

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СЕРВІСУ ДОСТАВКИ ЇЖИ
FOOD DELIVERY SERVICE SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ШЕПОТЬКО А.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОФМАН Є.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШЕПОТЬКА Андрія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення сервісу доставки їжі. Food Delivery Service Software

керівник проєкту (роботи) к.т.н., ГОФМАН Євгеній Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 04 ” квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГОФМАН Є.О., доцент		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “04” квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Андрій ШЕПОТЬКО
(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Євгеній ГОФМАН
(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
90 с., 9 табл., 38 рис., 4 дод., 15 джерел.

ANDROID, SQL, JAVA, ANDROID STUDIO, АЛГОРИТМ, ЇЖА,
ДОСТАВКА, МОБІЛЬНИЙ ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для сфери обслуговування.

Предмет дослідження – програмні засоби для підтримки сервісів доставки їжі.

Метою роботи є розробка програмного забезпечення сервісу доставки їжі.

Матеріали, методи та технічні засоби: мова програмування Java, розширювана мова розмітки XML, середовище розробки Android Studio, система керування базами даних SQLite.

Результати. Розроблено програмне забезпечення сервісу доставки їжі для платформи Android, що дозволяє забезпечити основні функціональні можливості відповідного сервісу.

Практична цінність роботи полягає у розробці програмного забезпечення сервісу доставки їжі для платформи Android.

Висновки. Виконано проєктування програмного забезпечення сервісу доставки їжі. Розроблено програмне забезпечення сервісу доставки їжі. Здійснено тестування розробленого програмного забезпечення сервісу доставки їжі.

Галузь використання – сервіси доставки їжі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 90 tables, 38 figures, 4 appendixes, 15 sources.

ANDROID, SQL, JAVA, ANDROID STUDIO, ALGORITHM, FOOD, DELIVERY, MOBILE APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of software development for the service sector.

The subject of the research is software to support food delivery services.

The purpose of this work is to develop software for food delivery service.

Materials, methods and technical tools: Java programming language, extensible XML markup language, Android Studio development environment, SQLite database management system.

Results. The software tools for the food delivery service based on the Android platform, which allows to provide the main functionality of the corresponding service have been developed.

The practical value of the work lies in the development of food delivery service software for the Android platform.

Conclusions. The software for the food delivery service has been designed. The software for the food delivery service has been developed. The developed software for the food delivery service has been tested.

Scope of use – food delivery services.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Поняття про програмне забезпечення для сервісів доставки їжі.....	10
1.2 Дослідження програмних засобів для підтримки процесів доставки їжі	13
1.3 Висновки за розділом 1	18
2 Матеріали і методи	19
2.1 Вибір мови програмування та інших технологій розробки	19
2.2 Вибір середовища розробки для створення програмного забезпечення	25
2.3 Висновки за розділом 2	28
3 Опис програми	29
3.1 Структурна схема програми сервісу доставки їжі	29
3.2 Опис основних розроблених модулів	30
3.3 Опис розроблених функцій програмного забезпечення	32
3.4 Функціональна схема програми	35
3.5 Проектування бази даних.....	36
3.6 Висновки за розділом 3	39
4 Експлуатація, тестування та експериментальне дослідження програми.....	40
4.1 Призначення й умови застосування програми	40
4.2 Характеристики програми	41
4.3 Інструкція по експлуатації програми.....	41
4.3.1 Звернення до програми	41
4.3.2 Вхідні й вихідні дані	42
4.3.3 Повідомлення.....	42
4.4 Виконання програмного забезпечення сервісу доставки їжі	43
4.5 Тестування програмного забезпечення сервісу доставки їжі	50
4.6 Висновки за розділом 4	51
Висновки	52

Перелік джерел посилань	54
Додаток А Технічне завдання	56
Додаток Б Опис програми	60
Додаток В Текст програми	64
Додаток Д Слайди презентації	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – прикладний програмний інтерфейс;

БД – база даних;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

ТЗ – технічне завдання.

ВСТУП

Їжа є однією з найважливіших фізіологічних потреб людини. Харчування є однією з основних умов існування людини. Їжа є джерелом енергії. Харчування впливає на здоров'я, настрій, працездатність, і тривалість життя. Для організму важливі кількість і якість їжі, вид споживаної їжі, час регулярних прийомів їжі [1]-[6].

Незалежно від того, працює організм, відпочиває чи спить, йому потрібно поповнити енергію. Своєчасне і нерегулярне вживання невідповідних інгредієнтів може призвести до порушення обміну речовин в організмі, що призведе до втоми, ожиріння і навіть до серйозних захворювань, які скорочують життя [1]-[5].

Якщо харчування є основою життя людини, то організація харчування є однією з культурних проблем людства. Послуги громадського харчування своєчасно вирішують деякі з цих питань, наприклад, універсальність. Людині потрібно тільки обрати необхідне блюдо, замовити і дочекатися доставки. У наш час існують багато мобільних застосунків сервісів доставки їжі, які суттєво спрощують процес доставки, оскільки сьогодні майже у кожного є смартфон [1]-[6].

У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення сервісу доставки їжі.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для сервісів доставки їжі;
- здійснити проектування програмного забезпечення сервісу доставки їжі;
- створити програмне забезпечення сервісу доставки їжі;
- виконати тестування розробленого програмного забезпечення сервісу доставки їжі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про програмне забезпечення для сервісів доставки їжі

В епоху доставки їжі онлайн-замовлення, доставка та винос є розумною стратегією для залучення широкого кола потенційних клієнтів. Коли ви починаєте доставляти їжу, важливо інвестувати в правильне програмне забезпечення для доставки їжі, щоб все було організовано та ефективно[1]-[6].

У найпростішій формі програмне забезпечення для управління доставкою їжі служить агрегатором сторонніх платформ доставки ресторану. Більш детальні центри управління доставкою «все-в-одному» також інтегруються із системами точок продажу, по суті створюючи автоматизацію доставки їжі та спрощуючи роботу виробників їжі [1]-[6].

Рішення для управління онлайн-замовленнями інтегрує POS-системи ресторанів у служби доставки, системи онлайн-замовлення, внутрішні програми харчування тощо із сторонніми рішеннями для онлайн-замовлення [1]-[6]. Іншими словами, програмне забезпечення долає розрив між POS-системою, з одного боку, і сторонньою платформою замовлень, з іншого. Завдяки інтеграції цих каналів розподілу в POS-систему процес онлайн-замовлення полегшується та автоматизується [1]-[6].

Програмне забезпечення для управління замовленнями онлайн може використовуватися незалежними та невеликими ресторанами, мережами ресторанів, кав'ярнями, ресторанами швидкого обслуговування та неформальними ресторанами – словом, будь-яким підприємством громадського харчування, яке пропонує доставку як послугу [1]-[6].

Інший тип компаній, які можуть отримати вигоду від цього типу проміжного програмного забезпечення для розповсюдження, - це темні кухні, також відомі як хмарні кухні, кухні-привиди або віртуальні ресторани. Ці кухні готують їжу лише на доставку (а іноді й на винос) [1]-[6].

Крім того, онлайн-центр розподілу їжі також може бути корисним для великих компаній з доставки, таких як супермаркети, компанії FMCG, а також бакалійні чи інші підприємства роздрібною торгівлі [1]-[6].

Найважливішим питанням є переваги, якими можуть користуватися підприємства доставки, використовуючи інтегроване рішення для доставки онлайн [1]-[6].

Ресторанам (та іншим підприємствам харчової промисловості), які використовують інтегроване рішення для онлайн-доставки, більше не потрібно вручну вводити замовлення на доставку в свою POS-систему, оскільки програмне забезпечення для доставки шлюзу автоматично надсилає всі онлайн-замовлення в POS. Це економить час ресторану на кожне замовлення, водночас уникаючи людської помилки під час копіювання квитків. Аналогічно, квитки замовлень автоматично друкуються на кухні, щоб додатково спростити потік замовлення [1]-[6].

Другою проблемою, яку вирішує інтеграція доставки у вашу торгову точку, є так звана «стіна для планшетів». Ресторани стикаються з кількома планшетами, з якими працювати з кількома постачальниками послуг доставки. Онлайн-система керування замовленнями дозволяє цим ресторанам використовувати лише один планшет, щоб отримати огляд усіх замовлень на доставку. Хоча всі планшети, надані сторонніми службами, мають бути відкритими, щоб ваш онлайн-ресторан працював, вам більше не потрібно звертати на них увагу [1]-[6].

Менеджер з доставки їжі також дозволяє ресторанам легко публікувати та редагувати меню на кількох платформах доставки та для кількох місць. Мережі ресторанів і темних кухонь із кількома локаціями та/або брендами можуть керувати своїми онлайн-меню в одному місці, знову ж таки, заощаджуючи багато часу. Інтеграція онлайн-замовлення повертає ресторанам контроль, дозволяючи їм самостійно створювати меню, не звертаючись через сторонню платформу доставки [1]-[6].

Ще одна велика перевага використання продовольчого центру – консолідована звітність. Коли ваші канали онлайн-замовлення їжі підключені до POS, POS надаватиме вам звіти про онлайн-продажі на додаток до внутрішніх даних про продажі. Ресторанам більше не потрібно отримувати дані з усіх різних платформ і об'єднувати їх вручну, що робить звітування швидшим і простішим [1]-[6].

Управління запасами також стає легшим: завдяки інтеграції з POS-доставкою, ваш інвентар автоматично оновлюється у вашій POS-системі після онлайн-замовлень на доставку. Через це товари, які немає в наявності, можна виявити і зробити непридатними для використання набагато швидше, не даючи онлайн-покупцям замовляти продукти, які немає в наявності [1]-[6].

Нарешті, проміжне програмне забезпечення для розповсюдження допомагає підприємствам гостинності виконувати податкові та інші правила. Інтеграція доставки реєструє всі онлайн-транзакції та замовлення у точці продажу, тому без будь-яких зусиль підприємство-виробник їжі буде дотримуватися закону [1]-[6].

Програмне забезпечення для доставки їжі або ресторанів — це комп'ютерна програма, яка надає власникам ресторанів і служб доставки використання вебінструментів для маркетингу та управління діяльністю свого ресторану. Ці операції включають замовлення їжі, варіанти доставки, обробку платежів, послуги постачальників, відносини з клієнтами та запаси. Програмне забезпечення для доставки в ресторан дозволяє надавати послуги доставки їжі на вимогу та відстежувати замовлення на доставку. Це допомагає ресторанам привернути увагу більшої кількості клієнтів і збільшити їхні продажі [1]-[6].

За останні роки доставка їжі стала одним з найбільш швидкозростаючих ринків. Послуги доставки їжі користуються великим попитом, і для ресторану з кількома локаціями програмне забезпечення служби доставки кількох ресторанів зможе приймати замовлення, які можна пересилати в найближче місце. Це програмне забезпечення для доставки онлайн-ресторанів дозволяє мережі ресторанів збільшувати виробництво та отримувати дохід, створюючи

організовану мережу для обслуговування ширшої території. Програмне забезпечення для доставки в кілька ресторанів дозволяє мережі ресторанів ефективно керувати інвентарем, меню, персоналом і вхідними замовленнями. Програмне забезпечення створює єдиний вебсайт ресторану та мобільний додаток для кількох ресторанів, заощаджуючи додаткові витрати на оновлення та рекламу. Це також гарантує, що створений імідж бренду залишається послідовним, щоб уникнути невідповідностей, які можуть виникнути під час обслуговування кількох веб-сайтів [1]-[6].

Таким чином, програмне забезпечення для доставки їжі або ресторанів — це комп'ютерна програма, яка надає власникам ресторанів, іншим виробникам готової їжі та службам доставки, використання вебінструментів для маркетингу та управління діяльністю свого підприємства. Таке програмне забезпечення дозволяє автоматизувати облік процесів замовлення їжі, варіанти доставки, обробку платежів, послуги постачальників, відносини з клієнтами та запаси. Воно допомагає ресторанам, іншим виробникам готової їжі та службам доставки залучити більшу кількість клієнтів та збільшити рівень продажів.

1.2 Дослідження програмних засобів для підтримки процесів доставки їжі

У наш час розроблено досить багато програм, що забезпечують підтримку процесів замовлення та доставки їжі. Виконаємо аналіз деяких з них.

Вебсервіс eda.ua – український онлайн-сервіс, в якому зібрані в єдину систему більш 900 ресторанів, кафе і магазинів. Сервіс надає можливість вибрати і замовити бажану їжу не блукаючи по безлічі сайтів закладів. Всі ціни на сервісі відповідають цінам в самих ресторанах [7].

Серед переваг даного сервісу можна виділити можливість обрати страви з різних ресторанів на свій смак, різноманітне меню, та достатній для комфортної роботи функціонал [7].

Серед недоліків виділяється погана персоналізація меню, що виражено у відсутності додавання улюблених страв, виставлення оцінок та написання коментарів [7].

На рис. 1.1 можна побачити користувацький інтерфейс вебзастосунок.

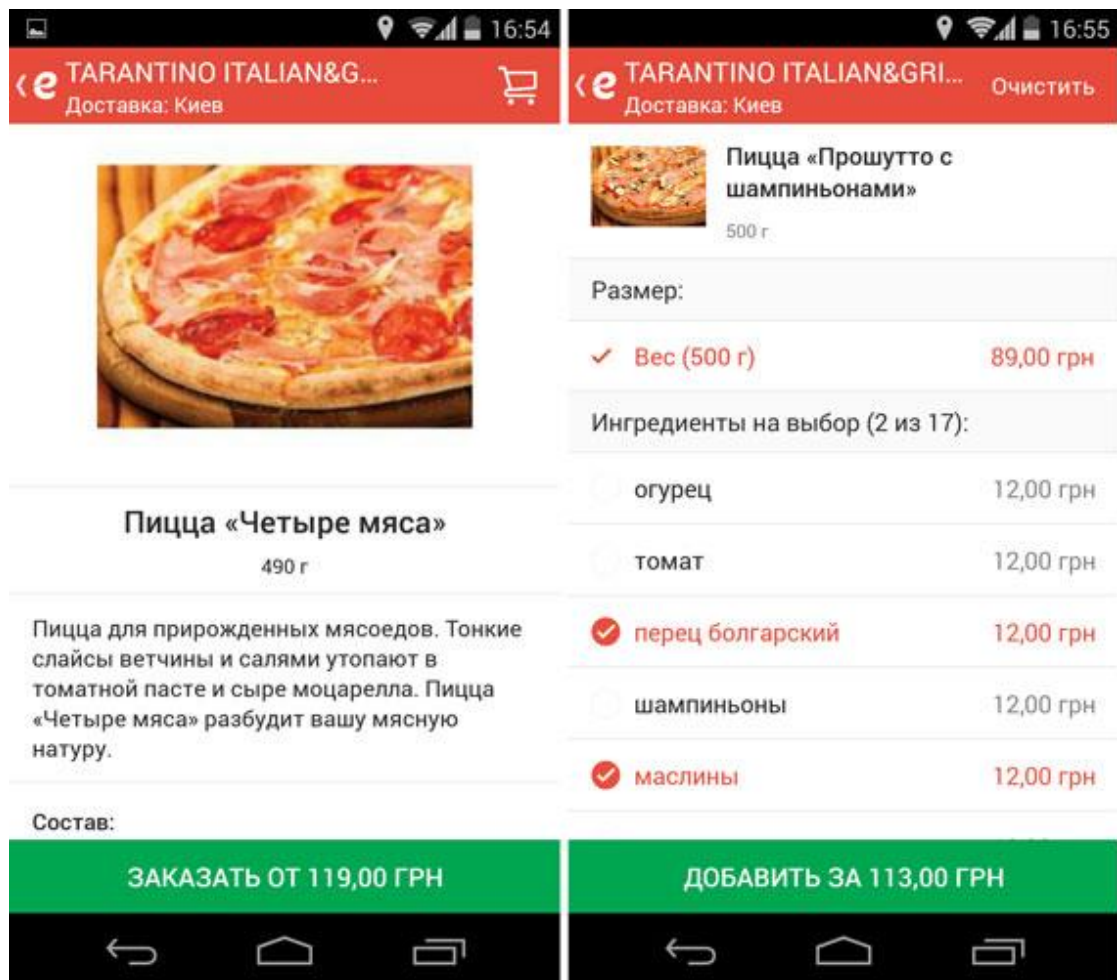


Рисунок 1.1 – Застосунок сервісу eda.ua [7]

Програмне забезпечення ekipazh-service – це зручний сервіс для швидкого і вигідного замовлення їжі і продуктів з ресторанів і магазинів. Послуги сайту абсолютно безкоштовні, а всі ціни ідентичні діючими цінами служб-партнерів. За допомогою Екіпаж-Сервіс можна отримати доступ до меню сотень ресторанів, а також можна ознайомитися з відгуками інших користувачів і спеціальними пропозиціями від закладів [8].

Перевагами цього сервісу є також великий вибір ресторанів, можливість доставки у декількох містах, багатий асортимент товарів [8].

Серед недоліків знову виділяється відсутність можливості у користувача висловити свою думку з тієї чи іншої страви, що може бути великим мінусом, так як багато людей спираються чи якимось чином орієнтуються на відгуки та оцінки інших користувачів [8].

На рис. 1.2 представлений застосунок даного сервісу [8].



Рисунок 1.2 – Вебзастосунок сервісу ekipazh-service [8]

Програмне забезпечення Red Box – у сервісі діє система оперативної роботи з клієнтами. Можна в будь-яку мить замовити їжу та отримати її протягом 90хв. абсолютно безкоштовно. Вигідні ціни на наші страви приємно здивують вас, а смак подарує гарний настрій на цілий день [9].

Перевагами цього сервісу може бути можливість пошуку користувачем страв, що значно поліпшує досвід роботи з програмою [9].

Недоліками є малий вибір страв та ресторанів, відсутність можливості висловити свою позицію з приводу якої-небудь страви [9].

На рис. 1.3 зображено додаток сервісу Red Box.



Рисунок 1.3 – Мобільний застосунок сервісу Red Box

Порівняльну характеристику програмного забезпечення сервісів доставки їжі наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення сервісів доставки їжі

Критерій порівняння	eda.ua	equipazh-service	Red Box
Реєстрація користувача	+	+	+
Різноманіття страв	+	+	—
Можливість залишити коментар	—	+	—
Можливість оцінити страву	—	—	—
Можливість додати страву до «улюблених»	—	—	—
Можливість вибору декількох ресторанів	+	+	—
Можливість пошуку страв	+	+	+

Як можна побачити серед недоліків усіх розглянутих аналогів виділяється відсутність можливості оцінити страву, додати її до улюблених та у деяких випадках залишити коментар. Тому для конкурентноспроможного вебзастосунку ці функції необхідно обов’язково реалізувати.

Таким чином, необхідно створити програмне забезпечення сервісу доставки їжі. При розробці програми необхідно забезпечити такі функціональні вимоги:

- можливість авторизації та реєстрації користувачів;
- додавання страв у корзину, видалення страв з корзини;
- оформлення замовлення;
- перегляд попередніх замовлень;
- перегляд меню;
- коментування страв;
- оцінювання страв;
- додавання страв до списку улюблених;

– виконання пошуку страв.

1.3 Висновки за розділом 1

Проаналізовано загальні поняття про онлайн-замовлення та сервіси доставки їжі. Визначено, що програмне забезпечення для доставки їжі – це комп’ютерна програма, яка надає власникам ресторанів, іншим виробникам готової їжі та службам доставки, використання вебінструментів для маркетингу та управління діяльністю свого підприємства. Таке програмне забезпечення дозволяє автоматизувати облік процесів замовлення їжі, варіанти доставки, обробку платежів, послуги постачальників, відносини з клієнтами та запаси. Воно допомагає ресторанам, іншим виробникам готової їжі та службам доставки залучити більшу кількість клієнтів та збільшити рівень продажів.

Визначено, що у наш час існує досить багато видів програмного забезпечення для підтримки процесів доставки їжі. Проте в існуючих програмах відсутні можливості оцінювання страв, додавання їх до улюблених та у деяких випадках залишення коментарів. Тому для конкурентноспроможного вебзастосунку ці функції необхідно обов’язково реалізувати.

Сформульовано функціональні вимоги до програмного забезпечення сервісу доставки їжі.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування та інших технологій розробки

При розробці програмного забезпечення сервісу доставки їжі були використані мова програмування Java, розширювані мови нотації XML і SQL.

Java — це об'єктно-орієнтована мова програмування, представлена Sun Microsystems у 1995 році як основний компонент платформи Java. Oracle, яка придбала Sun Microsystems того року, працює над мовою з 2009 року. Коли Java офіційно впроваджується, вона компілюється в байт-код, інтерпретований як віртуальна машина для даної платформи [10]-[11].

Oracle надає компілятори Java та віртуальні машини Java, які відповідають специфікаціям процесу спільноти Java під Загальною ліцензією GNU. Ця мова запозичила багато синтаксису з C і C++. Зокрема, об'єктна модель C++ була змінена. Це виключає можливість виникнення деяких конфліктних ситуацій, які можуть бути викликані помилками програміста, і спрощує процес розробки об'єктно-орієнтованої програми. Ряд операцій, які програмістам необхідно виконати на C/C++, довірено віртуальній машині. Оскільки Java була створена переважно як незалежна від платформи мова, вона має низькі апаратні можливості та уповільнює роботу програми порівняно, наприклад, із C++ [10]-[11].

Мова називалася Оак і була розроблена Джеймсом Гослінгом для програмування побутової електроніки. Пізніше він отримав назву Java і використовувався для написання клієнтських програм і серверного програмного забезпечення. Свою назву отримав від кавового бренду Java, який називають однойменним островом (Java), тому офіційним символом мови є чашка кави. Існує ще одна версія походження назви мови, пов'язана з кавоваркою, як приклад побутового приладу для програмування, який спочатку створив мову [10]-[11].

Основні цілі створення мови програмування Java такі [10]-[11]:

- синтаксис мови має бути простим, об'єктно-орієнтованим, знайомим;

- програма безпомилкова та безпечна;
- забезпечити архітектурну незалежність та мобільність;
- висока продуктивність;
- мова має бути описовою, багатопоточною, з динамічним зв'язком модулів [10]-[11].

На відміну від C++, Java більше об'єктно-орієнтована. Усі дані та дії розділені на категорії об'єктів. Винятки повної об'єктивності (наприклад, Smalltalk) є типами за замовчуванням (int, float тощо). Прискорити справу було свідомим рішенням мовних дизайнерів. Тому Java не вважається повноцінною об'єктно-орієнтованою мовою [10]-[11].

У Java всі об'єкти походять від батьківського об'єкта (просто званого Object) і успадковують від них базову поведінку та властивості [10]-[11].

Хоча в C++ можна вперше успадкувати багато, у Java можна успадкувати лише один, усуваючи конфлікти між членами класу (методами та змінними), успадкованими від базового класу [10]-[11].

Дизайнери розробили Java, щоб замінити C++, наступник об'єкта C. Дизайнери почали вивчати особливості C++, які є найпоширенішими помилками у створенні простої, безпечної та бездоганної мови програмування. У Java існує система винятків або випадків, коли програма стикається з несподіваними труднощами, такими як [10]-[11]:

- дії за межами елемента масиву або над порожнім елементом;
- читання з недоступних каталогів або неправильних URL-адрес;
- неправильне введення даних клієнта.

Однак не рекомендується використовувати мову програмування Java в системах, які можуть спричинити смерть, травми або серйозні травми (атомні електростанції, польоти, системи життєзабезпечення, програмне забезпечення для керування системами зброї тощо). Java використовує автоматичний збірник сміття для керування пам'яттю протягом усього життєвого циклу об'єкта. Програміст вирішує, коли створити об'єкт, а віртуальна машина відповідає за звільнення пам'яті, коли об'єкт більше не потрібен. Коли ви втрачаєте контакт

з певним об'єктом, збирач сміття може автоматично видалити його з пам'яті. Проте втрата пам'яті може статися, якщо код, написаний програмістом, посилається на непотрібні об'єкти, наприклад об'єкти, що зберігаються в існуючому контейнері [10]-[11].

Вивезення сміття дозволено завжди. В ідеалі це станеться, коли програма неактивна. Збір сміття виконується автоматично, коли в запасі немає вільної пам'яті для розміщення нових об'єктів, що призводить до зависання на кілька секунд. Тому існують реалізації віртуальних машин Java зі збирачами сміття, спеціально розробленими для програмування в реальному часі [10]-[11].

Java не підтримує покажчики C/C++. Це зроблено для забезпечення безпеки та безпеки сміттєзбірника, щоб дозволити йому переміщати вказівний об'єкт [10]-[11].

Обґрунтування вибору мови програмування для розробки програмного забезпечення сервісу доставки їжі наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення сервісу доставки їжі

Критерій порівняння мов програмування	Мова програмування		
	Java	C#	PHP
Зручність для створення ПЗ сервісу доставки їжі	+	—	—
Зручний синтаксис	+	+	+—
Висока продуктивність	+	+—	+—
Багатопоточність	+	+	+—
Архітектурна незалежність	+	—	+

Отже, для реалізації основного функціоналу програмного забезпечення сервісу доставки їжі обрано мову програмування Java.

При розробці програмного забезпечення сервісу доставки їжі обрано також використовувалася мова розширена мова розмітки (XML). Розширена мова розмітки (XML) — це стандарт, рекомендований Консорціумом World Wide Web (W3C) для створення ієрархічно структурованої мови розмітки для спільного використання між різними програмами, такими як Інтернет. SGML — це спрощена підмножина мови нотації. Документи XML складаються з текстових символів і читаються людиною [10]-[11].

Стандарт XML визначає основні лексичні та синтаксичні правила для створення інформативних описових мов за допомогою простих тегів. Цей формат є гнучким і підходить для різних галузей. Іншими словами, запропонований стандарт визначає металеву мову, на якій він визначає мову відображення конкретної тематично орієнтованої інформації шляхом обмеження структури та змісту документа [10]-[12].

Правильний документ відповідає всім правилам синтаксису XML. Недійсний документ не можна назвати документом XML. Відповідний аналітик не повинен розробляти такий документ. Особливо дійсними XML-документами є [10]-[12]:

- лише один елемент у корені;
- непорожні елементи позначаються тегами початку та кінця. Порожні елементи можна позначити як «закриті», наприклад <Назва тегу />;
- елемент не може мати більше одного атрибута з однаковою назвою. Значення атрибутів можуть бути в одинарних (') або подвійних (") дужках;
- теги можуть бути вкладеними, але не перекриватися. Кожен некореневий елемент повинен бути доповнений іншим елементом;
- документ повинен містити лише правильно закодовані та затверджені символи Unicode. Єдині кодування, які повинен розуміти процесор XML, це UTF-16 і UTF-8. Фактичне та заявлене кодування документа має збігатися. Під час шифрування через HTTP документ можна оголосити ззовні, використовуючи певну нотацію в заголовку «Content-Type» або на початку документа. У разі відсутності інформації про кодування документ повинен мати кодування UTF-8 (або його підпакет ASCII) [10]-[11].

XML-документ має ієрархічну логічну структуру і може бути представлений у вигляді дерева. Вузли цього дерева можуть бути такими. XML-документ повинен мати лише один базовий елемент. Інші елементи є субелементами цього базового елемента. У Android XML використовується для визначення різних структур даних, таких як шаблони, кольори та абзаци, з інтерфейсу користувача [10]-[12].

SQL — це мова програмування для взаємодії з базами даних користувачів і використовується для створення запитів, оновлення та керування базами даних зв'язку, створення та зміни схем баз даних, а також керування системами доступу до бази даних. Сам SQL не є ані системою керування базами даних, ані окремим програмним продуктом. SQL можна використовувати для створення інтерактивних запитів або як інструкцію з керування даними, якщо вона встановлена в програмі. Крім того, стандарт SQL включає функцію визначення, перевірки та захисту даних [13]-[14].

На додаток до управління базами даних, SQL є мовою програмування комунікацій для запитів і маніпулювання базами даних. Багато баз даних підтримують SQL зі стандартними мовними розширеннями. Ядро SQL створює мову сценаріїв, яка дозволяє шукати, вводити, оновлювати та видаляти дані за допомогою функцій керування та адміністрування [13]-[14].

Перша версія SQL була розроблена IBM на початку 1970-х років. Ця версія називається SEQUEL і призначена для обробки та відновлення даних, що містяться в реляційній базі даних IBM System R. SQL був додатково стандартизований Американським урядовим агентством стандартів (ANSI) у 1986 році. Спочатку SQL був задуманий як запит. мова управління даними. Пізніше постачальники управління базами даних зазнали змін у SQL, включаючи структуру процедур, команди керування потоком та спеціальні розширення мови. З виходом стандарту SQL: 1999 такі розширення були офіційно запозичені як частина мови SQL з використанням Persistent Stored Modules (SQL/PSM) [13]-[14].

Перевагами SQL є такі [13]-[14]:

– незалежність від певної бази даних. Хоча існують відмінності в тонах і синтаксисі, більшість текстів пошуку SQL, що містять DDL і DML, можна легко переміщати з однієї бази даних в іншу. Існують системи, де розробники зосереджуються на використанні принаймні кількох баз даних (наприклад, система електронного документообігу Documentum Oracle і Microsoft SQL Server можуть працювати з IBM DB2). Звичайно, цього рівня передачі дуже важко досягти, якщо у вас є певні можливості для практики [13]-[14];

– доступ до стандартів. Існування стандартів і наборів тестів для визначення сумісності певних реалізацій SQL із загальноприйнятими стандартами лише сприяє «стабілізації» мови. Однак слід зазначити, що сам стандарт дуже формальний і подекуди дуже великий, наприклад, основна частина стандарту SQL: 2003 містить понад 1300 сторінок тексту [13]-[14];

– декларація. За допомогою SQL програміст визначає лише ті дані, які потрібно витягти або змінити. При розробці SQL-запиту СУБД безпосередньо вирішує, як це зробити. Це, безумовно, не слід вважати універсальним принципом - програміст визначає набір даних для вибірки або модифікації, але база даних допомагає візуалізувати, як буде інтерпретований текст його запиту. Такі моменти особливо важливі при роботі з великими базами даних і складними запитами – чим складніший запит, тим більше варіантів. Ці параметри можуть значно відрізнятися за швидкістю продуктивності та використаними ресурсами, але результати (набори даних) мають бути однаковими [13]-[14].

До недоліків SQL можна віднести [13]-[14]:

– несумісність з моделлю даних зв'язку. Редактор моделі комунікаційних даних Едгар Кодд, Крістофер Дайт та їхні прихильники відзначають, що SQL не є справжньою мовою спілкування [13]-[14];

– відхилення від стандарту. Хоча ANSI є міжнародним стандартом (SQL-92, SQL: 1999 і пізніші), багато компаній, які розробляють СУБД (наприклад, Oracle, Sybase, Microsoft, MySQL), змінюють мову SQL, що використовується у своїх СУБД. Це створює передумову для відхилення від

стандарту. Тому в якості конкретної СУБД використовується певний діалект SQL [13]-[14];

- складність. Хоча спочатку SQL був розроблений як інструмент для кінцевого користувача, згодом він став дуже складним і став інструментом для програмістів [13]-[14];

- складність роботи з ієрархічною структурою. Раніше SQL не пропонував стандартного способу керування деревоподібними структурами. Деякі постачальники СУБД пропонують власні рішення. Наприклад, Oracle використовує оператор CONNECT BY. Сьогодні WITH приймає рекурсивні структури як стандарт [13]-[14].

2.2 Вибір середовища розробки для створення програмного забезпечення

В якості середовища розробки програмного забезпечення сервісу доставки їжі було обрано Android Studio [15].

Android Studio — це інтегрована платформа для розробки програм для платформи Android (IDE), представлена менеджером з продуктів Google Еллі Пейвз на конференції Google I/O 16 травня 2013 року. 8 грудня 2014 року компанія Google випустила першу стабільну версію Android Studio 1.0 [15].

Android Studio замінила плагін ADT на платформі Eclipse. Це середовище засноване на вихідному коді IntelliJ IDEA Community Edition, розробленому JetBrains. Android Studio була створена як частина моделі розробки з відкритим кодом і поширюється під ліцензією Apache 2.0 [15].

Бінарні версії доступні для Linux (використовується Ubuntu для тестування), Mac OS X і Windows. Media надає інструменти для розробки додатків не лише для смартфонів і планшетів, а й для носових пристроїв на базі Android Wear, TV (Android TV), Google Glass та автомобільної розважальної системи (Android Auto). Для програм, спочатку розроблених за допомогою

Eclipse та плагіна ADT, був розроблений інструмент для автоматичного імпорту існуючих проєктів у Android Studio [15].

Середовище розробки призначене для виконання звичних завдань, які вирішуються під час розробки додатків для платформи Android. До них належать інструменти спрощення програмного забезпечення для перевірки сумісності з різними версіями платформи та програмні засоби, які працюють на пристроях з різною роздільною здатністю (планшети, смартфони, ноутбуки, годинники, окуляри тощо). На додаток до функцій IntelliJ IDEA, Android Studio має ряд додаткових функцій, включаючи нову інтегровану підсистему для створення, тестування та розгортання програм, які підтримують використання інструментів збірки Gradle та інструментів безперервної інтеграції [15].

Для прискорення розробки додатків існує набір стандартних елементів інтерфейсу та візуальний редактор для їх організації, що дозволяє попередньо переглядати різні стани інтерфейсу програми (наприклад, можна побачити, як інтерфейс шукає різні версії Android і різні розміри екрана). Існує майстер створення власних елементів дизайну, який підтримує використання шаблонів для створення нестандартних інтерфейсів. Це середовище має вбудовані функції для завантаження звичайних зразків коду з GitHub [15].

Платформа Android також включає в себе розширені інструменти для вогнетривкості, сумісність з попередніми версіями, проблеми з продуктивністю, моніторинг використання пам'яті та оцінку використання. У редактор додано режим швидкого редагування. Опис, статичний аналіз і систему виявлення помилок розширено для підтримки Android API. Інтегрована підтримка оптимізації коду ProGuard. Вбудовані інструменти створення цифрового підпису. Він забезпечує інтерфейс для керування перекладом іншими мовами [15].

На даний момент середовище надає такі функції [15]:

– безпосередня організація (організація): пряме кодування – представлення програми в реальному часі;

- консоль розробника: поради щодо оптимізації, допомога в перекладі, відстеження, кампанії та реклама – показники Google Analytics;
- редизайн на базі Android і швидке виправлення;
- інструменти Lint, які охоплюють продуктивність, зручність використання, сумісність версій та інші питання;
- використання можливостей ProGuard та сигнатур програм;
- шаблони для створення поширених шаблонів та компонентів Android;
- розширений редактор діаграм, який дозволяє користувачам одночасно перетягувати компоненти інтерфейсу користувача та переглядати положення на різних налаштуваннях екрана [15].

Обґрунтування вибору середовища розробки для створення програмного забезпечення сервісу доставки їжі наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення сервісу доставки їжі

Критерій порівняння середовищ розробки	Середовище розробки		
	Android Studio	Atom	MS Visual Studio
Наявність шаблонів для створення поширених типів програм	+	+—	+
Зручність використання для створення програмного забезпечення сервісу доставки їжі	+	+—	+—
Зручність засобів для роботи з мовою Java	+	+—	+—
Консоль розробника	+	+—	+
Засоби Google Analytics	+	—	—

Таким чином, для створення програмного забезпечення сервісу доставки їжі обрано середовище розробки Android Studio.

2.3 Висновки за розділом 2

Для створення програмного забезпечення сервісу доставки їжі було обрано мову програмування Java, яка є досить простою та зручною у використанні, об'єктно-орієнтованою, а також забезпечує архітектурну незалежність та мобільність.

В якості середовища розробки програмного забезпечення сервісу доставки їжі було обрано Android Studio, що є інтегрованою платформою для розробки програм для платформи Android.

3 ОПИС ПРОГРАМИ

3.1 Структурна схема програми сервісу доставки їжі

У мобільному застосунку сервісу доставки їжі за допомогою окремих бібліотек `com.google.firebase:firebase-database:16.1.0` та `com.google.firebase:firebase-core:16.0.8` встановлюється зв'язок з базою даних Firebase Realtime Database, на сервері Firebase, для подальшої роботи з даними програмного забезпечення. Це необхідно для створення, редагування та видалення даних.

Програмне забезпечення сервісу доставки їжі складається з таких модулів:

- стартового екрану «MainActivity»;
- реєстрації «SignUp»;
- авторизації «SignIn»;
- меню «Home»;
- список їжі для пункту меню «FoodList»;
- страви детально «FoodDetail»;
- існуючих замовлень «OrderStatus»;
- корзини «Cart»;
- модуль для утримання елемента списків, що знаходяться у інших модулях «ViewHolder»;
- модуль моделі для утримання класів, що відповідають таблицям БД «Model»;
- модуль для збереження даних поточного користувача «Common»;
- модуль для роботи з локальною БД «Database».

Структурну схему програмного забезпечення сервісу доставки їжі наведено на рис. 3.1.

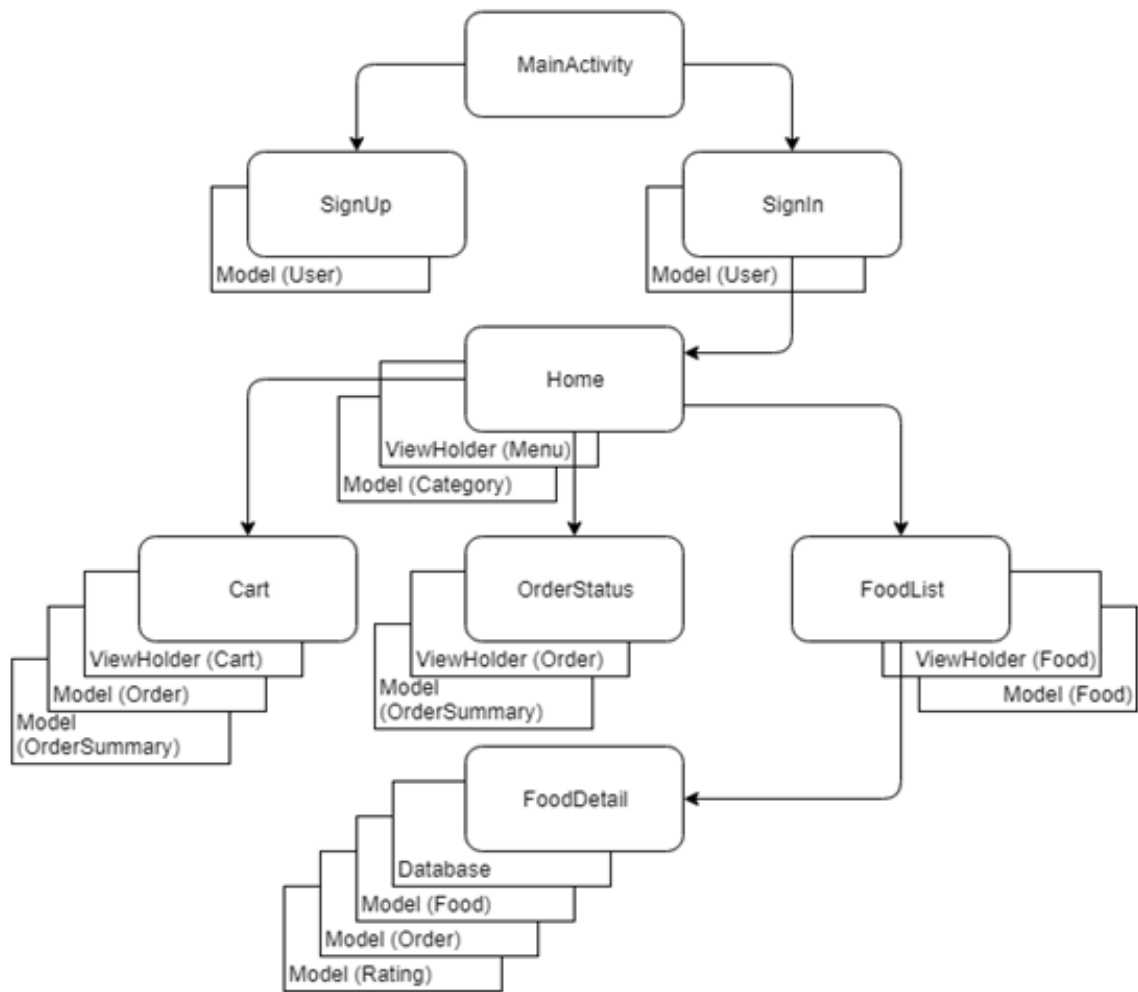


Рисунок 3.1 – Структурна схема програмного забезпечення
сервісу доставки їжі

3.2 Опис основних розроблених модулів

Усі головні модулі програмного забезпечення сервісу доставки їжі, що представлені на рис. 3.1, складаються з одного однойменного класу, який спадкується від класу AppCompatActivity. Останній клас є базовим для Activity (вікон програми), він забезпечує основний функціонал для вікон і всі Activity повинні бути його спадкоємцями.

Модуль стартового екрану «MainActivity» має дві кнопки для переходу до вікон реєстрації та авторизації, та відповідні функції переходу.

Модуль реєстрації «SignUp» містить вікно з деякими полями вводу тексту та кнопкою реєстрації, у якій є функція реєстрації (занесення у БД користувацьких даних).

Модуль авторизації «SignIn» містить поле логіна (номер телефона) та пароллю. Також кнопку та функцію входу у систему.

Модуль меню «Home» має список категорій їжі, натисненням на яку відкривається цей пункт та запускається модуль перегляду страв із конкретної категорії «FoodList», також є бокове меню з якого можна отримати доступ до корзини, попередніх замовлень, та виходу з облікового запису.

Модуль «FoodList» – це список їжі з пункту меню. Тут можна додати страву до улюблених натисненням на іконку серця, та відкрити її для детального перегляду.

Модуль «FoodDetail» призначений для перегляду конкретної страви детально, тут можна додати її у корзину, подивитись середню оцінку, виставити свою та залишити коментар.

У модулі «OrderStatus» користувач має можливість переглянути попередні замовлення, подивитись статус замовлення, ціну та іншу загальну інформацію.

Модуль «Cart» містить список товарів у корзині та кнопку підтвердження замовлення, також можливо видалити товари з неї.

Модуль «ViewHolder» складається з чотирьох класів «ViewHolderMenu», «ViewHolderCart», «ViewHolderOrder» та «ViewHolderFood». У кожному з них описано елемент списку та його поведінка.

Модуль «Model» було введено для зручності роботи з БД, що розташована на сервері, містить дев'ять класів, ідентичних до таблиць у БД.

З тих причин було створено модуль «Database», який забезпечує комфортну роботу з локальною БД, має методи роботи з нею.

Модуль «Common» введено задля швидкого та зручного доступу до даних поточного користувача застосунку.

3.3 Опис розроблених функцій програмного забезпечення

У таблиці 3.1 наведено розроблені функції (методи) в усіх класах програмного забезпечення сервісу доставки їжі, та описано їх призначення.

Таблиця 3.1 – Основні розроблені функції

Клас	Алгоритм	Призначення
1	2	3
Common	isConnectedToInternet	Перевірка підключення до інтернету
Database	getCart	Отримання замовлених поточним користувачем страв
	addToCart	Додавання страви у корзину
	cleanCart	Видалення усіх страв з корзини
	undoCart	Видалення останньої страви з корзини
	addToFavourites	Додання страви до улюблених
	removeFromFavourites	Видалення з улюблених
	isFavourite	Перевірка чи є страву улюбленою
Cart	showAlertDialog	Виведення на екран діалогу для завершення оформлення замовлення
	loadOrders	Завантаження на екран поточних доданих у корзину страв
	deleteCart	Видалення окремої страви з корзини
FoodDetail	getRating	Розрахування рейтингу страви
	showRatingDialog	Діалог для виставлення рейтингу та написання коментаря
	getFoodDetail	Завантаження детального опису страви
FoodList	startSearch	Пошук за назвою страви

Продовження таблиці 3.1

1	2	3
FoodList	loadSuggestions	Завантаження підказок для пошуку
	loadListFood	Завантаження списку страв для категорії меню
Home	loadMenu	Завантаження пунктів меню
	onNavigationItemSelected	Перехід до інших вікон програми
OrderStatus	loadOrders	Завантаження замовлень користувача
	getStatusFromCode	Визначення статусу замовлення по його коду
SignIn	btnSignIn.onClickListener	Авторизація у систему
SignUp	btnSignUp.onClickListener	Реєстрація нового користувача

На рис. 3.2 представлено діаграму станів програмного забезпечення сервісу доставки їжі. На рис. 3.3 наведено діаграму класів програмного забезпечення сервісу доставки їжі.

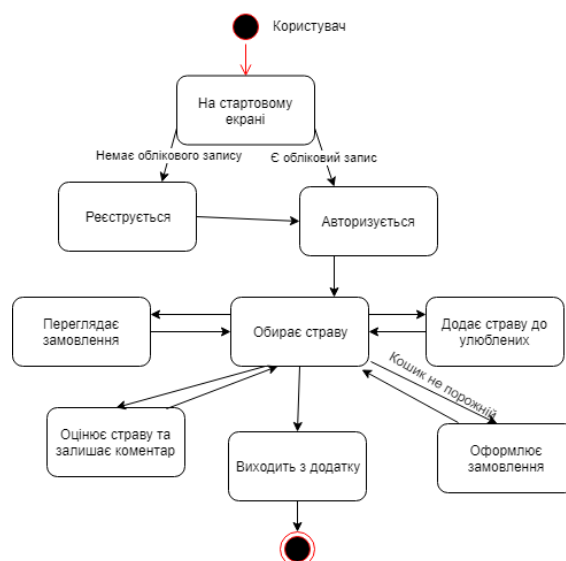


Рисунок 3.2 – Діаграма станів

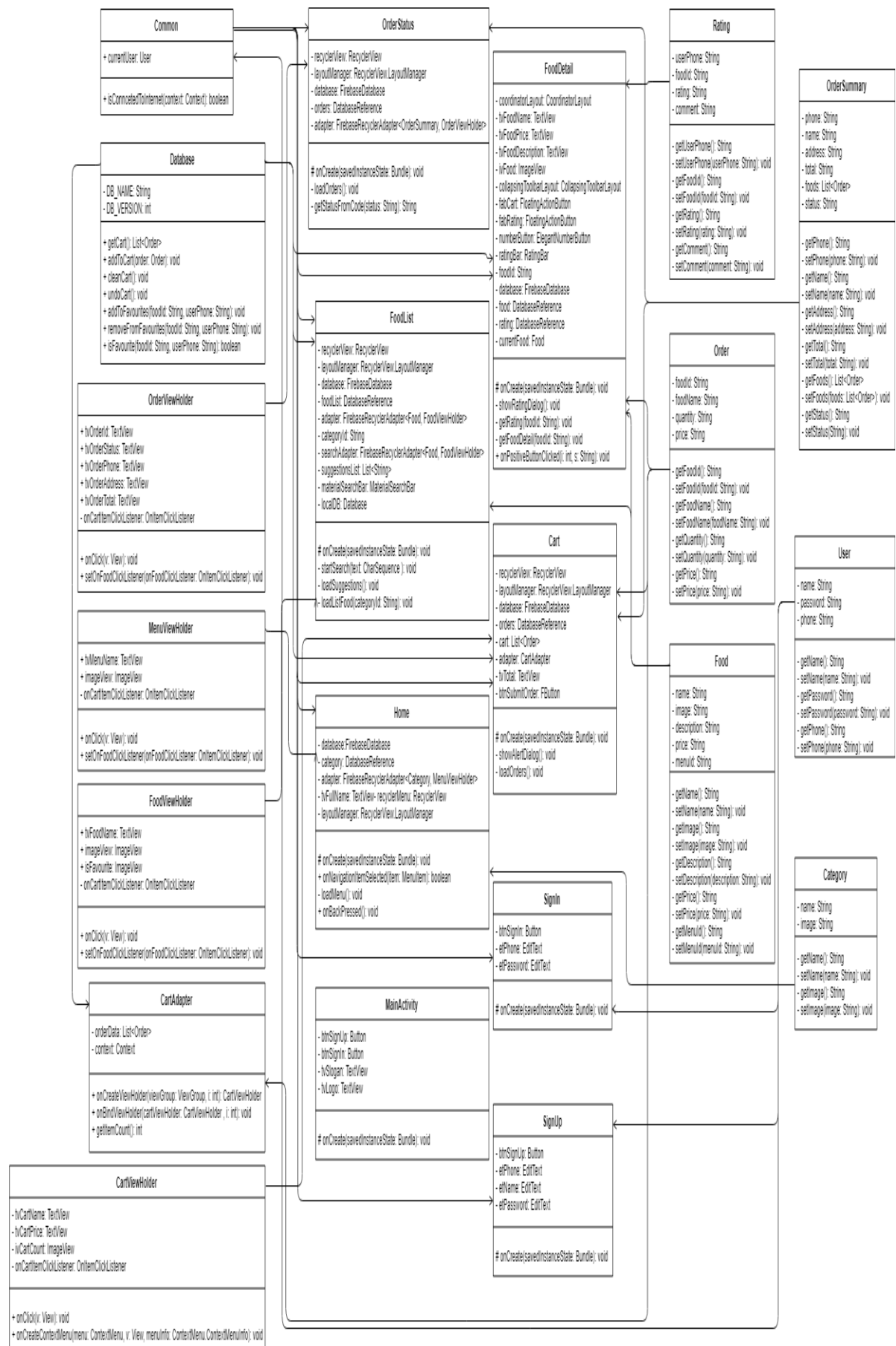


Рисунок 3.3 – Діаграма класів

На рис. 3.4 представлено діаграму діяльності під час користування програмним забезпеченням сервісу доставки їжі.

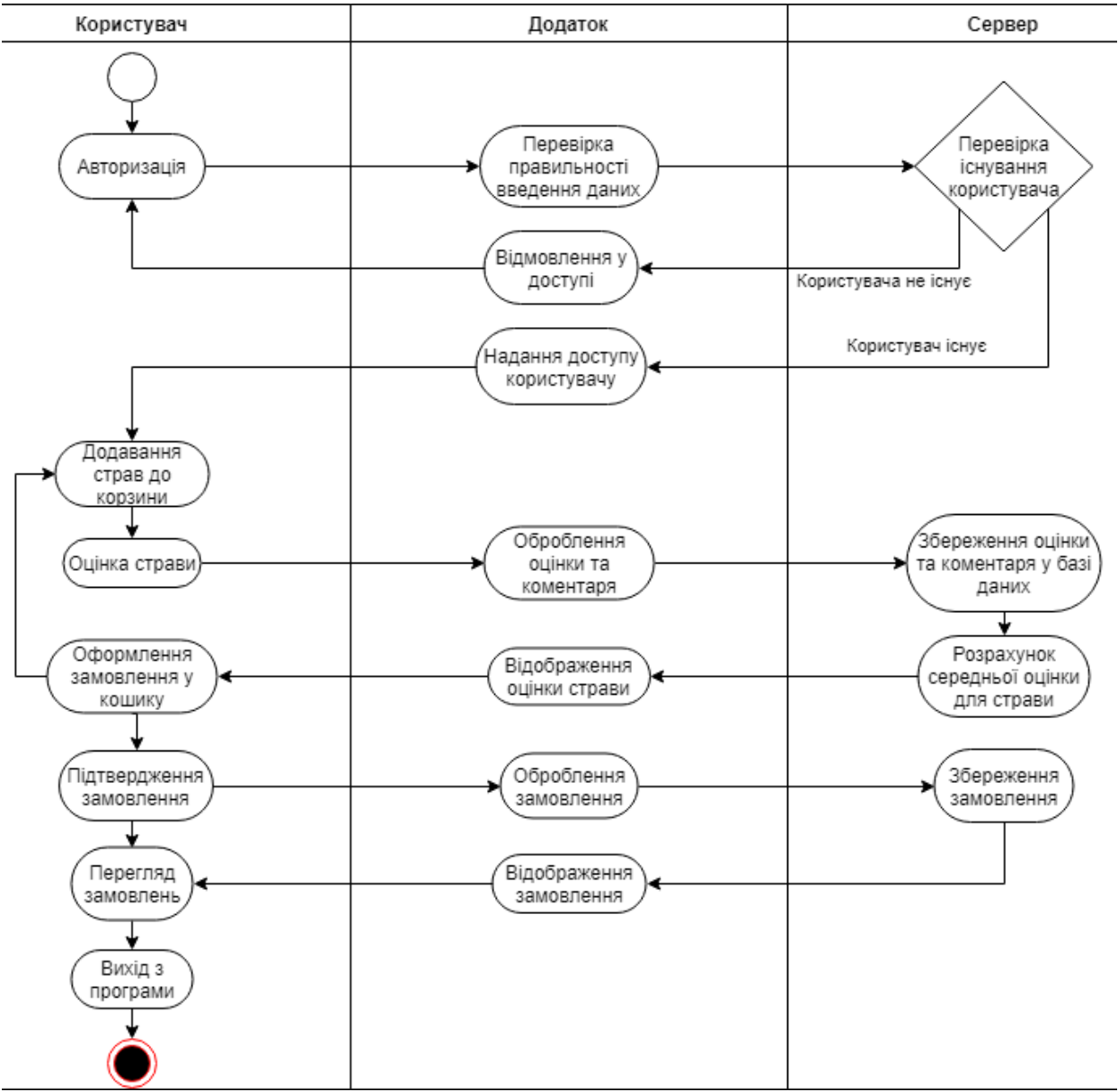


Рисунок 3.4 – Діаграма діяльності

3.4 Функціональна схема програми

Функціональну схему програмного забезпечення сервісу доставки їжі, що представлена на рис. 3.5.

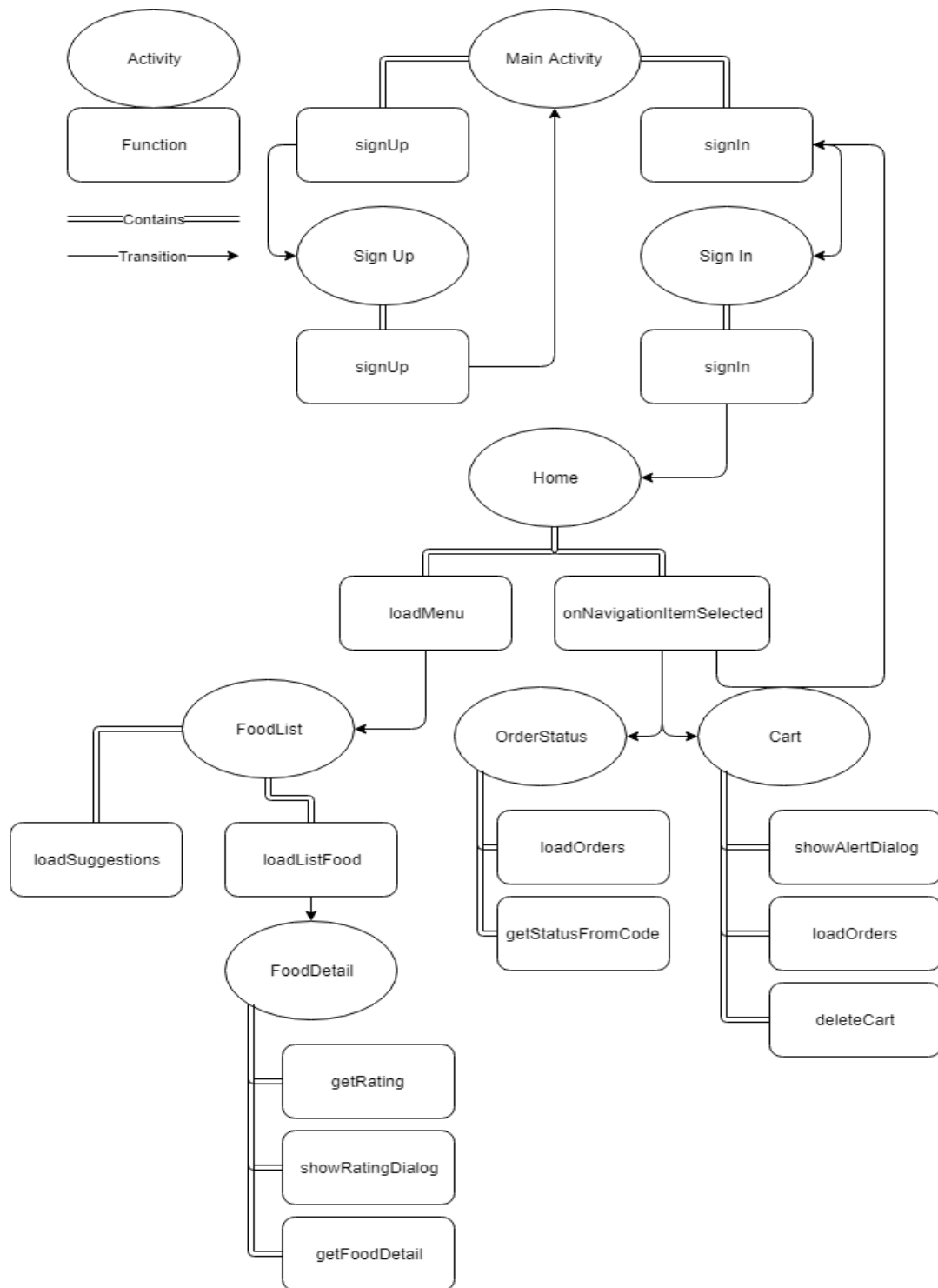


Рисунок 3.5 – Функціональна схема програми

3.5 Проєктування бази даних

На рис. 3.6 зображено концептуальну модель бази даних.

Сутність «Category» – необхідна для збереження даних про категорії їжі, опис полів наведено у таблиці 3.2.

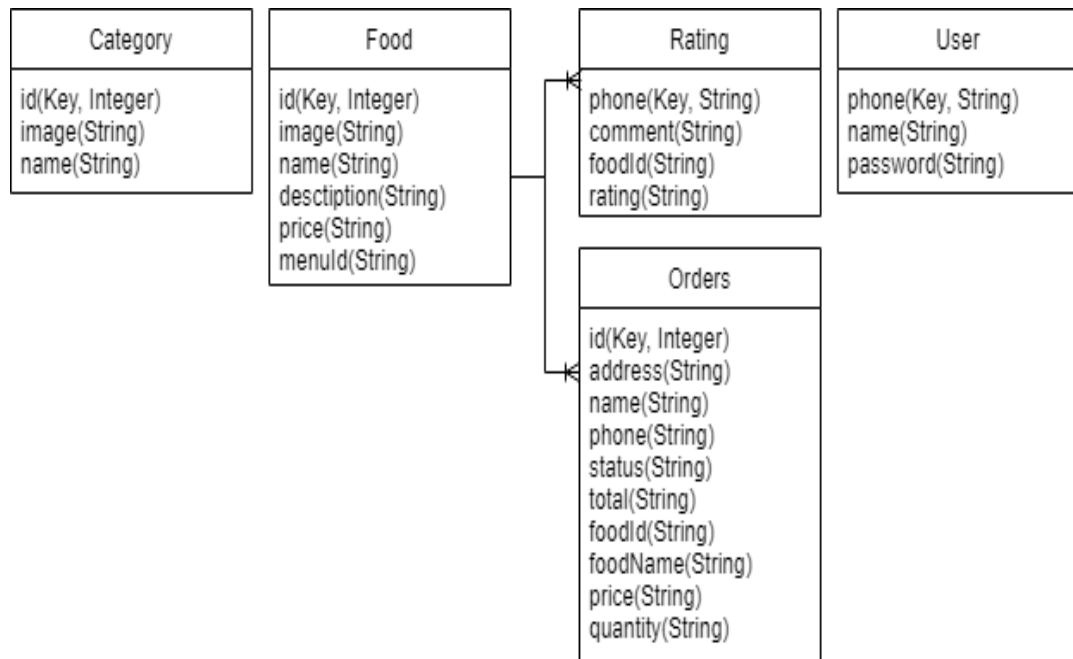


Рисунок 3.6 – Концептуальна модель бази даних

Таблиця 3.2 – Опис полів сутності «Category»

№	Назва	Тип	Ключ
1	id	Numeric	+
2	image	String	—
3	name	String	—

Сутність «Food» – необхідна для збереження даних про окремі страви, опис полів наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності «Food»

№	Назва	Тип	Ключ
1	id	Numeric	+
2	image	String	—
3	name	String	—
4	description	String	
5	price	String	

Сутність «Rating» – необхідна для збереження даних про оцінки та коментарі користувачів, опис полів наведено у таблиці 3.4.

Таблиця 3.4 – Опис полів сутності «Rating»

№	Назва	Тип	Ключ
1	phone	String	+
2	comment	String	–
3	foodId	String	–
4	rating	String	–

Сутність «User» – необхідна для збереження даних про користувачів, опис полів наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів сутності «User»

№	Назва	Тип	Ключ
1	phone	String	+
2	name	String	–
3	password	String	–

Сутність «Order» – необхідна для збереження даних про замовлення та керування ними, опис полів наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів сутності «Order»

№	Назва	Тип	Ключ
1	2	3	4
1	id	Numeric	+
2	address	String	–
3	name	String	–
4	phone	String	–

Продовження таблиці 3.6

1	2	3	4
5	status	String	—
6	total	String	—
7	foodId	String	—
8	foodName	String	—
9	price	String	—
10	quantity	String	—

3.6 Висновки за розділом 3

Розроблено програмне забезпечення сервісу доставки їжі. Запропоновано структуру та описано основні модулі програмного забезпечення сервісу доставки їжі. Визначено, що програмне забезпечення сервісу доставки їжі складається з таких модулів: модулю стартового екрану MainActivity, модулю реєстрації SignUp, модулю авторизації SignIn, модулю меню Home, модулю списку їжі для пункту меню FoodList, модулю страв FoodDetail, модулю існуючих замовлень OrderStatus, модулю корзини Cart, модулю для утримання елемента списків, що знаходяться у інших модулях ViewHolder, модулю моделі для утримання класів, що відповідають таблицям бази даних Model, модулю для збереження даних поточного користувача Common, модулю для роботи з локальною базою даних Database.

Описано розроблені функції програмного забезпечення сервісу доставки їжі. Розроблено функціональну схему програмного забезпечення сервісу доставки їжі.

Розроблено концептуальну модель бази даних програмного забезпечення сервісу доставки їжі. Визначено, що основними сутностями бази програмного забезпечення сервісу доставки їжі є: Category, Food, Rating, User, Order.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для надання доступу та базових функцій до сервісу доставки їжі. Програма написана на мові Java. База даних програмного забезпечення сервісу доставки їжі створена у системі керування базами даних SQLite.

Програмне забезпечення сервісу доставки їжі забезпечує виконання таких функцій:

- можливість авторизації та реєстрації користувачів;
- додавання страв у корзину, видалення страв з корзини;
- оформлення замовлення;
- перегляд попередніх замовлень;
- перегляд меню;
- коментування страв;
- оцінювання страв;
- додавання страв до списку улюблених;
- виконання пошуку страв.

Для експлуатації програмного забезпечення сервісу доставки їжі необхідно таке апаратне забезпечення:

- телефон на базі операційної системи Android;
- не менше 200 Мб вільного місця на жорсткому диску;
- процесор з тактовою частотою не менше 1 ГГц;
- оперативна пам'ять не менше 512 Мб;
- вільний доступ до мережі Internet.

Функціональні характеристики розробленого програмного забезпечення сервісу доставки їжі наведено у технічному завданні (додаток А). Опис програмного забезпечення сервісу доставки їжі та текст програмного забезпечення сервісу доставки їжі наведені у додатках Б та В.

4.2 Характеристики програми

Режим роботи програмного забезпечення сервісу доставки їжі є цілодобовим та безперервним. Програмне забезпечення сервісу доставки їжі використовує прийнятну кількість ресурсів пристрою.

Всі данні програмного забезпечення сервісу доставки їжі повинні зберігатися у розробленій базі даних.

Програмне забезпечення сервісу доставки їжі включає у себе такі модулі:

- стартового екрану «MainActivity»;
- реєстрації «SignUp»;
- авторизації «SignIn»;
- меню «Home»;
- список їжі для пункту меню «FoodList»;
- страви детально «FoodDetail»;
- існуючих замовлень «OrderStatus»;
- корзини «Cart»;
- модуль для утримання елементу списків, що знаходяться у інших модулях «ViewHolder»;
- модуль моделі для утримання класів, що відповідають таблицям БД «Model»;
- модуль для збереження даних поточного користувача «Common»;
- модуль для роботи з локальною БД «Database».

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Звертання до програми сервісу доставки їжі відбувається шляхом відкриття її натисненням іконки застосунку, попередньо встановивши

застосунок за допомогою файлу з розширенням «.apk», який необхідно завантажити окремо.

Доступ до серверної бази даних можна отримати на офіційному сайті Firebase, знаючи необхідний логін та пароль.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення сервісу доставки їжі є записи у серверній базі даних та у локальній базі даних.

Вихідними даними програмного забезпечення сервісу доставки їжі є вікна програми, що формуються безпосередньо за допомогою вхідних даних, з якими користувач взаємодіє та записи у відповідних базах даних.

4.3.3 Повідомлення

На екрані реєстрації користувач програмного забезпечення сервісу доставки їжі може отримати три повідомлення:

- про успішну реєстрацію;
- про те, що користувач програмного забезпечення сервісу доставки їжі з таким номером телефону вже зареєстрований;
- про відсутність інтернет-підключення у користувача програмного забезпечення сервісу доставки їжі.

На екрані авторизації користувач програмного забезпечення сервісу доставки їжі також може отримати три повідомлення:

- про те, що пароль введено невірно;
- про те, що користувача програмного забезпечення сервісу доставки їжі з таким номером телефону не існує;
- про відсутність інтернет-підключення у користувача програмного забезпечення сервісу доставки їжі.

На кожному подальшому екрані користувач програмного забезпечення сервісу доставки їжі так само може отримати повідомлення про відсутність у нього підключення до мережі Інтернет.

На екрані списку їжі може бути отримано такі повідомлення:

- про те, що страву додано до улюблених;
- про те, що страву видалено з улюблених.

На екрані корзини можна отримати повідомлення про те що вона порожня.

4.4 Виконання програмного забезпечення сервісу доставки їжі

Спочатку потрібно завантажити файл застосунку, встановити його на телефон та запустити натисненням на відповідну іконку. При запуску програми відкриється стартове вікно, зображене на рис. 4.1.



Рисунок 4.1 – Стартове вікно застосунку

Для реєстрації необхідно у відповідному вікні ввести всі необхідні дані (рис. 4.2) та натиснути «Зареєструватись».

A registration form is overlaid on a background image of various fruits and vegetables. The form contains three input fields with labels: 'Номер телефону' (Phone number), 'Ім'я' (Name), and 'Пароль' (Password). At the bottom of the form is a green button with the text 'ЗАРЕЄСТРУВАТИСЬ' (REGISTER).

Рисунок 4.2 – Вікно реєстрації

Для авторизації треба відкрити вікно авторизації, заповнити всі поля та натиснути «Увійти» (рис. 4.3).

An authorization form is overlaid on the same background image of various fruits and vegetables. The form contains two input fields with labels: 'Номер телефону' (Phone number) and 'Пароль' (Password). At the bottom of the form is a green button with the text 'УВІЙТИ' (LOGIN).

Рисунок 4.3 – Вікно авторизації

Після входу відкривається вікно «Меню» (рис. 4.4). У ньому можна відкрити бокове навігаційне меню свайпом зліва направо (рис. 4.5).



Рисунок 4.4 – Вікно меню

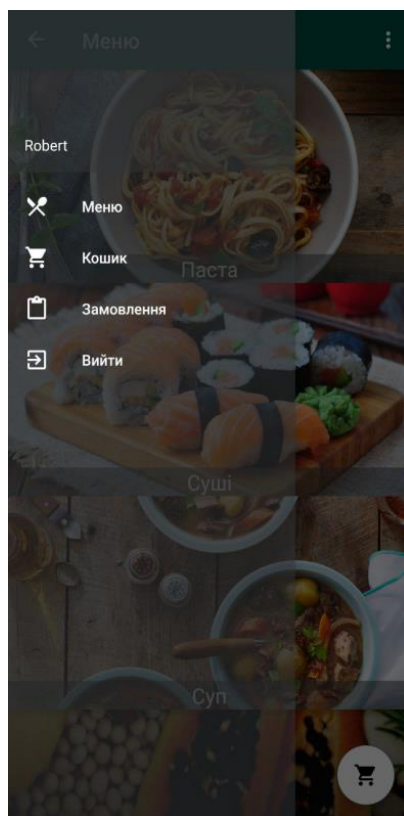


Рисунок 4.5 – Бокове навігаційне меню

Звідти користувач має можливість відкрити кошик (рис. 4.6), свої попередні замовлення (рис. 4.7), вийти з системи.



Рисунок 4.6 – Кошик



Рисунок 4.7 – Попередні замовлення

При натисканні на категорію меню можна відкрити список їжі для цієї категорії (рис. 4.8).

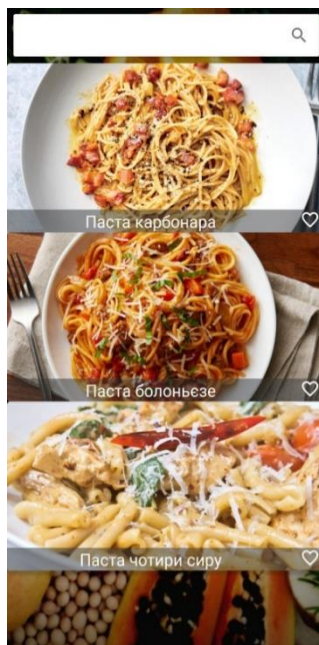


Рисунок 4.8 – Страви у категорії меню «Паста»

У вікні списку їжі можна додати страву до улюблених натисканням на значок серця, знайти необхідну страву за назвою у верхньому рядку пошуку, також можливо переглянути страву більш детально (рис. 4.9).

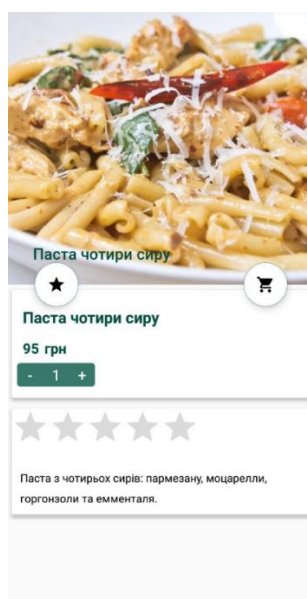


Рисунок 4.9 – Детальна інформація про страву

На вікні детального перегляду страви можна залишити оцінку (рис. 4.10), також додати її до кошику попередньо обравши бажану кількість. У спливаючому повідомленні можна відмінити додавання (рис. 4.11).

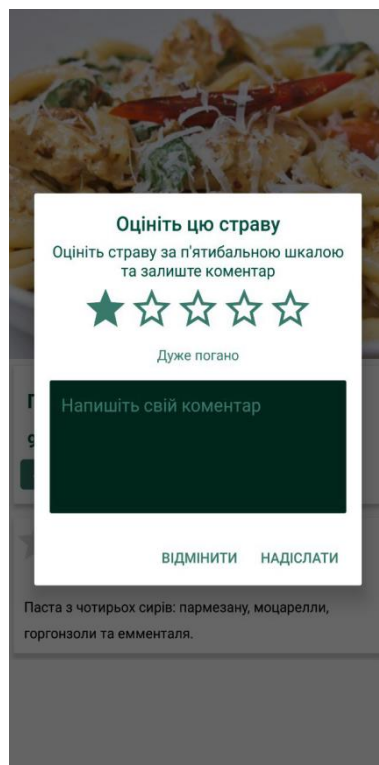


Рисунок 4.10 – Вікно оцінки страви

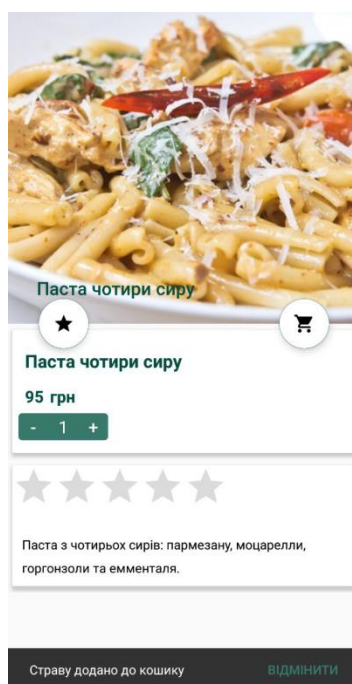


Рисунок 4.11 – Відмінити додавання до кошику

У вікні кошику можна видалити страву, для цього натиснути та утримати бажану страву, з контекстного меню обрати «Видалити» (рис. 4.12). Для підтвердження замовлення треба натиснути відповідну кнопку, з'явиться вікно для введення адреси (рис. 4.13), ввести її та підтвердити.



Рисунок 4.12 – Видалення страви з кошику

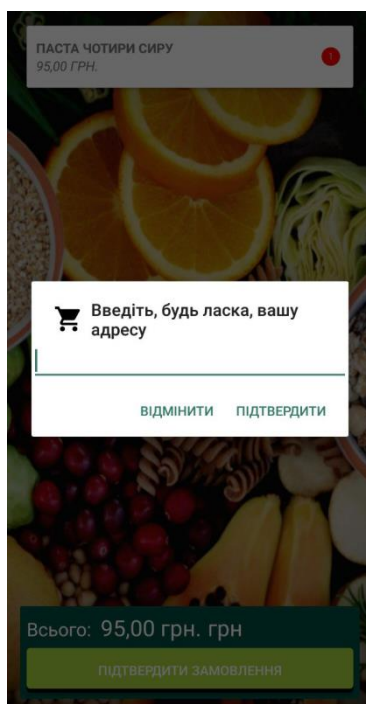


Рисунок 4.13 – Введення адреси та підтвердження замовлення

4.5 Тестування програмного забезпечення сервісу доставки їжі

Тестування було проведено на телефоні з операційною системою Android версії 8.0 Marshmallow, з оперативною пам'яттю 3Гб, процесором, що має 8 ядер із частотою 1.8ГГц.

Під час проведення тестування було перевірено такий функціонал:

- реєстрація;
- авторизація;
- вихід з облікового запису;
- додавання страви у кошик;
- видалення страви з кошику;
- оформлення замовлення;
- перегляд своїх попередніх замовлень;
- перегляд меню;
- написання коментаря для страви;
- оцінка страви;
- додавання страви до «улюблених»;
- видалення її звідти;
- виконання пошуку за назвою страв.

Тести приведених вище функцій проведені шляхом відтворення послідовності рекомендованих дій.

Тестування вважалось успішним, а застосунок готовим для роботи лише у тому разі коли всі функції та дії користувача були виконані правильно та не визивали помилок або збоїв у роботі мобільного додатку.

У результаті виконання тестування та використання мобільного застосунку було з'ясовано, що програмне забезпечення сервісу доставки їжі є готовим для роботи з користувачами. Всі функції, що було протестовано, показали гарний результат, повернувши необхідні дані і не викликавши виключень, що призвели б до негайного завершення програми.

Під час тестування у програмному забезпеченні сервісу доставки їжі не було виявлено помилок, збоїв або інших критичних недоліків які могли б вплинути на здатність програми виконувати її функції та працювати за призначенням.

4.6 Висновки за розділом 4

Описано програмне забезпечення сервісу доставки їжі. Виконано тестування розробленого програмного забезпечення сервісу доставки їжі. Результати тестування програмного забезпечення сервісу доставки їжі підтвердили його працездатність. У результаті виконання тестування та використання мобільного застосунку було з'ясовано, що програмне забезпечення сервісу доставки їжі є готовим для роботи з користувачами. Всі функції, що було протестовано, показали гарний результат, повернувши необхідні дані і не викликавши виключень, що призвели б до негайного завершення програми.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для сфери обслуговування.

Проаналізовано загальні поняття про онлайн-замовлення та сервіси доставки їжі. Визначено, що програмне забезпечення для доставки їжі – це комп'ютерна програма, яка надає власникам ресторанів, іншим виробникам готової їжі та службам доставки, використання вебінструментів для маркетингу та управління діяльністю свого підприємства. Таке програмне забезпечення дозволяє автоматизувати облік процесів замовлення їжі, варіанти доставки, обробку платежів, послуги постачальників, відносини з клієнтами та запаси. Воно допомагає ресторанам, іншим виробникам готової їжі та службам доставки залучити більшу кількість клієнтів та збільшити рівень продажів.

Визначено, що у наш час існує досить багато видів програмного забезпечення для підтримки процесів доставки їжі. Проте в існуючих програмах відсутні можливості оцінювання страв, додавання їх до улюблених та у деяких випадках залишення коментарів. Тому для конкурентноспроможного вебзастосунку ці функції необхідно обов'язково реалізувати.

Сформульовано функціональні вимоги до програмного забезпечення сервісу доставки їжі.

Для створення програмного забезпечення сервісу доставки їжі було обрано мову програмування Java, яка є досить простою та зручною у використанні, об'єктно-орієнтованою, а також забезпечує архітектурну незалежність та мобільність.

В якості середовища розробки програмного забезпечення сервісу доставки їжі було обрано Android Studio, що є інтегрованою платформою для розробки програм для платформи Android.

Розроблено програмне забезпечення сервісу доставки їжі. Запропоновано структуру та описано основні модулі програмного забезпечення

сервісу доставки їжі. Визначено, що програмне забезпечення сервісу доставки їжі складається з таких модулів: модулю стартового екрану MainActivity, модулю реєстрації SignUp, модулю авторизації SignIn, модулю меню Home, модулю списку їжі для пункту меню FoodList, модулю страв FoodDetail, модулю існуючих замовлень OrderStatus, модулю корзини Cart, модулю для утримання елемента списків, що знаходяться у інших модулях ViewHolder, модулю моделі для утримання класів, що відповідають таблицям бази даних Model, модулю для збереження даних поточного користувача Common, модулю для роботи з локальною базою даних Database.

Описано розроблені функції програмного забезпечення сервісу доставки їжі. Розроблено функціональну схему програмного забезпечення сервісу доставки їжі.

Розроблено концептуальну модель бази даних програмного забезпечення сервісу доставки їжі. Визначено, що основними сутностями бази програмного забезпечення сервісу доставки їжі є: Category, Food, Rating, User, Order.

Описано програмне забезпечення сервісу доставки їжі. Виконано тестування розробленого програмного забезпечення сервісу доставки їжі. Результати тестування програмного забезпечення сервісу доставки їжі підтвердили його працездатність. У результаті виконання тестування та використання мобільного застосунку було з'ясовано, що програмне забезпечення сервісу доставки їжі є готовим для роботи з користувачами. Всі функції, що було протестовано, показали гарний результат, повернувши необхідні дані і не викликавши виключень, що призвели б до негайного завершення програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Restaurant Delivery Software [Electronic resource]. – Access mode: <https://www.gloriafood.com/restaurant-delivery-software>.
2. Food Delivery Software Development [Electronic resource]. – Access mode: <https://binariks.com/industries/delivery/>.
3. Food Delivery Software [Electronic resource]. – Access mode: <https://www.getapp.com/retail-consumer-services-software/food-delivery/>.
4. What is food delivery management software? [Electronic resource]. – Access mode: <https://www.deliverect.com/us/blog/online-food-delivery/what-is-food-delivery-management-software>.
5. Online Food Delivery Software for your Restaurant [Electronic resource]. – Access mode: <https://www.softwaresuggest.com/food-delivery-software>.
6. Best Restaurant Delivery Software for Efficient Food Delivery [Electronic resource]. – Access mode: <https://www.getfareye.com/insights-blog/restaurant-delivery-software>.
7. Заовлення і доставка страв [Електрон. ресурс]. – Режим доступу: <https://eda.ua/>.
8. Екіпаж сервіс [Електрон. ресурс]. – Режим доступу: <https://ekipazh-service.com.ua/>.
9. Red Box, доставка їжі [Електрон. ресурс]. – Режим доступу: <https://top20.ua/lviv/restorani-kafe-bari/dostavka-pitsi-sushi-ta-obidiv/redbox-dostavka-yizhi.html>
10. C/C++ Pointers vs Java References [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/>.
11. Java Tutorial [Electronic resource]. – Access mode: <https://www.tutorialspoint.com/java/index.htm>.
12. Introduction to XML [Electronic resource]. – Access mode: https://www.w3schools.com/xml/xml_what.asp.

13. Introduction to SQL [Electronic resource]. – Access mode: https://www.w3schools.com/sql/sql_intro.asp.

14. SQL – Overview [Electronic resource]. – Access mode: <https://www.tutorialspoint.com/sql/sql-overview.htm>.

15. What language must one know to program in Android Studio [Electronic resource]. – Access mode: <https://www.quora.com/What-language-must-one-know-to-program-in-Android-Studio>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення сервісу доставки їжі може використовуватися для надання доступу та базових функцій до сервісу доставки їжі.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення сервісу доставки їжі», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 04 квітня 2023 р.

A.2 Призначення розробки

Програмний продукт сервісу доставки їжі призначений для вибору користувачем необхідних страв, та їх замовлення.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення сервісу доставки їжі забезпечує виконання таких функцій:

- можливість авторизації та реєстрації користувачів;
- додавання страв у корзину, видалення страв з корзини;
- оформлення замовлення;
- перегляд попередніх замовлень;
- перегляд меню;
- коментування страв;
- оцінювання страв;
- додавання страв до списку улюблених;
- виконання пошуку страв.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програми сервісу доставки їжі повинен бути зручним для користувачів. Взаємодія користувача з програмою сервісу доставки їжі можлива за допомогою користувальницьких форм.

A.3.3 Вимоги до надійності

Програмне забезпечення сервісу доставки їжі повинно передбачити відповідь на некоректність введеної користувачем інформації, забезпечити оброблення проблемних ситуацій, видавати відповідні повідомлення.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення сервісу доставки їжі необхідна наявність мобільного пристрою з параметрами, не гіршими від тих, які вказано у наступному пункті.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення сервісу доставки їжі необхідно таке апаратне забезпечення:

- телефон на базі операційної системи Android;
- не менше 200 Мб вільного місця на жорсткому диску;
- процесор з тактовою частотою не менше 1 ГГц;
- оперативна пам'ять не менше 512 Мб;
- вільний доступ до мережі Internet.

A.3.6 Вимоги до маркування і пакування

Програма сервісу доставки їжі може бути записана на будь-якому носії інформації. На пакуванні повинна бути назва програми – «Програмне забезпечення сервісу доставки їжі».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення сервісу доставки їжі є записи у серверній базі даних та у локальній базі даних.

Вихідними даними програмного забезпечення сервісу доставки їжі є вікна програми, що формуються безпосередньо за допомогою вхідних даних, з якими користувач взаємодіє та записи у відповідних базах даних.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програмне забезпечення сервісу доставки їжі розроблено за допомогою мови програмування Java та середовища розробки Android Studio. Для реалізації програмного забезпечення сервісу доставки їжі обрано систему керування базами даних SQLite.

Б.2 Функціональне призначення

Програма призначена для надання доступу та базових функцій до сервісу доставки їжі.

Програма сервісу доставки їжі забезпечує виконання таких функцій:

- можливість авторизації та реєстрації користувачів;
- додавання страв у корзину, видалення страв з корзини;
- оформлення замовлення;
- перегляд попередніх замовлень;
- перегляд меню;
- коментування страв;
- оцінювання страв;
- додавання страв до списку улюблених;
- виконання пошуку страв.

Б.3 Опис логічної структури

Програмне забезпечення сервісу доставки їжі включає у себе такі модулі:

- стартового екрану «MainActivity»;
- реєстрації «SignUp»;
- авторизації «SignIn»;
- меню «Home»;

- список їжі для пункту меню «FoodList»;
- страви детально «FoodDetail»;
- існуючих замовлень «OrderStatus»;
- корзини «Cart»;
- модуль для утримання елементу списків, що знаходяться у інших модулях «ViewHolder»;
- модуль моделі для утримання класів, що відповідають таблицям БД «Model»;
- модуль для збереження даних поточного користувача «Common»;
- модуль для роботи з локальною БД «Database».

Б.4 Використані технічні засоби

Для експлуатації програмного забезпечення сервісу доставки їжі необхідно таке апаратне забезпечення:

- телефон на базі операційної системи Android;
- не менше 200 Мб вільного місця на жорсткому диску;
- процесор з тактовою частотою не менше 1 ГГц;
- оперативна пам'ять не менше 512 Мб;
- вільний доступ до мережі Internet.

Б.5 Виклик і завантаження

Звертання до програми сервісу доставки їжі відбувається шляхом відкриття її натисненням іконки застосунку, попередньо встановивши застосунок за допомогою файлу з розширенням «.apk», який необхідно завантажити окремо.

Доступ до серверної бази даних можна отримати на офіційному сайті Firebase, знаючи необхідний логін та пароль.

Б.6 Вхідні та вихідні дані

Вхідними даними до програмного забезпечення сервісу доставки їжі є записи у серверній базі даних та у локальній базі даних.

Вихідними даними програмного забезпечення сервісу доставки їжі є вікна програми, що формуються безпосередньо за допомогою вхідних даних, з якими користувач взаємодіє та записи у відповідних базах даних.

ДОДАТОК В
Текст програми

```

package dntbother.foodservice5.Common;

public class Common {
    public static User currentUser;

    public static boolean isConnectedToInternet(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);

        if (connectivityManager != null) {
            NetworkInfo[] info =
connectivityManager.getAllNetworkInfo();
            if (info != null) {
                for (int i = 0; i < info.length; i++) {
                    if (info[i].getState() ==
NetworkInfo.State.CONNECTED) {
                        return true;
                    }
                }
            }
        }
        return false;
    }
}

package dntbother.foodservice5.Database;
import dntbother.foodservice5.Model.Order;

public class Database extends SQLiteAssetHelper {
    private static final String DB_NAME = "FoodService.db";
    private static final int DB_VERSION = 1;

    public Database(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    public List<Order> getCart() {
        SQLiteDatabase database = getReadableDatabase();
        SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();

        String[] sqlSelect = {"foodID", "foodName", "quantity",
"price"};
        String sqlTable = "OrderDetail";

        queryBuilder.setTables(sqlTable);
        Cursor cursor = queryBuilder.query(database, sqlSelect, null,

```

```

null, null, null, null);

    final List<Order> result = new ArrayList<>();
    if (cursor.moveToFirst()) {
        do {
            Order order = new
Order(cursor.getString(cursor.getColumnIndex("foodID")),

cursor.getString(cursor.getColumnIndex("foodName")),

cursor.getString(cursor.getColumnIndex("quantity")),

cursor.getString(cursor.getColumnIndex("price")));
            result.add(order);
        } while (cursor.moveToNext());
    }
    return result;
}

public void addToCart(Order order) {
    SQLiteDatabase database = getReadableDatabase();
    String query = String.format("INSERT INTO OrderDetail (foodID,
foodName, quantity, price) VALUES ('%s', '%s', '%s', '%s');",
        order.getFoodID(), order.getFoodName(),
order.getQuantity(), order.getPrice());
    database.execSQL(query);
}

public void cleanCart() {
    SQLiteDatabase database = getReadableDatabase();
    String query = String.format("DELETE FROM OrderDetail");
    database.execSQL(query);
}

public void undoCart(Order order) {
    SQLiteDatabase database = getReadableDatabase();
    String query = String.format("DELETE FROM OrderDetail WHERE ID
= (SELECT MAX(ID) FROM OrderDetail)");
    database.execSQL(query);
}

public void addToFavourites(String foodId, String userPhone) {
    SQLiteDatabase database = getReadableDatabase();
    String query = String.format("INSERT INTO Favourites
(foodID,userPhone) VALUES ('%s', '%s');", foodId, userPhone);

```

```

        database.execSQL(query);
    }

    public void removeFromFavourites(String foodId, String userPhone) {
        SQLiteDatabase database = getReadableDatabase();
        String query = String.format("DELETE FROM Favourites WHERE
foodId = '%s' AND userPhone = '%s'", foodId, userPhone);
        database.execSQL(query);
    }

    public boolean isFavourite(String foodId, String userPhone) {
        SQLiteDatabase database = getReadableDatabase();
        String query = String.format("SELECT * FROM Favourites WHERE
foodId = '%s' AND userPhone = '%s'", foodId, userPhone);
        Cursor cursor = database.rawQuery(query, null);
        if (cursor.getCount() <= 0) {
            cursor.close();
            return false;
        } else {
            cursor.close();
            return true;
        }
    }
}

package dntbother.foodservice5.Interface;
public interface OnItemClickListener {
void onClick(View view, int position, boolean isLongClick);
}

package dntbother.foodservice5.Model;

public class Category {
    private String name;
    private String image;

    public Category() {
    }

    public Category(String name, String image) {
        this.name = name;
        this.image = image;
    }

    public String getName() {
        return name;
    }
}

```

```

    }

    public void setName(String name) {
        this.name = name;
    }

    public String getImage() {
        return image;
    }

    public void setImage(String image) {
        this.image = image;
    }
}

package dntbother.foodservice5.Model;

public class Food {
    private String name;
    private String image;
    private String description;
    private String price;
    private String menuId;

    public Food() {
    }

    public Food(String name, String image, String description, String
price, String menuId) {
        this.name = name;
        this.image = image;
        this.description = description;
        this.price = price;
        this.menuId = menuId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getImage() {
        return image;
    }
}

```

```

    }

    public void setImage(String image) {
        this.image = image;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }

    public String getMenuId() {
        return menuId;
    }

    public void setMenuId(String menuId) {
        this.menuId = menuId;
    }
}

package dntbother.foodservice5.Model;

public class Order {
    private String foodID;
    private String foodName;
    private String quantity;
    private String price;

    public Order() {
    }

    public Order(String foodID, String foodName, String quantity,
String price) {
        this.foodID = foodID;
        this.foodName = foodName;

```

```

        this.quantity = quantity;
        this.price = price;
    }

    public String getFoodID() {
        return foodID;
    }

    public void setFoodID(String foodID) {
        this.foodID = foodID;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public String getQuantity() {
        return quantity;
    }

    public void setQuantity(String quantity) {
        this.quantity = quantity;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }
}

package dntbother.foodservice5.Model;

import java.util.List;

public class OrderSummary {
    private String phone;
    private String name;
    private String address;
    private String total;

```

```
private String status;
private List<Order> foods;

public OrderSummary() {
}

public OrderSummary(String phone, String name, String address,
String total, List<Order> foods) {
    this.phone = phone;
    this.name = name;
    this.address = address;
    this.total = total;
    this.foods = foods;
    this.status = "0";
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
```

```

        this.address = address;
    }

    public String getTotal() {
        return total;
    }

    public void setTotal(String total) {
        this.total = total;
    }

    public List<Order> getFoods() {
        return foods;
    }

    public void setFoods(List<Order> foods) {
        this.foods = foods;
    }
}

package dntbother.foodservice5.Model;

public class Rating {
    private String userPhone;
    private String foodId;
    private String rating;
    private String comment;

    public Rating() {
    }

    public Rating(String userPhone, String foodId, String rating,
String comment) {
        this.userPhone = userPhone;
        this.foodId = foodId;
        this.rating = rating;
        this.comment = comment;
    }

    public String getUserPhone() {
        return userPhone;
    }

    public void setUserPhone(String userPhone) {
        this.userPhone = userPhone;
    }
}

```

```

    public String getFoodId() {
        return foodId;
    }

    public void setFoodId(String foodId) {
        this.foodId = foodId;
    }

    public String getRating() {
        return rating;
    }

    public void setRating(String rating) {
        this.rating = rating;
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }
}

package dntbother.foodservice5.Model;

public class User {
    private String name;
    private String password;
    private String phone;

    public User() {
    }

    public User(String name, String password) {
        this.name = name;
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {

```

```

        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }
}

package dntbother.foodservice5.ViewHolder;

import dntbother.foodservice5.Database.Database;
import dntbother.foodservice5.Interface.OnItemClickListener;
import dntbother.foodservice5.Model.Order;
import dntbother.foodservice5.R;

class CartViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener, View.OnCreateContextMenuListener {

    TextView tvCartName, tvCartPrice;
    ImageView ivCartCount;

    private OnItemClickListener onCartItemClickListener;

    CartViewHolder(@NonNull View itemView) {
        super(itemView);

        tvCartName = itemView.findViewById(R.id.tv_cartItemName);
        tvCartPrice = itemView.findViewById(R.id.tv_cartItemPrice);
        ivCartCount = itemView.findViewById(R.id.iv_cartItemCount);

        itemView.setOnClickListener(this);
        itemView.setOnCreateContextMenuListener(this);
    }
}

```

```

@Override
public void onClick(View v) {
    onCartItemClickListener.onClick(v, getAdapterPosition(),
false);
}

void setOnCartItemClickListener(OnItemClickListener
onCartItemClickListener) {
    this.onCartItemClickListener = onCartItemClickListener;
}

@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
    menu.setHeaderTitle("Виберіть, що ви хочете зробити");
    menu.add(0, 0, getAdapterPosition(), "Видалити");
}
}

public class CartAdapter extends RecyclerView.Adapter<CartViewHolder> {

    private List<Order> orderData;
    private Context context;

    public CartAdapter(List<Order> orderData, Context context) {
        this.orderData = orderData;
        this.context = context;
    }

    @NonNull
    @Override
    public CartViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int i) {
        LayoutInflater inflater = LayoutInflater.from(context);
        View itemView = inflater.inflate(R.layout.cart_item, viewGroup,
false);
        return new CartViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(@NonNull CartViewHolder
cartViewHolder, int i) {
        TextDrawable drawable = TextDrawable.builder()
            .buildRound("" + orderData.get(i).getQuantity(),
Color.RED);

```

```

        cartViewHolder.ivCartCount.setImageDrawable(drawable);

        Locale locale = new Locale("ua", "UA");
        NumberFormat format = NumberFormat.getCurrencyInstance(locale);
        int price = (Integer.parseInt(orderData.get(i).getPrice()))
            * (Integer.parseInt(orderData.get(i).getQuantity()));
        cartViewHolder.tvCartPrice.setText(format.format(price));

    cartViewHolder.tvCartName.setText(orderData.get(i).getFoodName());
    }

    @Override
    public int getItemCount() {
        return orderData.size();
    }
}

package dntbother.foodservice5.ViewHolder;

import dntbother.foodservice5.Interface.OnItemClickListener;
import dntbother.foodservice5.R;

public class FoodViewHolder extends RecyclerView.ViewHolder implements
View.OnItemClickListener {

    public TextView tvFoodName;
    public ImageView imageView;
    public ImageView ivFavourite;

    private OnItemClickListener onFoodClickListener;

    public FoodViewHolder(@NonNull View itemView) {
        super(itemView);

        tvFoodName = itemView.findViewById(R.id.tv_foodName);
        imageView = itemView.findViewById(R.id.iv_food);
        ivFavourite = itemView.findViewById(R.id.iv_favourite);

        itemView.setOnClickListener(this);
    }

    public void setOnFoodClickListener(OnItemClickListener
onFoodClickListener) {
        this.onFoodClickListener = onFoodClickListener;
    }
}

```

```

@Override
public void onClick(View v) {
    onFoodClickListener.onClick(v, getAdapterPosition(), false);
}
}

package dntbother.foodservice5.ViewHolder;

import dntbother.foodservice5.Interface.OnItemClickListener;
import dntbother.foodservice5.R;

public class MenuViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    public TextView tvMenuName;
    public ImageView imageView;

    private OnItemClickListener onItemClickListener;

    public MenuViewHolder(@NonNull View itemView) {
        super(itemView);

        tvMenuName = itemView.findViewById(R.id.tv_menuName);
        imageView = itemView.findViewById(R.id.iv_menu);

        itemView.setOnClickListener(this);
    }

    public void setOnItemClickListener(OnItemClickListener
onItemClickListener) {
        this.onItemClickListener = onItemClickListener;
    }

    @Override
    public void onClick(View v) {
        onItemClickListener.onClick(v, getAdapterPosition(), false);
    }
}

package dntbother.foodservice5.ViewHolder;

import dntbother.foodservice5.Interface.OnItemClickListener;
import dntbother.foodservice5.R;

public class OrderViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

```

```

        public TextView tvOrderId, tvOrderStatus, tvOrderPhone,
tvOrderAddress, tvOrderTotal;

        private OnItemClickListener onOrderItemClickListener;

        public OrderViewHolder(@NonNull View itemView) {
            super(itemView);

            tvOrderId = itemView.findViewById(R.id.tv_orderItemName);
            tvOrderStatus = itemView.findViewById(R.id.tv_orderItemStatus);
            tvOrderPhone = itemView.findViewById(R.id.tv_orderItemPhone);
            tvOrderAddress =
itemView.findViewById(R.id.tv_orderItemAddress);
            tvOrderTotal = itemView.findViewById(R.id.tv_orderItemPrice);

            itemView.setOnClickListener(this);
        }

        public void setOnOrderItemClickListener(OnItemClickListener
onOrderItemClickListener) {
            this.onOrderItemClickListener = onOrderItemClickListener;
        }

        @Override
        public void onClick(View v) {
            onOrderItemClickListener.onClick(v, getAdapterPosition(),
false);
        }
    }

    import dntbother.foodservice5.Common.Common;
import dntbother.foodservice5.Database.Database;
import dntbother.foodservice5.Model.Order;
import dntbother.foodservice5.Model.OrderSummary;
import dntbother.foodservice5.ViewHolder.CartAdapter;
import info.hoang8f.widget.FButton;

    public class Cart extends AppCompatActivity {

        RecyclerView recyclerView;
        RecyclerView.LayoutManager layoutManager;

        FirebaseDatabase database;
        DatabaseReference orders;

```

```

List<Order> cart = new ArrayList<>();
CartAdapter adapter;

TextView tvTotal;
FButton btnSubmitOrder;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cart);

    //Init Firebase
    database = FirebaseDatabase.getInstance();
    orders = database.getReference().child("Orders");

    //Init Views
    tvTotal = findViewById(R.id.tv_total);
    btnSubmitOrder = findViewById(R.id.btn_submitOrder);
    recyclerView = findViewById(R.id.recycler_cart);

    layoutManager = new LinearLayoutManager(this);
    recyclerView.setLayoutManager(layoutManager);

    btnSubmitOrder.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (cart.size() > 0) {
                showAlertDialog();
            } else {
                Toast.makeText(Cart.this, "Корзина порожня",
Toast.LENGTH_SHORT).show();
            }
        }
    });

    loadOrders();
}

private void showAlertDialog() {
    AlertDialog.Builder alertDialog = new
AlertDialog.Builder(Cart.this);
    alertDialog.setTitle("Введіть, будь ласка, вашу адресу");

    final EditText etAddress = new EditText(Cart.this);
    LinearLayout.LayoutParams layoutParams = new

```

```

LinearLayout.LayoutParams (LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.MATCH_PARENT);
    etAddress.setLayoutParams (layoutParams);
    alertDialog.setView (etAddress);
    alertDialog.setIcon (R.drawable.ic_shopping_cart_black_24dp);

    alertDialog.setPositiveButton ("Підтвердити", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            OrderSummary orderSummary = new OrderSummary(
                Common.currentUser.getPhone(),
Common.currentUser.getName(),
                etAddress.getText().toString(),
tvTotal.getText().toString(), cart);

orders.child(String.valueOf(System.currentTimeMillis())).setValue(order
Summary);

            new Database(getBaseContext()).cleanCart();
            Toast.makeText(Cart.this, "Дякуємо, замовлення
прийнято", Toast.LENGTH_SHORT).show();
            finish();
        }
    });

    alertDialog.setNegativeButton ("Відмінити", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });

    alertDialog.show();
}

private void loadOrders() {
    cart = new Database(this).getCart();
    adapter = new CartAdapter(cart, this);
    adapter.notifyDataSetChanged();
    recyclerView.setAdapter(adapter);

    //Calculate price

```

```
int total = 0;
for (Order order : cart) {
    total += (Integer.parseInt(order.getPrice())) *
(Integer.parseInt(order.getQuantity()));

    Locale locale = new Locale("ua", "UA");
    NumberFormat format =
NumberFormat.getCurrencyInstance(locale);

    tvTotal.setText(format.format(total));
}
}
```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СЕРВІСУ ДОСТАВКИ ЇЖІ

Виконав
ст. гр. КНТ-229

Андрій ШЕПОТЬКО

Керівник
к.т.н., доцент

Євгеній ГОФМАН

1

Рисунок Д.1 – Слайд 1

Об'єктом дослідження є процес розробки програмного забезпечення для сфери обслуговування.

Предмет дослідження – програмні засоби для підтримки сервісів доставки їжі.

Мета роботи – розробка програмного забезпечення сервісу доставки їжі.

2

Рисунок Д.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

- виконати аналіз предметної області та програмних засобів для сервісів доставки їжі;
- здійснити проектування програмного забезпечення сервісу доставки їжі;
- створити програмне забезпечення сервісу доставки їжі;
- виконати тестування розробленого програмного забезпечення сервісу доставки їжі.

3

Рисунок Д.3 – Слайд 3

ПОРІВНЯННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СЕРВІСІВ ДОСТАВКИ ЇЖІ

Критерій порівняння	eda.ua	ekipazh-service	Red Box
Реєстрація користувача	+	+	+
Різноманіття страв	+	+	–
Можливість залишити коментар	–	+	–
Можливість оцінити страву	–	–	–
Можливість додати страву до «улюблених»	–	–	–
Можливість вибору декількох ресторанів	+	+	–
Можливість пошуку страв	+	+	+

4

Рисунок Д.4 – Слайд 4

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	Java	C#	PHP
Зручність для створення ПЗ сервісу доставки їжі	+	—	—
Зручний синтаксис	+	+	+-
Висока продуктивність	+	+-	+-
Багатопоточність	+	+	+-
Архітектурна незалежність	+	—	+

5

Рисунок Д.5 – Слайд 5

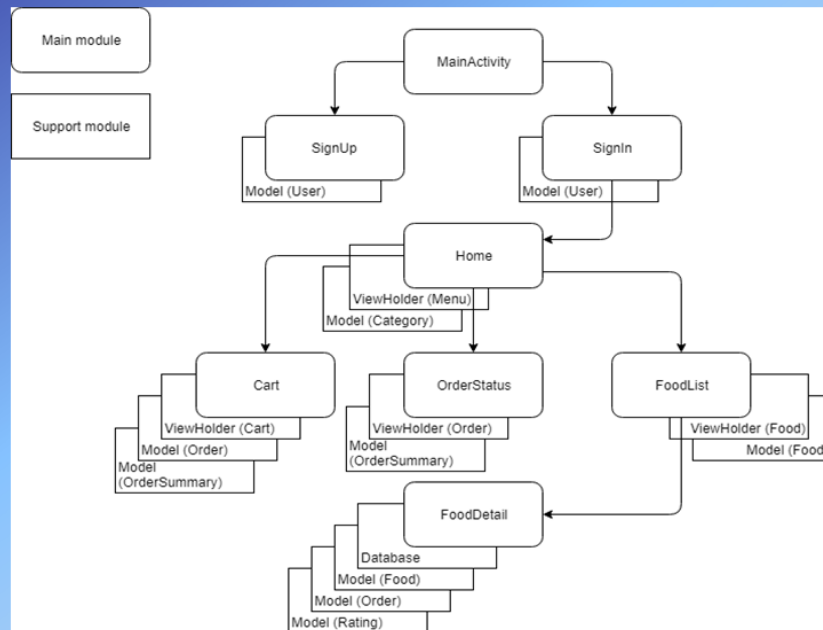
ПОРІВНЯЛЬНИЙ АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовище розробки		
	Android Studio	Atom	MS Visual Studio
Наявність шаблонів для створення поширених типів програм	+	+-	+
Зручність використання для створення програмного забезпечення сервісу доставки їжі	+	+-	+-
Зручність засобів для роботи з мовою Java	+	+-	+-
Консоль розробника	+	+-	+
Засоби Google Analytics	+	—	—

6

Рисунок Д.6 – Слайд 6

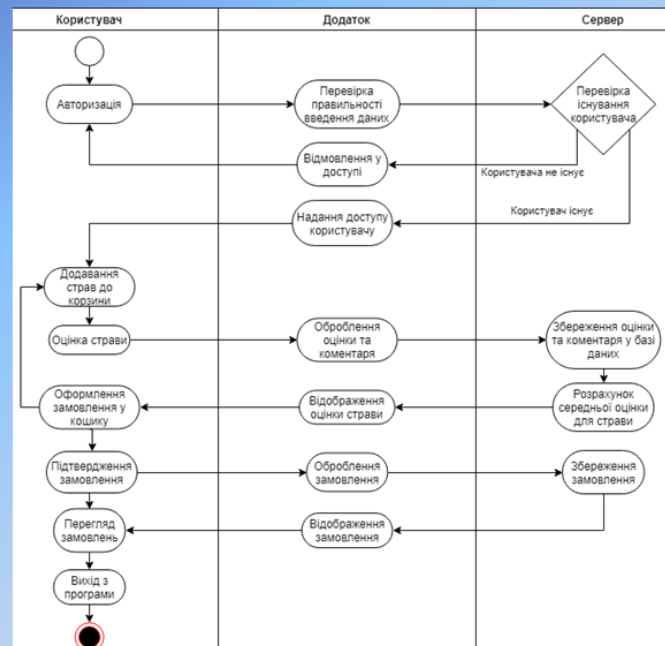
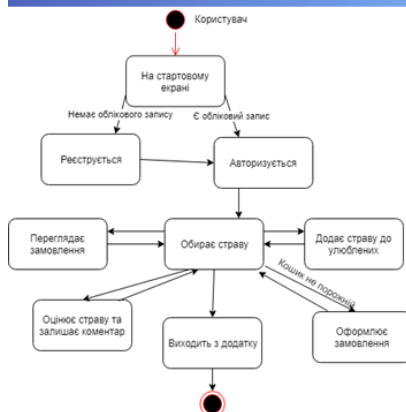
СТРУКТУРНА СХЕМА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СЕРВІСУ ДОСТАВКИ ЇЖІ



7

Рисунок Д.7 – Слайд 7

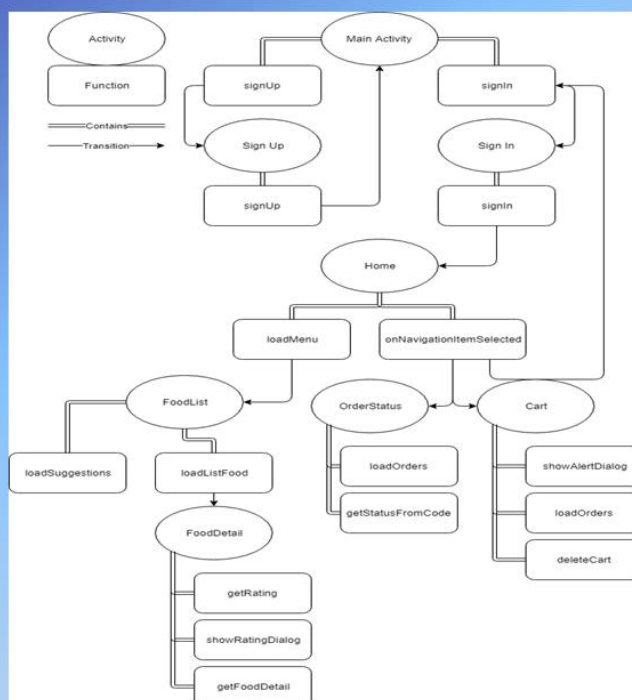
ДІАГРАМИ СТАНІВ ТА ДІЯЛЬНОСТІ ПРОГРАМИ



8

Рисунок Д.8 – Слайд 8

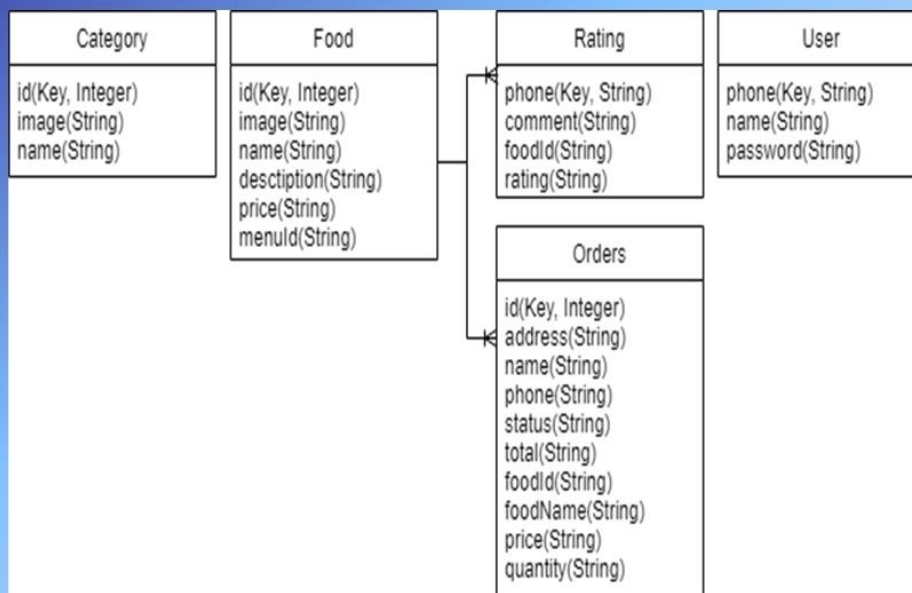
СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМИ



9

Рисунок Д.9 – Слайд 9

СХЕМА БАЗИ ДАНИХ



10

Рисунок Д.10 – Слайд 10

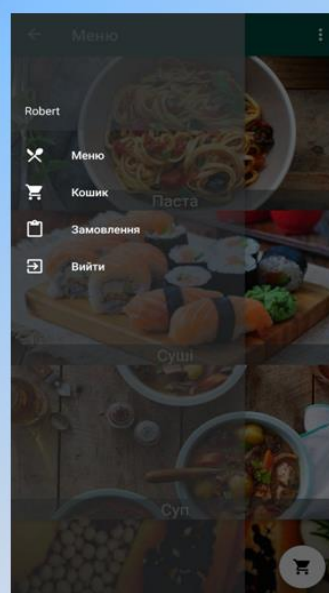
ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Стартова сторінка.



11

Рисунок Д.11 – Слайд 11

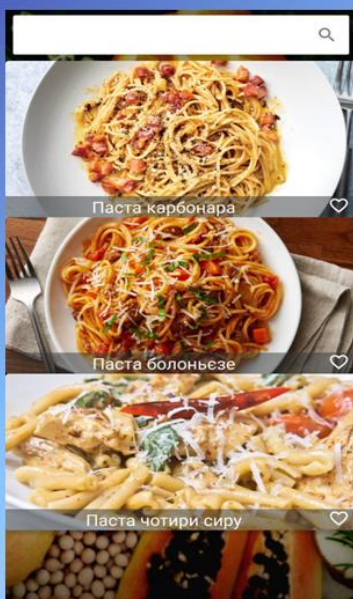
ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Вікно меню. Бокове навігаційне меню



12

Рисунок Д.12 – Слайд 12

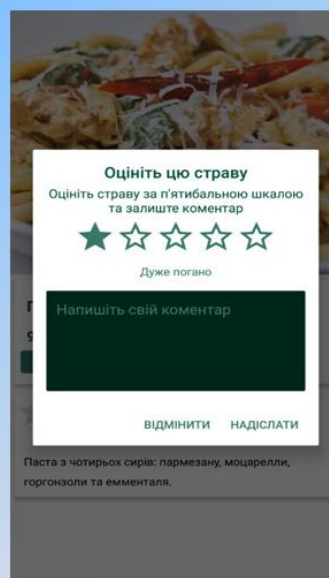
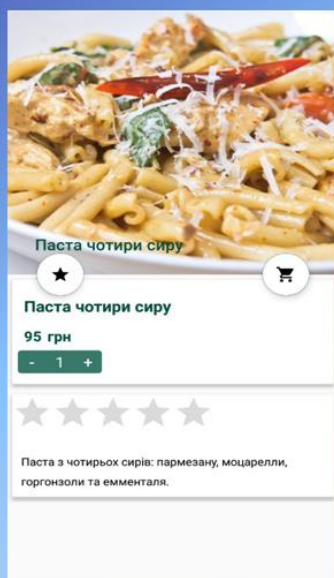
ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Страви у категорії “Паста”. Сторінка кошика



13

Рисунок Д.13 – Слайд 13

ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Детальна інформація про страву. Сторінка оцінювання страви



14

Рисунок Д.14 – Слайд 14

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для сфери обслуговування.

Проаналізовано загальні поняття про онлайн-замовлення та сервіси доставки їжі. Визначено, що програмне забезпечення для доставки їжі – це комп'ютерна програма, яка надає власникам ресторанів, іншим виробникам готової їжі та службам доставки, використання вебінструментів для маркетингу та управління діяльністю свого підприємства.

Сформульовано функціональні вимоги до програмного забезпечення сервісу доставки їжі.

Для створення програмного забезпечення сервісу доставки їжі було обрано мову програмування Java, яка є досить простою та зручною у використанні, об'єктно-орієнтованою, а також забезпечує архітектурну незалежність та мобільність.

В якості середовища розробки програмного забезпечення сервісу доставки їжі було обрано Android Studio, що є інтегрованою платформою для розробки програм для платформи Android.

15

Рисунок Д.15 – Слайд 15

ВИСНОВКИ

Розроблено програмне забезпечення сервісу доставки їжі. Запропоновано структуру та описано основні модулі програмного забезпечення сервісу доставки їжі. Визначено, що програмне забезпечення сервісу доставки їжі складається з таких модулів: модулю стартового екрану MainActivity, модулю реєстрації SignUp, модулю авторизації SignIn, модулю меню Home, модулю списку їжі для пункту меню FoodList, модулю страв FoodDetail, модулю існуючих замовлень OrderStatus, модулю кошика Cart, модулю для утримання елементу списків, що знаходяться у інших модулях ViewHolder, модулю моделі для утримання класів, що відповідають таблицям бази даних Model, модулю для збереження даних поточного користувача Common, модулю для роботи з локальною базою даних Database.

Результати тестування показали, що розроблений програмний продукт може використовуватися за призначенням.

16

Рисунок Д.16 – Слайд 16