ. https://elibrary.kubg.edu.ua/id/eprint/48587/

ДОДАТКИ

1. GoogleMapsService.java (Додаток А)

```
package org.example.service;

import org.example.dto.PlaceDto;
import org.example.dto.PlaceMarker;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.*;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;

import java.net.URI;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Service
public class GoogleMapService {
                private static final String GOOGLE_MAPS_URL =
"https://maps.googleapis.com/maps/api/place/details/json";
            private static final String GOOGLE_MAPS_PHOTO_URL =
"https://maps.googleapis.com/maps/api/place/photo";

                            private final String apiKey =
"AIzaSyCtyCHoLK5orbeDrsODnVTuWqwad0be3eQ";
```

```
    private final RestTemplate restTemplate = new RestTemplate();

                                private final String BASE_URL =
"https://maps.googleapis.com/maps/api/place/nearbysearch/json";

     private final List<String> FOOD_TYPES = List.of("restaurant", "cafe", "bar",
"bakery");

    private final String KIEV_LOCATION = "50.4501,30.5234";
    private final int RADIUS = 3000;

    public PlaceDto getPlaceDetailsFromGoogleMaps(String id) {
        RestTemplate restTemplate = new RestTemplate();
        String url = UriComponentsBuilder.fromUriString(GOOGLE_MAPS_URL)
                .queryParam("place_id", id)
                .queryParam("key", apiKey)
                .toUriString();

        HttpHeaders headers = new HttpHeaders();
        headers.set("Accept", "application/json");

                    RequestEntity<Void> request = new RequestEntity<>(headers,
HttpMethod.GET, URI.create(url));

        ResponseEntity<Map<String, Object>> response = restTemplate.exchange(
                request,
                new ParameterizedTypeReference<>() {
                }
        );
```

```
        Map<String, Object> responseBody = response.getBody();

        return mapToPlaceDto(responseBody);
    }

    private PlaceDto mapToPlaceDto(Map<String, Object> responseBody) {
        PlaceDto placeDto = new PlaceDto();

        if (responseBody != null && responseBody.containsKey("result")) {
                            Map<String, Object> result = (Map<String, Object>)
responseBody.get("result");


            if (result.containsKey("name")) {
                placeDto.setName((String) result.get("name"));
            }


                                if (result.containsKey("types") && ((List<String>)
result.get("types")).size() > 0) {
                String type = ((List<String>) result.get("types")).get(0);
                placeDto.setType(formatType(type));
            }


                                            if (result.containsKey("rating") &&
result.containsKey("user_ratings_total")) {
                double rating = (double) result.get("rating");
```

```
            int reviewCount = (int) result.get("user_ratings_total");
                placeDto.setDescription_short(String.format("Рейтинг %.1f на основі
%d відгуків", rating, reviewCount));
            }


            if (result.containsKey("formatted_address")) {
                placeDto.setAddress((String) result.get("formatted_address"));
            }


                        if (result.containsKey("geometry") && ((Map<String, Object>)
result.get("geometry")).containsKey("location")) {
                Map<String, Object> location = (Map<String, Object>) ((Map<String,
Object>) result.get("geometry")).get("location");
                Map<String, Double> locationMap = new HashMap<>();
                locationMap.put("latitude", (Double) location.get("lat"));
                locationMap.put("longitude", (Double) location.get("lng"));
                placeDto.setLocation(locationMap);
            }


                if (result.containsKey("opening_hours") && ((Map<String, Object>)
result.get("opening_hours")).containsKey("weekday_text")) {
                    List<String> weekdayText = (List<String>) ((Map<String, Object>)
result.get("opening_hours")).get("weekday_text");
                placeDto.setOpening_hours(String.join(", ", weekdayText));
            }
```

```
        if (result.containsKey("formatted_phone_number")) {
            placeDto.setPhone((String) result.get("formatted_phone_number"));
        }


        if (result.containsKey("website")) {
            placeDto.setWebsite((String) result.get("website"));
        }


                if (result.containsKey("photos") && ((List<Map<String, Object>>)
result.get("photos")).size() > 0) {
                    Map<String, Object> firstPhoto = ((List<Map<String, Object>>)
result.get("photos")).get(0);
            if (firstPhoto.containsKey("photo_reference")) {
                String photoRef = (String) firstPhoto.get("photo_reference");
                String photoUrl = buildPhotoUrl(photoRef);
                placeDto.setImage_url(photoUrl);
            }
        }
    }

    return placeDto;
  }

  private String formatType(String type) {
    return switch (type) {
        case "restaurant" -> "Ресторан";
```

```
            case "cafe" -> "Кафе";
            case "bar" -> "Бар";
            case "night_club" -> "Клуб";
            case "bakery" -> "Булочна";
            default -> "Заклад";
        };
    }

    private String buildPhotoUrl(String photoReference) {
                                                                    return
UriComponentsBuilder.fromUriString(GOOGLE_MAPS_PHOTO_URL)
                .queryParam("photoreference", photoReference)
                .queryParam("maxwidth", 400)
                .queryParam("key", apiKey)
                .toUriString();
    }

    public List<PlaceMarker> getAllPlaces() {
        List<PlaceMarker> allMarkers = new ArrayList<>();

        for (String type : FOOD_TYPES) {
                                                        String      url      =
String.format("%s?location=%s&radius=%d&type=%s&key=%s",
                    BASE_URL, KIEV_LOCATION, RADIUS, type, apiKey);

                        ResponseEntity<Map>  response  =  restTemplate.getForEntity(url,
Map.class);

            if (response.getStatusCode() == HttpStatus.OK && response.getBody() !=
```

```
null) {
                List<Map<String, Object>> results = (List<Map<String, Object>>)
response.getBody().get("results");

            for (Map<String, Object> result : results) {
                        Map<String, Object> geometry = (Map<String, Object>)
result.get("geometry");
                         Map<String, Object> location = (Map<String, Object>)
geometry.get("location");

                String placeId = (String) result.get("place_id");
                double lat = ((Number) location.get("lat")).doubleValue();
                double lng = ((Number) location.get("lng")).doubleValue();

                PlaceMarker marker = new PlaceMarker();
                marker.setPlaceId(placeId);
                marker.setType(type);
                marker.setLat(lat);
                marker.setLng(lng);

                allMarkers.add(marker);
            }
        }
    }

    return allMarkers;
  }


}
```

2. UserServiceImpl.java (Додаток Б)

```
package org.example.service;

import org.example.dto.PlaceDto;
import org.example.entity.User;
import org.example.exceptions.InvalidPlaceIdException;
import org.example.exceptions.NoUserFoundException;
import org.example.exceptions.UsernameAlreadyInUseException;
import org.example.jwt.JwtProvider;
import org.example.response.AddFavouritePlaceResponse;
import org.example.response.DeletePlaceResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import javax.security.auth.login.CredentialException;
import java.util.List;
import java.util.Optional;

@Service
public class UserServiceImpl implements UserService{
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
```

```
    private JwtProvider jwtProvider;
    @Autowired
    private GoogleMapService googleMapService;

    @Override
    public void registerUser(String username, String password) {
        if (userRepository.findUserByUsername(username).isPresent()){
            throw new UsernameAlreadyInUseException();
        }

        User newUser = new User();
        newUser.setUsername(username);
        newUser.setPassword(passwordEncoder.encode(password));

        userRepository.save(newUser);
    }

    @Override
        public String authenticate(String username, String password) throws
CredentialException {
        Optional<User> user = userRepository.findUserByUsername(username);

        if (user.isEmpty()){
            throw new NoUserFoundException(username);
        }

        if (!passwordEncoder.matches(password, user.get().getPassword())) {
            throw new CredentialException("Wrong password");
        }
```

```
        return jwtProvider.generateToken(user.get());
    }

    @Override
    public List<String> getFavouritePlaces() {

        User obtainedUser = getUserFromSecurityContext();

        return obtainedUser.getFavouritePlaces();
    }

    @Override
    public AddFavouritePlaceResponse addFavouritePlaceById(String placeId) {


        isPlaceIdValid(placeId);

        User obtainedUser = getUserFromSecurityContext();
        obtainedUser.getFavouritePlaces().add(placeId);
        userRepository.save(obtainedUser);

        return new AddFavouritePlaceResponse();
    }

    private void isPlaceIdValid(String placeId){
                                                PlaceDto placeDto =
googleMapService.getPlaceDetailsFromGoogleMaps(placeId);
        if (placeDto.getAddress() == null || placeDto.getName() == null){
```

```
            throw new InvalidPlaceIdException(placeId);
        }
    }

    @Override
    public DeletePlaceResponse deletePlaceById(String placeId) {
        isPlaceIdValid(placeId);

        User obtainedUser = getUserFromSecurityContext();
        obtainedUser.getFavouritePlaces().remove(placeId);
        userRepository.save(obtainedUser);

        return new DeletePlaceResponse();
    }


    private User getUserFromSecurityContext(){
                                                Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();

        Optional<User> user = userRepository.findUserByUsername(username);
        if (user.isEmpty()) {
                throw new RuntimeException("Something went wrong while processing
user data");
        }

        return user.get();
```

```
    }
}
```