Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
# Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для продажу кондитерських виробів виготовлених на замовлення. Backend.
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-8

Фарафонов Є.Ю.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доцент кафедри ПІ Кравець Н.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

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

**ЗАВДАННЯ**
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Фарафонову Євгенію Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для продажу кондитерських виробів виготовлених на замовлення. Backend.
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10.06.24
3. Вихідні дані до роботи Розробити backend частину застосуноку для замовлення кондитерських виробів, за допомогою якого можна буде замовляти як і готові продукти, так і створювати власні, мова програмування JavaScript за допомогою бібліотеки Express.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

# КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної галузі | 20.05.2024 | *виконано* |
| 2 | Створення специфікації ПЗ | 22.05.2024–23.05.2024 | *виконано* |
| 3 | Проектування ПЗ | 24.05.2024–25.05.2024 | *виконано* |
| 4 | Розробка ПЗ | 26.05.2024–01.06.2024 | *виконано* |
| 5 | Тестування ПЗ | 02.06.2024–05.06.2024 | *виконано* |
| 6 | Оформлення пояснювальної записки | 05.06.2024–07.06.2024 | *виконано* |
| 7 | Підготовка презентації та доповіді | 08.06.2024 | *виконано* |
| 8 | Попередній захист | 17.06.2024 | *виконано* |
| 9 | Нормоконтроль, рецензування | 10.06.2024 | *виконано* |
| 10 | Здача роботи у електронний архів | 15.06.2024 | *виконано* |
| 11 | Допуск до захисту у зав. кафедри | 17.06.2022 | *виконано* |

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Фарафонов Є.Ю.
(підпис)

Керівник роботи _____ доцент кафедри ПІ Кравець Н.С.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ / ABSTRACT

Звіт з кваліфікаційної роботи, 76 стор., 32 рис., 3 табл., 10 джерел.

ЗАМОВЛЕННЯ, ІНТЕРНЕТ МАГАЗИН, КОНДИТЕРСЬКІ ВИРОБИ, ТОРТ, JAVA SCRIPT, MONGOOSE, NODE.JS.

Об’єкт роботи – створення програмної системи. для продажу кондитерських виробів виготовлених на замовлення, яка матиме можливість зручно створювати персоналізовані замовлення, та відслідковувати процес їх виконання.

Мета роботи – створення backend частини програмної системи для продажу кондитерських виробів виготовлених на замовлення.

Метод рішення – редактор коду VS code, середа виконання Node.js, мова програмування JavaScript, база даних MongoDB. Object-Data Modeling бібліотека Mongoose. Google Cloud Storage для зберігання зображень. Веб-інтерфейс LiqPay для впровадження системи оплати.

У результаті розробки створено серверну частину програмної системи для продажу кондитерських виробів виготовлених на замовлення, яка реалізує функціонал для додавання та збереження нових тортів, оформлення та оплати замовлень, відстеження статусу замовлення.

ORDERS, INTERNET SHOPS, CONFECTIONERY PRODUCTS, CAKE, JAVA SCRIPT, MONGOOSE, NODE.JS.

The object of the work is the creation of a software system. for the sale of confectionery products made to order, which will be able to conveniently create personalized orders and track the process of their fulfillment.

The purpose of the work is to create the backend part of the software system for the sale of custom-made confectionery products.

Solution method – code editor VS code, Node.js execution environment, JavaScript programming language, MongoDB database. Object-Data Modeling Mongoose library.

Google Cloud Storage for image storage. LiqPay web interface for payment system implementation.

As a result of the development, the server part of the software system for the sale of custom-made confectionery products was created, which implements the functionality for adding and saving new cakes, placing and paying for orders, and tracking the status of the order.

Я, Фарафонов Євгеній Юрійович, студент гр. ПЗПІ-20-8, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для продажу кондитерських виробів виготовлених на замовлення. Backend. », що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

# ЗМІСТ

Вступ ..... 8
1 Аналіз предметної галузі ..... 10
1.1 Аналіз предметної галузі ..... 10
1.2 Виявлення та вирішення проблем ..... 15
1.3 Постановка задачі ..... 16
1.4 Цільова аудиторія ..... 18
2 Формування вимог до програмної системи ..... 20
2.1 Функціональні вимоги до програмного забезпечення ..... 20
2.2 Нефункціональні вимоги до програмного забезпечення ..... 21
2.3 Вимоги до технології розробки ..... 22
3 Архітектура та проектування програмного забезпечення ..... 23
3.1 UML проектування ..... 23
3.3 Проектування архітектури ..... 25
3.3 Проектування структури зберігання даних ..... 31
3.4 Приклади цікавих алгоритмів ..... 34
4 Опис прийнятих програмних рішень ..... 36
4.1 Вибір технологічного стеку та API інтеграцій ..... 36
4.2 Управління даними ..... 39
4.2.1 Контроллери ..... 39
4.2.1 API інтеграції ..... 43
5 Тестування програмного забезпечення ..... 45
Висновки ..... 49
Перелік джерел посилання ..... 51
Додаток А Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ ..... 52
Додаток Б  Слайди презентації ..... 53
Додаток В  Специфікація ПЗ ..... 61

# ПЕРЕЛІК СКОРОЧЕНЬ

БД – База Даних

ПС – Програмна Система

СКБД – Система Керування Базами Даних

API – Application Programming Interface

GCS – Google Cloud Storage

JSON – JavaScript Object Notation

HTTP – Hypertext Transfer Protocol

REST – Representational State Transfer

SEO – Search Engine Optimization

# ВСТУП

В сучасному світі інтернет-торгівля набуває все більшої популярності, а висока конкуренція змушує підприємства шукати нові шляхи для залучення клієнтів і збільшення обсягів продажів. Особливо актуальним це стає в галузі продажу продуктів харчування, до якої належить і кондитерська продукція[1, с. 20].

Програмна система для продажу кондитерських виробів виготовлених на замовлення є відмінним рішенням для підприємств, що спеціалізуються на виробництві та продажу цих товарів. Завдяки онлайн-магазину, клієнти матимуть змогу зручно та швидко замовляти унікальні та персоналізовані кондитерські вироби, а працівники магазину матимуть доступ до потрібних інструментів для ефективного керування продажами та замовленнями.

Актуальність теми полягає в тому, що в умовах постійного розвитку технологій і зростання вимог споживачів до сервісу, програмні рішення стають необхідними для успішного функціонування бізнесу. Така програмна система дозволить підприємствам у цій галузі підтримувати конкурентоспроможність і забезпечити задоволення потреб своїх клієнтів у зручних та ефективних способах придбання кондитерських виробів.

Основною метою цієї роботи є розробка веб-додатку для управління процесом замовлення, виготовлення та доставки кондитерських виробів. Веб-додаток повинен забезпечувати повний цикл роботи від прийому замовлення до його доставки клієнту. Це включає в себе функціональність для створення та управління замовленнями, обробку платежів, відстеження статусу замовлення та інтеграцію з системами доставки.

Темою кваліфікаційної роботи є програмна система для продажу кондитерських виробів виготовлених на замовлення. Основне завдання програмної системи– надання працівникам засобів зручного інструментарію керування онлайн магазином з продажу кондитерських виробів на замовлення а також можливості зручного процесу використання клієнтами магазину при замовленні.

Метою роботи є розробка програмної системи, яка являє собою сайт інтернет магазину. Програмної системи складається з веб-сайту який містить частину

користувача-покупця та користувача-адміністратора магазину. При виконанні роботи для створення frontend частини використовувалась бібліотека React та мова програмування JavaScript.

Для виконання backend частини використовувалася середа виконання Node.js з використанням фреймворку Express Відповідно до технологій використовуються середовища розробки VS Code.

Галузь застосування розробленої програмної системи охоплює підприємства, що займаються виготовленням і реалізацією кондитерських виробів на замовлення, а також інші бізнеси у сфері харчової промисловості, які потребують ефективного управління онлайн-замовленнями та доставки продукції.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Аналіз предметної галузі

Ринок виробу кондитерських виробів на замовлення не є дуже великим, але має постійний запит клієнтів. Переважно потреба виготовлення тортів на замовлення, у клієнтів, співпадає з якимись особистими святами як: день народження, весілля, річниці, або корпоративи компаній.

Таким чином переважно замовлення відбувається на один індивідуальний торт відповідно до потреб покупця, і може супроводжуватися додатковою купівлею якихось більш дешевих кондитерських виробів, що також можуть бути представлені в магазині готовими шаблонами.

Поточний розподіл ринку по пропозиції представляється двома головними опціями. Перший варіант представлений маленькими виробництвами, які ведуться через сторінки соціальної мережі Instagram (див. рис. 1.1).

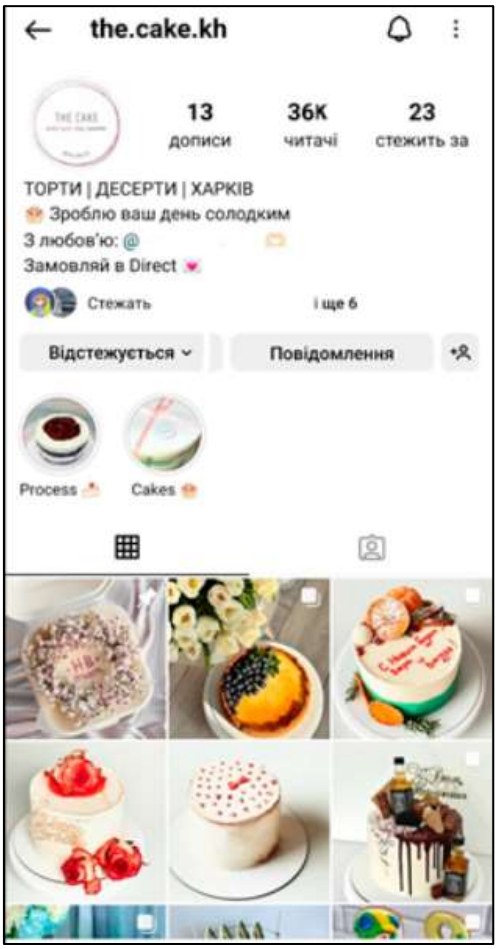


Рисунок 1.1 – Торти на замовлення в Instagram (за даними [2])

У таких магазинах клієнти зв’язуються з виробниками через прямі повідомлення, де обговорюється склад замовлення, ціна та інші деталі. Після факту передплати починається виробництво замовлення. Ці магазини, як правило, ведуться однією або двома особами і мають дуже обмежений ліміт обробки та виконання кількох замовлень одночасно

До головних переваг таких магазинів над програмною системою, що розробляється, є можливість користувачів побачити результати роботи виробника та переглянути відгуки, які дуже легко верифікувати, зв’язавшись з людьми, які робили замовлення в особистих повідомленнях. Це створює певний рівень довіри між покупцем і виробником, що важливо в умовах індивідуального підходу до замовлень.

Однак, такі магазини мають значні недоліки. По-перше, обмежена здатність до обробки замовлень означає, що вони не можуть виконувати великий обсяг замовлень одночасно, що може призводити до затримок і невдоволення клієнтів. По-друге, більша маржа і, відповідно, вища ціна для клієнта роблять такі послуги менш доступними для широкого кола споживачів. По-третє, такі магазини значною мірою потребують довіри від клієнта до виробника, оскільки ризики невиконання або поганого виконання замовлення, втрати грошей та відсутності компенсації значно вищі при роботі з індивідуальними виробниками, ніж при роботі з офіційним бізнесом.

Другий варіант представлений виробництвами малого, рідше середнього, розміру, які ведуть продаж через онлайн-магазини. Основний фокус таких підприємств переважно полягає у продажі стандартних товарів з заготовленого списку, з можливістю невеликих змін шаблонних заготовок тортів відповідно до потреб клієнта. Опція виробництва тортів за індивідуальним дизайном у таких випадках може бути відсутня, або здійснюватися через електронну пошту чи телефонний дзвінок. Далі наведено приклад онлайн-магазину з веб-сайтом (див. рис. 1.2).

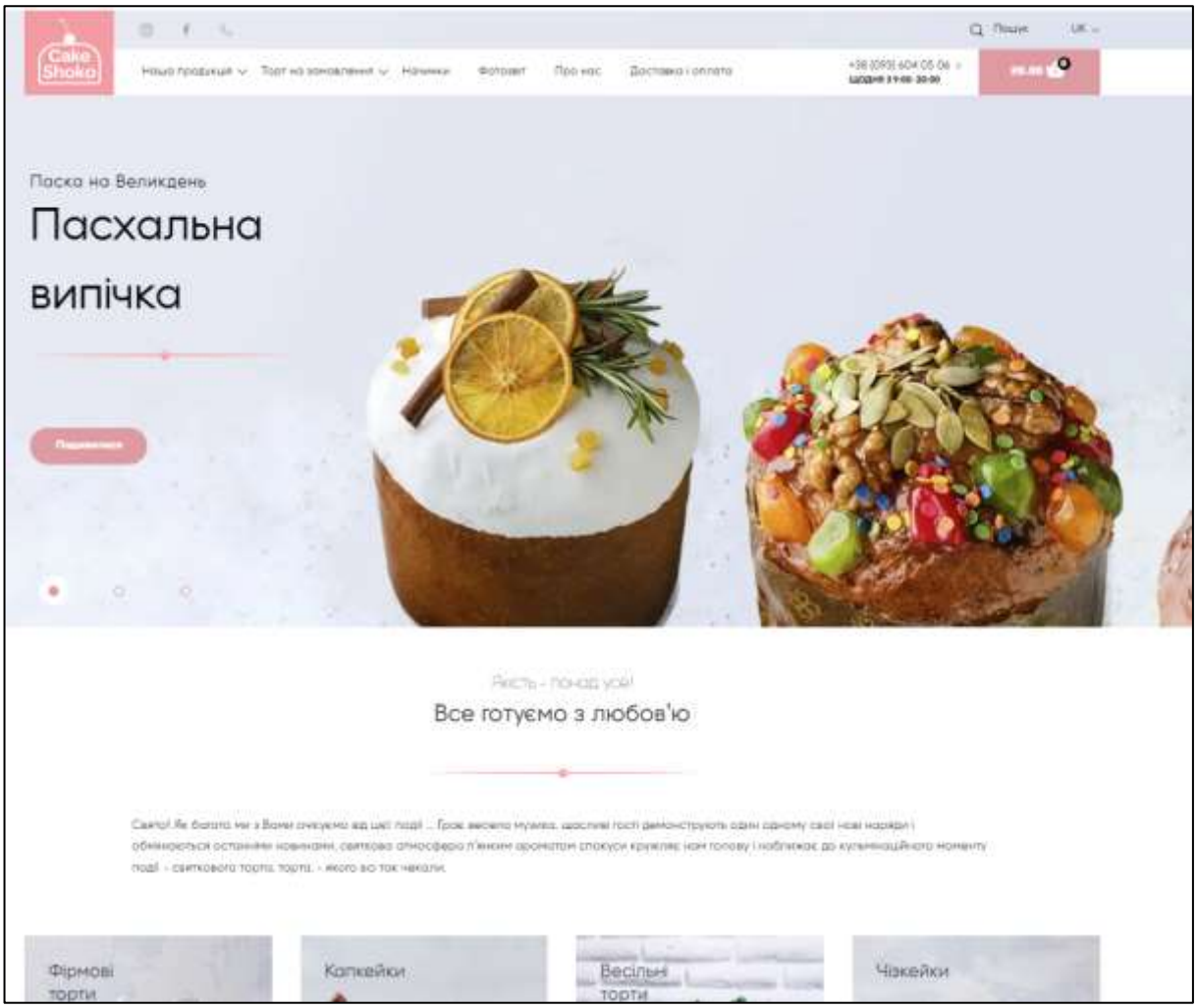


Рисунок 1.2 – Торти на в інтернет магазині (за даними [3])

Відповідно до цього можемо проаналізувати переваги та недоліки відносно ПС. Сайт має окрему сторінку з фотографіями від клієнтів які збільшують довіру до виробництва в нових потенційних покупців. Кількість поточних шаблонів, в наслідок певного часу роботи та розвитку цього підприємства, більша ніж те що включено в поточний вигляд програмної системи на старті існування.

Недоліки відповідно до розробленої ПС можна визначити, як: відсутність можливості замовлення по власному дизайну від користувача, та відсутність системи відстеження стану обробки та виконання замовлення.

Також можемо розглянути достатньо великі підприємства. Прикладом для цього будуть слугувати сайти групи компаній “Кулиничі”, яка є однією з найбільших постачальників хлібобулочних та кондитерських виробів на території України. “Кулиничі”

мають окремі сайти для роздрібної торгівлі та торгівлі оптом безпосередньо з ТОВ "КУЛИНИЧІВСЬКИЙ ХЛІБОЗАВОД".

Оскільки більшість користувачів нашої цільової аудиторії будуть переважно в пошуку можливостей купівлі кондитерської продукції в невеликих кількостях спочатку розглянемо сайт роздрібної торгівлі (див. рис. 1.3).

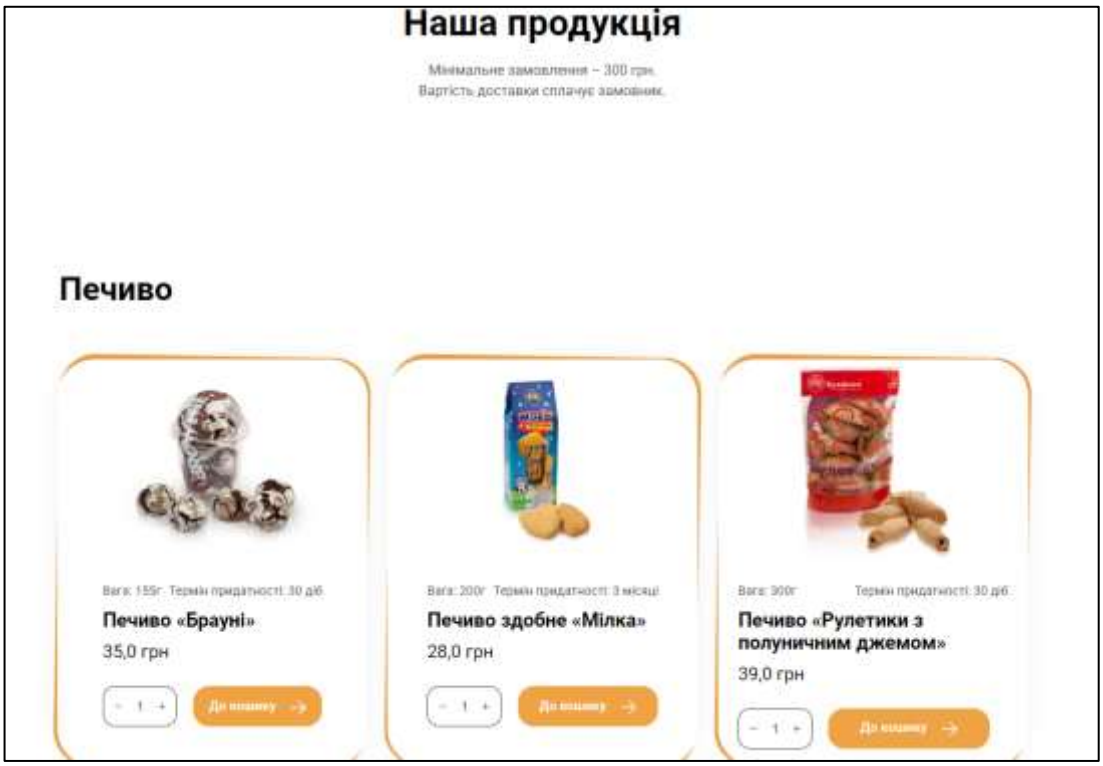


Рисунок 1.3 – Торти на в інтернет магазині (за даними [4])

З інформації консолі розробника зрозуміло, що сайт побудований за допомогою системи керування вмістом WordPress. В порівнянні з нашою програмною системо на React та Node.js можна побачити суттєві відмінності.

Проект, реалізований з використанням React та Node.js, надає більшу гнучкість та можливість індивідуальної настройки функціоналу в порівнянні з магазином на WordPress. Це надає можливість розробити власну логіку обробки замовлень, створити унікальні адміністративні панелі та функції, що відповідають конкретним вимогам до програмної системи.

Наприклад, на цьому сайті при оформленні замовлення і виборі місця доставки потрібно користувачу потрібно самостійно натиснути кнопку відобразити більше відділів

якщо потрібного вам не має у початковому списку і повторювати це поки не буде знайдений потрібний (див. рис. 1.4).


Рисунок 1.4 – Список відділень на сайті https://kulinichi.shop/checkout/ (за даними [4])

На сайті React та Node.js підгруження нових відділів можна реалізувати автоматично. Також не ефективні шаблони, або некоректне їх використання значно сповільнюють WordPress сайти. Наприклад кошик навіть з одного товару формується кілька секунд, деякі продукти не мають міні картинки в меню кошику, якщо оновити сторінку попередні товари в меню кошика не з’являться якщо не додати ще один . (див. рис. 1.5).

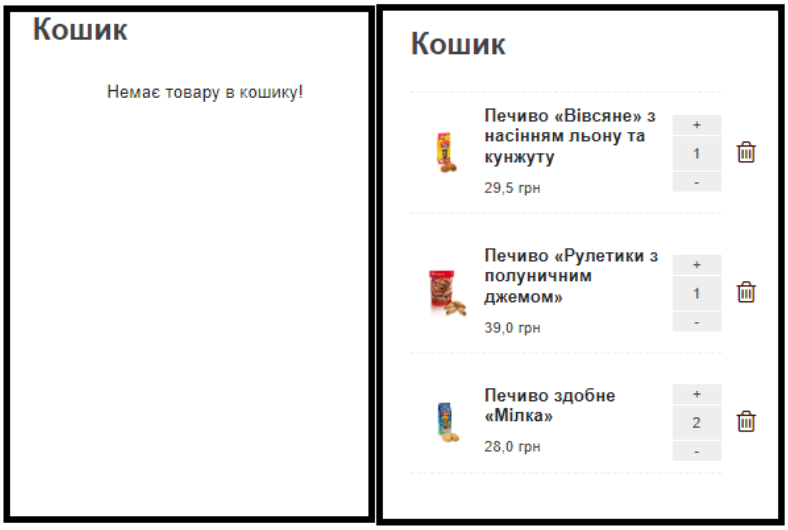


Рисунок 1.5 – Стан кошику після оновлення сторінки, та після додавання одного предмету у кошик(за даними [4])

Використання React та Node.js дозволяє створити швидкий та продуктивний веб-додаток з високою відзивчивістю. У порівнянні з WordPress, який може мати обмежену швидкість завантаження та продуктивність при великій кількості плагінів та додатків.

WordPress може бути простішим у встановленні та налаштуванні, особливо для невеликих магазинів. Проте, при розширенні функціоналу та потребах у складних налаштуваннях, він може потребувати додаткових зусиль та плагінів. З іншого боку, ваш проект на React та Node.js може вимагати більше технічних знань для налаштування та адміністрування, але забезпечить більшу гнучкість та контроль над системою.

Таким чином, можемо побачити, що для великої кількості конкурентів на ринку виготовлення тортів та кондитерських виробів, або не мають виробництва персоналізованих тортів зовсім, або мають дуже обмежену персоналізацію замовлень. Жоден з конкурентів не має можливості відслідковування стану замовлень після оплати. ПС я була створена в ході робити містить цей функціонал, що надає їй перевагу над конкурентами.

## 1.2 Виявлення та вирішення проблем

Найбільшими проблемами для бізнесу, що тільки розпочинає своє існування, є створення іміджу бренду, та початкової клієнтської бази, отриманнядостатнього рівню довіри від клієнтів [5, с. 2]. Вирішення цього питання потребує грамотної маркетингової стратегії де можна запропонувати кілька підходів:

- розміщення реклами в соціальних мережах (прик. Instagram Facebook);
- розміщення рекламних банерів та листівок на вулиці;
- акції для покупців що виставляють схвальний відгук про магазин на власних сторінках соціальних мереж;
- замовлення рекламної інтеграції у інфлюєнсерів відповідно до вподобань цільової аудиторії.

Окремо можна сказати про проведення якісної SEO оптимізації, що дозволить сайту з’являтися раніше в рекомендаціях користувачам, які просто шукають якісь варіанти магазинів в Інтернеті. Розташування в результатах пошуку

навіть на одну позицію вище може суттєво підвисити кількість відвідувань, відповідно кількість можливих замовників.

Суттєвою перепоною для реалізації програмної системи для продажу кондитерських виробів виготовлених на замовлення є обробка спеціальних замовлень. Оскільки цей процес потребує прямого зв'язку між клієнтом та працівником магазина для уточнення деталей та ціни на товар який завчасно не зареєстрований у системі, що робить цей процес більш складним для обробки та маркування в системі.

Для цього програмна система реалізує окрему процедуру додавання спеціального замовлення адміністратором, після успішного обговорення з клієнтом, та подальшого підтвердження замовлення користувачем зі свого аккаунту на сайті.

Це надійній процес який максимально зменшує можливість непорозумінь, які могли б виникнути, а також конкретно фіксує ціну, та згоду клієнта з ціною, що є важливим для подальшого ведення  бізнес процесів, таких як звітність.

### 1.3 Постановка задачі

Мета роботи – це розробити веб-додаток для замовлення кондитерських виробів з використанням технологій React та Node.js, який надасть користувачам можливість швидко та зручно здійснювати замовлення, а адміністраторам - ефективно управляти каталогом та обробляти замовлення.

ПС реалізована на основі схеми розподілення даних Model-View-Controller з яких Backend частина має реалізувати моделі та контролери. Моделі для цієї програми будуть слугувати шаблонами, на основі яких буде будуватися база даних. Контролери відповідно до кожної моделі є реалізаціями методів, необхідних для зв'язку між Frontend частиною та обраною БД, тобто для передачі, збереження та зміни даних. Крім того, серверна частина відповідає за безпекові аспекти програмної системи, такі як перевірка валідності токенів аутентифікації, перевірка валідності надісланих даних та забезпечення захисту користувацької інформації через шифрування і хешування паролів.

При використанні програмної системи користувачі повинні мати можливість:

- реєстрації нового акаунта;

- авторизації акаунта;
- можливості переглядати список продукції;
- можливості створювати торт за шаблоном;
- можливості оформлювати замовлення;
- сплачувати замовлення;
- можливості відстежувати стан замовлення;
- оплати через інтегровану з платіжну систему для здійснення безпечної оплати замовлення.

Адміністраторська частина має надавати можливість:

- авторизації акаунта;
- додавання нових адміністраторів;
- додавання та редагування каталогу;
- обробки та відстеження замовлень;
- обробки спеціальних тортів на індивідуальне замовлення.

Система замовлення кондитерських виробів буде побудована на клієнт-серверній архітектурі, де клієнтська частина розроблена на React, а серверна — на Node.js, для управління базою даних буде використана MongoDB. Використання такої архітектури дозволяє розділити логіку веб-додатку між клієнтом та сервером, що полегшує розширення та обслуговування системи.

Зокрема, клієнтська частина, створена на React, забезпечує динамічний та інтуїтивно зрозумілий інтерфейс користувача. Використання компонентного підходу у React дозволяє створювати повторно використовувані UI-компоненти, що значно підвищує ефективність розробки та підтримки інтерфейсу. Це також сприяє більш швидкому реагуванню додатку на дії користувача, що покращує загальний користувацький досвід.

На серверній стороні використання Node.js забезпечує асинхронність обробки запитів, що підвищує продуктивність та масштабованість додатку. Використання Express.js як фреймворку для Node.js дозволяє швидко і ефективно створювати RESTful API, які є основою для взаємодії між клієнтською та серверною частинами. Цей підхід забезпечує стандартизовані інтерфейси для

доступу до функціональності сервера, що спрощує інтеграцію з іншими сервісами та розширення додатку в майбутньому.

MongoDB, як обрана база даних, забезпечує гнучке зберігання даних завдяки своїй документно-орієнтованій структурі. Це дозволяє ефективно зберігати складні та різнорідні дані, такі як: інформація про користувачів, замовлення, продукцію та медіафайли (наприклад, фотографії тортів). Використання Mongoose, об'єктно-документного моделювального інструменту для MongoDB, спрощує роботу з даними завдяки його потужним можливостям для створення схем та валідації даних.

## 1.4 Цільова аудиторія

Цільова аудиторія ПС буде складатися з таких сегментів:

а) приватні клієнти:

- споживачі для особистих потреб: індивідуальні покупці, які бажають персоналізовані кондитерські вироби для особливих подій, таких як дні народження, весілля, ювілеї та інші святкування;
- подарунковий сегмент: люди, що шукають ексклюзивні та персоналізовані подарунки для друзів, родичів або колег;

б) корпоративні клієнти:

- бізнес-замовлення: компанії, що організовують корпоративні заходи, свята, конференції та інші заходи, де потрібні великі замовлення кондитерських виробів;
- маркетингові та рекламні акції: фірми, що використовують персоналізовані солодощі як частину промоакцій або корпоративних подарунків;

в) організатори подій та кейтерингові компанії:

- весільні планувальники: організатори свят, які замовляють кондитерські вироби для весільних церемоній;
- кейтерингові служби: сервіси, що постачають їжу на заходи та можуть потребувати унікальні кондитерські вироби для своїх клієнтів.

Характеристики цільової аудиторії:

Вікові групи: від молоді (20-30 років), що шукають цікаві та нестандартні десерти, до осіб середнього віку (30-50 років), які можуть замовляти вироби для своїх дітей, або на корпоративні події.

Доходи: вищий та середній клас, здатний оплачувати преміум-продукти та персоналізацію.

Географічне розташування: мешканці великих міст, та середніх де існує більший попит на унікальні, персоналізовані кондитерські вироби.

Соціальні характеристики: постійні користувачі соціальних мереж, що бажають урізноманітнити свою активність цікавим досвідом, які використовують Інтернет для пошуку подарунків та організації заходів.

Психографічні особливості: висока оцінка якості, унікальності та кастомізації продукції. Звертають увагу на деталі, індивідуальний підхід та сервіс.

# 2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

## 2.1 Функціональні вимоги до програмного забезпечення

Система передбачає наявність декількох ролей: адміністратор, клієнт, незареєстрований користувач.

Для незареєстрованого користувача, щоб створити акаунт, необхідно буде заповнити форму реєстрації зі збереженням основних даних: збереження основних даних: ім'я, електронна пошта, телефон і т.д.

Вже для зареєстрованого користувача (клієнта) передбачається клієнтська частина з наступним функціоналом:

а) авторизація:
   - вхід в акаунт за допомогою логіна та пароля;
   - забезпечення безпеки даних користувача;

б) взаємодія з профілем:
   - перегляд даних про користувача;
   - зміна даних про користувача;

в) вибір з каталогу:
   - перегляд доступних кондитерських виробів та їх опис;
   - фільтрація товарів за категоріями та характеристиками;

г) оплата:
   - інтеграція з платіжними системами для здійснення оплати;
   - збереження історії та статусу оплати;

д) конструктор:
   - можливість складання власних кондитерських виробів з різних компонентів;

е) оформлення замовлення:
   - вибір точки видачі замовлення;
   - визначення дати та часу отримання;

ж) відстеження замовлення:
   - доступ до інформації щодо статусу замовлення;

– оформлення замовлення.

Також система має роль адміністратора, щоб стати адміністратором зареєстрований акаунт має отримати статус адміністратора від вже наявного адміністратора. Адміністраторська частина передбачає:

а) авторизація:

– вхід в акаунт адміністратора з логіном та паролем;

б) додання нових позицій:

– можливість додавання нових кондитерських виробів до каталогу;

– вказання інформації про продукт, ціни та характеристик;

в) обробка замовлень:

– отримання підтвердження оплати;

– перегляд та підтвердження/відхилення замовлень;

– взаємодія з клієнтами щодо замовлень (зворотний зв'язок);

г) управління акаунтами:

– перегляд інформації про акаунти користувачів ;

– можливість надавати статус адміністратора.

## 2.2 Нефункціональні вимоги до програмного забезпечення

Нефункціональні вимоги для програмної системи включають декілька аспектів.

Безпека даних: захист особистих даних клієнтів та адміністраторів, захист несанкціонованого доступу.

Відмовостійкість: забезпечення стабільності та продуктивності системи при кількості користувачів до 100 одночасно.

Інтерфейс має бути зручним та інтуїтивно зрозумілим як для клієнтів, так і для адміністраторів.

Оптимізація роботи системи для швидкого відгуку на запити користувачів.

Забезпечення сумісності системи з різними пристроями та браузерами.

## 2.3 Вимоги до технології розробки

Програмна система буде складатися з Frontend та Backend частин які будуть розроблені відповідно на React та Node.js. Далі наведені більш детальні вимоги.

Frontend (React):

а) компонентна архітектура:
   - використання компонентної архітектури React для розділення інтерфейсу на невеликі та переиспользувані компоненти;

б) аутентифікація і авторизація:
   - реалізація механізмів аутентифікації та авторизації для клієнтської частини;

в) управління станом:
   - використання стейт-менеджменту (Redux) для збереження та обробки глобального стану додатку;

Backend (Node.js):

а) API розробка:
   - реалізація ефективного та добре документованого REST API для взаємодії з клієнтською частиною.;

б) безпека:
   - застосування стандартів безпеки, таких як обробка паролів, запобігання SQL-ін'єкціям та інші заходи для захисту від атак;

в) валідація даних:
   - залідація даних на рівні сервера для запобігання некоректних або шкідливих даних;

г) база даних:
   - підключення та використання бази даних MongoDB;

д) оплата:
   - підключення системи оплати LiqPay;

е) аутентифікація та авторизація.

# 3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 UML проектування

Перед початком розробки програмної системи для продажу кондитерських виробів виготовлених на замовлення були визначені основні функції зі сторони покупця та адміністратора. Після детального аналізу були створени Use-case діаграми. Далі наведена діаграма покупця (див. рис. 3.1).

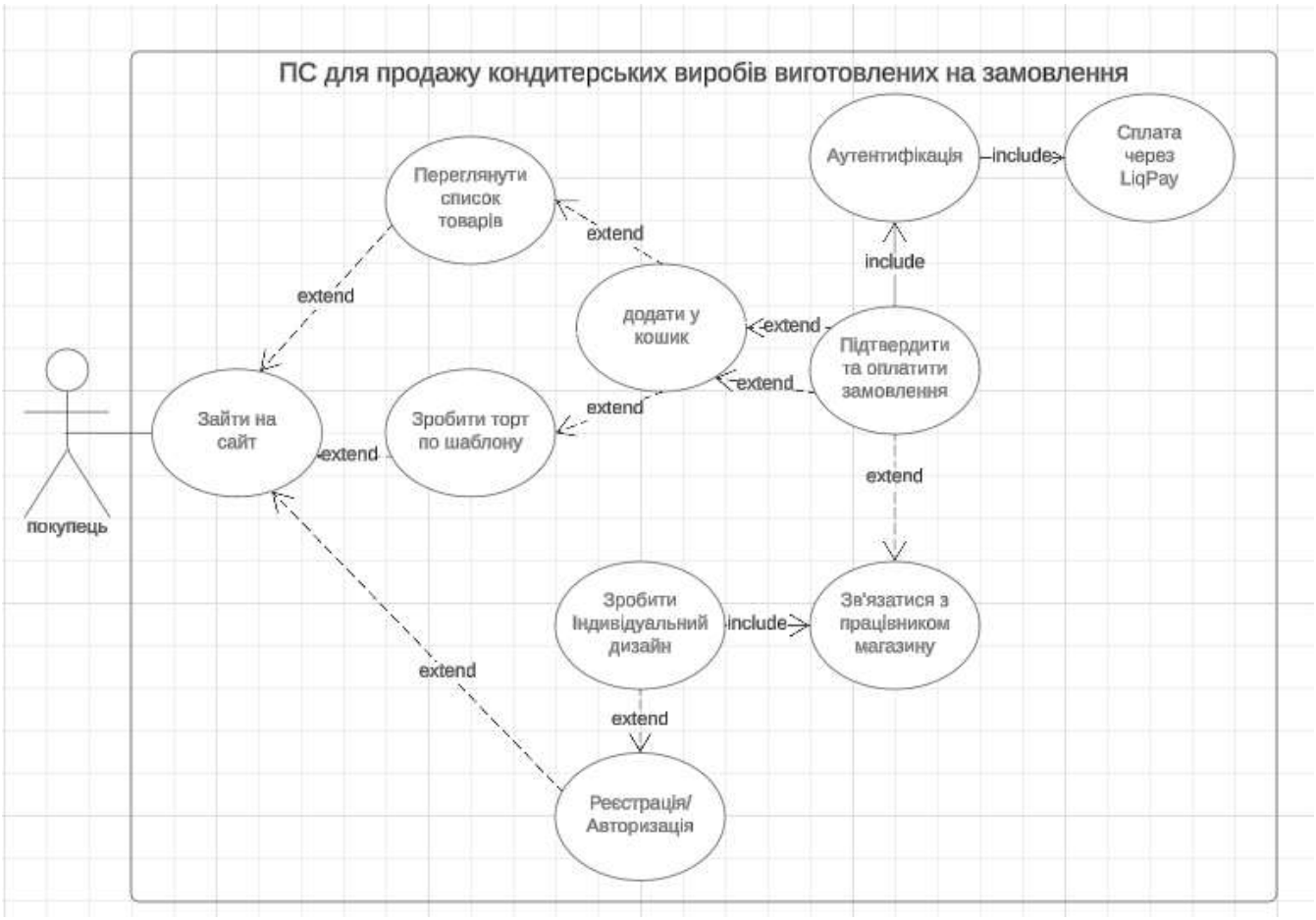


Рисунок 3.1 – Use-case діаграма покупця(рисунок виконаний самостійно)

Користувачем в даному випадку є покупець. Після того як користувач заходить на сайт, в нього є опції зареєструватися в системі або авторизуватися, якщо аккаунт вже є, переглянути список запропонованої продукції, сформувати власний торт по шаблонам. Для створення індивідуального дизайну потрібно спочатку пройти авторизацію. Після надсилання форми на індивідуальне замовлення з користувачем зв’яжеться працівник магазину, та після обговорення

замовлення клієнт отримає можливість підтвердити власне замовлення на сайті, підтвердити замовлення тортів по шаблону чи зі списку можна одразу після додавання їх у кошик. Підтвердити та сплатити замовлення можна в разі проходження аутентифікації.

Також був сформований Use-case сценарій та діаграма до нього для найманого працівника, адміністратора магазину(див. рис. 3.2).

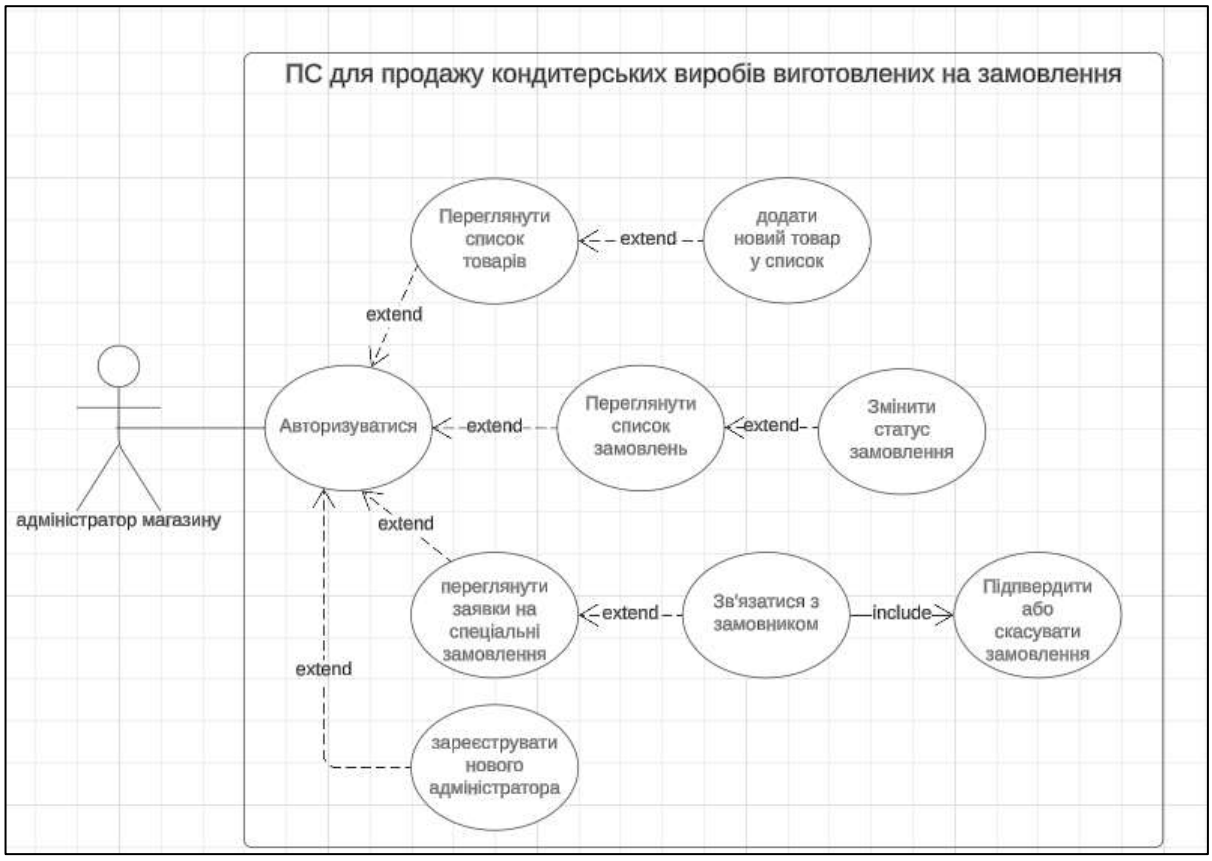


Рисунок 3.2 – Use-case діаграма адміністратора магазину (рисунок виконаний самостійно)

Для виконання будь яких дій в системі від імені адміністратора обов’язково потрібно пройти авторизацію. Авторизувавшись такий користувач може зареєструвати нового адміністратора, переглянути список товарів, список замовлень, та запитів на спеціальні замовлення. Зі спуску товарів можна додати нові позиції. Зі списку замовлень можна змінювати статус виконання окремих замовлень.

Після перегляду заявки на спец замовлення адміністратор має звертатися до покупців що їх оформили через пошту або телефон. Після цього, або скасувати замовлення, або підтвердити його, таким чином надаючи можливість покупцю оплатити домовлену суму з сайту.

### 3.3 Проектування архітектури

Обрана архітектура клієнт-сервер є оптимальним варіантом для цієї роботи через її здатність до ефективного розподілу завдань між клієнтською і серверною частинами програми [6]. Це дозволяє чітко розділити бізнес-логіку та інтерфейсну взаємодію, що спрощує розробку, тестування та масштабування програмного забезпечення.На серверній частині можна ефективно контролювати доступ до даних, виконувати процедури автентифікації та авторизації, що робить систему більш безпечною для користувачів. Додатково, використання архітектури клієнт-сервер сприяє покращенню продуктивності та ефективності роботи програми, оскільки завдяки розділенню функцій між клієнтською і серверною частинами можливе оптимізоване виконання завдань. Крім того, ця архітектура забезпечує більшу гнучкість і можливість розширення системи у майбутньому. Загальна схема подібної архітектури наведена далі (див. рис. 3.3).

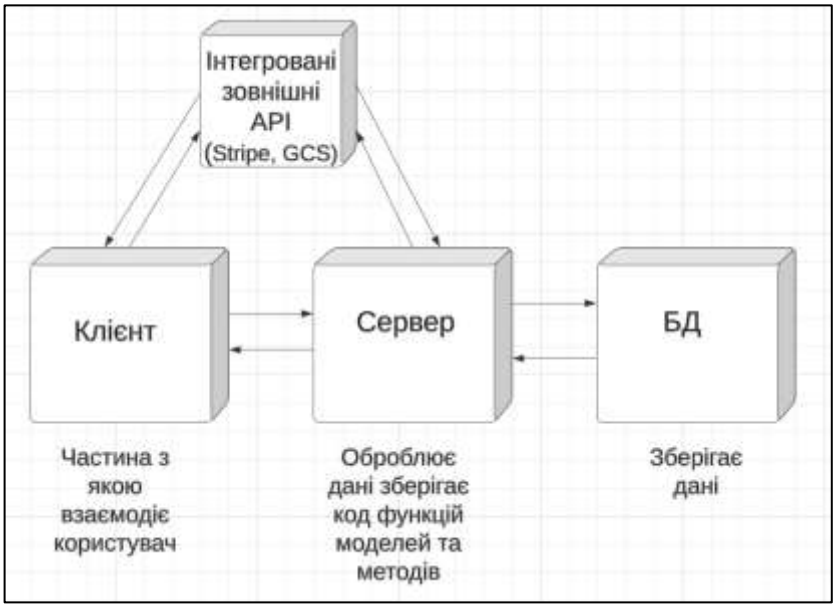


Рисунок 3.3 – Загальна схема клієнт-серверної архітектури(рисунок виконаний самостійно)

Backend додатку реалізовано на платформі Node.js, що забезпечує високу швидкодію та ефективність в обробці запитів. Використання мови програмування JavaScript дозволяє забезпечити однорідний стек технологій та прискорити процес розробки завдяки широкому спектру наявних бібліотек і фреймворків.

Зв'язок між серверною частиною та веб-сайтом відбувається за допомогою HTTP запитів, що дозволяє передавати дані між клієнтом і сервером. У разі потреби дані з сайту передаються на сервер у форматі JSON (JavaScript Object Notation), який є легким у сприйнятті і обробці як на стороні клієнта, так і на стороні сервера. (див. рис. 3.4).

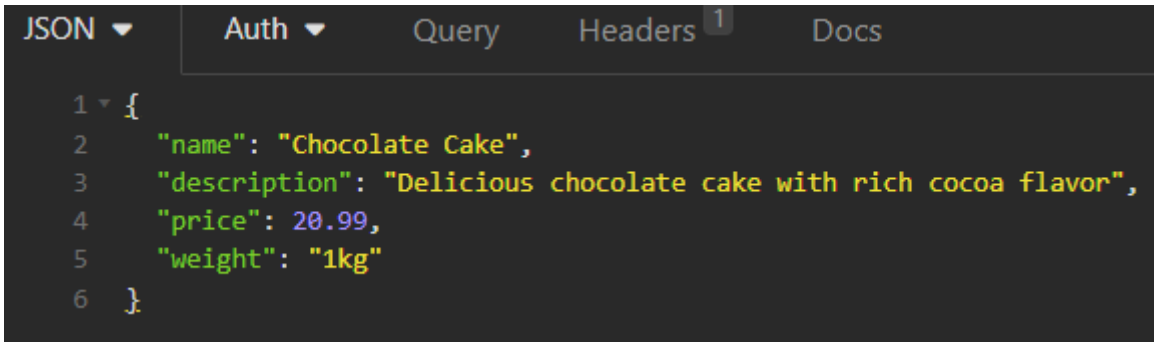


Рисунок 3.4 – Приклад JSON файлу відправленого на сервер (рисунок виконаний самостійно)

Серверна архітектура базується на REST API, що забезпечує стандартизований підхід до створення веб-сервісів і спрощує їхню інтеграцію з іншими системами [7]. Використання REST API дозволяє створювати легкі та ефективні інтерфейси для взаємодії з сервером, що сприяє підвищенню швидкодії та масштабованості додатку. REST (Representational State Transfer) є архітектурним стилем, який базується на протоколі HTTP і використовує стандартні методи, такі як GET, POST, PUT, DELETE для виконання CRUD операцій. Це робить взаємодію з API інтуїтивно зрозумілою і легкою для реалізації.

Завдяки REST API, кожен ресурс в системі має унікальний ідентифікатор (URL), що дозволяє легко ідентифікувати та маніпулювати ресурсами. Це дозволяє створювати модульну і розширювану архітектуру, де кожен компонент може розвиватися незалежно один від одного. Наприклад, додавання нового типу

ресурсу або оновлення існуючого не вимагає зміни всієї системи, а тільки відповідних контролерів і маршрутів.

REST API також сприяє покращенню продуктивності за рахунок використання кешування. HTTP протокол підтримує кешування відповідей, що дозволяє зменшити навантаження на сервер і скоротити час відповіді для клієнтів. Це особливо важливо для великих систем з високим навантаженням, таких як інтернет-магазини, де швидкість обробки запитів може прямо впливати на задоволеність користувачів і конверсії.

Для забезпечення функціональності нашого додатку був реалізований набір кінцевих точок (ендпоінтів) API(табл. 3.1).

Таблиця 3.1 – REST специфікація (таблиця виконана самостійно)

| **Назва** | **Метод** | **Ендпоінт** | **Тіло** |
|---|---|---|---|
| User | POST | /user/register | email,fullName, password, phoneNumber |
| | POST | /user/login | email, password |
| | GET | /user | - |
| | POST | /user/register/admin | email,fullName, password,phoneNumber |
| | PUT | /user | email,fullName, password, phoneNumber |
| Regular Bakery | POST | /regularbakery | category,name, description,weight, price, img |
| | DELETE | /regularbakery/ | |
| | PUT | /regularbakery/ | category, name, description, weight, price, img |
| | GET | /regularbakery | |
| | GET | /regularbakery/:id | |
| | GET | /regularbakery/category/:id | |
| SpecialCake | POST | /specialcake | |
| | DELETE | /specialcake/:id | |
| | PUT | /specialcake/:id | |

Продовження таблиці 3.1

| | | | |
|---|---|---|---|
| | GET | /specialcake | |
| | GET | /specialcake/:id | |
| | GET | /specialcakes/user/:id | |
| Order | POST | /order | cakeIds,tasteIds, numbers,weights, words,prices, bakeryIds,userId, totalPrice |
| | GET | /order | |
| | GET | /order/:id | |
| | PUT | /order/:id | words, status |
| | DELETE | /order/:id | |
| | GET | /order/user/ | |
| Shablon Cake | POST | /shabloncake | name,description,price, weight, img |
| | GET | /shabloncake | |
| | GET | /shabloncake/:id | |
| | PUT | /shabloncake/:id | name,description, price, weight, img |
| | DELETE | /shabloncake/:id | |
| Taste | POST | /taste | cakeId,name, description, img |
| | DELETE | /taste/:id | |
| | PUT | /taste/:id | cakeId,name, description, img |
| | GET | /taste | |
| | GET | /taste/cake/:id | |
| uploadImagesToGCS | POST | /upload/tastes | tasteId,tastes (images) |
| | POST | /upload/bakery | bakeryId,bakery (images) |
| | POST | /upload/cake | cakeId,cake (images) |
| Password Reset | POST | /forgot-password-email | email |
| | POST | /check-reset-code | email, code |
| | PATCH | /password-reset | email, pass, code |

Загальна кількість ендпоінтів 39. Реалізовані методи покривають визначені у вимогах потреби з приводу функціональності Backend частини. Згідно до схеми

Model-View-Controller серверна частина має моделі та контролери. Повна структура файлів серверної частина наведена нижче(див. рис. 3.5).

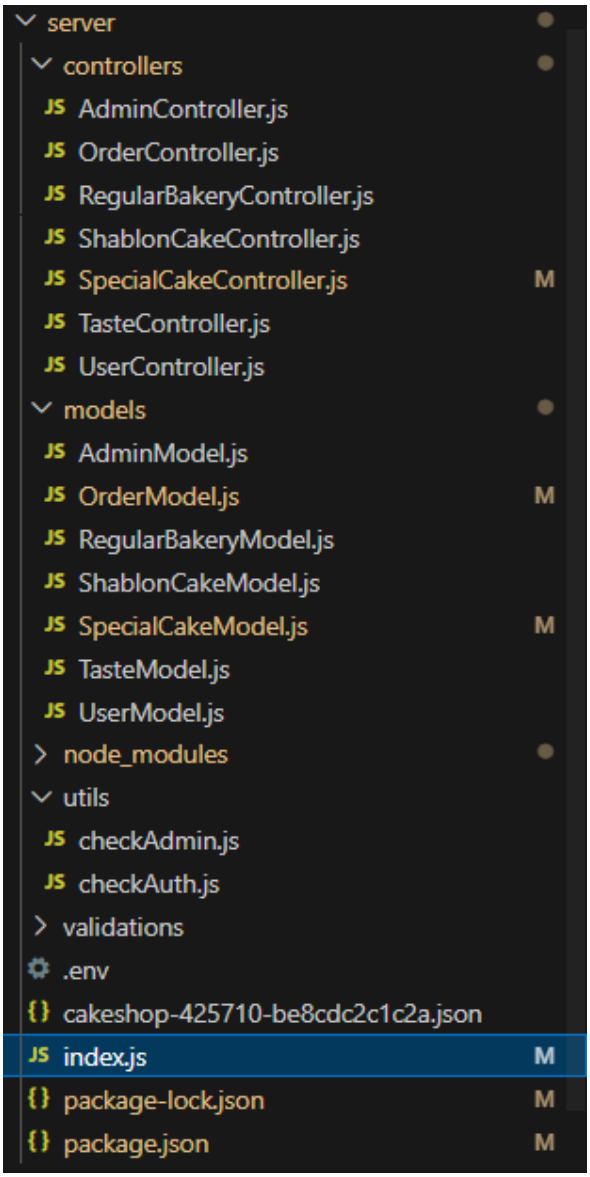


Рисунок 3.5 – Структура файлів серверної частини (рисунок виконаний самостійно)

Моделі містять опис необхідних для роботи ПС об’єктів, необхідні їм змінні, що визначає схему того, як вони будуть зберігатися та відображуватися у БД. Далі наведено приклад схеми користувача.

```
const UserSchema = new mongoose.Schema({
    fullName:{
        type: String,
```

```
        required: false,
    },
    email: {
        type: String,
        required: true,
        unique: true,
    },
    phoneNumber: {
        type: String,
        required: false,
    },
    passwordHash:{
        type: String,
        required: true,
    },
    role: {
        type: String,
        default: 'customer'
    }
},{
    timestamps: true,
});
export default mongoose.model('User', UserSchema);
```

Далі наведено приклад того як виглядають контролери, в прикладі це контролер тортів за шаблоном.

```
import ShablonCake from '../models/ShablonCakeModel.js';

export const create = async (req, res) => {
};

export const update = async (req, res) => {
};

export const remove = async (req, res) => {
};

export const findAll = async (req, res) => {
};
```

Методи що відповідають опрацюванню конкретних моделей зберігаються в окремих файлах за схемою ModelNameController.js в загальній папці Controllers що значно спростило процес розробки ПС, та облегшує потенційні розробки під час масштабування, чи пошуку можливих проблем.

## 3.3 Проектування структури зберігання даних

У цій ПС використовувалася база даних на MongoDB для зберігання та управління даними про торти, смаки, користувачів, замовлення та інші необхідні для функціонування веб-додатку дані. Головною перевагою обраної СКБД MongoDB є її гнучка схема даних, яка дозволяє легко змінювати структуру без необхідності переробляти всю базу даних [8, с. 35].

Крім того, MongoDB добре масштабується і забезпечує високу продуктивність навіть при великій кількості записів. Для візуалізації даних бази при розробці ПЗ використовувався застосунок MongoDB Compass який відображає усі сутності які формують базу, та об’єкти які збережені у ці сутності (див. рис. 3.6).

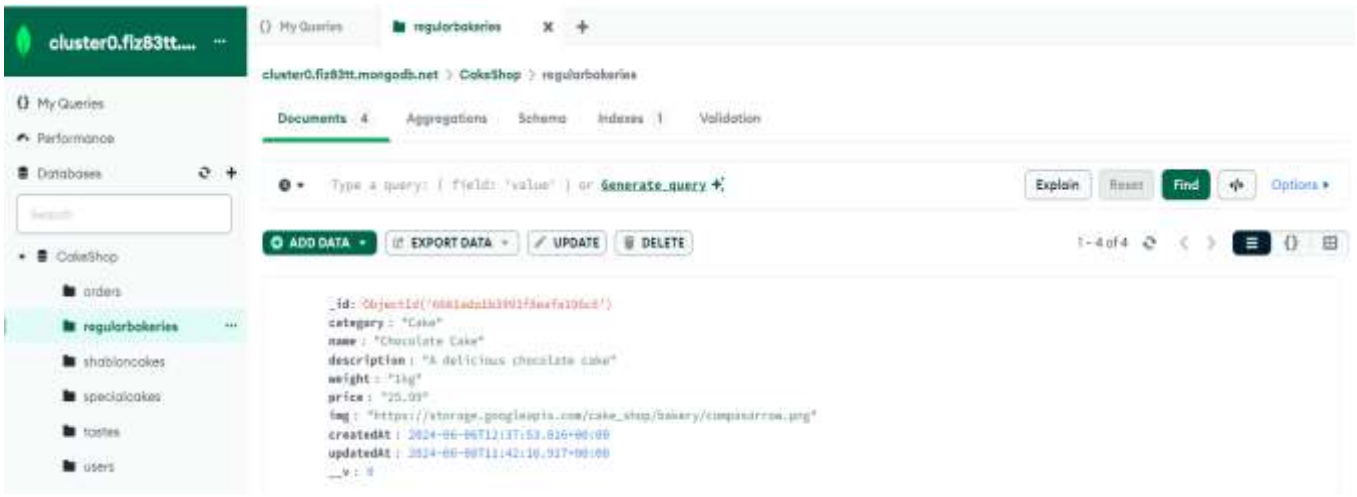


Рисунок 3.6 – Візуалізація БД через MongoDB Compass (рисунок виконаний самостійно)

В процесі розробки була утворена база даних для програмної системи для продажу кондитерських виробів. Дані зберігаються в 6 сутностях: користувач, торт, торт по шаблону, смак, спеціальний торт, замовлення. Замовлення містить у собі список замовлених тортів, їх загальну ціну та ID користувача.

Спеціальні торти оформлюються по одному на замовлення. Звичайні торти та торти створені по шаблону зберігаються у замовлені через масиви їхніх ID. Кожен смак зв’язаний з якимось тортом по шаблону і містить його ID, власний окремий опис та картинку.

Схема бази даних представлена ER-діаграмою (див. рис. 3.7).

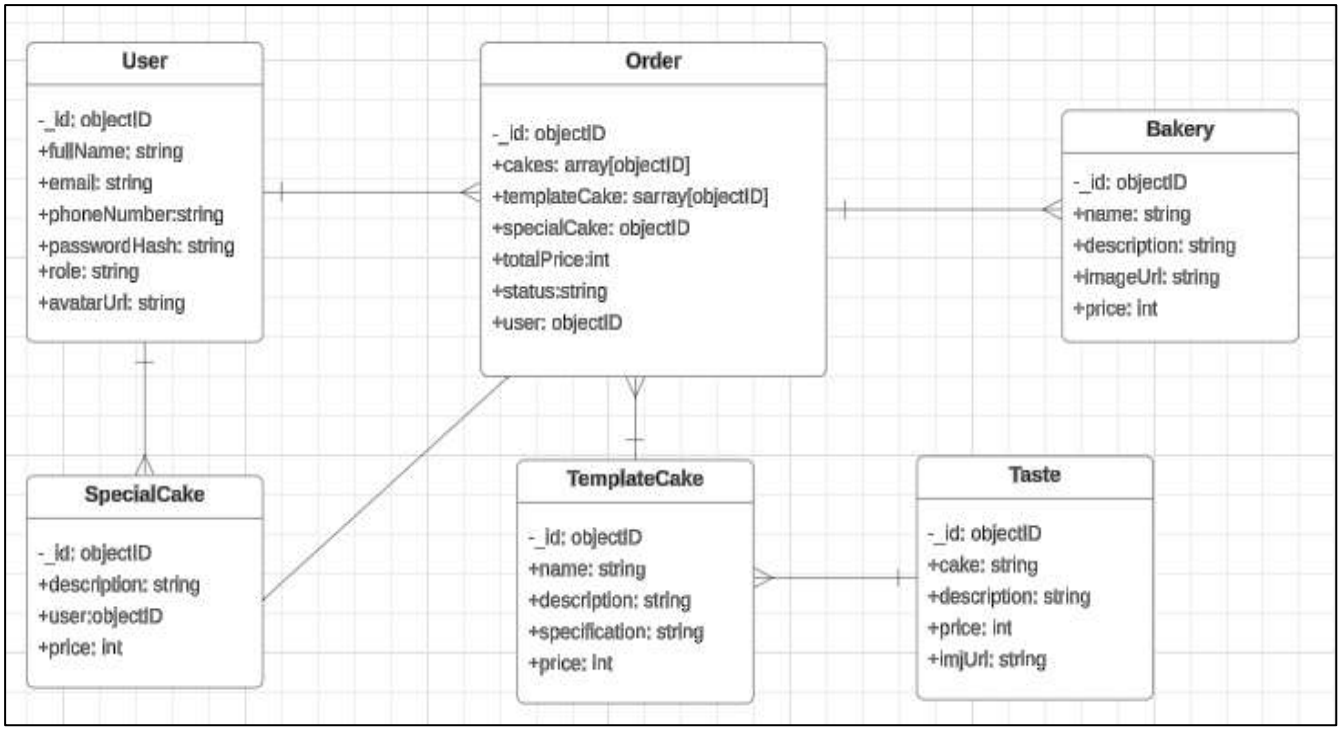


Рисунок 3.7 – ER-діаграма (рисунок виконаний самостійно)

Програмна система передбачає зберігання великої кількості зображень для різних сутностей: звичайних тортів, тортів за шаблоном, смаків до них, тому для їх зберігання було вирішено застосовувати хмарне сховище Google Cloud Storage. На сховищі зображення зберігаються в окремих розділах, відповідно до сутностей яким вони належать (див. рис. 3.8).

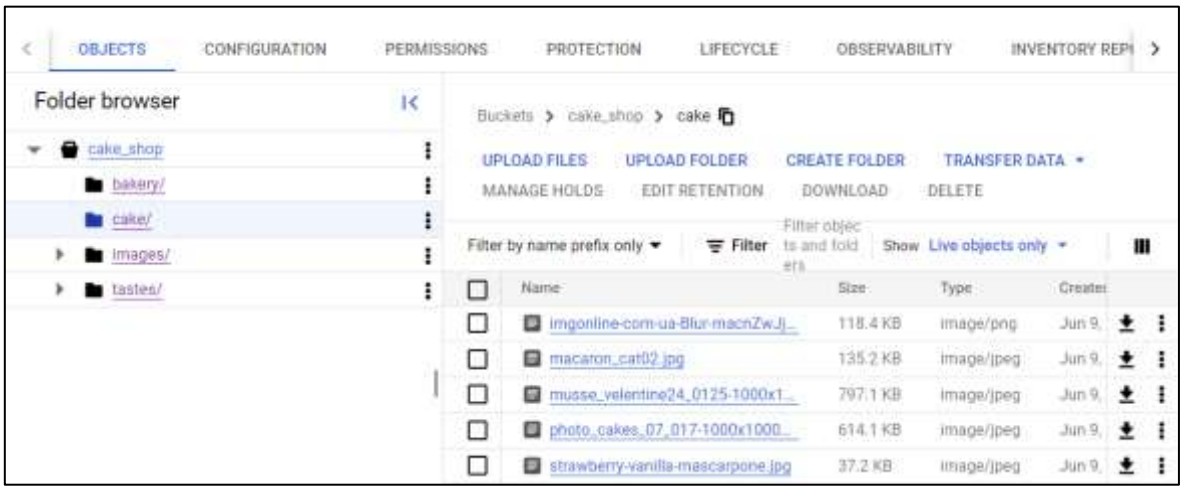


Рисунок 3.8 – Зберігання зображень у GCS(рисунок виконаний самостійно)

При цьому у базі даних в сутностях зберігаються лише посилання на зображення у Google Cloud. В коді цей процес реалізований за допомогою бібліотеки multer. Приклад коду функції яка відповідає за зберігання зображень, в даному випадку смаку:

```
app.post('/upload/tastes', uploadImages.array('tastes'), async (req,
res) => {
    try {
        const files = req.files;
        const tasteId = req.body.tasteId;
        if (files && tasteId) {
           const publicUrls = await uploadImagesToGCS(files, 'tastes');
                const id=tasteId;
                const img= publicUrls[0]
                const updatedTaste = await updateImg([id,img], res);
            res.status(200).json(updatedTaste);
        } else {
            throw new Error('Taste ID and images are required.');
        }
    } catch (error) {
        console.error('Error in route handler:', error);
        res.status(500).json({ message: 'Images upload failed', error:
error.message });
    }
});
```

Функція отримає масив зображень та id смаку для якого зберігається зображення. Якщо немає проблем з отриманими даними зображення надсилається на Google Cloud Storage за допомогою функції uploadImagesToGCS яка повертає URL посилання на збережене зображення, яке зберігається в БД функцією updatedTaste.

```
export const uploadImagesToGCS = async (files, folder) => {
    const uploadPromises = [];

    try {
        files.forEach((file) => {
            console.log("Processing file:", file.originalname);
            const { originalname, buffer } = file;
            const fileName = `${folder}/${originalname}`;
            const blob = bucket.file(fileName);
            const blobStream = blob.createWriteStream({
                resumable: false,
            });
```

```
const uploadPromise = new Promise((innerResolve, innerReject) => {
                blobStream
                    .on('finish', () => {
                        const                publicUrl                =
`https://storage.googleapis.com/${bucket.name}/${blob.name}`;
console.log("Upload successful, public URL:", publicUrl);
                        innerResolve(publicUrl);
                    })
                    .on('error', (err) => {
                        console.error("Error  during  file  upload:",
err);
                        innerReject(err);
                    }).end(buffer);
            });

            uploadPromises.push(uploadPromise);
        });

        const publicUrls = await Promise.all(uploadPromises);
        return publicUrls;
    } catch (error) {
        console.error("Error in uploadImagesToGCS:", error);
        throw error;
    }};
```

### 3.4 Приклади цікавих алгоритмів

Враховуючи те, що ПС включає необхідність живої комунікації адміністратора та користувача та клієнта для оформлення індивідуальних тортів на замовлення, потрібно запобігати певним потенційним проблемам та колізіям які можуть через це виникнути.

По-перше, потрібно запобігти можливому засміченню БД замовленнями які не будуть сплачені, або зроблені для спаму системи. Так було вирішено додати у процедуру створення нового запиту на спеціальний торт перевірку на те скільки вже тортів зі статусом “відправлено” є в користувача і якщо їх 5, то нові більше не будуть прийматися.

Також для замовлень які були оброблені адміністратором але не були сплачені користувачем існує функціонал видалення з БД після 7 днів існування запиту на замовлення. Код для перевірки статусу замовлення виконується раз на день та виглядає так:

```
cron.schedule('0 0 * * *', async () => {
```

```
try {
    const sevenDaysAgo = new Date();
    sevenDaysAgo.setDate(sevenDaysAgo.getDate() - 7);

    const specialCakesToDelete = await SpecialCake.find({
        status: 'received',
        createdAt: { $lte: sevenDaysAgo },});
    for (const specialCake of specialCakesToDelete) {
        await specialCake.remove();
        console.log(`Deleted special cake ${specialCake._id}`);
    }
} catch (error) {
    console.error('Error deleting old special cakes:', error);
}});
```

Другою можливою проблемою можуть бути колізії між двома різними адміністраторами які оброблятимуть замовлення. Вони бачитимуть список запитів які відправлені користувачами і перед тим як зв’язатися з користувачем змінюватимуть їх статус на “Запит отримано”, після чого напишуть їм на пошту. Якщо , наприклад через неоновлену сторінку списку замовлень, адміністратор спробує змінити статус на “Запит отримано” на замовленні, що вже оброблюється іншим адміністратором він отримає про це повідомлення.

```
if (status === 'received') {
            const specialCake = await SpecialCake.findById(id);
            if (specialCake.status === 'received') {
                return res.status(400).json({ message: 'Cake order
was already received.' });
            }
        }
```

Ці алгоритми допомагають зберегти плавний процес обробки замовлень та запобігають засміченню бази даних, що є критичними аспектами для будь-якого онлайн-бізнесу, особливо у сфері обробки замовлень. Наприклад, регулярне видалення застарілих спеціальних тортів забезпечує, що база даних залишається актуальною та не перевантаженою застарілими даними.

Крім того, використання cron для автоматичного видалення застарілих записів дозволяє підтримувати ефективну продуктивність системи та уникнути затримок у відповіді на нові замовлення. Такі підходи до керування даними допомагають забезпечити безперебійну роботу онлайн-магазину та забезпечують задоволення клієнтів шляхом швидкої та ефективної обробки їх замовлень.

# 4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

## 4.1 Вибір технологічного стеку та API інтеграцій

Для створення веб-застосунку "CakeShop" був обраний сучасний технологічний стек, який забезпечує гнучкість, продуктивність та масштабованість. Основною платформою для серверної частини виступає Node.js, що є популярною технологією завдяки своїй швидкості та ефективності.

Разом із Node.js використовується фреймворк Express.js, який значно спрощує створення RESTful API та обробку HTTP-запитів. Це дозволяє швидко розробляти та розгортати серверні функції, знижуючи складність розробки.

Для роботи з базою даних було обрано MongoDB, яка є документоорієнтованою NoSQL базою даних. MongoDB забезпечує гнучкість у зберіганні та обробці даних, що ідеально підходить для потреб веб-застосунку, який повинен обробляти різноманітні дані, пов'язані з тортами, випічкою та замовленнями. Для інтеграції MongoDB з Node.js використовується Mongoose, бібліотека ODM (Object Data Modeling), яка спрощує взаємодію з базою даних, надаючи зручний інтерфейс для роботи з моделями та схемами даних.

Аутентифікація та авторизація користувачів здійснюється за допомогою JSON Web Tokens (JWT). Ця технологія забезпечує безпечний спосіб передачі інформації про користувача між клієнтом і сервером, дозволяючи створювати безпечні сеанси користувачів. Використання JWT спрощує керування сесіями та забезпечує високий рівень безпеки, необхідний для роботи з чутливими даними користувачів[9].

Для завантаження та зберігання зображень використовується бібліотека Multer, яка дозволяє обробляти multipart/form-data, що є стандартом для завантаження файлів у веб-застосунках. Зображення зберігаються в Google Cloud Storage, що забезпечує надійне та масштабоване зберігання зображень. Використання хмарного сховища дозволяє знижувати навантаження на сервер та забезпечує швидкий доступ до зображень з будь-якого місця.

Конфігурація застосунку керується за допомогою бібліотеки dotenv, яка завантажує змінні середовища з файлу .env. Це дозволяє зручно керувати

конфігурацією та забезпечувати безпеку конфіденційних даних, таких як ключі доступу до бази даних та хмарних сервісів.

У додатку також використовуються спеціальні middleware для обробки запитів. Зокрема, CORS (Cross-Origin Resource Sharing) використовується для налаштування політик доступу до ресурсу з різних доменів, забезпечуючи безпечну взаємодію між клієнтами та сервером. Додатково, були створені власні middleware, такі як checkAuth та checkAdmin, які перевіряють аутентифікацію та права адміністратора відповідно.

Реалізовані API інтеграції забезпечують функціональність для управління користувачами, списком очікування, звичайною випічкою, особливими тортами, замовленнями, шаблонними тортами та смаками. Користувачі можуть реєструватися, входити в систему, отримувати свою інформацію, а також керувати своїми даними. Система підтримує додавання, оновлення та видалення записів у списку очікування, а також управління випічкою та тортами.

Завдяки використанню цих технологій та інтеграцій, веб-застосунок "CakeShop" забезпечує високий рівень продуктивності, безпеки та зручності для користувачів, дозволяючи ефективно керувати різноманітними аспектами кондитерського бізнесу в онлайні.

Процес впровадження програмного забезпечення для ПС "Cake Shop" складався з декількох ключових етапів, що забезпечили успішне розгортання сервісу на платформі Render. Основні кроки включали налаштування середовища розробки, інтеграцію з системою контролю версій Git, налаштування безперервної інтеграції та розгортання на віддаленому сервері.

Перший етап передбачав налаштування локального середовища розробки. Було створено необхідні репозиторії на GitHub для зберігання та управління вихідним кодом. Використання Git дозволило ефективно та зручно архівувати версії коду, та надавало можливість зв’язуватися з кодом з іншої частини, спочатку через localhost. Всі основні компоненти ПС, включаючи бекенд на Node.js та MongoDB для зберігання даних, були розміщені у відповідних репозиторіях (див.рис.4.1).

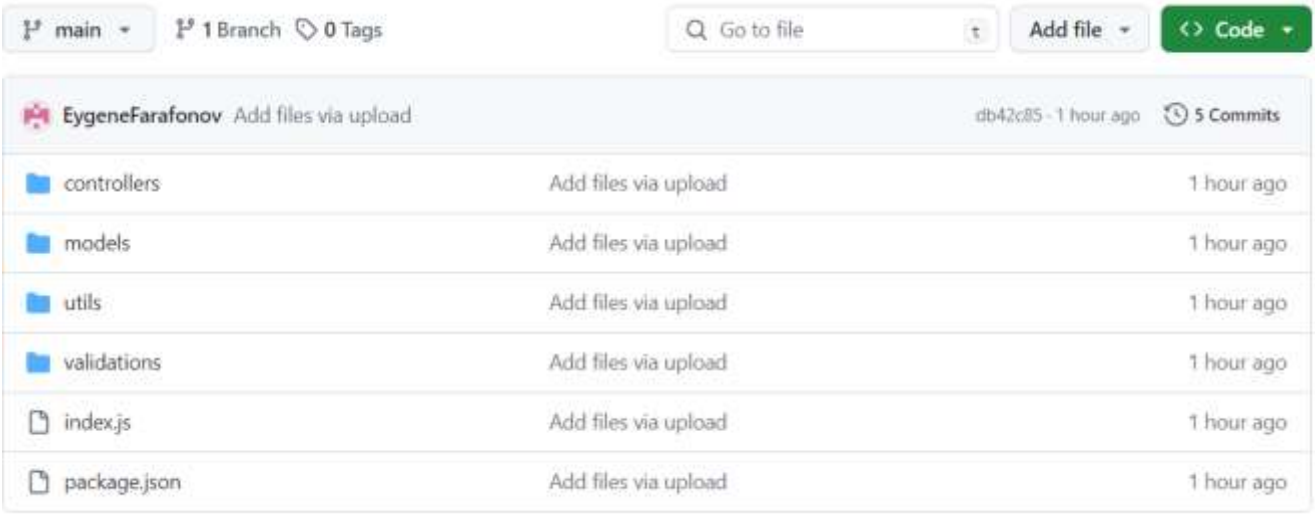


Рисунок 4.1 – Розташування необхідних даних на Git (рисунок виконаний самостійно)

Наступним важливим кроком було налаштування процесу безперервної інтеграції. Завдяки інтеграції з GitHub, кожне внесення змін у репозиторій автоматично проходило процес перевірки, тестування та підготовки до розгортання. Це забезпечувало виявлення та виправлення потенційних помилок ще на етапі розробки, що значно підвищувало якість кінцевого продукту.

Фінальним етапом було розгортання на платформі Render(див.рис.4.2).

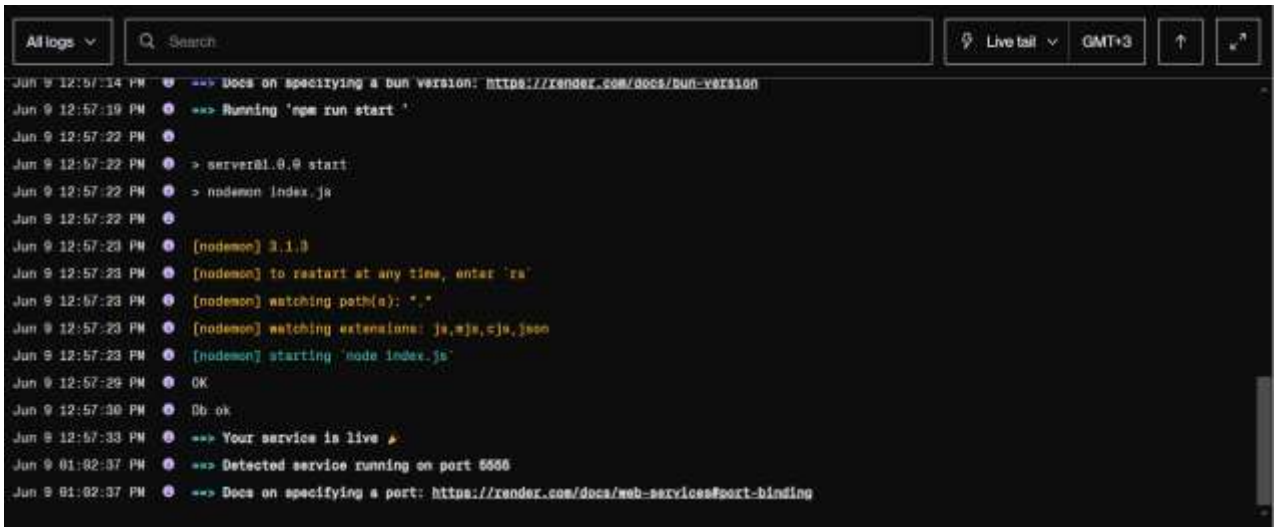


Рисунок 4.2 – Успішний запуск серверу на Render (рисунок виконаний самостійно)

Ця платформа має достатньо простий засіб розгортання web-сервісу через інтеграції з GitHub. Репозиторій ПС був успішно завантажений на Render, а також підвантажені значення констант середовища, та секретні файли. Після цього відбулося автоматичне налаштування серверного середовища, необхідного для роботи додатку (див.рис.4.2).

Серверний додаток був успішно запущений, і отримав публічну адресу https://test-e3px.onrender.com, що дозволило фронт частині отримати доступ до веб-додатку.

## 4.2 Управління даними

Управління даними в веб-застосунку "CakeShop" базується на використанні MongoDB як головної бази даних. MongoDB, як NoSQL база даних, пропонує гнучкість у зберіганні даних, що особливо важливо для динамічних веб-застосунків.

Дані передаються у вигляді документів JSON, що дозволяє легко маніпулювати ними та забезпечує високу продуктивність при обробці великих обсягів інформації. Для роботи з MongoDB використовується Mongoose – об'єктно-документний моделювальний інструмент (ODM).Він дозволяє зручно працювати з моделями та схемами даних, забезпечуючи валідацію, прив'язку бізнес-логіки до моделей, а також простий спосіб для реалізації CRUD (створення, читання, оновлення, видалення) операцій.

Використання Mongoose значно спрощує управління даними в MongoDB, забезпечуючи чистий і зрозумілий код, а також додатковий рівень абстракції, що дозволяє зосередитися на бізнес-логіці замість роботи з базою даних на низькому рівні.

### 4.2.1 Контроллери

Ключову роль в управлінні даними відіграють контролери, які обробляють запити користувачів і взаємодіють з базою даних. Кожен контролер має набір

методів для обробки CRUD операцій (створення, читання, оновлення та видалення) і реалізує бізнес-логіку для відповідної сутності. Наприклад, контролер для управління замовленнями включає методи для створення нового замовлення, отримання деталей замовлення за його ідентифікатором, оновлення статусу замовлення та видалення замовлення. Аналогічно, контролери для управління користувачами, тортами та іншими сутностями забезпечують відповідну функціональність для обробки запитів, пов'язаних з цими об'єктами.

Кожен метод контролера ретельно обробляє вхідні дані, виконує необхідні перевірки та валідації, а потім взаємодіє з моделями Mongoose для виконання запитів до MongoDB. Наприклад, при створенні нового замовлення контролер перевіряє наявність необхідних даних, таких як список тортів і інформація про користувача, а потім використовує моделі для збереження нового замовлення в базі даних. Під час оновлення замовлення контролер перевіряє нові дані на відповідність встановленим правилам і оновлює відповідний документ в базі даних.Нижче розглянуто основні контролери та їхні завдання:

а) UserController:
   - register: реєструє нового користувача, зберігаючи його дані в колекції користувачів. Валідує дані та хешує пароль перед збереженням;
   - login: аутентифікує користувача, перевіряючи його електронну пошту та пароль, після чого генерує та повертає JWT токен;
   - userInf: повертає інформацію про поточного користувача на основі переданого JWT токена;
   - updateUser: новлює дані користувача, дозволяючи змінювати персональну інформацію та налаштування;

б) RegularBakeryController:
   - createRegularBakery: додає новий продукт до колекції звичайної випічки. Включає інформацію про назву, опис, категорію та ціну продукту;
   - deleteRegularBakery: видаляє продукт з колекції звичайної випічки за його ідентифікатором;

- updateRegularBakery: оновлює дані про продукт звичайної випічки, дозволяючи змінювати його атрибути;
- getAllRegularBakery: повертає всі продукти звичайної випічки.
- getRegularBakeryById: повертає конкретний продукт звичайної випічки за його ідентифікатором;
- getRegularBakeryByCategory: повертає продукти звичайної випічки, відфільтровані за категорією;

в) SpecialCakeController:

- createSpecialCake: додає новий особливий торт до колекції. Включає детальну інформацію про інгредієнти, декор та спеціальні побажання клієнта;
- deleteSpecialCake: видаляє особливий торт за його ідентифікатором;
- updateSpecialCake: оновлює інформацію про особливий торт;
- getAllSpecialCakes: повертає всі особливі торти;
- getSpecialCakeById: повертає конкретний особливий торт за його ідентифікатором;
- getSpecialCakesByUserId: повертає всі особливі торти, замовлені конкретним користувачем;

г) OrderController:

- createOrder: створює нове замовлення, включаючи деталі про користувача, продукти, кількість та ціну;
- findAllOrders: повертає всі замовлення, що дозволяє адміністратору переглядати всі активні та минулі замовлення;
- findOrderById: повертає конкретне замовлення за його ідентифікатором;
- updateOrder: оновлює деталі замовлення;
- deleteOrder: видаляє замовлення за його ідентифікатором;
- findOrdersByUserId: повертає всі замовлення, зроблені конкретним користувачем;

д) ShablonCakeController:

- create: додає новий шаблонний торт до колекції. Включає деталі про стандартний дизайн та інгредієнти;
- findAll: повертає всі шаблонні торти;
- findOneById: повертає конкретний шаблонний торт за його ідентифікатором;
- update: оновлює інформацію про шаблонний торт;
- remove: видаляє шаблонний торт за його ідентифікатором;

е) TasteController:

- createTaste: додає новий смак до колекції. Включає деталі про інгредієнти та поєднання смаків;
- deleteTaste: видаляє смак за його ідентифікатором;
- updateTaste: оновлює інформацію про смак;
- getAllTastes: повертає всі доступні смаки;
- getTastesByCakeId: повертає всі смаки, які підходять до конкретного торта.

Далі наведено приклад одного з методів контролеру для звичайної випічки, функція додавання нової випічки у БД.

```
import RegularBakery from '../models/RegularBakeryModel.js';

export const createRegularBakery = async (req, res) => {
    try {
        console.log('Request to create bakery:', req.body);
           const { category, name, description, weight, price, img } =
     req.body;

        if (!category || !name || !description || !weight || !price) {
            return res.status(400).json({ message: 'Category, name,
     description, weight, and price are required.' });
        }

        const newRegularBakery = new RegularBakery({
            category,
            name,
            description,
            weight,
            price,
            img,
        });

        const savedBakery = await newRegularBakery.save();
```

```
            res.status(201).json(savedBakery);
        } catch (error) {
            console.error('Error creating regular bakery:', error);
            res.status(500).json({ message: 'Internal server error' });
        }
};
```

Метод перевіряє заповнення необхідних даних, проводить збереження даних у MongoDB повертає збережений об’єкт в разі успіху.

### 4.2.1 API інтеграції

Для забезпечення надійної та ефективної роботи веб-додатку "Cake Shop" було впроваджено ряд API інтеграцій, які дозволяють автоматизувати ключові процеси, забезпечити збереження даних та покращити користувацький досвід. Основні інтеграції включають Google Cloud Storage, Insomnia, GitHub Desktop та Render.

Однією з ключових інтеграцій є Google Cloud Storage, який використовується для зберігання зображень тортів та інших медіафайлів, що завантажуються користувачами. Це забезпечує надійне та масштабоване рішення для зберігання даних, а також дозволяє швидко отримувати доступ до файлів через HTTP-запити. Завдяки цьому, користувачі можуть завантажувати та переглядати зображення з високою швидкістю та надійністю.

Для тестування та налагодження API використовується Insomnia, потужний інструмент для роботи з HTTP-запитами. З його допомогою розробники можуть швидко перевіряти роботу різних кінцевих точок API, відстежувати відповіді сервера та виявляти можливі помилки. Insomnia дозволяє ефективно тестувати всі аспекти API, що суттєво полегшує процес розробки та підтримки додатку.

GitHub Desktop забезпечує зручну та ефективну роботу з системою контролю версій Git. Завдяки цьому інструменту, можна легко керувати своїми репозиторіями, відстежувати зміни у вихідному коді та влаштовувати співпрацю між учасниками команди. GitHub Desktop спрощує процес інтеграції та

розгортання, забезпечуючи безперервний робочий процес та збереження всіх змін в одному місці.

Render використовується для розгортання та хостингу серверної частини додатку. Ця платформа дозволяє автоматично розгортати додаток безпосередньо з репозиторію GitHub, забезпечуючи безперервну інтеграцію та доставку. Завдяки Render, серверний додаток був успішно розгорнутий та отримав публічну адресу https://test-e3px.onrender.com, що дозволило користувачам отримати доступ до веб-додатку в режимі реального часу.

Загалом, впровадження цих API інтеграцій дозволило створити надійний та масштабований веб-додаток з високою продуктивністю. Використання сучасних інструментів та платформ забезпечило безперебійний робочий процес, високу швидкість обробки даних та зручність для користувачів.

# 5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування Backend частини відбувалося у 3 етапи: пряме надсилання тестування запитів за допомогою застосунку Insomnia на localhost, потім по веб-посиланню та тестування надсилання запитів з Frontend частини для перевірки цілісності зв'язку між частинами програмної системи(див. рис. 5.1):

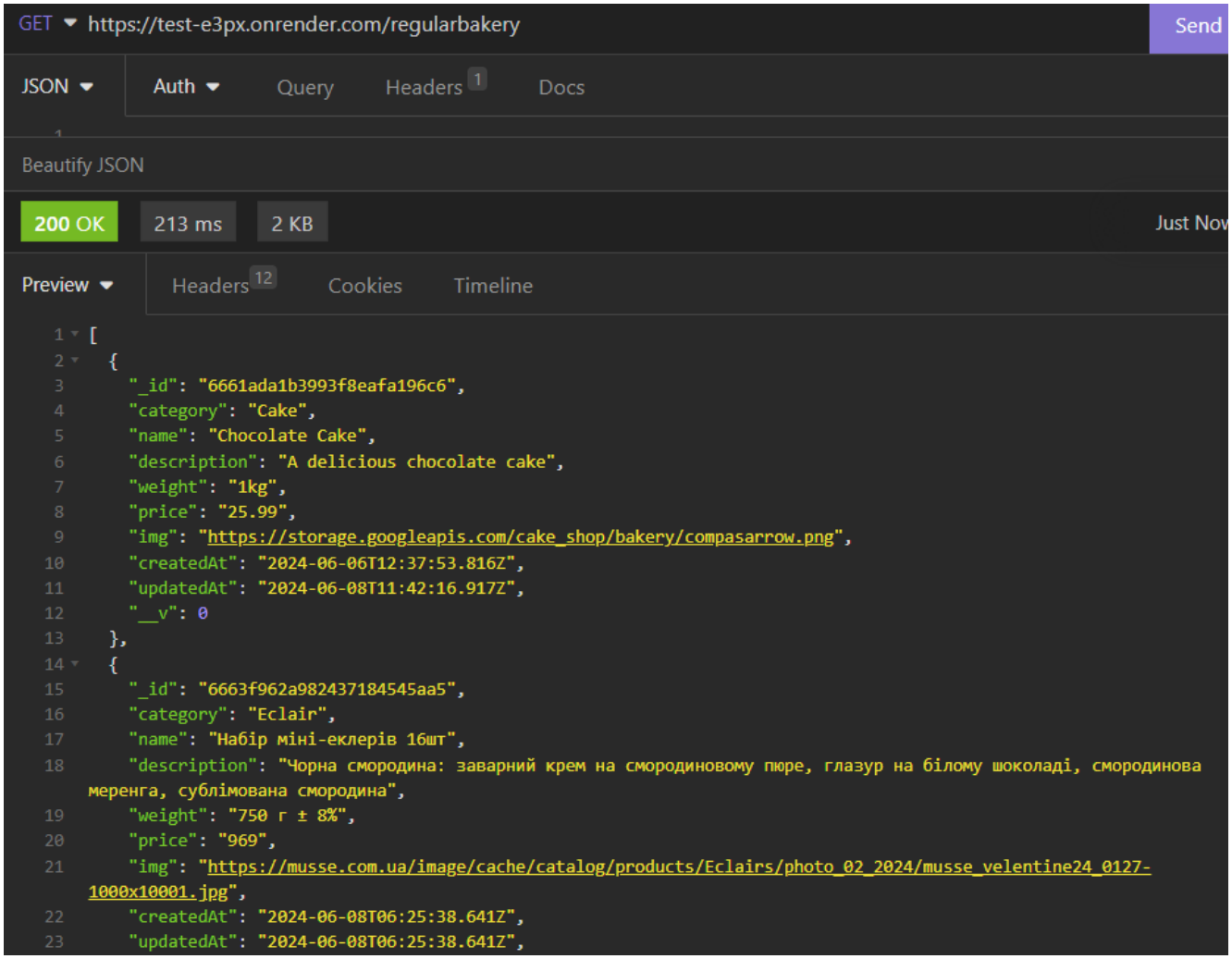


Рисунок 5.1 – Запит в Insomnia до веб-сервісу(рисунок виконаний самостійно)

Використання Insomnia було необхідним для перевірки функціонування методів до того як їх виклик буде реалізований на Frontend частині застосунку.

У рамках тестування серверної частини відбувалося виключно функціональне тестування [10]. Далі буде наведено приклад тест-кейсу які

виконувалися після реалізації нового функціоналу, для того щоб побачити які недоробки могли залишитись непоміченими і чи не ламає новий функціонал вже працюючі правильно частини коду.

Наступний приклад зображує процес тестування операції по замовленню торту за шаблоном покупцем з авторизованого аккаунту. Ціллю тесту було побачити коректність передачі інформації, в тому числі зображень(табл. 5.1).

Таблиця 5.1 – Тест-кейс №1 (таблиця виконана самостійно)

| Інформація про тест-кейс | | | |
|---|---|---|---|
| Ідентифікатор тесту: | Тест-кейс №1 | | |
| Опис функції: | Замовлення торта по шаблону | | |
| Власник тесту: | Фарафонов Євгеній Юрійович | | |
| Дата створення: | 06.06.2024 | | |
| Мета тесту: | Перевірити коректність процесу замовлення тортів користувачем | | |
| Передумова | | | |
| № | Опис випадку | Очікуваний результат | Висновок |
| 1 | Відкрити веб-застосунок | Користувач має доступ до сайту, який відкритий | Пройдено |
| 2 | Має обліковий запис та авторизований | Перенаправлено на головну сторінку | Пройдено |
| Створення заявки на фінансову допомогу | | | |
| № | Опис випадку | Очікуваний результат | Висновок |
| 1 | Відкрити сторінку переліку тортів-шаблонів. | З’являється сторінка зі списком усіх шаблонних тортів | Пройдено |
| 2 | Натиснути на потрібний торт | Перенаправлено на сторінку обраного торту | Пройдено |
| 3 | Обрати вагу | Обрано вагу зі списку | Пройдено |
| 4 | Обрати смак торту | Обрано смак зі списку. Змінюється картинка смаку змінюється опис смаку. | Пройдено |
| 5 | Заповнити поле «Текст на торті» | Заповнено правильно поле «Текст на торті» | Пройдено |
| 6 | Додати торт у корзину | Торт відображається у козині | Пройдено |
| 7 | Додати той самий торт | Кількість торту збільшена на 1 | Пройдено |
| 8 | Додати той самий торт з іншим смаком | Відображається ще один новий торт | Пройдено |

Продовження таблиці 5.1

| | | | |
|---|---|---|---|
| 9 | Додати той самий торт з іншою вагою | Відображається ще один новий торт | Пройдено |
| 10 | Додати той самий торт з незаповненим полем «Текст на торті» | Відображається ще один новий торт | Пройдено |
| 11 | Додати інший торт | Відображається ще один торт | Пройдено |
| 12 | Натиснути кнопку «Корзина» | Відображаються усі торти та їх кількість правильно відображається загальна ціна | Пройдено |
| 13 | Оплатити замовлення | Відбувається успішна оплата замовлення. | Пройдено |
| 14 | Переглянути замовлення | Відображається замовлення відповідно до попередньо заповнених даних | Пройдено |
| 15 | Написати більше 30 символів у полі «Текст на торті» | Відображається помилка «Забагато символів для напису». Торт не можливо додати в корзину | Пройдено |
| 16 | Перервати процес оплати замовлення. | Замовлення не додається до списку | Пройдено |
| Результати тестування | | | |
| Тестувальник | | Дата прогону тесту | Результат тесту (P/F/B) |
| Фарафонов Є.Ю. | | 06.06.2024 | ПРОЙДЕНО (P) |

В результаті проходження тест-кейсу вдалося впевнитися в повноцінній коректності функцій що супроводжують процес замовлення торта по шаблону, в тому числі їх резистентність до неправильно введених даних.

Наступний тест-кейс, це повне проходження процесу необхідного для додавання нового шаблонного торту користувачем-адміністратором. Адміністратор має бути попередньо авторизованим на сайті для доступу до додавання та редагування будь-яких полів(табл. 5.2).

Таблиця 5.2 – Тест-кейс №2 (таблиця виконана самостійно)

| Інформація про тест-кейс | |
|---|---|
| Ідентифікатор тесту: | Тест-кейс №2 |
| Опис функції: | Додавання нового торта-шаблону |
| Власник тесту: | Фарафонов Євгеній Юрійович |
| Дата створення: | 06.06.2024 |

Продовження таблиці 5.2

| Мета тесту: | Перевірити коректність процесу додавання адмінітратором нового торта-шаблону | | |
|---|---|---|---|
| Передумова | | | |
| № | Опис випадку | Очікуваний результат | Висновок |
| 1 | Відкрити веб-застосунок | Адміністратор має доступ до сайту, який відкритий | Пройдено |
| 2 | Має обліковий запис та авторизований | Перенаправлено на головну сторінку | Пройдено |
| Створення заявки на фінансову допомогу | | | |
| № | Опис випадку | Очікуваний результат | Висновок |
| 1 | Відкрити сторінку переліку тортів-шаблонів. | З’являється сторінка зі списком усіх шаблонних тортів | Пройдено |
| 2 | Натиснути на кнопку створення нового торту | Перенаправлено сторінку створення нового торту | Пройдено |
| 3 | Заповнити назву опис вагу та ціну | Правильно заповнені поля З назва опис вагу та ціну | Пройдено |
| 4 | Додати смак торту | Додати назву опис та картинку смаку | Пройдено |
| 5 | Переглянути список тортів за шаблоном | Відображується новий торт | Пройдено |
| 6 | Додати новий торт але жодного смаку | Торт відображається у адміністратора але не користувача | Пройдено |
| Результати тестування | | | |
| Тестувальник | Дата прогону тесту | Результат тесту (P/F/B) | |
| Фарафонов Є.Ю. | 06.06.2024 | ПРОЙДЕНО (P) | |

Тестування показало, що веб-додаток відповідає усім заявленим вимогам і готовий до використання кінцевими користувачами. Програмне забезпечення виявилося стабільним та надійним, здатним обробляти запити користувачів без збоїв та помилок. Завдяки проведеному тестуванню були виявлені та виправлені незначні недоліки, що підвищило загальну якість та надійність системи. Загальний результат тестування є позитивним, і додаток готовий до запуску та подальшої експлуатації.

# ВИСНОВКИ

В результаті роботи був проведений аналіз предметної галузі та конкурентного середовища. Було визначено поточний стан ринку вироблення кондитерських виробів на замовлення, визначено які особливості ПС роблять її конкурентоспроможною, та вирізняють її з поміж альтернативних варіантів. Зокрема, проаналізовано основних конкурентів, їхні стратегії та технологічні рішення, що дозволило виявити ключові аспекти для покращення власного продукту.

Відповідно до аналізу були сформовано характеристики цільової аудиторії – потенційних користувачів, специфіка бізнес моделі, проблеми та можливість їх вирішення. Це включало визначення вікових категорій, професій, вподобань та поведінкових особливостей користувачів, що дозволило більш точно налаштувати функціонал та інтерфейс системи. Сформовані функціональні та нефункціональні вимоги до програмного забезпечення, виконання яких забезпечить якісний програмний продукт, необхідний для ведення онлайн магазину кондитерських виробів виготовлених на замовлення.

Для безпосереднього виконання завдання по розробці програмної системи був обраний стиль архітектури клієнт-сервер з імплементацією схеми Model-View-Controller для розподілення даних, де Backend частина відповідає за аспекти Model та Controller. Клієнтська частина відповідає за представлення (View) та взаємодію з користувачем, що дозволяє зменшити навантаження на сервер та покращити швидкодію системи.

Backend був розроблений на Node.js з використанням фреймворку Express та по методології REST при програмуванні методів. Це забезпечує модульність, простоту в розширенні функціоналу та високу продуктивність системи. Для збереження даних використовувалася БД MongoDB. Структура БД визнається та формується створеними на серверній частині моделями, що дозволяє гнучко змінювати структуру даних та швидко адаптуватися до нових вимог.

Сервер проводить валідацію отриманих даних, перевіряє дані при авторизації за допомогою токенів, які шифрують інформацію про користувача. Це забезпечує

захист від несанкціонованого доступу та підвищує безпеку системи. Збереження даних про пароль користувача відбувається за допомогою хешування даних для додаткової безпеки, що мінімізує ризики витоку паролів.

Для зберігання зображень кондитерських виробів було інтегровано Google Cloud Storage, що дозволило забезпечити надійне та масштабоване рішення для зберігання медіафайлів. Це рішення забезпечує високу доступність та безпеку даних, що є критично важливим для роботи онлайн-магазину. Використання Google Cloud Storage також дозволяє оптимізувати витрати на зберігання даних завдяки гнучким тарифним планам та можливості автоматичного масштабування.

Серверна частина додатку була успішно розгорнута на платформі Render через Git-репозиторій, що дозволило забезпечити безперервну інтеграцію та доставку (CI/CD) програмного забезпечення. Завдяки Render, сервер автоматично оновлюється з новими змінами в коді, що спрощує процес розгортання та адміністрування додатку. Сервіс отримав адресу https://test-e3px.onrender.com, що дозволяє легко доступатися до сервера та перевіряти його працездатність у режимі реального часу. Такий підхід значно скорочує час на оновлення системи та мінімізує ризики простоїв, що критично важливо для підтримки високого рівня сервісу в онлайн-магазині.

Таким чином у результаті розробки створено серверну частину програмної системи для продажу кондитерських виробів виготовлених на замовлення, яка реалізує функціонал для додавання та збереження нових тортів, оформлення та оплати замовлень, відстеження статусу замовлення.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. publisher B.-n. Bakery Order Log Book: Daily Bakery Order Form Log Book for Professional Bakery Businesses - Customer Order Tracker for Cakes, Cookies, Brownies, and More. Independently Published, 2021. 110 c.

2. Instagram магазин The Cake KH (2023/2024) [Електронний ресурс] – URL: https://www.instagram.com/the.cake.kh/ (дата звернення: 12.05.2024)

3. Магазин Cake Shoko (2023/2024) [Електронний ресурс] – URL: https://cake.shoko.com.ua/uk/ (дата звернення: 12.05.2024)

4. Інтернет магазин Кулиничі (2024) [Електронний ресурс] – URL: https:// www.kulinichi.com (дата звернення: 26.05.2024)

5. Dubovyk T. CONCEPTUAL MODEL OF CONSUMERS TRUST TO ONLINE SHOPS. Bulletin of Taras Shevchenko National University of Kyiv Economics. 2014. № 160. С. 33–37. URL: https://doi.org/10.17721/1728-2667.2014/160-7/7 (дата звернення: 17.06.2024).

6. Oluwatosin H. S. Client-Server Model. IOSR Journal of Computer Engineering. 2014. Т. 16, № 1. С. 57–71. URL: https://doi.org/10.9790/0661-16195771 (дата звернення: 17.06.2024).

7. Tyagi P. Integrating REST API. Pragmatic Flutter. Boca Raton, 2021. С. 189–202. URL: https://doi.org/10.1201/9781003104636-12 (дата звернення: 17.06.2024).

8. Sharma M. MongoDB Complete Guide: Develop Strong Understanding of Administering MongoDB, CRUD Operations, MongoDB Commands, MongoDB Compass, MongoDB Server, ... and MongoDB Sharding. BPB Publications, 2021. 470 p.

9. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC Editor, 2015. URL: https://doi.org/10.17487/rfc7519 (дата звернення: 17.06.2024).

10. Mosley D. F. Client-server user-interface testing. IEEE Software. 1995. Т. 12, № 1. С. 124–127. URL: https://doi.org/10.1109/52.363159 (дата звернення: 17.06.2024).

# ДОДАТОК А

## Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

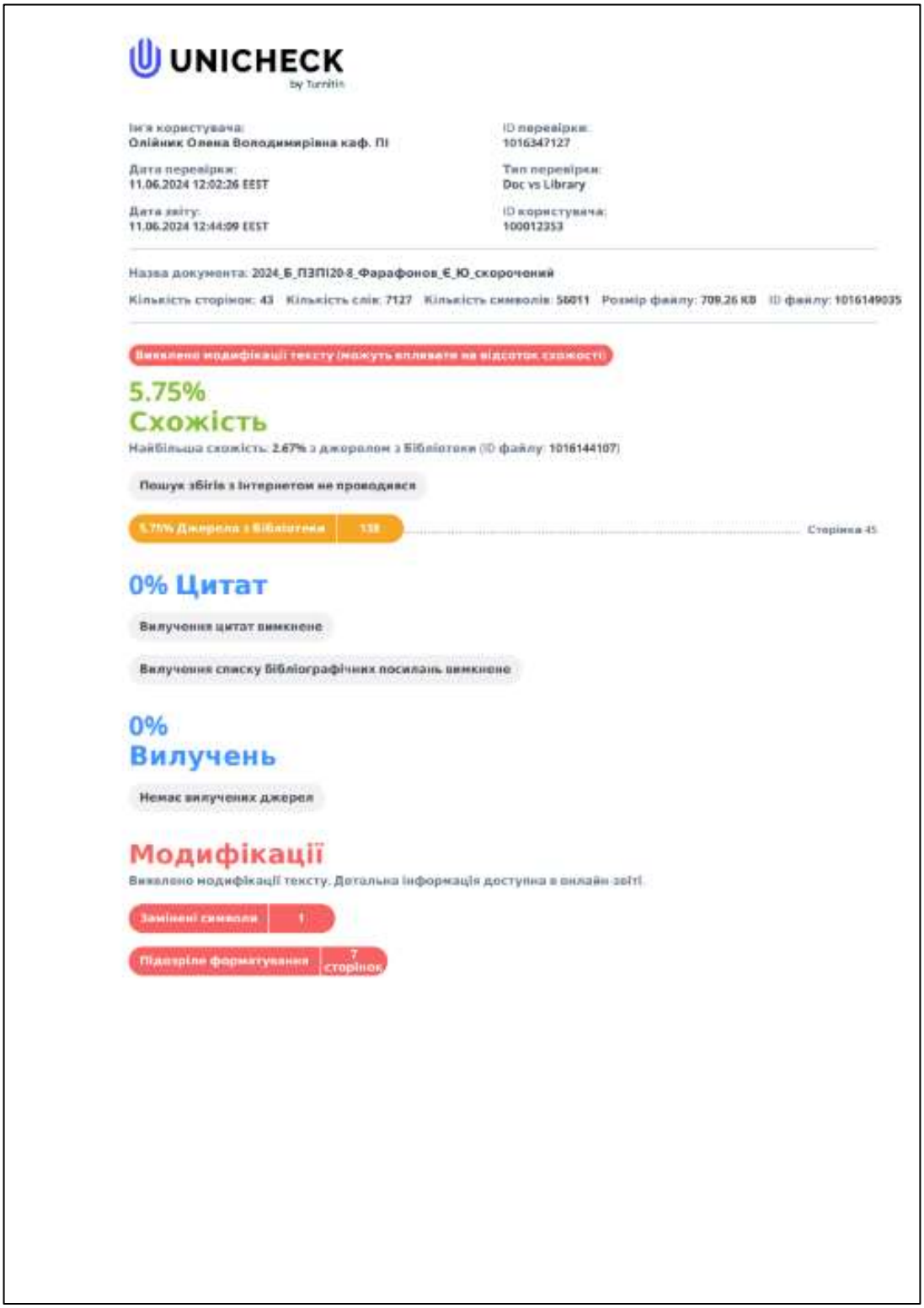


Рисунок А.1 ¬ Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ (рисунок виконаний самостійно)

# ДОДАТОК Б

## Слайди презентації

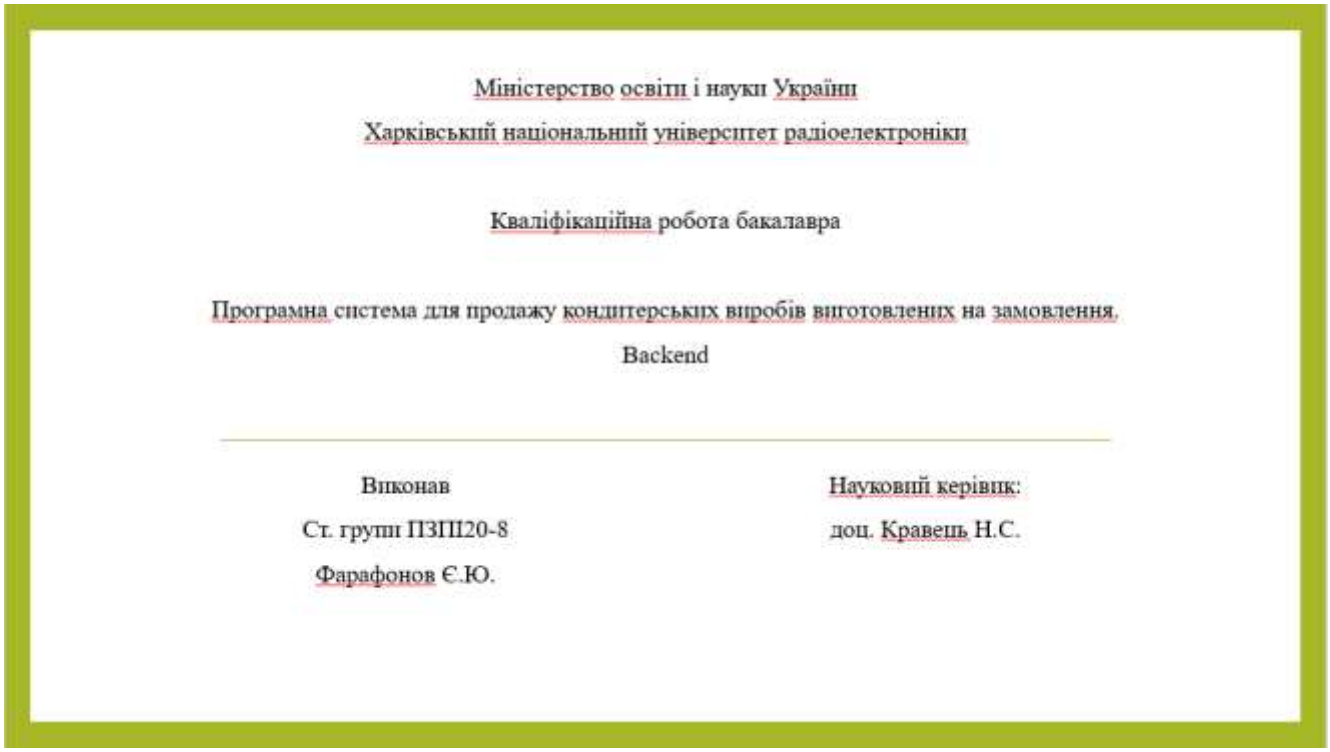


Рисунок А.1 – Титульний слайд (рисунок виконаний самостійно)


Рисунок А.2 – Слайд вступ(рисунок виконаний самостійно)


Рисунок А.3 – Слайд актуальність теми (рисунок виконаний самостійно)

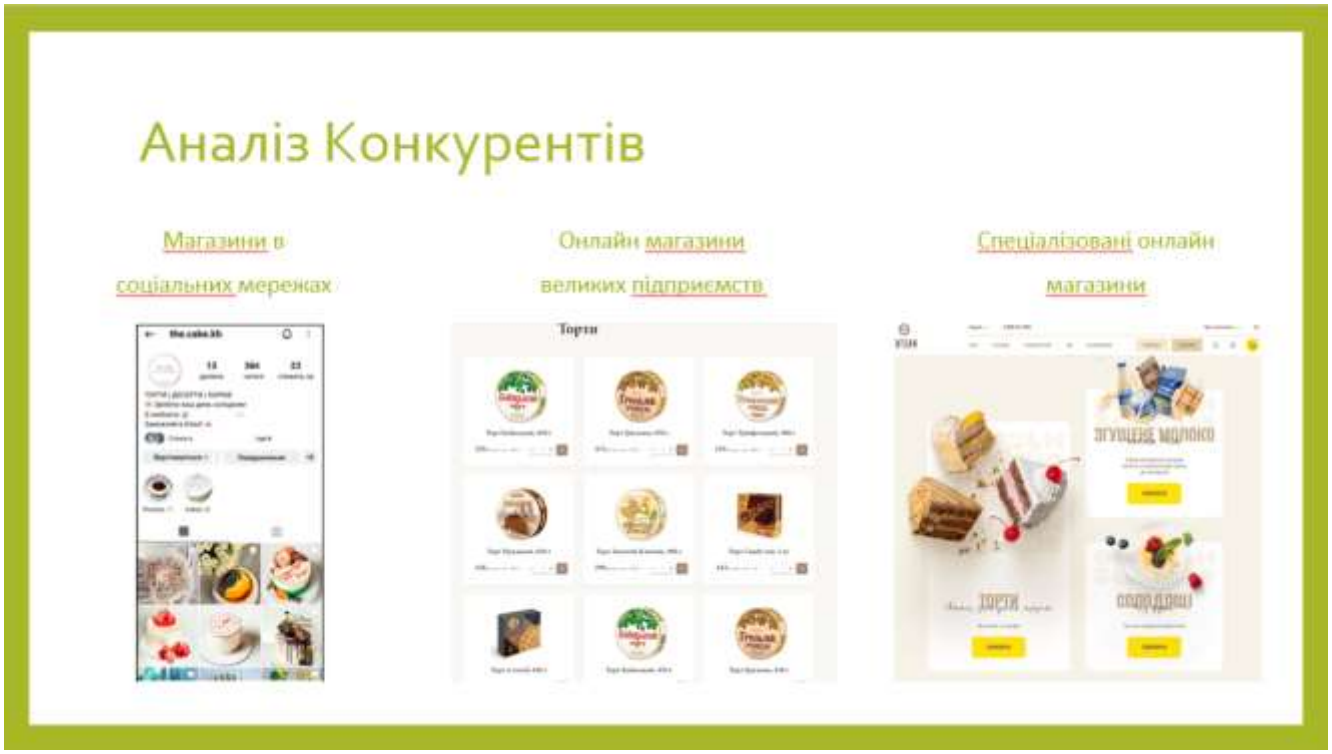


Рисунок А.4 – Слайд аналіз конкурентів (рисунок виконаний самостійно)


Рисунок А.5 – Слайд цільова аудиторія (рисунок виконаний самостійно)

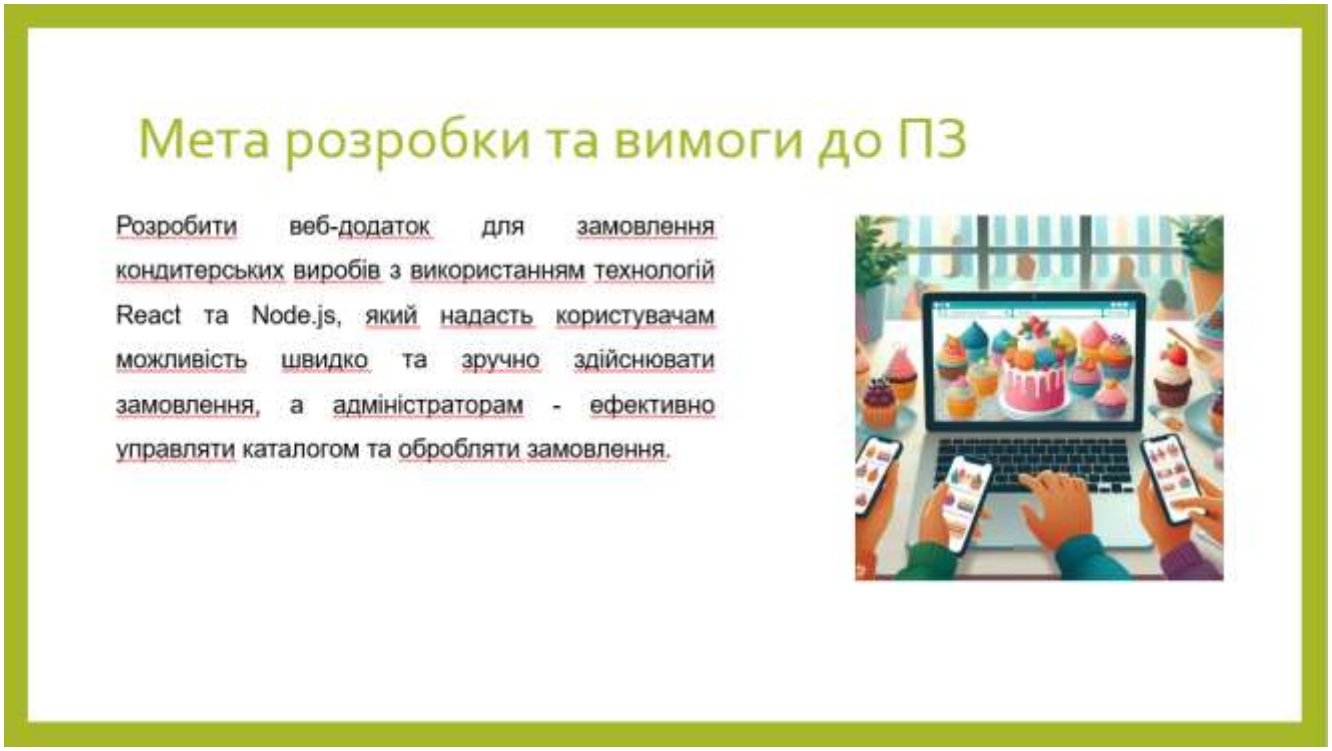


Рисунок А.6 – Слайд мета розробки (рисунок виконаний самостійно)

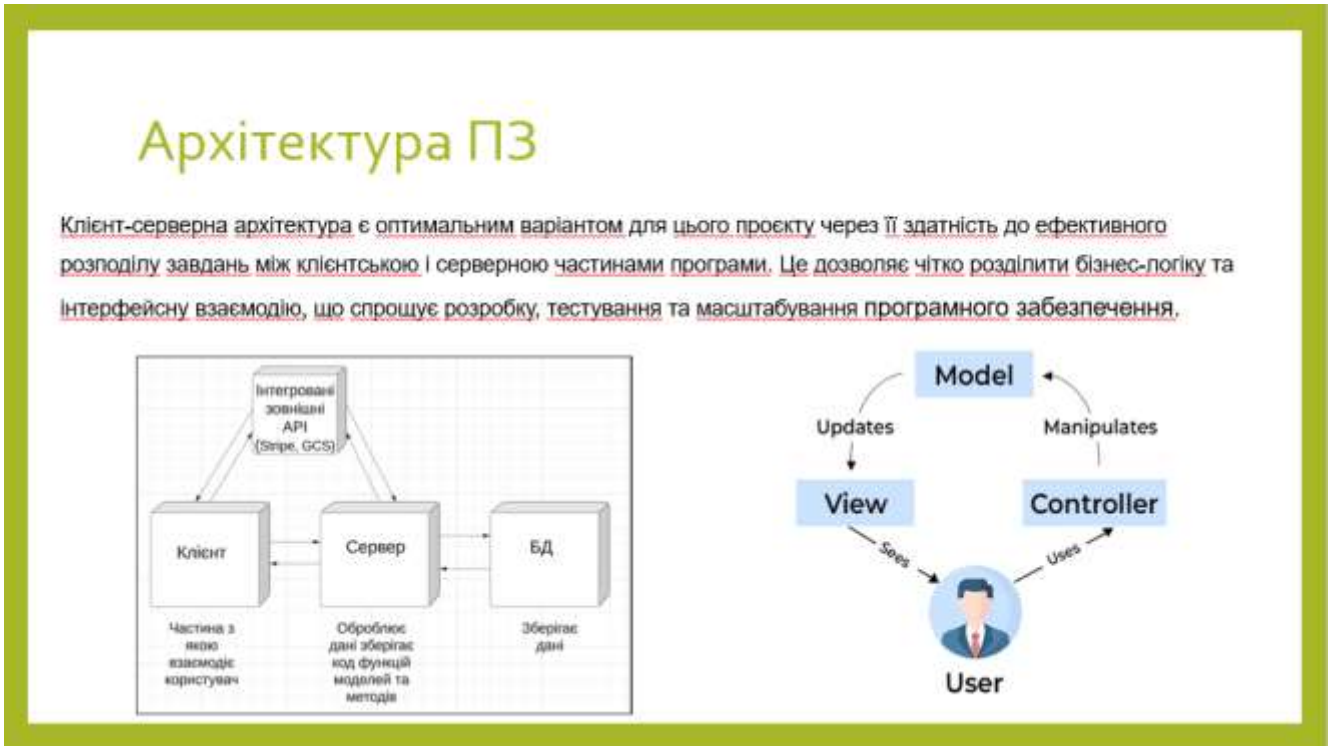


Рисунок А.7 – Слайд архітектура ПЗ(рисунок виконаний самостійно)


Рисунок А.8 – Слайд технології розробки (рисунок виконаний самостійно)


Рисунок А.9 – Слайд маршрутизація та REST (рисунок виконаний самостійно)

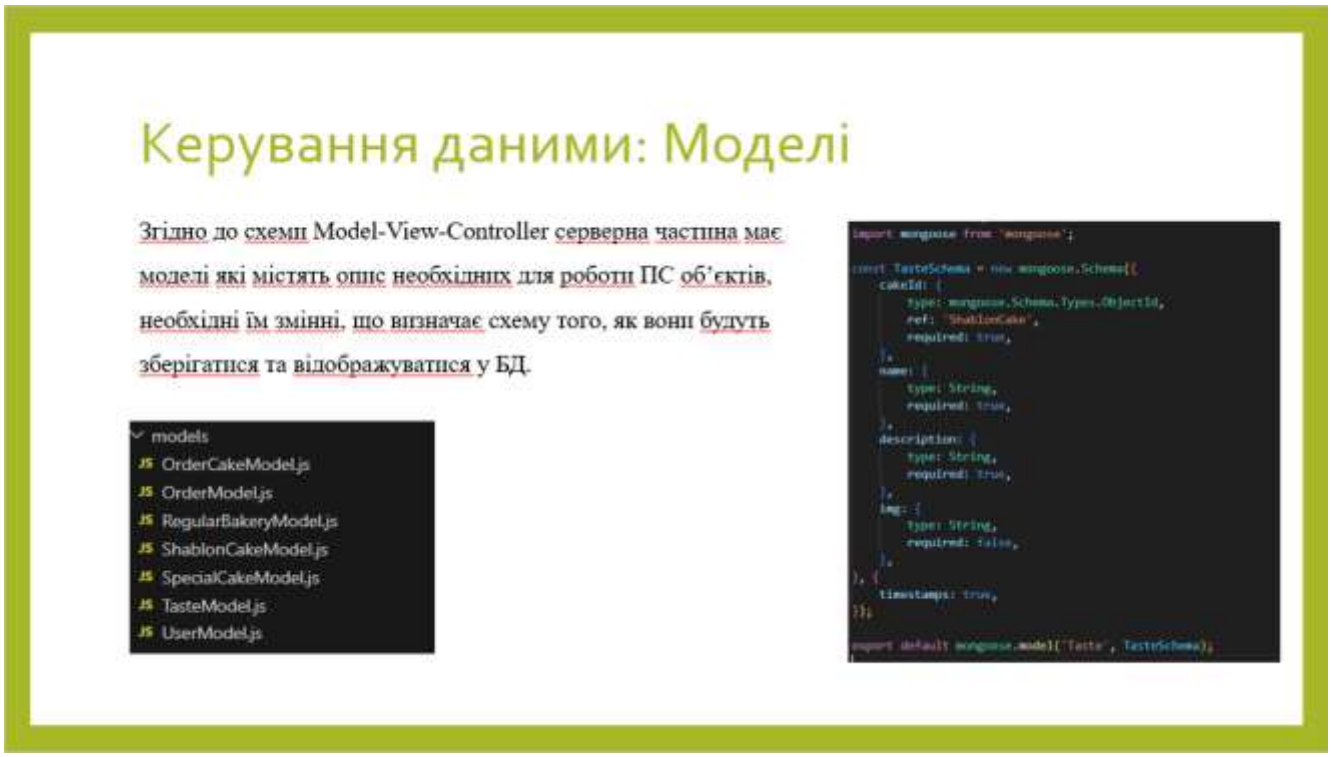


Рисунок А.10 – Слайд моделі (рисунок виконаний самостійно)

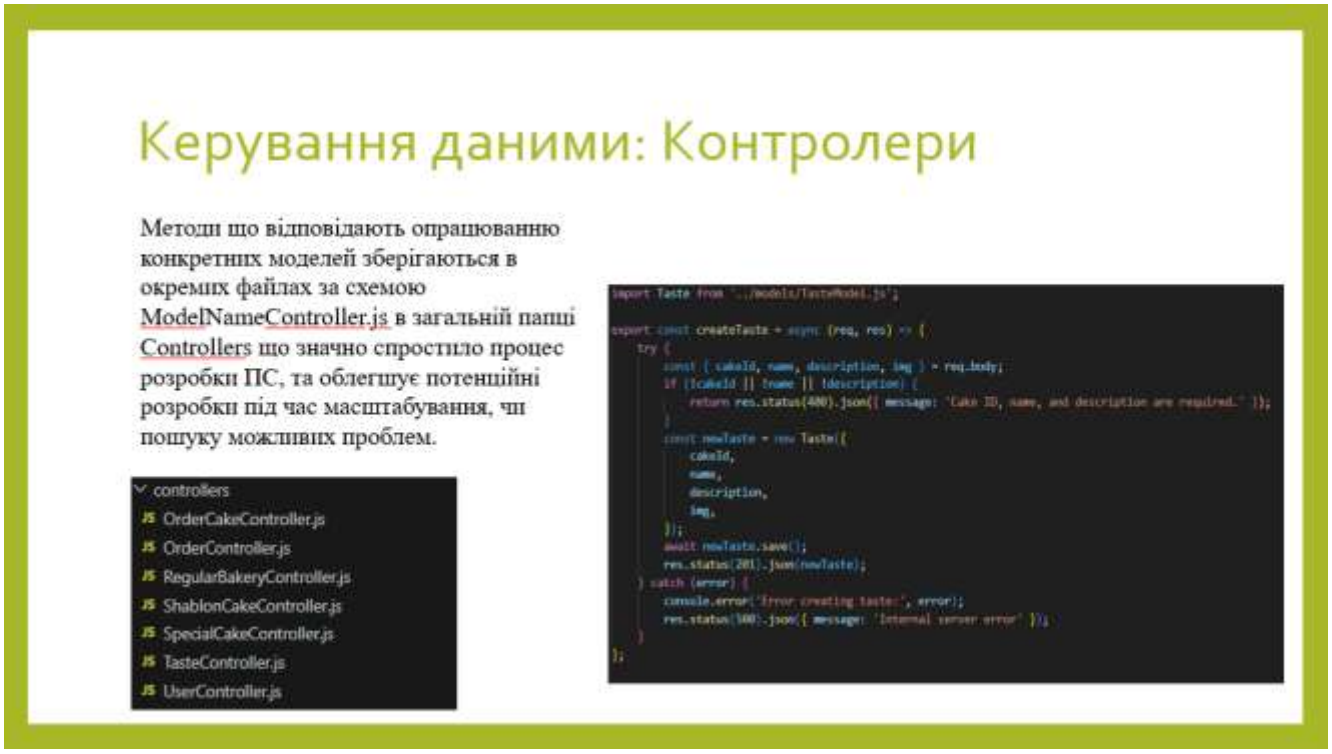


Рисунок А.11 – Слайд контролери (рисунок виконаний самостійно)

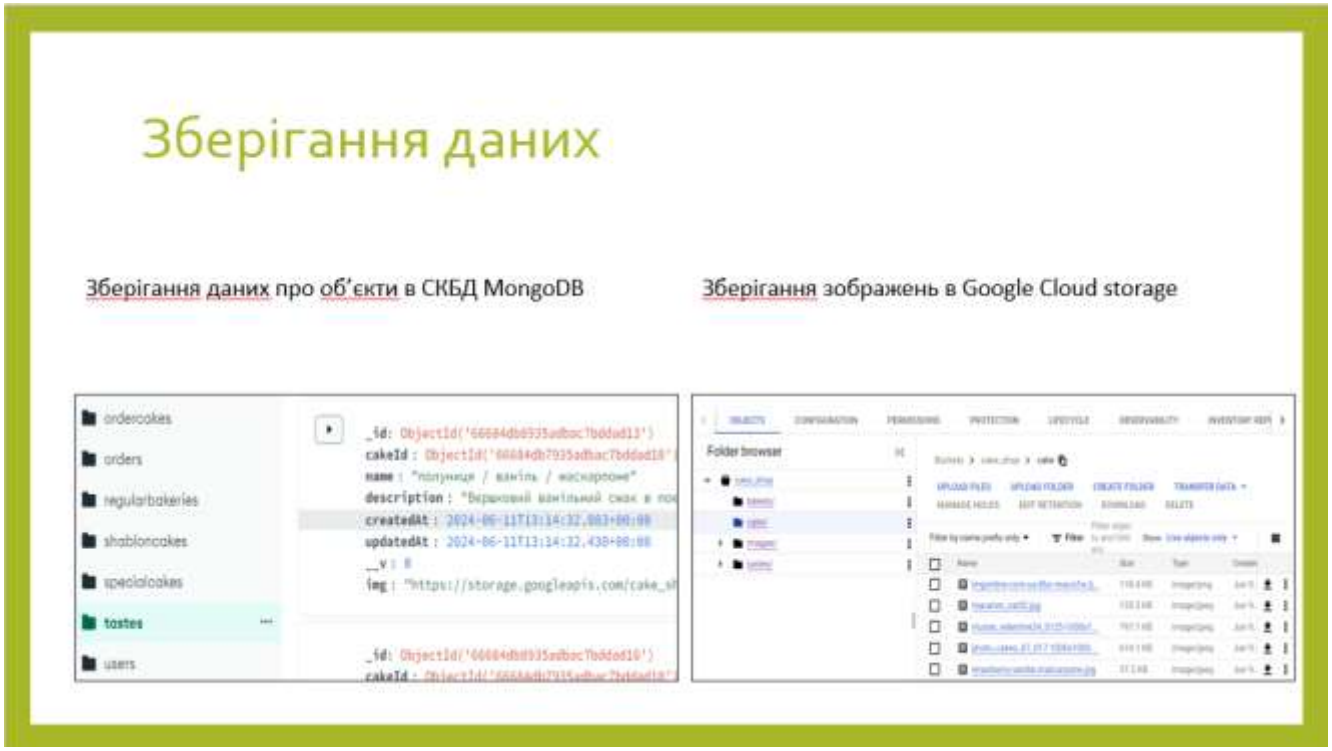


Рисунок А.12 – Слайд зберігання даних (рисунок виконаний самостійно)

## Безпека даних та middleware

Для забезпечення безпеки даних і підвищення надійності серверної програми важливо використовувати різноманітні middleware.

1) Аутентифікація та авторизація
2) Обробка помилок
3) Захист від атак
4) Обробка валідації даних

Рисунок А.13 – Слайд безпека даних (рисунок виконаний самостійно)

## Тестування

Тестування Backend частини відбувалося у 3 етапи: пряме надсилання тестування запитів за допомогою застосунку Insomnia на localhost, потім по веб-посиланню та тестування надсилання запитів з Frontend частини для перевірки цілісності зв'язку між частинами програмної системи

Рисунок А.14 – Слайд тестування (рисунок виконаний самостійно)

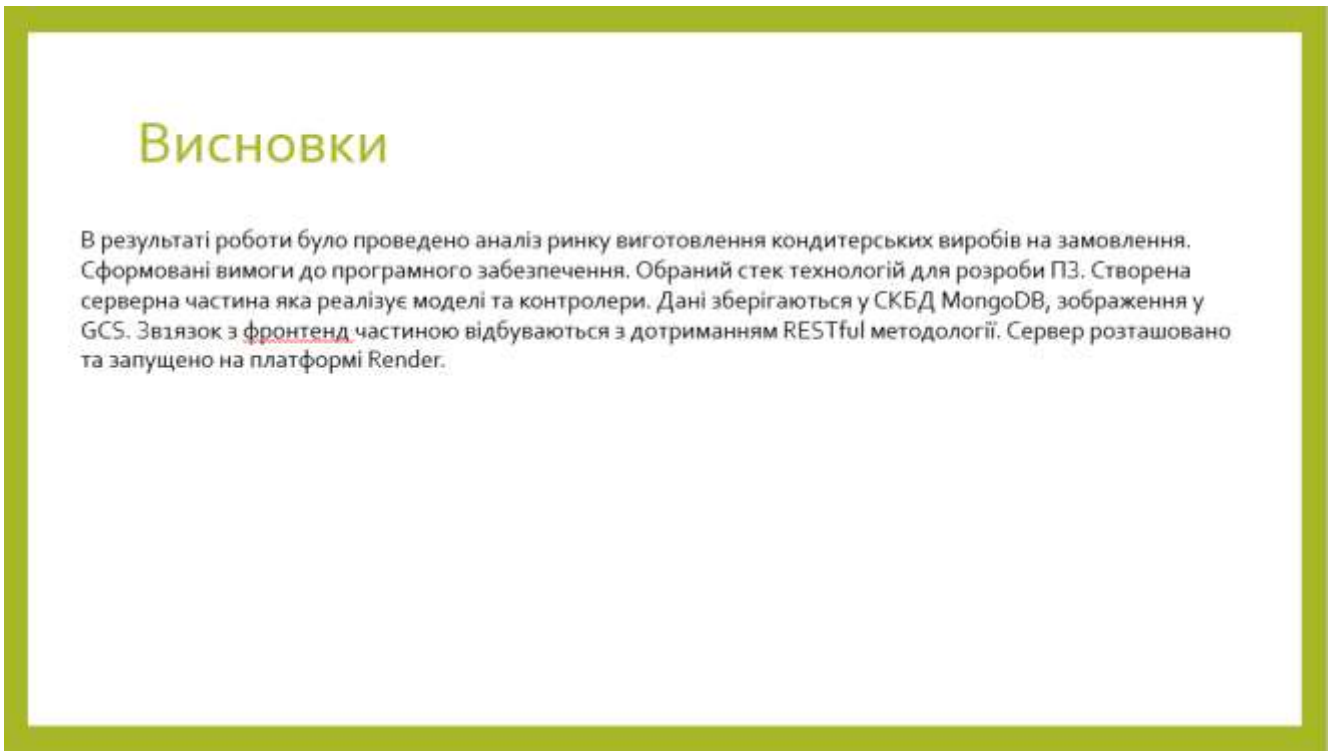


Рисунок А.15 – Слайд висновки (рисунок виконаний самостійно)


Рисунок А.16 – Фінальний слайд (рисунок виконаний самостійно)

ДОДАТОК В

Специфікація ПЗ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
Кафедра програмної інженерії

# СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмна система для продажу кондитерських виробів виготовлених на замовлення.

Студент гр. ПЗПІ-20-8 _____ Фарафонов Є.Ю.

Харків
2024

# ЗМІСТ

1 Вступ ..... 63
1.1 Огляд проекту ..... 63
1.2 Мета ..... 64
1.3 Межа ..... 64
1.4 Посилання ..... 65
1.5Визначення та абривіатури ..... 65
2 Загальний опис ..... 66
2.1 Перспективи продукту ..... 66
2.2 Функції продукту ..... 67
2.3 Характеристика користувачів ..... 67
2.4 Загальні обмеження ..... 68
2.5 Припущення і залежності ..... 69
3 Специфікація вимог ..... 71
3.1 Вимоги до зовнішніх інтерфейсів ..... 71
3.1.1 Інтерфейс користувача ..... 71
3.1.2 Програмні інтерфейси ..... 72
3.1.3 Комунікаційний протокол ..... 73
3.2 Атрибути програмного продукту ..... 73
3.2.1 Надійність ..... 74
3.2.2 Доступність ..... 75
3.2.3 Безпека ..... 75
3.2.4 Вимоги баз даних ..... 76

# 1 ВСТУП

## 1.1 Огляд проекту

ПС Cake Shop представляє собою комплексний веб-застосунок для замовлення тортів. Його основною метою є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, які бажають замовити торти для різних подій, таких як дні народження, весілля або інші свята. Система дозволяє користувачам вибирати торти з каталогу готових шаблонів, а також створювати індивідуальні замовлення з урахуванням особливих побажань щодо смаку, ваги та дизайну торта.

ПС включає в себе як фронтенд, так і бекенд компоненти. Фронтенд частина розроблена з використанням сучасних веб-технологій, таких як React, що забезпечує динамічність та високу продуктивність інтерфейсу. Бекенд частина реалізована на Node.js та Express, що дозволяє обробляти запити користувачів, управляти даними в базі даних та взаємодіяти з зовнішніми сервісами. Зберігання даних реалізоване за допомогою MongoDB, що забезпечує масштабованість та надійність.

Основні функціональні можливості системи включають реєстрацію та авторизацію користувачів, створення та управління замовленнями, перегляд каталогу тортів, додавання тортів до кошика та оформлення замовлення. Крім того, система підтримує завантаження зображень тортів та їх подальше зберігання у хмарному сховищі Google Cloud Storage. Це дозволяє користувачам бачити візуальні приклади тортів та робити більш обґрунтований вибір.

ПС Cake Shop також має адміністраторську панель, яка дозволяє керувати користувачами, замовленнями та каталогом тортів. Адміністратори можуть додавати нові шаблони тортів, редагувати існуючі та видаляти непотрібні. Вони також можуть обробляти замовлення, переглядати їх статус та змінювати його у разі необхідності.

Таким чином, ПС Cake Shop представляє собою багатофункціональну систему для зручного та ефективного управління процесом замовлення тортів, що відповідає потребам як кінцевих користувачів, так і адміністраторів системи.

## 1.2 Мета

Метою ПС Cake Shop є створення зручного та ефективного веб-застосунку для замовлення тортів, який дозволяє користувачам легко обирати та замовляти торти з каталогу або створювати індивідуальні замовлення з урахуванням особистих вподобань. ПС також передбачає забезпечення адміністративних функцій для управління каталогом тортів, обробки замовлень та взаємодії з користувачами, що сприяє оптимізації бізнес-процесів і покращенню користувацького досвіду.

## 1.3 Межа

ПС Cake Shop має чітко визначені межі доступу та функціональності для трьох основних категорій користувачів: незареєстрованих користувачів, зареєстрованих користувачів та адміністраторів. Незареєстровані користувачі можуть вільно переглядати каталог тортів, ознайомлюватися з описами та зображеннями, що надає їм можливість побачити доступний асортимент. Однак, ці користувачі не мають доступу до можливостей додавання тортів до кошика або оформлення замовлень, що мотивує їх зареєструватися на сайті для отримання повного функціоналу.

Зареєстровані користувачі, після успішної авторизації, отримують розширений доступ до функцій системи. Вони можуть створювати замовлення, вибираючи торти з каталогу, персоналізуючи їх за смаком, вагою та додаванням тексту на торт. Крім того, зареєстровані користувачі можуть переглядати історію своїх замовлень, редагувати свій профіль та отримувати персоналізовані рекомендації. Цей рівень доступу дозволяє їм повністю використовувати можливості платформи для замовлення тортів.

Адміністратори мають найвищий рівень доступу в системі. Вони можуть керувати користувачами, включаючи блокування або видалення облікових записів, якщо це необхідно. Адміністратори також мають можливість додавати нові торти до каталогу, редагувати деталі існуючих тортів та видаляти ті, що більше не актуальні. Крім цього, вони обробляють замовлення, слідкуючи за їх статусом від

створення до виконання. Це забезпечує гладку роботу системи та задоволення користувачів. Такий розподіл прав та обов’язків між різними категоріями користувачів забезпечує ефективне управління проектом та безпеку даних.

## 1.4 Посилання

Усі посилання є у переліку роботи.

## 1.5Визначення та абривіатури

БД – База Даних
ПС – Програмна Система
API – Application Programming Interface
GCS – Google Cloud Storage
JSON – JavaScript Object Notation
HTTP – Hypertext Transfer Protocol
REST – Representational State Transfer
SEO – Search Engine Optimization

# 2 ЗАГАЛЬНИЙ ОПИС

## 2.1 Перспективи продукту

Проект Cake Shop має значний потенціал для розвитку та розширення, враховуючи поточні тенденції в онлайн-торгівлі та попит на персоналізовані продукти. Однією з ключових перспектив є розширення асортименту тортів, включаючи нові рецепти та дизайни, що дозволить задовольнити смаки та вподобання більшої кількості клієнтів. Це може включати розробку спеціальних тортів для різних свят, дієтичних варіантів для людей з особливими потребами в харчуванні та ексклюзивних колекцій для особливих випадків.

Іншою важливою перспективою є інтеграція з соціальними мережами та платформами для обміну зображеннями, що дозволить користувачам легко ділитися своїми замовленнями та отримувати відгуки від друзів та родичів. Це не тільки збільшить залученість користувачів, але й посприяє природному маркетингу продукту через рекомендації.

Автоматизація та оптимізація бізнес-процесів також є важливим напрямком розвитку. Впровадження системи автоматичного відстеження статусу замовлень, управління запасами інгредієнтів та інтеграція з платіжними системами забезпечить більш ефективну роботу платформи та підвищить задоволеність клієнтів. Крім того, розвиток мобільного застосунку дозволить користувачам зручно робити замовлення зі своїх смартфонів, що відповідає сучасним тенденціям мобільності та швидкості.

Перспективним напрямком є також розширення географії доставки, що дозволить обслуговувати клієнтів у нових регіонах та містах. Це може бути досягнуто через партнерство з місцевими пекарнями та службами доставки, що забезпечить швидке та якісне виконання замовлень.

Нарешті, збір та аналіз даних про вподобання клієнтів дозволить впроваджувати індивідуальні рекомендації та пропозиції, що підвищить лояльність користувачів та їх задоволення від використання платформи. Використання технологій штучного інтелекту для аналізу замовлень та поведінки користувачів відкриває нові можливості для персоналізації та покращення сервісу.

## 2.2 Функції продукту

Програмна система передбачає існування декількох ролей: адміністратора, клієнта і незареєстрованого користувача. Незареєстрований користувач може створити акаунт, заповнивши форму реєстрації зі збереженням основних даних, таких як ім'я, електронна пошта, телефон і т.д.

Для зареєстрованого користувача (клієнта) передбачено клієнтську частину з функціоналом авторизації за допомогою логіна та пароля, забезпеченням безпеки даних, можливістю перегляду та зміни особистих даних, вибором продуктів з каталогу з можливістю фільтрації за категоріями та характеристиками, інтеграцією з платіжними системами для оплати замовлень, можливістю створення власних кондитерських виробів через конструктор, оформленням та відстеженням замовлень.

Адміністратор має можливість авторизуватися в системі за допомогою логіна та пароля, додавати нові позиції до каталогу кондитерських виробів з вказанням інформації про продукт, ціни та характеристики, обробляти замовлення, взаємодіяти з клієнтами щодо замовлень та управляти акаунтами користувачів, включаючи призначення статусу адміністратора.

У системі існують нефункціональні вимоги, включаючи захист особистих даних, відмовостійкість, зручний інтерфейс для користувачів та адміністраторів, оптимізація продуктивності та сумісність з різними пристроями та браузерами. Frontend реалізований на React з компонентною архітектурою, використанням Redux для управління станом, механізмами аутентифікації та обробкою помилок. Backend розроблений на Node.js з реалізацією REST API, забезпеченням безпеки даних, валідацією та підключенням до бази даних MongoDB.

## 2.3Характеристика користувачів

Програмна система буде цілеспрямована на різні сегменти аудиторії, включаючи приватних клієнтів, які шукають персоналізовані кондитерські вироби для особистих святкувань та подарунків. Цей сегмент включає індивідуальних

покупців усіх вікових груп, зокрема молодь, що цінує нестандартні десерти, і осіб середнього віку, які організовують святкові заходи для родини та друзів.

Корпоративні клієнти також є важливим сегментом, зокрема компанії, які замовляють великі партії кондитерських виробів для корпоративних заходів та рекламних акцій. Цей сегмент включає бізнес-замовлення на свята, конференції та інші події, де важлива якість та ефектність продуктів.

До організаторів подій та кейтерингових компаній належать організатори свят та кейтерингові служби, які шукають унікальні кондитерські вироби для весільних церемоній та подій різного масштабу. Ці клієнти цінують індивідуальний підхід та можливість персоналізації продуктів.

Основні характеристики цільової аудиторії включають вікові групи від молоді до середнього віку, з високими доходами та здатністю оплачувати преміум-продукцію. Їх географічне розташування зазвичай великі міста, де є підвищений попит на унікальні та персоналізовані кондитерські вироби.Ця аудиторія активно використовує соціальні мережі для пошуку ідей та організації подій, вони цінують якість, унікальність та індивідуальний підхід у обслуговуванні.

## 2.4 Загальні обмеження

При розробці та впровадженні проекту Cake Shop слід враховувати кілька важливих обмежень, які можуть вплинути на функціональність та ефективність роботи системи. Ці обмеження стосуються як технічних аспектів, так і користувацького досвіду.

Технічні обмеження: Система має обмеження на обсяг даних, що зберігаються в базі даних, а також на розмір файлів, що завантажуються користувачами (наприклад, фотографій тортів). Обмеження на обсяг оперативної пам'яті та процесорних ресурсів на сервері може вплинути на швидкість обробки запитів та загальну продуктивність.

Обмеження доступу: Незареєстровані користувачі мають обмежений доступ до функціональності платформи. Вони можуть переглядати каталог тортів, але не мають можливості робити замовлення або залишати відгуки. Зареєстровані

користувачі мають доступ до особистого кабінету, історії замовлень та можливості персоналізації тортів, але деякі функції, такі як доступ до адміністративних інструментів, для них недоступні.

Адміністративні обмеження: Адміністратори мають повний доступ до управління користувачами, замовленнями та каталогом тортів. Однак, вони обмежені в можливостях змінювати критично важливі налаштування системи, такі як конфігурація сервера або бази даних, що забезпечує безпеку та стабільність платформи.

Юридичні обмеження: Система повинна відповідати нормам законодавства щодо захисту персональних даних (наприклад, GDPR) та умовам використання платіжних систем. Це включає необхідність отримання згоди користувачів на обробку їхніх даних та забезпечення конфіденційності фінансової інформації.

Обмеження на інтерфейс: Інтерфейс користувача має бути оптимізований для роботи на різних пристроях (настільних комп'ютерах, планшетах, смартфонах). Однак, існують обмеження щодо підтримки застарілих браузерів та операційних систем, що може вплинути на доступність платформи для деяких користувачів.

Мережева інфраструктура: Швидкість та надійність інтернет-з'єднання можуть впливати на користувацький досвід, особливо при завантаженні великих зображень або здійсненні онлайн-платежів. Обмеження на швидкість мережі можуть призводити до затримок у обробці запитів та оновлення даних в реальному часі.

Розуміння та врахування цих обмежень є важливими для забезпечення стабільної та ефективної роботи платформи Cake Shop, а також для задоволення потреб користувачів та дотримання юридичних вимог.

## 2.5 Припущення і залежності

У розвитку і впровадженні проекту Cake Shop існують кілька ключових припущень і залежностей, які варто враховувати для успішного виконання завдань і досягнення поставлених цілей.

а) Технічні припущення:

– Хостинг і серверна інфраструктура: Платформа буде розгорнута на хостинг-провайдері Render з використанням git-репозиторію для автоматизації процесу деплою. Припущення базується на надійності і швидкодії інфраструктури Render,
– Технології: Використання Node.js для серверної частини дозволить ефективно керувати запитами та оптимізувати швидкодію системи. Фронтенд буде реалізований з використанням React.js для покращення користувацького досвіду,

б) Функціональні припущення:

– Реєстрація і авторизація користувачів: Платформа передбачає можливість реєстрації, авторизації та керування особистими даними користувачів. Це припущення базується на необхідності забезпечення конфіденційності та безпеки персональних даних,
– Інтеграція платіжних систем: Для здійснення онлайн-платежів передбачається інтеграція з платіжними системами, такими як Stripe або PayPal. Це припущення базується на забезпеченні зручного та безпечного способу оплати для користувачів,

в) Організаційні залежності:

– Залучення команди розробників: Для успішної реалізації проекту необхідно мати кваліфіковану команду розробників з досвідом у розробці веб-додатків та знанням необхідних технологій,
– Управління проектом: Ефективне управління проектом та звітність перед замовником є ключовими аспектами для забезпечення вчасної поставки продукту,

г) Юридичні залежності:

– Дотримання законодавства: Платформа повинна відповідати вимогам щодо захисту персональних даних та правилам використання інформації, що регулюються GDPR та іншими законодавчими актами.

Ці припущення і залежності є важливими для правильного планування, реалізації та успішного впровадження проекту Cake Shop.

# 3 СПЕЦИФІКАЦІЯ ВИМОГ

## 3.1 Вимоги до зовнішніх інтерфейсів

Вимоги до зовнішніх інтерфейсів системи включають інтуїтивно зрозуміле та зручне користувацьке середовище як для клієнтів, так і для адміністраторів. Клієнтський інтерфейс має бути легким у використанні і ефективно взаємодіяти з користувачем, забезпечуючи швидкий доступ до всіх функцій системи. Наявність інтуїтивних елементів керування та навігації дозволить клієнтам легко знаходити потрібну інформацію і здійснювати покупки без зайвих труднощів.

Для адміністраторського інтерфейсу важливо мати зручний доступ до управління та моніторингу замовлень, продуктів і користувачів. Інтерфейс повинен надавати інструменти для швидкої обробки замовлень, а також можливість додавати і редагувати інформацію про продукцію без зайвих складнощів.

Обидва інтерфейси повинні бути адаптивними та сумісними з різними типами пристроїв (комп'ютери, планшети, мобільні телефони), щоб забезпечити зручний доступ до системи незалежно від того, де знаходиться користувач. Також важливо, щоб інтерфейси були естетично приємними і відповідали сучасним тенденціям дизайну, що позитивно впливає на користувацький досвід і сприяє залученню клієнтів до використання системи.

### 3.1.1 Інтерфейс користувача

Навігаційне меню: На головній сторінці повинно бути присутнє зручне навігаційне меню, яке включає категорії продуктів: торти, еклери, випічка, татрти, тощо. Також необхідні посилання на сторінки про нас, контакти, кошик, Вхід/Реєстрація. Важливо мати динамічні зображення зі знижками та спеціальними пропозиціями, щоб привертати увагу користувачів до акційних товарів. На головній сторінці повинні бути представлені найпопулярніші товари, щоб користувачі могли швидко ознайомитися з хітами продажів.

Кожен товар має бути представлений зображенням, назвою, коротким , ціною та кнопкою «Переглянути деталі».

На сторінці кожного продукту повинна бути детальна інформація, включаючи назву продукту, детальний опис, список інгредієнтів, вагу та ціну. Поле для вибору кількості товару повинно бути доступним для зручного додавання потрібної кількості до корзини.

У кошику має бути список продуктів із зображенням, назвою, кількістю, ціною та можливістю редагування кількості або видалення товару. Відображення вартості товарів, податків та загальної суми до оплати допоможе користувачам контролювати свої витрати. Кнопка "Оформити замовлення" повинна забезпечувати можливість переходу до оформлення замовлення. Опції кур'єрської доставки та самовивозу повинні бути доступними для вибору.

Користувачі повинні мати можливість заповнити особисті дані, включаючи ім'я, прізвище, контактний номер та електронну пошту. Також повинна бути можливість вказати адресу доставки. перевірка всіх даних та підтвердження замовлення дозволить користувачам бути впевненими у правильності введеної інформації перед оформленням покупки.

### 3.1.2 Програмні інтерфейси

Програмні інтерфейси системи мають бути добре документовані та доступні для інтеграції з іншими системами або сервісами. Для клієнтської частини системи (Frontend) передбачається API, яке забезпечує взаємодію з сервером через HTTP-запити. Це включає отримання інформації про продукти, обробку замовлень, аутентифікацію користувачів та управління їх профілями.

Backend частина системи (Node.js) також надає API для взаємодії з базою даних та обробки бізнес-логіки. API повинно бути стабільним, ефективним та безпечним. Зокрема, важливо застосовувати стандарти безпеки для захисту від SQL-ін'єкцій, XSS-атак та інших потенційних загроз безпеці.

Документація API повинна містити чіткі описи кожного ендпоінту, формати даних, необхідні параметри запитів та очікувані формати відповідей. Це дозволяє іншим розробникам легко інтегрувати вашу систему та ефективно використовувати її можливості в їх додатках чи сервісах.

### 3.1.3 Комунікаційний протокол

Комунікаційний протокол системи базується на стандартах веб-комунальних технологій, зокрема HTTP і HTTPS для забезпечення безпеки та шифрування даних. Всі взаємодії між клієнтською та серверною частинами програмної системи здійснюються за допомогою RESTful API.

HTTP (Hypertext Transfer Protocol) використовується для передачі запитів і відповідей між клієнтом (веб-браузером чи іншим клієнтським додатком) та сервером. Всі запити, що ініціюються користувачами або іншими системами, адресуються через HTTP-методи GET, POST, PUT, DELETE та інші, залежно від потреби.

HTTPS (HTTP Secure) використовує TLS (Transport Layer Security) або його попередника SSL (Secure Sockets Layer) для захищеної передачі даних між клієнтом і сервером. Це забезпечує шифрування даних під час їх транспортування, що робить важливим захист особистої інформації користувачів.

REST (Representational State Transfer) є архітектурним стилем для взаємодії між компонентами програмного забезпечення. Він використовує стандартні HTTP-методи і URL для доступу до ресурсів (наприклад, замовлень, продуктів) та використовує JSON як формат даних для обміну інформацією між клієнтом і сервером.

Цей комунікаційний протокол забезпечує ефективну та безпечну взаємодію між всіма частинами системи, зменшуючи затримки і забезпечуючи стабільність роботи відповідно до вимог сучасних веб-додатків.

## 3.2 Атрибути програмного продукту

Атрибути програмного продукту включають різноманітні характеристики, які визначають його функціональність, якість і здатність задовольняти потреби користувачів. Основними атрибутами програмного продукту є:

Функціональність: Програмний продукт повинен забезпечувати всі функції, необхідні для задоволення вимог користувачів. Це включає функції авторизації,

управління профілем, вибір товарів, оформлення замовлень, оплату і відстеження статусу замовлень.

Ефективність: Продукт повинен працювати ефективно навіть при великому обсязі даних та великій кількості користувачів одночасно. Він повинен мати швидкий час відгуку і обробки запитів.

Надійність: Програмний продукт повинен бути стійким до відмов і збоїв. Він повинен відновлювати свою працездатність після помилок та забезпечувати високу доступність для користувачів.

Легкість використання: Інтерфейс користувача повинен бути зручним і інтуїтивно зрозумілим для всіх категорій користувачів: незареєстрованих користувачів, клієнтів і адміністраторів.

Безпека: Програмний продукт повинен захищати конфіденційні дані користувачів та інформацію про замовлення. Він повинен використовувати сучасні методи шифрування і захисту даних.

Сумісність: Продукт повинен бути сумісним з різними пристроями (комп'ютери, планшети, смартфони) і браузерами (Chrome, Firefox, Safari, Edge тощо).

Складність: Рівень складності програмного продукту повинен відповідати технічним знанням і вмінням користувачів. Він повинен бути достатньо простим для новачків і достатньо гнучким для досвідчених користувачів.

Ці атрибути допомагають визначити якість та придатність програмного продукту для використання в реальних умовах та задоволення потреб його користувачів.

### 3.2.1 Надійність

Надійність системи є однією з важливих характеристик, яка визначає здатність програмного забезпечення працювати без збоїв і відмов протягом тривалого часу. В нашому проекті ця надійність забезпечується завдяки застосуванню передових технологій, ретельному архітектурному дизайну і систематичному тестуванню. Ми приділяємо особливу увагу тестуванню перед

кожним випуском нових функцій і оновлень, що дозволяє виявляти та усувати помилки на ранніх етапах розробки і після випуску продукту.

Автоматизоване тестування відіграє ключову роль у нашому процесі, прискорюючи перевірку функціональності і покращуючи загальну якість програмного забезпечення. Окрім цього, постійний моніторинг системи допомагає вчасно виявляти і усувати проблеми, що забезпечує безперебійну роботу нашої системи в умовах реального використання.

### 3.2.2 Доступність

Всі зображення, включаючи банери, продукти та інші графічні елементи, повинні мати альтернативний текст, що описує їх вміст. Всі тексти повинні мати достатній контраст із фоном для забезпечення читабельності.

Всі елементи списку повинні мати чітко визначені межі для екранних читачів та бути доступними для навігації з клавіатури. Кожен товар у списку повинен мати достатній опис, який буде доступний екранним читачам. Кнопки "Додати до корзини" повинні бути легко доступними з клавіатури.

Вся текстова інформація про продукт повинна бути доступною для екранних читачів. Зображення продукту повинні мати альтернативний текст. Всі поля форми повинні мати мітки для екранних читачів та бути доступними для навігації з клавіатури. Інструкції для заповнення повинні бути чіткими та зрозумілими. Поля для введення платіжної інформації повинні мати мітки для екранних читачів та бути доступними з клавіатури.

### 3.2.3 Безпека

Безпека є однією з найважливіших складових нашого проекту. Ми приділяємо особливу увагу захисту особистих даних користувачів, забезпечуючи їх конфіденційність і цілісність. Наш підхід до забезпечення безпеки включає декілька ключових аспектів.

По-перше, ми застосовуємо сучасні стандарти безпеки для захисту даних, включаючи шифрування і захист від несанкціонованого доступу. Всі особисті дані

передаються через захищені канали з використанням протоколів шифрування, таких як SSL/TLS, що гарантує конфіденційність під час трансмісії.

По-друге, ми ретельно валідуємо і фільтруємо всі введені дані на рівні сервера для запобігання SQL-ін'єкціям, XSS атакам та іншим видам атак на додаток. Це дозволяє уникнути введення шкідливих даних та зберегти інформацію в цілісному стані.

### 3.2.4 Вимоги баз даних

Вимоги до баз даних в нашому проекті включають кілька ключових аспектів, які гарантують ефективність, надійність і безпеку зберігання і обробки даних.

По-перше, ми використовуємо базу даних MongoDB для зберігання і управління даними нашої програмної системи. MongoDB вибрана через її гнучкість, швидкість і масштабованість, що дозволяє ефективно виконувати завдання зберігання та доступу до даних в умовах великого обсягу інформації.

По-друге, структура бази даних ретельно продумана для відповідності потребам нашого додатку. Ми використовуємо нормалізовану структуру даних там, де це необхідно для забезпечення цілісності і ефективного управління даними. Деякі таблиці і сутності можуть бути денормалізовані для забезпечення оптимальної продуктивності під час операцій зчитування.

По-третє, забезпечення безпеки даних є невід'ємною частиною нашого підходу до роботи з базами даних. Ми використовуємо механізми шифрування для захисту конфіденційної інформації під час зберігання та передачі даних, а також застосовуємо механізми контролю доступу для обмеження прав доступу до бази даних.

Додатково, для забезпечення відмовостійкості системи, ми реалізуємо резервне копіювання і відновлення даних, що дозволяє відновлювати інформацію в разі випадкового видалення або системних збоїв.

Всі ці вимоги до баз даних спрямовані на створення надійного, безпечного і ефективного середовища для зберігання і обробки даних користувачів нашого проекту.