Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія замовлення їжі»

Виконав: студент 2-го курсу, групи
2КН-21м спеціальності 122 –
Комп`ютерні науки

Павлюк В.О.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри КН
Колодний В.В.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: к.т.н., доцент кафедри КСУ
Юхимчук М. С.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ - 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації Кафедра комп'ютерних наук
Рівень вищої освіти II-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф.. Яровий А.А.

"14" 09 **2022 року**

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Павлюка Владислава Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія замовлення їжі
керівник роботи к.т.н., доцент кафедри КН Колодний В.В.
затверджені наказом вищого навчального закладу від "14 "09 2022 року№ 203
2. Строк подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
вхідні дані : кількість закладів не менше 10 од.; кількість товарів на кожен заклад не менше 15 од., розділені як мінімум на 3 групи; кількість користувачів для одночасної сесії не менше 200 чол.
4. Зміст текстової частини:
Вступ; аналіз сучасного рівня технології замовлення їжі; розробка інформаційної технології замовлення їжі; програмна реалізація інформаційної технології замовлення їжі; економічна частина; висновки; перелік використаних джерел; додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)

Блок-схема алгоритму обробки запитів сервером, алгоритм авторизації та контролю доступу до даних.

6.Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	вико прий
1-3	Колодний В. В., к.т.н., доцент кафедри КН	14.09.2022	14.11.202
4	Бурєннікова Н. В., к. е. н., доц. каф. ЕПВМ	13.09.2022	12.12.202

7.Дата видачі завдання 14.09. 2022 року

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ з/п	Назва етапів Магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз сучасного рівня інформаційних технологій замовлення їжі	14.09.2022р. – 01.10.2022р.
2	Аналіз алгоритмів рекомендацій закладів та страв у них	1.10.2022р. – 16.10.2022р.
3	Практичне застосування та оцінка якості надання рекомендацій щодо закладів та страв у них	17.10.2022р. – 07.11.2022р.
4	Підготовка економічної частини	08.11.2022р. – 21.11.2022р
5	Апробація та/або впровадження результатів дослідження	23.11.2022р. – 01.12.2022р.
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.2022р. – 14.12.2022р.

Студент _____ Павлюк В.О.
(підпис)

Керівник роботи _____ Колодний В.В.
(підпис)

АНОТАЦІЯ

УДК 004.8

Павлюк В. О. Інформаційна технологія замовлення їжі. Магістерська кваліфікаційна робота зі спеціальності 122 – комп'ютерні науки, освітня програма – комп'ютерні науки. Вінниця: ВНТУ, 2022. 105 с.

На укр. мові. Бібліогр.: 16 назв; рис.: 28; табл. 8.

Дана магістерська кваліфікаційна робота присвячена удосконаленню інформаційної технології замовлення їжі онлайн. Проаналізовано особливості систем для замовлення їжі. Також було проведено аналіз переваг та недоліків сучасних систем-аналогів, які використовуються для замовлення доставки їжі. Покращено алгоритм рекомендацій що до підбору закладів для замовлення та товарів, покращення базується на доданню до рекомендацій списку закладів та можливих товарів які можуть бути замовлені на шляху від початкового замовлення до місця доставки. Розроблено алгоритми роботи системи. Побудовано інформаційну модель основних модулів інформаційної технології. Для програмної реалізації мобільного додатку інформаційної системи замовлення їжі було обрано ОС Android.

Графічна частина складається з 2 плакатів.

У економічному розділі розраховано суму витрат на розробку та виготовлення нового технічного рішення, яка складає 499067,91 гривень, спрогнозовано орієнтовану величину витрат по кожній з статей витрат, розраховано чистий прибуток, термін окупності витрат для виробника 2 роки, 7 місяців та економічний ефект для споживача при виконанні даної розробки.

Ключові слова: інформаційна технологія, замовлення їжі, алгоритм рекомендацій, он-лайн мапи.

ABSTRACT

Pavliuk V.O. Information technology of an online assistant for the sale of goods. Master's qualification thesis on specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2022. 105 p.

In Ukrainian language. Bibliographer: 16 titles; fig.: 28; table 8.

This master's thesis is devoted to the development and improvement of the information technology of ordering food online. The features of food ordering systems are analyzed. An analysis of the advantages and disadvantages of modern analogue systems used to order food delivery was also carried out. The recommendation algorithm for selecting establishments for ordering and goods has been improved, the improvement is based on adding to the recommendations a list of establishments and possible goods that can be ordered on the way from the initial order to the place of delivery. Algorithms for system operation have been developed. An information model of the main modules of information technology has been built. The Android OS was chosen for the software implementation of the mobile application of the food ordering information system.

The graphic part consists of 2 posters.

In the economic section, the amount of costs for the development and production of a new technical solution is calculated, which is 499,067.91 hryvnias, the estimated amount of costs for each of the cost items is predicted, the net profit is calculated, the payback period for the manufacturer is 2 years, 7 months, and the economic effect for the consumer during the implementation of this development.

Keywords: information technology, food ordering, recommendation algorithm, online maps.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Оскільки «світ» їсть, отже він змінюється. Ще десятиріччя тому доставка їжі обмежувалась лише піцою та популярною традиційною кухнею. В 2021 році глобальний ринок доставки їжі оцінювався в 150 мільярдів доларів, що в тричі більше ніж він оцінювався в 2017 році. Основний бум росту, очевидно припав на 2020 рік через COVID-19 пандемію, у цей час сервіси для замовлення їжі стали найпопулярнішими по завантаженням на платформах Android і iOS. Але навіть опускаючи цей потужний поштовх ця галузь активно розвивалась.

Розвитку даної галузі також сприяє активний перехід людей безготівкову, а в деяких випадках і на безконтактну оплату. Люди частіше шукають саме такі сервіси і можливості, які дозволять їм отримати бажане за менший обсяг зусиль, саме таким критеріям і відповідають служби доставки їжі [2].

Сьогодні з’являється все більше кухонь на виніс, які займаються тільки приготуванням їжі і не мають свого місця для клієнтів, як звичні нам кафе і ресторани. Такі заклади при такій же кількості клієнтів, як і в ресторані приносять більше прибутку оскільки не потрібно платити за оренду та наймати велику кількість обслуговуючого персоналу. Також є сервіси, які окрім їжі доставляють різного роду речі, і мають можливість замовлення товарів з декількох точок за одне замовлення, що ще більше може зацікавити користувачів.

Також причинами популярності сервісів доставки їжі є економія часу, та широкий вибір закладів. Ви можете отримати їжу з іншого кінця міста максимально швидко, просто зробивши замовлення та забравши її у кур’єра, а заощаджений час витратити на себе, або роботу.

Використання методів штучного інтелекту у такого роду сервісах допомагає як втримати старих клієнтів пропонуючи та підбираючи рекомендації основані на їх попередніх замовлення так і зацікавити нових використовуючи

цільову рекламу. Цікавою можливістю штучного інтелекту в такому сервісі може бути рекомендації для клієнтів що до магазинів, ресторані, та кафе вздовж шляху доставки, така функція збільшить прибуток і зацікавить клієнтів.

Тема магістерської кваліфікаційної роботи є актуально, оскільки тенденція на послуги доставки у продовж останніх років зростає, разом з браком часу у середнього працюючого класу. Також окрім клієнтів сервісу, також зростає кількість закладів, що працюють саме з сервісами доставки. Даний сервіс буде доступний будь-якому користувачу зацікавленому в кур'єрській доставці їжі.

Також впровадження методів штучного інтелекту в сферу доставки їжі забезпечує подальше впровадження і розвиток інформаційних технологій в даному напрямі.

Мета та завдання дослідження. Метою дослідження є підвищення якості рекомендацій, які надаються, щодо вибору закладів та замовлень доставки їжі у них.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз проблем сервісів доставки їжі;
- розглянути існуючі приклади сервісів доставки їжі, їх методи вирішення задачі обслуговування та обрати й обґрунтувати вибір методу, який задовольняє мету магістерської кваліфікаційної роботи;
- розробити удосконалений алгоритм рекомендацій що до вибору закладів та страв у них для інформаційної системи замовлення їжі
- сформулювати стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології доставки їжі;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – це процес замовлення доставки їжі.

Предмет дослідження – це інформаційні технології замовлення доставки їжі.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи та моделі подання знань, методи теорії прийняття рішень, нечіткої логіки, методи тестування програмних продуктів.

Наукова новизна полягає в удосконаленні інформаційної технології замовлення їжі, яка відрізняється від існуючих використанням удосконаленої математичної моделі рекомендаційної системи щодо замовлення їжі, що забезпечує збільшення якості процесу надання рекомендації щодо вибору закладів та замовлень доставки їжі у них.

Практичне значення одержаних результаті полягає в наступному:

1. Удосконалено алгоритм рекомендацій вибору закладів та замовлень доставки їжі у них.
2. Здійснено програмну реалізацію інформаційної технології замовлення їжі.

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці» (м. Вінниця, Україна, 2022 р.) [1],

Публікації. За результатами досліджень опубліковано тези доповіді на конференції [1].

1 АНАЛІЗ СУЧАСНОГО РІВНЯ ТЕХНОЛОГІЇ ЗАМОВЛЕННЯ ЇЖІ

1.1 Аналіз предметної області та існуючих програмних реалізацій

Замовлення доставки їжі це економія сил та часу, за невелику додаткову плату ви отримуєте щойно придбану їжу з улюбленого закладу, не відриваючись від свої справ. Останні роки конкуренція на робочих місцях сильно зросла, люди працюють забуваючи про відпочинок, а іноді і про прийоми їжі, оскільки кожна хвилина цінна і може відобразитись на кар’єрному рості, також є важливим аспектом особисті потреби, наприклад людині можливо необхідний спокійний відпочинок, а не подорож за улюбленим обідом, саме такі ситуації зробили доставку їжі такою невід’ємною частиною нашого життя.

Нинішні сервіси замовлення їжі це заклади харчування з усього міста на відстані декількох кліків по екрану вашого смартфону, це і рекомендації популярних замовлень, кастомні набори, або набори від сервісу, а також зручність і зрозумілість. Це підтримка і зворотній зв’язок на всьому шляху від закладу харчування до ваших дверей, нинішні сервіси повідомляють і про затримку доставки і про відсутність товару, а також корегують замовлення згідно побажань користувача.

Сервіси замовлення їжі в більшості своїй поділяються на два типи, це локального плану, який обумовлений внутрішньою функцією закладу, або мережею закладів, тобто ресторан, кафе, або іншого типу заклад, використовує свій персонал та свої засоби для доставки своєї продукції [2]. Такий тип доставки був одним з єдиних видів доставки їжі десятиліття тому, використання його закладом було обумовлено простотою та можливістю розширити клієнту базу, захвативши клієнтів, які зацікавлені в їх продукції, але не бажають відвідувати даний заклад. Такого роду сервіс не потребує великих постійних затрат, а іноді може бути вигіднішим ніж звичайна діяльність, оскільки для доставки замовлень використовували звичайний персонал закладу, та дешеві

особисті транспортні засоби, велосипед, уживані скутер або автомобіль; а ціна доставки окупала витрати на пальне [4].

Такі заклади спеціалізуються на доставці групи своїх страв, які вони подають на стіл, прикладами є популярна мережа ресторанів “TerraMare” та паста-паб “Tamero”. Замовлення і оплата в даних закладах здійснюється через їхні сайти. Використання сайтів в такого роду закладах також є додатковою рекламою, оскільки користувач може натрапити на заклад просто вбивши «замовлення їжі» у своєму браузері, також дана платформа не потребує завантаження додаткових додатків для доступу до сервісу. Мінуси використання клієнтом сервісу це – відсутність варіативності, як в товарах так і в цінах, користувач вимушений користуватись послугами тільки даних закладів, або мати контакти кожного закладу який його цікавить. Нижче на рисунках 1.1-1.2 зображені скріншоти сайтів.

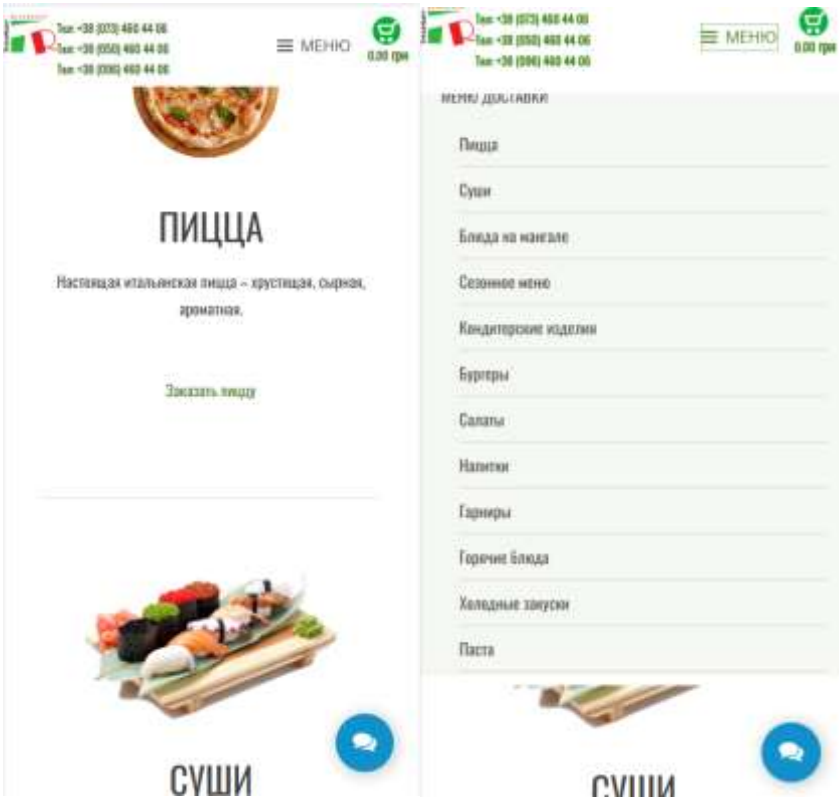


Рисунок 1.1 – Скріншоти головної сторінки сайту і меню закладу “TerraMare”

Хоча дані заклади і має доволі великий вибір страв, однак ви все ще маєте справу тільки з одним закладом з однією ціновою політикою.

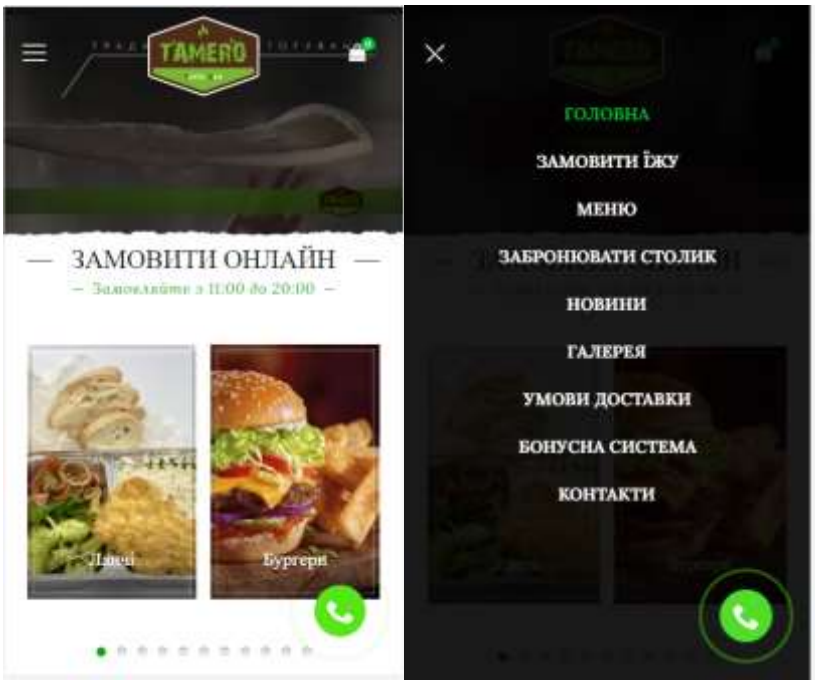


Рисунок 1.2 – Скріншоти сайту закладу “Tamero”

Також з даних прикладів видно що хоч ці заклади надають тільки свою продукцію, вони намагаються. як можна більше розширити свій асортимент, для того щоб охопити більшу цільову аудиторію, і тому стали більш схожі на другий тип сервісів доставки їжі.

Другий тип доставки, можна назвати глобальним, це не мережа закладів чи заклад який доставляє свою продукцію, а кур’єрський сервіс, який доставляє товари з будь-яких закладів.

Такий сервіс, представлений у вигляді додатку на телефон, надає інформацію про заклади та їх товари, або просто приймає замовлення клієнтів з описом потрібних товарів, також такі сервіси зазвичай мають вбудований чат клієнта з кур’єром або оператором чи менеджером, для корегування замовлень при різного роду ситуаціях, а також можливість відстеження руху кур’єру. Особистий кабінет є невід’ємною частиною таких додатків, для кращої роботи з клієнтами, також присутній GPS модуль, який зчитує інформацію про перебування клієнта, для легшого встановлення адреси доставки.

Саме такі сервіси-додатки захопили галузь в останні роки, прикладами таких є іспанськй стартап Glovo та естонська копанія Bolt з її сервісом Bolt Food. Нижче на рисунку 1.3 зображено логотип компанї Glovo

Рисунок 1.3– Логотипи компаній Glovo

Компанії Glovo та Bolt доставляють їжу і товари з багатьох закладів різного роду, але у більшості випадків оновлення асортименту відбувається рідко та влучну, через складність автоматизації даного процесу, що э недоліком та потребує постійної підтримки та спілкуванню з закладами приготування їжі. Однак можливість доставки товарів з будь-яких магазинів, якщо розміри його відповідають кур'єрській рюкзаку, це є цілковитою перевагою, проте для створення сервісу спрямованого тільки на доставку немає необхідності для такого сервісу [4].

На рисунку 1.4 зображено логотип компанії Bolt Food.

Рисунок 1.4 – Логотип компанії Bolt Food

Компанії Glovo та Bolt є свого роду лідерами, але своє місце вони тримають також із-зі своєї різної направленості, сам слоган компанії Glovo

говорить за себе «Ми доставимо все, що поміститься в рюкзак кур’єра», ці дві компанії займаються доставкою не тільки їжі, але й любих товарів, які спроможні доставити їх кур’єри. Саме така політика та можливості вивели ці дві компанії на їх лідерські позиції.

Також окремим пунктом таких систем є їх системи та алгоритми рекомендацій страв та товарів. Вони допомагають клієнтам одразу, які є популярні товари та заклади, або просто ті, що знаходяться неподалік від клієнта. Це є перший рівень рекомендацій який ґрунтується на загальних оцінках користувачів і місті розташування. Алгоритми рекомендацій можна порівняти з айсбергом, де кількість інформації про користувача це глибина дослідження, з якою все більше і більше взаємодіє інформація з усієї системи і тим більше і точніше можна бачити всю картину в цілому [2].

Найпростіші алгоритми рекомендацій можуть базуватись на простих критеріях, таких як: популярність, рейтинг та розташування; це неначе таблиця лідерства у певному районі міста і звісно очевидною стає проблема таких алгоритмів, коли в лідери виходять певні заклади і тільки стосовно них надаються незмінні рекомендації. Хоча проблеми таких рекомендацій очевидні, але плюсом є те що вони починають працювати з самого початку, як тільки користувач запустив додаток.

Більш складніші алгоритми, це збір інформації про користувачів, такі як: статі, місце перебування, пунктів призначення замовлення, а також інформація про самі замовлення: що замовлено, звідки, о котрій годині, кількість замовлень на день чи неділю. Вся ця інформація оброблюється штучним інтелектом і при достатній інформації починає впливати на рекомендації, при чому чим більше інформації тим більш точні рекомендації. Одразу зрозумілий недолік таких алгоритмів – чи менше інформації тим менш точні рекомендації вони надають, або зовсім не працюють корректно.

Тому більшість додатків використовують декілька алгоритмів рекомендацій які працюють паралельно, а також додаткові спеціальні

рекомендації, які ґрунтуються на рекомендаціях від модераторів додатку, деякої інформації про акції в закладах і тому подібних маркетингових стратегіях.

1.2 Постановка задачі сервісу доставки їжі

Постановка задачі для сервісу доставки їжі буде відбуватись на основі переваг та недоліків розглянутих сервісів, а також включатимуть стандарти створення подібних сервісів.

Головною метою є створення сервісу замовлення їжі, у вигляді мобільного додатку. Даний додаток матиме наступні функції:

- Логінг користувача.
- Можливість замовлення їжі з різних закладів.
- Відстеження статусу замовлення.
- Історія попередніх замовлень.
- Збір інформації про місце перебування за допомогою геолокації.

Окремою задачею буде реалізація грунту для подальшої підтримки, яка полягає у полегшенні модерування асортименту закладів. Цю задачу можна розбити на декілька компонентів функцій менеджменту:

- Створення інструменту для швидкого додання страви в базу даних.
- Створення інструменту для швидкого додання закладу в базу даних.
- Створення інструменту для легкого видалення страв або закладу і його страв з бази даних.

Отже відповідно до аналізу предметної області та прикладів реалізація можна представити, як буде виглядати даний сервіс. Даний сервіс буде розділений на 3 сегменти, клієнтський додаток для замовлення, додаток для менеджменту та управління та база даних. Клієнтський додаток буде містити систему авторизації за допомогою пошти або номеру телефону, і буде отримувати користувацькі дані активного користувача і дані про заклади та їх асортимент. Клієнт обравши цікаві йому товари виконує замовлення, ці дані передаються на сервер, після чого назначається кур’єр на доставку. Коли кур’єр

прибуває до користувача він зв’язується через повідомлення додатку або особисто.

Сервер даного сервісу буде розділений на 2 бази даних, перша з даними користувача, а друга з даними закладів. Додаток менеджменту буде містити інструменти для управління базами даних, та інструментами для перегляду внутрішніх повідомлень від користувачів.

1.3 Висновок до розділу 1

У даному розділі було розглянуто предметну область замовлення їжі та відомі реалізації сервісів доставки їжі та їх переваги і недоліки. Було розглянуто типи сервісів, а також виділено їх функції і особливості. Також були розглянуті алгоритми рекомендацій, які зазвичай використовуються в сервісах замовлення доставки їжі і проаналізовані їх переваги та недоліки.

Далі була наведена постановка задачі, де визначались функції сервісів замовлення доставки їжі, та представлений умовний приклад роботи такого сервісу

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕЗНОЛОГІЇ ЗАМОВЛЕННЯ ЇЖІ

2.1 Обгрунтування вибору платформи та інструментів для розробки

Сьогодні розробка мобільних додатків розділилась на дві барикади, це розробка під платформи Android та iOS. Для кожної із платформ існує свої інструменти для розробки. Для iOS це мова програмування Swift та середовище розробки XCode [6].

Xcode — інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання «універсальних» прикладних програм для Mac OS X. Дане середовище з’явилося у квітні 2017 року та на сьогодні являється найпопулярнішим інструментом розробки додатків для iOS оскільки дає змогу розробляти додатки для всієї лінійки продукції Apple [7].

Swift — багатопарадигмова компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду. Swift була представлена на конференції розробників WWDC 2014. Компілятор Swift побудований з використанням технологій вільного проекту LLVM. Swift успадковує найкращі елементи мов C і Objective-C, тому синтаксис звичний для знайомих з ними розробників, але водночас відрізняється використанням засобів автоматичного розподілу пам'яті і контролю переповнення змінних і масивів, що значно збільшує надійність і безпеку коду. При цьому Swift-програми компілюються у машинний код, що дозволяє забезпечити високу швидкодію. За заявою Apple, код Swift виконується в 1.3 рази швидше коду на Objective-C. Замість збирача сміття Objective-C в Swift використовуються засоби підрахунку посилань на об'єкти, а також надані у LLVM оптимізації, такі як автовекторизація. Мова також пропонує низку сучасних методів програмування, таких як замикання, узагальнене

програмування, лямбда-вирази, кортежі і словникові типи, швидкі операції над колекціями, елементи функційного програмування. Основним застосуванням Swift є розробка користувацьких застосунків для macOS, iOS, tvOS, watchOS з використанням набору інструментів Cocoa і Cocoa Touch. При цьому Swift надає об'єктну модель, сумісну з Objective-C. Сирцевий код мовою Swift може змішуватися з кодом на C і Objective-C в одному проекті. [7]

Розглянемо розробку додатків для платформи Android. Розробка мобільних додатків під цю платформу відбувається на мові Java та Kotlin середовищем для розробки являється Android Studio, також існує можливість розробки в IDE Eclipse Intelij IDE з використанням відповідних плагінів, але ці методі не є доцільними оскільки Android Studio має більший обсяг інструментів для розробки додатків під цю платформу та була створена саме для цього. Розглянемо їх.

Android Studio прийшло на зміну плагіну ADT для платформи Eclipse. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0.

Крім можливостей, присутніх в IntelliJ IDEA, в Android Studio реалізовано кілька додаткових функцій, таких як нова уніфікована підсистема складання, тестування і розгортання застосунків, заснована на складальному інструментарії Gradle і підтримуюча використання засобів безперервної інтеграції.

Бінарні складання підготовлені для Linux (для тестування використаний Ubuntu), macOS і Windows. Середовище надає засоби для розробки застосунків не тільки для смартфонів і планшетів, але і для носимих пристроїв на базі Wear OS, телевізорів (Android TV), окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для застосунків, спочатку розроблених з використанням Eclipse і ADT Plugin, підготовлений інструмент для автоматичного імпорту існуючого проекту в Android Studio [9].

Також середовище має інструменти для прискорення розробки такі як: колекція типових елементів інтерфейсу та редактор для їхнього компонування, який надає змогу переглянути, як буде виглядати інтерфейс для різних версій Android і для різних розмірів екрану. Також присутній майстер створення власних елементів оформлення інтерфейсу що підтримує використання шаблонів. У середовище вбудовані функції завантаження типових прикладів коду з GitHub.

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

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За

необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Стандартні бібліотеки Java забезпечують загальний спосіб доступу до таких платформозалежних особливостей, як обробка графіки, багатопотоковість та роботу з мережами. У деяких версіях задля збільшення продуктивності JVM байт-код можна компілювати у машинний код до або під час виконання програми.

Чудова мова програмування для розробки провірена часом, використовується в немалій кількості проектів.

Kotlin (Ко́тлін) — статично типізована мова програмування, що працює поверх JVM і розробляється компанією JetBrains. Також компілюється в JavaScript. Мову названо на честь острова Котлін у Фінській затоці, на якому розміщена частина Кронштадту [8].

Автори ставили перед собою ціль створити лаконічнішу та типо-безпечнішу мову, ніж Java, і простішу, ніж Scala. Наслідками спрощення, порівняно з Scala стали також швидша компіляція та краща підтримка IDE [9].

Мова розробляється з 2010 року, публічно представлена в липні 2011. Сирцевий код було відкрито в лютому 2012. Синтаксис мови використовує елементи з JavaScript, Паскаля, TypeScript, Haxe, PL / SQL, F #, Go і Scala, C ++, Java, C #, Rust і D.

Отже розглянувши платформи та інструменти для розробки додатків під них було прийнято рішення вибору платформи Android та мову програмування Java оскільки Android-пристрої є більш популярними на території України, по оцінкам інформаційних порталів приблизно 75% мобільних пристроїв є пристроями на базі Android. Також велику роль грає те що для розробки інших частин проекту буде теж використана мова програмування Java.

2.2 Обгрунтуванн вибору інструментів для розробки серверу та додатку керування базою даних

Для розробки додатку керування базою даних та серверу було обрано мову програмування Java та середовище програмування Intelij IDE, вибір обумовлений полегшенням розробки.

2.2.1 Вибір інструментів для розробки серверу та бази даних

Розробка бази даних відбуватиметься в середовищі MySQL, оскільки дане середовище є досить простим та зрозумілим, а також має всі інструменти для роботи та створення SQL-баз даних.

MySQL — вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. Нижче на рисунку 2.1 зображено логотип середовища MySQL [16].

Рисунок 2.1 – Логотип середовища MySQL

Також при розробці серверу, який буде підключений до бази даних буде використана технологія JDBC.

Java DataBase Connectivity (англ. Java DataBase Connectivity — з'єднання з базами даних на Java), скорочено JDBC) — прикладний програмний інтерфейс Java, який визначає методи, з допомогою яких програмне забезпечення на Java здійснює доступ до бази даних. JDBC — це платформо-незалежний промисловий стандарт взаємодії Java-застосунків з різноманітними СУБД, реалізований у вигляді пакета java.sql, що входить до складу Java SE. В основі JDBC лежить концепція так званих драйверів, що дозволяють отримувати з'єднання з базою даних по спеціально описаному URL. Драйвери можуть завантажуватись динамічно (під час роботи програми). Завантажившись, драйвер сам реєструє себе й викликається автоматично, коли програма вимагає URL, що містить протокол, за який драйвер «відповідає» [13].

JDBC API містить два основні типи інтерфейсів: перший — для розробників застосунків і другий (нижчого рівня) — для розробників драйверів. З'єднання з базою даних описується класом, що реалізує інтерфейс java.sql.Connection. Маючи з'єднання з базою даних, можна створювати об'єкти типу Statement, використовувані для здійснення запитів до бази даних на мові SQL. Існують такі види типів Statement, що відрізняються своїм призначенням:

– java.sql.Statement — Statement загального призначення;

– java.sql.PreparedStatement — Statement, що служить для здійснення запитів, котрі містять підставні параметри (позначаються символом '?' у тілі запиту);

– java.sql.CallableStatement — Statement, призначений для виклику збережених процедур.

Клас java.sql.ResultSet дозволяє легко обробляти результати запитів.

Для реалізації вводу-виводу(клієнт-сервер з'єднання) буде використана технологія NIO(Non-blocking I/O або java New I/O) .

2.2.2 Вибір інструментів для розробки програми управління

Для розробки програми управління неодмінно потрібно використання візуального інтерфейсу. Для реалізації графічного інтерфейсу в додатках розроблених на мові Java найчастіше використовують дві технології, це Java Swing та JavaFX

Swing Java - це інструментарій графічного інтерфейсу користувача (GUI), що включає компоненти GUI. Swing надає багатий вибір віджетів та пакетів для створення вишуканих компонентів GUI для Java-додатків [13].

Swing є частиною Java Foundation Classes (JFC), який є API для програмування GUI на Java, що забезпечує графічний інтерфейс користувача. Бібліотека Java Swing побудована поверх Java Abstract Widget Toolkit (AWT), старішого, залежить від платформи набору інструментів GUI. Ви можете використовувати прості компоненти програмування графічного інтерфейсу Java, такі як кнопки, текстові поля тощо з бібліотеки, і вам не доведеться створювати компоненти з нуля.

Це чудовий спосіб щоб познайомитись з графічним інтерфейсом java та поглибити свої знання, однак має деяку складність з налаштуванням інтерфейсу оскільки розташування кожного елемент у вікні прописується окремо, тому розглянемо більш простіше рішення JavaFX.

JavaFX — платформа та набір інструментів для створення насичених інтернет-застосунків з можливістю підвантаження медіа та змісту. Вперше продемонстровано Sun Microsystems на Міжнародній конференції Java-розробників JavaOne у травні 2007. JavaFX містить у собі набір утиліт, за допомогою яких веброзробники та дизайнери можуть швидко створювати та надавати розвинуті інтернет-застосунки для десктопів, мобільних пристроїв, телебачення та інших платформ. JavaFX складається з JavaFX Script і JavaFX Mobile. Починаючи з випуску JavaFX 2.0 забезпечено можливість створення JavaFX-застосунків, написаних цілком мовою Java. Для розробки застосунків доступний багатий графічний і мультимедійний API, що спрощує створення

візуальних програм. Даний інструмент дозволяє розробляти візуальний інтерфейс в XML файлах, які керуються відповідними класами-контролерами.

2.3 Розробка алгоритмів роботи системи

Розпочнемо з розробки алгоритму роботи серверу, оскільки в подальшій розробці всі компоненти будуть зв’язані саме через нього і їх роботоспроможність буде перевірятись саме завдяки роботі з сервером.

Сервер повинен виконувати керуючу функцію та контроль доступу до бази даних і в загальному до інформації. Тож було розроблено дві схеми алгоритмів, які описують загальну роботу та обробку типових запитів, а також алгоритм контролю доступу до даних.

Сервер буде весь робочий час знаходитись в режимі очікування запиту від клієнту, без різниці це клієнт мобільного додатку для замовлення чи менеджер, який управляє базою даних. При отриманні запиту сервер перевіряє чи авторизований клієнт, тобто перевіряє надісланий цей запит з довірених додатків чи з посторонніх, якщо виявляється що запит з недовірених додатків у відповідь відправляються помилка, а також відбувається запис у журнал.

Якщо запит з довіреного пристрою відбувається обробка типу запиту, якщо запит на аутентифікацію то відбувається процес авторизації користувача у систему, якщо запит на отримання інформації з бази даних, то відбувається передача відповідної інформації, якщо запит на відключення відбувається відключення клієнту. Також після обробки будь-якого із запитів відбувається запис події в журнал, повернення до обробки запитів

На рисунку 2.2 зображено блок-схема алгоритму обробки запитів сервером.

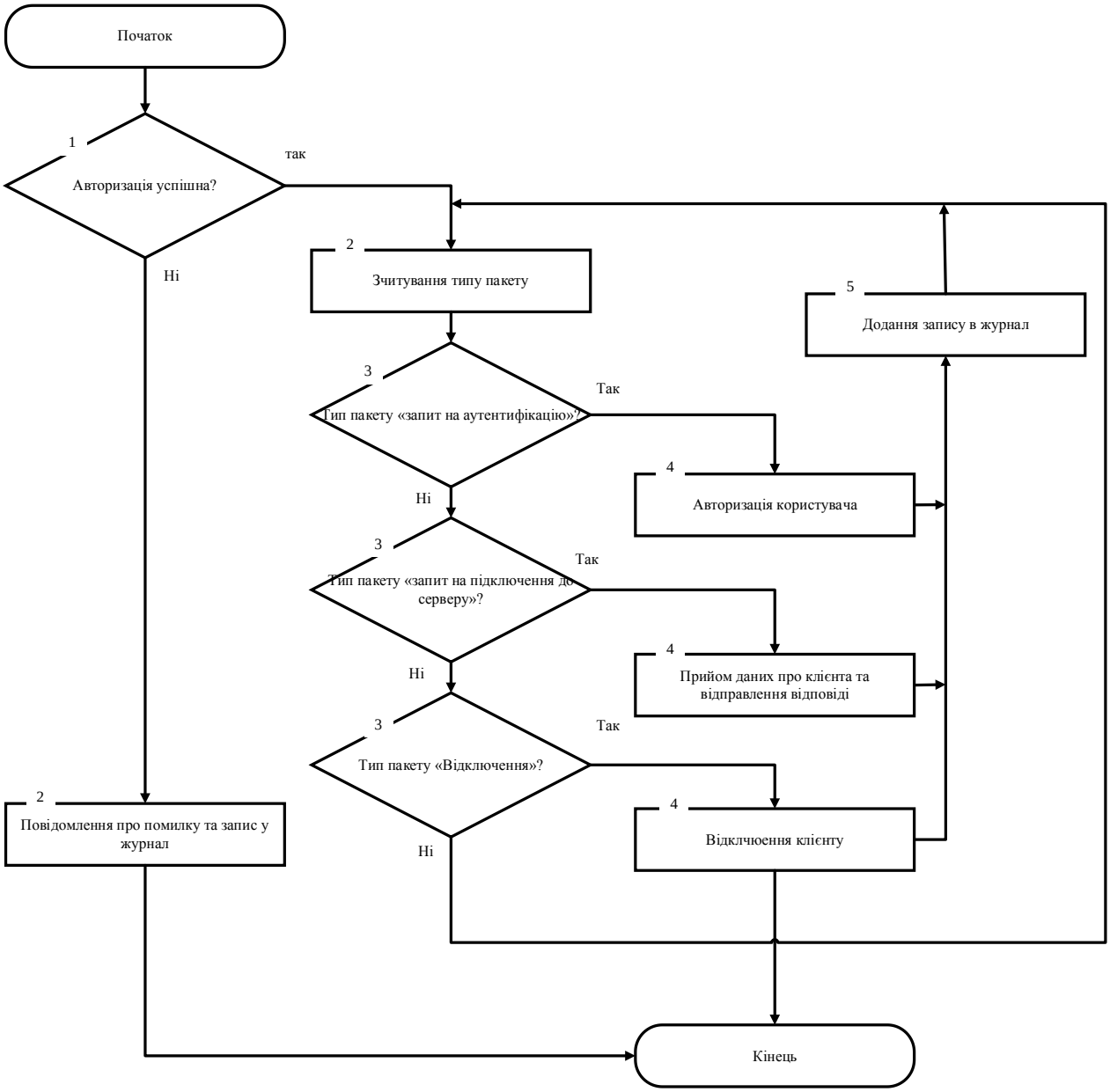


Рисунок 2.2 – Блок-схема алгоритму обробки запитів сервером

Алгоритм контролю доступу до даних буде виконувати авторизацію користувачів та контроль доступу до даних. При отриманні запиту сервер перевірить чи авторизований користувач, якщо користувач авторизований і відправив запит на доступ до даних, відбувається перевірка типу користувача, «клієнт» чи «менеджер» якщо користувач «клієнт» то відбувається передача особистої інформації про клієнта та інформації про заклади, якщо тип користувача «менеджер» то відбувається передача інформація про базу користувачів та закладів, якщо тип не співпадає з вище вказаними то відправляється відповідь «помилка» та виконується відповідний запис. Якщо

користувач не авторизований то відбувається перевірка на тип запиту авторизації, відповідно «клієнт» чи «менеджер» і відповідно до типу запиту відбувається відповідна авторизація користувача, і очікування запиту на доступ до інформації або інших запитів. Нижче на рисунку 2.3 зображено алгоритм авторизації та контролю доступу до даних.

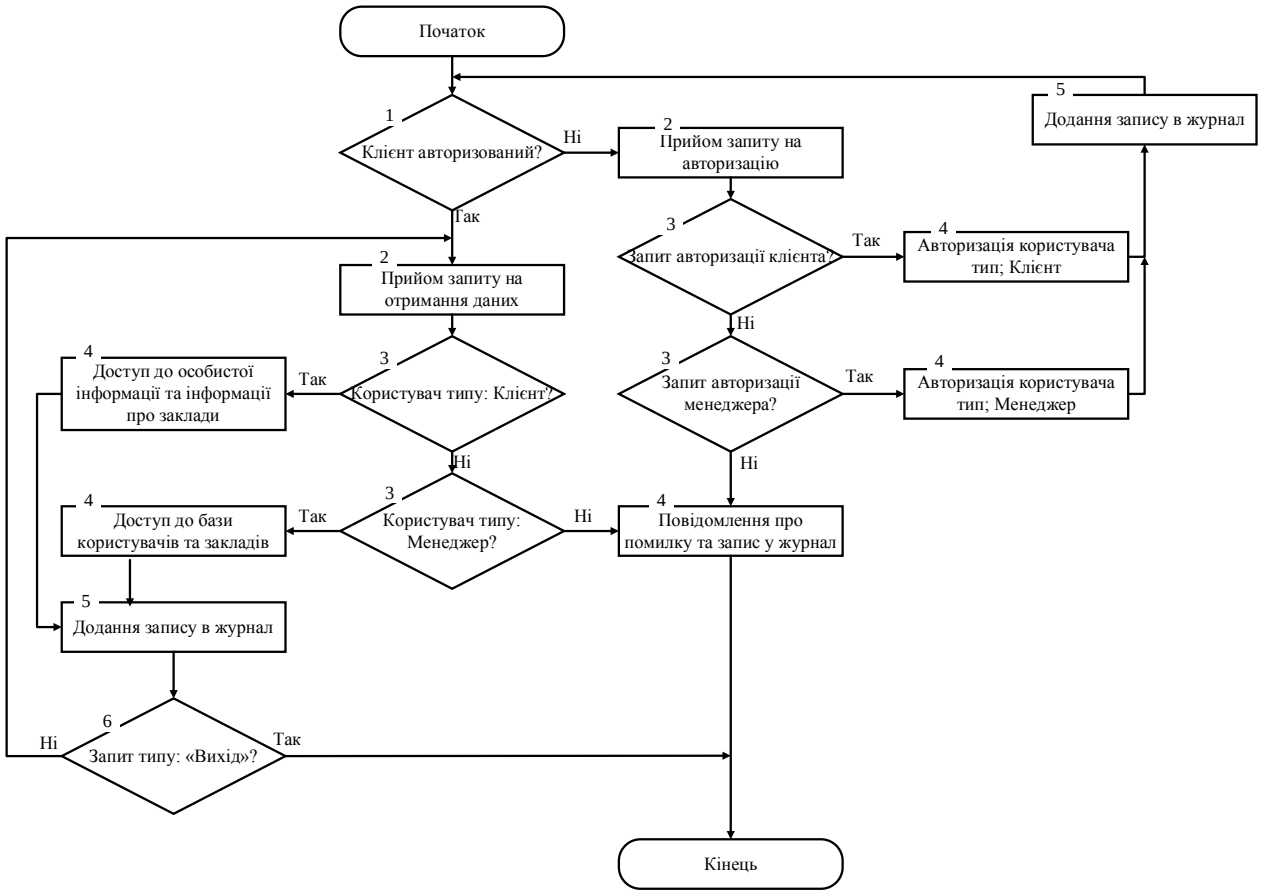


Рисунок 2.3 – Алгоритм авторизації та контролю доступу до даних

2.4 Удосконалення алгоритму рекомендацій закладів

Розглянемо алгоритми про які йшла мова в першому розділі. Перший алгоритм рекомендацій, це рекомендації за оцінками, популярністю та місцем розташування відносно користувача. Його основною проблемою є рекомендації на основі загальних оцінок, що корисно коли користувач замовляє вперше, а наступні рази, коли користувач захоче спробувати щось інше йому потрібно буде витрачати більше часу для пошуку цікавого йому, і цей час не буде

зменшуватись скільки би користувач унікальних замовлень не зробив, що є не відповідністю до поняття рекомендацій.

Проблеми такого типу зазвичай вирішуються розширенням збору інформації та створенню різного типу рекомендацій, наприклад маючи інформацію про замовлення великої кількості людей їх можна типізувати, для цього існують такі інструменти, як кластеризація яку виконує штучний інтелект, він розділить всіх користувачів на певні групи відповідно до їх даних про замовлення і на основі цих груп можна надавати рекомендації користувачам по мірі їх взаємодії із системою. Це вже є другий тип алгоритму, який створює типізовані рекомендації і з кожним унікальним замовленням його рекомендації будуть більш точні, а необхідний користувачу час на пошук цікавинок буде зменшуватись.

Розглянемо такий алгоритм, першим кроком є збір інформації про користувачів та їх замовлення, ця інформація включає в себе стать, пункти призначення замовлень(багато різних, чи декілька однакових), час доби в який відбувається замовлення(ранок, обід, на протязі всього дня чи ввечері), далі вже більш проста інформація, це заклади, страви, їх групи, кількість порцій і частота замовлень.

Ці дані опрацьовуються штучним інтелектом який формує «кластери», це групи користувачів поведінка, яких схожа між собою. Тобто формуються групи за схожими ознаками, де для кожної групи будуть відповідні їй рекомендації, які формуватимуться по відгуках цієї групи що до рекомендацій.

Наступний крок це формування особистих рекомендацій відповідно до місця знаходження користувача і вивід цих рекомендацій на екран користувачу.

Для стимулювання користувача на додаткові замовлення удосконалимо даний алгоритм. Оскільки коли користувач вибирає заклад в якому бажає замовити і пункт призначення доставки, ми можемо розрахувати маршрут чи декілька маршрутів та запропонувати користувачу, замовити додаткову позицію за невелику доплату, яка менше за стандартну ціну доставки. Для цього після узгодження першого замовлення створимо запит на додаткове замовлення при

позитивній відповіді, передаємо інформації про маршрути в алгоритм рекомендацій який буде шукати заклади з відхиленням від маршруту 300-500 метрів, і сформує додатковий список рекомендацій який базується вже не тільки на місці розташування користувача, а й на маршруті замовлення. Зобразимо наш алгоритм. Удосконалений алгоритм рекомендацій зображений на рисунку 2.4.

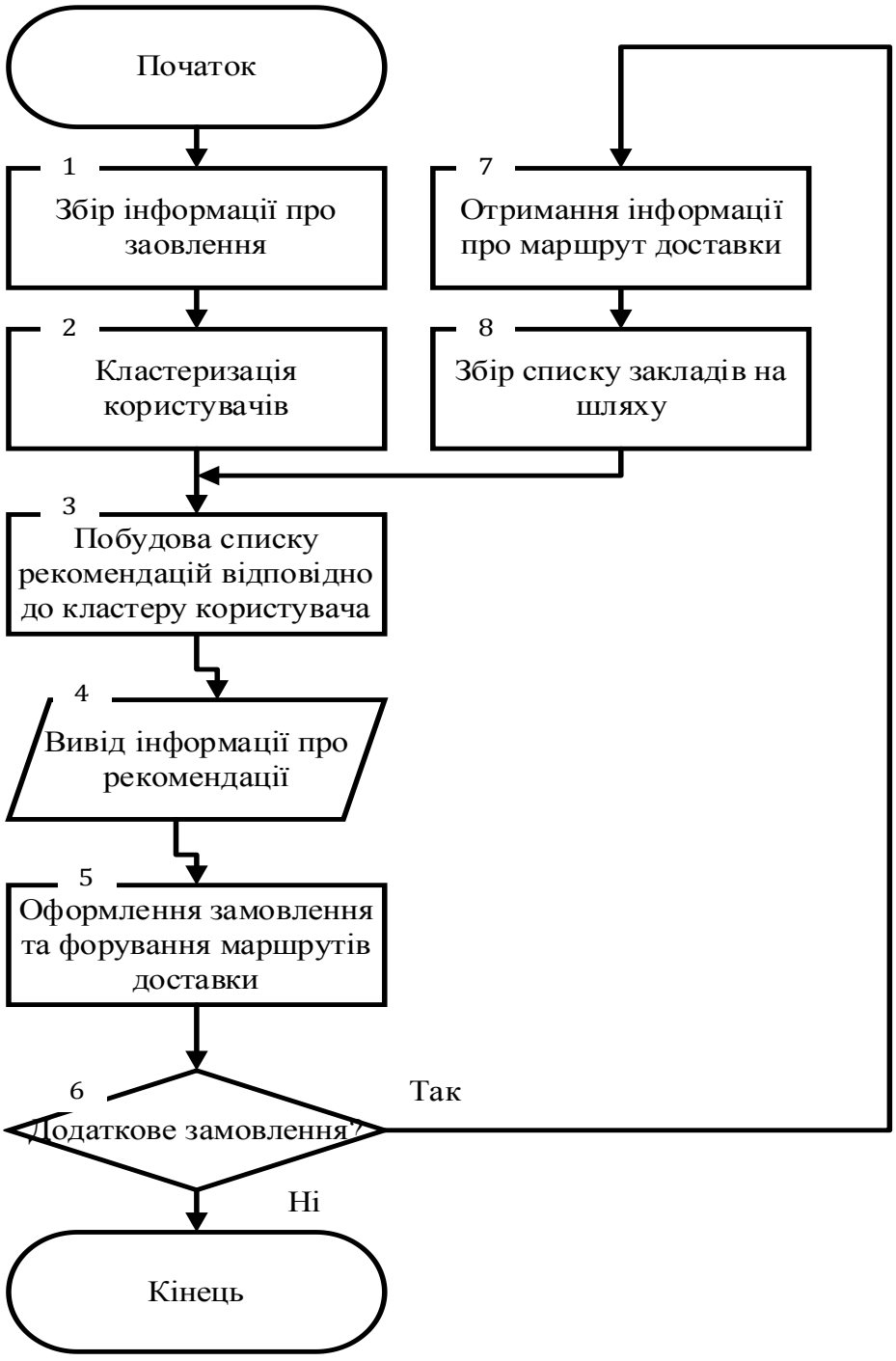


Рисунок 2.4. – Блок-схема удосконаленого алгоритму рекомендацій

2.4 Висновок до розділу 2

В даному розділі були проаналізовані засоби розробки кожного компоненту системи. Були розглянуті дві платформи для розробки мобільного додатку Android і Apple також були розглянуті середовища розробки та мови програмування до кожної з платформ, були обрані та обгрунтовані найбільш підходящі інструменти для розробки.

Розглянуті та обгрунтовані інструменти розробки інших компонентів системи. Розроблені алгоритми загальної роботи серверу та алгоритму авторизації, контролю доступу до даних. Також були розглянуті способи вирішення проблем типових алгоритмів рекомендацій закладів та закладів у них і розроблено удосконалений алгоритм рекомендацій.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕЇНОЛОГІЇ ЗАМОВЛЕННЯ ЇЖІ

3.1 Опис використаних шаблонів та моделей

Після аналізу шаблонів розробки для створення мобільного додатку було обрано модель MVVM(Model View ViewModel) з використання м структурного шаблону Adapter.

Модель MVVM це архітектурний шаблон суть якого полягає в створенні взаємодії між компонентами Model(клас) і View(графічний інтерфейс наприклад view layer) за допомогою об’єкту ViewModel. Об’єкт ViewModel надаэ команди для вью і та зв’язує їх з моделю, коли модель обновляється відповідно обновляється вью через зв’язані данні. Точно так же коли користувач взаємодіє з вью зв’язування працює в протилежному напрямі автоматично обновляючи модель [15].

Структурний шаблон Adapter один з популярніших шаблонів при розробці android додатків, данний структурний шаблон є частиною моделі MVVM, тобто саме за цим шаблоном відбувається зкріплення класів таких як User та Product з графічною моделлю.Діаграма реалізації архітектури MVVM зображена на рисунку 3.1.

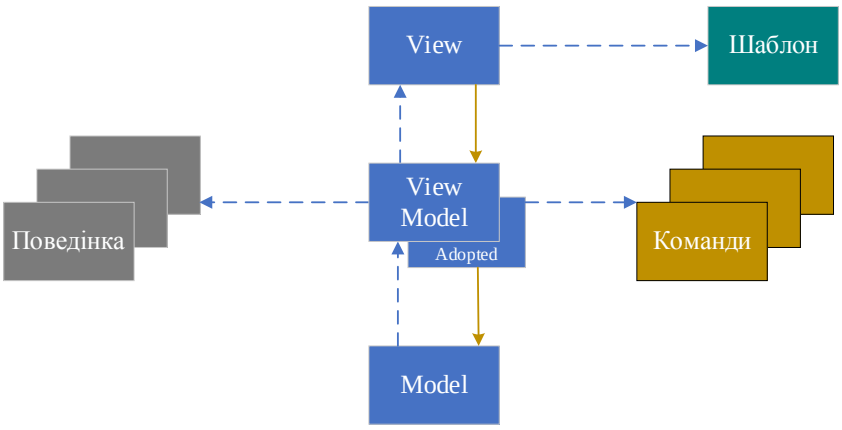


Рисунок 3.1 — Діаграма загальної реалізації архітектури MVVM

3.2 Програмна реалізація сервер-додатку

Реалізація серверу буде відбуватись в декілька кроків, по-перше це написання консольного додатку спроможного приймати веб повідомлення та обробляти їх, наступним кроком є підключення бази даних, а також написання методів для управління даними, і передачею їх додаткам-клієнтам. Останнім шагом реалізації серверу буде розробка методів отримання та передача інформації про заклади.

Розпочнемо розробку консольного сервер-додатку. Як уже було сказано раніше, розробка буде відбуватись в середовищі Intellij IDE на мові програмування java. Встановити дане середовище можна з офіційного сайту. При запуску його ми побачимо його стартове меню, яке зображене на рисунку 3.2 нижче.

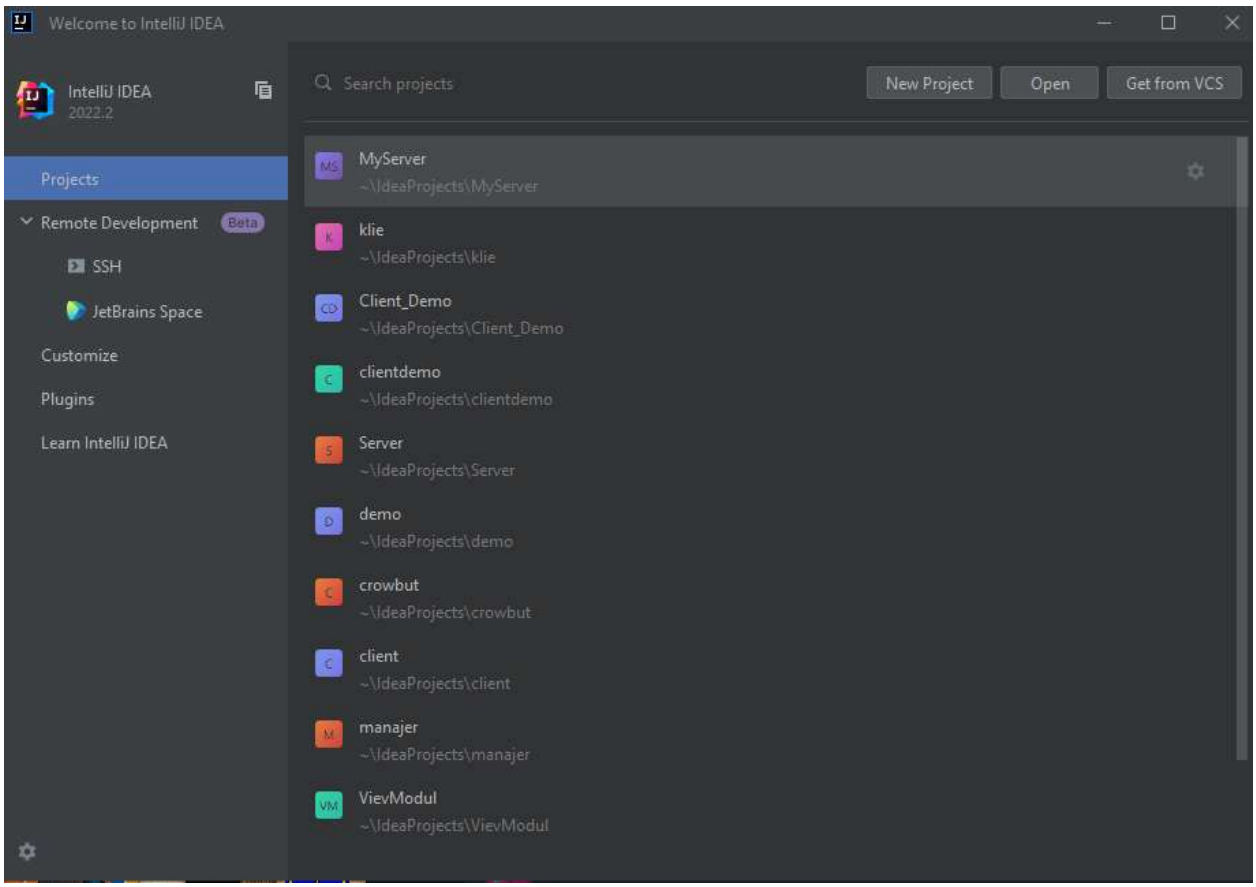


Рисунок 3.2 – Скріншот стартого вікнаа середовища Intellij IDE

Створюємо новий проект Java та вибираємо збірну систему Maven. Після чого відкриється основне вікно програми де відбувається розробка. В першу чергу відкривається файл pom.xml в якому оголошені налаштування системи Maven для даного проекту, також сюди додаються деякі залежності до бібліотек перед використанням їх методів, наприклад методи роботи з Json-файлами такі як «jackson-databind» та «GJosn» [13]. Нижче на рисунку 3.3 зображено вікно Intellij з підключеними необхідними залежностями.

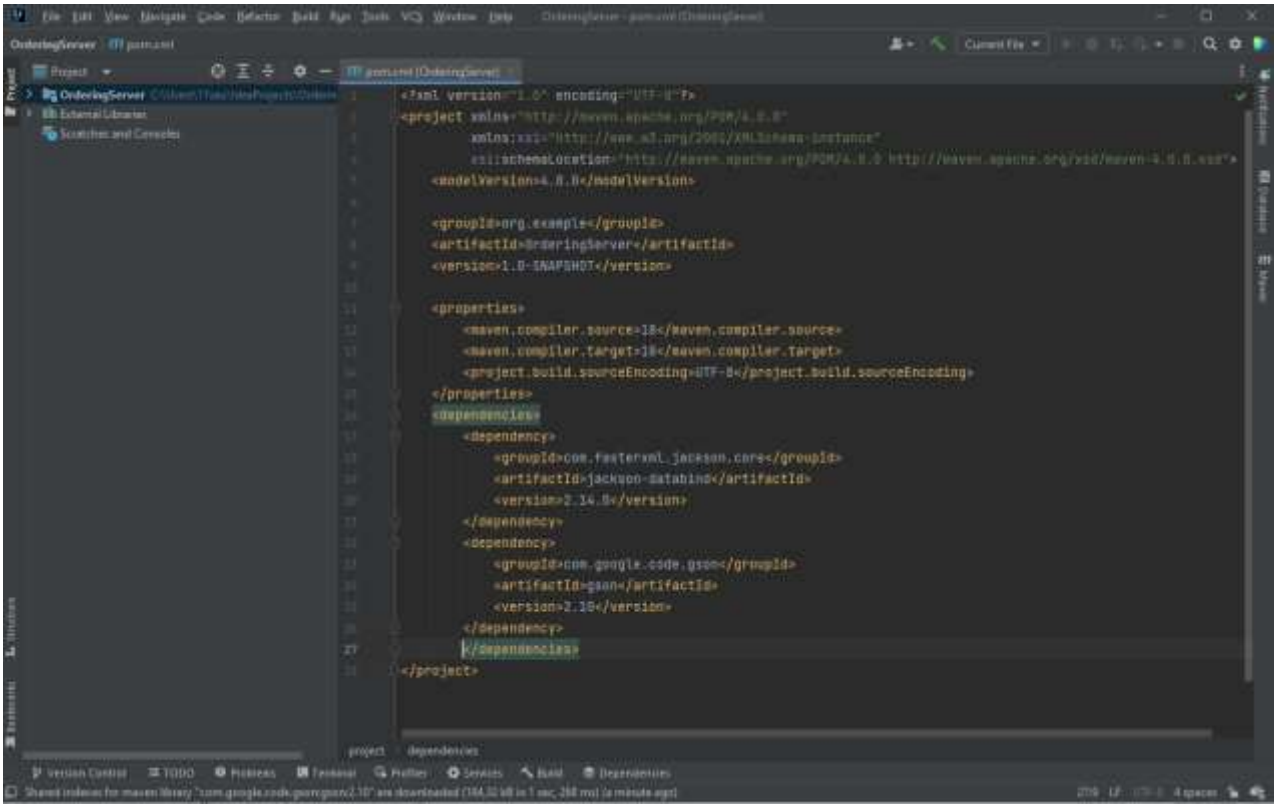


Рисунок 3.3 – Вікно Intellij з підключеними залежностями

Наступним кроком є створення класів та методів для запуску серверу. Створюємо клас SocketServer а також його поля, селектор, список dataMapper типу list в якому будуть зберігатись об’єкти SocketChanel, а також змінну listenAddress в якій буде зберігатись адрес серверу. Розробляємо конструктор для швидкого запуску полями якого будуть адрес та порт які передаватимуться в змінну listenAddress і створюємо список dataMapper.

Реалізуємо метод startServer створюючи об’єкт serverChannel типу ServerSocketChannel, відклчаємо його блокування, це потрібно для роботи селектора. Та запускаємо сервер на вказаному в listenAddress адресі. Виводимо в консоль повідомлення що сервер запущений та запускаємо цикл while(true) в якому очікуємо підключення, а також створюємо конструкцію, яка оброблює події та запускає відповідні методи. Нижче на рисунку 3.4 приведено фото реалізованого коду методу startServer.

```
1 usage
private void startServer() throws IOException {
    this.selector = Selector.open();
    ServerSocketChannel serverChannel = ServerSocketChannel.open();
    serverChannel.configureBlocking(false);
    serverChannel.socket().bind(listenAddress);
    serverChannel.register(this.selector, SelectionKey.OP_ACCEPT);
    System.out.println("Server started...");

    while (true) {

        this.selector.select();

        Iterator keys = this.selector.selectedKeys().iterator();
        while (keys.hasNext()) {
            SelectionKey key = (SelectionKey) keys.next();

            keys.remove();

            if (!key.isValid()) {continue;}

            if (key.isAcceptable()) {
                this.accept(key);
            }
            else if (key.isReadable()) {
                this.read(key);
            }
            else if (key.isWritable()) {
                this.write(key);
            }
        }
    }
}
```

Рисунок 3.4 – Скріншот лістингу коду методу startServer

Наступним кроком реалізуємо методи accept, read і write, метод accept повинен підтвержувати вхідне підключеня та з’єднувати сокети, а також переводить канал в положення читання. Метод read виконує зчитування з каналу, а метод write запис в канал.

Реалізувавши дані методи провіримо роботу серверу написавши простий клієнт додаток який відправить декілька повідомлень на отримає відповідь від серверу нижче на рисунку 3.5 зображено скріншот результату роботи тестового клієнту .

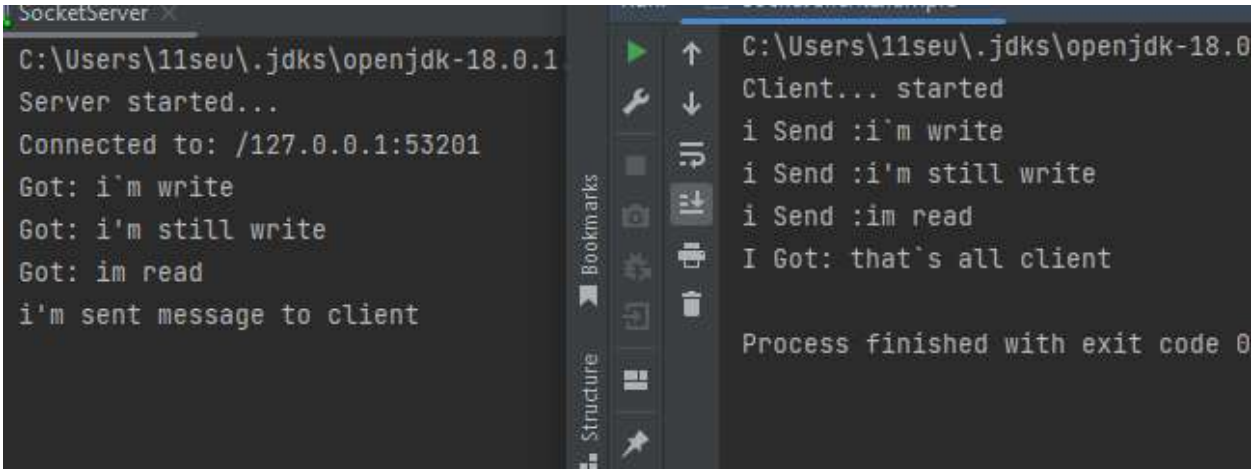


Рисунок 3.5 – Скріншот результату роботи тестового клієнту

Наступним кроком розробки серверу є підключення його до бази даних. Цей виконується за допомогою спеціального драйверу JDBC. Скачаємо бібліотеку та добавимо у наш репозиторій. База даних буде використовуватись для збереження інформації про користувачів та їх замовлення, буде мати 2 таблиці: «Користувачі» та «Історія замовлень». Таблиця користувачі має 8 полів:

- id
- Прізвище
- Ім’я
- Логін
- Пароль
- Номер телефону

−Електрона пошта

−Стать

Таблиці зв’язані між собою полем логін, тип зв’язку один до багатьох, тобто один логін до багатьох замовлень. Таблиця «Історія замовлень» має наступні поля:

−id

−Логін

−Замовлення

−Ціна за доставку

−Ціна замовлення

−Загальна сума

Створимо нашу базу даних у додатку MySQL та додамо до неї нашу табличку користувачів, нижче на рисунку 3.6 скріншот налаштувань таблиці.

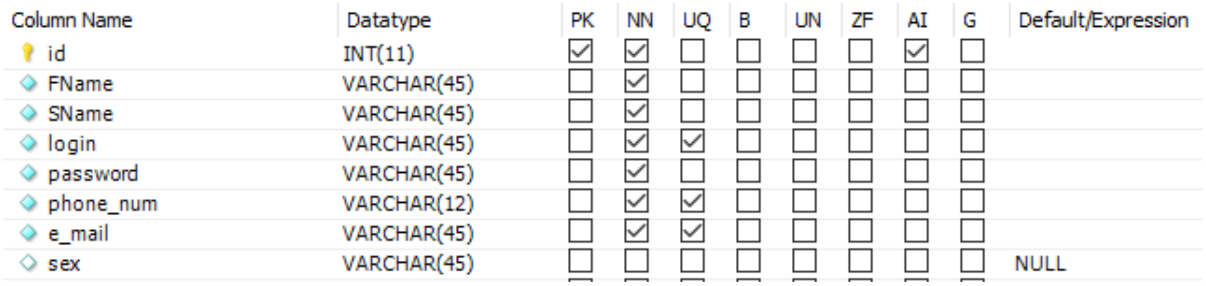


Рисунок 3.6 – Скріншот налаштувань таблиці в додатку MySQL

Наступним кроком підключимо нашу базу даних до сервер-додатку. Для цього використаємо драйвер JDBC, а також вказуємо в білдері що наш додаток працює саме з MySQL. Підключення бази даних буде відбуватись через клас DatabaseHandler, який наслідується від нашого класу Config в якому ми описали адрес підключення та дані для підключення до бази даних. Підключення до бази даних реалізовано в методі Connection, який повертає об’єкт dbConection. Також було реалізовано метод signUpUser для тестування підключення, скріншот коду реалізації методу зображена на рисунку 3.7.

```
public void signUpUser(String firstName, String secondName, String login,
                       String password,String phone, String sex, String email){
    String insert = "INSERT INTO " + DB_Const.USER_TABLE +
            " ("+DB_Const.USER_FNAME+","+DB_Const.USER_SNAME+","+DB_Const.USER_LOGIN
            +","+DB_Const.USER_PASS+","+DB_Const.USER_PHONE+","+DB_Const.USER_EMAIL+" " +
            ","+DB_Const.USER_SEX+") " +"VALUES(?,?,?,?,?,?,?)";
    try {
    PreparedStatement prSt = getDbConection().prepareStatement(insert);
    prSt.setString( parameterIndex: 1, firstName);
    prSt.setString( parameterIndex: 2, secondName);
    prSt.setString( parameterIndex: 3, login);
    prSt.setString( parameterIndex: 4, password);
    prSt.setString( parameterIndex: 5, phone);
    prSt.setString( parameterIndex: 6, email);
    prSt.setString( parameterIndex: 7, sex);
        prSt.executeUpdate();
    } catch (SQLException e) {
        //throw new RuntimeException(e);
        String[] ExeptionMessage = e.getMessage().split( regex: " ");
        String mess = "what is it?";
        if(ExeptionMessage[0].equals("Duplicate")){
            switch (ExeptionMessage[5].substring(1,ExeptionMessage[5].indexOf("_UNI"))) {
                case "phone_num": mess ="that's phone"; break;
                case "login": mess = "that's login"; break;
                case "e_mail":mess = "that's e-mail"; break;
                default: break;
            }
            System.out.println(mess);
        }

    } catch (ClassNotFoundException e) {
        throw new RuntimeException(e);
    }
```

Рисунок 3.7 – скріншот коду реалізації методу signUpUser

Створимо таблицю для збереження інформації про замовлення. Останнім кроком реалізації сервер-додатку є створення інтерфейсів взаємодії. Тобто реалізувати передання даних, їх збереження та інше, це буде реалізовано за допомогою Json-файлів та відповідно бібліотек які працюють з ними, це бібліотеки Jackson та Gson. Бібліотека Jackson допомагає перетворювати об’єкти та їх поля у строку форматовану під Json-файл, який вже можна передавати, а бібліотека Gson відповідно виконує зворотну функцію, перетворюючи Json-файл

у відповідний об'єкт, або масив об'єктів та заповнює поля відповідними значеннями.

Для початку потрібно створити класси шаблон які будуть приймати відповідні поля, вони будуть називатись відповідно до файлів та мати приставку Root, наприклад UserRoot, Root, OrderRoot. Відповідно ці класи будуть формуватись перед відправкою, або отриманням для роботи з ними. Нижче на рисунку 3.8 зображено клас EstablishmentRoot та його вкладені класи, а також скріншот коду класу Menu

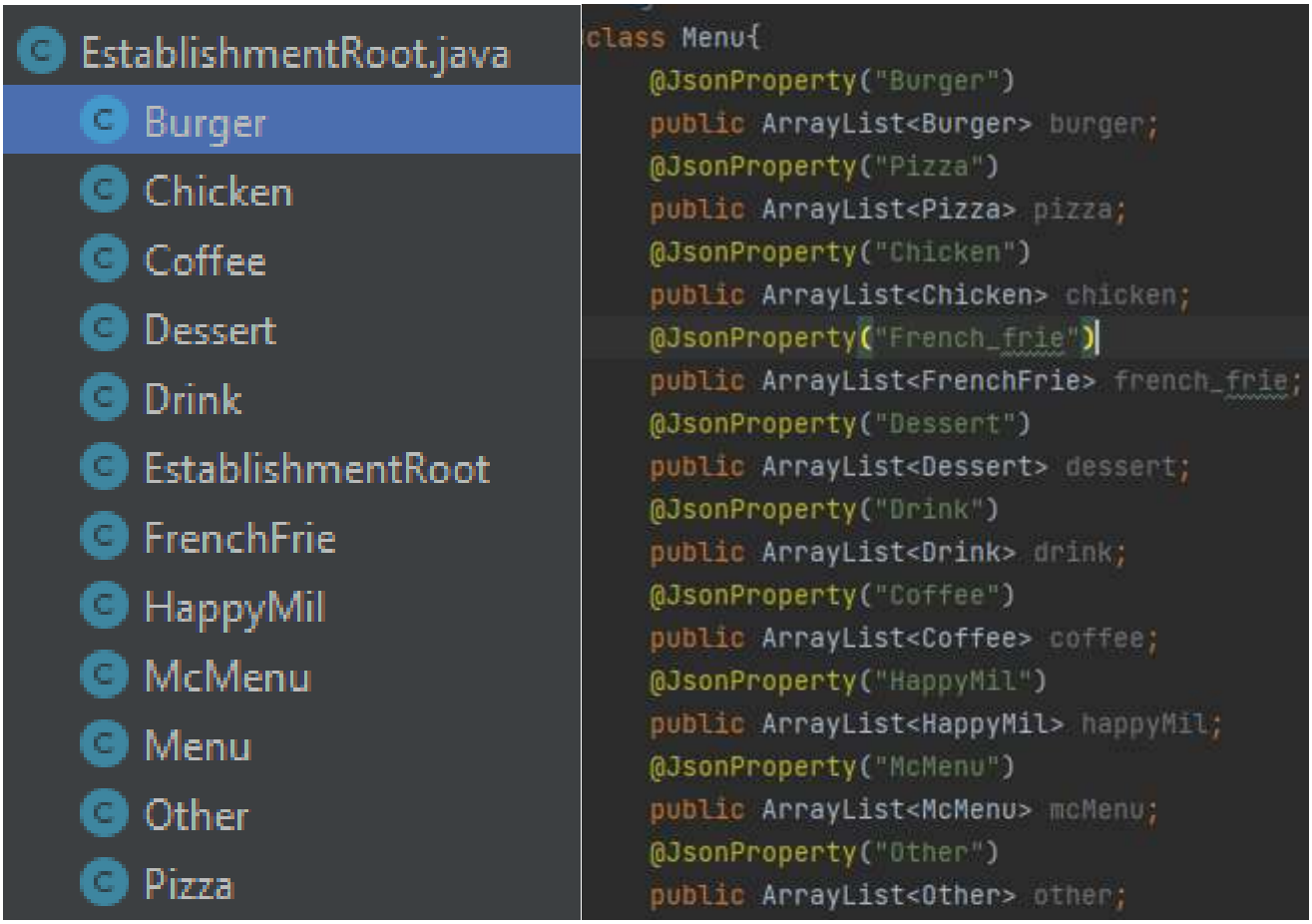


Рисунок 3.8 – Скріншоти класу EstablishmentRoot, його вкладених класів та коду класу Menu

Реалізувавши ці класи, їх методи, та парсер, розробка, можна перейти до наступного пункту, розробки.

3.3 Програмна реалізація десктоп-додатку для менеджменту

Додаток для менеджменту буде реалізован з візуальним інтерфейсом використовуючи JavaFX. Для початку розробимо інтерфейс додатку, а слідом перейдемо до реалізації, для більш простої розробки інтерфейсу використовується SceneBuilder.

Розробимо шаблон вікна для роботи з папками та файлами, в ньому ми будимо зчитувати файли та папки в заданій дерикторії, а також відкривати чи переходити до інших каталогів. . Нижче на рисунку 3.9 зображено скріншот макету вікна для роботи з папками та файлами

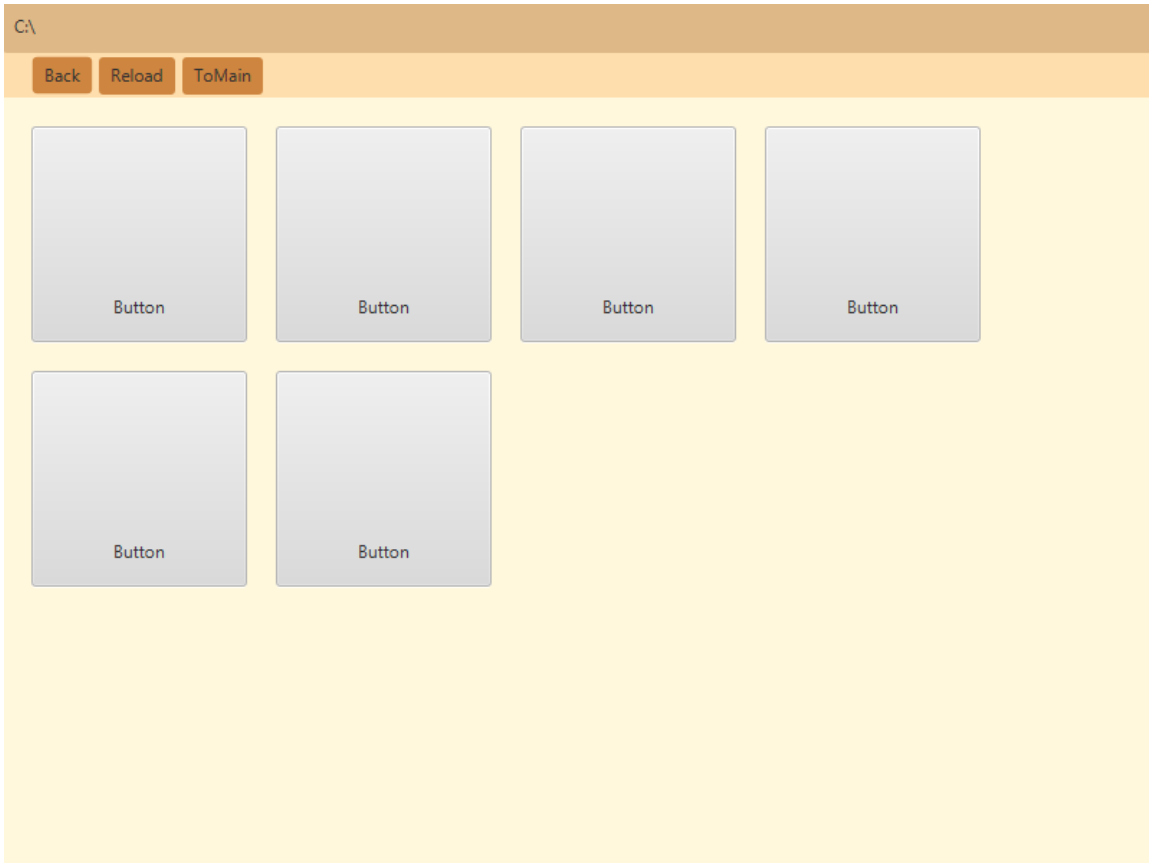


Рисунок 3.9 Скріншот макету вікна для роботи з файлами та папками

Дані кожного закладу зберігаються в json файлах, в відповідних папках, тобто для відкриття певного файлу потрібно, відкрити папку мережі закладів певного типу, вибрати заклад за адресою і в цій папці буде знаходитись шуканий

файл. При виборі файлу відбувається парсинг в клас десктоп-додатку EstablishmentRoot а також і подальше виведення інформації про заклад на екран. Нижче на рисунку 3.10 зображено макет вікно редагування товарів

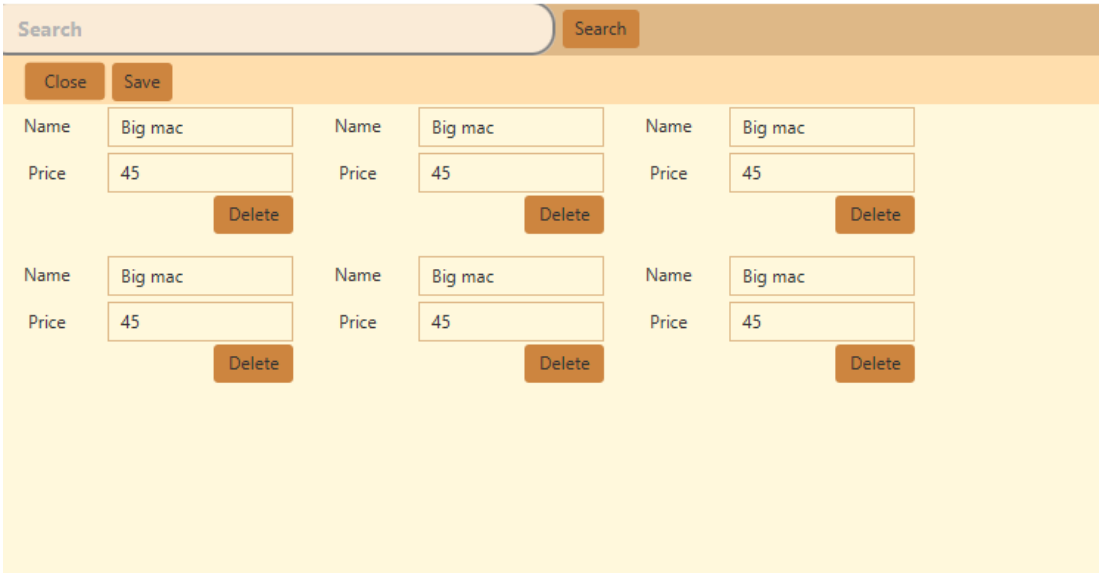


Рисунок 3.10 – Скріншот макету вікна редагування товарів

Також було створено макет вікна додання закладу, вікно управління даними користувачів та вікно авторизації.

Наступним кроком реалізуємо логіку роботи вікна роботи з файлами та каталогами. Для цього ми будемо використовувати клас типу Files та його метод walk та метод treewalk перший повертає лист файлів в директорії а другий повертає всі дерикторії в даній дерикторії. Потім за допомогою створеного методу CreateButtons() ми створюємо лист об’єктів Button з їм’ям відповідним до файлу чи дерикторії. Далі розробимо головне вікно з якого ми будемо переходити у інші необхідні нам розділи програми.

Програма матиме 3 розділи це «Користувачі», «Замволення», та «Заклади». Нижче на рисунку 3.11 зображено головне вікно програми.

Рисунок 3.11 – Головне вікно програми

3.4 Програмна реалізація мобільного додатку-клієнта

Даний додаток маэ велику кількість вікон тому для початку розробимо візуальний інтерфес з яким в подальшому будемо взаємодіяти. Почнемо з вікна activity_login, в данному вікні будуть присутні 2 кнопки Login, Register, поля для вводу електронної пошти, паролю, картинки-іконки для пошти і паролю, картинка-фон, та декілька надписів. Вони повинні бути зручно розташовані не нагромаджувати інформацією та бути інтуїтивно зрозумиліми навіть без надписів. Для розположення даних об'єктів було використано контейнери LinearLayout і RelativeLayout. Нижче на писунку 3.2 зрбражене розроблений інтерфейс вікна activity_login.

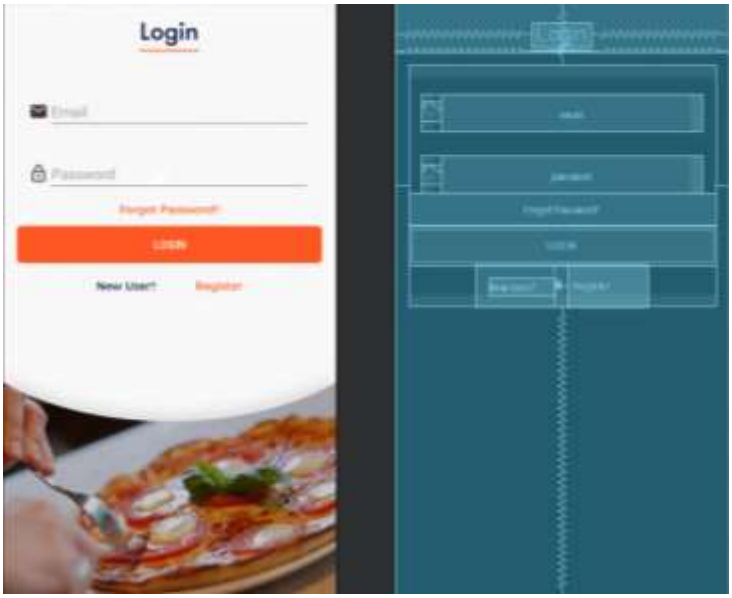


Рисунок 3.12 — Скріншот інтерфейсу вікна activity_login в конструкторі

Тепер розглянемо розглянемо реалізацю аутитифікації та деякі особливості класу LoginActivity. Спочатку оголошуються деякі змінні та створюються обїєкт модуля Auth для подальшого входу в систему, як зображено нижче

```
private Auth mAuth;
    private EditText etEmail, etPassword;
    private String email, password;
    ProgressDialog pDialog;
    private static final String KEY_EMPTY = "";
```

Напним в методі onCreate() ми зв'язуємо наші об'єкти з елементами View та під’єднуємося до серверу, як зображено нижче.

```
setContentView(R.layout.activity_login);
        getSupportActionBar().hide();
        etEmail = findViewById(R.id.email);
        etPassword = findViewById(R.id.password);
        mAuth = Auth.getInstance();
```

Вхід в систему відбувається відправкою на метод signInWithEmailAndPassword() объэкту mAuth, електронної пошти та паролю при позитивній відповіді відбувається вхід в систему, а при негативній повідомлення «Invalid email or password», нижче наведений лістинг коду методу LoginUser

```
public void LoginUser(){
        displayLoader();
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        if (task.isSuccessful()){
                            Toast.makeText(LoginActivity.this, "login
successful",
                                    Toast.LENGTH_SHORT).show();
                            pDialog.dismiss();
                           Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
                           startActivity(intent);
                           finish();
                        }else {
                            Toast.makeText(LoginActivity.this, "Invalid
email or password",
                                    Toast.LENGTH_SHORT).show();
                            pDialog.dismiss();
                        }
                    }
                });
    }
```

Також для обробки деяких виняткових ситуацій таких як наприклад пусті поля використані наступний метод

```
private boolean validateInputs() {
```

```
        if(KEY_EMPTY.equals(email)){
            etEmail.setError("Invalid input");
            etEmail.requestFocus();
            return false;
        }
        if(KEY_EMPTY.equals(password)){
            etPassword.setError("Invalid input");
            etPassword.requestFocus();
            return false;
        }
        return true;
    }
```

Наступне вікно це activity_register вікно реєстрації має більше полів для вводу тексту, а також більше обробників подій невірного заповнення полів. На рисунку 3.13 зображено вигляд activity_register в конструкторі.

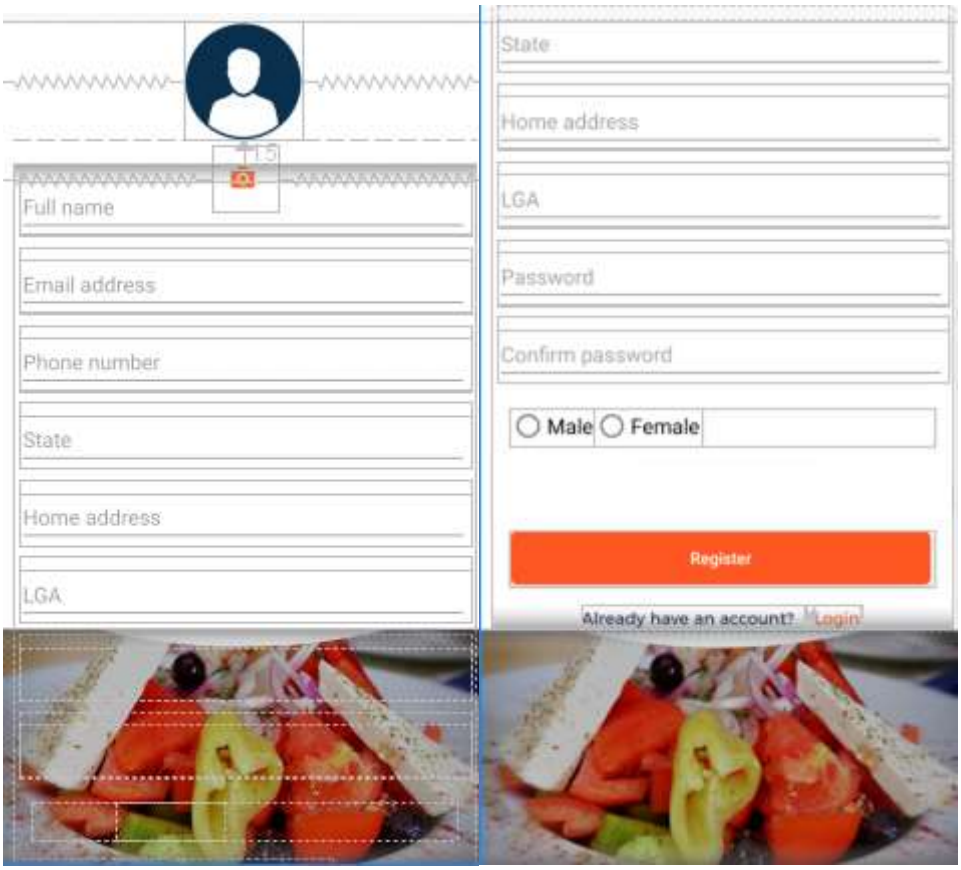


Рисунок 3.13 — Скріншоти вигляду вікна activity_register в конструкторі Android Studio

Для регістрації нового користувача використовується 8 полів вводу, та два RadioButton. Інформація з цих полів зчитується класом RegisterActivity та передається в базу данних серверу нижче наведений лістниг коду методів register та writeNewUser

```
public void register(View view) {
    name = full_Name.getText().toString();
    email = etEmail.getText().toString().toLowerCase().trim();
    phone = etPhone.getText().toString();
    home = etHome.getText().toString();
    password = etPassword.getText().toString().trim();
    Cpassword = etConfirmPassword.getText().toString().trim();
    lga = etLga.getText().toString().toLowerCase();
    state = etState.getText().toString().toLowerCase();
    if (validateInputs()) {
        upload_imageTo_storage(); }
```

Як ми бачимо клас register зчитує інформацію з полів в відповідні змінні, які в подальшому будуть використані в методі writeNewUser для передачі в локальну базу даних

```
private void writeNewUser() {
    displayLoader();
    Map<String, Object> newContact = new HashMap<>();
    newContact.put("Name", name);
    newContact.put("Gender", gender);
    newContact.put("Email", email);
    newContact.put("Profile_pic", photo_url);
    newContact.put("Password", password);
    newContact.put("UserID", userId);
    newContact.put("Phone", phone);
    newContact.put("State", state);
    newContact.put("Lga", lga);
    newContact.put("Country", "null");
```

```
        newContact.put("HomeAddress", home);
        newContact.put("Created_date", formattedDate);
        newContact.put("Date", date);

db.collection("Customers").document("users").collection("users").document
(userId).set(newContact)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
```

Вкладений метод який викликається при успішному переданні данних та виконує відправлення по листа верифікації на вклазану пошту

```
                    public void onSuccess(Void aVoid) {
                        sendEmailVerification();
                        Toast.makeText(RegisterActivity.this,
"Registration Successful", Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                                        @Override
                                        public void onFailure(@NonNull
Exception e) {

Toast.makeText(RegisterActivity.this, "ERROR" + e.toString(),

Toast.LENGTH_SHORT).show();
                                                pDialog.dismiss(); }}); }
```

Також як і в вікні логування в даному вікні присутній мето validateInputs який виконує перевірку введених даних, а для передачі даних на сервер використовується метод RegisterUser, лістинг коду якого описаний нижче

```
public void RegisterUser() {
        displayLoader();
        mAuth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new
OnCompleteListener<AuthResult>() {
                    @Override
```

```
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        try {
                            //check if successful
                            if (task.isSuccessful()) {

                                User user =
Auth.getInstance().getCurrentUser();

                                userId = user.getUid();
                                registerUser(name, image, status,
thumb_image);
                                writeNewUser();

                            } else {
                                Toast.makeText(RegisterActivity.this,
"Couldn't register, try again",
                                        Toast.LENGTH_SHORT).show();
                                pDialog.dismiss();
                            }
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }
                });
```

Наступним розробленим вікном було головне вікно додатку activity_main, яке відображається після входу в систему або при відкритті додатку якщо ви раніше ввійшли в систему. Дане вікно дає можливість перейти до вікон з продуктами, також до вікон Favorite, Search, Cart, Profile. У вікні Favorite відображаються страви які ви добавляєте туди як улюблені, у вікні Cart відображаються страви які ви добавили до корзини, а у вікні профіль дані користувача, які при запуску додатку завантажуються з серверу в локальну БД та відображаться за допомогою класу ProfileActivity. Вікно activity_main має наступний вигляд зображений на рисунку 3.14.

Рисунок 3.14 — Скріншот головного вікна

Також при вході в додаток він зчитує інформацію з геолокації і оновлює данні в базі даних серверу. Лістинг коду роботи з геолокацією зображено нижче, вікно LocationActivity з google map зображено на рисунку 3.15.

```
Geocoder geocoder = new Geocoder(context, Locale.getDefault());
                String result = null;
                try {
                    List<Address> addressList = geocoder.getFromLocation(
                            latitude, longitude, 1);
                    if (addressList != null && addressList.size() > 0) {
                        Address address = addressList.get(0);
                        StringBuilder sb = new StringBuilder();
                        for (int i = 0; i <
address.getMaxAddressLineIndex(); i++)
{sb.append(address.getAddressLine(i)).append("\n");}
sb.append(address.getAdminArea()).append(", ");
sb.append(address.getLocality()).append(", ");
sb.append(address.getFeatureName()).append(", ");
```

```
sb.append(address.getSubAdminArea()).append(", ");result =
sb.toString();}}
```

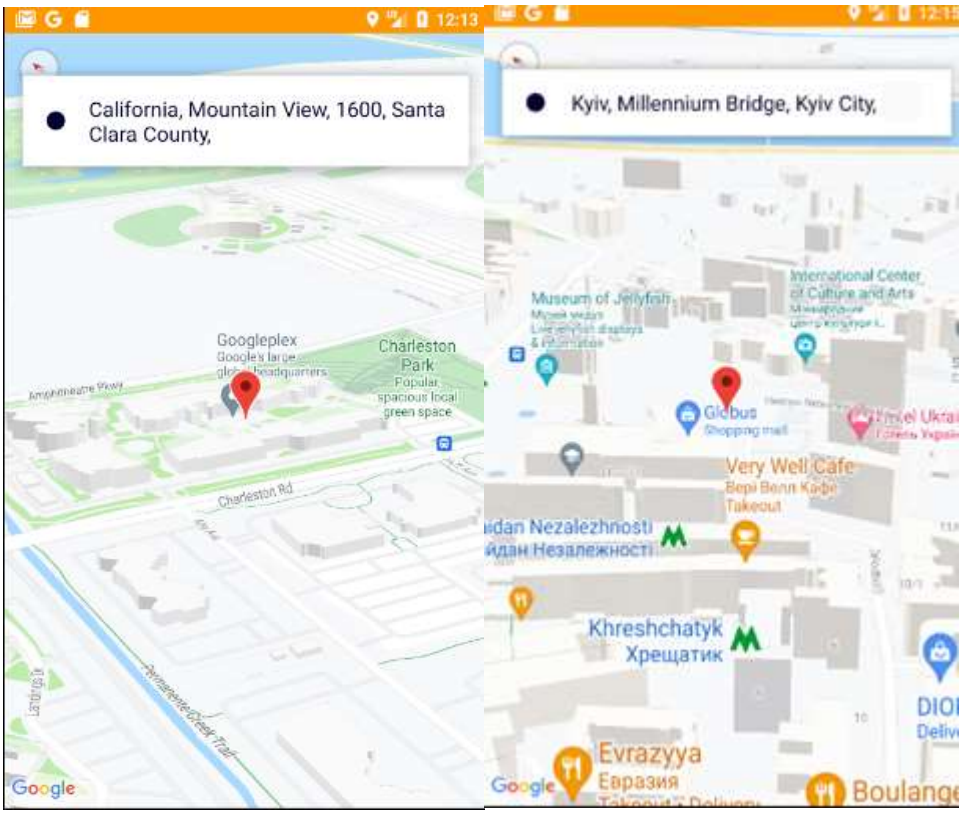


Рисунок 3.15 — Скріншоти вікна геолокації

Наступним кроком є реалізація вікон з стравами для замовлення які будуть реалізовані за допомогою структурного шаблону Adapter, візуальний інтерфейс даного вікна буде реалізовано в файлі activity_view_food, класи в яких зберігатиметься інформація про об’єкти: ItalianClass, FastfoodClass, TraditionalfoodClass, DrinkClass. Дані класи наслідуються від класу Food який має наступні поля дескриптор, та гетери для роботи з даними класами:

```
public class Food {
    private String Itemname;
    private String Itemprice;
    private int Itemimage;
    public String Itemquantity;
```

```
    public Food(String iName, String iPrice, int iImage, String
iquantity)
    {
        Itemname = iName;
        Itemprice = iPrice;
        Itemimage = iImage;
        Itemquantity = iquantity;
    }
    public String getItemName() {
        return Itemname;
    }
    public String getItemPrice() {return Itemprice;}
    public int getImageResourceId() {return Itemimage;}
    public String getItemquantity() {return Itemquantity;}
}
```

А реалізація зв’язування моделі з вью відбувається за допомогою відповідних класів Adapter, приклад створення інтерфейсу зображено на лістингу коду класу DrinkAdapter

```
DrinkClass currentcontifood = getItem(position);
        ImageView imageView = (ImageView)
listItemView.findViewById(R.id.item_image);

imageView.setImageResource(currentcontifood.getImageResourceId());
        TextView nameTextView = (TextView)
listItemView.findViewById(R.id.item_name);
        nameTextView.setText(currentcontifood.getItemName());
        TextView priceTextView = (TextView)
listItemView.findViewById(R.id.item_price);
        priceTextView.setText("Price " + currentcontifood.getItemPrice()
);
```

За таким сами принципом описані класи ItalianAdapter, FastfoodAdapter, TraditionalfoodAdapter.

Ще одним основним вікном яке ми розглянемо це вікно activity_menu, скріншот даного вікна зображений на рисунку 3.6

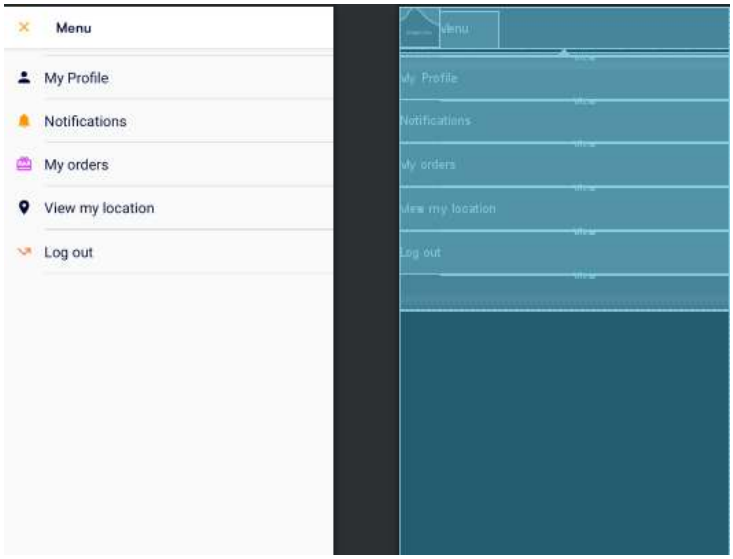


Рисунок 3.16 — Скріншот вікна activity_menu в редакторі

З даного меню ми також можемо перейти до вікна профілю, ще нам доступні такі вікна, Notification(також доступне з вікна activity_main через відповідне зображення), My orders в якому відображені наші минулі замовлення, та вікно View my location зображено на скріншотах 3.16 вище.

3.5 Тестування

При запуску додатку вперше відображається вікно входу в систему, та додаток запросить доступ до даних геолокації, якщо користувач був зареєстрований раніше то потрібно просто ввести валідні данні та натиснути кнопку «Login», якщо ж користувача немає аккаунту то потрібно натиснути на «Register» щоб перейти до вікна реєстрації. Ввівши валідні данні в відповідні поля в вікні регестрації та настиснувши кнопку «Register» ви перейдете до головного меню. В головному меню мають відображатися групи страв для замовлення. Тестування вікна реєстрації зображено на рисунку 3.17.

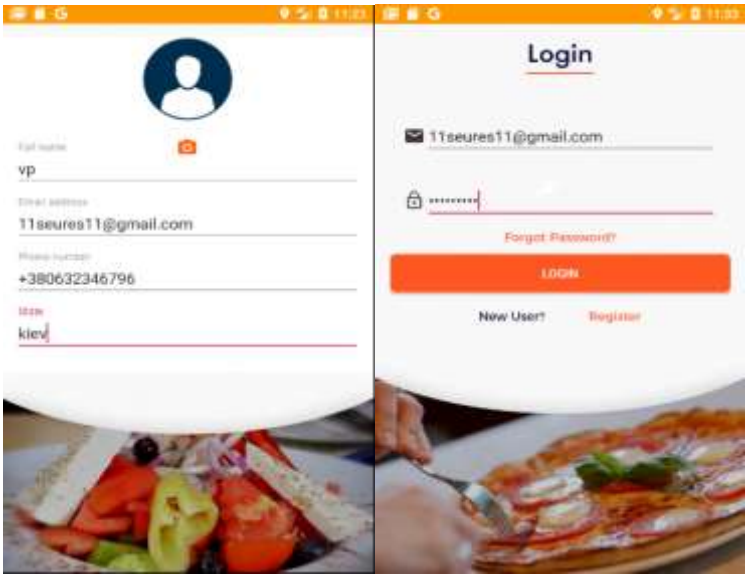


Рисунок 3.17 — Скріншоти вікон логіну та реєстрації

Після входу в систему нам відобображається головна сторінка з якої використовуючи меню навігації ми переходимо в профіль користувача для перевірки данних а також відкриваємо базу даних для перевірки інформації, скріншоти вікна користувача. Перехід повинен відбуватись коректно та на обране користувачем вікно. Також перевіремо редагування профілю, та звіримо інформацію з бази даних, нижче на рисунку 3.18 зображені скріншоти вікна користувача до і після редагування.

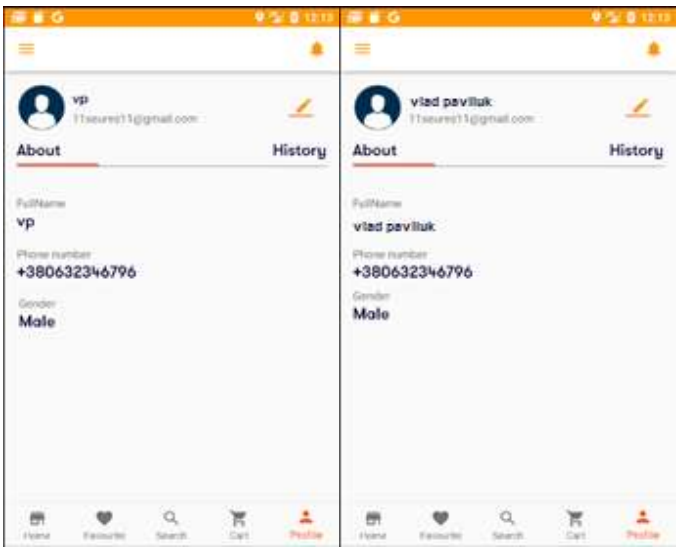


Рисунок 3.18 — скріншоти вікна користувача

Також перевіремо відображення даних в базі даних, та їх відповідну зміну до і після редагування, відповідні скріншоти з бази даних зображені на рисунку 3.19 нижче

id	FName	SName	login	password	phone_num	e_mail	sex
1	V	P	seu	8829224	380632346796	11seures11@gmail.com	Male
3	Олеся	Швець	teardekv	2855257	380974866144	teardekv@mail.com	Female
4	Артем	Сковорода	apotiavivm	1995083	380633510185	apotiavivm@mail.com	Male

id	FName	SName	login	password	phone_num	e_mail	sex
1	vlad	paliuk	seu	8829224	380632346796	11seures11@gmail.com	Male
7	Іван	Скрипка	ThathasbBagwp	2065150	380661650556	ThathasbBagwp@mail.com	Male
8	Іван	Поштар	okresanomuk	0255860	380665278537	okresanomuk@mail.com	Male

Рисунок 3.19 – Скріншоти з бази даних до і після редагування

Далі протестуємо роботу замовлення та відображення інформації про нього в десктоп додатку.

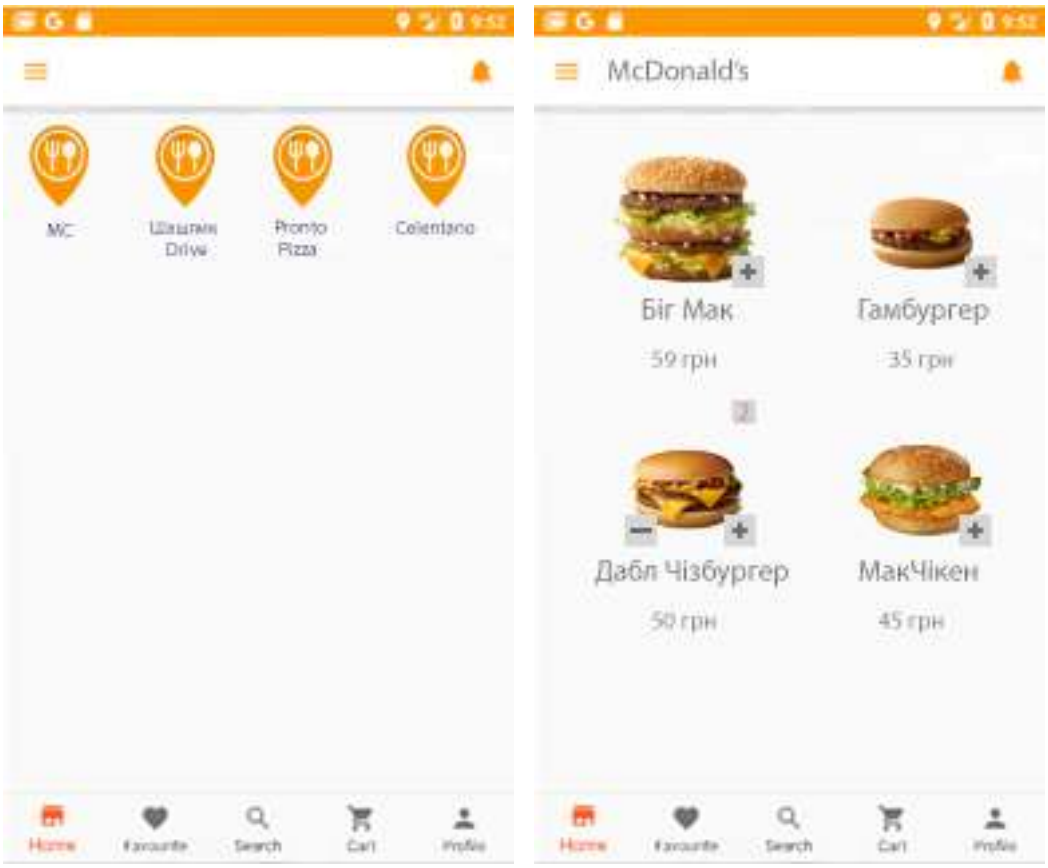


Рисунок 3.20 – Вибір замовлення

Обравши замовлення та назначивши пункт доставки, переходимо до десктоп додатку для перевірки даних про замовлення. Відкривши програму бачимо що в нас є нове замовлення. Далі переходим до замовлень та переглядаємо інформацію про замовлення, нижче на а рисунку 3.21 зображено головне вікно з повідомленням, та інформація про замовлення.

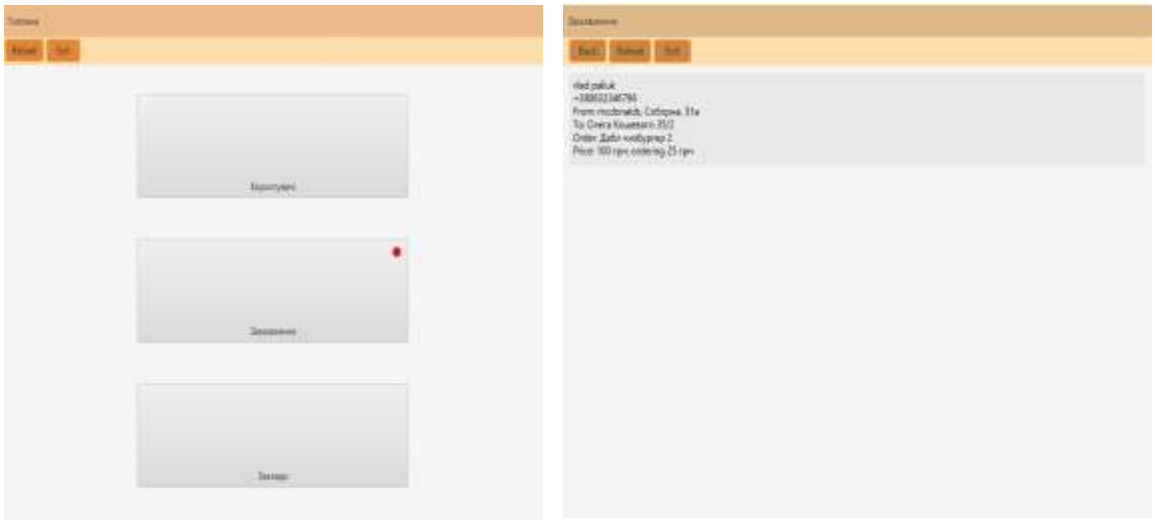


Рисунок 3.21 – Скріншот головного вікна та вікна заовлення.

Як бачимо інформація про замовлення відображається коректно а система працює справно.

3.6 Тестування якості формування рекомендацій

Для перевірки якості формування рекомендацій було обрано контрольну групу з 10 чоловік, які активно користуються додатками для замовлення доставки їжі. Кожен з яких створив новий профіль у додатках якими вони зазвичай користуються і також створили нові особисті профілі в тестованому додатку, і здійснювали замовлення користуючись рекомендаціями. А також при виконанні замовлення у тестованому додатку користувачі відкривали список додаткових замовлень, переглядали та у випадку бажання використовували дану додаткову можливість.

Під час тестування було проведено опитування що до якості надання рекомендацій закладів та страв у них. Оцінки були зібрані на наступних відрізках: 1-ше , 3-тє 5-те і 10-те замовлення; Обрахувавши оцінки надані оцінки надані тестовою групую були отримані наступні результати. На перше замовлення якість рекомендацій у аналоговому додатку була в середньому: 3.7 із 10, а у тестованому 3.5; на 3-тє: замовлення 4.8 в обох доатках; на 5-те: 6.1 проти 6.9; на 10-те: 7 в аналоговому проти 8.2 в тестованому. В той час додатковою функцією скористались на перше замовлення 4 із 10 опрошених, а в цілому за все тестування нею користувались приблизно у 35% .

Отже як бачимо з результатів на більшому відрізку якість формування рекомендацій збільшується в тестованому додатку, цьому сприяє удосконалений алгоритм рекомендацій, який в свою чергу збільшує кількість інформації про користувачів за рахунок більшої кількості замовлень.

3.6 Висновок до розділу 3

В даному розділі було програмно реалізовано 3 компонента системи, сервер-додаток, з підключеною базою даних, дексктоп-додаток для менеджменту, та мобільний клієнт-додаток. Також було реалізовано передачу даних між ними та формат зберігання локальних даних. Мобільний додаток було реалізовано згідно архітектури MVVM використовуючи плагін Dager.

Також було проведене тестування системи і тестування якості надання рекоендацій в порівнянні з аналогами, у результаті середня якість надання рекомендацій вища на 4.5%, а додатковою функцією користуютсья в 35% випадків.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки «інформаційна технологія замовлення їжі»

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, наведеними в таблиці. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали					
	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепції підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах

Продовження таблиці 4.1

1	2	3	4	5	6
Ринкові переваги					
1	2	3	4	5	6
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічна та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продуктивності продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Продовження таблиці 4.1

1	2	3	4	5	6
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 4.1

1	2	3	4	5	6
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5- ти років. Термін окупності інвестицій більший 5- ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки відображені в таблиці 4.2. В ній наведені опитування трьох експертів в галузі інформаційних технологій замовлення їжі. Після обрахування результатів оцінення буде сформований висновок щодо рівня науково-технічної розробки.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
Технічна здійсненність концепції	3	2	3
Ринкові переваги (наявність аналогів)	2	2	0
Ринкові переваги (ціна продукту)	3	3	3
Ринкові переваги (технічні властивості)	3	2	2
Ринкові переваги (експлуатаційні витрати)	2	2	3
Ринкові переваги (розмір ринку)	2	2	1
Ринкові переваги (конкуренція)	1	2	1
Практична здійсненність(наявність фахівців)	3	4	3
Практична здійсненність(наявність фінансів)	3	1	2
Практична здійсненність(необхідність нових матеріалів)	3	4	3
Практична здійсненність(термін реалізації)	3	4	3
Практична здійсненність(розробка документів)	2	2	2
Сума балів	30	30	26
Середньоарифметична сума балів	28.6		

Результати оцінки експертів було обраховано, та визначено рівень та комерційний потенціал розробки згідно рекомендацій наведених в таблиці 4.3, зображеній нижче

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Результатом рівня та потенціалу роботи стало значення 28.6, що відповідає значенню Середній відповідно до таблиці 4.3. Такі показники є результатом того що продукт не є новизною і має невеликі покращення на відміну від конкурентів та більш дешевий у розробці, а ціна надання послуг нижче ніж середня у конкурентів, що збільшує конкурентоспроможність проекту.

4.2 **Розрахунок витрат на здійснення науково-дослідної роботи**

Витрати, пов’язані з проведенням науково-дослідної, дослідно-конструкторської, конструкторсько-технологічної роботи, створенням дослідного зразка і здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

− витрати на оплату праці;

− відрахування на соціальні заходи;

− матеріали;

− паливо та енергія для науково-виробничих цілей;

− витрати на службові відрядження;

− спецустаткування для наукових (експериментальних) робіт;

− програмне забезпечення для наукових (експериментальних) робіт;

− витрати на роботи, які виконують сторонні підприємства, установи і організації;

− інші витрати;

− накладні (загальновиробничі) витрати.

4.2.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадкових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} * t_i}{T_p},$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p = 21…23 дні.

Проведені розрахунки зведено до таблиці 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	13200	742,84	23	17085
Науковий співробітник	10500	717,86	21	15075
Тестувальник	9500	616,7	22	13567
Лаборант	7300	526,38	21	11055
Всього				56782

Основна заробітна плата робітників: витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i * t_i,$$

де $С_і$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

$t_і$ – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду Сі можна визначити за формулою:

$$C_i = \frac{M_M * K_i * K_c}{T_p * t_{зм}},$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати, грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Координація роботи партнерів	7	5	1.5	106,12	742,84
Участь у впровадженні результатів проведених досліджень та розробок	7	4	1.5	102,55	717,86
Тестування	7	3	1,5	88,1	616,7
Проведення дослідів	6	2	1,5	87,73	526,38
Всього					2603,77

$$C_1 = \frac{6700 * 1{,}5 * 1{,}7}{23 * 7} = \frac{17085}{161} = 106{,}12;$$

$$C_2 = \frac{6700 * 1{,}5 * 1{,}5}{21 * 7} = \frac{15075}{147} = 102{,}55;$$

$$C_3 = \frac{6700 * 1{,}5 * 1{,}35}{22 * 7} = \frac{13567}{154} = 88{,}1;$$

$$C_4 = \frac{6700 * 1{,}5 * 1{,}1}{21 * 6} = \frac{11055}{126} = 87{,}73;$$

$$З_p = \sum_{i=1}^{n} C_i * t_i = 106{,}12 * 7 + 102{,}55 * 7 + 88{,}1 * 7 + 87{,}73 * 6$$

$$= 742{,}84 + 717{,}86 + 616{,}7 + 526{,}38 = 2603{,}77;$$

Додаткова заробітна плата: розраховується як 10-12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%} = (56782 + 2603{,}77) * \frac{10\%}{100\%};$$

$$З_{дод} = 5938{,}57$$

4.2.2. Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) * \frac{Н_{зп}}{100\%},$$

Де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (56782 + 2603{,}77 + 5938{,}57) * \frac{22\%}{100\%},$$

$$З_н = 14371{,}356$$

4.2.3 Сировина та матеріали

При виконанні науково-дослідницької роботи необхідності сировині та матеріалах не було.

4.2.4 Розрахунок витрат на комплектуючі

При виконанні науково-дослідницької роботи всі роботи проводились з використанням лише наявних комплектуючих.

4.2.5 Спецустаткування для наукових робіт

При виконанні науково-дослідницької роботи всі роботи проводились з використанням лише наявного спеціального обладнання.

4.3.6 Програмне забезпечення для наукових робіт

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10-12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$B_{\text{прг}} = \sum_{i=1}^{k} \text{Ц}_{\text{іпрг}} * \text{С}_{\text{прг.і}} * K_i,$$

де $\text{Ц}_{\text{іпрг}}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$\text{С}_{\text{прг.і}}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = 1,10…1,12);

k – кількість найменувань програмних засобів.

Отримані результати зведено до таблиці 4.6.

Таблиця 4.6 – Витрати на придбання програмних засобів кожного виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Intellij IDEA	1	500	500
Domain Website	1	400	400
Host Website	1	1100	1100
Всього			2000

$$B_{\text{прг}} = (500 * 1 * 1{,}1) + (400 * 1 * 1{,}1) + (1100 * 1 * 1{,}1) =$$

$$550 + 440 + 1210 = 2200;$$

4.3.7 Амортизація обладнання, програмних засобів та приміщень

При виконанні науково-дослідницької роботи всі роботи проводились з використанням лише наявного обладнання.

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію розраховуються за формулою:

$$B_{\text{е}} = \sum_{i=1}^{k} \frac{W_{yi} * t_i * \text{Ц}_{\text{е}} * K_{\text{впі}}}{\eta_i},$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$\text{Ц}_{\text{е}}$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії);

$K_{\text{впі}}$ – коефіцієнт, що враховує використання потужності;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Проведені розрахунки зведено до таблиці 4.7.

Таблиця 4.7 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Системний блок	0.500	184	713
Монітор	0.02	184	28,5
Всього			741.5

$$B_{\text{е}} = \frac{0{,}5 * 184 * 6{,}2 * 100\%}{80\%} + \frac{0{,}02 * 184 * 6{,}2 * 100\%}{80\%} = 713 + 28{,}5 = 741{,}5;$$

4.3.9 Службові відрядження

При виконанні науково-дослідницької роботи за даною темою, згідно плану, ніякі службові відрядження не проводились.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

У виконанні науково-дослідницької роботи сторонні організації участі не приймали.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуються як 50-100% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{І}_{\text{в}} = \left(З_{\text{о}} + З_{\text{р}}\right) * \frac{H_{\text{ів}}}{100\%},$$

де $H_{\text{ів}}$ – норма нарахування за статтею «Інші витрати».

$$\text{І}_{\text{в}} = (56782 + 2603{,}77) * \frac{65\%}{100\%} = 38600{,}75;$$

4.3.12 Накладні витрати

Витрати за статтею «Накладні витрати» розраховується як 100-150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%},$$

Де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) ви-трати».

$$В_{нзв} = (56782 + 2603{,}77) * \frac{132\%}{100\%} = 78389{,}21;$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + В_{прг} + В_е + І_в + В_{нзв},$$

$$В_{заг} = 56782 + 2603{,}77 + 5938{,}57 + 14371{,}356 + 2200 + 741{,}5 + 38600{,}75 + 78389{,}21$$

$$В_{заг} = 199627{,}1624;$$

Загальні витрати ЗВ на завершення науково-технічної роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta},$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться на стадії:

науково-дослідних робіт, то =0,1; технічного проектування, то η =0,2; розробки конструкторської документації, то η =0,3; розробки технологій, то η =0,4; розробки дослідного зразка, то η =0,5; розробки промислового зразка, то η =0,7; впровадження, то η =0,9.

$$ЗВ = \frac{В_{заг}}{\eta} = 199627{,}1624/0{,}4 = 499067{,}91;$$

4.3 Оцінювання економічної ефективності науково-технічної розробки

На ринку інвестицій фактор який може привернути увагу потенційних інвесторів це ріст чистого прибутку цього інвестора .

У ході дослідження за темою «Інформаційна система замовлення їжі» були отримані результати які передбачають комерціалізацію на протязі 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту буде формувати:

ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу;

Таблиця 4.8 – Збільшення кількості споживачів за кожен рік

Показник	1й рік	2й рік	3й рік	4й рік
Збільшення кількості споживачів, осіб(ΔN)	8000	9500	10500	13000

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження нової науково-дослідної розробки, приймемо 35000 осіб;

$Ц_б$ - вартість послуги у році до впровадження інформаційної системи, приймемо 150,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо 25,00 грн;

Розрахунок прибутку у потенційного інвестора $\Delta П_i$ для кожного з чотирьох років, у результаті можливого впровадження і комерціалізації розробки розраховується за наступною формулою:

$$\Delta П_i = (\pm\Delta Ц_i * N + Ц_o * \Delta N) * \lambda * \rho * \left(1 - \frac{\vartheta}{100}\right),$$

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість становить 20%, а коефіцієнт λ = 0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо $\rho = 0{,}35$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ = 18%.

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (25{,}00 * 35000 + 175 * 8000) * 0{,}83 * 0{,}35 * \left(1 - \frac{18\%}{100}\right);$$

$$\Delta П_1 = 541927{,}8 \text{ грн};$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (25{,}00 * 35000 + 175 * 17500) * 0{,}83 * 0{,}35 * \left(1 - \frac{18\%}{100}\right);$$

$$\Delta П_2 = 937951{,}9 \text{ грн};$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (25{,}00 * 35000 + 175 * 28000) * 0{,}83 * 0{,}35 * \left(1 - \frac{18\%}{100}\right);$$

$$\Delta П_3 = 1375663 \text{ грн};$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (25{,}00 * 35000 + 175 * 41000) * 0{,}83 * 0{,}35 * \left(1 - \frac{18\%}{100}\right);$$

$$\Delta П_4 = 1917591 \text{ грн};$$

Приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-дослідної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t},$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}1$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році;

$$ПП = \frac{541927{,}8}{1{,}1^1} + \frac{937951{,}9}{1{,}1^2} + \frac{1375663}{1{,}1^3} + \frac{1917591}{1{,}1^4};$$

$$ПП = 492661{,}6 + 775166{,}8 + 1033555{,}1 + 3611124{,}3;$$

$$ПП = 3611124{,}3 \text{ грн};$$

Величина початкових інвестицій , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{\text{інв}} * ЗВ,$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науко-во-технічної розробки та її комерціалізацію. Це можуть бути витрати на під-готовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$ = 2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 499067,91 грн.

$$PV = 2 * 499067{,}91 = 998135{,}82 \text{ грн};$$

Абсолютний економічний ефект або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{abc} = ПП - PV,$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, приймаємо 3611124,3 грн;

PV – теперішня вартість початкових інвестицій, 998135,82 грн.

$$Е_{abc} = 3611124{,}3 - 998135{,}82 = 2612988{,}48 \text{ грн};$$

Внутрішня економічна дохідність інвестицій , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$\mathrm{E_в} = \sqrt[\text{Тж}]{1 + \frac{E_{abc}}{PV}} - 1,$$

де E_{abc} – абсолютний економічний ефект вкладених інвестицій, 2612988,48 грн;

PV – теперішня вартість початкових інвестицій, 998135,82 грн;

Тж – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$\mathrm{E_в} = \sqrt[4]{1 + \frac{2612988,48}{998135,82}} - 1 = \sqrt[4]{3,62} - 1 = 1,379 - 1 = 0,379;$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій визначається за формулою:

$$\tau_{\text{мін}} = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = 0,1;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f = 0,1.

$$\tau_{\text{мін}} = 0{,}1 + 0{,}1 = 0{,}2;$$

$0{,}2 < 0{,}33$ свідчить про те, що внутрішня економічна дохідність інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-дослідної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія замовлення їжі» доцільно.

Період окупності інвестицій, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$\text{Т}_{\text{ок}} = \frac{1}{\text{Е}_{\text{в}}},$$

де $\text{Е}_{\text{в}}$ – внутрішня економічна дохідність вкладених інвестицій

$$\text{Т}_{\text{ок}} = \frac{1}{0{,}379} = 2{,}637;$$

$\text{Т}_{\text{ок}} < 3 -$ х років, це свідчить про комерційну привабливість науково-технічної розробки і може спонукати інвесторів профінансувати впровадження даної розробки та виведення її на ринок.

4.4 Висновок до розділу 4

У даному розділі було проведено науковий аудит науково-дослідної роботи «Інформаційна система замовлення їжі». Було виконано оцінювання наукового ефекту. Визначено новизну роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації.

Також були проведені розрахунки затрат, які пов’язані з витратами на науково-дослідницьку розробку, загальні витрати на розробку склали 499067,91 грн. У ці витрати були включені витрати на оплату праці персоналу, ціна на електроенергію, на програмне забезпечення та інші витрати. Після чого було проведено розрахунок періоду окупності інвестицій, він склав 2,637 роки, приблизно 2 роки 7 місяців.

Результат цього розділу є розрахунок вартості та реалізації науково-дослідницької роботи «Інформаційна система замовлення їжі»

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було розроблено удосконалений алгоритм рекомендацій закладів та продуктів для замовлення, який на відміну від аналогів надає більш якісні рекомендації що до закладу та страв у них, використовуючи не тільки дані про знаходження користувача, а маршрут замовлення, і відповідно до маршруту надаються додаткові рекомендації.

В першому розділі було проаналізовано особливості роботи систем замовлення їжі. Були розглянуті переваги та недоліки сучасних програм-аналогів які використовуються для замовлення доставки їжі. Також було розглянуто алгоритми надання рекомендацій які використовуються в аналогових додатках розглянуто їх переваги та недоліки. Наведено опис основних функцій, які виконують дані програми, у результаті цього був сформовані вимоги до розробки інформаційної технології замовлення їжі.

У другому розділі були проаналізовані засоби розробки кожного компоненту системи. Були розглянуті дві платформи для розробки мобільного додатку Android і Apple також були розглянуті середовища розробки та мови програмування до кожної з платформ, були обрані та обгрунтовані найбільш підходящі інструменти для розробки.

Також були розглянуті та обгрунтовані інструменти розробки інших компонентів системи. Розроблені алгоритми загальної роботи серверу та алгоритму авторизації та контролю доступу до даних і спроектовано удосконалений алгоритм рекомендацій закладів та страв у них.

У третьому розділі було програмно реалізовано 3 компонента системи, сервер-додаток, з підключеною базою даних, дексктоп-додаток для менеджменту, та мобільний клієнт-додаток. Також було реалізовано передачу даних між ними та формат зберігання локальних даних. Мобільний додаток було реалізовано згідно архітектури MVVM використовуючи плагін Dager.

Було проведене тестування системи, всі виявлені у результаті тестування помилки та недоліки було усунено. Також була здійснена оцінка якості надання

рекоендацій додатком у порівнянні з аналогами, в результаті в середьому якість надання рекомендацій вища на 4,5%, а функцією додатковго замовлення користувались у 35% випадках

У четвертому розділі було здійснене економічне обґрунтування розробки інформаційної технології замовлення їжі. Були спрогнозовані можливі витрати по кожній із статей, також був розрахований чистий прибуток. По даним розрахункам можна сказати, що розробка інформаційної технології замовлення їжі є доцільно. Період окупності складає 2 роки 7 місяці а загальні витрати становлять 499067,91 грн

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

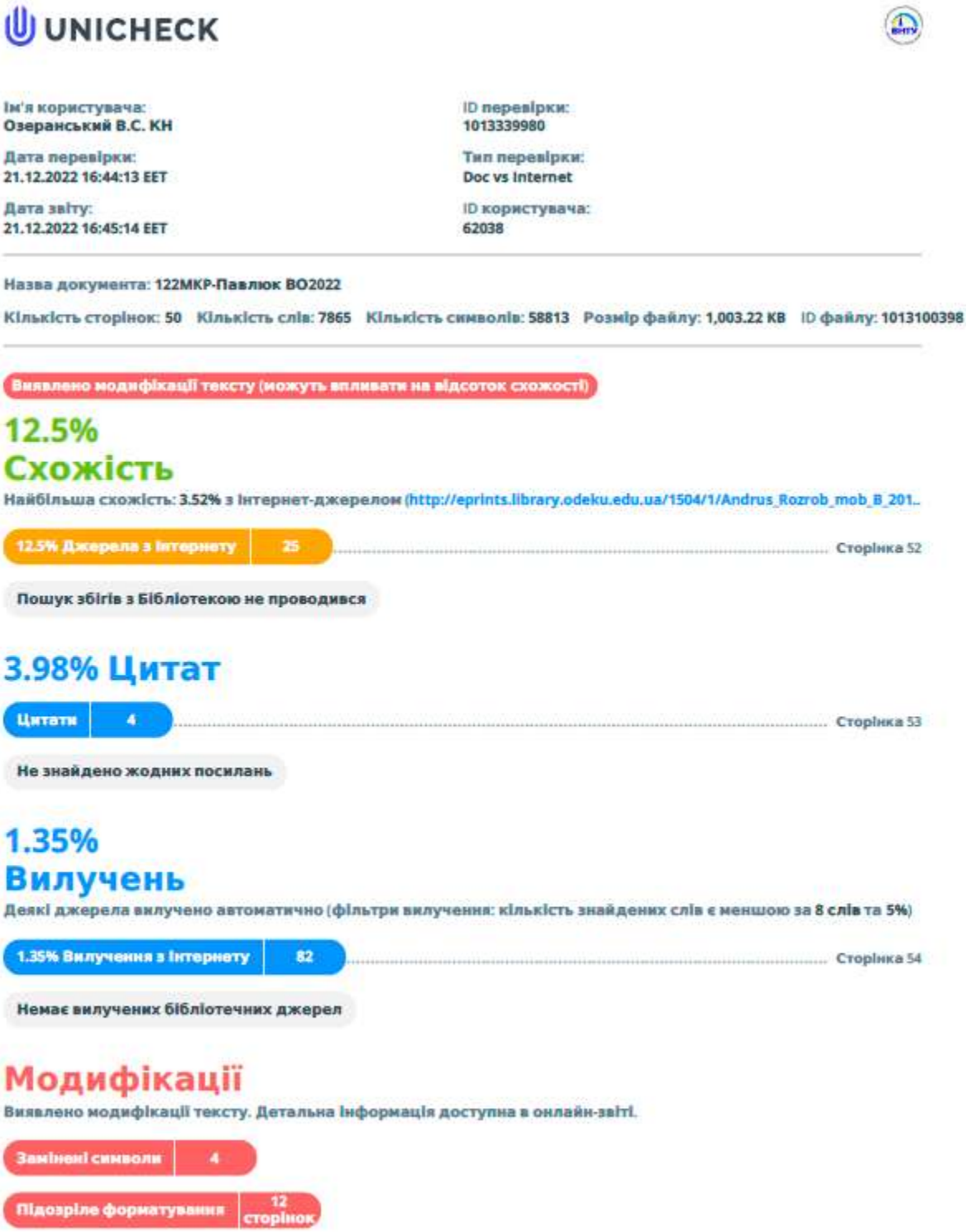
1. Павлюк В. О., Колодний В. В. «Особливості алгоритмів рекомонедації закладів та товарів для додатків замовлення їжі», в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)», Вінниця, 2022, [Електронний ресурс]. Режим доступу:
2. Caviar’s Food Recommendation Platform [Електронний ресурс] Режим доступу: https://developer.squareup.com/blog/caviars-food-recommendation-platform/
3. What is cluster analysis? [Електронний ресурс] Режим доступу: https://www.qualtrics.com/experience-management/research/cluster-analysis/
4. Системи рекомендацій, задачі, підходи, алгоритми. [Електронний ресурс] Режим доступу: https://datareview.info/article/sistemyi-rekomendatsiy-zadachi-podhodyi-algoritmyi/
5. Recommendation systems in E-commerce: What’s the thing you’ve never known, but always wanted to? [Електронний ресурс] Режим доступу: https://www.be-terna.com/insights/recommendation-systems-in-e-commerce-whats-the-thing-youve-never-known-but-always-wanted-to
6. iOS [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/IOS
7. Xcode [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Xcode
8. Kotlin [Електронний ресурс] // Режим доступу: https:// https://uk.wikipedia.org/wiki/Kotlin
9. Android studio [Електронний ресурс] // Режим доступу: https:// https://uk.wikipedia.org/wiki/Android_Studio
10. Андерс Йоранссон Ефективне використання потоків в операційній системі Android
11. Ияну Аделекан Kotlin. Програмування на прикладах

12. Федотенко М. Розробка мобільних додатків. Перші кроки.
13. Герберт Шилдт. Java 8. Посібник для початківців 2016. – 669 с. – (2015669). – (958-7-8459-1938-1; кн. 789).
14. Fundamentals of Testing [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://developer.android.com/training/testing/fundamentals
15. Software R. Understanding The Difference Between MVC, MVP and MVVM Design Patterns [Електронний ресурс] / Rishabh Software. – 2021. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/understanding-difference-between-mvc-mvp-mvvm-design-rishabh-software/.
16. SQL і NoSQL: розбираємося в основних моделях баз даних[Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://tproger.ru/translations/sql-nosql-database-models/.

ДОДАТКИ

Додаток А (обов'язковий)

Результат перевірки на плагіат в онлайн-системі UNICHECK



Додаток Б (обов'язковий)

Лістинг програми

```
PUBLIC CLASS PROFILEACTIVITY EXTENDS APPCOMPATACTIVITY {
    DBSTORE DB;
    DBUSER USER;
    DOCUMENTREFERENCE USERS;
    PRIVATE STRING EMAIL, STATE, LGA, NAME, WORK, PHONE, HOME,
USERID, STATUS, GENDER, PHOTO_URL, ID;
    PRIVATE IMAGEVIEW IMAGEVIEW;
    TEXTVIEW NAME1, NAME2, EMAILTEXT, PHONETEXT, LOCATIONTEXT,
GENDERTEXT;
    @OVERRIDE
    PROTECTED VOID ONCREATE(BUNDLE SAVEDINSTANCESTATE) {
        SUPER.ONCREATE(SAVEDINSTANCESTATE);
        SETCONTENTVIEW(R.LAYOUT.ACTIVITY_PROFILE);
        GETSUPPORTACTIONBAR().HIDE();
        TRANSPARENTSTATUSANDNAVIGATION();
        IMAGEVIEW = FINDVIEWBYID(R.ID.CIRCLEIMAGEVIEWUSERIMAGE);

        NAME1 = FINDVIEWBYID(R.ID.NAME);
        NAME2 = FINDVIEWBYID(R.ID.NAME2);
        EMAILTEXT = FINDVIEWBYID(R.ID.EMAIL);
        PHONETEXT = FINDVIEWBYID(R.ID.PHONE);
        LOCATIONTEXT = FINDVIEWBYID(R.ID.LOCATION);
        GENDERTEXT = FINDVIEWBYID(R.ID.GENDER);
        USER = DBAUTH.GETINSTANCE().GETCURRENTUSER();
        USERID = USER.GETUID();
        DB = DBSTORE.GETINSTANCE();
        GETUSERS();

    }

    PRIVATE VOID GETUSERS() {
        USERS =
DB.COLLECTION("CUSTOMERS").DOCUMENT("USERS").COLLECTION("USERS").DO
CUMENT(USERID);
        USERS.GET().ADDONCOMPLETELISTENER(NEW
ONCOMPLETELISTENER<DOCUMENTSNAPSHOT>() {
@OVERRIDE
PUBLIC VOID ONCOMPLETE(@NONNULL TASK<DOCUMENTSNAPSHOT> TASK) {
IF (TASK.ISSUCCESSFUL()) {
DOCUMENTSNAPSHOT DOC = TASK.GETRESULT();
  IF (DOC.EXISTS()) {
             NAME = DOC.GETSTRING("NAME");
    EMAIL = DOC.GETSTRING("EMAIL");
          PHOTO_URL = DOC.GETSTRING("PROFILE_PIC");
          PHONE = DOC.GETSTRING("PHONE");
    GENDER = DOC.GETSTRING("GENDER");
       HOME = DOC.GETSTRING("HOMEADDRESS");
           NAME1.SETTEXT(NAME);
NAME2.SETTEXT(NAME);
```

```
EMAILTEXT.SETTEXT(EMAIL);
LOCATIONTEXT.SETTEXT(HOME);
PHONETEXT.SETTEXT(PHONE);
GENDERTEXT.SETTEXT(GENDER);
PICASSO.WITH(PROFILEACTIVITY.THIS)
LOAD(PHOTO_URL)
PLACEHOLDER(R.DRAWABLE.PIC)
INTO(IMAGEVIEW);
}ELSE {
TOAST.MAKETEXT(PROFILEACTIVITY.THIS, "USER DOES NOT EXIT",
TOAST.LENGTH_SHORT).SHOW();
}}}}).ADDONFAILURELISTENER(NEW ONFAILURELISTENER() {
@OVERRIDE
PUBLIC VOID ONFAILURE(@NONNULL EXCEPTION E) {
TOAST.MAKETEXT(PROFILEACTIVITY.THIS, "ERROR" + E,
TOAST.LENGTH_LONG).SHOW();
}});}
    @OVERRIDE
    PUBLIC VOID ONBACKPRESSED() {
        SUPER.ONBACKPRESSED();}
    PUBLIC VOID BACK(VIEW VIEW) {
        ONBACKPRESSED();}
    PUBLIC VOID CHAT(VIEW VIEW) {
        FINISH();
        INTENT INTENT = NEW INTENT(THIS,
EDITPROFILEACTIVITY.CLASS);
        INTENT.PUTEXTRA("NAME", NAME);
        INTENT.PUTEXTRA("EMAIL", EMAIL);
        INTENT.PUTEXTRA("GENDER", GENDER);
        INTENT.PUTEXTRA("PHOTO", PHOTO_URL);
        INTENT.PUTEXTRA("PHONE", PHONE);
        INTENT.PUTEXTRA("HOME", HOME);
        INTENT.PUTEXTRA("USERID", USERID);
        STARTACTIVITY(INTENT);}
    PRIVATE VOID TRANSPARENTSTATUSANDNAVIGATION() {
        //MAKE FULL TRANSPARENT STATUSBAR
        IF (BUILD.VERSION.SDK_INT >= 19 && BUILD.VERSION.SDK_INT <
21) {
            SETWINDOWFLAG(THIS,
WINDOWMANAGER.LAYOUTPARAMS.FLAG_TRANSLUCENT_STATUS, TRUE);
        }
        IF (BUILD.VERSION.SDK_INT >= 19) {

GETWINDOW().GETDECORVIEW().SETSYSTEMUIVISIBILITY(VIEW.SYSTEM_UI_FLA
G_LAYOUT_STABLE | VIEW.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        }
        //MAKE FULLY ANDROID TRANSPARENT STATUS BAR
        IF (BUILD.VERSION.SDK_INT >= 21) {
            SETWINDOWFLAG(THIS,
WINDOWMANAGER.LAYOUTPARAMS.FLAG_TRANSLUCENT_STATUS, FALSE);
            GETWINDOW().SETSTATUSBARCOLOR(COLOR.TRANSPARENT);}}
    PUBLIC STATIC VOID SETWINDOWFLAG(ACTIVITY ACTIVITY, FINAL INT
BITS, BOOLEAN ON) {
        WINDOW WIN = ACTIVITY.GETWINDOW();
```

```
        WINDOWMANAGER.LAYOUTPARAMS WINPARAMS = WIN.GETATTRIBUTES();
        IF (ON) {
            WINPARAMS.FLAGS |= BITS;
        } ELSE {
            WINPARAMS.FLAGS &= ~BITS;
        }
        WIN.SETATTRIBUTES(WINPARAMS);
    }
}


PUBLIC CLASS PROFILEFRAGMENT EXTENDS FRAGMENT {
PRIVATE VIEW ROOTVIEW;
PRIVATE TEXTVIEW ABOUT, HISTORY, NAMETEXT, EMAILTEXT, NAMETEXT2,
PHONETEXT, LOCATIONTEXT, GENDERTEXT;
    PRIVATE STRING EMAIL, STATE, LGA, NAME, WORK, PHONE, HOME,
USERID, STATUS, GENDER, PHOTO_URL, ID;
PRIVATE IMAGEVIEW EDIT_PROFILE;
PRIVATE LINEARLAYOUT ABOUT_LAYOUT;
PRIVATE RECYCLERVIEW HISTORY_LAYOUT;
PRIVATE VIEW ABOUT_LINE, HISTORY_LINE;
PRIVATE IMAGEVIEW IMAGEVIEW, EDITPROFILE;
    DBSTORE DB;
    DBUSER USER;
    DOCUMENTREFERENCE USERS;
    PRIVATE HISTORYADAPTER RECYCLERVIEWADAPTER;
    PRIVATE LIST<FOOD> EVENTLIST;

    PUBLIC PROFILEFRAGMENT() {
    }
    @OVERRIDE
    PUBLIC VIEW ONCREATEVIEW(LAYOUTINFLATER INFLATER, VIEWGROUP
CONTAINER,
                             BUNDLE SAVEDINSTANCESTATE) {

        ROOTVIEW = INFLATER.INFLATE(R.LAYOUT.FRAGMENT_PROFILE,
CONTAINER, FALSE);
        ABOUT = ROOTVIEW.FINDVIEWBYID(R.ID.ABOUT_TEXT);
        HISTORY = ROOTVIEW.FINDVIEWBYID(R.ID.HISTORY_TEXT);
        ABOUT_LAYOUT = ROOTVIEW.FINDVIEWBYID(R.ID.LAYOUT_ABOUT);
        HISTORY_LAYOUT =
ROOTVIEW.FINDVIEWBYID(R.ID.HISTORY_RECYCLER);
        ABOUT_LINE = ROOTVIEW.FINDVIEWBYID(R.ID.ABOUT_LINE);
        HISTORY_LINE = ROOTVIEW.FINDVIEWBYID(R.ID.HISTORY_LINE);
        EDIT_PROFILE = ROOTVIEW.FINDVIEWBYID(R.ID.EDIT_PROFILE);
        IMAGEVIEW = ROOTVIEW.FINDVIEWBYID(R.ID.IMGSENDER);
        NAMETEXT = ROOTVIEW.FINDVIEWBYID(R.ID.NAME);
        EMAILTEXT = ROOTVIEW.FINDVIEWBYID(R.ID.EMAIL);
        EDIT_PROFILE = ROOTVIEW.FINDVIEWBYID(R.ID.EDIT_PROFILE);
        NAMETEXT2 = ROOTVIEW.FINDVIEWBYID(R.ID.NAME2);
        PHONETEXT = ROOTVIEW.FINDVIEWBYID(R.ID.PHONE);
        LOCATIONTEXT = ROOTVIEW.FINDVIEWBYID(R.ID.LOCATION);
        GENDERTEXT = ROOTVIEW.FINDVIEWBYID(R.ID.GENDER);
        USER = DBAUTH.GETINSTANCE().GETCURRENTUSER();
```

```
        USERID = USER.GETUID();
        DB = DBSTORE.GETINSTANCE();
        GETUSERS();
        EVENTLIST = NEW ARRAYLIST<>();
        LINEARLAYOUTMANAGER RECYCLERLAYOUTMANAGER =
                NEW LINEARLAYOUTMANAGER(GETCONTEXT());
        HISTORY_LAYOUT.SETLAYOUTMANAGER(RECYCLERLAYOUTMANAGER);
        DIVIDERITEMDECORATION DIVIDERITEMDECORATION =
                NEW
DIVIDERITEMDECORATION(HISTORY_LAYOUT.GETCONTEXT(),
                        RECYCLERLAYOUTMANAGER.GETORIENTATION());
        HISTORY_LAYOUT.ADDITEMDECORATION(DIVIDERITEMDECORATION);
        GETFRIENDLIST();
        EDIT_PROFILE.SETONCLICKLISTENER(NEW VIEW.ONCLICKLISTENER()
{
            @OVERRIDE
            PUBLIC VOID ONCLICK(VIEW V) {
                INTENT INTENT = NEW INTENT(GETCONTEXT(),
EDITPROFILEACTIVITY.CLASS);
                INTENT.PUTEXTRA("NAME", NAME);
                INTENT.PUTEXTRA("EMAIL", EMAIL);
                INTENT.PUTEXTRA("GENDER", GENDER);
                INTENT.PUTEXTRA("PHOTO", PHOTO_URL);
                INTENT.PUTEXTRA("PHONE", PHONE);
                INTENT.PUTEXTRA("HOME", HOME);
                INTENT.PUTEXTRA("USERID", USERID);
                STARTACTIVITY(INTENT);
            }});
        ABOUT.SETONCLICKLISTENER(NEW VIEW.ONCLICKLISTENER() {
            @OVERRIDE
            PUBLIC VOID ONCLICK(VIEW V) {
                HISTORY_LAYOUT.SETVISIBILITY(VIEW.GONE);
                ABOUT_LAYOUT.SETVISIBILITY(VIEW.VISIBLE);
                HISTORY_LINE.SETVISIBILITY(VIEW.GONE);
                ABOUT_LINE.SETVISIBILITY(VIEW.VISIBLE);
            }});
        HISTORY.SETONCLICKLISTENER(NEW VIEW.ONCLICKLISTENER() {
            @OVERRIDE
            PUBLIC VOID ONCLICK(VIEW V) {
                HISTORY_LAYOUT.SETVISIBILITY(VIEW.VISIBLE);
                ABOUT_LAYOUT.SETVISIBILITY(VIEW.GONE);
                HISTORY_LINE.SETVISIBILITY(VIEW.VISIBLE);
                ABOUT_LINE.SETVISIBILITY(VIEW.GONE);
            }});
        RETURN ROOTVIEW; }
    PRIVATE VOID GETUSERS() {
        USERS =
DB.COLLECTION("CUSTOMERS").DOCUMENT("USERS").COLLECTION("USERS").DO
CUMENT(USERID);
        USERS.GET().ADDONCOMPLETELISTENER(NEW
ONCOMPLETELISTENER<DOCUMENTSNAPSHOT>() {
                                                @OVERRIDE
                    PUBLIC VOID ONCOMPLETE(@NONNULL
TASK<DOCUMENTSNAPSHOT> TASK) {
```

```
                        IF (TASK.ISSUCCESSFUL()) {
                             DOCUMENTSNAPSHOT DOC =
TASK.GETRESULT();
                                  IF (DOC.EXISTS()) {
                                       NAME =
DOC.GETSTRING("NAME");
                                       EMAIL =
DOC.GETSTRING("EMAIL");
                                       PHOTO_URL =
DOC.GETSTRING("PROFILE_PIC");
                                       PHONE =
DOC.GETSTRING("PHONE");
                                       GENDER =
DOC.GETSTRING("GENDER");
                                       HOME =
DOC.GETSTRING("HOMEADDRESS");
                                       NAMETEXT.SETTEXT(NAME);
                                       NAMETEXT2.SETTEXT(NAME);
                                       EMAILTEXT.SETTEXT(EMAIL);
                                       LOCATIONTEXT.SETTEXT(HOME);
                                       PHONETEXT.SETTEXT(PHONE);
                                       GENDERTEXT.SETTEXT(GENDER);

        PICASSO.WITH(GETCONTEXT()).LOAD(PHOTO_URL)

        .PLACEHOLDER(R.DRAWABLE.PIC)
                                       .INTO(IMAGEVIEW);
                                       }ELSE {

        TOAST.MAKETEXT(GETCONTEXT(), "USER DOES NOT EXIT",
TOAST.LENGTH_SHORT).SHOW();}}}})
                .ADDONFAILURELISTENER(NEW ONFAILURELISTENER() {
                    @OVERRIDE
                    PUBLIC VOID ONFAILURE(@NONNULL EXCEPTION E) {
                        TOAST.MAKETEXT(GETCONTEXT(), "ERROR" + E,
TOAST.LENGTH_LONG).SHOW();
                    }  }); }
    PRIVATE VOID GETFRIENDLIST() {

DB.COLLECTION("CUSTOMERS").DOCUMENT("USERS").COLLECTION("USERS").DO
CUMENT(USERID).COLLECTION("HISTORY").ORDERBY("DATE",
QUERY.DIRECTION.DESCENDING)
                .GET()
                .ADDONCOMPLETELISTENER(NEW
ONCOMPLETELISTENER<QUERYSNAPSHOT>() {
                    @OVERRIDE
                    PUBLIC VOID ONCOMPLETE(@NONNULL
TASK<QUERYSNAPSHOT> TASK) {
                        IF (TASK.ISSUCCESSFUL()) {
                            FOR (DOCUMENTSNAPSHOT DOC :
TASK.GETRESULT()) {
                                FOOD E = DOC.TOOBJECT(FOOD.CLASS);
                                IF (E != NULL){
```

```
                                    }
                                    E.SETID(DOC.GETID());
                                    EVENTLIST.ADD(E);
                                }
                                RECYCLERVIEWADAPTER = NEW
                                        HISTORYADAPTER(EVENTLIST,
                                        GETACTIVITY());

HISTORY_LAYOUT.SETADAPTER(RECYCLERVIEWADAPTER);

                            } ELSE {
                                TOAST.MAKETEXT(GETCONTEXT(), "ERROR: "
+ TASK.GETEXCEPTION(), TOAST.LENGTH_LONG).SHOW();
                            }}});

DB.COLLECTION("CUSTOMERS").DOCUMENT("USERS").COLLECTION("USERS").DO
CUMENT(USERID).COLLECTION("HISTORY").ORDERBY("DATE",
QUERY.DIRECTION.DESCENDING)
                    .ADDSNAPSHOTLISTENER( NEW
EVENTLISTENER<QUERYSNAPSHOT>() {
                        @OVERRIDE
                        PUBLIC VOID ONEVENT(QUERYSNAPSHOT
DOCUMENTSNAPSHOTS, DBSTOREEXCEPTION E) {

                            FOR (DOCUMENTCHANGE DOC :
DOCUMENTSNAPSHOTS.GETDOCUMENTCHANGES()) {
                                DOC.GETDOCUMENT().TOOBJECT(FOOD.CLASS);
}}});}}
```

Додаток В (обов’язковий)

Додаток В (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ЗАМОВЛЕНЯ ЇЖІ

Виконав: студент 2-го курсу,
групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Павлюк В.О.
(прізвище та ініціали)

Керівник: к.т.н., ст. доцент каф. КН
Колодний В.В.
(прізвище та ініціали)
«15» 12 2022 р.

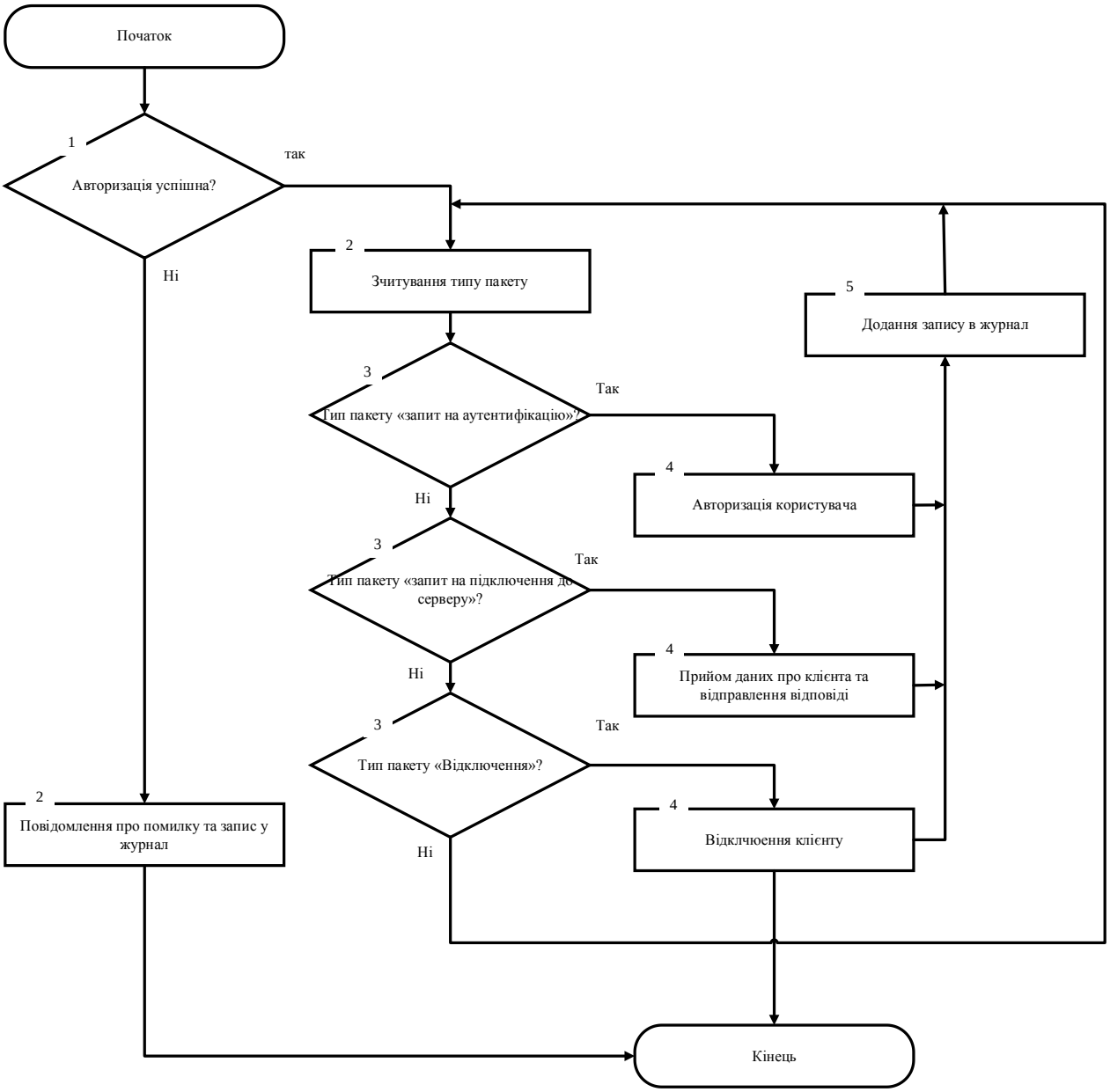


Рисунок В.1 – Блок-схема алгоритму обробки запитів сервером

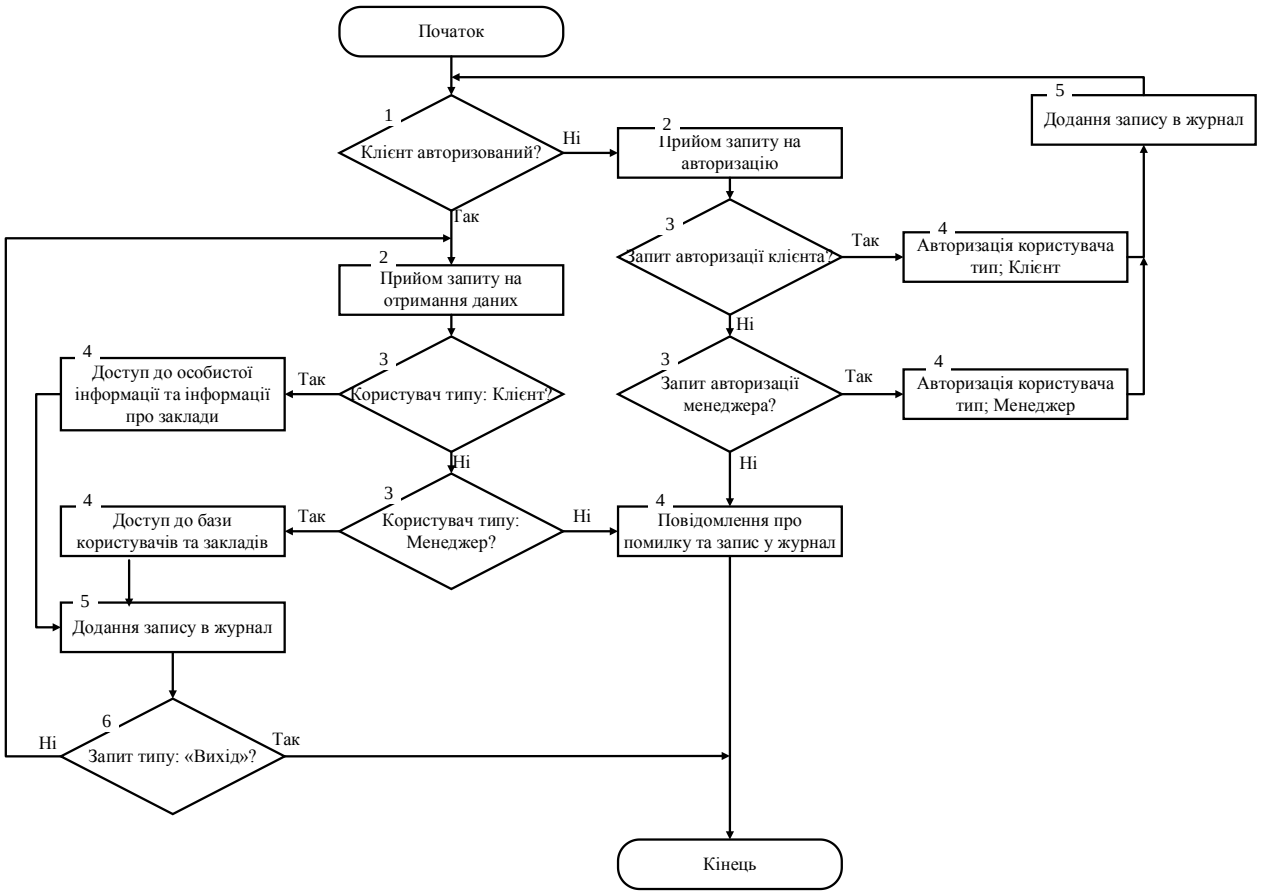


Рисунок В.2 – Алгоритм авторизації та контролю доступу до даних.

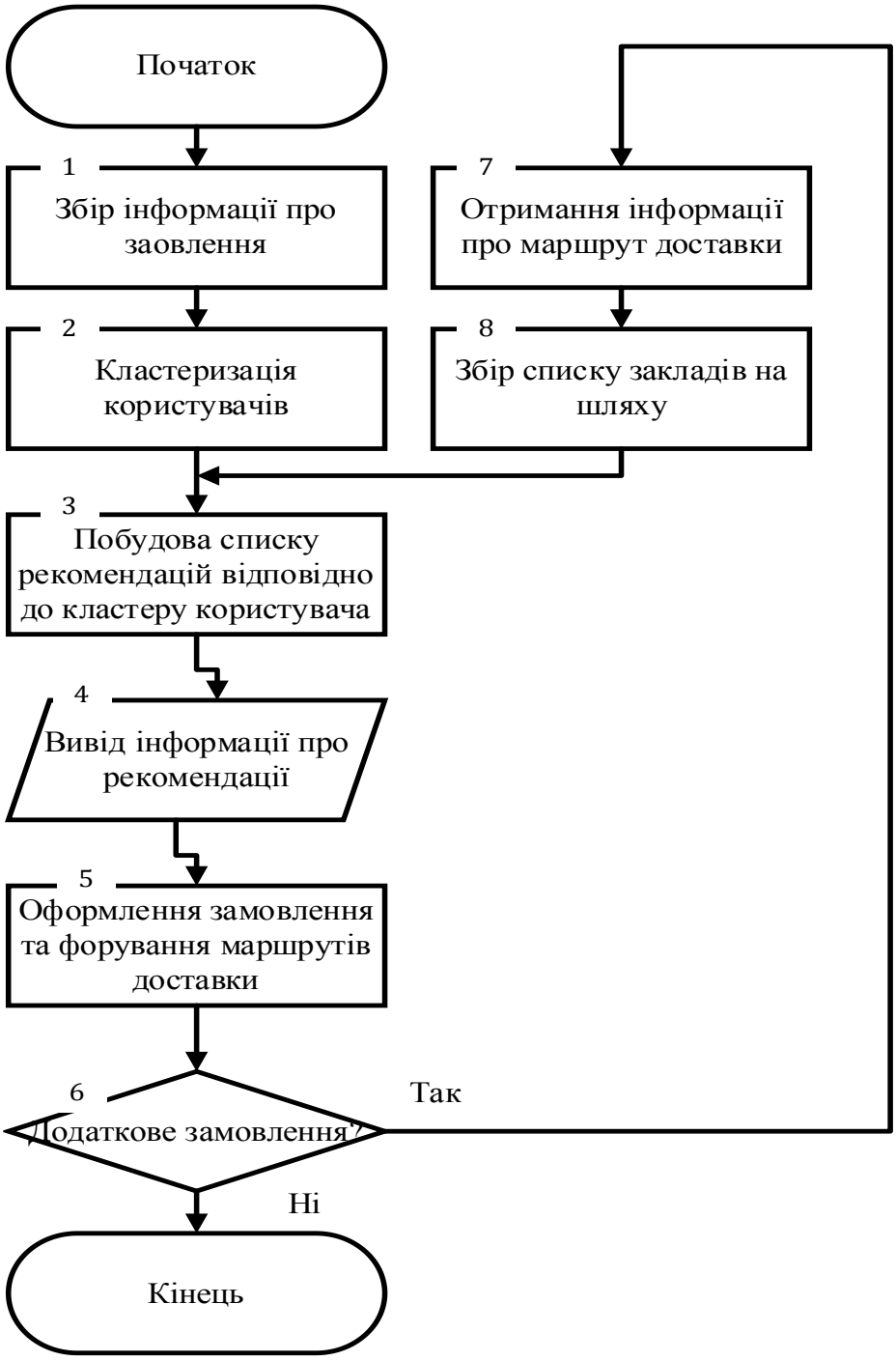


Рисунок В.3. – Блок-схема удосконаленого алгоритму рекомендацій

Додаток Г (довідковий)

Інструкція користувача

1. Завантажити архів OrderingSystem.zip, та розархівувати його.
2. Запустити серевр додаток який знаходиться в папці Server, виконавчим файлом OrderingServer.exe.
3. Запустити десктоп-додаток для менеджменту, у вікні входу ввести логін та пароль по замовчуванню логін :admin, пароль: adminpass.
4. Перейти в додатку для менджменту у налаштування та змінити пароль та логін доступу.
5. Встановити мобільний додаток на телефон, далі система повністю готова до використання .