Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота бакалавра

з теми:

«СИСТЕМА ЗАМОВЛЕННЯ ТА ДОСТАВКИ ЇЖІ ЧЕРЕЗ МОБІЛЬНИЙ ДОДАТОК НА БАЗІ ТЕХНОЛОГІЙ СТЕКУ MEAN»

Виконав: здобувач вищої освіти групи
KNms1-B21
спеціальності 122 Комп’ютерні науки
Денис ХОРОЛЬСЬКИЙ

Керівник:

Віталій ІВАНЮК, доктор технічних наук, завідувач, доцент кафедри комп’ютерних наук

Рецензент:

Богдана ОПРЯ, кандидат історичних наук, доцент, доцент кафедри туризму та готельно-ресторанної справи

Кам’янець-Подільський – 2024 р.

АНОТАЦІЯ

Хорольський Д.З. Система замовлення та доставки їжі через мобільний додаток на базі технологій стеку MEAN. Кваліфікаційна робота бакалавра на здобуття освітньо-кваліфікаційного рівня вищої освіти спеціалізації 122 «Комп'ютерні науки». – Національний університет імені Івана Огієнка, Кам'янець-Подільський, 2024.

У процесі роботи було проведено огляд та аналіз предметної області, включаючи дослідження сучасних методів і технологій для розробки мобільних додатків з функцією замовлення та доставки їжі. Вивчено основні потреби користувачів, аналіз конкурентних рішень, а також визначено ключові функціональні та нефункціональні вимоги до системи. Це дозволило сформулювати завдання проекту та обрати відповідний технологічний стек для його реалізації.

Було реалізовано систему замовлення та доставки їжі, що включає мобільний додаток і серверну частину на базі технологій стеку MEAN (MongoDB, Express.js, Angular, Node.js). Система дозволяє користувачам входити в додаток, переглядати меню ресторанів, додавати страви до кошика, оформлювати замовлення та відстежувати його статус у реальному часі. Весь процес, від вибору страв до підтвердження оплати і доставки, реалізовано з акцентом на зручність, продуктивність та безпеку даних.

Проведено економічне обґрунтування проєкту, яке включало аналіз витрат на розробку, впровадження та підтримку системи. Проаналізовано витрати на ліцензії, програмне забезпечення, маркетинг і просування. Оскільки проєкт є дипломною роботою, витрати на розробку були мінімальні, бо значну частину роботи виконав автор. Аналіз показав, що при ефективному управлінні проєктом і належному просуванні, додаток може забезпечити стабільний дохід у майбутньому.

Ключові слова: *мобільний додаток, стек MEAN, MongoDB, Express.js, Angular, Node.js, доставка їжі, система замовлення.*

ABSTRACT

Khorolskyi D.Z. Food Ordering and Delivery System via a Mobile Application Based on MEAN Stack Technologies. Bachelor's Qualification Work for Obtaining the Educational and Qualification Level of Higher Education in Specialty 122 "Computer Science". – Ivan Ohienko National University, Kamianets-Podilskyi, 2024.

During the work, a review and analysis of the subject area were conducted, including the study of modern methods and technologies for developing mobile applications with food ordering and delivery functionality. The main user needs, analysis of competitive solutions on the market, and key functional and non-functional requirements for the system were studied. This allowed us to clearly formulate the project tasks and choose the most suitable technology stack for its implementation.

A food ordering and delivery system was implemented, including a mobile application and a server part based on MEAN stack technologies (MongoDB, Express.js, Angular, Node.js). The system allows users to log into the application, view restaurant menus, add dishes to the cart, place orders, and track their status in real time. The entire process, from selecting dishes to confirming payment and delivery, is implemented with an emphasis on user convenience, high performance, and data security.

An economic justification of the project was conducted, which included an analysis of costs for development, implementation, and support of the system. Costs for licenses, software, marketing, and promotion were analyzed. Since the project is a thesis, development costs were minimal, as the author performed a significant part of the work. The analysis showed that with effective project management and proper promotion, the application can ensure a stable income in the future.

Keywords: *mobile application, MEAN stack, MongoDB, Express.js, Angular, Node.js, food delivery, ordering system.*

ЗМІСТ

ВСТУП

У сучасному світі швидкий темп життя вимагає ефективних рішень для вирішення повсякденних завдань, таких як замовлення та доставка їжі. Розвиток мобільних технологій значно спростив цей процес, дозволяючи користувачам замовляти їжу з будь-якого місця і в будь-який час. Однак, наявні рішення часто не відповідають всім потребам користувачів, таким як зручність використання, швидкість обслуговування та широкий асортимент страв. Впровадження нових технологій та підходів у розробку систем замовлення їжі може суттєво покращити цей процес.

Використання стеку технологій MEAN (MongoDB, Express.js, Angular, Node.js) дозволяє створити високоефективні, масштабовані та зручні у використанні додатки. MEAN-стек надає всі необхідні інструменти для розробки повного циклу додатку - від серверної частини до інтерфейсу користувача. Це робить його ідеальним вибором для створення системи замовлення та доставки їжі.

Метою даної роботи є розробка системи замовлення та доставки їжі через мобільний додаток на базі технологій стеку MEAN. Для досягнення поставленої мети необхідно виконати такі завдання: провести аналіз існуючих рішень для замовлення та доставки їжі; вивчити можливості та переваги стеку технологій MEAN; розробити вимоги до системи, включаючи функціональні та нефункціональні аспекти; спроектувати архітектуру системи та базу даних; розробити серверну та клієнтську частини додатку; провести тестування системи та оцінити її продуктивність; підготувати систему до впровадження та оцінити її ефективність у реальних умовах.

Об'єктом дослідження є процес замовлення та доставки їжі за допомогою мобільних додатків. Предметом дослідження є розробка та впровадження системи замовлення та доставки їжі на основі технологій стеку MEAN.

Для досягнення мети дослідження були використані наступні методи: аналіз літератури та існуючих рішень, проектування та моделювання, програмування,

тестування та експериментальний метод. Аналіз літератури та існуючих рішень включав дослідження наукових праць, статей та технічної документації для вивчення поточного стану технологій та рішень у сфері замовлення та доставки їжі.

Проектування та моделювання включали розробку архітектури системи, моделювання бази даних та проектування інтерфейсів користувача. Програмування охоплювало реалізацію серверної та клієнтської частини системи за допомогою технологій MEAN. Тестування включало проведення модульного, інтеграційного та системного тестування для перевірки працездатності та продуктивності системи. Експериментальний метод передбачав оцінку ефективності розробленої системи в реальних умовах експлуатації.

Практичне значення роботи полягає в можливості впровадження розробленої системи в реальні умови, що дозволить значно підвищити ефективність процесу замовлення та доставки їжі, забезпечивши користувачам зручний інструмент для швидкого та комфортного отримання необхідних послуг.

1. ОГЛЯД ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Ринок додатків для замовлення їжі

У сучасному світі технології відіграють ключову роль у поліпшенні комфорту життя людей. Однією з таких сфер є замовлення та доставка їжі. Сьогодні на ринку присутня велика кількість різних мобільних додатків та онлайн-сервісів, які надають послуги замовлення та доставки їжі. Розглянемо детально існуючі рішення, їх особливості, функціональні можливості та обмеження.

Ринок додатків для замовлення їжі насичений різноманітними продуктами, які пропонують широкий спектр послуг. Найбільш популярними серед них є Uber Eats, Grubhub, DoorDash та Glovo. Uber Eats, один з найбільших гравців на ринку, відомий своєю широкою мережею ресторанів та швидкою доставкою. Серед його переваг можна виділити зручний інтерфейс, можливість відстеження замовлення в реальному часі та великий вибір ресторанів. Проте, до його недоліків відносяться високі комісійні для ресторанів та інколи високі ціни на доставку.

Grubhub пропонує широку базу користувачів та багато функціональних можливостей, таких як детальний пошук, перегляд меню, відгуки користувачів та різноманітні опції оплати. Однак, інтерфейс іноді критикують за складність, а час доставки може бути довшим у порівнянні з конкурентами. DoorDash, у свою чергу, пропонує послуги доставки з широкого спектру ресторанів та магазинів. Відмінною рисою цього сервісу є можливість попереднього замовлення їжі. Недоліком є інколи довгий час доставки та висока комісія.

Glovo спеціалізується не тільки на доставці їжі, але й на доставці інших товарів. Цей сервіс відомий своєю універсальністю та швидкістю. Проте, вартість доставки може бути вищою у порівнянні з іншими сервісами, а доступність може бути обмеженою в менших містах.

1.2 Функціональні можливості та обмеження

Більшість додатків для замовлення їжі мають схожий набір функціональних можливостей, які забезпечують зручність використання для користувачів. Основні з них включають пошук ресторанів, перегляд меню, розміщення замовлень, оплата онлайн, відстеження замовлень у реальному часі, збереження улюблених ресторанів та страв, а також відгуки користувачів.

Пошук ресторанів дозволяє користувачам знаходити ресторани за різними критеріями, такими як розташування, тип кухні, рейтинг тощо. Перегляд меню надає можливість користувачам ознайомитися з асортиментом страв, їх описами, фотографіями та цінами. Розміщення замовлень дає змогу швидко і зручно замовити їжу через додаток, а оплата онлайн підтримує різні методи оплати, включаючи кредитні картки та електронні гаманці. Відстеження замовлень у реальному часі дозволяє користувачам відстежувати статус замовлення та місцезнаходження кур'єра, що підвищує прозорість та зручність сервісу. Збереження улюблених ресторанів та страв забезпечує швидкий доступ до них у майбутньому, а відгуки користувачів допомагають зробити правильний вибір, спираючись на досвід інших клієнтів.

Попри численні переваги, існуючі додатки для замовлення їжі мають певні обмеження. Наприклад, у пікові години (наприклад, обідній час чи вечори у вихідні) час доставки може бути значно довшим через високе навантаження на систему. Деякі додатки стягують значні комісії з ресторанів, що може впливати на ціни для кінцевих користувачів. У деяких регіонах доступність ресторанів може бути обмеженою, що зменшує вибір для користувачів. Багато додатків не забезпечують достатньо індивідуалізованих рекомендацій, що знижує користувацький досвід. Деякі додатки можуть мати проблеми з інтеграцією платіжних систем або інтерфейсу, що ускладнює процес замовлення.

1.3 Порівняння з новими рішеннями

Успіх існуючих додатків для замовлення їжі створив сприятливе середовище для розвитку нових, інноваційних рішень. Нові додатки часто орієнтовані на вирішення поточних проблем та надання додаткових можливостей.

Одним із таких нововведень є алгоритми рекомендацій, які використовують машинне навчання для створення індивідуальних рекомендацій на основі попередніх замовлень та вподобань користувача. Це дозволяє значно покращити користувацький досвід, пропонуючи страви, які найбільше відповідають смаковим уподобанням клієнта. Іншим важливим аспектом є зниження комісійних зборів. Деякі нові сервіси прагнуть зменшити комісійні збори для ресторанів, що робить їх більш привабливими для малого бізнесу і може позитивно впливати на ціни для кінцевих користувачів.

Нові додатки також фокусуються на підтримці та співпраці з локальними ресторанами і кафе, що сприяє розвитку місцевого бізнесу та забезпечує користувачам унікальні кулінарні пропозиції. Крім того, деякі нові додатки пропонують не лише доставку їжі, але й інших товарів, що розширює можливості сервісу та підвищує його цінність для користувачів. Використання новітніх технологій для забезпечення безперебійної роботи платіжних систем, інтеграції з іншими сервісами та оптимізації процесу замовлення є ще одним важливим аспектом [1].

Аналізуючи існуючі рішення та порівнюючи їх з новими розробками, можна зробити висновок, що ринок додатків для замовлення їжі постійно еволюціонує, пропонуючи користувачам дедалі більше можливостей. Водночас, нові технології та підходи дозволяють вдосконалювати ці системи, роблячи їх більш зручними, швидкими та ефективними.

1.4 Технологічні основи систем замовлення їжі

Сучасні системи замовлення їжі базуються на складних технологічних рішеннях, які забезпечують безперебійну роботу всіх етапів процесу — від вибору страви до її доставки до кінцевого споживача. Ці технології включають багаторівневу архітектуру, інтеграцію з платіжними системами та геолокаційними сервісами, що забезпечує високу продуктивність, зручність і безпеку користувацького досвіду.

Архітектура сучасних систем замовлення їжі зазвичай має багаторівневу структуру, яка включає клієнтську частину (мобільний додаток або веб-сайт), серверну частину (сервери, які обробляють запити користувачів) та базу даних (сховище даних про користувачів, ресторани, меню та замовлення). Клієнтська частина відповідає за взаємодію з користувачем, надаючи інтерфейс, через який користувачі можуть переглядати меню, робити замовлення, здійснювати оплату та відстежувати статус доставки. Інтерфейс зазвичай розробляється з використанням фреймворків для веб-розробки (наприклад, Angular, React) або мобільної розробки (наприклад, React Native, Flutter).

Серверна частина обробляє запити від клієнтської частини, керує бізнес-логікою додатку і взаємодіє з базою даних. Для реалізації серверної частини часто використовують такі технології, як Node.js з Express.js, що забезпечує асинхронну обробку запитів і високу продуктивність. База даних зберігає всі необхідні дані для функціонування системи: інформацію про користувачів, ресторани, меню, замовлення та інше. Для цього можуть використовуватися реляційні бази даних (наприклад, MySQL, PostgreSQL) або нереляційні бази даних (наприклад, MongoDB), в залежності від вимог до гнучкості та масштабованості системи [9].

Взаємодія між цими компонентами забезпечується через API (Application Programming Interface), що дозволяє клієнтській частині взаємодіяти з сервером і базою даних. API реалізується з використанням RESTful або GraphQL підходів, що дозволяє стандартизувати і спростити обмін даними між компонентами системи [10].

Одним із ключових аспектів систем замовлення їжі є інтеграція з платіжними системами, що забезпечує можливість швидкої та безпечної оплати замовлень. Сучасні рішення використовують різні платіжні шлюзи, такі як Stripe, PayPal, Square, які підтримують широкий спектр платіжних методів, включаючи кредитні та дебетові картки, електронні гаманці та інші. Інтеграція з платіжними системами передбачає забезпечення високого рівня безпеки, оскільки обробка фінансових даних потребує захисту від несанкціонованого доступу та шахрайства. Для цього використовуються протоколи шифрування даних (наприклад, TLS), а також дотримання стандартів безпеки платіжних карток (PCI DSS).

Відстеження замовлень у реальному часі є важливою функцією, яка забезпечує прозорість процесу доставки для користувачів. Це досягається за допомогою інтеграції з геолокаційними сервісами, такими як Google Maps API, які дозволяють відстежувати місцезнаходження кур’єрів і надавати інформацію про статус замовлення в режимі реального часу. Геолокаційні сервіси надають можливість кур'єрам отримувати оптимальні маршрути для доставки, що зменшує час доставки та покращує ефективність роботи. Користувачі, в свою чергу, можуть бачити точне місцезнаходження свого замовлення на карті, що підвищує їхню довіру до сервісу і задоволеність від користування ним. Крім того, інтеграція з геолокаційними сервісами дозволяє додаткам надавати більш персоналізовані послуги, такі як рекомендації ресторанів на основі місцезнаходження користувача, що робить процес замовлення більш зручним і ефективним.

Для забезпечення високої продуктивності та масштабованості систем замовлення їжі використовуються різні технології та підходи. Використання асинхронних та неблокуючих моделей обробки запитів (наприклад, з використанням Node.js) дозволяє обробляти великий обсяг запитів одночасно, що підвищує продуктивність системи. Масштабованість досягається за рахунок горизонтального масштабування, коли додаткові сервери додаються до інфраструктури для обробки зростаючого навантаження. Використання контейнеризації (наприклад, Docker) та оркестрації (наприклад, Kubernetes)

дозволяє легко розгортати та керувати масштабованими додатками. Для зберігання та обробки великих обсягів даних використовуються розподілені бази даних та системи кешування (наприклад, Redis), що забезпечують швидкий доступ до даних та зменшують навантаження на основну базу даних.

Забезпечення надійності та безперебійної роботи систем замовлення їжі є критично важливим аспектом. Використання реплікації даних та резервного копіювання дозволяє забезпечити високу доступність та відновлення даних у разі збою. Моніторинг та логування дозволяють вчасно виявляти та усувати проблеми, що виникають у системі. Крім того, для забезпечення безпеки системи використовуються різні методи аутентифікації та авторизації користувачів, включаючи багатофакторну аутентифікацію (MFA) та рольову модель доступу (RBAC), що дозволяє контролювати доступ до чутливих даних та функцій системи.

Таким чином, технологічні основи систем замовлення їжі включають багаторівневу архітектуру, інтеграцію з платіжними та геолокаційними сервісами, використання передових технологій для забезпечення продуктивності, масштабованості, надійності та безпеки. Це забезпечує ефективну, зручну та безпечну роботу систем замовлення їжі, що відповідає потребам сучасних користувачів.

1.5 Стек технологій MEAN: огляд і переваги

Стек технологій MEAN (MongoDB, Express.js, Angular, Node.js) є популярним вибором для розробки сучасних веб-додатків завдяки своїй ефективності, гнучкості та можливості створення повноцінних додатків на одній мові програмування – JavaScript. Кожен компонент стека відіграє важливу роль у побудові функціонального, масштабованого та високопродуктивного додатку.

MongoDB є нереляційною базою даних, яка зберігає дані у форматі документів JSON. Ця база даних надає можливість зберігати великі обсяги даних без фіксованої схеми, що забезпечує високу гнучкість і масштабованість. MongoDB

добре підходить для додатків, які мають справу з великою кількістю даних та потребують високої продуктивності при виконанні операцій зчитування та запису. Завдяки своїй документно-орієнтованій моделі, MongoDB дозволяє легко інтегруватися з іншими компонентами стека MEAN, що робить її зручним вибором для розробників.

Express.js є фреймворком для веб-розробки на основі Node.js, який спрощує процес створення серверної частини додатків. Express.js забезпечує мінімалістичний, але потужний набір інструментів для створення веб-серверів та обробки запитів. Завдяки своїй простоті та гнучкості, Express.js дозволяє швидко розробляти та розгортати серверні додатки, а також інтегрувати їх з базою даних та клієнтською частиною. Він підтримує різні методи HTTP і середовища, що дозволяє легко налаштовувати маршрутизацію та обробку запитів.

Angular є фреймворком для розробки клієнтської частини додатків, який надає багатий набір інструментів для створення інтерактивних веб-інтерфейсів. Angular підтримує двосторонню прив'язку даних, що дозволяє автоматично синхронізувати модель та представлення. Це забезпечує високу продуктивність і зручність розробки складних інтерфейсів користувача. Крім того, Angular надає вбудовані механізми для керування станом додатку, обробки подій та створення багаторазових компонентів, що дозволяє розробникам легко масштабувати та підтримувати код.

Node.js є середовищем виконання для JavaScript, яке дозволяє запускати серверний код на основі двигуна V8 від Google Chrome. Node.js забезпечує асинхронну, неблокуючу модель обробки запитів, що дозволяє досягти високої продуктивності та масштабованості. Завдяки використанню одного модуля для обробки вхідних і вихідних запитів, Node.js дозволяє обробляти велику кількість одночасних з'єднань, що робить його ідеальним для реальних додатків та систем з високим навантаженням. Node.js також має великий екосистемний набір модулів та пакетів, доступних через npm, що дозволяє легко додавати нові функціональні можливості до додатку.

Використання стека MEAN має кілька ключових переваг. По-перше, весь стек використовує одну мову програмування — JavaScript, що значно спрощує розробку і дозволяє розробникам працювати над усіма частинами додатку без необхідності перемикатися між різними мовами. Це також сприяє зниженню витрат на навчання та підтримку коду, оскільки одна команда розробників може займатися як клієнтською, так і серверною частиною додатку.

По-друге, стек MEAN забезпечує високу продуктивність і масштабованість додатків завдяки використанню асинхронної обробки запитів у Node.js і гнучкості MongoDB. Це дозволяє легко обробляти великі обсяги даних та підтримувати високу швидкість роботи додатку навіть при значних навантаженнях.

По-третє, використання Angular для розробки клієнтської частини додатків дозволяє створювати сучасні, інтуїтивно зрозумілі та інтерактивні інтерфейси користувача. Завдяки багатому набору вбудованих інструментів та можливостей Angular, розробники можуть швидко створювати та підтримувати складні інтерфейси, що відповідають вимогам сучасних користувачів [2].

По-четверте, Express.js надає простий та потужний інструментарій для розробки серверної логіки додатків. Це дозволяє швидко створювати RESTful API, налаштовувати маршрутизацію та інтегрувати серверну частину з базою даних та клієнтською частиною.

В цілому, стек MEAN є потужним інструментом для розробки сучасних веб-додатків, що забезпечує високу продуктивність, масштабованість та зручність розробки. Завдяки своїй гнучкості та широкому набору інструментів, MEAN дозволяє створювати ефективні, інтерактивні та масштабовані додатки, що відповідають сучасним вимогам і очікуванням користувачів [14].

Висновки розділу «Огляд та аналіз предметної області»

У розділі було здійснено детальний розгляд ключових аспектів розробки системи замовлення та доставки їжі через мобільний додаток.

По-перше, було проаналізовано ринок додатків для замовлення їжі, що дозволило визначити основні тенденції та вимоги користувачів. Це дослідження показало, що зростаюча популярність мобільних додатків вимагає від розробників впровадження інноваційних функцій та забезпечення високого рівня зручності використання.

Далі були вивчені функціональні можливості та обмеження існуючих систем замовлення їжі. Було виявлено, що сучасні рішення часто мають обмеження щодо зручності користування, швидкості обслуговування та асортименту доступних страв. Це обґрунтувало необхідність розробки нової системи, яка б вирішувала ці проблеми.

Порівняння з новими рішеннями дозволило визначити переваги та недоліки різних підходів до розробки систем замовлення їжі. Було встановлено, що використання сучасних технологій може суттєво підвищити ефективність та зручність таких систем.

Технологічні основи систем замовлення їжі були розглянуті з метою визначення найкращих практик та підходів до розробки. Це включало аналіз архітектурних рішень, вибір баз даних, серверних та клієнтських технологій, що дозволило сформувати оптимальний план розробки системи.

Нарешті, було проведено огляд і аналіз стеку технологій MEAN (MongoDB, Express.js, Angular, Node.js). Встановлено, що MEAN-стек надає всі необхідні інструменти для створення високоефективних, масштабованих та зручних у використанні додатків. Переваги MEAN-стека включають єдину мову програмування на всіх рівнях системи, високу продуктивність, легкість інтеграції компонентів та активну підтримку спільноти розробників.

Проведений огляд та аналіз предметної області дозволив чітко визначити вимоги до системи замовлення та доставки їжі, обґрунтувати вибір технологій для її розробки та сформувати план реалізації проекту з урахуванням сучасних тенденцій і потреб користувачів. Вибір стеку технологій MEAN забезпечує ефективну реалізацію поставлених завдань та досягнення високого рівня продуктивності та зручності системи.

2. ПРОЄКТУВАННЯ ТА ВИМОГИ ДО СИСТЕМИ ЗАМОВЛЕННЯ ТА ДОСТАВКИ ЇЖІ

2.1 Вимоги до системи

Вимоги до системи можуть бути розділені на дві категорії: функціональні та нефункціональні.

Функціональні вимоги:

Реєстрація користувачів: Може включати в себе такі аспекти, як створення облікового запису, підтвердження електронної адреси або номеру телефону, заповнення особистої інформації тощо.

Пошук та замовлення їжі: Це може включати пошук за різними категоріями (наприклад, тип кухні, ціновий діапазон, наявність акцій тощо) і можливість додавання товарів у кошик та оформлення замовлення.

Оплата та обробка замовлення: Система повинна підтримувати безпечні методи оплати, такі як кредитні картки, мобільні платежі або онлайн-платіжні системи, і забезпечувати обробку замовлень.

Відстеження замовлення: Користувачам слід мати можливість відстежувати статус їхніх замовлень у реальному часі, включаючи підтвердження, підготовку, доставку та отримання.

Нефункціональні вимоги:

Безпека: Система повинна забезпечувати конфіденційність, цілісність та доступність даних, а також захист від несанкціонованого доступу та зловживань.

Швидкодія: Система повинна бути достатньою продуктивною, щоб обробляти великий потік замовлень та запитів користувачів у реальному часі.

Доступність: Система повинна бути доступною 24/7 для користувачів у будь-який час із різних пристроїв та мереж, забезпечуючи мінімальний час простою.

Інтерфейс: Інтерфейс користувача повинен бути інтуїтивно зрозумілим, зручним у використанні та естетично приємним, щоб забезпечити зручність користування та підвищити задоволення користувачів.

Вибираючи вимоги для вашої системи замовлення та доставки їжі через мобільний додаток на базі технологій стеку MEAN, ми відштовхувалися від конкретних потреб бізнесу та аудиторії користувачів, а також з ресурсних можливостей для реалізації та підтримки системи. Наприклад, ми зосередилися на важливих функціях, які відповідають цілям бізнесу та додають йому конкурентну перевагу, та на необхідних нефункціональних характеристиках, які забезпечують ефективне та безперебійне функціонування системи.

2.2 Архітектура системи

Архітектура системи замовлення та доставки їжі через мобільний додаток, розроблена на базі технологій стеку MEAN (MongoDB, Express.js, Angular, Node.js), відображає вибір оптимальних технологій та підходів для створення високопродуктивного, масштабованого та надійного рішення. Ця система має клієнт-серверну архітектуру, де мобільний додаток (клієнтська частина) взаємодіє з серверною частиною через RESTful API, що реалізовано за допомогою Express.js на Node.js.

Рис. 2.2.1 – Загальна схема архітектури

Клієнтська частина додатку реалізована з використанням фреймворку Angular, забезпечуючи зручний та ефективний інтерфейс користувача. Серверна частина системи, також розроблена на Node.js, використовує Express.js для створення веб-сервера та обробки запитів від клієнтів. Вона взаємодіє з базою даних MongoDB для зберігання та отримання інформації про користувачів, ресторани, страви та замовлення [3].

Важливою складовою архітектури є RESTful API, яке визначає правила та формати взаємодії між клієнтською та серверною частинами системи, забезпечуючи стандартизований та ефективний спосіб обміну даними. Даний підхід дозволяє системі бути гнучкою, легко масштабованою та реагувати на зміни в потребах користувачів та бізнес-логіки. Взаємодія між компонентами системи,

такими як клієнтська та серверна частини, а також база даних, забезпечує високу продуктивність та надійність роботи системи в цілому.

Архітектура системи детально розроблена з урахуванням потреб користувачів та вимог бізнесу, забезпечуючи оптимальну функціональність та зручний інтерфейс користувача. Вона покладається на сучасні технології та найкращі практики розробки програмного забезпечення для забезпечення ефективності та успішності додатку на ринку онлайн-замовлень їжі.

2.3 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача (UI) для мобільного додатку замовлення та доставки їжі є важливим етапом, який визначає зручність використання та загальний користувацький досвід. Основні завдання цього етапу включають створення інтуїтивно зрозумілого дизайну, забезпечення високої швидкості взаємодії та надання користувачам усієї необхідної інформації у зручному форматі.

Прототипи інтерфейсів:

Для розробки прототипів інтерфейсів для мобільного додатку системи замовлення та доставки їжі були використані спеціалізовані програми для створення макетів сторінок та їх взаємодії, що дозволило візуалізувати функціональність та дизайн додатку перед його реалізацією.

Прототипи інтерфейсів включають в себе всі основні елементи додатку, такі як головне меню, сторінки пошуку ресторанів і страв, сторінки корзини та оформлення замовлення, а також сторінки профілю користувача. Кожен елемент був уважно розроблений з урахуванням зручності використання та естетичного вигляду [12].

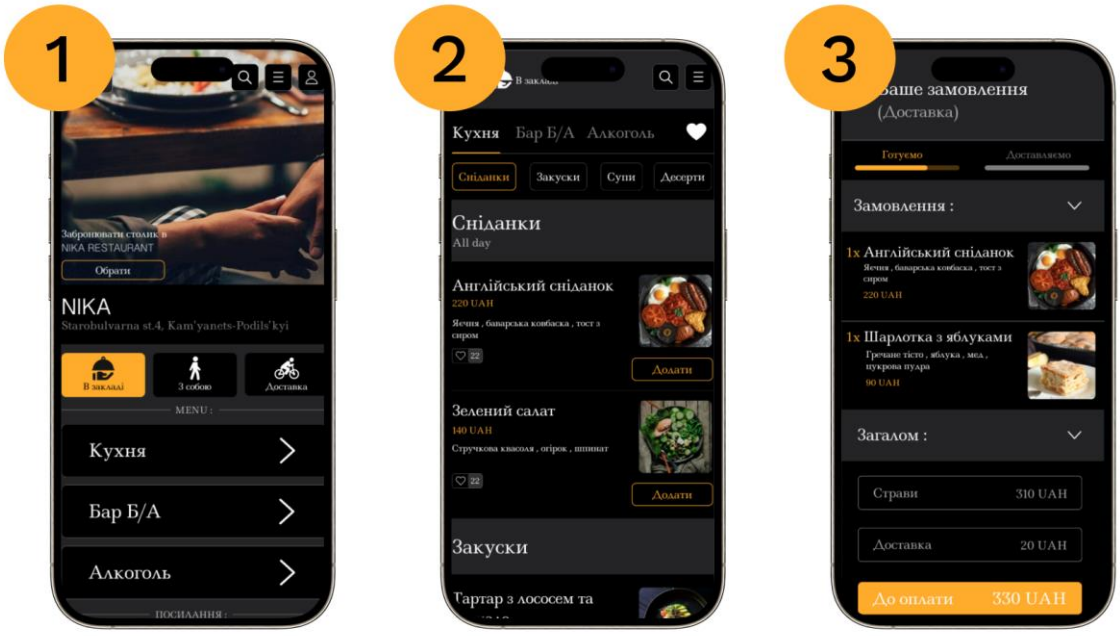

Рис. 2.3.1 – Прототип інтерфейсу

На основі прототипів створюється детальний дизайн інтерфейсу. Важливо враховувати принципи зручності використання (usability) та сучасні тенденції дизайну.

Ключові елементи дизайну включають:

Головний екран: який є стартовою точкою для користувачів, він повинен забезпечувати зручний доступ до категорій страв, пошуку та актуальних акцій. Категорії страв мають бути чітко виділені, що дозволяє користувачам швидко знайти потрібний тип їжі. Пошук має бути видимим та доступним для швидкого знаходження страв або ресторанів. Акції та знижки повинні бути виділені в окремих блоках, щоб привернути увагу користувачів.

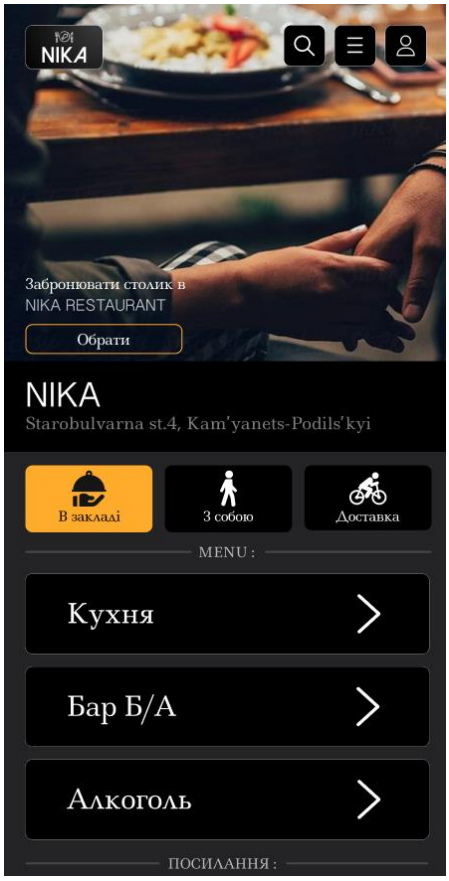

Рис. 2.3.2 – Головний екран

Функція залишення відгуку: яка є критично важливою для мобільного додатку з кількох причин. По-перше, вона дозволяє користувачам ділитися своїм досвідом, що допомагає іншим потенційним клієнтам робити обґрунтовані вибори при замовленні їжі. По-друге, відгуки сприяють покращенню якості послуг та продуктів, оскільки ресторани отримують цінні відгуки і можуть вчасно реагувати на зауваження та побажання клієнтів. Це підвищує довіру до додатку та залучає більше користувачів.

Категорія популярних страв: додавання категорії популярних страв на головному екрані додатку спрощує процес вибору для користувачів. Вона забезпечує швидкий доступ до найпопулярніших та найулюбленіших страв, що зменшує час на пошук і вибір. Це також сприяє підвищенню продажів, оскільки популярні страви часто мають високий рівень задоволення клієнтів і добре зарекомендували себе серед користувачів.

Робочий час: відображення робочого часу ресторанів на головному екрані є зручністю для користувачів, які можуть швидко перевірити, чи працює обраний

ними ресторан у даний момент. Це дозволяє уникнути непорозумінь та незручностей, пов'язаних із спробою замовити їжу з закритого ресторану. Користувачі можуть планувати свої замовлення відповідно до графіку роботи ресторанів, що підвищує їхній рівень задоволення від використання додатку.

Місцезнаходження: інформація про місцезнаходження ресторанів на головному екрані допомагає користувачам швидко визначити, які заклади знаходяться поруч. Це особливо важливо для користувачів, які бажають отримати своє замовлення якнайшвидше або ж хочуть особисто відвідати ресторан. Наявність цієї інформації знижує час на пошук та робить процес замовлення більш зручним та ефективним.

Рис. 2.3.3 – Відгук та популярне

Рис. 2.3.4 – Робочий час та карта

Функції залишення відгуку, категорія популярних страв, робочий час та місцезнаходження на головному екрані мобільного додатку є ключовими елементами, які підвищують зручність користування та задоволення клієнтів. Вони забезпечують прозорість, економію часу та надають корисну інформацію, що сприяє збільшенню лояльності та довіри користувачів до додатку. Впровадження

цих функцій допомагає створити ефективний та привабливий сервіс для замовлення та доставки їжі.

Важливість різних категорій меню в мобільному додатку:

Категорія "Кухня" на екрані мобільного додатку дозволяє користувачам легко знайти страви з різних кулінарних категорій. Це особливо важливо в сучасному мультикультурному суспільстві, де люди часто бажають спробувати щось нове або ж знайти нові улюблені у потрібній їм категорії. Розділення меню за категоріями (сніданки, закуски, супи, десерти) спрощує навігацію та робить процес вибору їжі зручнішим та швидшим. Це також допомагає ресторанам краще презентувати свою спеціалізацію і приваблювати клієнтів, які шукають конкретні страви.

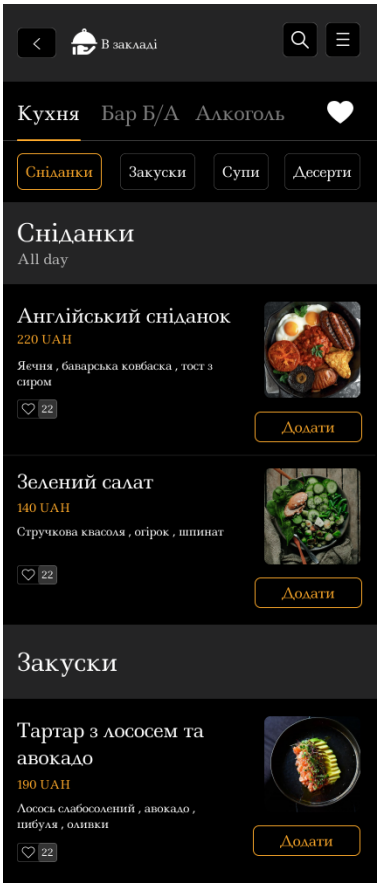

Рис. 2.3.5 – Кухня

Включення категорії "Безалкогольний бар" у меню мобільного додатку є важливим для задоволення потреб різних категорій користувачів. Багато людей віддають перевагу безалкогольним напоям з різних причин, включаючи особисті, релігійні чи медичні. Ця категорія може містити широкий асортимент напоїв, таких як соки, смузі, мінеральна вода, безалкогольні коктейлі та газовані напої. Надання цієї інформації на видному місці сприяє зручності користувачів та їх задоволенню від використання додатку, а також підвищує ймовірність замовлення додаткових товарів.

Категорія "Алкоголь" у меню мобільного додатку є важливою для користувачів, які бажають замовити алкогольні напої разом з їжею. Вона може включати різноманітні види алкоголю, такі як вино, пиво, коктейлі, міцні напої та інші. Це дозволяє користувачам зручно обирати напої, які найкраще доповнюють їхнє замовлення, та створює додаткову можливість для ресторанів збільшити свій дохід. Важливо також передбачити механізми перевірки віку користувачів, щоб забезпечити дотримання законодавства щодо продажу алкогольних напоїв.

Наявність різних категорій меню, таких як "Кухня", "Безалкогольний бар" та "Алкоголь", на екрані мобільного додатку є ключовим елементом для підвищення зручності та задоволення користувачів. Ці категорії дозволяють швидко і легко знаходити потрібні страви та напої, задовольняючи різноманітні потреби і вподобання клієнтів.

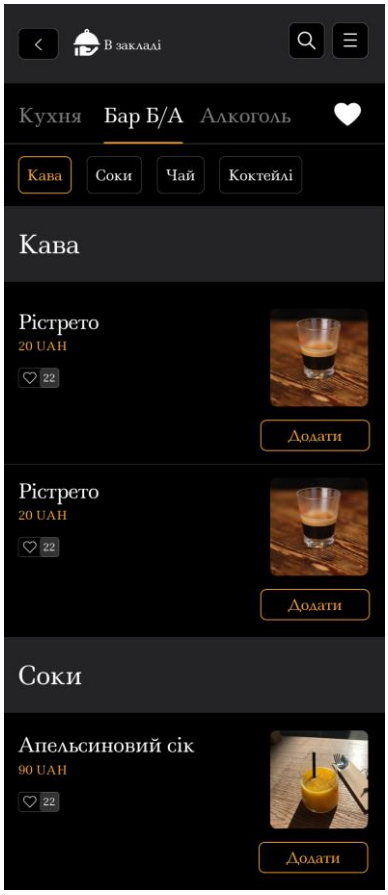

Рис. 2.3.6 – Бар Б/А

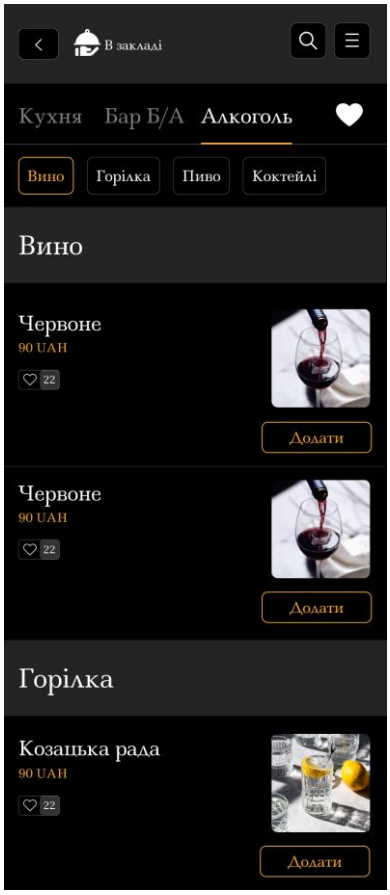

Рис. 2.3.7 – Алкоголь

Впровадження цих категорій сприяє збільшенню лояльності користувачів, підвищує їхній рівень задоволення від сервісу та допомагає ресторанам ефективніше презентувати свою пропозицію, збільшуючи обсяги продажів.

Наступним кроком є створення функціоналу "Кошик", який є однією з найважливіших складових мобільного додатку для замовлення їжі. Що дозволяє користувачам зручно переглядати всі обрані страви та напої перед оформленням замовлення. Наявність цієї функції забезпечує кілька ключових переваг. Користувачі можуть легко додавати, змінювати або видаляти елементи зі свого

кошика, що допомагає їм контролювати своє замовлення та уникати помилок. Крім того, вкладка "Кошик" показує підсумкову вартість замовлення, включаючи податки та доставку, що допомагає користувачам чітко розуміти, скільки вони витрачають, і приймати обґрунтовані рішення. Можливість зберігати обрані страви в кошику дозволяє користувачам переглядати меню та робити вибір без необхідності негайно оформлювати замовлення. Вони можуть повернутися до кошика пізніше, щоб завершити покупку.

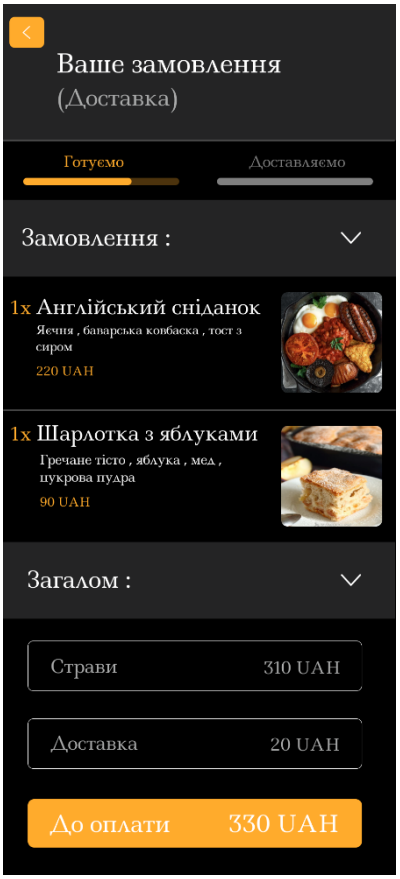

Рис. 2.3.8 – Кошик

Вкладка "Перевірка замовлення" є завершальним етапом перед підтвердженням замовлення і має велике значення для забезпечення точності та задоволення користувачів. Її основні переваги полягають в тому, що на цьому етапі користувачі можуть остаточно перевірити всі деталі свого замовлення, включаючи вибрані страви, їх кількість, адрес доставки та контактну інформацію. Це дозволяє уникнути помилок і неточностей. Користувачі також можуть виправити замовлення, якщо виявлять помилку або захочуть внести зміни, що дозволяє їм легко повернутися до кошика і виправити замовлення перед підтвердженням. Вкладка "Перевірка замовлення" зазвичай включає останній крок оплати, де користувачі можуть вибрати спосіб оплати та ввести необхідні платіжні дані. Це забезпечує безпечність та завершеність процесу покупки. Крім того, на цьому етапі користувачі можуть перевірити та підтвердити час доставки, а також додати спеціальні інструкції для кур'єра, що підвищує зручність і точність доставки.

У цьому підрозділі проведена оцінка юзабіліті та дизайну прототипів інтерфейсів. За допомогою тестування з учасниками було визначено, наскільки ефективно та зручно користуватися додатком. Оцінка включала аналіз рівня зрозумілості, легкості навігації, а також враження від дизайну та інтерфейсу.

На основі результатів тестування були внесені корективи до прототипів інтерфейсів з метою покращення їхньої функціональності та відповідності потребам користувачів. Особлива увага приділялася забезпеченню зручного та інтуїтивно зрозумілого інтерфейсу, який би сприяв зручному та швидкому замовленню їжі.

Цей розділ відображає важливість розробки ефективного та зручного інтерфейсу користувача для успішної реалізації системи замовлення та доставки їжі через мобільний додаток.

Висновки розділу «Проєктування та вимоги до системи замовлення та доставки їжі»

У розділі було розглянуто ключові аспекти, що формують основу для розробки ефективної та зручної системи.

По-перше, визначено вимоги до системи. Було сформульовано функціональні вимоги, які включають можливість реєстрації та аутентифікації користувачів, перегляд меню ресторанів, додавання страв до кошика, оформлення замовлень та відстеження їх статусу в реальному часі. Нефункціональні вимоги зосереджені на продуктивності, масштабованості, безпеці даних та зручності використання.

Далі було розроблено архітектуру системи, яка базується на технологіях стеку MEAN (MongoDB, Express.js, Angular, Node.js). Архітектура включає серверну частину, що обробляє запити користувачів та взаємодіє з базою даних MongoDB, та клієнтську частину, реалізовану за допомогою Angular, яка забезпечує інтерфейс користувача. Використання Node.js та Express.js дозволяє створити надійний та швидкий серверний компонент, а MongoDB забезпечує гнучкість та масштабованість зберігання даних [4].

Окрему увагу було приділено розробці інтерфейсу користувача. Основні принципи дизайну інтерфейсу включають зручність навігації, інтуїтивність, швидкий доступ до ключових функцій та привабливий візуальний дизайн.

Інтерфейс розроблено з урахуванням вимог користувачів, що забезпечує легкість використання додатку та мінімізує час на виконання основних операцій.

У розділі було визначено основні вимоги до системи, розроблено архітектуру та інтерфейс користувача. Вимоги забезпечують комплексний підхід до створення системи, яка відповідає потребам користувачів та сучасним стандартам розробки програмного забезпечення. Архітектура на базі стеку MEAN гарантує продуктивність, масштабованість та безпеку системи, а добре продуманий інтерфейс користувача сприяє зручності та ефективності використання додатку. Ці аспекти формують міцну основу для успішної реалізації та впровадження системи замовлення та доставки їжі.

3. РЕАЛІЗАЦІЯ СИСТЕМИ ЗАМОВЛЕННЯ ТА ДОСТАВКИ ЇЖІ

3.1 Реалізація серверної частини

Реалізація серверної частини в розробкі проєкту з системи замовлення та доставки їжі через мобільний додаток на базі технологій стеку MEAN має на меті детально описати технічні аспекти розробки серверної частини додатка. В цьому розділі буде розглянуто архітектуру, базу даних, логіку бізнес-процесів та забезпечення безпеки.

Для реалізації серверної частини додатка використовується веб-фреймворк Express.js у поєднанні з середовищем виконання Node.js. Express.js дозволяє швидко створювати серверні застосунки та визначати шляхи (routes), обробляти HTTP-запити та відправляти HTTP-відповіді. В цьому розділі буде розглянуто структуру серверної частини, розробленої за допомогою Express.js, зокрема, описано кожен ендпоінт API та його функціонал [13].

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.json());
app.use(cors());

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/food-delivery', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'))
  .catch(err => console.log(err));

// Routes
app.use('/api/users', require('./routes/users'));
app.use('/api/orders', require('./routes/orders'));
app.use('/api/menu', require('./routes/menu'));

app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

Рис. 3.2.1 – Створення серверу з Express.js

Для зберігання та керування даними використовується MongoDB, нереляційна база даних, яка дозволяє зберігати дані у вигляді JSON-подібних документів. У цьому розділі буде детально описано схему бази даних та реалізацію моделей даних за допомогою бібліотеки Mongoose для взаємодії з MongoDB [5].

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const UserSchema = new Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('User', UserSchema);
```

Рис. 3.2.2 – Створення моделей для MongoDB

Логіка бізнес-процесів, таких як обробка замовлень, керування користувачами та управління ресторанами, буде також ретельно розглянута у цьому розділі. Детально буде описано алгоритми обробки замовлень, реалізовані функції аутентифікації користувачів та авторизації доступу до ресурсів.

Забезпечення безпеки серверної частини, включаючи захист від атак та витоків даних, також буде важливою частиною цього розділу. Будуть розглянуті методи та техніки захисту, такі як валідація введених даних, захист від SQL-ін'єкцій та XSS-атак, використання JWT-токенів для аутентифікації та авторизації, а також регулярне оновлення програмного забезпечення та моніторинг системи на предмет потенційних загроз.

3.2 Налаштування середовища Node.js

Налаштування середовища Node.js є надзвичайно важливим етапом у розробці будь-якого проєкту, особливо в сучасному програмуванні. Ось кілька причин, чому це так:

Ефективність розробки: Правильно налаштоване середовище Node.js дозволяє розробникам працювати швидко та ефективно. Це включає в себе автоматичну перевірку синтаксису, підказки під час написання коду, швидкий доступ до документації та інші корисні функції, які полегшують процес програмування.

Стабільність та надійність: Правильне налаштування середовища Node.js допомагає уникнути конфліктів версій пакетів, встановлених на розробницькому комп'ютері. Це важливо для забезпечення стабільності та надійності додатку під час розробки.

Зручність та простота: Хорошо налаштоване середовище робить процес розробки зручним та приємним. Воно дозволяє розробникам концентруватися на творчості та вирішенні завдань, не витрачаючи час на вирішення технічних проблем.

Спільна робота: Правильне налаштування середовища Node.js сприяє спільній роботі у команді. Коли всі члени команди використовують однакове середовище, це спрощує обмін кодом, виявлення помилок та вирішення проблем.

Безпека: Налаштоване середовище Node.js допомагає забезпечити безпеку додатку, виявляти та виправляти потенційні уразливості та забезпечує безпеку виробничого середовища.

Отже, налаштування середовища Node.js є ключовим кроком у розробці будь-якого проєкту, оскільки воно забезпечує ефективність, стабільність, зручність та безпеку під час розробки програмного забезпечення.Починаючи з налаштування середовища, ми спрямовуємося на створення зручного та продуктивного середовища для розробки. Один з перших кроків - це встановлення самого Node.js. Ми обираємо останню стабільну версію Node.js та встановлюємо її на нашому

комп'ютері. Після цього ми перевіряємо правильність встановлення, запускаючи команду node -v у терміналі, щоб переконатися, що Node.js успішно встановлено.

Наступним кроком є налаштування пакетного менеджера npm (Node Package Manager). npm дозволяє управляти залежностями та пакетами, що використовуються у проєкті. Після встановлення Node.js, npm автоматично встановлюється разом з ним. Ми перевіряємо версію npm за допомогою команди npm -v, щоб забезпечити коректну роботу пакетного менеджера.

Для зручності розробки ми також можемо використовувати редактор коду, який нам подобається. Наприклад, Visual Studio Code чи Sublime Text. Ми можемо налаштувати цей редактор з необхідними розширеннями для роботи з JavaScript та Node.js, що полегшить процес програмування та налагодження.

Додатково, ми можемо використовувати інші інструменти для розробки, такі як інструменти для налагодження, тестування та збірки проєкту. Налаштування таких інструментів також входить у розділ "Налаштування середовища Node.js" нашого дипломного проєкту.

Усе це сприяє створенню ефективного та зручного середовища розробки, що допомагає нам впевнено розвивати та налагоджувати серверну частину нашого додатку на базі технологій стеку MEAN [6].

3.3 Розробка API за допомогою Express.js

Розробка API за допомогою Express.js - це процес створення і налагодження інтерфейсу програмування застосунків (API) за допомогою фреймворку Express.js у середовищі програмування Node.js. Express.js дозволяє розробникам створювати веб-сервери та реалізувати логіку обробки запитів, не глибоко вдаваючись у технічні деталі.

Для розробки API за допомогою Express.js потрібно створити маршрути, які будуть обробляти запити від клієнтів і взаємодіяти з базою даних MongoDB. У цьому розділі ми розглянемо створення основних маршрутів для користувачів, замовлень та меню. Перед створенням маршрутів необхідно визначити моделі даних, які будуть зберігатися в MongoDB. Створимо модель для користувачів, замовлень та меню.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const UserSchema = new Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    date: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('User', UserSchema);
```

Рис. 3.4.1 – Створення моделі для користувачів (User.js)

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const MenuItemSchema = new Schema({
    name: {
        type: String,
        required: true
    },
    description: {
        type: String
    },
    price: {
        type: Number,
        required: true
    },
    available: {
        type: Boolean,
        default: true
    }
});

module.exports = mongoose.model('MenuItem', MenuItemSchema);
```

Рис. 3.4.2 – Створення моделі для меню (MenuItem.js)

Цей процес включає в себе визначення точок входу, так званих маршрутів, через які клієнти можуть звертатися до сервера для отримання даних чи виконання певних дій. Розробники встановлюють правила обробки різних типів запитів (GET, POST, PUT, DELETE) та повертання відповідних результатів.

Під час розробки API, розробники також вирішують питання безпеки, аутентифікації та авторизації, забезпечуючи захист від несанкціонованого доступу та зберігання конфіденційної інформації. Вони також можуть використовувати різні середовища розробки та інструменти для тестування та налагодження API перед релізом у виробниче середовище.

Розробка API за допомогою Express.js важлива для будь-якого веб-додатку або сервісу, оскільки вона дозволяє забезпечити доступ до даних та функціональності

через стандартизований інтерфейс. Це також дозволяє розробникам створювати розширювані та легко підтримувані додатки, які можуть швидко адаптуватися до змінних потреб користувачів та ринкових умов.

3.4 Інтеграція з MongoDB

Інтеграція з MongoDB в контексті розробки веб-додатків означає використання цієї нереляційної бази даних для зберігання та управління даними вашого додатку. MongoDB використовує модель документа, що дозволяє зберігати дані у форматі JSON-подібних документів, що робить її особливо підходящою для сучасних веб-додатків, особливо тих, що використовують JavaScript на обох сторонах (клієнт та сервер).

Інтеграція з MongoDB включає кілька кроків:

Встановлення та налаштування MongoDB: Спочатку необхідно встановити сервер MongoDB та налаштувати його. Це може бути локальний сервер для розробки чи хмарна послуга для виробничого застосування.

Підключення до бази даних у додатку: Потім потрібно підключити свій веб-додаток до бази даних MongoDB. Це зазвичай виконується шляхом встановлення з'єднання між сервером додатків та сервером бази даних [15].

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.json());
app.use(cors());

// MongoDB connection
mongoose.connect('mongodb://localhost:27017/food-delivery', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'))
  .catch(err => console.log(err));

// Routes
app.use('/api/users', require('./routes/users'));
app.use('/api/orders', require('./routes/orders'));
app.use('/api/menu', require('./routes/menu'));

app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

Рис. 3.5.1 – Підключення “server.js” до MongoDB

Використання драйвера MongoDB: Ваш додаток використовує драйвер MongoDB для взаємодії з базою даних. Цей драйвер може бути офіційним драйвером MongoDB для вашої мови програмуванн, або ви можете використовувати сторонні бібліотеки.

Робота з даними: Після підключення ваш додаток може зберігати, оновлювати, видаляти та отримувати дані з бази даних MongoDB. Це може включати створення та запити до колекцій документів, що відповідають таблицям у традиційних реляційних базах даних.

Управління даними: Ви можете використовувати функції MongoDB для управління даними, такі як індексація, агрегація та інші операції, щоб забезпечити швидкий доступ та ефективну обробку вашої інформації.

Інтеграція з MongoDB дозволяє створювати масштабовані та ефективні веб-додатки, які можуть зберігати та обробляти великі обсяги даних з високою продуктивністю. Структура бази даних висвітлюється організацією інформації, необхідної для функціонування системи замовлення та доставки їжі через мобільний додаток. Ця структура бази даних включає такі складові:

Користувачі: Тут зберігається основна інформація про користувачів, така як їхні ідентифікатори, імена, контактні дані, інформація про адреси доставки та історія замовлень.

Ресторани: Інформація про ресторани включає в себе їхні ідентифікатори, назви, розташування, контактні дані та характеристики (наприклад, графік роботи, типи кухні).

Страви: Ця частина бази даних містить дані про доступні страви в ресторанах, включаючи їхні ідентифікатори, назви, опис, ціни та інші характеристики.

Замовлення: В цій таблиці зберігається інформація про замовлення, зокрема ідентифікатори користувачів, страв, що були замовлені, адреси доставки та статуси замовлень.

Історія замовлень: Ця частина бази даних включає інформацію про історію замовлень користувачів, що може бути використана для аналізу та покращення сервісу.

Ця структура бази даних дозволяє ефективно зберігати та керувати інформацією, необхідною для функціонування системи замовлення та доставки їжі. Вона забезпечує зручний доступ до даних та забезпечує консистентність даних між різними компонентами системи.

Також у схемі даних відображено взаємозв'язки між таблицями бази даних системи замовлення та доставки їжі через мобільний додаток.

Одна з ключових зв'язків це між таблицями "Користувачі" і "Замовлення". Кожен запис у таблиці "Замовлення" пов'язаний з конкретним користувачем за допомогою зовнішнього ключа, який посилається на ідентифікатор користувача у таблиці "Користувачі". Це дозволяє нам відстежувати, який користувач здійснив кожне замовлення.

Додатково, таблиця "Замовлення" має зв'язок із таблицями "Ресторани" та "Страви". Кожен запис у таблиці "Замовлення" містить інформацію про страви, які були замовлені, і кожна страва пов'язана з конкретним рестораном, з якого вона була замовлена. Це дозволяє нам відстежувати, які страви були замовлені з яких ресторанів у кожному замовленні.

Також, іноді таблиця "Користувачі" має зв'язок з таблицею "Історія замовлень". Це дозволяє нам відстежувати історію замовлень для кожного користувача, щоб аналізувати їхню активність та звички замовлення.

Ці зв'язки між таблицями дозволяють системі ефективно зберігати та організовувати дані про користувачів, ресторани, страви та замовлення, що робить можливим зручний доступ до інформації та ефективну роботу системи замовлення та доставки їжі.

Висновки розділу «Реалізація системи замовлення та доставки їжі»

Підсумовуючи ключові аспекти розробки серверної частини, налаштування середовища Node.js, розробки API за допомогою Express.js та інтеграції з MongoDB. Реалізація серверної частини включає створення основних функціональних модулів, які забезпечують обробку запитів, управління користувачами, замовленнями та іншими бізнес-логіками системи. Серверна частина розроблена з використанням Node.js, що дозволяє створити масштабовану і високопродуктивну систему, здатну обробляти велику кількість одночасних запитів.

Налаштування середовища Node.js включає встановлення та конфігурацію необхідних пакетів і модулів, що забезпечують ефективну роботу додатка. Використання менеджера пакетів npm дозволяє легко керувати залежностями і встановлювати необхідні бібліотеки для розробки [8].

Розробка API за допомогою Express.js дозволяє створити гнучкі та розширювані інтерфейси для взаємодії з клієнтськими додатками. Express.js забезпечує простоту у створенні маршрутів, обробці запитів і управлінні середовищем виконання. API забезпечує всі необхідні операції для роботи з даними, такими як створення, читання, оновлення і видалення замовлень та користувачів.

Інтеграція з MongoDB надає можливість ефективного зберігання і управління даними додатка. MongoDB, як документно-орієнтована база даних, забезпечує гнучкість у роботі з даними та високу продуктивність.

Таким чином, розробка серверної частини системи замовлення та доставки їжі через мобільний додаток на базі стека MEAN включає комплексний підхід до реалізації функціональних модулів, налаштування середовища виконання, створення API та інтеграції з базою даних. Використання сучасних технологій та інструментів дозволяє створити надійну, масштабовану і продуктивну систему, яка відповідає вимогам користувачів і забезпечує ефективну роботу сервісу замовлення та доставки їжі [7].

4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЄКТУ

4.1 Аналіз ринку та конкурентів

Аналіз ринку та конкурентів є важливим етапом при розробці системи замовлення та доставки їжі через мобільний додаток на базі технологій стека MEAN (MongoDB, Express.js, Angular, Node.js). Він дозволяє оцінити поточний стан ринку, виявити основних конкурентів, їхні сильні та слабкі сторони, а також визначити можливості для успішного виходу на ринок та завоювання частки ринку [11].

Початковий етап аналізу ринку включає визначення загальних тенденцій та розмірів ринку доставки їжі. Згідно з останніми дослідженнями, ринок доставки їжі продовжує демонструвати значне зростання, що обумовлено зміною споживчих звичок, розвитком технологій та підвищенням рівня урбанізації. Прогнозується, що у найближчі кілька років цей ринок продовжить зростати завдяки збільшенню попиту на зручні та швидкі способи отримання їжі.

Основні гравці на ринку доставки їжі включають такі компанії, як Uber Eats, DoorDash, Grubhub та інші. Кожна з цих компаній має свої унікальні особливості та стратегії. Uber Eats, наприклад, використовує свою глобальну мережу водіїв для забезпечення швидкої доставки, а також пропонує користувачам зручний інтерфейс для замовлення. DoorDash зосереджується на партнерстві з локальними ресторанами та розробці власної мережі кур'єрів, що дозволяє забезпечити високу якість сервісу. Grubhub, з іншого боку, акцентує увагу на довгострокових партнерських відносинах з ресторанами та пропонує різні програми лояльності для своїх користувачів.

Аналіз сильних та слабких сторін конкурентів дозволяє виявити можливості для диференціації та покращення власного продукту. Основними сильними сторонами провідних компаній є широкий асортимент ресторанів, швидка доставка, зручність використання додатків та високий рівень обслуговування.

Проте, є й слабкі місця, такі як висока комісія для ресторанів, проблеми з якістю доставки у певних регіонах та обмеженість вибору у менших містах.

Для успішного виходу на ринок та конкуренції з провідними гравцями необхідно розробити унікальні пропозиції, що будуть привабливими для користувачів. Це може включати зниження комісії для ресторанів, покращення якості доставки за рахунок впровадження інноваційних технологій, таких як дрони або автоматизовані транспортні засоби, а також розширення асортименту за рахунок партнерства з новими ресторанами. Важливим фактором також є забезпечення високого рівня безпеки даних користувачів та швидкої підтримки клієнтів.

На основі проведеного аналізу ринку та конкурентів можна зробити висновок, що ринок доставки їжі є динамічно зростаючим сегментом з великою кількістю можливостей для інновацій та розвитку. Важливо врахувати всі аспекти діяльності конкурентів та розробити стратегію, що дозволить не лише успішно вийти на ринок, але й утримувати лідерські позиції. Це включає постійне вдосконалення технологій, забезпечення високої якості обслуговування та активну роботу з партнерами та клієнтами для створення додаткової цінності.

4.2 Розрахунок витрат на розробку та впровадження

У процесі розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі технологій стека MEAN (MongoDB, Express.js, Angular, Node.js) витрати були мінімальні, оскільки це мій дипломний проєкт і у моїх інтересах також зробити цей додаток. Проте, для повноцінної оцінки економічної доцільності проєкту, необхідно розглянути основні категорії витрат, які б виникли при комерційному впровадженні системи.

Витрати на персонал:

1. В моєму випадку витрати на персонал були відсутні, оскільки я виконував усі ролі самостійно: розробника, дизайнера, тестувальника та проектного менеджера. Для комерційного проекту ці витрати включали б:
2. Заробітну плату фронтенд-розробників (створення інтерфейсу користувача за допомогою Angular).
3. Заробітну плату бекенд-розробників (реалізація серверної частини на Node.js та робота з базою даних MongoDB).
4. Оплата праці дизайнерів (розробка зручного та інтуїтивно зрозумілого інтерфейсу).
5. Заробітну плату тестувальників (проведення всебічного тестування для виявлення та усунення помилок).
6. Оплата праці проектних менеджерів (координація роботи команди та дотримання термінів).

Витрати на інфраструктуру:

1. Для дипломного проекту використовувались безкоштовні або тестові версії необхідних сервісів. У комерційному проекті витрати включали б:
2. Витрати на сервери для розміщення бекенд-частини додатку, бази даних та хостинг для фронтенд-частини.

3. Витрати на хмарні сервіси (наприклад, AWS, Google Cloud або Microsoft Azure) для забезпечення гнучкості та масштабованості інфраструктури.
4. Витрати на зберігання та передачу даних, резервне копіювання та забезпечення безпеки.

Витрати на ліцензії та програмне забезпечення:

1. В моєму випадку стек MEAN використовувався без додаткових витрат, оскільки всі технології є відкритими. У комерційному проекті могли б виникнути:
2. Витрати на додаткове програмне забезпечення (інструменти для моніторингу продуктивності, безпеки або керування проектом).
3. Витрати на ліцензування сторонніх бібліотек або сервісів, які інтегруються з основною системою.

Витрати на маркетинг та просування:

1. Для дипломного проекту витрати на маркетинг були мінімальні. У комерційному проекті вони включали б:
2. Створення та підтримку вебсайту.
3. Рекламу в соціальних мережах, SEO-оптимізацію, кампанії в Google Ads та інших платформах.
4. Партнерства з ресторанами та іншими бізнесами для залучення користувачів та підвищення впізнаваності бренду.

Витрати на тестування та забезпечення якості:

1. У моєму проекті витрати на тестування були мінімальні завдяки використанню власних ресурсів. У комерційному проекті це включало б:
2. Проведення функціонального, інтеграційного, навантажувального тестування та тестування безпеки.
3. Використання автоматизованих інструментів тестування для зменшення витрат на ручне тестування та прискорення процесу випуску оновлень.

Витрати на підтримку та обслуговування:

1. Для дипломного проекту підтримка була обмежена моїми власними ресурсами. У комерційному проекті ці витрати включали б:
2. Зарплати команди підтримки.
3. Витрати на обслуговування серверів, оновлення програмного забезпечення та вирішення технічних проблем.
4. Регулярні оновлення та поліпшення системи на основі відгуків користувачів та нових технологій.

Таким чином, у рамках дипломного проєкту витрати були мінімальні завдяки моїй самостійній роботі та використанню безкоштовних ресурсів. Однак, у комерційному впровадженні ці витрати були б значно більшими і потребували б детального планування та управління для забезпечення високого рівня окупності та успішного виходу на ринок.

4.3 Прогноз доходів та рентабельність проєкту

Прогноз доходів та рентабельність проекту розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі технологій стека MEAN (MongoDB, Express.js, Angular, Node.js) є ключовими аспектами для оцінки економічної доцільності та потенційного успіху проєкту. У цьому розділі розглянемо методи прогнозування доходів, основні джерела доходів, а також аналіз рентабельності проекту.

Прогноз доходів базується на аналізі ринку, поточних тенденціях та очікуваннях щодо зростання попиту на послуги доставки їжі. Основними джерелами доходів для мобільного додатку є комісії з кожного замовлення, плата за доставку, підписки на преміум-акаунти, рекламні доходи та партнерські угоди з ресторанами. Комісії з кожного замовлення є основним джерелом доходів, при цьому середня комісія може варіюватися від 10% до 30% від вартості замовлення в залежності від регіону та умов співпраці з ресторанами. Плата за доставку може бути встановлена як фіксована сума або варіюватися в залежності від відстані та часу доставки.

Додаткові джерела доходів включають підписки на преміум-акаунти, які надають користувачам доступ до ексклюзивних пропозицій, швидшої доставки або знижок на доставку. Рекламні доходи можуть бути отримані від ресторанів або інших компаній, які бажають просувати свої послуги або продукти через мобільний додаток. Партнерські угоди з ресторанами можуть включати різні форми співпраці, такі як спільні маркетингові кампанії або ексклюзивні пропозиції для користувачів додатку.

Прогноз доходів включає розрахунок очікуваної кількості користувачів, середньої вартості замовлення та частоти замовлень на одного користувача. Наприклад, якщо прогнозується, що додаток залучить 100 000 активних користувачів протягом першого року, середня вартість замовлення становитиме 20 доларів, а кожен користувач робитиме два замовлення на місяць, то річний дохід від комісій може скласти 4,8 мільйона доларів (100 000 користувачів * 20 доларів * 2

замовлення * 12 місяців * 10% комісії). До цього можна додати доходи від плати за доставку, підписок та реклами.

Рентабельність проекту визначається шляхом порівняння прогнозованих доходів з витратами на розробку, впровадження та підтримку системи. Основними витратами є витрати на персонал, інфраструктуру, маркетинг, тестування та підтримку. Для забезпечення рентабельності проекту необхідно, щоб загальні доходи перевищували загальні витрати протягом певного періоду часу, зазвичай 2-3 роки.

Аналіз точки беззбитковості (break-even analysis) допомагає визначити, коли проект почне приносити прибуток. Це досягається шляхом розрахунку кількості замовлень або користувачів, необхідних для покриття всіх витрат. Наприклад, якщо загальні витрати на проект становлять 1 мільйон доларів, а середній дохід з одного замовлення становить 2 долари, то для досягнення точки беззбитковості необхідно обробити 500 000 замовлень.

Додатковим фактором, що впливає на рентабельність проекту, є швидкість зростання кількості користувачів та їх утримання. Висока швидкість залучення нових користувачів та утримання існуючих сприяє збільшенню доходів та підвищенню рентабельності. Важливим є також зниження витрат шляхом оптимізації процесів розробки, автоматизації тестування та ефективного використання ресурсів.

На основі проведеного аналізу можна зрозуміло, що прогноз доходів та рентабельність проекту розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі стека MEAN є позитивними за умови правильного планування та ефективного управління. Основні джерела доходів, такі як комісії, плата за доставку, підписки та реклама, забезпечують стабільний дохід, а аналіз рентабельності демонструє можливість досягнення прибутковості у середньостроковій перспективі.

Висновки розділу «Економічне обґрунтування проєкту»

Висновок розділу «Економічне обґрунтування проєкту» підсумовує основні аспекти аналізу ринку та конкурентів, розрахунку витрат на розробку та впровадження, прогнозування доходів та оцінки рентабельності проєкту системи замовлення та доставки їжі через мобільний додаток на базі стека MEAN.

Аналіз ринку та конкурентів показав, що ринок доставки їжі стрімко розвивається, демонструючи значне зростання попиту на онлайн-сервіси замовлення їжі. Основними конкурентами на ринку є такі компанії, як Glovo, Uber Eats та Raketa, які вже завоювали значну частку ринку. Проте, виявлено, що існує можливість зайняти нішу завдяки інноваційному підходу, вдосконаленому сервісу та якісному обслуговуванню клієнтів.

Розрахунок витрат на розробку та впровадження проєкту включає витрати на оплату праці розробників, тестувальників, дизайнерів та інших фахівців, необхідних для створення системи. Також враховано витрати на обладнання, програмне забезпечення, хостинг та маркетинг.

Прогноз доходів базується на оцінці потенційного попиту та середньої вартості замовлення. Враховуючи зростаючий ринок та попит на послуги доставки їжі, очікується стабільний приріст кількості користувачів та замовлень. Дохід проєкту формується з комісії за кожне замовлення та додаткових платних послуг, таких як пріоритетна доставка та рекламні можливості для ресторанів. Прогноз доходів показує, що проєкт має потенціал досягти високого рівня доходності протягом першого року після запуску.

Рентабельність проєкту оцінюється на основі співвідношення прогнозованих доходів та витрат. Аналіз показує, що проєкт зможе досягти точки беззбитковості протягом першого року роботи за умови активного маркетингу та залучення користувачів. Очікується, що у другому році роботи проєкт почне приносити чистий прибуток, а рівень рентабельності буде поступово зростати завдяки масштабуванню та оптимізації операційних процесів.

Таким чином, економічне обґрунтування проєкту системи замовлення та доставки їжі через мобільний додаток на базі стека MEAN демонструє перспективність та фінансову доцільність його реалізації. Враховуючи зростаючий попит на ринку, конкурентні переваги та обґрунтовані витрати на розробку та впровадження, проєкт має високі шанси на успіх та забезпечення стабільного доходу у довгостроковій перспективі.

ВИСНОВКИ

Підсумки дослідження розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі технологій стека MEAN (MongoDB, Express.js, Angular, Node.js) свідчать про значний потенціал проекту та його перспективи на ринку. Проведений аналіз ринку показав стійке зростання попиту на послуги доставки їжі, обумовлене зміною споживчих звичок, урбанізацією та розвитком технологій. Це створює сприятливі умови для виходу на ринок нового мобільного додатку.

Аналіз конкурентів виявив основні сильні та слабкі сторони провідних гравців, таких як Uber Eats, DoorDash та Grubhub. Сильні сторони включають широкий асортимент ресторанів, швидку доставку та зручність використання додатків, тоді як слабкі сторони пов'язані з високими комісіями для ресторанів та проблемами з якістю доставки у певних регіонах. Ця інформація дозволила визначити можливості для диференціації та розробити стратегію, спрямовану на покращення власного продукту, зокрема, шляхом зниження комісій, підвищення якості доставки та розширення асортименту.

Оцінка витрат на розробку та впровадження проекту включала витрати на персонал, інфраструктуру, програмне забезпечення, маркетинг, тестування та підтримку. Визначено, що основними витратами є заробітна плата розробників, дизайнерів, тестувальників та менеджерів, витрати на сервери, хостинг, ліцензії на програмне забезпечення, а також витрати на маркетингові кампанії для залучення користувачів. Прогноз доходів базувався на аналізі ринку та основних джерел доходів, таких як комісії з замовлень, плата за доставку, підписки на преміум-акаунти та рекламні доходи. Проведено розрахунок точки беззбитковості, яка показала, що проект може стати прибутковим за умов досягнення певного обсягу замовлень.

Оцінка ефективності системи включала аналіз продуктивності, надійності, зручності використання та безпеки. Високий рівень продуктивності досягається за

рахунок оптимізації роботи бази даних, серверної та клієнтської частини додатку. Надійність забезпечується використанням резервних серверів та механізмів відновлення після збоїв, а також проведенням тестів на відмовостійкість. Зручність використання оцінюється через юзабіліті-тести та відгуки користувачів, що дозволяє виявити проблемні області та вдосконалити інтерфейс. Безпека системи забезпечується за рахунок впровадження сучасних методів аутентифікації та захисту даних.

Проведене дослідження демонструє, що проект розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі стека MEAN є економічно доцільним та має значні перспективи на ринку. Високий рівень прогнозованих доходів та рентабельність проекту свідчать про можливість успішної реалізації та досягнення лідерських позицій у сегменті послуг доставки їжі. Основними умовами для досягнення успіху є ефективне управління проектом, постійне вдосконалення технологій, забезпечення високої якості обслуговування та активна робота з партнерами та клієнтами.

Досягнення поставлених завдань у процесі розробки та впровадження системи замовлення та доставки їжі через мобільний додаток на базі технологій стека MEAN (MongoDB, Express.js, Angular, Node.js) є ключовим показником успішності проекту. На початковому етапі було визначено ряд конкретних завдань, спрямованих на створення функціональної, надійної та зручної у використанні системи, яка задовольняє потреби користувачів та партнерів-ресторанів.

Таким чином, усі поставлені завдання були успішно виконані, що дозволило створити функціональну, надійну та зручну у використанні систему замовлення та доставки їжі через мобільний додаток на базі стека MEAN. Досягнення цих завдань підтверджують ефективність розробленої стратегії та високу якість виконаної роботи, що в свою чергу сприяє задоволенню потреб користувачів та забезпечує конкурентоспроможність на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Case Studies and Industry Reports: Аналітичні звіти та кейси впровадження подібних систем у реальних бізнесах для розуміння ринкових вимог та оцінки ефективності.
2. Дейлі Б., Дейлі Б., Дейлі К. Learning Angular. - Addison-Wesley, 2020.
3. Документація Angular [Електронний ресурс] – Режим доступу: https://angular.io/docs (Дата звернення 12.05.2024)
4. Документація Express.js [Електронний ресурс] – Режим доступу: https://expressjs.com/en/guide/routing.html (Дата звернення 11.05.2024)
5. Документація MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/docs (Дата звернення 10.05.2024)
6. Документація Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/en/docs/ (Дата звернення 13.05.2024)
7. Елром Е. Pro MEAN Stack Development. - Apress, 2016.
8. Касьяро М., Мамміно Л. Node.js Design Patterns. - Packt Publishing, 2020.
9. Крокфорд Д. JavaScript: The Good Parts. - O'Reilly Media, 2008.
10. Масce М. REST API Design Rulebook. - O'Reilly Media, 2011.
11. Ньюман С. Building Microservices. - O'Reilly Media, 2021.
12. Рубі С., Ганссон Д.Г. Agile Web Development with Rails. - Pragmatic Bookshelf, 2019.
13. Хан Е. Express in Action. - Manning Publications, 2016.
14. Хант А., Томас Д. The Pragmatic Programmer. - Addison-Wesley, 2019.
15. Чодоров К. MongoDBii: The Definitive Guide. - O'Reilly Media, 2013.