Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РЕСТОРАНУ З ДОСТАВКОЮ ЇЖІ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-3
Поляков В.Д.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Полякову Владиславу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інформаційної системи ресторану з доставкою їжі

затверджена наказом університету від 20 травня 2024 року № 464 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25 травня 2024 р.
3. Вихідні дані до роботи зображення страв, дані інтернет-мережі.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих систем з доставки та онлайн ресторанів з доставкою.
2. Моделювання інформаційної системи ресторану з доставкою їжі.
3. Розробка інформаційної системи ресторану з доставкою їжі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)Актуальність системи ресторану, постановка задачі, алгоритми системи, результати роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-10.04.24	
3	Аналіз літератури з досліджуваної проблеми	11.04.24-15.04.24	
4	Аналіз програмних засобів	15.04.24-19.04.24	
5	Розробка алгоритмів	20.04.24-23.04.24	
6	Програмна реалізація	24.04.24-12.05.24	
7	Оформлення пояснювальної записки	14.05.24-25.05.24	
8	Перевірка на плагіат	26.05.24	
9	Рецензування	27.05.24	
10	Підготовка презентації та доповіді	27.05.24-04.06.24	
11	Занесення роботи в електронний архів	05.06.24	
12	Попередній захист кваліфікаційної роботи	05.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ _____ проф. Машталір С.В.____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 64 с., 3 табл., 31 рис., 30 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, JAVASCRIPT, VITE, DATABASE, VISUAL STUDIO CODE, MONGO.

Об’єктом роботи є інформаційна система ресторану з доставкою їжі.

Метою роботи є створення інформаційного застосунку для ресторану з додаванням нових страв, а також звичайне видалення страв, перегляду усіх замовлень створених користувачами, можливістю замовлення їжі на дім та перегляд своїх замовлень для кожного унікального користувача.

Клієнтська частина та сторінка адміністратора були створені за допомогою React, серверна частина створена за допомогою Node.js і Express.js. Розробка вебзастосунку виконувалась в середовищі Visual Studio Code.

У результаті роботи здійснена програмна реалізація вебзастосунка ресторану для перегляду усіх страв та замовлення на адресу та редагування меню ресторану з сторінки адміністратора, та перегляду усіх замовлень.

INFORMATION SYSTEM, JAVASCRIPT, VITE, DATABASE, VISUAL STUDIO CODE, MONGO.

The object of the work is the information system of a restaurant with food delivery.

The goal of the work is to create an informational application for a restaurant with the addition of new dishes, as well as the usual removal of dishes, viewing all orders created by users, the ability to order food at home and view their orders for each unique user.

The client side and the admin page were built using React, the server side was built using Node.js and Express.js. The development of the web application was carried out in the Visual Studio Code environment.

As a result of the work, a software implementation of the restaurant’s web application for viewing all dishes and orders to the address and editing the restaurant’s menu from the administrator’s page, and viewing all orders was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IOS – iPhone Operating System (мобільна операційна система)

БД – база даних

API – Application Programming Interface (програмний інтерфейс застосунку)

UML – Unified Modeling Language (мова графічного опису для об’єктного моделювання)

JSON – JavaScript Object Notation (текстовий формат обміну даними, заснований на JavaScript)

SQL – Structured Query Language (це стандартна мова запитів для взаємодії з реляційними базами даних)

ES – Export System (система експорту)

DOM – Document Object Model (інтерфейс програмування програм)

XML – Extensible Markup Language (мова розмітки, що розширюється)

ЦП – центральний процесор

ВСТУП

Постійний розвиток технологій та прагнення до більш комфортнішого життя вимагають нових підходів до організації та управління ресторанним бізнесом. Швидкість та зручність стали ключовими факторами, визначальними у виборі споживачем місця для прийому їжі. У зв’язку з цим, розробка інформаційних систем для ресторанів з доставкою їжі стає актуальною задачею, вимагаючи комплексного підходу та використання нових технологій.

Ця кваліфікаційна робота спрямована на розробку інформаційної системи, яка не лише сприятиме оптимізації управління ресторанним бізнесом, але й забезпечить максимальний рівень задоволення клієнтів шляхом поліпшення процесу замовлення та доставки страв, а також допоможе власникам розширити базу клієнтів і кількість замовлень.

Актуальність роботи полягає у тому що, ресторани які побажають масштабуватися їм прийдеться думати над тим як вони будуть уподобати клієнтським потребам з доставкою їжі, а завдяки цієї роботи я можу продемонструвати можливості функціоналу та дизайну на який результат вони можуть розраховувати.

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ З ДОСТАВКИ ТА ОНЛАЙН РЕСТОРАНІВ З ДОСТАВКОЮ

1.1 Сучасний стан доставки їжі та ресторанів в Україні та інших країнах світу

У сучасному світі, де суб’єктивний час стає все більш цінним ресурсом, суспільство все більше стурбоване своїм часом. Люди прагнуть оптимізувати свій час та уникати витрат на рутинні або малоприємні справи, такі як похід у ресторан. Вони все частіше віддають перевагу проведенню часу вдома з родиною або виконанню своїх особистих або професійних справ.

Час, який раніше був б витрачений на дорогу до ресторану та очікування на обслуговування, тепер можна провести вдома, насолоджуючись якісними стравами та приємною атмосферою. Завдяки зростанню популярності послуг доставки їжі, навіть без виходу з дому люди можуть смакувати різноманітні страви, які раніше вони могли спробувати тільки у вишуканих ресторанах.

Цей тренд демонструє зміну підходу до харчування та відпочинку, де люди більше цінують свої особистість та зручність. Таким чином, навіть у затишку свого домівка, вони можуть насолоджуватись вишуканими стравами та якісним сервісом, що відкриває нові можливості для розвитку ресторанного бізнесу та сервісу доставки їжі.

Крім ресторанних сервісів доставки, існують також окремі сервіси, які розширюють свої послуги на доставку не лише з ресторанів, але й з магазинів. Це дозволяє клієнтам замовляти не лише готові страви, а й продукти харчування та інші товари безпосередньо до свого дому.

Проте, на суб’єктивну думку, існують деякі недоліки у таких сервісах. По-перше, відсутність особистого контакту з менеджером ресторану може ускладнити спілкування щодо уточнення замовлення або вирішення проблем. У разі, якщо потрібно обговорити деталі з поваром або зробити спеціальний запит щодо страви, відсутність прямого зв’язку може бути недоліком.

По-друге, оскільки такі сервіси працюють з різними ресторанами та магазинами, немає гарантії, що якість та імідж закладу будуть збережені належним чином під час доставки. Не всі ресторани можуть мати однакову стандартизацію сервісу та якості приготування страв, що може призвести до ризику отримання неякісної їжі або незадовільного обслуговування.

З розвитком ресторанного сервісу і появою нових закладів зі своїми унікальними поглядами на вже існуючі стави, так і на зовсім нові ми отримали більше різноманітності при виборі страв. Завдяки зростанню кількості ресторанів, які представлені також онлайн, ми маємо доступ до широкого асортименту кулінарних напрямків і стилів кухонь. Ресторани конкурують між собою, що стимулює їх розширювати та удосконалювати меню, щоб привернути більше клієнтів.

Це означає, що як споживачі, ми можемо насолоджуватися більшим розмаїттям страв та вибирати ті, які відповідають нашим смаковим уподобанням та дієтичним потребам. Завдяки онлайн сервісам ми можемо швидко та зручно знайти та замовити саме те, що забажаємо, без зайвих зусиль та часових витрат на обхід багатьох ресторанів.

Отже, розвиток ресторанних сервісів в Інтернеті відкриває перед нами більше можливостей та сприяє розширенню нашого кулінарного досвіду, дозволяючи насолоджуватися різноманіттям блюд з усього світу не виходячи з дому.

Конкуренція між ресторанами є важливим фактором, що стимулює їх швидкий розвиток і удосконалення сервісу. Для того щоб зберегти та привернути нових клієнтів, ресторани стають більш інноваційними та розширюють свої можливості. Це включає в себе розширення меню, введення нових кулінарних та культурних напрямків, вдосконалення обслуговування та впровадження нових технологій.

Проте, коли мова йде про сервіс доставки їжі, ситуація є трохи іншою. Хоча ресторани стараються зробити процес доставки більш зручним і ефективним, обмеженість в можливостях прогресу стається очевидною.

Головною метою у цьому випадку залишається зменшення часу доставки та мінімізація ризику пошкодження чи порушення їжі під час транспортування.

Таким чином, хоча ресторани продовжують конкурувати за увагу та лояльність клієнтів через розвиток та удосконалення своїх послуг, сервіси доставки їжі залишаються зосередженими на забезпеченні якісної та швидкої доставки, що стає важливим чинником в їхній конкурентній боротьбі.

Особливий зріст популярності доставки стався під час пандемії. Під час COVID-19 сервіс доставки їжі в ресторанах, звичайних супермаркетах став невід'ємною частиною сучасного способу життя для багатьох людей по всьому світу.

За час карантинних обмежень багато закладів змушені були переорієнтувати свою діяльність на доставку їжі замість обслуговування у приміщенні. Це стало причиною ще більшого попиту на послуги доставки та росту кількості ресторанів, які використовують цей спосіб залучення клієнтів.

Україна також не стала виключенням з цього тренду. Останнім часом на ринку постійно з'являються нові ресторани та кафе, які активно використовують послуги доставки для розширення своєї аудиторії та збільшення обігу. Це стосується як мережевих закладів, так і невеликих кулінарних майстерень, які спеціалізуються на конкретному виді кухні.

У світі також спостерігається аналогічний тренд. У багатьох країнах росте популярність сервісів доставки їжі, таких як Uber Eats, DoorDash, Deliveroo та інші. Ці компанії здійснюють доставку їжі з різноманітних ресторанів безпосередньо до дому або офісу замовника, забезпечуючи зручний та швидкий спосіб харчування.

Зростаюча кількість ресторанів та кафе, які почали пропонувати свої послуги доставки їжі або почали працювати з іншими сервісами які пропонують свої послуги по замовленню і доставці їжі, свідчить про поширення цього сервісу по всьому світу.

1.2 Сучасні ресторани в Україні

Україна відома своєю багатою культурою гастрономії, яка здатна конкурувати з найпрестижнішими гастрономічними центрами світу, такими як Нью-Йорк. У багатьох містах країни можна знайти унікальні авторські заклади, що відзначаються своєрідними концепціями та вишуканим меню.

Проте, на жаль, не всі ресторани в Україні прагнуть розвивати сервіс доставки їжі. Багато з них акцентують увагу на обслуговуванні клієнтів в приміщенні, вважаючи це більш перспективним інвестиційним напрямком. Втім, навіть серед цих закладів не варто забувати про широке розмаїття ресторанних мереж та високопрофільних авторських закладів, які надають послуги доставки їжі:

– «Mafia» – це мережа ресторанів італійської та японської кухні, який працює у декількох містах України. Вони можуть запропонувати достатньо велике меню: піци, суші та сети, салати, десерти, супи, гарячі страви, бургери, напої. Заклад має не тільки багато ресторанів, але і пропонує і доставку на дім [1];

– «Япошка» – це мережа закладів японської та корейської кухні, який працює в чотирьох містах України, та здійснює доставку по цим містам. Заклад може пропонувати велике різноманіття вибору суші та сетів [2];

– «Georgia» – заклад у місті Київ який готує грузинську кухню. Цей заклад не являє собою велику мережу, тому замовлення страв на дім виконується на невеликій території лівобережної сторони Києва [3].

Всі три заклади об’єднує те, що вони надають можливість замовлення напряму з ресторану. Проте, вони також поширюють свої послуги, представляючи свої заклади в окремих застосунках, де клієнти можуть легко та зручно замовити доставку. Цей підхід допомагає розширити базу клієнтів і забезпечити більшу доступність для користувачів, які шукають улюблені страви без виходу з дому.

Замовлення не тільки напряму з ресторану, а також з окремих застосунків роблять процес замовлення ще більш зручним і доступним для клієнтів. Цей підхід дозволяє користувачам легко переглядати меню, обирати бажані страви та замовляти доставку безпосередньо зі свого смартфона або комп’ютера.

Розповсюдження ресторанів через окремі застосунки також сприяє розширенню аудиторії клієнтів. Користувачі, які раніше можливо не мали можливості відвідувати заклади на місці, тепер можуть насолоджуватися їхніми стравами безпосередньо вдома, завдяки зручному сервісу доставки. Це створює додаткові можливості для ресторанів привернути нових клієнтів та розширити своє вплив.

Як вже було сказано, після початку пандемії в Україні спостерігалося масове поліпшення та розповсюдження застосунків з доставкою. В умовах обмежень та карантину більше людей почали шукати альтернативні способи отримання продуктів та послуг, в тому числі й харчування. Це призвело до зростання популярності мобільних програм для замовлення їжі, оскільки вони надають зручний та безпечний спосіб отримання необхідних продуктів прямо до дверей клієнтів. Це сприяло активному розвитку ринку доставки та побудові ефективної інфраструктури для задоволення зростаючого попиту [4, 5]. Тому буде правильно розглянути ще окремі застосунки які працюють в Україні:

– «Glovo» – це міжнародний сервіс доставки, який надає користувачам можливість замовляти різноманітні товари включаючи не тільки їжу, а також продукти, ліки, косметику та багато іншого. Однією з особливостей Glovo є його широкий асортимент партнерських закладів та магазинів, з яких можна замовити продукти або послуги. Користувачі мають можливість вибирати із великої кількості варіантів і замовляти все, що їм потрібно, зручно та швидко;

– «Raketa» – ще один із популярних сервісів доставки Rocket працює вже більш ніж у 20 містах України, зокрема у Києві, Львові, Дніпрі, Харкові та Одесі. Однак сервіс не має комп’ютерної версії, тому скористатися

послугами доставки можна тільки зі смартфона або планшета на IOS або Android. Доставка їжі коштує 40 гривень, але періодично на платформі відбуваються акції, тому з деяких ресторанів вона може бути навіть безкоштовною. У Rocket можна замовити товари з магазинів, які розташовані поруч з адресою доставки. Проте за великої завантаженості сервіс іноді досить довго шукає кур'єрів;

– «UberEats» – це відомий міжнародний сервіс доставки їжі, який діє в багатьох країнах світу, включаючи Україну. Цей сервіс є частиною компанії Uber, яка відома своїми послугами з таксі. UberEats відомий своїм широким асортиментом закладів-партнерів та різноманітністю кухонь, які представлені у його програмі. Користувачі можуть легко переглядати меню різних ресторанів, кафе та інших закладів, обирати бажані страви та робити замовлення всього за кілька клацань. Однією з переваг UberEats є швидка та надійна доставка. Завдяки великій мережі кур'єрів, замовлення доставляються вчасно і з дотриманням високих стандартів якості. Крім того, користувачі можуть відстежувати свої замовлення у реальному часі та отримувати сповіщення про статус доставки;

– «Bolt Food» – компанія Bolt на зміну UberEats запустила сервіс з доставки їжі Bolt Food у Києві. До платформи вже приєдналися понад 150 ресторанів, серед яких KFC, Пузата Хата, Тайський привіт, Китайський привіт, В'єтнамський привіт, Meiwei, БагатоЛосося, Musafir, Cinnabon. Поки що доставка в програмі безкоштовна, проте за доставку з деяких ресторанів все ж таки доведеться заплатити, але вона коштує символічну суму. Згодом доставка має стати платною;

– «Mister.am» – сервіс менш популярний, ніж згадуваний вище, але його плюсом є те, що він представлений у невеликих містах України. А саме у Вінниці, Чернігові, Житомирі, Івано-Франківську, Кременчуці, Кропивницькому, Луцьку. Вартість послуги доставки починається від 45 грн., проте за доставку до районів міста може стягуватися додаткова плата. Замовлення можна зробити на сайті або зателефонувати менеджеру.

Якщо проаналізувати ринок на поприщі доставок то можна виділити такі критерії та їх характеристики:

– широкий асортимент продуктів та послуг. У всіх сервісах доставки їжі, що діють в Україні, існує великий вибір закладів-партнерів, що пропонують різноманітні кухні та страви. Це робить їх привабливими для широкого кола користувачів з різними уподобаннями щодо їжі;

– зручний інтерфейс та простий процес замовлення. Успішні сервіси доставки їжі пропонують зручні мобільні програми та вебзастосунки, що дозволяють користувачам легко шукати, обирати та замовляти страви зі списку партнерських закладів. Простий та зрозумілий процес замовлення зменшує бар’єри для користувачів і сприяє збільшенню популярності сервісу;

– швидка та надійна доставка. Важливим аспектом є якість та швидкість доставки. Успішні сервіси забезпечують своїх користувачів швидкою та надійною доставкою, що сприяє задоволенню клієнтів та позитивному досвіду використання сервісу;

– акції та знижки. Багато сервісів доставки їжі регулярно проводять акції та пропонують знижки своїм користувачам. Це може включати безкоштовну доставку, знижки на конкретні заклади або страви, що робить користування сервісом більш привабливим та вигідним для клієнтів;

– клієнтська підтримка та відгуки. успішні сервіси доставки їжі приділяють увагу клієнтській підтримці та враховують відгуки своїх користувачів для постійного вдосконалення сервісу та вирішення проблем, які можуть виникнути під час користування.

1.3 Використання технологій в сучасних сервісах

Сучасні ресторанні сервіси з доставкою їжі використовують різноманітні технології для покращення ефективності та зручності процесу замовлення та доставки. Ось деякі з технологій, які вони використовують:

– мобільні програми: багато ресторанів мають власні мобільні програми, які дозволяють клієнтам зручно оформляти замовлення, вибирати страви, сплачувати за них та відстежувати процес доставки;

– вебсайти: крім мобільних програм, багато ресторанів також мають вебсайти, де клієнти можуть здійснювати замовлення через браузер;

– системи управління замовленнями: ресторани використовують системи управління замовленнями для ефективного прийому, обробки та виконання замовлень клієнтів. Ці системи дозволяють автоматизувати багато процесів, що допомагає знизити час очікування та запобігти помилкам в замовленнях;

– оптимізація маршрутів доставки: деякі ресторани використовують програми для оптимізації маршрутів доставки, які дозволяють кур’єрам ефективно виконувати замовлення та скорочувати час доставки;

– системи відстеження замовлень: більшість сервісів доставки їжі надають можливість клієнтам відстежувати свої замовлення в реальному часі через мобільні програми або вебсайти. Це дозволяє користувачам бути в курсі, де знаходиться їх замовлення та коли вони можуть очікувати його отримання;

– інтеграція з платіжними системами: для зручності клієнтів багато ресторанів і сервісів доставки інтегруються з різними платіжними системами, що дозволяє здійснювати оплату за замовлення онлайн через кредитні картки, електронні гаманці та інші способи оплати.

Ці технології допомагають ресторанам та сервісам доставки їжі забезпечити зручний та ефективний процес замовлення та доставки для своїх клієнтів. Але звичайно сучасні сервіси мають і свої недоліки, ось декілька із них:

– технічні проблеми: мобільні програмні застосунки та вебсайти можуть мати технічні проблеми, такі як зависання, збої в роботі або нездатність завантажувати сторінки. Це може спричинити незручності для клієнтів та призвести до втрати замовлень для ресторанів;

– помилки в замовленнях: використання систем управління замовленнями може призвести до помилок в обробці замовлень, таких як неправильні страви, неправильні адреси доставки або втрати замовлень. Це може призвести до незадоволення клієнтів та втрати репутації для ресторанів;

– затримки в доставці: хоча існують системи для оптимізації маршрутів доставки, проте затримки все ще можуть виникати через різні обставини, такі як непередбачувані транспортні пригоди, погодні умови або недостатня кількість кур’єрів. Це може призвести до незадоволення клієнтів і втрати їхньої лояльності;

– проблеми з платіжними системами: інтеграція з різними платіжними системами може призвести до проблем з оплатою за замовлення, таких як помилкові транзакції, відмови в оплаті або затримки в обробці платежів. Це може призвести до незадоволення клієнтів та втрати довіри до сервісу;

– проблеми з безпекою даних: використання мобільних програм та вебсайтів для замовлення їжі може ставити під загрозу конфіденційність особистих даних клієнтів, якщо система не забезпечує високий рівень захисту даних. Це може призвести до проблем з приватністю та потенційно до кібератак або витоку даних.

Не всі проблеми легко вирішуються, але з кожним роком ці недоліки все більше проробляють, щоб позбавитись їх.

1.4 Логістика

Різні сервіси доставки їжі в Україні використовують різні підходи до визначення маршрутів доставки. Використання картографічних сервісів, таких як Google Maps. Багато сервісів використовують інтегровані картографічні сервіси, такі як Google Maps, для планування оптимальних маршрутів доставки [6].

Google Maps надає детальну інформацію про дороги, трафік, розташування ресторанів та адреси доставки, що дозволяє сервісам ефективно планувати маршрути кур’єрів.

Прописування власних логістичних маршрутів. Деякі сервіси доставки можуть мати власні системи логістики та маршрутизації, розроблені внутрішньою командою або за допомогою сторонніх розробників програмного забезпечення.

Ці системи можуть бути спеціально налаштовані для потреб конкретного сервісу доставки та враховувати різні фактори, такі як час, відстань, обсяг замовлення, трафік тощо.

Кожен з цих підходів має свої переваги і недоліки, і вибір конкретного методу може залежати від потреб і можливостей конкретного сервісу доставки. Наприклад, якщо заклад або сервіс з доставки використовує транспортні засоби: машина, велосипед, самокат, тоді краще і логічніше використовувати картографічні сервіси, аніж самому прораховувати можливі маршрути. Такі сервіси, як Google Maps, можуть показувати затори на проїзних частинах, а це може зберегти час який критично важливий при доставці страв. Однак незалежно від використаного методу, головною метою є забезпечення швидкої та надійної доставки їжі до клієнтів.

1.5 Варіативність меню

Деякі сервіси доставки їжі та ресторани з доставкою надають користувачам можливість додавати додаткові інгредієнти до блюд за їхнім вибором. Це дозволяє клієнтам персоналізувати свої замовлення та насолоджуватися стравами, які відповідають їхнім смаковим уподобанням.

Наприклад, якщо клієнт замовляє піцу, він може мати можливість додати до неї більше грибів, сиру, оливок або інших інгредієнтів. Також

можуть бути доступні різні варіанти начинки або соусів для вибору, які можна додати до блюда.

Крім того, деякі сервіси після додавання блюд в корзину можуть надавати рекомендації до замовлення. Це можуть бути рекомендовані додаткові страви, напої або десерти, які доповнюють вибір клієнта та роблять його замовлення більш насиченим та задовільним.

Ці можливості дозволяють користувачам насолоджуватися більш персоналізованим досвідом замовлення їжі та отримувати рекомендації щодо страв, які можуть їм сподобатися, збагачуючи їхній вибір та роблячи процес замовлення більш зручним і приємним.

1.6 Програма лояльності

Програми лояльності, які впроваджують ресторани та сервіси доставки їжі, є ефективним інструментом для залучення та утримання клієнтів. Ці програми спрямовані на стимулювання повторних замовлень та підвищення задоволеності клієнтів, пропонуючи різноманітні переваги та бонуси. Ось деякі характеристики програм лояльності:

– бонусні бали: користувачі можуть отримувати бонусні бали або очки за кожну замовлену страву або послугу доставки. Ці бали можна накопичувати та обмінювати на знижки, безкоштовні страви або інші подарунки;

– знижки та спеціальні пропозиції: учасники програм лояльності можуть отримувати ексклюзивні знижки, спеціальні пропозиції або доступ до обмежених акцій, які не доступні іншим клієнтам;

– преміальні рівні: деякі програми лояльності можуть мати систему преміальних рівнів, де клієнти можуть отримувати додаткові переваги та пільги, якщо вони досягають певного рівня витрат або активності;

– збереження історії замовлень: програми лояльності можуть зберігати історію замовлень клієнта та забезпечувати зручний доступ до неї для повторного замовлення улюблених страв.

Ці програми створюють взаємовигідні умови як для клієнтів, так і для ресторанів та сервісів доставки, сприяючи збільшенню лояльності клієнтів та підвищенню їхньої задоволеності, що відіграє ключову роль у розвитку та успіху бізнесу.

1.7 Постановка задачі

Таким чином, розробка інформаційної системи ресторану з доставкою є актуальним завданням. Тому ставиться завдання розробки вебзастосунку, який буде включати в собі сучасний дизайн і усі необхідні функції.

Об’єктом роботи є інформаційна система ресторану з доставкою їжі.

Метою роботи є створення інформаційного застосунку для ресторану з додаванням нових страв, а також звичайне видалення страв, перегляду усіх замовлень створених користувачами, можливістю замовлення їжі на дім та перегляд своїх замовлень для кожного унікального користувача.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих ресторанів та програм які надають можливість замовити страви;

– проаналізувати популярні меню страв для створення власного меню;

– визначити як здійснюють доставку інші заклади;

– визначити як ресторани впровадили систему і як вона працює;

– проаналізувати технології які можна використати для реалізації проєкту;

– змоделювати архітектуру алгоритмів вебзастосунку;

– налаштувати навігацію вебзастосунку;

– реалізувати систему авторизації та реєстрації користувача;

- реалізувати алгоритм додавання страв в кошик;
- реалізувати алгоритм підтвердження в кошику;
- реалізувати форму замовлення з кошику;
- реалізувати функцію оплати замовлення;
- реалізувати перегляд своїх заказів у користувача;
- реалізувати сторінку в адмін-панелі для управління;
- реалізувати алгоритм додавання нових страв в меню;
- реалізувати алгоритм для перегляду всіх замовлень користувачів;
- реалізувати алгоритм для перегляду всіх страв в меню і їх видалення.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ РЕСТОРАНУ З ДОСТАВКОЮ ЇЖІ

2.1 Вимоги до інформаційної системи ресторану

Система ресторанного сервісу має застосовувати ряд зручних функцій для забезпечення комфорту та ефективності використання. Основні можливості включають демонстрацію всіх страв на головній сторінці. При вході на сайт або застосунок користувач може переглянути повний перелік доступних страв з їх зображеннями та основною інформацією.

Також система повинна надавати список меню з можливістю фільтрації за категоріями. Меню буде розподілене за категоріями, щоб користувач міг швидко знаходити потрібні страви.

Користувач може створити свій особистий обліковий запис, використовуючи електронну пошту та пароль. Зареєстровані користувачі можуть увійти в свій обліковий запис, де зможуть переглянути усі свої минулі замовлення та відстежувати статус поточного замовлення.

Для адміністратора доступна окрема сторінка, де він може керувати контентом застосунку. Він може додавати нові страви, видаляти, переглядати замовлення користувачів та змінювати їх статус доставки.

Ці можливості забезпечують зручність використання платформи як для користувачів, так і для адміністратора, забезпечуючи швидке та ефективне оформлення замовлень та управління ресторанним бізнесом.

Всього є 2 види користувачів: адміністратор і клієнт ресторану. У кожного є свої власні можливості в користуванні сервісу. Наглядно це можна побачити на рисунку 2.1.

Можливості клієнта сервісу:

– перегляд усіх доступних страв на головній сторінці. Коли користувач заходить на головну сторінку сайту, перше що він побачить буде навігаційна панель, де він може перейти на частину з меню та контактною інформацією;

– фільтрація страв за категоріями з меню. Користувач може відсортувати загальний список всіх страв з меню, після фільтрації користувач нижче побачить тільки ті страви якої категорії він обрав меню;

– створення особистого облікового запису з використанням імені, електронної пошти та пароля. Якщо користувач ніколи не користувався застосунком, то для повноцінної роботи потрібно створити обліковий запис;

– вхід у свій власних обліковий запис з використанням електронної пошти та пароля. Якщо у користувача вже створений обліковий запис, то він може авторизуватися за тією інформацією яку він вказував при реєстрації;

– додавання усіх бажаних страв в кошик. На карточці зі стравою впроваджена іконка додавання страви кошик;

– регулювання кількості доданих в кошик страв на головній сторінці. Після натиснення іконки додавання в кошик, користувачу буде надано можливість обрати кількість страви яку він бажає додати в кошик;

– перегляд кошику та усіх страв які були додані в кошик. Користувач може побачити на навігаційному полі в шапці застосунку іконку кошику, натиснувши її буде відкрито нове вікно в якому користувач побачить ті страви які він обрав;

– заповнення даних адреси та власних контактів для можливості зв’язку з замовником. Якщо користувач підтвердив свій кошик, відкриється нове вікно де він може заповнити інформацію для підтвердження замовлення;

– оплата замовлення з використанням платіжної системи Stripe. Після заповнення інформації користувач може оплатити замовлення в платіжній системі;

– перехід до сторінки своїх замовлень, де можна переглянути усі свої минулі замовлення та відстежувати статус поточного замовлення. Якщо оплата пройшла вдало, то користувача буде перенесено до його сторінки з замовленнями;

– оновлення статусу замовлення. Коли користувач перебуває на сторінці своїх замовлень, він може оновити статус свого замовлення якщо адміністратор його змінив;

– вийти зі свого облікового запису. Користувач може вийти зі свого облікового запису.

Можливості адміністратора:

– додавання нової страви та усієї необхідної інформації. Адміністратор може додавати нові страви та встановлювати усю необхідну інформацію для цієї страви [7];

– перегляд усіх існуючих страв в меню. Адміністратор у своєму вікні може бачити усі страви які відображаються на головній сторінці;

– видалення страви з меню. У тому ж вікні де адміністратор бачить всі страви, він може і видалити необхідну страву;

– перегляд усіх заказів всіх користувачів. Окреме вікно у адміністратора відображає список усіх замовлень всіх користувачів;

– регулювання статусу замовлення. У тому ж вікні, де адміністратор бачить всі замовлення, він може редагувати статус кожного замовлення, кожного з користувачів.

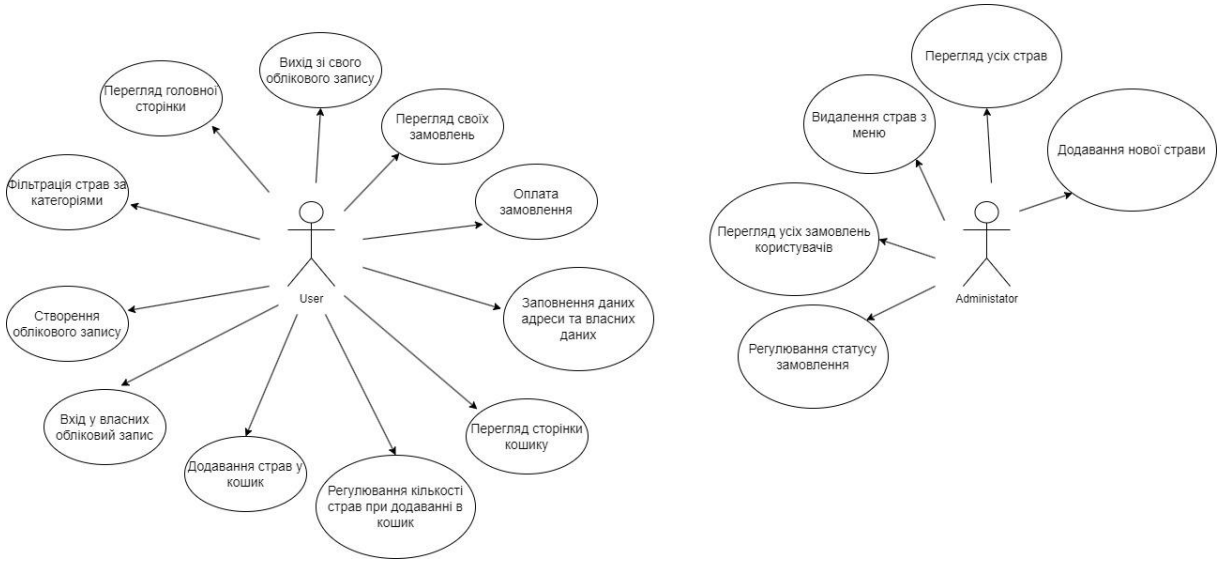

Рисунок 2.1 – Діаграма use case варіантів користуванням застосунку.

Головна Use Case діаграма яка демонструє усі варіанти користування програмою для користувача і адміністратора [8].

2.2 Моделювання архітектури клієнтської частини

Для клієнтської частини вебзастосунків розробники використовують різні інструменти та технології, серед яких основні [9]:

– React.js: це бібліотека JavaScript для розробки інтерфейсів користувача. React дозволяє розбити інтерфейс на компоненти, які зручно керувати та оновлювати, що полегшує розробку складних інтерфейсів;

– Vue.js: це прогресивний фреймворк JavaScript, який також використовується для розробки інтерфейсів користувача. Vue дозволяє створювати динамічні та інтерактивні компоненти зі зручним синтаксисом та швидким вивченням;

– Angular: це фреймворк JavaScript, розроблений компанією Google, який дозволяє створювати потужні односторінкові застосунки. Angular надає розширені можливості для роботи зі структурою застосунку та його компонентами [10].

Ці інструменти допомагають розробникам створювати ефективні та інтерактивні інтерфейси користувача, що забезпечують зручну та привабливу взаємодію зі застосунком. Вони також дозволяють швидко реагувати на зміни та покращення в дизайні та функціональності.

Для ресторанного сервісу буде обрано кольорову палітру яка не буде заважати користувачу і буде органічно виглядати. На головній сторінці буде навігаційна панель, яка буде допомагати користувачу у навігації по сайту. У навігаційному меню буде знаходитися кошик та кнопка з можливістю увійти в обліковий запис, або кнопка профілю, в залежності від статусу користувача. Щоб увійти у свій обліковий запис, користувач повинен буде правильно заповнити поле з електронною поштою та паролем.

Під навігаційною панеллю буде знаходитись секція з рекламним банером, він буде розрахований на якісь акційні пропозицій в ресторані, також на цьому банері буде знаходитись кнопка яка в подальшому буде переводити на секцію сайту яку зазначить власник сайту.

Нижче буде сектор де відображається меню ресторану. На вибір користувачу буде 8 видів страв, приклад страв як на рисунку 2.2. Користувач може обрати одне з цього меню і загальний список страв відфільтрується за тією категорією. Якщо користувач натисне на один із видів страв в меню, іконка трохи зміниться в розмірі і з’явиться темно-синій контур який дасть видиму інформацію користувачу, що кнопка активна. Під полем категорії страв буде відображатися повний список страв які є в базі даних, рисунок 2.3 для прикладу блоків зі стравами . Користувач може побачити фотографії страв, ціні, опис страв та може обирати кількість кожної страви яку він бажає замовити [11, 12].

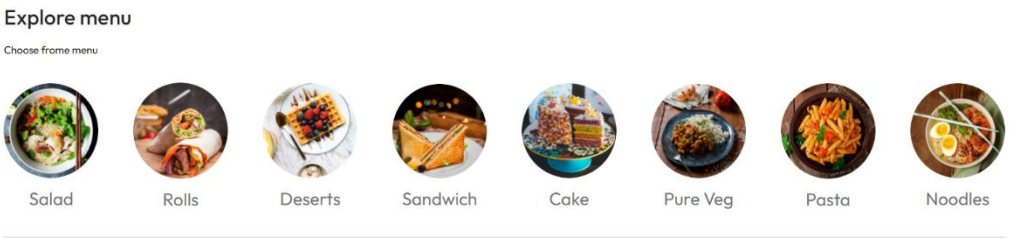

Рисунок 2.2 – Поле з меню

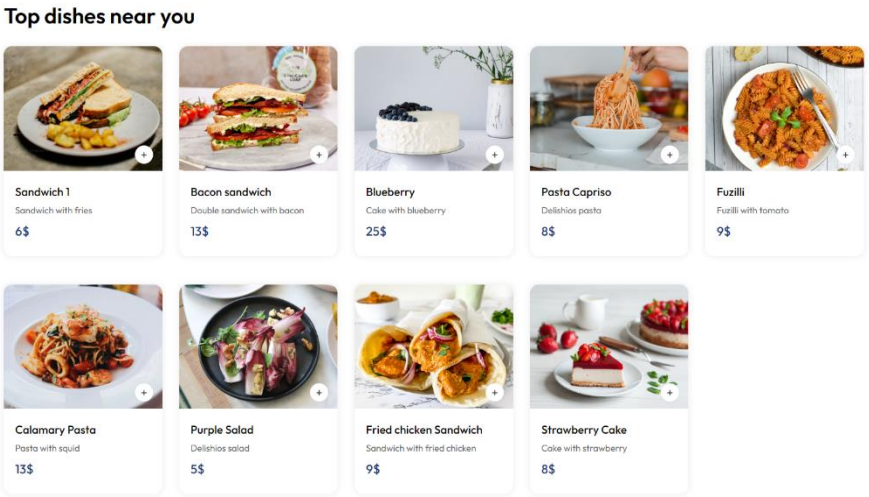

Рисунок 2.3 – Секція з усіма стравами

В кінці сторінки сайту буде відображатися окрема секція де користувач може побачити іконки до соціальний мереж ресторану в яких зареєстрований ресторанний сервіс. По середині будуть поля які зможуть відображати всю інформацію що стосується ресторану. В правому куті буде відображатися контактна інформація: номер телефону ресторану, електронна пошта для технічних питань.

Користувач при вході у свій обліковий запис зможе користуватися кошиком. До кошику будуть потрапляють страви які користувач додав з головної сторінки. На сторінці кошику користувач зможе переглянути усі страви які він обрав. Також користувач зможе видаляти з кошику кожну унікальну страву. Користувач зможе підтвердити своє замовлення.

Після підтвердження замовлення користувача буде перенесено до іншої сторінки, де він зможе заповнити усю необхідну інформацію. Поля які він повинен буде заповнити: ім’я, прізвище, електронну пошту, місто, вулицю, будинок, номер квартири та свій номер телефону за яким з ним зможуть зв’язатися за виникненням якихось питань. Також, користувач буде бачити загальну ціну та окремо ціну за доставку. Користувач зможе перейти до розділу оплати, де за допомогою спеціальних сервісів йому буде надано можливість оплатити замовлення.

Після успішної операції, користувач буде автоматично перенаправлений на сторінку, де буде відображатися його поточне замовлення, а також вся його історія замовлень. У цьому архіві користувач зможе побачити назви всіх страв, їх кількість та загальну вартість замовлення. Крім того, сторінка повинна мати поточний статус замовлення.

Сторінка адміністратора має містити наступне наповнення:

– список усіх страв які відображаються на головній сторінці;
– функції з додаванням і видаленням страви;
– архів усіх замовлень з можливістю коригування стану.

У замовлення зазвичай розрізняють 3 стани:

– приготування страви: замовлення було прийнято та знаходиться стані приготування;

– у дорозі: замовлення вже було відправлено на доставку. Кур’єр або служба доставки взяла замовлення з ресторану;

– доставлено: Замовлення було успішно доставлено клієнту.

На цьому моделювання дизайну клієнтської частини завершено.

2.3 Аналіз БД

Для управління базами даних в сучасних вебзастосунках використовуються різноманітні інструменти та технології. Ось кілька з них:

– MySQL Workbench: це офіційний графічний інструмент для розробки та управління базами даних MySQL. MySQL Workbench надає інтерфейс для моделювання баз даних, виконання SQL запитів, налагодження та оптимізації баз даних;

– pgAdmin: це графічний інструмент для управління базами даних PostgreSQL. pgAdmin дозволяє адміністраторам створювати та керувати базами даних, таблицями, індексами, користувачами та іншими об’єктами PostgreSQL;

– MongoDB Compass: це офіційний графічний інтерфейс для управління базами даних MongoDB. MongoDB Compass дозволяє розробникам візуально досліджувати та маніпулювати даними в MongoDB, створювати та виконувати запити, а також аналізувати структуру даних;

– Redis Commander: це графічний інтерфейс для управління базами даних Redis. Redis Commander надає інтерактивний спосіб перегляду та редагування даних, виконання команд Redis та моніторингу стану серверів Redis;

– DataGrip: це універсальний інструмент для роботи з різними базами даних, який надає можливості редагування, виконання запитів, візуальне

моделювання та адміністрування баз даних, таких як MySQL, PostgreSQL, MongoDB та інші.

Ці інструменти допомагають адміністраторам та розробникам ефективно управляти базами даних, виконувати рутинні завдання та забезпечувати надійність та продуктивність системи [13, 14].

2.4 Моделювання архітектури алгоритмів серверної частини

Для серверної частини вебзастосунків розробники використовують різні інструменти та технології, серед яких основні:

– Express.js: це легкий та гнучкий вебфреймворк для Node.js, який дозволяє розробникам швидко створювати серверні застосунки та API;

– Ruby on Rails: це фреймворк на мові програмування Ruby, який дозволяє розробникам створювати потужні та ефективні застосунки шляхом використання конвенцій над конфігурацією;

– ASP.NET: це фреймворк для розробки застосунків на мові програмування C# в середовищі .NET. ASP.NET надає різноманітні інструменти для створення високопродуктивних та масштабованих застосунків;

– Spring Boot: це фреймворк для розробки застосунків на мові програмування Java, який дозволяє розробникам швидко створювати стабільні та надійні застосунки;

– Fastify: це швидкий та ефективний вебфреймворк для Node.js, який надає високу продуктивність та масштабованість для розробки серверних застосунків.

Завдяки цим інструментам створюються потужні та надійні серверні застосунків, які можуть обробляти запити від клієнтів, взаємодіяти з базою даних, а також забезпечувати безпеку та швидкість роботи застосунку.

2.4.1 Алгоритми кошику

Цей алгоритм дозволяє користувачам додавати елементи до корзини замовлення, керуючись їх ідентифікаторами та кількістю. Якщо елемент вже присутній у корзині, кількість його одиниць збільшується, в іншому випадку він додається з початковою кількістю одиниць. Після додавання елемента до корзини дані користувача оновлюються з новою інформацією про корзину, і користувач отримує повідомлення про успішне додавання до корзини. У випадку помилки користувач отримає відповідне повідомлення про помилку у консоль.

Алгоритм додавання страви в кошик:

Крок 1. Отримання ідентифікатора користувача та ідентифікатора елемента меню з запиту.

Крок 2. Пошук користувача в базі даних за допомогою ідентифікатора користувача.

Крок 3. Отримання даних корзини користувача з об’єкта користувача.

Крок 4. Перевірка, чи існує вже елемент меню в корзині користувача за допомогою ідентифікатора елемента меню. Якщо елемент меню ще не присутній у корзині, створюється новий запис з ключем, що відповідає ідентифікатору елемента меню, і значенням 1.

Крок 5. Якщо елемент меню вже присутній у корзині, збільшення кількості цього елемента на одиницю.

Крок 6. Оновлення даних користувача з оновленою корзиною в базі даних.

Крок 7. Повернення успішного повідомлення про додавання елемента до корзини.

Крок 8. У випадку виникнення помилки, її треба належним чином опрацювати та вивести в консоль браузера зрозумілий для користувача текст помилки.

Алгоритм видалення страви з кошику:

Крок 1. Отримання об’єкта користувача з бази даних за допомогою методу «findById» моделі «userModel», використовуючи ідентифікатор користувача, переданий у запиті.

Крок 2. Отримання об’єкта кошика користувача з об’єкта користувача, який містить дані про кошик користувача.

Крок 3. Перевірка, чи кількість предметів в кошику більше 0. Якщо так, то зменшення кількості предметів в кошику на 1.

Крок 4. Оновлення об’єкта користувача у базі даних з новими даними про кошик.

Крок 5. Повернення успішного відповіді з відповідним повідомленням про успішне видалення елемента з кошика.

Крок 6. У випадку виникнення помилки, її треба належним чином опрацювати та вивести в консоль браузера зрозумілий для користувача текст помилки.

Алгоритм отримання даних кошика користувача:

Крок 1. Отримання об’єкта користувача з бази даних за допомогою методу «findById» моделі «userModel», використовуючи ідентифікатор користувача, переданий у запиті.

Крок 2. Отримання об’єкта кошика користувача з об’єкта користувача, який містить дані про кошик користувача.

Крок 3. Повернення успішної відповіді з об’єктом кошика користувача у форматі JSON, з властивістю «success», що має значення «true», та з об’єктом «cartData».

Крок 4. У випадку виникнення помилки, її треба належним чином опрацювати та вивести в консоль браузера зрозумілий для користувача текст помилки.

2.4.2 Алгоритми меню

Алгоритм перегляду всіх страв:

Крок 1. Отримання списку усіх страв з бази даних шляхом виклику методу моделі «foodModel».

Крок 2. Очікування отримання результатів запиту до бази даних за допомогою асинхронного запиту.

Крок 3. Повернення успішного відповіді зі списком усіх страв у форматі JSON, якщо запит успішний.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

Алгоритм видалення страви:

Крок 1. Отримання об’єкта страви з бази даних за допомогою методу «findById» моделі «foodModel», використовуючи ідентифікатор, переданий у запиті.

Крок 2. Видалення зображення страви, що зберігається у каталозі «uploads», за допомогою функції «unlink» модуля «fs» [15].

Крок 3. Видалення об’єкта страви з бази даних за допомогою методу «findByIdAndDelete» моделі «foodModel», використовуючи ідентифікатор, переданий у запиті.

Крок 4. Повернення успішного відповіді з відповідним повідомленням, що страва була успішно видалена.

Крок 5. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

Алгоритм отримання всіх страв з меню:

Крок 1. Отримання списку усіх страв з бази даних шляхом виклику методу «find» моделі «foodModel».

Крок 2. Очікування отримання результатів запиту до бази даних за допомогою асинхронного запиту.

Крок 3. Повернення успішного відповіді зі списком усіх страв у форматі JSON, якщо запит успішний.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

2.4.3 Алгоритми замовлення користувача

Алгоритм розміщення замовлення користувача для вебінтерфейсу:

Крок 1. Визначення URL адреси вебінтерфейсу для передачі результатів операції розміщення замовлення.

Крок 2. Вивід в консоль отриманих даних з тіла запиту.

Крок 3. Спроба створення нового замовлення в базі даних з використанням моделі «orderModel». Це включає збереження інформації про користувача, елементи замовлення, суму та адресу доставки.

Крок 4. Оновлення кошика користувача у базі даних до пустого значення, оскільки замовлення було розміщено і товари були придбані.

Крок 5. Створення масиву який містить елементи замовлення у форматі, придатному для передачі в Stripe API. Кожен елемент містить дані про товар, включаючи назву, ціну та кількість.

Крок 6. Додавання до масиву, з Кроку 5, пункту, що відповідає вартості доставки.

Крок 7. Створення нової сесії оплати через Stripe API за допомогою спеціального методу, передаючи масив, з Кроку 5, режим оплати та URL адреси для перенаправлення після успішної або скасованої оплати.

Крок 8. Повернення успішної відповіді з URL адресою створеної сесії оплати для вебінтерфейсу.

Крок 9. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

Алгоритм перевірки статусу замовлення:

Крок 1. Отримання інформації унікального номеру замовлення і статусу з тіла запиту.

Крок 2. Перевірка, чи була оплата успішною. Якщо статус дорівнює рядку «true», оновлення запису замовлення в базі даних, встановлюючи значення «payment» в «true». Повернення відповіді про успішну оплату.

Крок 3. У випадку невдалої оплати, видалення запису замовлення з бази даних. Повернення відповіді про неуспішну оплату.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення.

Алгоритм для отримання замовлень користувача для відображення на фронтенді:

Крок 1. Отримання ідентифікатора користувача з тіла запиту.

Крок 2. Пошук у базі даних всіх замовлень, які належать користувачу за отриманим ідентифікатором.

Крок 3. Повернення успішної відповіді разом зі списком замовлень користувача у вигляді об’єкту даних.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

Алгоритм для переліку замовлень для панелі адміністратора:

Крок 1. Запит до бази даних для отримання всіх замовлень.

Крок 2. Отримання списку замовлень.

Крок 3. Повернення успішної відповіді разом зі списком замовлень у вигляді об’єкту даних.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

Алгоритм для оновлення статусу замовлення через API:

Крок 1. Отримання інформації про ідентифікатор замовлення та новий статус з запиту.

Крок 2. Оновлення статусу замовлення в базі даних за допомогою ідентифікатора замовлення та нового статусу.

Крок 3. Повернення успішного повідомлення про оновлення статусу.

Крок 4. У випадку виникнення помилки виведення помилки в консоль та повернення відповідного повідомлення про помилку.

2.4.4 Алгоритми авторизації та реєстрації користувача

Алгоритм авторизації користувача:

Крок 1. Отримання електронної пошти та пароля користувача з тіла запиту.

Крок 2. Пошук користувача за його електронною поштою в базі даних.

Крок 3. Перевірка, чи існує користувач з вказаною електронною поштою. Якщо користувач не знайдений, повертаємо відповідне повідомлення про помилку [16].

Крок 4. Порівняння введеного пароля з паролем користувача з бази даних за допомогою «bcrypt».

Крок 5. Якщо паролі співпадають, генеруємо токен для автентифікації користувача.

Крок 6. Повертаємо успішну відповідь разом з токеном автентифікації.

Крок 7. У випадку виникнення помилки виводимо її в консоль та повертаємо відповідне повідомлення про помилку.

Алгоритм реєстрації користувача:

Крок 1. Отримання даних про ім’я, пароль та електронну пошту нового користувача з тіла запиту.

Крок 2. Перевірка, чи користувач із вказаною електронною поштою вже існує в базі даних. Якщо користувач вже існує, повертаємо відповідне повідомлення про помилку

Крок 3. Перевірка формату введеної електронної пошти та надійності паролю. Якщо формат електронної пошти неправильний або пароль надто слабкий, повертаємо відповідне повідомлення про помилку.

Крок 4. Хешування паролю користувача за допомогою «bcrypt» для збереження його в безпечному вигляді в базі даних.

Крок 5. Створення нового об’єкту користувача на основі отриманих даних та збереження його в базі даних.

Крок 6. Генерація токена для автентифікації нового користувача.

Крок 7. Повернення успішної відповіді разом з токеном автентифікації.

Крок 8. У випадку виникнення помилки виводимо її в консоль та повертаємо відповідне повідомлення про помилку.

Алгоритм генерації ключа:

Крок 1. Приймання ідентифікатора користувача як параметра функції.

Крок 2. Використання методу «sign» з бібліотеки «jsonwebtoken» для створення токена.

Крок 3. Утворення токена з використанням секретного ключа, який зберігається під назвою «JWT_SECRET».

Крок 4. Повернення створеного токена.

На цьому розробка алгоритмів для серверної частини завершена.

2.5 UML діаграма класів

UML – це діаграми класів які є потужним інструментом для візуалізації та моделювання об'єктно-орієнтованого дизайну програмного забезпечення. Вони надають графічний спосіб відображення структури класів та їх взаємозв’язків у програмному проєкті, що дозволяє моделювати абстракції, включаючи дані та функції, які обробляють ці дані. Основна мета UML діаграм класів полягає в наданні чіткого огляду структури програми, а також відносин між різними класами. Вони дозволяють моделювати абстракції, які включають дані та функції, які обробляють ці дані.

У випадку ресторанного програмного застосунку, UML діаграма класів допоможе візуалізувати ключові компоненти застосунку, такі як класи

користувачів, страв, замовлень та їх взаємозв’язки. Це забезпечує чітке уявлення про те, як різні частини системи взаємодіють одна з одною, та обов’язків кожного класу. Діаграму класів ресторанного програмного застосунку можна побачити на рисунку 2.4.

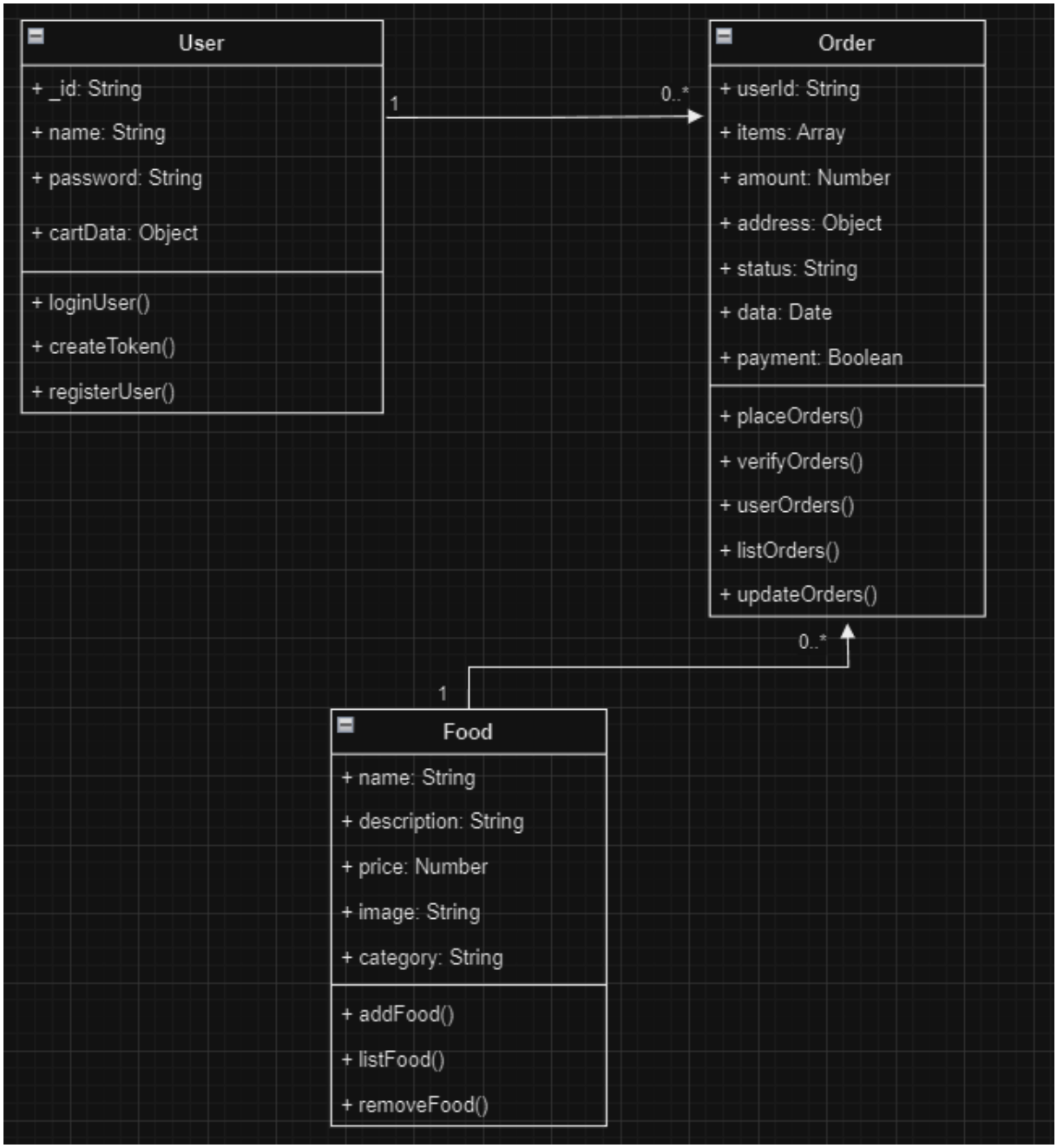

Рисунок 2.4 – UML діаграма класів

3 РОЗРОБКА ІНФОРМАТИЧНОЇ СИСТЕМИ РЕСТОРАНУ З ДОСТАВКОЮ ЇЖІ

3.1 Обґрунтування вибору середовища програмної реалізації

Вибір інструментів для розробки інформаційної системи ресторану базується на потужній комбінації інструментів, які допомагають автоматизувати робочі процеси та забезпечують ефективну розробку.

Для початку, для автоматизації робочих процесів в проєкті було обрано Vite.js [17]. Vite використовує сучасні API браузера для швидкої компіляції коду, забезпечуючи швидкий час створення та миттєві оновлення в браузері. Такий підхід усуває потребу в групувальнику під час розробки, що може значно скоротити час, витрачений на створення та розгортання застосунку. Вбудований сервер розробки у Vite оптимізований для швидкого перезавантаження та гарячої заміни модулів, що дозволить бачити зміни, які вносяться у код у режимі реального часу, без необхідності повного оновлення сторінки. Vite було обрано за його перелік переваг:

– покращений робочий процес розробки. Інноваційний підхід Vite до розробки зовнішнього інтерфейсу забезпечує розробникам більш спрощений досвід розробки. Швидкий час створення, миттєві оновлення в браузері та вбудований сервер розробки з можливістю гарячої заміни модулів забезпечують покращений робочий процес розробки, скорочуючи час, витрачений на ручне тестування, і дозволяючи розробникам зосередитися на написанні коду;

– швидший час створення. Однією з головних переваг використання Vite є значне скорочення часу створення. Інноваційний підхід Vite усуває потребу в групувальнику під час розробки, що сприяє швидкому збиранню та миттєвим оновленням у браузері. Це може заощадити значну кількість часу;

– оптимізовані розміри коду. Використання Vite також може призвести до оптимізації розмірів коду завдяки його відкладеному завантаженню

модулів і функціям дерева. Ці функції дозволяють розробникам зменшувати розмір свого коду, що призводить до підвищення продуктивності для користувачів. Це може бути особливо корисним для великих проєктів і програм із великою кількістю модулів;

– підвищення продуктивності. Швидший час створення, покращений досвід розробки та оптимізовані розміри коду у Vite можуть призвести до підвищення продуктивності розробників. Це може призвести до швидшого виходу на ринок і більш ефективного процесу розробки, дозволяючи вашій команді швидше постачати високоякісні програми;

– підтримує сучасні вебстандарти. Vite розроблено для використання власних модулів ES і сучасних API браузера, що робить його ідеальним вибором для розробників, які прагнуть використовувати найновіші стандарти розробки інтерфейсу. Це гарантує, що ваші проєкти створюються з використанням сучасного коду, який можна підтримувати та масштабується, зменшуючи потребу в майбутніх оновленнях і полегшуючи підтримку ваших програм з часом.

Для розробки користувацького інтерфейсу було обрано бібліотеку React [18]. React являється потужним і популярним JavaScript фреймворком, який дозволяє розробникам легко створювати складні та інтерактивні вебзастосунки. Однією з найбільших переваг React.js є його здатність покращувати взаємодію з користувачем. React.js дозволяє розробникам створювати високоінтерактивні та чуйні інтерфейси користувача, що забезпечує більш плавну та привабливу роботу для користувачів. Бібліотеку React було обрано за деякими перевагами:

– віртуальний DOM і оптимізація. React використовує віртуальний DOM, який є представленням фактичного DOM у пам’яті. Цей віртуальний DOM дозволяє React.js ефективно оновлювати та відображати лише ті компоненти, які були змінені, замість повторного відтворення всієї сторінки. Цей процес, відомий як узгодження, значно покращує продуктивність програм React.js. Крім того, React.js використовує алгоритм розрізнення, щоб

визначити відмінності між віртуальним DOM і фактичним DOM. Це дозволяє React.js оновлювати лише необхідні частини інтерфейсу користувача, що призводить до швидшого рендерингу та покращення загальної продуктивності;

– багаторазові компоненти та повторне використання коду. React.js просуває концепцію повторно використовуваних компонентів, які є самодостатніми модулями, які можна використовувати в різних частинах програми. Цей модульний підхід полегшує створення складних користувальницьких інтерфейсів, розбиваючи їх на менші компоненти, які можна багаторазово використовувати. Можливість багаторазового використання компонентів не тільки економить час розробки, але й покращує зручність підтримки кодової бази. Розробники можуть легко оновити або змінити компонент, не впливаючи на інші частини програми, полегшуючи додавання нових функцій або виправлення помилок;

– ефективний процес розробки. React забезпечує середовище розробки, яке сприяє ефективності та продуктивності. Він пропонує багатий набір інструментів і бібліотек, які полегшують процес розробки, наприклад React Developer Tools, Redux і React Router. Ці інструменти допомагають у налагодженні, управлінні станом і маршрутизації відповідно. Крім того, React має простий та інтуїтивно зрозумілий синтаксис, що полегшує розробникам розуміння та написання коду. Декларативний характер React дозволяє розробникам описувати, як вони хочуть, щоб виглядав інтерфейс користувача, а React піклується про оновлення та рендеринг інтерфейсу користувача відповідно;

– сильна підтримка спільноти. React має велике й активне співтовариство розробників, які постійно роблять внесок у його розвиток і вдосконалення. Ця потужна підтримка спільноти гарантує, що розробники можуть легко знайти рішення своїх проблем і отримати допомогу, коли це необхідно. Спільнота також регулярно випускає оновлення, виправлення помилок і нові функції, що робить React надійною технологією, яка добре

підтримується. Крім того, спільнота React надає численні онлайн-ресурси, навчальні посібники та документацію, що полегшує розробникам вивчення та опанування React.js. Цей величезний фонд знань і підтримки ще більше посилює переваги використання React у веброзробці;

– масштабованість і гнучкість. React дуже масштабований і гнучкий, що робить його придатним для створення програм будь-якого розміру та складності. Його компонентна архітектура дозволяє розробникам легко додавати, видаляти або змінювати компоненти, не впливаючи на решту програми. Ця масштабованість гарантує, що застосунки React можуть розвиватися та адаптуватися до мінливих вимог без значного рефакторингу коду. Крім того, React можна легко інтегрувати з іншими технологіями та фреймворками, такими як Redux для управління станом або Graph QL для отримання даних. Ця гнучкість дозволяє розробникам використовувати сильні сторони різних технологій і створювати потужні та надійні програми;

– легка інтеграція з іншими технологіями. React легко інтегрується з іншими технологіями та фреймворками, що робить його універсальним вибором для веброзробки. Його можна використовувати з такими популярними бібліотеками, як Axios, для надсилання запитів API, або з фреймворками інтерфейсу користувача, як Bootstrap, для стилізації компонентів.

React широко віддають перевагу розробникам за низку переваг у створенні інтерфейсів користувача. Це покращує взаємодію з користувачем, підвищує продуктивність, сприяє повторному використанню коду та забезпечує масштабовану та гнучку розробку. Завдяки надійній підтримці спільноти та документаціям, React є найкращим вибором у веброзробці.

Мова розробки – JavaScript, використовується для написання коду, що забезпечує простоту та зручність в розробці. Також для деякий файлів було використано розширення до JavaScript – JSX. Це розширення синтаксису JavaScript, яке виглядає як XML.

Для розробки серверної частини я обрав Node.js та Express.js. Це було гарним рішенням з кількох причин. По-перше, Node.js – це засіб, який дозволяє використовувати JavaScript для розробки на серверному боці, що спрощує розробку за рахунок можливості використання тієї ж мови програмування як на клієнтській, так і на серверній стороні. Крім того, Node.js відомий своєю високою швидкістю та ефективністю завдяки використанню неблокуючого вводу/виводу. Express.js, у свою чергу, є легким та гнучким фреймворком для Node.js, що дозволяє швидко створювати надійні вебзастосунки та API [19, 20].

Для розробки серверної частини було обрано Node.js у зв’язку з Express.js враховуючи їх переваги:

– висока продуктивність. Найбільшою перевагою Node.js є висока продуктивність для застосунків реального часу. Використання Node.js також сприяє створенню супершвидких застосунків. Фреймворки застосунків, які використовують Node.js, можуть легко виконати кілька завдань. залежно від складності та вартості інтеграції одного процесора, структура IPsec дуже масштабована і дозволяє IPSec обробляти одночасні запити без перезавантаження оперативної пам’яті. Його подійно-зв’язана та неблокуюча операція дозволяє працювати код з прискореною швидкістю, яка має властивості на продуктивність. Node.js дозволяє GoDaddy випередити своїх конкурентів;

– масштабованість для сучасних застосунків. Node.js – це нова технологія, і пов’язана з нею компоненти довели свою ефективність у вирішенні завдань промисловості. Вона включає в себе широкий спектр функцій, таких як модулі кластерів. Вона дозволяє навантажувальний баланс на кількох ядрах ЦП, сприяючи досягненню результатів потрібних через менші модулі з меншим споживанням енергії ЦП. Node.js також використовує механізм неблокуючої петлі подій, що забезпечує високу масштабованість і дозволяє серверу легко обробляти запити;

– розробка вебсервісів. Переваги впровадження Node.js на вебсервісах варіюють від великої кількості в конкурентній компанії та її IT- стратегії. Приклади можуть розмахуватися від технічної масштабованості, швидкості та продуктивності до обмеження застосування. Рішення на поточному етапі початку, як швидко ви вийдете на ринок. Java домінувала на вебсторінках як популярна мова програмування для клієнтів. JavaScript був лише ідеєю, поки не був випущений Node.js. За допомогою переваги Node.js розробка дуже проста, ефективна з точки зору вартості та суперефективна, якщо встановлено JavaScript;

– гнучкість. Якщо ви вносите зміни у Node.js, змінюється лише цей вузол. Де інші середовища чи структура запуску можуть вимагати внесення змін для досягнення кінцевого програмування, для нього потрібна лише змінена вузла. І найкраще в тому, що коли ви виконуєте JSON з Node.js, ви можете легко обмінюватися даними між клієнтськими серверами та вебсервером.

Переваги завдяки яким було обрано Express.js:

– швидка розробка. Фреймворк Express.js дозволяє використовувати ту саму мову, якою є JavaScript, як на серверній, так і на передній частині. Це дає розробникам JavaScript можливість стати повним стеком. У результаті процес розробки набагато швидший і легший, оскільки одна особа може керувати як презентацією, так і рівнем доступу до даних;

– спільнота з відкритим кодом. Express.js є одним із найбільш підтримуваних фреймворків Node.js. Він має спільноту з відкритим кодом, тому код завжди переглядається та вдосконалюється;

– обробка запитів введення/виведення. Express JS – чудовий вибір для програм, які обробляють багато запитів і сповіщень від користувачів. Node.js особливо добре підходить для написання систем, які зберігають увесь свій стан у пам’яті. Їм не потрібно виносити назовні проблеми розподіленої системи. Як наслідок, системи можуть бути більш доступними, і вони можуть швидше відповідати на запити, усуваючи читання/запис і серіалізацію стану в

базі даних. Якщо підсумовуючи, Node.js разом із Express.js здатні підтримувати тисячі одночасних дій.

Для розробки бази даних для ресторанного сервісу я обрав MongoDB. Ця база даних приваблює деякими перевагами, особливо для вебзастосунків, що розробляються на JavaScript.

MongoDB відома своєю гнучкістю схеми даних, що дозволяє змінювати структуру даних без міграцій, що особливо важливо для постійно розвиваючихся проєктів.

Обираючи MongoDB можна отримати гнучкість схеми даних, простоту використання та можливість ефективно масштабувати застосунок з ростом обсягу даних.

Така комбінація інструментів дозволить забезпечити швидку та ефективну розробку, а також забезпечить потужність та гнучкість для подальшого розвитку проєкту.

3.2 Створення моделей в MongoDB

Для створення схем моделей MongoDB в коді потрібно визначити структуру даних для кожного типу об’єкта, який буде зберігатися в базі даних. Кожна модель відповідає певній колекції в базі даних, а кожному полю моделі відповідає поле в документі цієї колекції [21].

У системі ресторану було створено три моделі в базі даних, які допомагають організувати та зберігати інформацію про користувачів, страви та замовлення. Кожна модель описана окремо у власному файлі в папці для моделей, що сприяє більшій чіткості та структурованості коду.

Ці моделі допомагають системі ресторану ефективно керувати даними та виконувати різноманітні функції, необхідні для її роботи, такі як зберігання профілю користувача, опис страви та інформація про замовлення. Вони визначають основну структуру даних і забезпечують можливість звертатися до цих даних з інших частин програмного забезпечення системи.

Таблиця 3.1 представляє модель таблиці, яка буде створена у MongoDB для зберігання даних про страву. Ця модель визначає структуру даних, що включає поля, необхідні для опису кожної страви. Вона містить такі поля: назва страви, опис, ціна, зображення, та категорія в меню [22, 23].

Таблиця 3.1 – Таблиця моделі «foodModel»

Назва поля	**Тип поля**
name	String
description	String
price	Number
image	String
category	String

Таблиця 3.2 визначає модель таблиці для зберігання даних про замовлення у MongoDB. Вона містить різні поля, щоб описати кожне замовлення у ресторанній системі. Серед цих полів є ідентифікатор користувача, список елементів у замовленні, загальна сума замовлення, адреса доставки, статус замовлення, дата створення замовлення та підтвердження оплати.

Таблиця 3.2 – Таблиця моделі «orderModel»

Назва поля	**Тип поля**
userId	String
items	Array
amount	Number
address	Object
status	String
date	Date
payment	Boolean

Таблиця 3.3 визначає модель таблиці для зберігання даних користувачів у MongoDB. Вона містить наступні поля: ім’я користувача, електронна пошта, пароль та дані корзини. Для кожного користувача зберігаються їх основні особисті дані, такі як ім’я та адреса електронної пошти, а також захешований

пароль для безпеки. Поле «cartData» зберігає дані про корзину користувача, яка містить обрані ним товари та їх кількість.

Таблиця 3.3 – Таблиця моделі «userModel»

Назва поля	Тип поля
name	String
email	String
password	String
cartData	Object

Ця структура дозволяє зручно та ефективно зберігати та оновлювати інформацію про користувачів та їх активність на платформі.

3.3 Демонстрація роботи програми

Під час демонстрації роботи програми буде показано всі основні функції та можливості, які вона пропонує . В процесі демонстрації будуть функції якими можуть користуватися як звичайний користувач так і процеси які виконує адміністратор. Усі результаті будуть продемонстровані у виді скриншотів програми [24].

Коли користувач заходить на сторінку ресторану, він потрапляє на головну сторінку, як на рисунку 3.1.

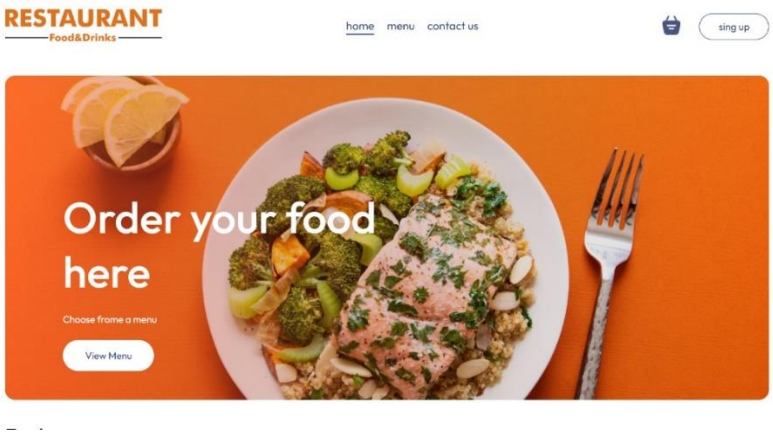

Рисунок 3.1 – Демонстрація головної сторінки

Оскільки користувач не має облікового запису, йому потрібно його створити. Для цього на навігаційній панелі розміщується кнопка «sign up». Натиснувши на неї, він побачить невелике вікно поверх екрану, яке дозволить йому авторизуватися, але оскільки користувач ще не має облікового запису, він повинен натиснути на виділений текст «Click here», модальне вікно зміниться на форму реєстрації, як на рисунку 3.2. Тоді користувач має ввести необхідну інформацію: ім’я, електронну пошту, але яка ще не використовувалась в застосунку, пароль, приклад я навів також на рисунку 3.1. Якщо реєстрація сталася вірно, то користувача буде автоматично введено в його обліковий запис.

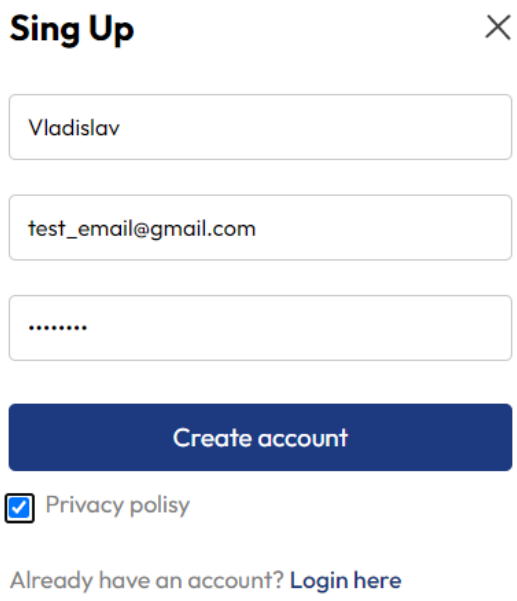

Рисунок 3.2 – Демонстрація реєстрації користувача

На рисунку 3.3, продемонстровано пусту сторінку заказів користувача, це зроблено для того, щоб можна було побачити, що це новий обліковий запис і він не містить ніяких замовлень. Після цього користувач може знов перейти на головну сторінку він може перейти до страв прогорнувши сторінку вниз, або на шапці сторінки може натиснути на поле «menu» і сторінка автоматично перенесе користувача на поле з меню, яке продемонстровано на рисунку 3.4.

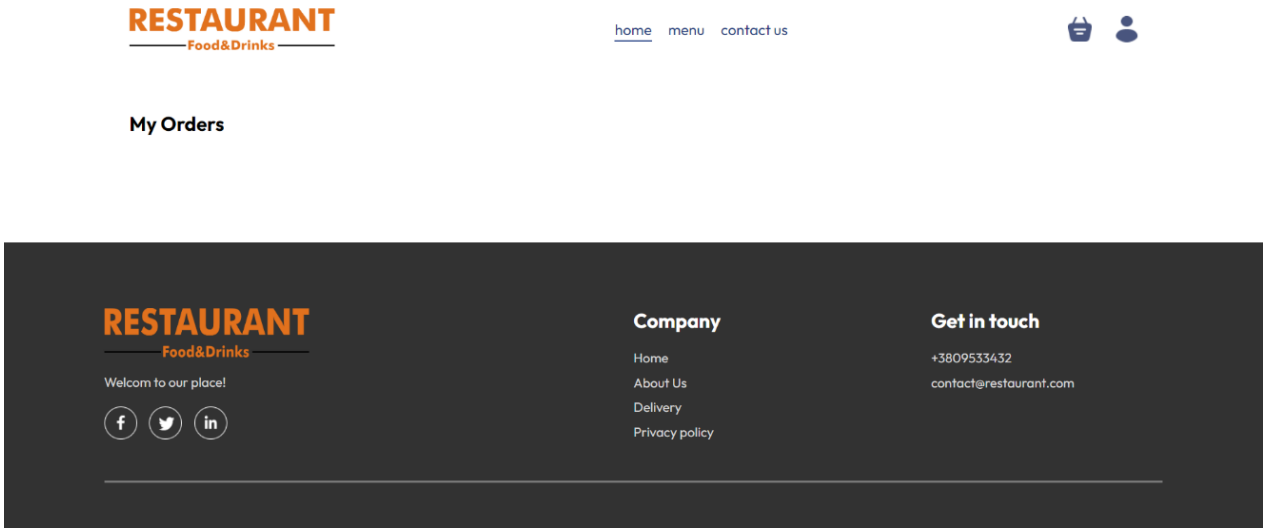

Рисунок 3.3 – Демонстрація сторінки замовлень користувача

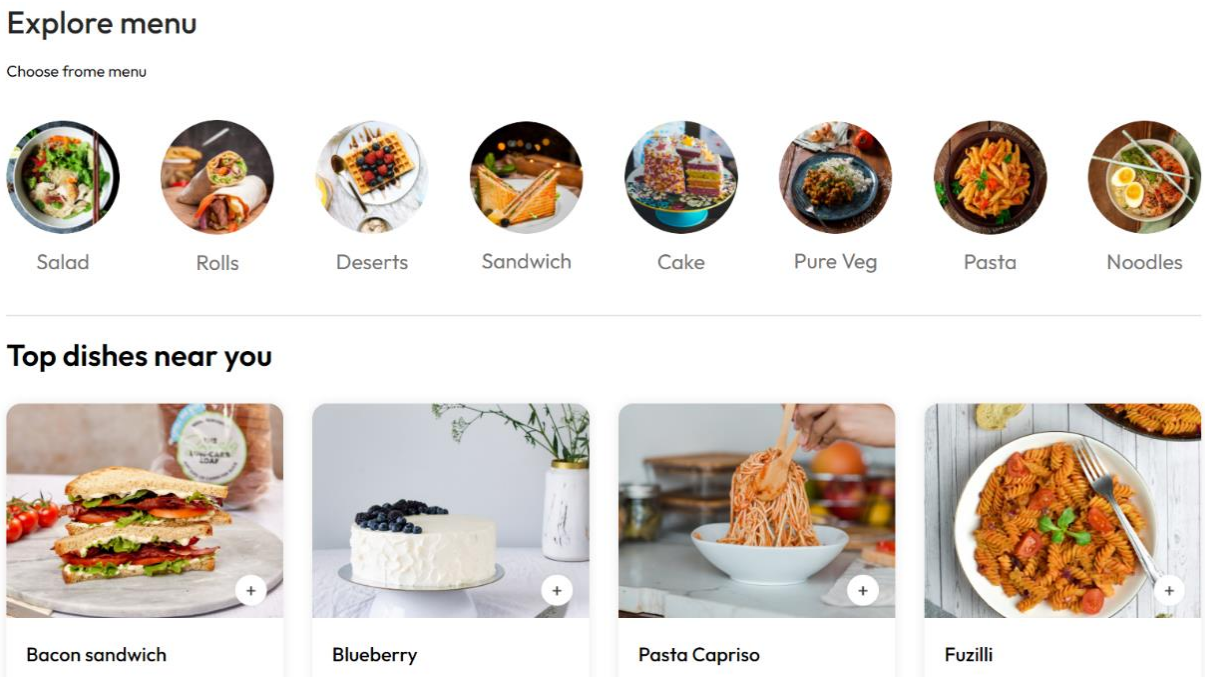

Рисунок 3.4 – Демонстрація переходу до частини з меню

На рисунку 3.5, продемонстровано як користувач обрав декілька страв і їх кількість [25, 26]. Це означає, що обрані страви і їх кількість вже знаходяться в кошику. Перейти до кошику можна за допомогою кнопки в шапці сайту.

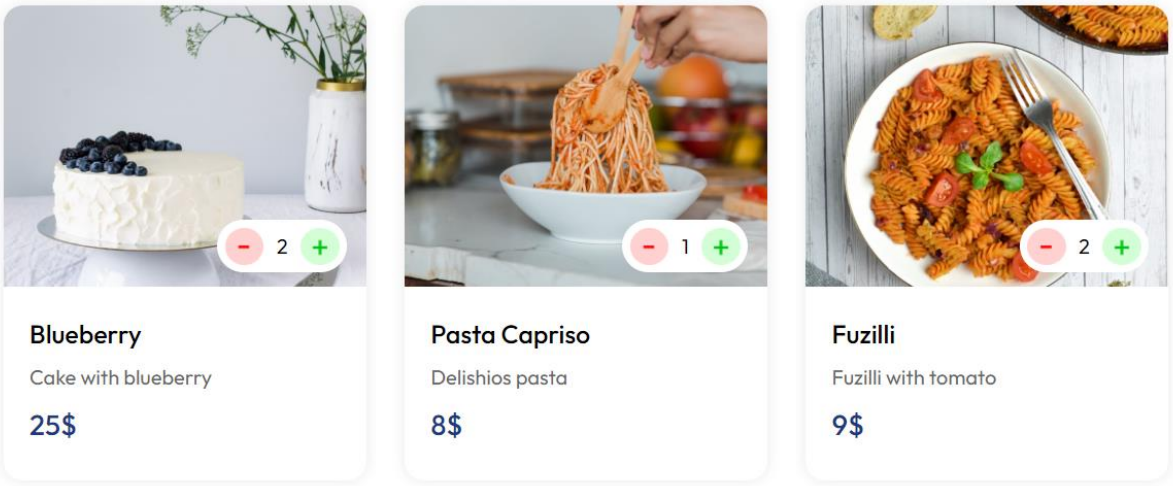

Рисунок 3.5 – Демонстрація додавання страв в кошик

На рисунку 3.6, продемонстровано як користувач знаходиться в кошику. На цій сторінці він може побачити список страв які були обрані, ціну за одну страву, загальну кількість кожної зі страв та їх ціну рахуючи з кожної окремої позиції [27, 28]. Під списком знаходиться більш спрощена інформація. Якщо користувачу все підходить, то він може натиснути кнопку «Proceed to checkout» і перейти на сторінку підтвердження замовлення. Якщо якась страва зайва – то він може видалити страву кнопкою «x».

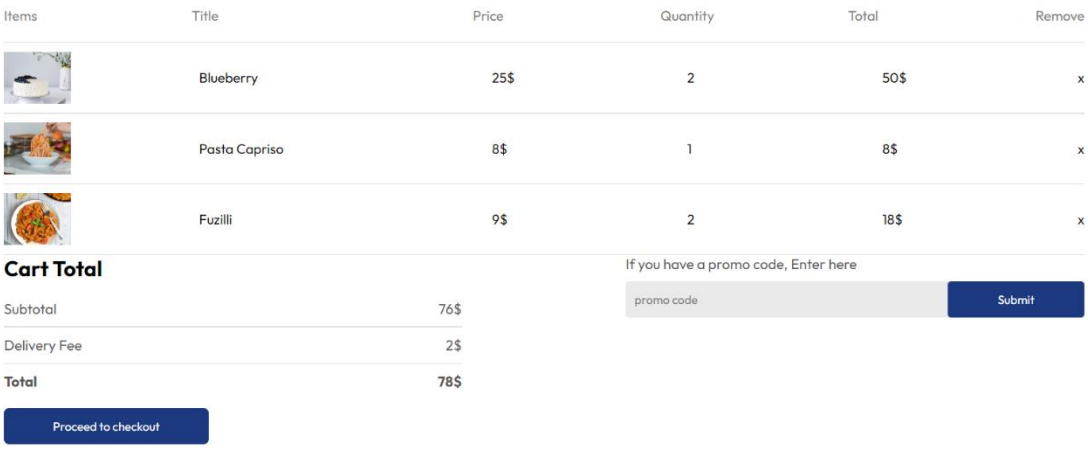

Рисунок 3.6 – Демонстрація кошику користувача

На рисунку 3.7, продемонстровано сторінку підтвердження замовлення на якій користувач може заповнити поля:

– «First Name»: користувач вводить своє, або ім’я того хто буде приймати замовлення;

– «Second Name»: користувач вводить своє прізвище, або прізвище того хто буде приймати замовлення;

– «Email»: користувач вводить свою пошту, або іншу пошту якою він може користуватися;

– «Region»: користувач вводить область в яку він хоче здійснити замовлення, якщо ресторан здійснює доставку в цій області;

– «Street»: користувач вводить вулицю на яку він здійснює замовлення;

– «House»: користувач вводить будинок на який він здійснює замовлення;

– «Phone number»: користувач вводить телефон на який в разі якихось проблем або уточнюючих запитань користувачу буде здійснений дзвінок від ресторану;

– «Apartment number»: користувач вводить номер квартири на яку він здійснює замовлення.

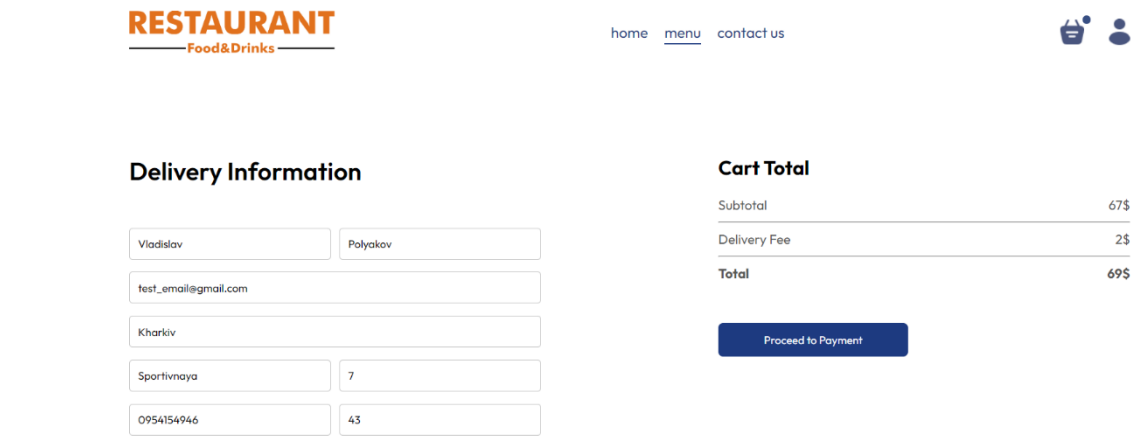

Рисунок 3.7 – Демонстрація заповнення інформації для замовлення

На рисунку 3.8, продемонстровано сторінку оплати на якій користувач може оплатити замовлення. Оскільки в цій програмі реалізовано тестовий

рахунок, номер тестової карти можна взяти на сторінці цієї платіжної системи, дату, якщо вона дійсна, і CVV код можна вводити будь-яку [29].

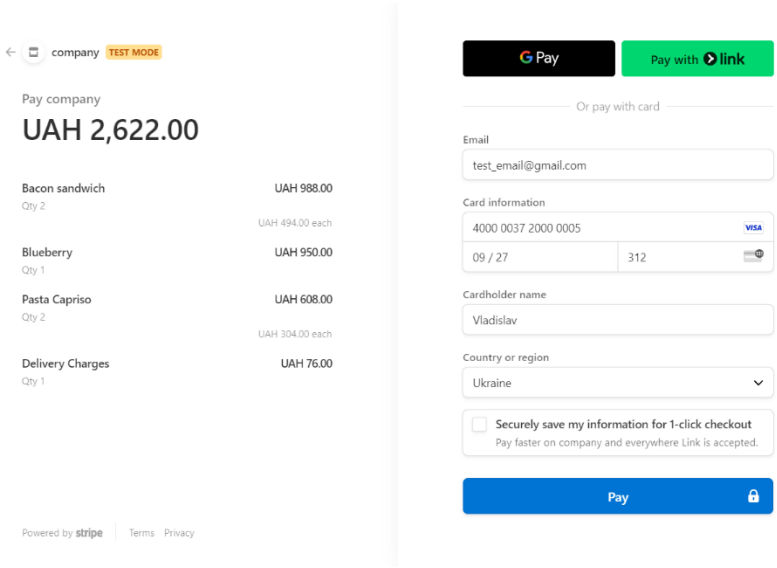

Рисунок 3.8 – Демонстрація платіжної системи

Якщо оплата пройшла вдало, користувача буде відправлено на сторінку з його замовленнями, як на рисунку 3.9, там він може побачити всі свої замовлення. На цій сторінці буде перелік страв та їх кількість, ціна яку заплатив користувач, статус замовлення і кнопку завдяки якій користувач може оновити статус замовлення.

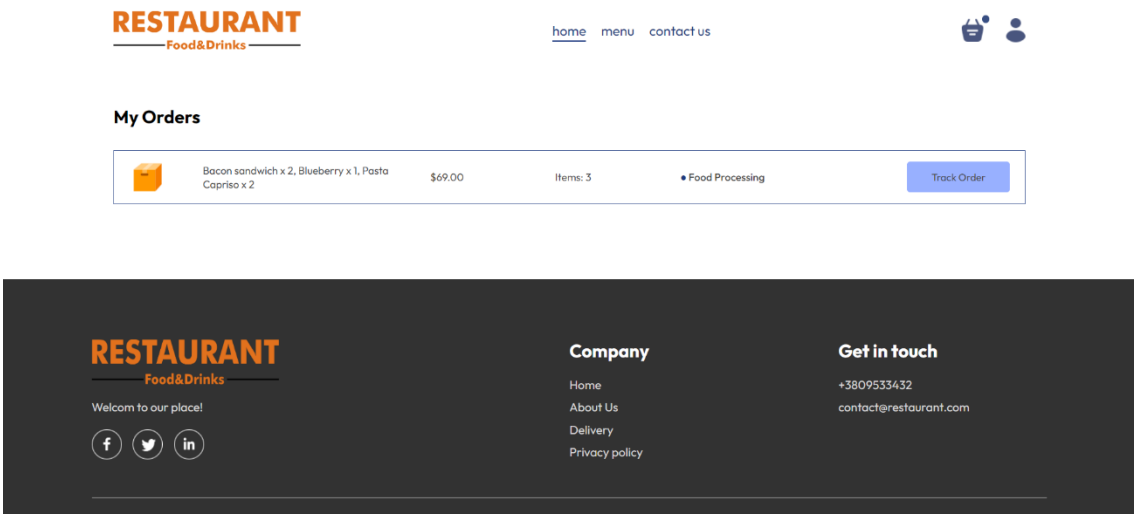

Рисунок 3.9 – Демонстрація замовлення користувача

На рисунку 3.10, продемонстровано замовлення користувача, яке змінило свій статус після того як адміністратор змінив статус замовлення на «Out for delivery».

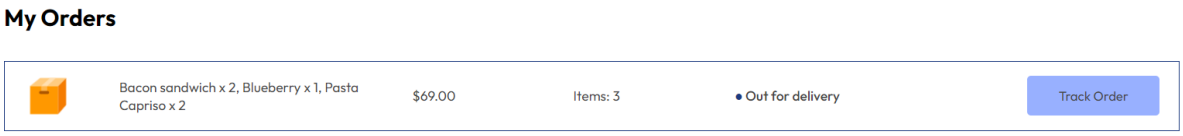

Рисунок 3.10 – Демонстрація оновленого замовлення користувача

Демонстрація основних функцій користувача завершено.

Сторінка адміністратора відкривається окремо від клієнтської частини і у адміністратора є 3 активних вікна в яких він може працювати. Сторінка адміністратора на якій він може переглядати усі замовлення, усіх користувачів, а також змінювати статус замовлення за розрахунком стадії готовності замовлення, відображено на рисунку 3.11.

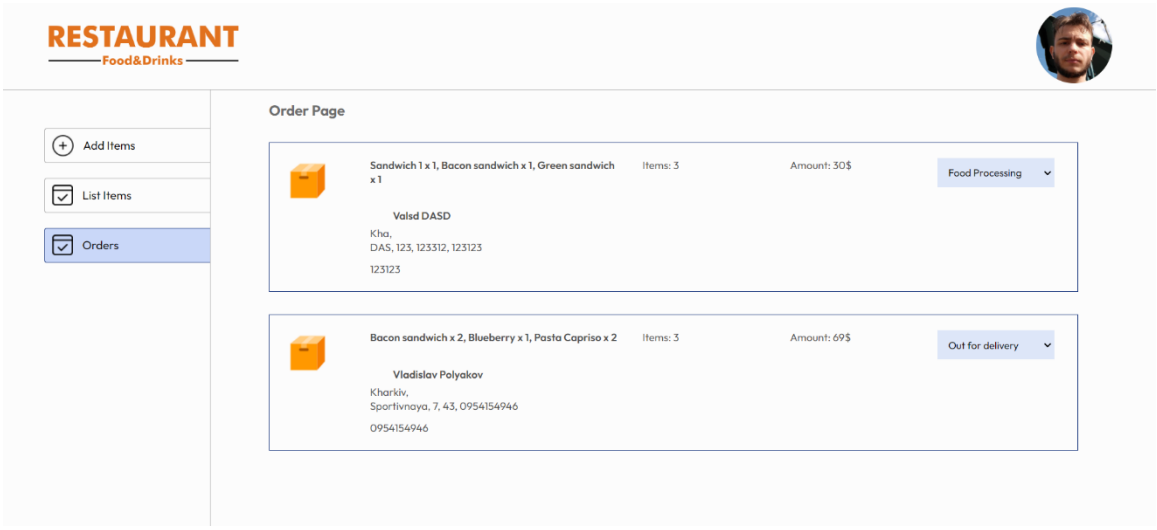

Рисунок 3.11 – Демонстрація вікна адміністратора з усіма замовленнями

На рисунку 3.12 можна побачити вікно, на якому адміністратор може додати в меню страву. Заповнивши для цього декілька форм:

– «Upload Image»: адміністратор натискає на форму після чого в нього відкривається провідник в якому він обирає потрібну фотографію для страви яку він додає;

– «Product name»: адміністратор вводить назву страви;

– «Product description»: адміністратор вводить опис страви;

– «Product category»: адміністратор обирає одне з 8 категорій страви: «Pizza», «Rolls», «Salad», «Pasta», «Noodles», «Sandwich», «Deserts», «Cake»;

– «Product price»: адміністратор вводить ціну для страви в американських доларах.

Після того як адміністратор ввів усю необхідну інформацію про страву, він може натиснути на кнопку «Submit». Якщо вся інформація введена правильно – користувач побачить очищені поля і повідомлення в верхньому правому куті екрану, як продемонстровано на рисунку 3.13.

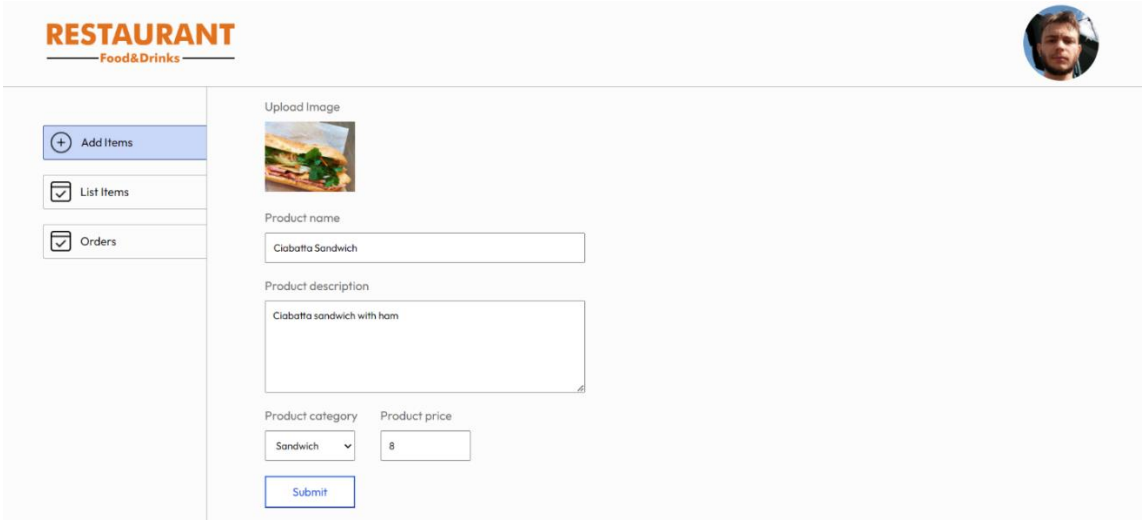

Рисунок 3.12 – Демонстрація заповнення інформації про нову страву

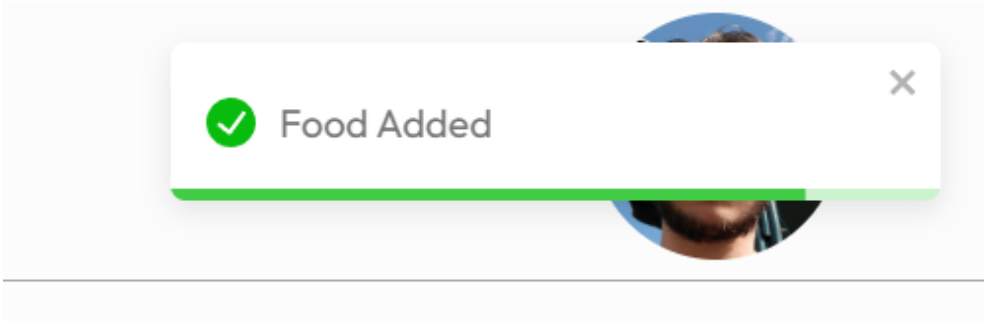

Рисунок 3.13 – Демонстрація вдалого додавання нової страви

Якщо після повідомлення про те що страва додана перейти на головній сторінці буде відображена нова страва. На рисунку 3.14 продемонстровано відсортований список всіх страв по категорії «Sandwich». Як можна бачити, страва відображається на головній сторінці.

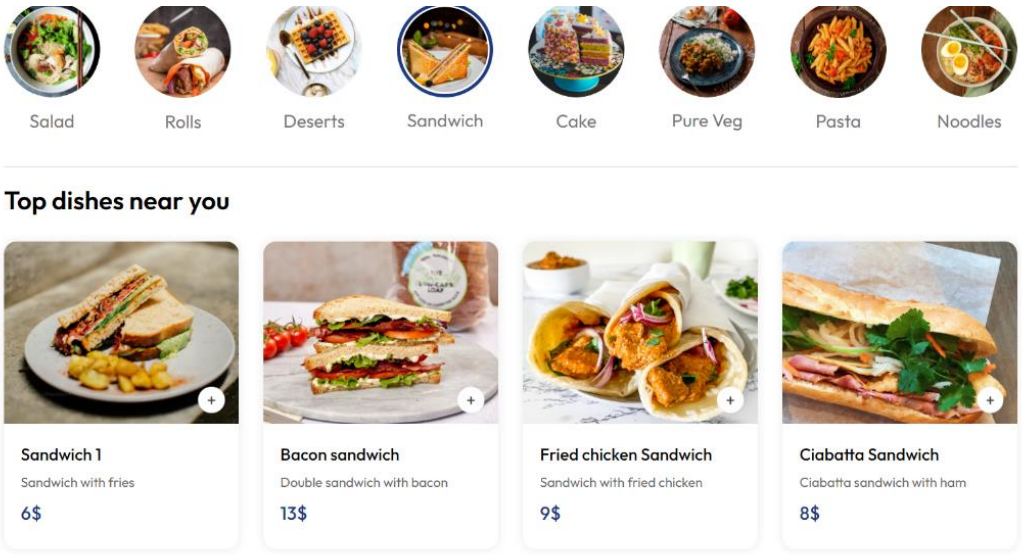

Рисунок 3.14 – Демонстрація зміни меню

На рисунку 3.15 продемонстровано вікно з списком всіх страв які є в БД. У цьому вікні адміністратор може переглянути увесь список страв а також видалити з БД якусь позицію.

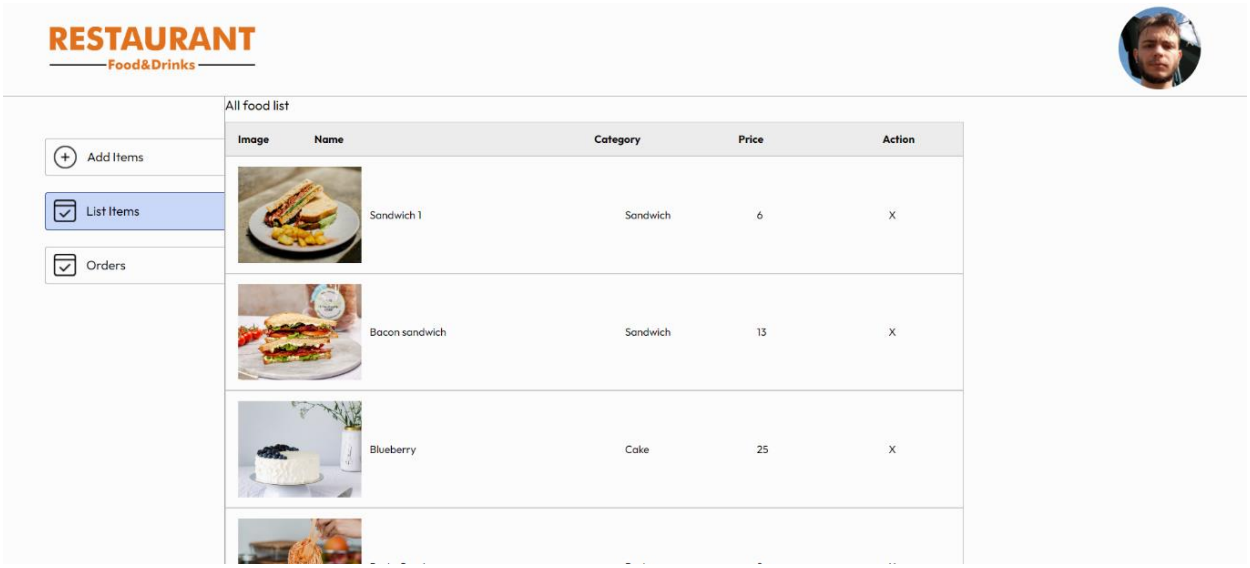

Рисунок 3.15 – Демонстрація списку меню ресторану

На рисунку 3.16 продемонстровано видалення однієї з позицій. Коли користувач натиснув на «x», можна побачити повідомлення про вдале вилучення зі списку «Sandwich 1». Результат цієї функції можна побачити на рисунку 3.17 і зрівняти з рисунком 3.14. Останнє вікно продемонстроване на рисунку 3.16, у цьому вікні можна побачити список всіх замовлень від усіх користувачів.

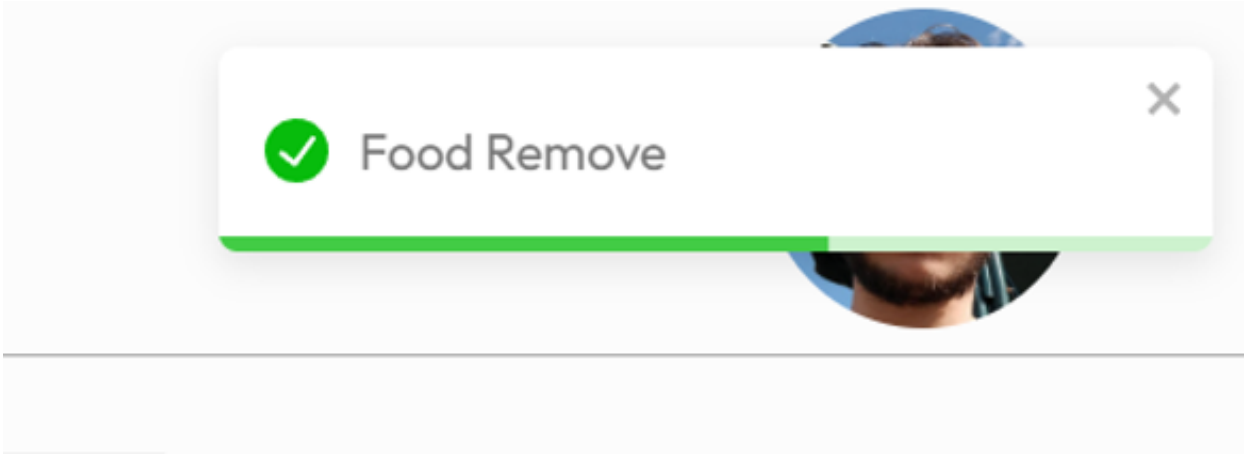

Рисунок 3.16 – Демонстрація вдалого видалення 1 позиції в меню

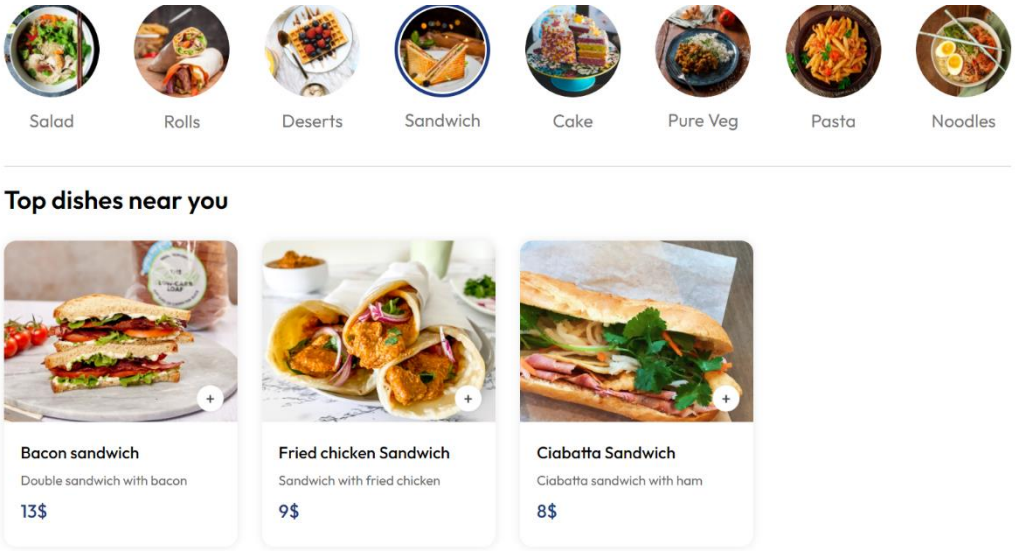

Рисунок 3.17 – Демонстрація зміни меню після видалення

На цій демонстрації було показано всі основні функції користувача і адміністратора ресторанної системи.

3.4 Тестування програми

Якщо користувач при авторизації ввів електронну пошту, яка не була використана при реєстрації, як на рисунку 3.18, тоді користувач побачить повідомлення «User Doesnt exist» [30].

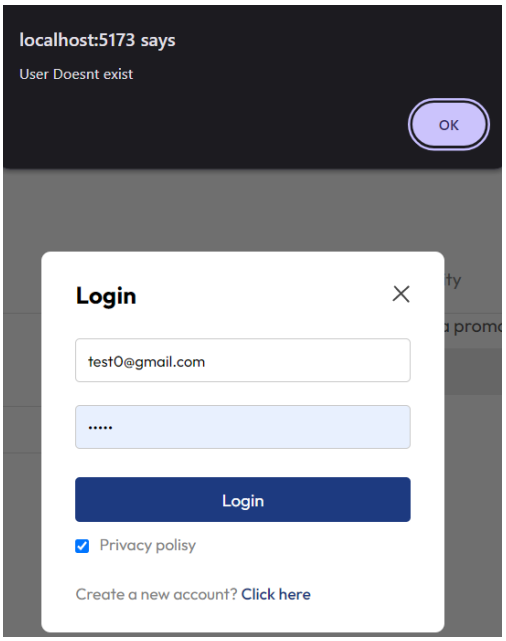

Рисунок 3.18 – Помилка при введені неіснуючої пошти

Якщо користувач ввів невірний пароль, тоді повідомлення зміниться на «Wrong password», як на рисунку 3.19.

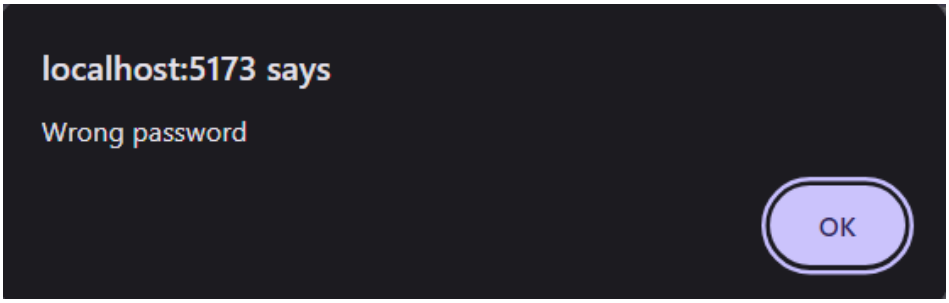

Рисунок 3.19 – Повідомлення помилки при введені неіснуючої пошти

Якщо при реєстрації користувач намагається створити новий обліковий запис, але електронна пошта вже використовувалась раніше, тоді користувач побачить повідомлення «User alredy exists», як на рисунку 3.20.

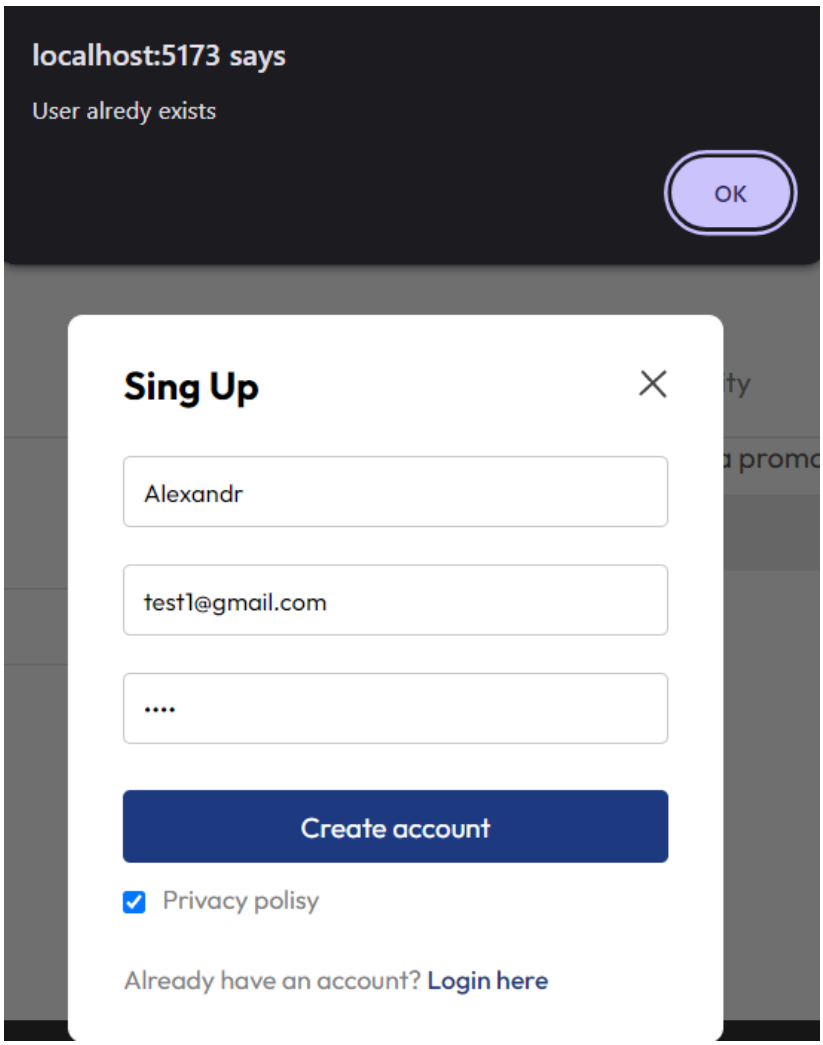

Рисунок 3.20 – Повідомлення помилки про вже існуючого користувача

Якщо користувач при реєстрації користувач намагається ввести в поле електронної пошти, то буде перевірка від бібліотеки «validatorjs». Він перевіряє поле яке повинно бути відформатовано як адрес електронної пошти. Тоді користувач побачить повідомлення, як на рисунку 3.21.

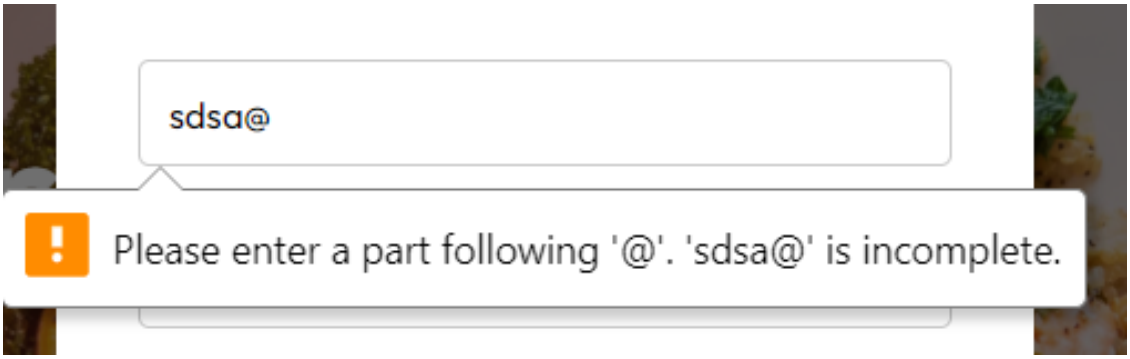

Рисунок 3.21 – Помилка коректності вводу електронної пошти

Також є ще одна перевірка на коректність вводу електронної пошти, але вже вона впроваджена окремо в коді функції програми, вона також перевіряє на коректність і якщо ця помилка спрацьовує – користувач бачить повідомлення, як на рисунку 3.22.

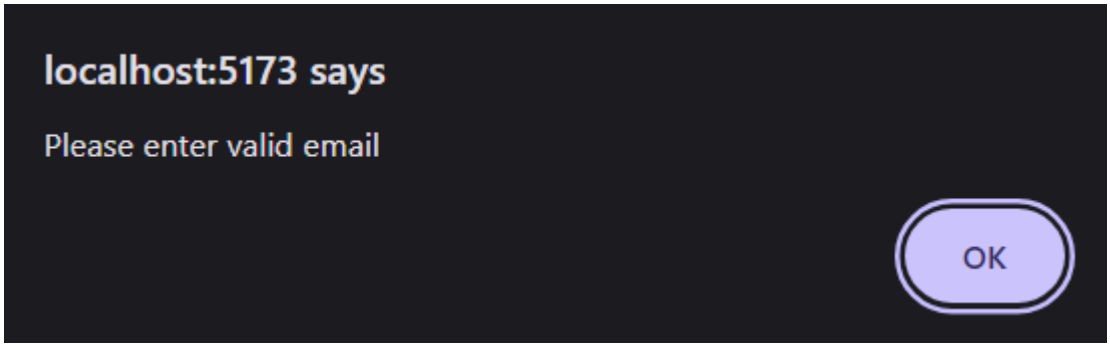

Рисунок 3.22 – Повідомлення про невалідний ввод електронної пошти

Якщо користувач при реєстрації вводить замалий пароль, тоді користувач побачить інше повідомлення «Please enter a strong password», як на рисунку 3.23. Пароль має складати не менш ніж 8 символів.

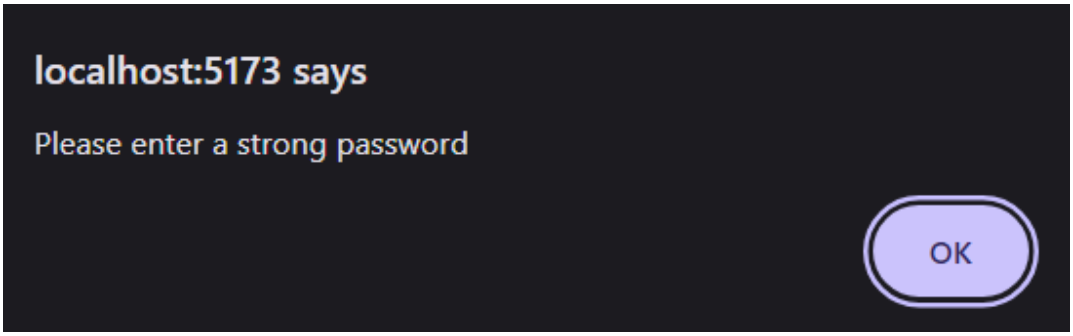

Рисунок 3.23 – Повідомлення про слабкий пароль

Якщо користувач намагається увійти в свій обліковий запис або зареєструватися, при цьому не відмітивши поле «Privacy polisy», як продемонстровано на рисунку 3.24. Тоді користувач побачить повідомлення про обов’язкове поле.

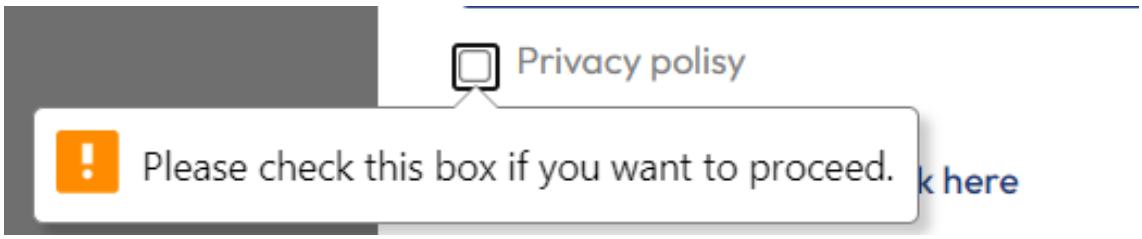

Рисунок 3.24 – Повідомлення про обов’язкове поле

Якщо користувач при заповненні інформації не заповнить кожне з полей, тоді з’явиться повідомлення на тому полі, або якщо цих полей декілька, як на рисунку 3.25, що поле пусте і має буди заповнено.

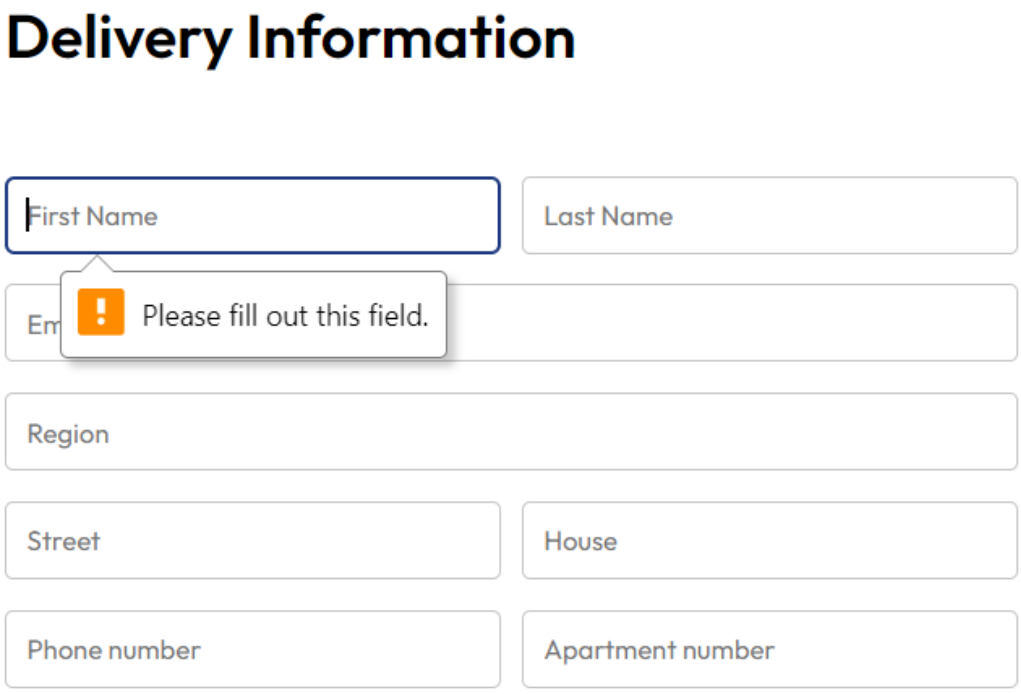

Рисунок 3.25 – Повідомлення про пусте поле

Коли користувач перебуває на сторінці оплати і при вводі банківської карти він вводить невірний номер карти, тоді поле номеру карти покаже помилку «Your card number is invalid». Приклад цієї помилки продемонстровано на рисунку 3.26.

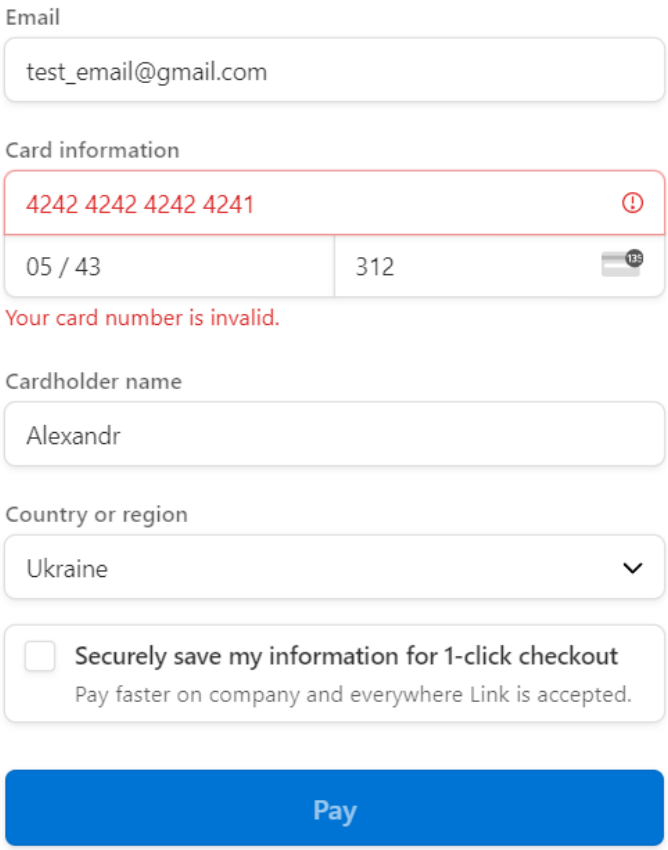

Рисунок 3.26 – Помилка невірної карти

Якщо користувач вводить термін дії карти, яка вже закінчилась, тоді поле стане червоним і з’явиться повідомлення «Your card`s expiration year is in the past», як на рисунку 3.27.

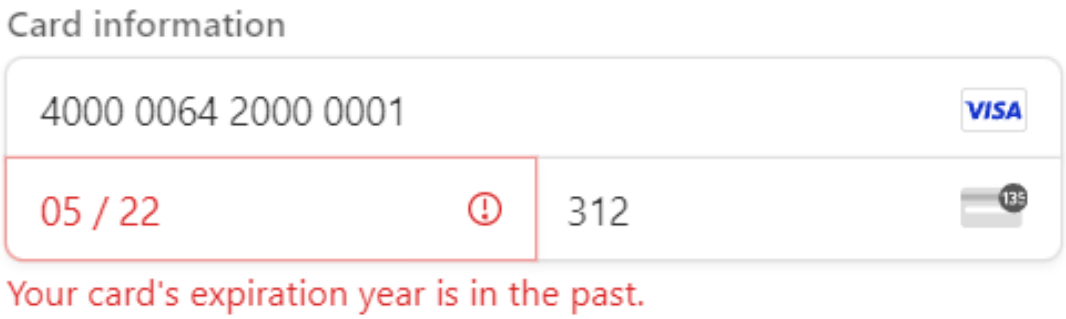

Рисунок 3.27 – Помилка некоректної дати

На цьому тестування програми завершено. Процес тестування був спрямований на перевірку всіх функцій застосунку на наявність помилок та

аналіз їх роботи в різних сценаріях використання. Кожна функція була ретельно перевірена для впевненості в її працездатності та відповідності вимогам.

У разі виявлення помилок під час тестування, в коді застосунку було належним чином прописано вивід повідомлень про їх наявність. Це дозволяє оперативно виявляти та виправляти будь-які проблеми, які можуть виникнути в процесі використання застосунку. Крім того, детально задокументовані виводи про помилки допомогли швидше розуміти та усувати проблеми.

В результаті тестування було впевнено, що всі функції застосунку працюють стабільно, а всі виявлені помилки були успішно виправлені.

ВИСНОВКИ

У рамках кваліфікаційної роботи було реалізовано інформаційну систему ресторану з доставкою їжі. Для досягнення цієї мети було проаналізовано і розглянуто інші ресторанні сервіси і сервіси з доставки їжі, та результатами аналізу було обрано загальне бачення роботи системи, а також функції які повинні бути реалізовані в проєкті.

Були виконані всі поставлені задачі, а саме:

– розглянуто сучасний стан доставки їжі та ресторанної галузі в Україні та світі, особливості ринку їжі, а також вплив пандемії COVID-19 на ресторанний бізнес;

– проаналізовано роль технологій у розвитку сучасних ресторанних сервісів, зосередившись на впровадженні онлайн-платформ для замовлення їжі, відслідковуванні замовлень;

– розглянуто і проаналізовано варіанти розробки: клієнтської частини, серверної частина, а також баз даних;

– розроблені алгоритми функцій які впроваджені в систему;

– розроблено клієнтську і серверну частину, а також окремо створена частина адміністратора;

– проведено тестування системи на правильність роботи системи.

Даний інформаційний застосунок має працювати для демонстрації можливостей роботи інформаційної системи ресторану з доставкою їжі. Враховуючи що даний застосунок буде модернізований та розширений до необхідного результату.

Ця система незважаючи на проведені тести може нести в собі баги або якісь неточності які не були враховані або розглянуті мною.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mafia: Italian and Japanese restaurant. URL: https://mafia.ua/ua/kharkov (дата звернення 19.04.2024).

2. Япошка: Korean and Japanese restaurant. URL: https://yaposhka.com.ua/ua/ (дата звернення 19.04.2024).

3. Georgia: Georgian restaurant. URL: https://georgia.ua/ (дата звернення 19.04.2024).

4. BizMag: website of list of top rated delivery company. URL: https://bizmag.com.ua/kompanii-zoseredzhuiutsia-na-shvydkii-dostavtsi/ (дата звернення 19.04.2024).

5. Stfalcon: website with delivery growth trend. URL: https://stfalcon.com/uk/blog/post/on-demand-food-delivery-apps (дата звернення 19.04.2024).

6. Google Maps: web service of maps. URL: https://www.google.com.ua/maps/@50.4851493,30.4721233,14z?hl=ru&entry=ttu (дата звернення 21.04.2024).

7. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images.

8. Draw io: web service for creating use case diagrams. URL: https://app.diagrams.net/ (дата звернення 09.05.2024).

9. Ткачов, В. А. (2024). Дослідження та порівняльний аналіз JavaScript фреймворків для розроблення вебзастосунків.

10. Devoteam: website with list of benefits of Angular. URL: https://www.devoteam.com/expert-view/angular-front-end-framework/ (дата звернення 30.04.2024).

11. DARADKEH, Y. I., GOROKHOVATSKYI, V., TVOROSHENKO, I., & ZEGHID, M. Tools for Fast Metric Data Search in Structural Methods for Image Classification.

12. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

13. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, IEEE Access, vol. 11, pp. 126938-126949.

14. Гороховатський, В. О., & Творошенко, І. С. (2022). Аналіз багатовимірних даних за описом у формі множини компонент.

15. Захаров, Є. М. (2024). Дослідження методів онлайн достовірної кластеризації даних.

16. Mashtalir, S. V., & Nikolenko, O. V. (2023). Data preprocessing and tokenization techniques for technical Ukrainian texts. *Applied Aspects of Information Technology*, *3*(6), 318-326.

17. Vite.js: documentation to a development environment in JavaScript. URL: https://vitejs.dev/ (дата звернення 07.05.2024).

18. React: documentation to JavaScript framework. URL: https://react.dev/ (дата звернення 07.05.2024).

19. Node.js: documentation to open-source, cross-platform JavaScript runtime environment. URL: https://nodejs.org/en (дата звернення 07.05.2024).

20. Apico: website with list of benefits of Express.js. URL: https://apiko.com/blog/express-mobile-app-development/ (дата звернення 30.04.2024).

21. MongoDB: data base service. URL: https://www.mongodb.com/ (дата звернення 09.05.2024).

22. Lyashenko, V., Kobylin, O., & Selevko, O. (2020). Wavelet analysis and contrast modification in the study of cell structures images.

23. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, 33(1), pp. 113-125.

24. Assets photo for this project was taken from google drive link from description under this video. URL: https://www.youtube.com/watch?v=DBMPXJJfQEA&t=35336s (дата звернення 01.05.2024).

25. Lyashenko, V., Kobylin, O., Ryazantsev, O., Ryazantsev, I., Barbaruk, V., & Zhychenko, Y. (2020). General Ideology of Analysis Digital Medical Images in RGB Format.

26. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

27. Daradkeh, Y., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of Images Based on a System of Hierarchical Features. *Computers, Materials & Continua*, 72(1).

28. O. Yakovleva, & K. Nikolaieva (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

29. Stripe: payment system api for developers. URL: https://docs.stripe.com/development (дата звернення 09.05.2024).

30. Tvoroshenko, I. S., & Maksimenko, H. (2021). To the question of analysis of existing mechanisms of web application testing.