Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: "Розробка інформаційної системи для керування закладами громадського харчування".

Виконав: студент 4 курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кізін Гліб Олегович
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ткаченко О. М.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф ОТ Богомолов С.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О. Н.
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк _____
«12» березня 2023 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кізін Гліб Олегович

1. Тема роботи – Розробка інформаційної системи для керування закладами громадського харчування.

Керівник роботи: Ткаченко Олександр Миколайович, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: інформація про заклади харчування – назви та розташування закладів, характеристики закладів, контактна інформація; інформація про меню та страви – перелік страв, які пропонуються в закладі, ціни на страви, інгредієнти та рецепти страв, інформація про алергени та харчові обмеження; інформація про персонал – склад персоналу, графіки роботи персоналу; фінансові дані – доходи та витрати закладу, інформація про оплату; координати вершин трикутників; вихідні дані – система контролю за закладом громадського харчування, звіти сформовані на основі даних з бази даних і рекомендації запропоновані нейронною мережею на основі даних зі звітів.

4. Зміст текстової частини: аналіз певних існуючих реалізацій їх порівняльний аналіз; обґрунтування вибору архітектурних рішень; моделювання основних части; розробка інтерфейсу програми; розробка загального алгоритму роботи програми; процес навчання нейронної мережі.

5. Перелік ілюстративного матеріалу: приклади реалізації інформаційних систем; моделі графічного інтерфейсу; діаграми алгоритмів; графіки процесу навчання нейронної мережі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ткаченко О. М. к.т.н. доц. каф ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку та наявних аналогів інформаційних систем для закладів громадського харчування	13.03.24 – 22.03.24	Виконано
2	Розробка моделей та загальної архітектури інформаційної системи	25.03.24 – 19.04.24	Виконано
3	Розробка алгоритмів роботи та інтерфейсу системи	22.04.24 – 10.05.24	Виконано
4	Навчання нейронної мережі, тестування системи та нейронної мережі	13.05.24 – 24.05.24	Виконано
5	Оформлення матеріалів до захисту БДР	27.05.24 – 03.06.24	Виконано

Студент Кізін Г.О. Кізін

(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи О. М. Ткаченко

(підпис) (ініціали та прізвище)

Анотація

УДК 00492

Кізін Г. О. Розробка інформаційної системи для керування закладами громадського харчування : бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 65 с.

На укр. мові. Рестхелп. : 23 назв ; рис. : 38; табл. 2.

У бакалаврській дипломній роботі було розроблено програмні засоби для інформаційної системи, яка допомагає у керуванні закладами громадського харчування, використовуючи новітні технології. Основною метою цієї роботи було створення комплексного рішення, яке забезпечить ефективне управління ресурсами, оптимізацію процесів обслуговування клієнтів та підвищення загальної продуктивності закладу. Розроблені програмні модулі дозволяють ефективно відслідковувати та контролювати роботу таких закладів, включаючи управління запасами, моніторинг діяльності персоналу, аналіз продажів та генерацію звітів. Крім того, система надає можливість клієнтам зручно замовляти страви та бронювати столи через інтерфейс веб-додатку, що значно спрощує процес взаємодії з закладом.

Для створення цього застосунку використовувались мова програмування Java та фреймворк Spring, а зберігання даних здійснюється за допомогою реляційної бази даних PostgreSQL. Такий підхід дозволяє створити систему, яка відповідає найсучаснішим стандартам та вимогам до розробки програмних засобів для управління закладами громадського харчування.

Ключові слова: програмні засоби, інформаційна система, заклад громадського харчування, новітні технології, оптимізація, продуктивність.

Abstracts

UDC 00492

Kizin H.O. Development of an information system for managing public catering establishments : bachelor's thesis in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 65 p.

In Ukrainian language. Resthelp: 20 titles ; fig. : 38; tabl. 2.

In the bachelor's qualification work, software tools were developed for an information system that helps manage catering establishments using the latest technologies. The main goal of this project was to create a comprehensive solution that ensures efficient resource management, optimization of customer service processes, and overall productivity improvement of the establishment. The developed software modules allow for effective tracking and control of such establishments, including inventory management, staff activity monitoring, sales analysis, and report generation. Additionally, the system provides customers with the convenience of ordering dishes and booking tables through a web application interface, significantly simplifying the interaction process with the establishment.

The application was developed using the Java programming language and the Spring framework, while data storage is managed using the PostgreSQL relational database. This approach allows for the creation of a system that meets the latest standards and requirements for the development of software tools for managing catering establishments.

Keywords: software tools, information system, catering establishment, latest technologies, optimization, productivity.

ЗМІСТ

ВСТУП

Актуальність вибору теми дослідження. Зараз сфера громадського харчування переживає активний розвиток, що вимагає ефективних технологічних рішень для оптимізації процесів управління. З урахуванням зростання популярності ресторанів, кафе, закладів швидкого харчування тощо, виникає потреба у зручних інструментах для автоматизації управління цими закладами. Заклади громадського харчування потребують комплексного підходу до управління, який включає в себе облік і контроль запасів, управління персоналом, обробку замовлень і платежів, а також аналіз даних для прийняття управлінських рішень. Розробка програмних засобів інформаційної системи для керування закладами громадського харчування дозволить оптимізувати бізнес-процеси, зменшити час і ресурси, витрачені на рутинні операції, та забезпечити більш ефективне управління ресурсами [1].

Інформаційна система дозволить вдосконалити обслуговування клієнтів, спростити процес замовлення страв, покращити сервіс і зробити його більш індивідуалізованим. З впровадженням новітніх технологій, таких як інтернет речей (IoT), штучний інтелект (AI) та аналітика даних, можливості для створення потужних програмних засобів для управління громадським харчуванням значно розширюються [2].

Отже, розробка програмних засобів інформаційної системи для керування закладами громадського харчування відповідає сучасним потребам ринку та відкриває широкі можливості для підвищення ефективності та якості обслуговування у цій галузі.

Мета та завдання дослідження. *Мета бакалаврської дипломної роботи* полягає у і підвищення ефективності функціонування бізнес-процесів за рахунок розробки інформаційної системи для управління закладами громадського харчування.

Основними **задачами** дослідження є:

- Розробка архітектури програмного забезпечення, включаючи вибір технологічних стеків, баз даних та архітектурних патернів, які найбільш відповідають потребам цільових користувачів.
- Створення функціональних модулів системи, таких як управління замовленнями, облік запасів, управління персоналом, обробка платежів тощо.
- Реалізація програмного забезпечення на базі розробленої архітектури та функціональності.
- Впровадження програмної системи та проведення тестування для перевірки її працездатності та відповідності вимогам.

Ці завдання спрямовані на створення ефективної та інтуїтивно зрозумілої інформаційної системи, яка відповідає потребам сучасних закладів громадського харчування і сприяє підвищенню їхньої продуктивності та конкурентоспроможності.

Об’єкт дослідження – процес розробки інформаційной системи для керування закладами громадського харчування.

Предмет дослідження - методи і засоби розробки програмних засобів для керування закладами громадського харчування мовою програмування Java**.**

Методи дослідження. У процесі досліджень використовувались теорія інформаційних систем для розробки загальної архітектури інформаційної системи; системний аналіз для визначення вимог до системи, розробки технічного завдання та моделювання бізнес-процесів та ідентифікації ключових елементів системи та взаємозв'язків між ними; теорія проектування баз даних для розробки структури баз даних; об'єктно-орієнтоване програмування для реалізації програмних компонентів системи на основі принципів об'єктно-орієнтованого підходу.

Новизна отриманих результатів

1. **Вперше запропоновано** архітектуру нейронної мережі, особливість якої полягає в можливості формування прогнозів та рекомендацій для подальшого

розвитку закладів громадського харчування на основі аналізу даних продажів, що дозволило реалізувати гнучке налаштування системи до потреб користувачів.

2. **Отримала подальшого розвитку** архітектура серверної частини інформаційної системи, у якій на відміну від існуючих, додано модуль спеціально навченої нейронної мережі, що дозволило отримати можливість використовувати її для формування звітів з пропозиціями щодо покращення роботи закладів.

Практична цінність. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для управління закладами громадського харчування. Розроблені програмні модулі дозволяють ефективно відслідковувати та контролювати роботу таких закладів, включаючи управління запасами, моніторинг діяльності персоналу, аналіз продажів та генерацію звітів.

Особистий внесок здобувача. Всі результати, представлені у бакалаврській дипломній роботі, отримані автором особисто. У роботі, написаній у співавторстві, автору належить розробка мікросервісної архітектури інформаційної системи для керування закладами громадського харчування.

Апробація. Описані у роботі положення доповідалися на LIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2024) і опубліковані в тезах доповіді цієї конференції.

Публікації. Результати роботи були опубліковані в матеріалах конференцій: LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024) [3].

1 АНАЛІЗ РОЗВИТКУ ПРОГРАМНИХ ЗАСОБІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ КЕРУВАННЯ ЗАКЛАДАМИ ГРОМАДСЬКОГО ХАРЧУВАННЯ

1.1 Аналіз стану програмного забезпечення для інформаційних систем для керування закладами громадського харчування

На сьогоднішній день ринок програмного забезпечення для інформаційних систем управління закладами громадського харчування відзначається різноманіттям рішень з різними функціональними можливостями та характеристиками. Усі з цих систем мають певні переваги та недоліки.

Економічні системи управління – програми спеціалізуються на фінансовому обліку, включаючи облік доходів та витрат, управління витратами і звітність. Вони зазвичай надають зручний інтерфейс для ведення бухгалтерського обліку, але можуть бути обмежені в інших аспектах управління, таких як управління персоналом чи облік запасів.

POS-системи (Point of Sale) – спеціалізуються на обробці замовлень, оплаті та обліку продажів. Вони зазвичай дуже ефективні у роботі з фронт-офісом закладу, але можуть бути менш ефективними в управлінні аспектами, що не пов'язані з продажами, такими як управління запасами чи планування персоналу.

Інтегровані ERP-системи – надають комплексний підхід до управління закладами громадського харчування, об'єднуючи фінансовий, виробничий, кадровий та інші аспекти управління в єдиній системі. Вони можуть бути дуже потужними, але вимагають значних зусиль на впровадження та підтримку.

Хмарні інформаційні системи – надають зручний доступ до даних з будь-якого пристрою та можуть бути більш еластичними у впровадженні, але можуть виникати питання щодо конфіденційності даних.

Ці рішення відображають різноманітні потреби та вимоги закладів громадського харчування. Підбір оптимального програмного забезпечення

повинен базуватися на конкретних потребах та бізнес-процесах кожного закладу, а також на можливостях та обмеженнях кожного рішення.

1.2 Порівняльний аналіз аналогів

Порівняльний аналіз існуючих аналогів програмного забезпечення для інформаційних систем управління закладами громадського харчування розглянувши переваги та обмеження деяких з них, а саме Toast POS, Square for Restaurants, Revel Systems, TouchBistro.

Toast POS (рис. 1.1) є хмарною системою точки продажу (POS), спеціально розробленою для ресторанного бізнесу. Вона дозволяє ресторанам керувати замовленнями, оплатами, меню та персоналом [4].

Ключовими функції Toast POS включають: управління замовленнями, а саме Toast POS дозволяє ресторанам приймати замовлення легко та ефективно, будь то в режимі обслуговування за столом, забирання з собою або доставка, налаштування меню, ресторани можуть легко налаштовувати свої меню в Toast POS, додавати або видаляти позиції, налаштовувати ціни та конфігурувати модифікатори, сервери можуть приймати замовлення безпосередньо за столом, використовуючи пристрої з Toast POS, що підвищує точність та швидкість обслуговування, система підтримує різні способи оплати, включаючи кредитні картки, мобільні платежі та подарункові картки, система інтегрується з популярними платіжними процесорами для забезпечення безпеки та ефективності транзакцій, Toast POS надає детальні звіти та інструменти аналітики, що дають власникам ресторанів уявлення про їхні продажі, інвентар та результативність персоналу, ресторани можуть відстежувати свій інвентар в реальному часі з Toast POS, допомагаючи їм керувати рівнями запасів, зменшувати втрати та оптимізувати закупівлі, система містить функції для розкладу, обліку часу та управління заробітною платою, що дозволяє ресторанам ефективно керувати персоналом.

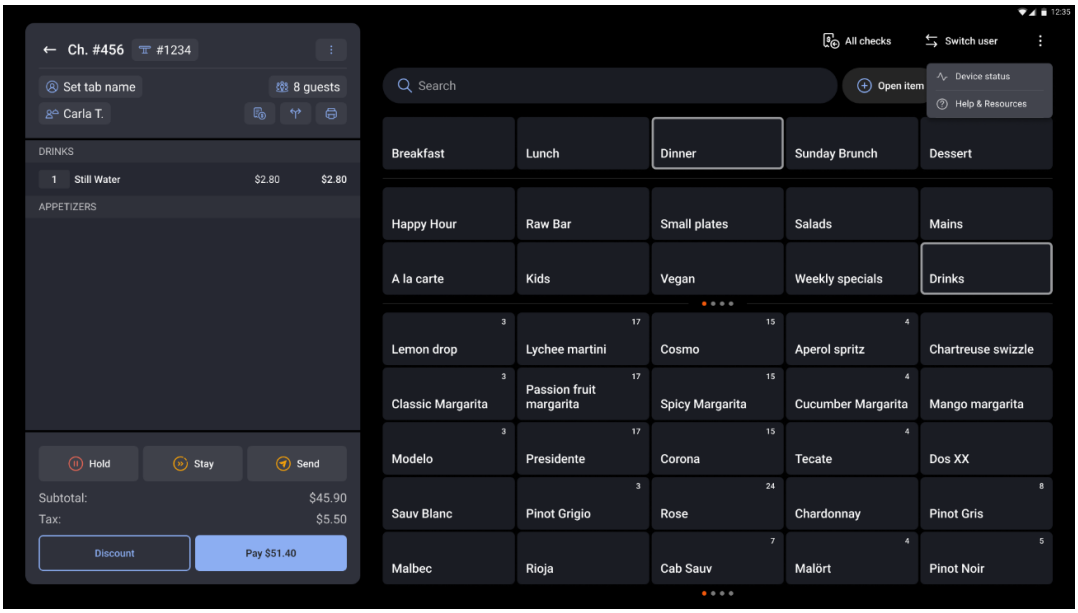

Рисунок 1.1 – Інтерфейс програмного застосунку «Toast POS»

До переваг Toast POS можна віднести: легкість використання, а саме Toast POS має інтуїтивний і простий інтерфейс, що полегшує навчання персоналу та швидше впровадження системи, Toast POS розроблено спеціально для потреб ресторанного бізнесу, що означає, що вона має функції, які ідеально підходять для цієї галузі, система дозволяє ресторанам налаштовувати свої меню, акції, знижки та інші аспекти відповідно до їхніх потреб, Toast POS працює в хмарі, що робить доступ до даних можливим з будь-якого місця з Інтернетом і забезпечує безпеку даних, система надає детальні звіти та аналітику, які допомагають ресторанам аналізувати їхній бізнес та приймати обґрунтовані рішення, Toast POS може інтегруватися з іншими програмами, такими як облік, інвентаризація, бронювання столиків тощо, для більшої ефективності управління рестораном.

До переваг Toast POS можна віднести: вартість впровадження та використання Toast POS може бути високою, особливо для менших ресторанів, так як Toast POS працює в хмарі, він потребує постійного підключення до Інтернету, що може стати проблемою в разі відключення мережі, налаштування системи може вимагати певного часу та зусиль, особливо якщо ресторан має складну структуру меню або процеси, впровадження нової системи може вимагати часу на тренування персоналу, щоб вони могли використовувати її

ефективно, Toast POS розроблено для ресторанного бізнесу, вона може бути менш підходящою для певних типів закладів, таких як кафе або бари.

Загалом, Toast POS - це потужний інструмент для автоматизації управління рестораном, проте перед вибором системи варто ретельно вивчити її переваги та недоліки і порівняти з потребами конкретного бізнесу.

Square for Restaurants (рис. 1.2) є програмою для точок продажу (POS), розробленою компанією Square, яка спеціалізується на фінансових технологіях та рішеннях для бізнесу. Ця програма призначена для ресторанного бізнесу і пропонує рішення для керування замовленнями, оплатами, інвентарем та персоналом [5].

Основні функції Square for Restaurants включають: додаток дозволяє приймати замовлення легко та швидко, включаючи замовлення за столиками, на винос або з доставкою, ресторани можуть легко керувати своїм меню, додавати, видаляти або змінювати позиції, налаштовувати ціни та вказувати інші деталі, Square for Restaurants допомагає вести облік інвентарю, автоматично оновлюючи його після кожного продажу, що допомагає уникнути нестачі товарів, програма підтримує різні способи оплати, включаючи кредитні картки, мобільні платежі та інші, Square for Restaurants надає звіти та аналітику про продажі, інвентар та продуктивність персоналу, допомагаючи ресторанам приймати обґрунтовані рішення, програма дозволяє керувати розкладами, оплатою праці та іншими аспектами персоналу.

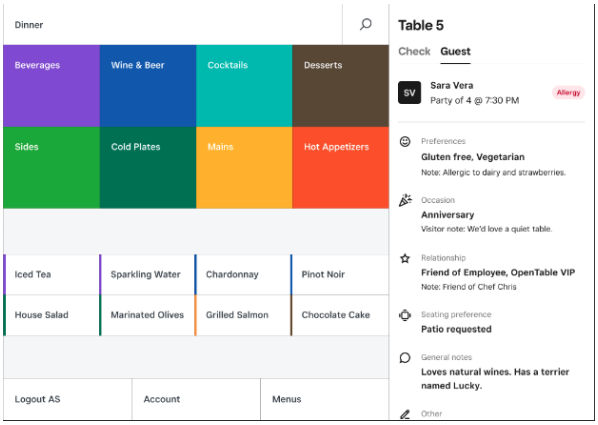

Рисунок 1.2 – Інтерфейс програмного застосунку «Square for Restaurants»

Переваги Square for Restaurants: простота використання інтерфейс Square for Restaurants інтуїтивно зрозумілий, що спрощує навчання персоналу та швидке впровадження системи, широкі можливості функціоналу Square for Restaurants пропонує розширений набір функцій, включаючи керування замовленнями, інвентарем, персоналом та аналітикою, програма може легко інтегруватися з іншими продуктами та послугами Square, такими як системи бронювання, програми лояльності та облік, Square for Restaurants доступна на різних пристроях, що дозволяє персоналу приймати замовлення та керувати рестораном з будь-якого місця, Square має високі стандарти безпеки даних, що забезпечує захист інформації про клієнтів та операцій.

Недоліки Square for Restaurants: обмежені можливості для великих підприємств деякі великі ресторани можуть знайти обмеження у функціоналі Square for Restaurants, особливо якщо потрібні спеціалізовані функції або інтеграції, вартість використання Square for Restaurants може бути високою, особливо якщо потрібні додаткові функції або платні послуги, як і в інших хмарних системах, Square for Restaurants потребує стабільного підключення до Інтернету, щоб працювати ефективно, технічна підтримка Square може бути обмеженою в порівнянні з іншими системами підтримки клієнтів, деякі ресторани можуть виявити, що можливості налаштування Square for Restaurants недостатні для їхніх унікальних потреб.

Revel Systems (рис. 1.3) - це хмарна платформа POS (точка продажу) для ресторанів, роздрібної торгівлі та обслуговування клієнтів, яка пропонує комплексні рішення для автоматизації бізнес-процесів. Розроблена компанією Revel Systems, ця система дозволяє підприємствам управляти замовленнями, оплатами, інвентарем, персоналом та іншими аспектами їхнього бізнесу [6].

Основні функції Revel Systems включають: платформа дозволяє приймати замовлення легко та ефективно, включаючи різні види замовлень: в ресторані, на винос або з доставкою, Revel Systems дозволяє ресторанам легко керувати своїм меню, включаючи додавання, видалення та редагування позицій, налаштування

цін та описів, система допомагає вести облік інвентарю, автоматично оновлюючи дані після кожного продажу та допомагаючи уникнути нестачі товарів, Revel Systems підтримує різні методи оплати, включаючи кредитні картки, мобільні платежі та інші, платформа надає розширені звіти та аналітику про продажі, інвентар, продуктивність персоналу та інші аспекти бізнесу, Revel Systems дозволяє керувати розкладами, оплатою праці та іншими аспектами персоналу, платформа працює на різних пристроях, що дозволяє персоналу приймати замовлення та керувати бізнесом з будь-якого місця.

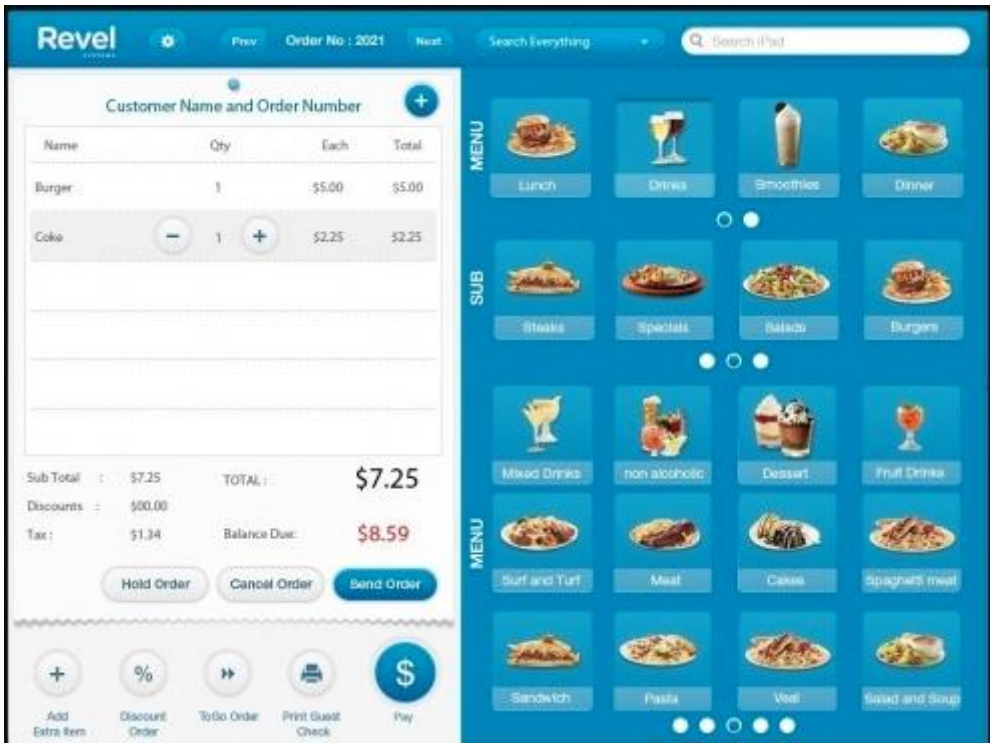

Рисунок 1.3 – Інтерфейс програмного застосунку «Revel Systems»

Переваги Revel Systems: Revel Systems пропонує розгалужений набір функцій, що дозволяє ресторанам та роздрібним підприємствам керувати замовленнями, інвентарем, персоналом та іншими аспектами бізнесу, платформа працює на різних пристроях, що дозволяє персоналу приймати замовлення та керувати бізнесом з будь-якого місця, дані зберігаються в хмарі, що дозволяє забезпечити доступ до них з будь-якого місця та зменшує ризик втрати даних, Revel Systems надає розширені звіти та аналітику, що допомагає ресторанам

аналізувати їхній бізнес та приймати обґрунтовані рішення, платформа може інтегруватися з іншими продуктами та послугами, що дозволяє підприємствам розширити функціоналість своєї системи.

Недоліки Revel Systems: вартість використання Revel Systems може бути високою, особливо для менших підприємств, що може стати перешкодою для їх використання, налаштування системи може вимагати певного часу та зусиль, особливо якщо ресторан має складну структуру меню або процеси, як і в інших хмарних системах, Revel Systems потребує стабільного підключення до інтернету для ефективної роботи, що може бути проблемою у випадку відключення мережі, деякі типи бізнесу можуть знайти обмеження у функціоналі Revel Systems, особливо якщо вони потребують спеціалізованих функцій або інтеграцій, технічна підтримка може бути не такою доступною або ефективною, як хотілося б підприємствам, що може викликати проблеми у разі виникнення проблем.

TouchBistro (рис. 1.4) - це програмне забезпечення для точок продажу (POS), розроблене спеціально для ресторанного бізнесу. Воно призначене для автоматизації різних аспектів управління рестораном, включаючи прийом замовлень, керування меню, оплату, інвентаризацію та управління персоналом [7].

Ключові функції TouchBistro включають: TouchBistro дозволяє ресторанам приймати замовлення легко та швидко, включаючи замовлення за столиками, на винос або з доставкою, ресторани можуть легко керувати своїм меню, включаючи додавання, видалення та редагування позицій, налаштування цін та описів, TouchBistro допомагає вести облік інвентарю, автоматично оновлюючи дані після кожного продажу та допомагаючи уникнути нестачі товарів, програма підтримує різні методи оплати, включаючи кредитні картки, мобільні платежі та інші, TouchBistro надає розширені звіти та аналітику про продажі, інвентар, продуктивність персоналу та інші аспекти бізнесу, програма дозволяє керувати розкладами, оплатою праці та іншими аспектами персоналу. TouchBistro також може інтегруватися з іншими продуктами та послугами, що дозволяє ресторанам

розширити функціоналість своєї системи і працювати більш ефективно. Це робить TouchBistro популярним вибором для ресторанів різних розмірів і типів.

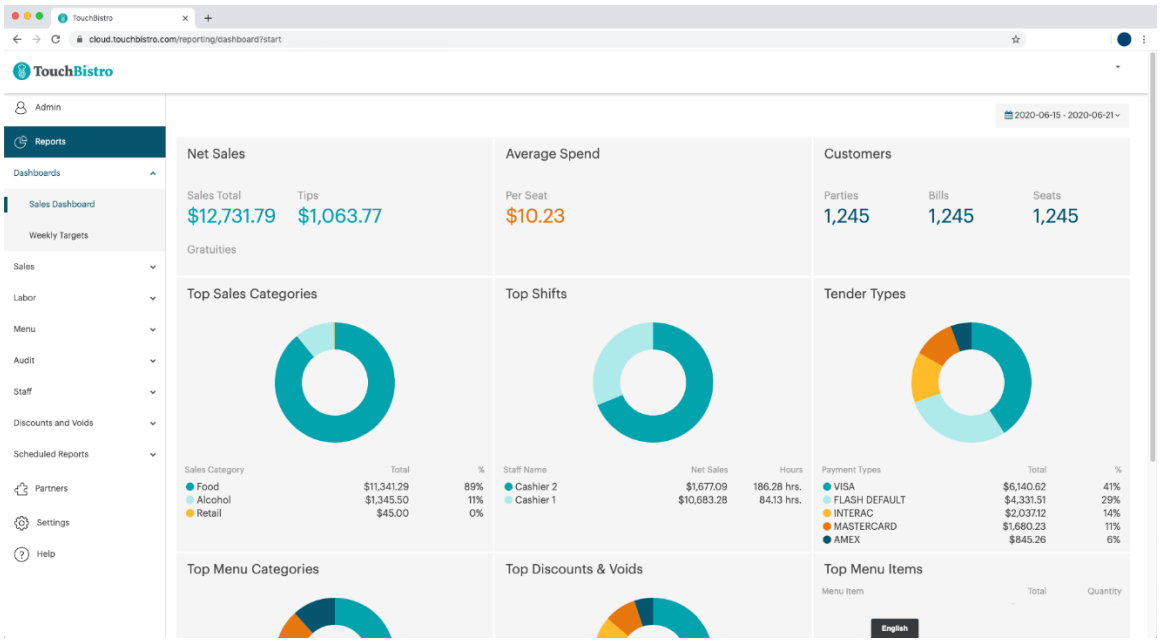

Рисунок 1.4 – Інтерфейс програмного застосунку «TouchBistro»

Переваги TouchBistro: TouchBistro спеціалізується на потребах ресторанного бізнесу, що робить його ідеальним вибором для ресторанів будь-якого типу і розміру, програма має простий та зрозумілий інтерфейс, що спрощує навчання персоналу та використання системи в щоденній роботі, TouchBistro працює на iPad, що дозволяє персоналу приймати замовлення та керувати рестораном з будь-якого місця у закладі, програма пропонує розгалужений набір функцій, включаючи керування замовленнями, інвентарем, персоналом, аналітику та інші аспекти ресторанного бізнесу, TouchBistro дозволяє збирати та аналізувати дані про клієнтів, що дозволяє ресторанам створювати персоналізовані пропозиції та програми лояльності.

Недоліки TouchBistro: вартість використання TouchBistro може бути високою, особливо для менших ресторанів або стартапів, для ефективної роботи програма потребує стабільного підключення до Інтернету, що може бути проблемою в разі відключення мережі, налаштування системи може вимагати певного часу та зусиль, особливо якщо ресторан має складну структуру меню або

процеси, для деяких великих ресторанів або ланцюжків ресторанів можуть виявитися обмеження у функціоналі або масштабованості TouchBistro, підтримка може бути не завжди такою доступною або ефективною, як хотілося б користувачам.

Результати порівняння аналогів наведено в табл. 1.1.

Таблиця 1.1 – Функціональні характеристики програмних додатків

	Власна реалізація	Toast POS	Square for Restaurants	Revel Systems	TouchBistro
Безпека даних	1	0	1	1	0
Зручний інтерфейс	1	1	1	1	1
Простота налаштування	1	0	1	0	0
Простота навчання персоналу	1	1	0	1	1
Загальна оцінка	100%	50%	75%	75%	50%

Отже, відповідно до порівняльної таблиці, розробка інформаційної системи для керування закладами громадського харчування є доцільним.

1.3 Аналіз методів розв’язання задачі

Розвиток інформаційних технологій суттєво вплинув на функціонування закладів громадського харчування, зокрема зробив можливим автоматизацію багатьох процесів управління [8]. Розробка програмних засобів інформаційної системи для керування такими закладами є актуальною задачею, яка має вирішити ряд проблем, пов'язаних з ефективним веденням бізнесу і задоволенням потреб клієнтів.

Розробка програмних засобів інформаційної системи для керування закладами громадського харчування є складною задачею, яка вимагає ретельного аналізу потреб бізнесу та вибору найбільш підходящих технологій. Врахування переваг і недоліків різних методів розв’язання є ключовим для успішної імплементації інформаційної системи, що задовольнятиме потреби як управлінців, так і клієнтів закладу.

Розробка веб-додатків для керування закладами громадського харчування базується на створенні програмного забезпечення, яке доступне через веб-браузер. Цей метод передбачає розміщення програми на веб-сервері та доступ до неї через мережу Інтернет [9]. Веб-додатки можуть включати в себе різноманітні функції, такі як управління замовленнями, реєстрація клієнтів, управління меню та інвентарем, аналітика продажів тощо.

Переваги: Веб-додатки можуть бути доступні з будь-якого пристрою, що має браузер і підключення до Інтернету, це означає, що користувачі можуть управляти закладом громадського харчування зі свого комп'ютера, планшета або смартфона, відсутність обмежень щодо платформи робить цей метод дуже зручним для клієнтів і персоналу закладу, легка масштабованість веб-додатки можуть бути легко масштабовані для відповіді на зростаючий обсяг користувачів або розширення функціональності, зміни в програмному забезпеченні зазвичай вносяться централізовано на сервері, це означає, що вам не потрібно розгортати нову версію додатка на кожному пристрої, а просто оновлювати сервер, веб-додатки дають можливість реалізації онлайн-сервісів, таких як замовлення їжі через інтернет, бронювання столиків або перегляд меню, це може значно полегшити взаємодію з клієнтами і підвищити задоволення від обслуговування, веб-додатки можуть легко інтегруватися з іншими сервісами, такими як платіжні системи, системи управління відносинами з клієнтами (CRM), системи управління відносинами з постачальниками (SRM) тощо, це дозволяє підвищити ефективність роботи закладу.

Веб-додатки є потужним інструментом для розробки програмних засобів інформаційної системи для керування закладами громадського харчування. Їх

гнучкість, легка масштабованість та можливість реалізації різноманітних онлайн-сервісів роблять їх привабливим варіантом для бізнесу у цій сфері. Однак важливо бути готовими до вирішення проблем, таких як залежність від Інтернету, безпека та вимоги до дизайну інтерфейсу, для забезпечення успішної експлуатації таких систем.

1.4 Аналіз проблематики розвитку нейронної мережі

Розробка нейронні мережі для інформаційної системи для керування закладами громадського харчування, стикається з низкою специфічних проблем та викликів.

Нейронні мережі потребують великої кількості якісних даних для навчання. У закладах громадського харчування збір даних може бути нерівномірним або неповним, що ускладнює навчання моделей. Не кожен власник захоче оприлюднювати дані власного закладу з різних причин, що не дає можливості збирати велику кількість даних, також деякі власники можуть змінювати дані для своїх цілей, що призведе до спотворення даних і вони не будуть відповідати дійсності.

Дані клієнтів та операцій повинні бути захищені відповідно до регуляторних вимог, що додає додатковий рівень складності при зборі та обробці даних. Оскільки отримані дані будуть оброблятися нейронною мережею яка буде доступна кожному користувачу інформаційної системи то не на пряму у кожного користувача буде доступ до даних які вона обробила, користувачі будуть користуватися досвідом який отримала нейрона мережа на даних інших користувачів.

Заклади громадського харчування працюють у динамічному середовищі, де попит може швидко змінюватися залежно від часу доби, дня тижня, сезону чи зовнішніх подій. Нейронна мережа повинна вміти адаптуватися до цих змін. Також потреба у регулярному оновленні моделі для врахування нових даних та умов, що може потребувати повторного навчання і адаптації системи.

Нейронна мережа може мати більше досвіду в деяких напрямках більше ніж в інших, тому для деяких користувачів вона може бути менш корисною через брак даних у певній специфічній галузі конкретного закладу. Такі випадки можуть виникати, коли певні заклади не бажають надавати власні дані для подальшого розвитку нейронної мережі. Це може викликати певні конфлікти та обмеження у розвитку системи. Для пояснення розглянемо таку теоретичну ситуацію: якщо існують два або більше закладів схожої спеціалізації і вони є конкурентами у даній сфері, то у випадку, якщо вони користуються системою з нейронною мережею, яка може давати поради на основі зібраних даних, може виникнути ситуація, коли жоден з закладів не захоче надавати власні дані для розвитку нейронної мережі. Це відбувається через побоювання, що їхні конкуренти отримають перевагу від використання покращеної мережі, яка буде давати кращі результати на основі наданих даних. Це може призвести до зупинки розвитку нейронної мережі у цьому напрямку.

Така ситуація є класичним прикладом з теорії ігор. Уявімо, що є дві сторони: якщо одна сторона надасть дані, а інша ні, то в виграші буде сторона, яка не надала дані, оскільки вона скористається результатами роботи нейронної мережі без внеску у її розвиток. Якщо обидві сторони нададуть дані, то обидві сторони будуть у виграші, оскільки нейронна мережа стане більш ефективною і корисною для обох. А коли обидві сторони не нададуть дані, то обидві сторони програють, оскільки нейронна мережа не розвиватиметься через брак даних для аналізу.

Таким чином, для стабільного розвитку нейронної мережі необхідно спонукати заклади до надання даних для аналізу. Важливо пояснювати їм, що найоптимальнішою стратегією є співпраця та надання власних даних, що сприятиме розвитку нейронної мережі. Це допоможе забезпечити ефективність системи і її користь для всіх учасників. Механізми мотивації можуть включати певні стимули для закладів, які надають свої дані, а також розробку політики конфіденційності, яка забезпечить безпеку та анонімність переданих даних, щоб мінімізувати ризики конкурентних втрат. Заклади повинні розуміти, що

співпраця у зборі даних для нейронної мережі призведе до кращих результатів для всіх учасників і сприятиме загальному прогресу та інноваціям у сфері громадського харчування.

1.5 Постановка задач бакалаврської дипломної роботи

Розглядаючи аналіз функціональності застосунків для управління доставкою вантажів, було встановлено перелік завдань, які потрібно виконати для створення системи, а саме:

1) створення структури бази даних для зберігання інформації про клієнтів, запаси інгредієнтів, меню та інші важливі дані;

2) розробка інтерфейсу для прийому та обробки замовлень, керуванням меню, відслідковування запасів тощо;

3) створення засобів для аналізу ефективності робти, включаючи звіти про замвлення;

4) створення зручного та інтуїтивно зрозумілого інтерфейсу системи для замовлення, перегляду та зміни меню, звітності.

1.6 Висновки

У першому розділі бакалаврської дипломної роботи було проведено детальний огляд існуючих програмних продуктів, що використовуються для управління закладами громадського харчування, таких як «Toast POS», «Square for Restaurants», «Revel Systems» та «TouchBistro». Аналіз цих систем включав розгляд їх функціональних можливостей, архітектури, зручності користування та особливостей інтеграції з іншими системами.

Порівняння цих систем дозволило визначити їх переваги та недоліки у порівнянні з пропонованим рішенням, що вказало на доцільність створення власного програмного продукту. Аналіз показав, що існуючі системи мають свої

сильні та слабкі сторони, але жодна з них не задовольняє всіх вимог, які можуть виникати у різних закладів громадського харчування.

Це стало підставою для розробки унікального програмного забезпечення, яке враховуватиме специфічні потреби цільової аудиторії та забезпечуватиме більшу гнучкість та адаптивність. На основі цього аналізу був сформований перелік завдань, необхідних для успішної розробки власних програмних засобів інформаційної системи для керування закладами громадського харчування.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТ

2.1 Аналіз вхідних даних системи

Для реалізації програмного продукту для керування закладами громадського харчування буде використана мова програмування Java, фреймворк Spring та об'єктно-реляційна система керування базами даних PostgreSQL .

Java – це об’єктно-орієнтована мова програмування загального призначення з простим і зрозумілим синтаксисом, яка підходить для різних платформ. Завдяки широким можливостям, бібліотекам і портативності Java дозволяє створювати ПЗ для різних компаній і сфер: ігри, мобільні застосунки, корпоративні рішення тощо [10].

Spring Framework – це платформа з відкритим вихідним кодом для створення веб-додатків з Java як мовою програмування. Він потужний і легкий, але простий у використанні, і забезпечує підтримку легкої розробки додатків Java [11].

PostgreSQL це система управління реляційними базами даних з відкритим вихідним кодом, що використовує модель клієнт-сервер. Він використовує мову SQL для управління даними і запитів до них і може бути доступна через різноманітні протоколи, включаючи TCP / IP і доменні сокети Unix [12].

Розробка програмних засобів інформаційної системи для керування закладами громадського харчування.

Першим етапом є проектування системи. Розробка архітектури системи - це включає в себе визначення структури системи, вибір архітектурного стилю (наприклад, клієнт-сервер, багатошарова архітектура тощо), інтеграцію різних компонентів, які будуть взаємодіяти між собою. Вибір технологій - це включає в себе вибір мов програмування, бази даних, фреймворків та інших інструментів, які найбільше підходять для потреб розробки системи. На цьому етапі

розробляються схеми бази даних, визначається спосіб взаємодії з користувачем через інтерфейси, такі як веб-інтерфейс.

Другим етапом є розробка. Перехід до фази створення програмного коду на основі вже розробленої архітектури та вимог. Кожен компонент системи розробляється відповідно до специфікацій і вимог до функціональності. Додавання функціоналу системи, що включає в себе створення модулів обробки замовлень, управління персоналом, управління складом тощо.

Третім етапом є тестування. Проводяться тести для перевірки того, чи працює кожна функція системи вірно та відповідно до специфікацій. Система піддається тестам на відмовостійкість, тобто перевіряється, як вона поводиться в умовах відмови або навантаження. Проводяться тести на виявлення потенційних вразливостей системи та забезпечення заходів безпеки. Якщо виявляються помилки, розробники виправляють їх і повторно проводять тести для підтвердження коректності виправлень.

Розробка програмних засобів інформаційної системи для керування закладами громадського харчування є складним і багатоетапним процесом, що вимагає детального аналізу потреб бізнесу, дотримання вимог якості та відповідності стандартам розробки програмного забезпечення. Ефективне виконання кожного з етапів є ключовим для успішної імплементації системи та задоволення потреб користувачів.

2.2 Розробка моделей інтерфейсу програмного додатку

Моделі відіграють важливу роль у розробці інтерфейсу, а саме для продуманої взаємодії людини з технологією, оскільки вони визначають спосіб цієї взаємодії також при розробці додатку оскільки потрібно ретельно продумати усі частини інтерфейсу для зручного та ефективного користування додатком користувачем.. Фізіологічно, інтерфейси повинні бути спроектовані з урахуванням можливостей та обмежень людського організму, щоб забезпечити максимальний рівень комфорту та ефективності взаємодії [13].

Для користувача важливо мати зручний доступ до інформації та можливість взаємодії з системою без надмірних зусиль. Інтерфейси повинні бути легкими у використанні, інтуїтивно зрозумілими та ергономічними, щоб не перевантажувати користувача та уникнути помилок.

Правильно спроектований інтерфейс може покращити продуктивність користувача, знизити його втомлюваність та збільшити задоволення від використання системи. Такий підхід дозволяє створювати ефективні та приємні у користуванні програмні продукти.

Сторінка реєстрації користувача (рис. 2.1), дає змогу користувачу занести свої дані в базу системи для авторизації і в подальшому користуватися функціями системи. До полів для введення даних передбачається такі поля, як ім’я користувача, адреса електронної пошти (Email), пароль та повторення паролю. Ці поля є ключовими для проведення реєстрації користувача до бази даних і дання можливості користуватися усіма функціями інформаційної системи. Ім’я користувача та пароль буде використовуватися в подальшому при авторизації користувача, під час реєстрації та авторизації вміст поля пароль буде прихованим для безпечності, тому є потреба в існуванні додаткового поля для паролю для запобігання введення користувачем не вірного паролю.

Також, передбачається посилання на сторінку входу в разі якщо користувач вже зареєстрований в системі.

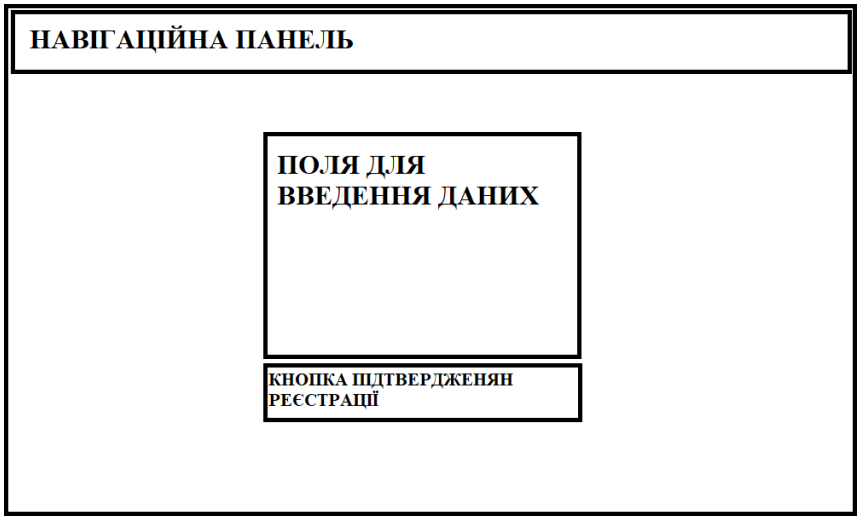

Рисунок 2.1 – Сторінка реєстрації користувача

Після реєстрації в системі користувач бачить сторінку входу після авторизації користувач може почати кристуватися функціями системи (рис. 2.2). Для авторизації потрібно ввести ім’я користувача і пароль далі система перевіряє чи є в базі даних такий користувач і або авторизує користувача, або відхиляє запит на авторизацію.

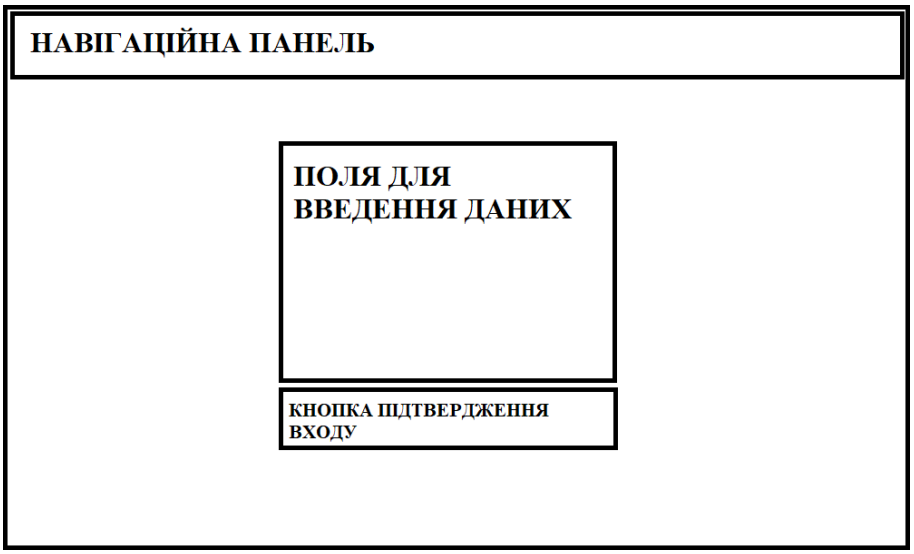

Рисунок 2.2 – Сторінка входу користувача

Основною сторінкою для користувача буде перегляд меню (рис. 2.3). На даній сторінці користувач зможе ознайомитися з доступним переліком страв, побачити їх ціну та прочитати короткий опис кожної страви. Ця сторінка стане ключовою для взаємодії користувача з додатком, оскільки вона надає всю необхідну інформацію для прийняття рішення про замовлення. Після авторизації користувач отримає можливість робити замовлення. Авторизація дозволить зберігати персональні дані користувача, що полегшить процес замовлення при наступних відвідинах додатка. Користувач зможе додавати страви до кошика, редагувати замовлення, вказувати кількість порцій та залишати коментарі або побажання до замовлення. Таким чином, основна сторінка перегляду меню є центральним елементом додатка, що забезпечує користувачу зручний доступ до інформації про страви та спрощує процес замовлення.

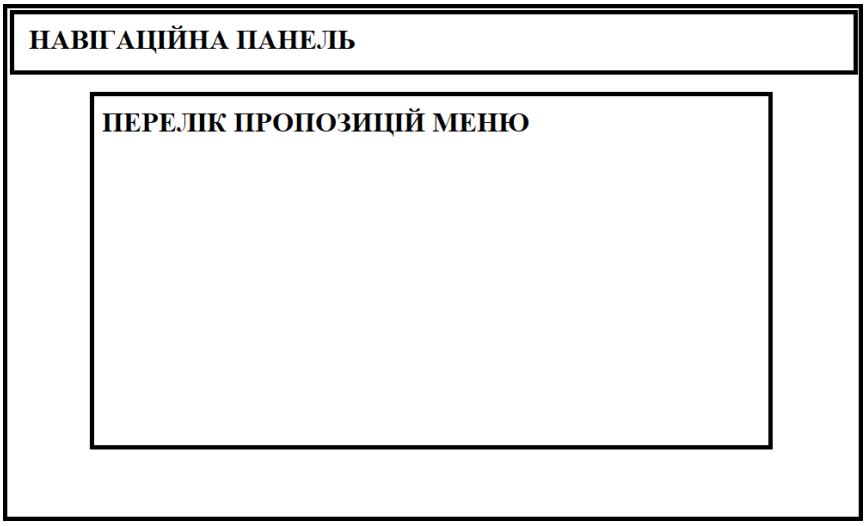

Рисунок 2.3 – Сторінка меню

Для менеджерів та власників закладів основною сторінкою стане сторінка формування звіту (рис. 2.4). На даній сторінці вони матимуть можливість переглядати детальну статистику продажів для кожного товару в реальному часі. Ця сторінка надасть вичерпну інформацію, яка допоможе в аналізі ефективності продажів, плануванні закупівель та розробці стратегій для покращення роботи закладу. Сторінка формування звіту буде мати різні розділи та фільтри, що дозволять менеджерам та власникам легко знаходити потрібну інформацію. Вони зможуть переглядати дані за різні періоди часу: день, тиждень, місяць або рік. Також буде можливість обрати конкретний товар або категорію товарів для детального аналізу. Інформація буде представлена у вигляді таблиць та графіків, що візуально спростить сприйняття даних та полегшить прийняття рішень.

Загалом, сторінка формування звіту є важливим інструментом для ефективного управління закладом, оскільки вона забезпечує доступ до актуальної та детальної інформації про продажі, допомагає відслідковувати результати роботи та планувати подальші дії для покращення бізнес-процесів.

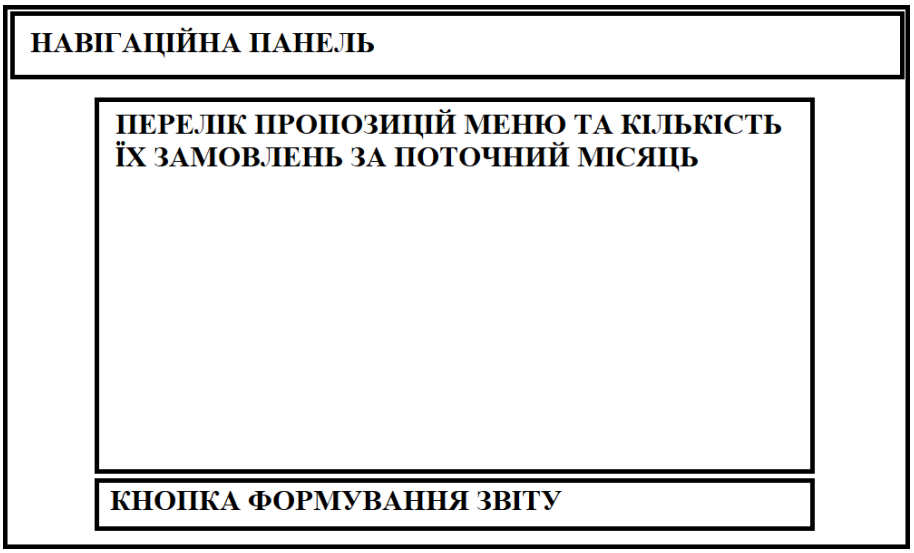

Рисунок 2.4 – Сторінка формування звіту

Таким чином створивши моделі для найважливіших сторінок інтерфейсу системи, можна створювати реалізацію цих сторінок з врахуванням цих моделей, що покращить їх якість та ефективність користування завдяки планування розташування елементів до початку реалізації.

2.3 Розробка архітектури роботи серверної частини

Розробка серверної частини програми є вкрай корисною, оскільки вона дозволяє ефективно постачати інформацію, придатну для конкретних користувачів, що в результаті сприяє покращенню користувацького досвіду [14]. Завдяки добре спроектованій серверній частині, програма може обробляти великі обсяги даних, швидко реагувати на запити користувачів та забезпечувати безперервний доступ до інформації. Це особливо важливо для систем, які обслуговують велику кількість користувачів одночасно. Стандартна модель роботи серверної частини системи продемонстрована на рисунку 2.5. Вона складається з декількох основних компонентів, кожен з яких виконує свою специфічну функцію

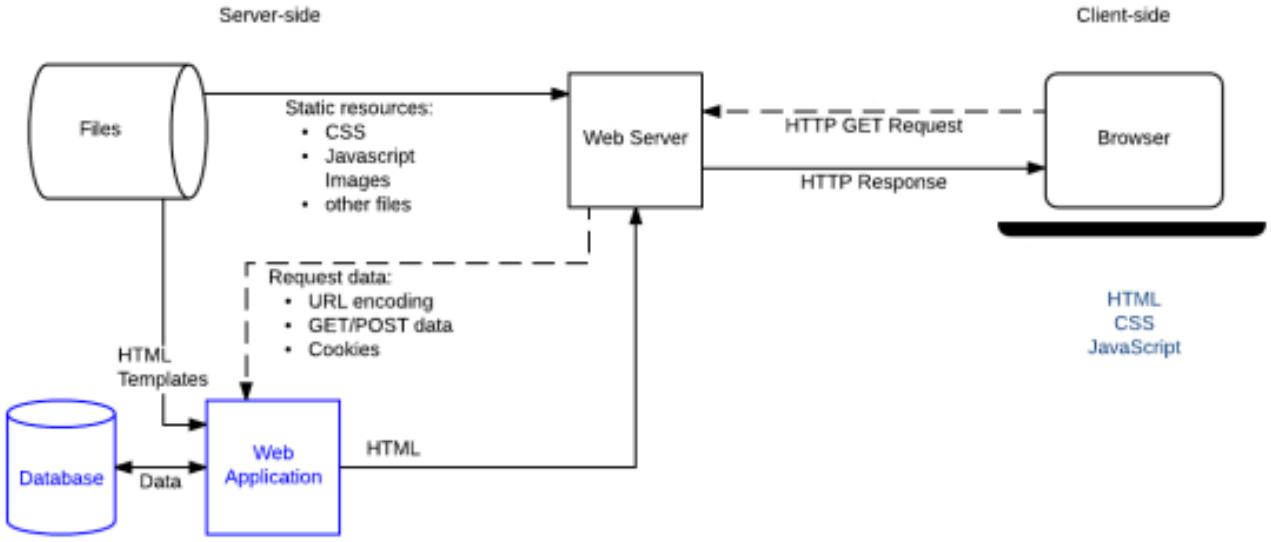

Рисунок 2.5 – Архітектура серверної частини системи

Створення оптимізованої та гнучкої архітектури сервера сприяє покращенню користувацького досвіду та забезпечує більшу адаптивність до потреб різних користувачів. Це досягається завдяки швидкому реагуванню системи на запити, високій надійності та можливості легко масштабувати ресурси у випадку збільшення навантаження. Враховуючи це, розробка архітектури роботи серверної частини є ключовим етапом у створенні успішних та конкурентоздатних програмних продуктів, оскільки саме вона визначає стабільність, продуктивність та безпеку системи.

Для власної розробки було прийнято рішення модифікувати стандартну архітектуру серверної частини системи, додавши модуль з нейронною мережею що продемонстровано на рис. 2.6. Цей модуль буде інтегрований в серверну інфраструктуру та працюватиме, аналізуючи дані та надаючи рекомендації. Цей підхід забезпечить кілька ключових переваг.

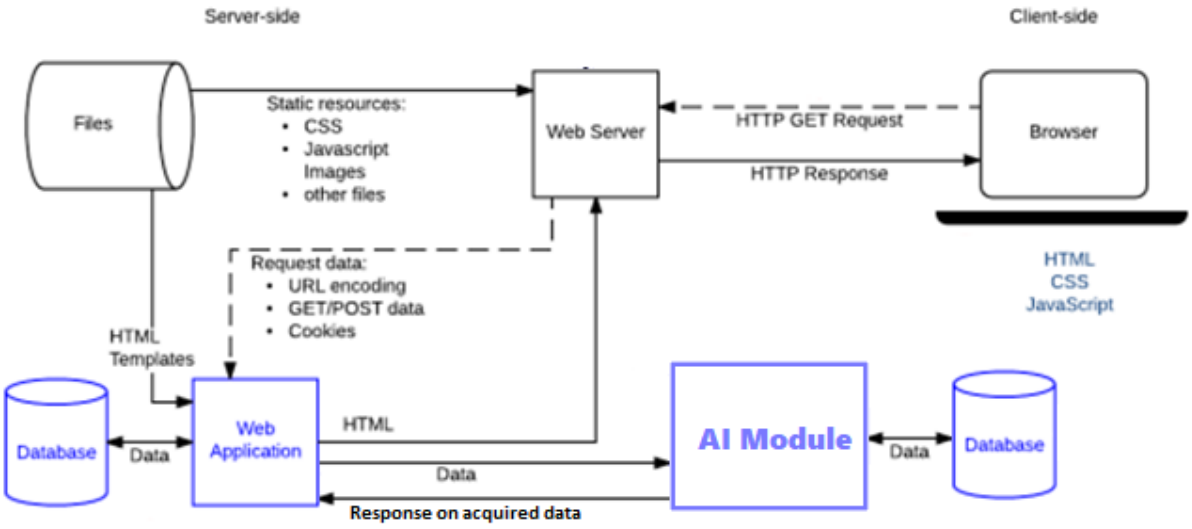

Рисунок 2.6 – Модифікована архітектура серверної частини системи

По-перше, користувачі системи зможуть отримувати більш точні та релевантні рекомендації, що сприятиме підвищенню їхнього задоволення від використання продукту. Наприклад, система може пропонувати страви, які найбільше відповідають смакам користувачів або підказувати оптимальні часи для замовлення, щоб уникнути пікових навантажень.

По-друге, аналіз даних, проведений нейронною мережею, допоможе виявляти тенденції та проблеми, які можуть бути непомітними при звичайному аналізі. Це дозволить менеджерам та власникам закладів приймати більш обґрунтовані рішення щодо оптимізації роботи та підвищення ефективності. Наприклад, аналіз може показати, які страви користуються найбільшою популярністю у певний час доби або в які дні тижня спостерігається найбільший потік клієнтів. Крім того, інтеграція нейронної мережі в серверну частину системи відкриває можливості для подальшого розвитку та вдосконалення продукту. Можна додавати нові функції та можливості без необхідності кардинальних змін в основній архітектурі, що забезпечить гнучкість та адаптивність системи до нових вимог і викликів. Таким чином, модифікація стандартної архітектури серверної частини шляхом додавання модуля з нейронною мережею є важливим кроком, що дозволяє не тільки підвищити

ефективність роботи системи, але й забезпечити високий рівень задоволення користувачів та конкурентоздатність продукту на ринку.

Також така архітектура дозволить декілька варіантів конфігурації системи для користувачів да декількох варіантів доступу до нейронної мережі, а саме власники закладів які бажають впровадити розроблену систему у власний заклад будуть мати такі варіанти.

По-перше, при встановлені інформаційної системи також можна додати модуль з нейронною мережею яка буду знаходитись разом з основою системою на одному сервері або в одній локальній мережі, тоді модератори інформаційної системи будуть мати повний доступ до нейронної мережі, це буде базова нейронна мережа вона буде мати невеликий фундамент який буде навчений на відкритих джерелах інформації та в подальшому буде навчатись саме на даних які буде надавати даний заклад, дані які аналізує нейрона мережа за вибором власника можуть бути надані для навчання основної нейронної мережі.

По-друге, не встановлювати модуль нейронної мережі при встановлені інформаційної системи тоді буде варіант користуватися основною нейронною мережею яка знаходиться на сервері не власника закладу, але для повного доступу до усіх можливостей нейронної мережі буде потрібно надавати власні дані які були зібрані під час роботи закладу та інформаційної системи, ця нейрона мережа навчається на усіх даних що надають користувачі даної системи тому є найпотужнішою з усіх. В разі якщо власник не зацікавлений в надані власних даних є можливість користуватися обмеженою нейронною мережею яка є такою як і та що встановлюється як базова при обрані першого варіанту, проте вона знаходиться не на сервері власника, в нього не буде доступу до неї і вона не буде навчатися на наданих ним даних.

По-третє, варіант коли власник взагалі не зацікавлений в функціях нейронної мережі, тоді інформаційна система не буде мати ні власної нейронної мережі ні доступу до інших. Інформаційна система буде функціонувати як звичайна і не буде надсилати нікуди ніякі дані які зберігає.

2.4 Розробка алгоритмів роботи системи

Для успішної розробки програмних засобів інформаційної системи, спрямованих на керування закладами громадського харчування, необхідно створити вдосконалений алгоритм, який охоплює процеси авторизації та формування звітності. Цей алгоритм є основою функціональності програмного забезпечення, забезпечуючи його коректну роботу та відповідність вимогам. Згідно з визначеним алгоритмом, зображеним на рисунку 2.7, користувач повинен пройти процедуру авторизації для доступу до функцій системи, а саме перегляд меню. Це включає в себе введення облікових даних та підтвердження їх правильності для отримання доступу до функціоналу програми. Таким чином, алгоритм авторизації є важливою складовою частиною інформаційної системи, яка забезпечує безпеку та ефективність її роботи.

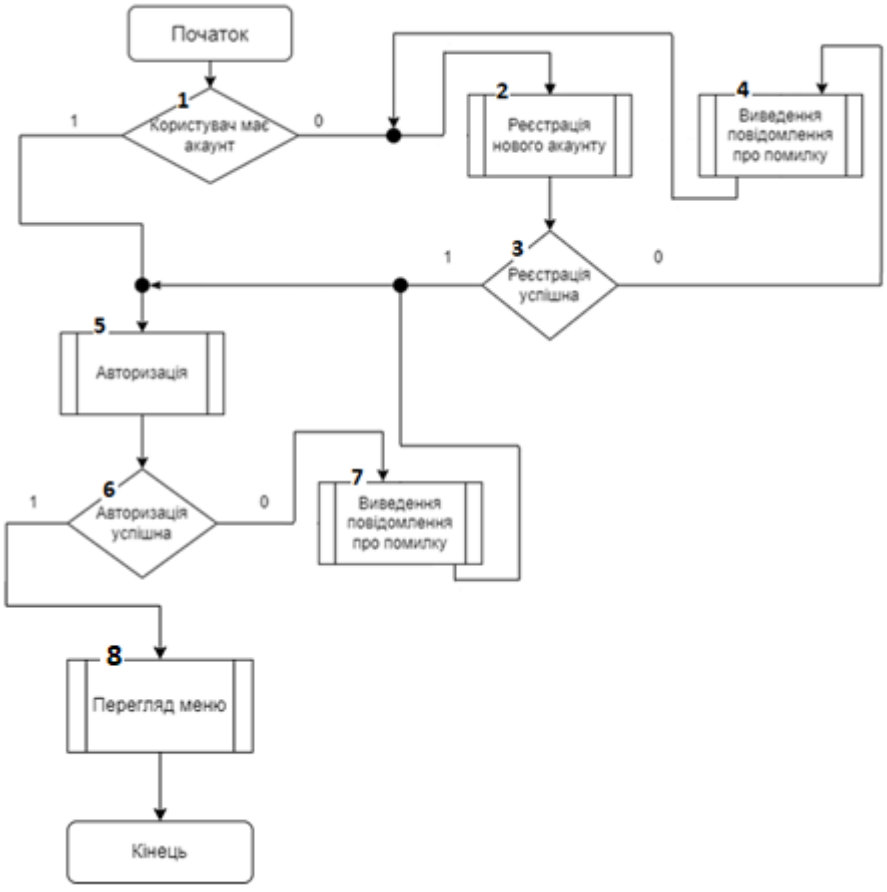

Рисунок 2.7 – Авторизація та перегляд меню

Якщо виникла помилка під час авторизації чи реєстрації, система негайно повідомляє користувача про необхідність пройти процес авторизації чи реєстрації ще раз. Це дозволяє швидко виявити та виправити можливі помилки, такі як неправильне введення даних або технічні збоїв.

Після успішної реєстрації користувач отримує підтвердження про створення облікового запису та може переходити до авторизації. Після успішної авторизації система перевіряє, чи є користувач менеджером або клієнтом. Це здійснюється шляхом порівняння введених облікових даних із записами в базі даних. Якщо аккаунт належить звичайному клієнту, користувач не бачить відповідних частин інтерфейсу для менеджерів і не може користуватися їх функціоналом. Це гарантує, що доступ до конфіденційної інформації та управлінських функцій мають лише авторизовані менеджери.

Щоб переглянути меню, користувачу потрібно лише натиснути відповідну частину інтерфейсу у навігаційній панелі, після чого відобразиться сторінка з меню. Меню організоване так, щоб користувач міг легко знайти потрібні страви або напої, переглянути їх опис та ціни. Це робить процес вибору та замовлення максимально простим та зручним.

Згідно з визначеним алгоритмом, зображеним на рисунку 2.8, користувач повинен пройти процедуру авторизації для доступу до функцій системи, зокрема формування звіту. Це включає в себе введення облікових даних та підтвердження їх правильності для отримання доступу до функціоналу програми. Лише користувачі, які мають обліковий запис менеджера, можуть скористатися функцією формування звіту. Це забезпечує, що тільки уповноважені особи можуть переглядати та аналізувати статистику продажів, звіти про діяльність закладу та інші важливі дані.

Таким чином, алгоритм авторизації та аутентифікації є важливою складовою частиною інформаційної системи, яка забезпечує безпеку та ефективність її роботи. Завдяки цьому алгоритму система захищає конфіденційну інформацію від несанкціонованого доступу та забезпечує належне управління доступом до різних функцій програми. Це сприяє

підвищенню загальної надійності та безпеки системи, а також покращує користувацький досвід, забезпечуючи легкий та безперешкодний доступ до необхідних функцій для авторизованих користувачів.

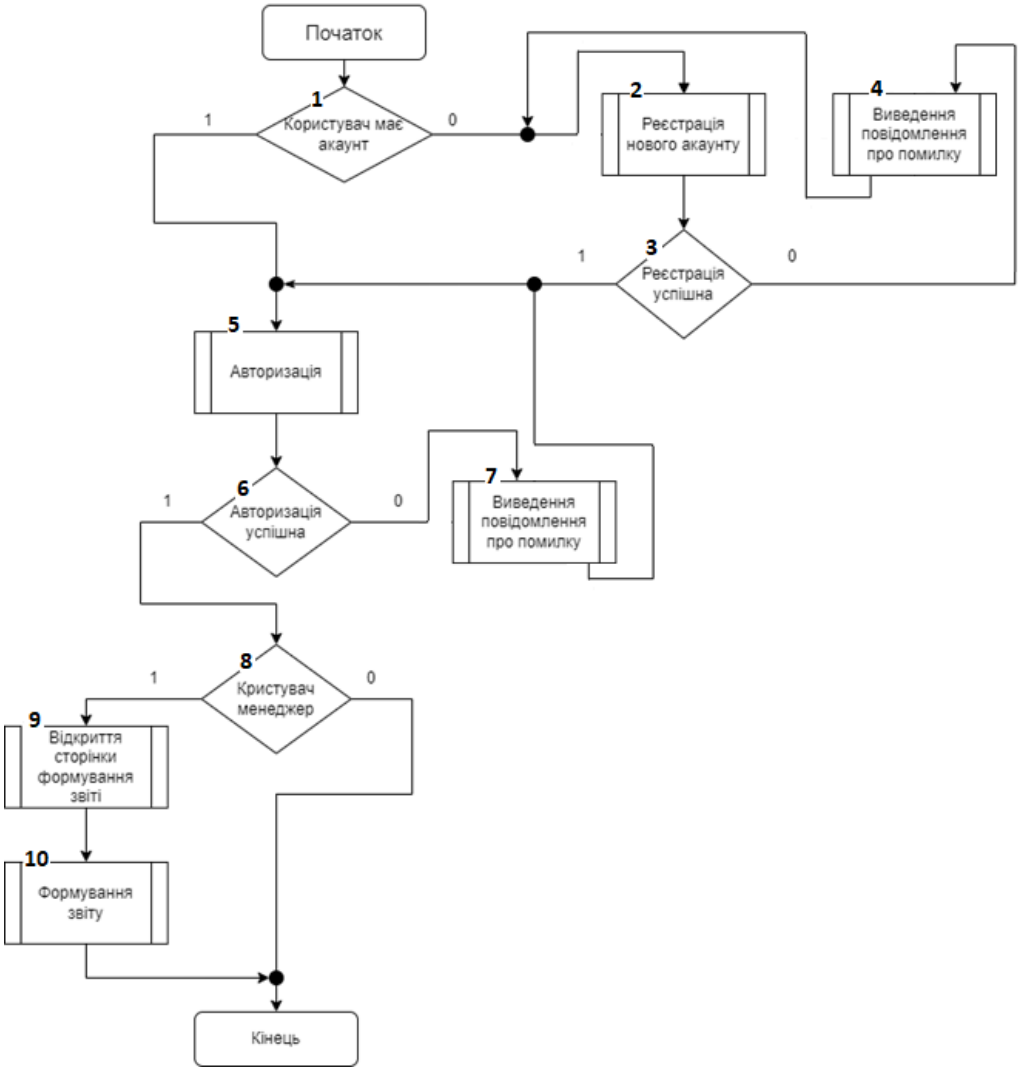

Рисунок 2.8 – Сторінка формування звіту

Розглянувши дані алгоритми, можна зрозуміти, що розроблений застосунок дає змогу клієнтам відслідковувати актуальне меню закладу, а менеджерам створювати звіти для легкого формування звіту. Алгоритм дозволяє повністю автоматизувати систему, для перегляду та оновлення меню до його актуальної версії та створення звіту.

2.5 Висновки

У другому розділі бакалаврської дипломної роботи було виконано проектування програмного забезпечення, що включало в себе визначення структурних компонентів системи, їх взаємозв'язків та принципів взаємодії. Також були розроблені моделі, які відображають основні концепції та принципи функціонування системи. Окрім цього, створені структури програмних модулів та алгоритми, що визначали логіку роботи системи та методи вирішення поставлених завдань. Додатково був реалізований інтерфейс користувача з метою забезпечення зручної та ефективної взаємодії з мобільним додатком.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ПРОГРАМНИХ ЗАСОБІВ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Для розробки було використано такі технології, як мова прграмування Java, Spring Framework та деякі його модулі, а саме Spring Security, Spring Web, Lombok, PostgreSQL Driver також був використаний Maven та база даних PostgreSQL.

Java - це високорівнева, об'єктно-орієнтована мова програмування, яка відома своєю переносимістю та простотою використання. Вона була розроблена компанією Sun Microsystems у 1995 році і з тих пір стала однією з найпопулярніших мов програмування у світі. Однією з головних особливостей Java є її платформа незалежність, що означає, що програми, написані на Java, можуть виконуватися на будь-якій платформі, яка підтримує Java Virtual Machine (JVM). Це досягається завдяки тому, що компіляція Java-коду відбувається у байт-код, який може бути виконаний на будь-якому пристрої з JVM. Java має велику кількість бібліотек та фреймворків, що значно полегшує процес розробки. Бібліотеки, такі як Java Standard Library, надають безліч корисних функцій для роботи з текстом, мережею, файловою системою, графічним інтерфейсом користувача та багато іншого. Популярні фреймворки, такі як Spring, Hibernate, Struts, JavaServer Faces (JSF) та Apache Maven, дозволяють розробникам швидко створювати складні програми, забезпечуючи інструменти для управління залежностями, конфігурації, тестування та розгортання додатків. Крім того, Java відома своєю простою синтаксичною структурою, що робить її доступною для новачків у програмуванні, а також потужною для досвідчених розробників. Вона підтримує об'єктно-орієнтований підхід до програмування, що дозволяє створювати модульні та багаторазові компоненти коду. Java також забезпечує

високу безпеку, що особливо важливо для розробки інтернет-додатків та сервісів. Таким чином, Java є потужною, гнучкою та надійною мовою програмування, яка підходить для розробки різноманітних додатків, від невеликих мобільних додатків до великих корпоративних систем. Її переносимість, велика кількість бібліотек та фреймворків, проста синтаксична структура та активна спільнота роблять її одним з найпопулярніших інструментів у світі програмування.

PostgreSQL - це потужна та надійна система керування базами даних, яка підтримує широкий спектр функціональності і є однією з найпопулярніших і найнадійніших відкритих баз даних у світі. Вона була створена з метою забезпечення високої продуктивності, надійності та можливості масштабування для великих обсягів даних та складних запитів. PostgreSQL має безліч вбудованих можливостей, які роблять її ідеальним вибором для різних типів додатків, від невеликих проектів до великих корпоративних систем. Однією з ключових особливостей PostgreSQL є її вбудована підтримка JSON, що дозволяє зберігати та опрацьовувати дані у форматі JSON. Вона забезпечує повну підтримку ACID-транзакцій (атомарність, консистентність, ізольованість, довговічність), що гарантує надійність та цілісність даних навіть у випадку збоїв системи. Це робить PostgreSQL ідеальним вибором для додатків, де критично важливо забезпечити безпеку та консистентність даних, таких як фінансові системи, системи електронної комерції та інші. Окрім цього, PostgreSQL пропонує відмінну підтримку мови SQL, включаючи розширені можливості, такі як CTE (Common Table Expressions), віконні функції, підзапити, тригери, перегляди та багато інших. Це дозволяє розробникам писати складні та оптимізовані запити для отримання необхідної інформації з бази даних. Крім того, PostgreSQL підтримує широкий спектр типів даних, включаючи числові, строкові, часові, геометричні та інші спеціалізовані типи, що дозволяє зберігати різноманітні види даних ефективно і безпечно. Завдяки своїм потужним функціям, високій продуктивності, гнучкості та надійності, PostgreSQL є

відмінним вибором для сучасних розробників та підприємств, які шукають стабільну та масштабовану систему керування базами даних.

Spring Framework - це один з найпопулярніших фреймворків для розробки додатків на мові Java. Він надає ряд модулів, таких як Spring Core, Spring MVC, Spring Data, Spring Security тощо, які спрощують розробку різних аспектів програми. Наприклад, використання Spring MVC дозволяє легко створювати веб-додатки з розділенням логіки та представлення. Spring Framework заснований на принципах інверсії керування та внедрення залежностей, що дозволяє зменшити залежність між компонентами додатка та спрощує тестування. Крім того, Spring надає велику кількість готових модулів та рішень для різних аспектів розробки, таких як веб-розробка, безпека, робота з базами даних тощо.

Spring Security - це потужний інструмент для забезпечення безпеки ваших додатків [15]. Він надає можливості для налаштування різних аспектів безпеки, таких як аутентифікація користувачів, авторизація доступу до ресурсів, захист від CSRF-атак, контроль сесій та багато іншого. Використовуючи Spring Security, розробники можуть створювати надійні системи, які відповідають високим стандартам безпеки. Цей фреймворк підтримує різні методи аутентифікації, включаючи базову аутентифікацію, форму входу, аутентифікацію через сторонні системи, такі як OAuth, LDAP, SAML та інші. Завдяки гнучкості конфігурацій, Spring Security дозволяє налаштовувати безпеку додатків на різних рівнях, забезпечуючи як прості, так і складні сценарії захисту. Spring Security також пропонує потужні засоби для управління сесіями користувачів, включаючи можливість відстеження активних сесій та завершення сесій за певних умов. Це забезпечує додатковий рівень захисту, запобігаючи несанкціонованому доступу до системи. Крім того, вбудовані механізми захисту від CSRF-атак допомагають захистити додатки від підробки запитів між сайтами, що є критичним для веб-додатків, які обробляють конфіденційні дані.

Spring Web дозволяє легко створювати веб-додатки за допомогою Spring Framework [16]. Він надає інструменти для розробки контролерів, валідації даних, обробки запитів та відображення результатів на стороні клієнта.

Використовуючи Spring Web, розробники можуть створювати RESTful веб-сервіси, що дозволяють додаткам взаємодіяти з іншими системами через HTTP. Spring Web також включає підтримку для обробки асинхронних запитів, що забезпечує високу продуктивність і масштабованість веб-додатків. Завдяки інтуїтивному синтаксису та багатій документації, розробка з використанням Spring Web стає швидкою та ефективною, дозволяючи зосередитися на бізнес-логіці, а не на інфраструктурних питаннях.

Lombok спрощує розробку Java-програм за допомогою автоматичної генерації методів, які зазвичай вимагають багато коду, таких як гетери, сетери, конструктори та інші [17]. Використовуючи анотації, розробники можуть значно зменшити обсяг шаблонного коду, що підвищує читабельність та підтримуваність коду. Lombok автоматично генерує методи під час компіляції, що дозволяє зберігати код чистим і лаконічним, одночасно забезпечуючи всі необхідні функціональні можливості. Це особливо корисно при роботі з великими класами, де багато полів і необхідно реалізувати безліч методів для доступу до них.

Використання офіційного драйвера PostgreSQL забезпечить оптимальну швидкість та надійність підключення до бази даних PostgreSQL з вашого Java-додатка [18]. Офіційний драйвер розробляється і підтримується спільнотою PostgreSQL, що гарантує його актуальність і відповідність останнім версіям бази даних. Драйвер підтримує всі основні функції PostgreSQL, включаючи транзакції, підготовлені запити, обробку великих обсягів даних та роботу з різними типами даних. Використання офіційного драйвера також забезпечує сумісність з іншими інструментами та фреймворками, що використовуються в екосистемі Java.

Maven є потужним інструментом для управління залежностями та збірки проекту [19]. Він надає можливість ефективно використовувати сторонні бібліотеки та інструменти розробки, а також автоматизує процес збірки та розгортання додатків. Використовуючи файл конфігурації (pom.xml), розробники можуть легко визначати залежності проекту та керувати ними.

Maven автоматично завантажує необхідні бібліотеки з центрального репозиторію, що значно спрощує процес налаштування середовища розробки. Крім того, Maven забезпечує стандартизований підхід до структури проектів, що полегшує співпрацю в команді та інтеграцію з іншими інструментами, такими як CI/CD системи. Maven також підтримує плагіни, що розширюють його функціональність, дозволяючи автоматизувати різні аспекти розробки, тестування та розгортання додатків.

3.2 Розробка модуля авторизації та реєстрації

Головною метою створення веб-застосунку є моніторинг та управління закладом громадського харчування. Веб-застосунок дозволяє відстежувати різноманітні аспекти роботи закладу, такі як замовлення, обслуговування клієнтів, а також забезпечує ефективний контроль та оптимізацію процесів. Для того щоб система знала, який користувач авторизувався чи зареєструвався, потрібно надіслати на сервер певний об’єкт типу користувач, який містить всю необхідну інформацію для ідентифікації та автентифікації користувача (рисунок 3.1).

```
1   package com.kurs.kurs;
2
3   import lombok.Getter;
4   import lombok.Setter;
5
6   import java.util.Date;
7
    no usages
8   @Getter
9   @Setter
10  public class User {
11      private int id;
12      private String username;
13      private String password;
14      private String email;
15      private Date registrationDate;
16
17      // Конструктор класу
        1 usage
18      public User(int id, String username, String password, String email, Date registrationDate) {...}
25
26      @Override
27      public String toString() {...}
35  }
36
```

Рисунок 3.1 – Об’єкт «Користувач»

Після того, як система отримала потрібні дані, відбувається обробка цієї інформації. Процес обробки включає валідацію отриманих даних, їх зберігання в базі даних, а також створення полів для зберігання або використання відповідної інформації. Це забезпечує можливість подальшого використання даних для управління доступом користувачів до різних функцій веб-застосунку, а також для надання персоналізованих послуг.

На рисунку 3.2 зображено контролери, які відповідають за отримання та обробку інформації під час реєстрації.

```
@PostMapping("/register")
public ResponseEntity<?> registerUser(@RequestBody UserRegistrationRequest request) {
    // Перевірка, чи існує користувач з таким же ім'ям
    if (userRepository.findByUsername(request.getUsername()) != null) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body("User with this username already exists.");
    }

    // Хешування паролю перед збереженням у базі даних
    String hashedPassword = bCryptPasswordEncoder.encode(request.getPassword());

    // Створення нового користувача
    User newUser = new User(request.getUsername(), hashedPassword, request.getEmail());
    userRepository.save(newUser);

    return ResponseEntity.status(HttpStatus.CREATED).body("User registered successfully.");
}
```

Рисунок 3.2 – Обробка реєстрації користувача

На рисунку 3.3 зображено контролери, які відповідають за отримання та обробку інформації під авторизації.

```
@PostMapping("/login")
public ResponseEntity<?> loginUser(@RequestBody UserLoginRequest request) {
    // Знаходження користувача за ім'ям користувача
    User user = userRepository.findByUsername(request.getUsername());

    // Перевірка наявності користувача та відповідності паролю
    if (user == null || !bCryptPasswordEncoder.matches(request.getPassword(), user.getPassword())) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid username or password.");
    }

    // Вхід успішний
    return ResponseEntity.ok( body: "Login successful.");
}
```

Рисунок 3.3 – Обробка авторизації користувача

Контролер авторизації отримує дані про користувача, перевіряє їх на відповідність збереженим у базі даних обліковим записам та надає доступ до системи у разі успішної перевірки. Контролер реєстрації, у свою чергу, обробляє нові реєстраційні дані, перевіряє їх на унікальність та коректність, і створює новий обліковий запис у системі. Обидва контролери відіграють важливу роль у забезпеченні надійної та безпечної роботи веб-застосунку, гарантуючи, що тільки авторизовані користувачі мають доступ до його функціональності. Таким чином, для забезпечення безпеки системи від не контрольованого внесення змін в базі даних.

3.3 Розробка сервісу для формування звіту

Основною метою модуля для формування звіту є надання менеджерам можливості швидкого і ефективного формування звіту на основі кількості проданих позицій меню. Такий модуль дозволяє керівництву закладу громадського харчування отримувати актуальну та детальну інформацію про продажі, що допомагає приймати обґрунтовані управлінські рішення. Завдяки цьому інструменту, менеджери можуть легко відстежувати, які страви та напої є найпопулярнішими серед клієнтів, які з них приносять найбільший дохід, а також виявляти тенденції в продажах протягом різних періодів часу. На рисунку 3.4 продемонстрований конструктор, який обробляє запит на формування звіту.

Цей конструктор є центральною частиною модуля звітності. Він приймає параметри, що визначають критерії відбору даних для звіту, такі як дата, час, категорія меню та інші. Після отримання запиту конструктор обробляє дані, зберігає їх у тимчасових структурах і форматує для представлення у зручному для аналізу вигляді. Такий підхід до формування звітів дозволяє забезпечити високу гнучкість і налаштованість системи під потреби конкретного закладу.

```java
@RestController
public class ReportController {

    1 usage
    @Autowired
    private ReportService reportService;

    no usages
    @GetMapping("/report")
    public List<Report> generateReport(@RequestParam("startDate") Date startDate,
                                       @RequestParam("endDate") Date endDate) {
        // Отримання звіту від сервісу на основі вказаних дат
        return reportService.generateReport(startDate, endDate);
    }
}
```

Рисунок 3.4 – Контролер для запиту на формування звіту

На рисунку 3.5 сервіс який витягує дані з бази даних та формує об’єкт звіту який відображається користувачеві.

```java
@Service
public class ReportService {

    1 usage
    @Autowired
    private SalesRepository salesRepository ;

    1 usage
    public List<Sale> generateReport(Date startDate, Date endDate) {
        return salesRepository.findBySaleDateBetween(startDate, endDate);
    }
}
```

Рисунок 3.5 – Сервіс для формування звіту

Отже, сервіс, який надає можливість створення звіту, відповідає потребам менеджера у забезпеченні контролю над функціонуванням закладу. Завдяки швидкому формуванню звітів досягається підвищення продуктивності працівників і оптимізується їхня робота.

Також для візуалізацій системи було створена графічне представлення на рисунку 3.6, взаємодії користувача, модератора та модулів системи, а саме нейронної мережі, модуля авторизації та реєстрації.

.

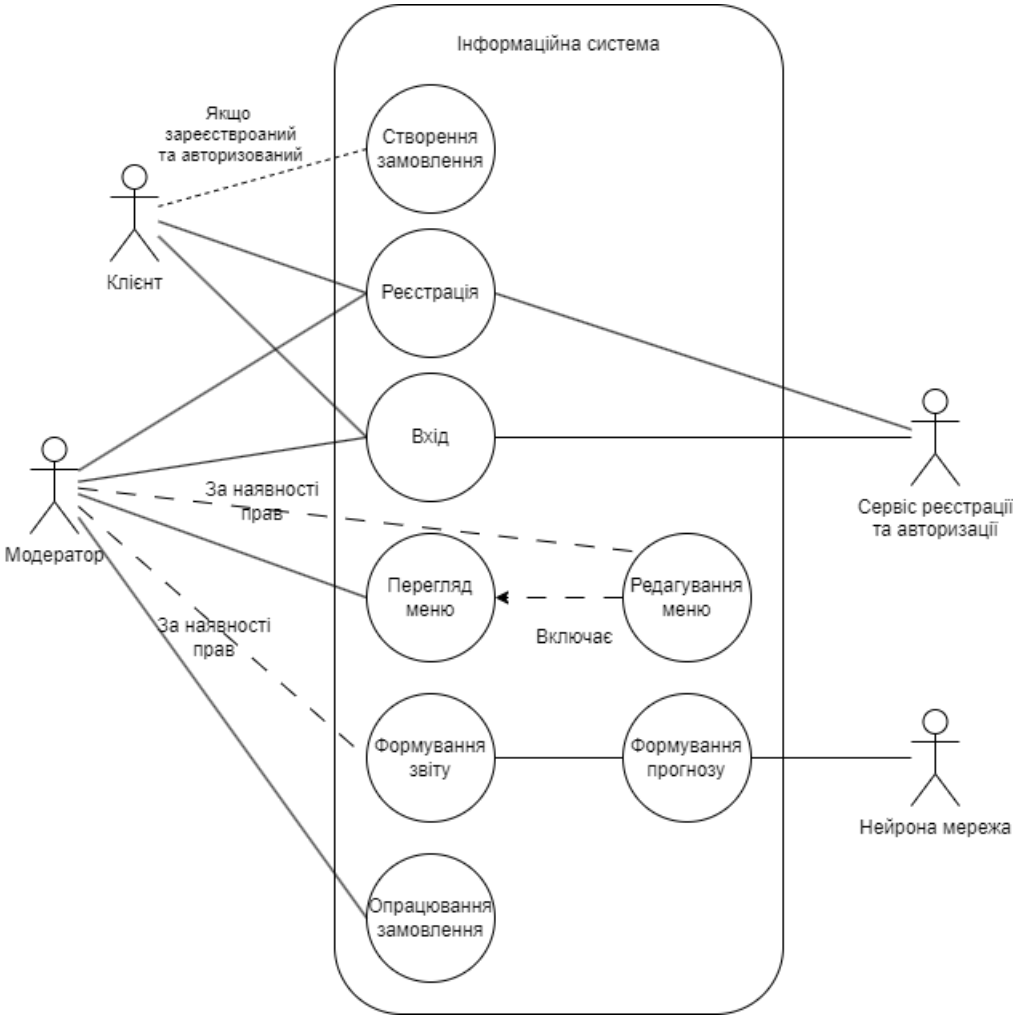

Рисунок 3.6 – Use case діаграма взаємодії з системою

Як видно з діаграмі модуль реєстрації та авторизації системи є необхідним, а модуль нейронної мережі, тому система може працювати і без нього, це забезпечує роботу інформаційної системи навіть за відсутності нейронної мережі, але за такої ситуації функціонал системи буде не повний.

3.4 Розробка інтерфейсу інформаційної системи

На основі створених моделей було реалізовано низку сторінок з інтерфейсом, кожна з яких відіграє ключову роль у забезпеченні зручності та ефективності взаємодії користувачів із системою. Ці сторінки розроблені з урахуванням потреб користувачів та вимог до функціональності системи, що дозволяє максимально спростити та покращити користувацький досвід. Однією з найважливіших сторінок є сторінка реєстрації користувача. Вона забезпечує новим користувачам можливість швидко і легко створити обліковий запис у системі. На цій сторінці представлені всі необхідні поля для введення особистих даних, таких як ім'я, електронна адреса, та пароль.

Крім того, сторінка включає механізми валідації введених даних для забезпечення коректності та безпеки. Наприклад, пароль має відповідати певним критеріям складності, а електронна адреса повинна бути у правильному форматі. Це знижує ризик помилок під час реєстрації та підвищує загальну безпеку системи.

На рисунку 3.7 наведено приклад сторінки реєстрації користувача з усіма необхідними полями. Ця сторінка має зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам без зайвих труднощів завершити процес реєстрації. Оформлення сторінки, її дизайн та розміщення елементів ретельно продумані для забезпечення максимального комфорту користувачів. Поля введення розташовані таким чином, щоб користувач міг послідовно заповнювати їх, не пропускаючи жодного важливого кроку.

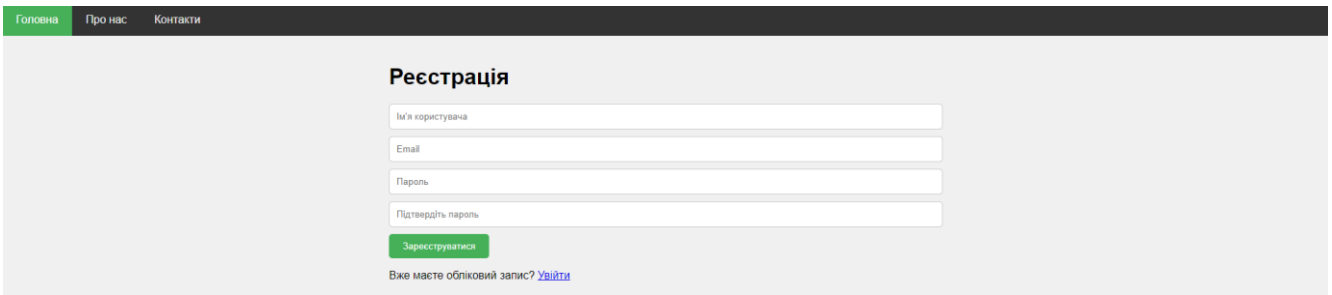

Рисунок 3.7 – Сторінка реєстрації користувача

Сторінка входу користувача з усіма необхідними полями наведена на рисунку 3.8. Вона була розроблена з урахуванням сучасних вимог до зручності та безпеки користувачів. Інтерфейс цієї сторінки містить такі елементи, як поля для введення імені користувача та пароля, кнопки для входу в систему та посилання для реєстрації.

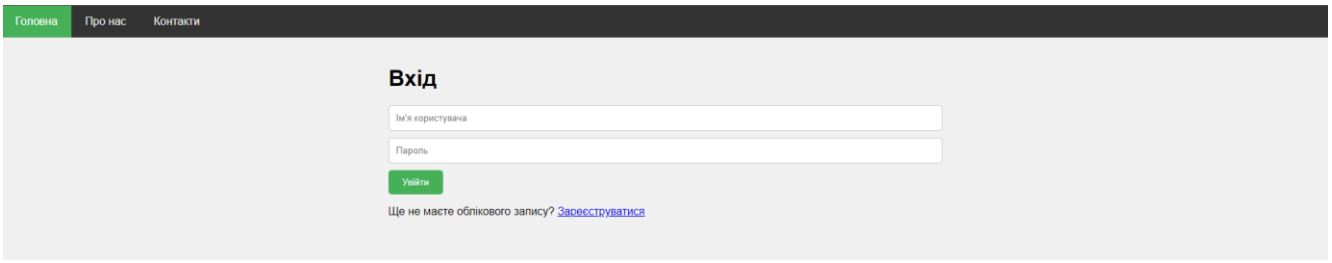

Рисунок 3.8 – Сторінка входу користувача

Сторінка меню з усіма необхідними елементами наведена на рисунку 3.9. Ця сторінка була розроблена з урахуванням потреб користувачів, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс. Вона містить усі необхідні елементи для ефективної навігації та взаємодії з функціоналом системи.

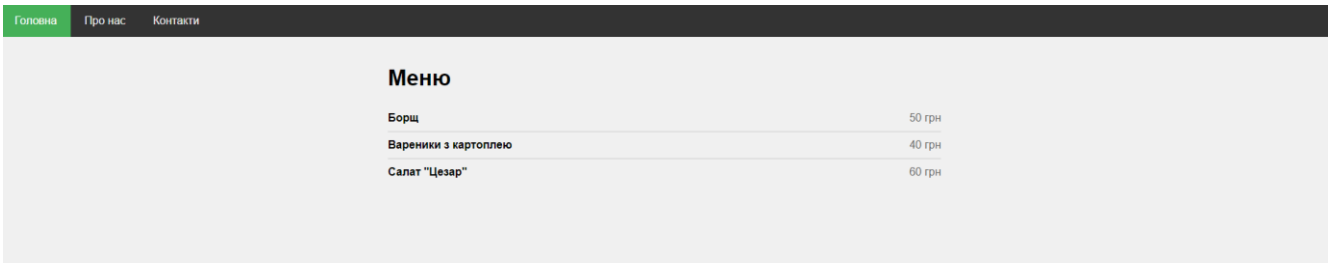

Рисунок 3.9 – Сторінка меню

Сторінка формування звіту з усіма необхідними елементами, наведена на рисунку 3.10, є ключовим інструментом для аналізу та візуалізації даних в інформаційній системі. Ця сторінка розроблена з урахуванням потреб користувачів у швидкому та зручному формуванні звітів.

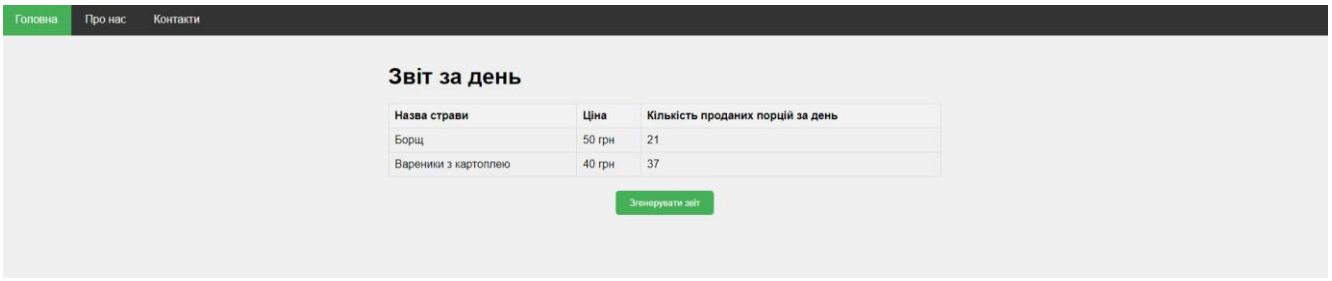

Рисунок 3.10 – Сторінка формування звіту

Отже, створення графічного інтерфейсу є ключовим аспектом успіху будь-якого програмного продукту, оскільки від нього залежить сприйняття системи користувачем. Якісно розроблений інтерфейс забезпечує зручність та ефективність взаємодії, поліпшує користувацький досвід і задоволення від використання продукту. Правильно спроектований інтерфейс сприяє приваблennю користувачів, підвищує їхню віддачу та створює позитивне враження від продукту, що може значно підвищити конкурентоспроможність системи на ринку. Розробка графічного інтерфейсу була виконана та реалізована в середовищі IntelliJ IDEA за допомогою технологій HTML, CSS та Java Script.

3.5 Архітектура нейронної мережі

Для розробки власної нейронної мережі була обрана архітектура багатошаровий персептрон [20].

Багатошаровий персептрон (MLP) є одним з найбільш поширених типів штучних нейронних мереж, використовуваних для задач регресії та класифікації структурованих даних. Для аналізу звітів закладів харчування, таких як фінансові показники та показники продажів, MLP може ефективно обробляти та аналізувати ці дані для прогнозування та прийняття рішень.

Зважаючи на всі дані, вхідний шар буде мати 6 нейронів, відповідно до характеристик, таких як обсяг продажів, витрати, кількість клієнтів, середній чек тощо. Перший прихований шар буде мати 64 нейрони, другий прихований шар буде мати 32 нейрони, і вихідний шар буде мати 1 нейрон для прогнозування.

На рисунку 3.11 зображено графічне представлення обраної архітектури MLP для аналізу звітів закладів харчування. Це представлення ілюструє структуру моделі з вхідним шаром, прихованими шарами та вихідним шаром, а також показує, як дані проходять через ці шари для отримання прогнозованого результату. Вибір правильної архітектури MLP є ключовим для досягнення високої точності та ефективності моделі. Оптимальна кількість шарів, нейронів у кожному з них, а також використання відповідних функцій активації та оптимізаційних методів грають важливу роль у здатності моделі до навчання та узагальнення. Це графічне представлення допомагає візуалізувати процес роботи MLP і розуміти, як модель обробляє дані для генерації корисних висновків з звітів про заклади харчування.

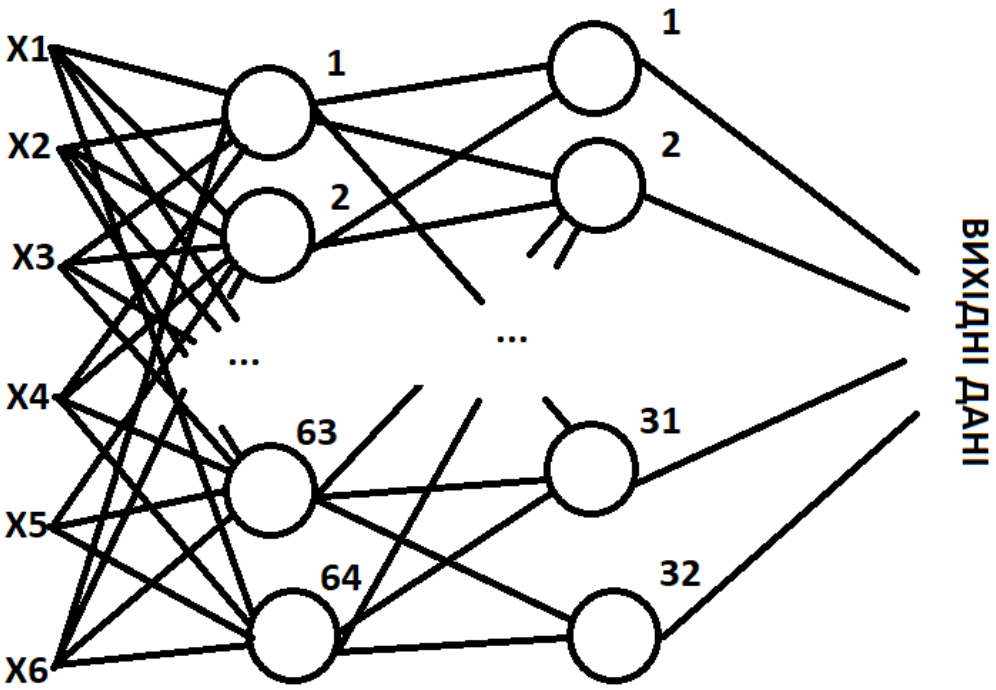

Рисунок 3.11 – Графічне зображення архітектури нейронної мережі

Архітектура багатошарового персептрона (MLP) для аналізу звітів закладів харчування складається з кількох важливих компонентів, які визначають його ефективність і точність. Основні складові включають вхідний шар, приховані шари та вихідний шар. Ефективна архітектура MLP для аналізу звітів закладів

харчування вимагає уважного підбору всіх її компонентів. Це включає вибір оптимальної кількості шарів, нейронів, функцій активації та використання регуляризаційних та оптимізаційних технік. Лише з правильним налаштуванням цих параметрів можна досягти високої точності прогнозування та здатності моделі до узагальнення на нові дані.

3.6 Висновки

У третьому розділі проведено ретельний аналіз та аргументовано вибір конкретних інструментів, які будуть використовуватися для розробки програмного продукту. Це рішення приймалося на основі комплексного дослідження вимог до системи, технологічних можливостей, зручності використання та масштабованості. Серед цих інструментів зазначаються такі технології, як Java, Spring Framework, Spring Security, Spring Web, Lombok, PostgreSQL Driver, Maven і PostgreSQL. Кожен із них має свої унікальні переваги та можливості, які сприятимуть ефективній розробці та функціонуванню програми.

Були реалізовані ключові модулі, необхідні для роботи інформаційної системи, включаючи модулі для обробки користувацьких даних, управління доступом, обробки запитів та зберігання інформації. Було розглянуто їх взаємодію, що дозволило створити злагоджену і ефективну архітектуру системи. Кожен модуль був ретельно протестований для забезпечення його стабільності та продуктивності.

Також було обрано тип нейронної мережі для розробки інтелектуальної компоненти додатку, а саме багатошаровий персептрон (MLP). MLP є одним з найпоширеніших типів штучних нейронних мереж і використовується для розв'язання задач класифікації, регресії та інших. Архітектура нейронної мережі була ретельно розроблена з урахуванням специфічних вимог проекту. Вибір MLP обґрунтований його здатністю навчатися на великих обсягах даних і знаходити складні залежності між вхідними та вихідними даними. Нейронна

мережа була протестована та оптимізована для забезпечення високої точності та продуктивності. Таким чином, проведений аналіз та вибір технологій забезпечили надійну основу для розробки програмного продукту, що відповідає високим вимогам до якості, продуктивності та безпеки.

4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ ВИКОРИСТАННЯ НЕЙРОНОЇ МЕРЕЖІ В ЗАКЛАДАХ ГРОМАДЦЬКОГО ХАРЧУВАННЯ

4.1 Аналіз загальної ситуації про використання нейронної мережі на комерційних установах

Штучний інтелект (ШІ) став однією з найбільш обговорюваних інновацій у світі сучасних технологій. Він швидко перетворюється з теоретичної концепції на реальність, особливо в контексті комерційних компаній. Використання ШІ в системах комерційних компаній має потенціал перевернути традиційні моделі бізнесу, забезпечити конкурентні переваги та відкрити нові можливості. Давайте розглянемо детальніше, як ШІ використовується в системах комерційних компаній.

Оптимізація процесів: Одні з перших областей, де ШІ застосовується в комерції, - це оптимізація бізнес-процесів. Системи, що використовують ШІ, можуть автоматизувати та оптимізувати рутинні завдання, що дозволяє компаніям зосередитися на стратегічних аспектах діяльності.

Персоналізація: ШІ дозволяє створювати персоналізовані продукти та послуги, адаптовані до потреб кожного конкретного клієнта. Це може включати індивідуалізовані рекомендації, персоналізовану рекламу та послуги з урахуванням унікальних вподобань та попередньої поведінки користувача.

Підвищення продуктивності Використання: ШІ може підвищити продуктивність робочих процесів, зменшити кількість помилок та сприяти швидшому прийняттю рішень. Автоматизація рутинних завдань і оптимізація робочих процесів можуть значно збільшити ефективність бізнесу.

Прогнозування та аналітика: ШІ дозволяє проводити більш точний аналіз даних та прогнозувати майбутні тенденції. За допомогою алгоритмів машинного навчання, компанії можуть виявити приховані закономірності в даних, що допоможе їм приймати кращі рішення.

Клієнтський сервіс: ШІ дозволяє розвивати інтелектуальні системи підтримки клієнтів, які забезпечують 24/7 підтримку через чат-боти, голосових асистентів та інші інтерактивні інтерфейси.

Боротьба з шахрайством та злочинністю: ШІ може використовуватися для виявлення шахрайства та злочинної діяльності, а також для захисту від кібератак та витоку даних.

Недоліки: Незважаючи на безліч переваг, використання ШІ в системах комерційних компаній також має свої недоліки. Наприклад, високі витрати на впровадження та обслуговування систем, проблеми з етикою та конфіденційністю даних, а також потенційні помилки в алгоритмах. Загалом, використання ШІ в системах комерційних компаній відкриває безліч можливостей для інновацій та покращення бізнес-процесів. Проте важливо розуміти, що успіх в цій області вимагає глибокого розуміння технологій та готовності до постійних змін і вдосконалення.

Використання штучного інтелекту (ШІ) в сфері послуг має як переваги, так і недоліки:

Переваги:

1. Автоматизація процесів: ШІ може автоматизувати рутинні та повторювані завдання, що дозволяє звільнити час працівників для виконання більш складних завдань або збільшення ефективності роботи.

2. Покращення обслуговування клієнтів: Застосування ШІ в електронних системах підтримки може покращити обслуговування клієнтів через автоматизовану відповідь на запитання, персоналізовані рекомендації тощо.

3. Аналіз великих обсягів даних: ШІ може швидко аналізувати великі обсяги даних і витягати з них корисні відомості для прийняття рішень щодо оптимізації бізнес-процесів.

4. Зменшення витрат: Автоматизація завдань за допомогою ШІ може зменшити витрати на робочу силу та покращити ефективність операцій.

5. Покращення прогнозування та планування: ШІ може допомогти у прогнозуванні попиту на послуги, що дозволяє бізнесу краще планувати свої дії та ресурси.

Недоліки:

1. Відсутність людського контакту: Використання ШІ може призвести до відчуття відсутності особистого контакту з клієнтом, що може негативно позначитися на сприйнятті сервісу.

2. Потенційні помилки: Навіть найрозробленіші системи ШІ можуть робити помилки, особливо в нестандартних ситуаціях або при неправильному навчанні моделей.

3. Проблеми з конфіденційністю даних: Використання ШІ може створити проблеми з конфіденційністю даних клієнтів, особливо якщо ці дані зберігаються в електронній формі та піддаються обробці алгоритмами ШІ.

4. Потреба у підтримці та обслуговуванні: Впровадження систем ШІ вимагає постійної підтримки та обслуговування для забезпечення їх коректної роботи та адаптації до змін у бізнес-процесах.

5. Етичні питання: Існує ризик виникнення етичних проблем у використанні ШІ, таких як біаси у моделях, ризик дискримінації або використання систем для маніпуляції користувачами.

Загалом, використання ШІ в сфері послуг має великий потенціал для покращення ефективності та якості обслуговування, проте вимагає уваги до різноманітних аспектів, включаючи етичні, конфіденційність даних та підтримку користувачів.

Прогнозування та аналітика з використанням штучного інтелекту (ШІ) є однією з ключових областей, де ця технологія може значно покращити ефективність та результативність діяльності комерційних компаній. Давайте розглянемо детальніше, як саме ці аспекти впливають на бізнес. Прогнозування попиту Прогнозування попиту на товари та послуги є критичним для планування виробництва, логістики, складського управління та маркетингу. ШІ дозволяє аналізувати великі обсяги даних про попередні продажі, клієнтські вподобання,

сезонні та інші фактори, що впливають на попит, та робити точні прогнози на майбутнє. Це допомагає компаніям зменшити ризики перепродажу або недопродажу товарів, оптимізувати запаси та забезпечити задоволення попиту клієнтів. Оптимізація ціноутворення Аналітика даних в поєднанні з ШІ дозволяє компаніям аналізувати конкурентні ціни, динаміку попиту та інші фактори, що впливають на ціноутворення. На основі цього аналізу можуть бути визначені оптимальні цінові стратегії для максимізації прибутку або ринкової частки. Аналіз клієнтського поведінки Використання ШІ дозволяє аналізувати клієнтську поведінку на основі даних про їхні покупки, перегляди товарів, взаємодії з веб-сайтами та іншої інформації. Це дозволяє розуміти потреби та вподобання клієнтів, створювати персоналізовані пропозиції, підтримувати зв'язок з клієнтами та покращувати їхнє задоволення від обслуговування. Виявлення тенденцій та відхилень За допомогою аналізу даних та машинного навчання компанії можуть виявляти тенденції та відхилення в ринкових умовах, споживчому попиті та інших аспектах діяльності. Це дозволяє бізнесу оперативно реагувати на зміни, приймати вчасні стратегічні рішення та використовувати нові можливості для розвитку. Прогнозування ризиків Аналітика даних та ШІ можуть бути використані для прогнозування ризиків, пов'язаних з фінансовими операціями, логістикою, ринковою конкуренцією та іншими аспектами бізнесу. Це допомагає компаніям управляти ризиками більш ефективно та уникати потенційних проблем. Заключення Прогнозування та аналітика з використанням штучного інтелекту можуть значно покращити стратегічне планування, оптимізацію операцій та прийняття рішень в комерційних компаніях. Ці технології дозволяють підвищити ефективність, знизити ризики та забезпечити конкурентні переваги на ринку.

Перед використанням штучного інтелекту при розробці комерційних інформаційних систем потрібно враховувати кілька важливих аспектів:

1. Збір та обробка даних: Потрібно мати достатньо якісних та репрезентативних даних для навчання моделей штучного інтелекту. Ці дані повинні бути коректно зібрані, очищені від шуму та аналізовані з урахуванням

конфіденційності та відповідності законодавству про захист персональних даних.

2. Точність інтелектуальних моделей: Інтелектуальні моделі, створені за допомогою штучного інтелекту, повинні бути точними і надійними. Вони повинні правильно адаптуватися до різноманітних сценаріїв використання та враховувати можливість виникнення непередбачених ситуацій.

3. Етичність і прозорість: Розробники повинні враховувати етичні аспекти використання штучного інтелекту, зокрема уникати виникнення помилок у моделях, захищати приватність користувачів та гарантувати прозорість у рішеннях, прийнятих інтелектуальними системами.

4. Масштабованість та інтеграція: Інформаційні системи, що використовують штучний інтелект, повинні бути готові до масштабування та інтеграції з існуючими системами організації. Вони повинні легко взаємодіяти з іншими додатками та програмним забезпеченням.

5. Безпека: Важливо забезпечити високий рівень захисту системи від потенційних кіберзагроз та атак, оскільки інформаційні системи, які використовують штучний інтелект, можуть містити чутливі дані про користувачів і організацію в цілому.

6. Підтримка та навчання персоналу: Користувачі та адміністратори системи повинні отримати достатню підтримку та навчання для ефективного використання і управління інтелектуальними функціями системи.

Врахування цих аспектів допоможе забезпечити успішну розробку та впровадження інформаційних систем з використанням штучного інтелекту для комерційних цілей.

Створення інформаційної системи для керування закладами громадського харчування з використанням штучного інтелекту для формування звітів може бути дуже корисним і ефективним підходом. Така система може включати в себе різноманітні функції:

Управління запасами і складами: Система може автоматично відстежувати кількість продуктів на складі і робити прогнози їхнього поповнення на основі попередніх продажів та інших факторів.

Планування меню: Штучний інтелект може аналізувати попередні замовлення та враховувати популярність різних страв для підготовки оптимального меню.

Управління витратами: Система може аналізувати витрати на продукти, працю, енергію та інші ресурси для оптимізації витрат і збільшення прибутковості.

Формування звітів: Інтелектуальні алгоритми можуть автоматично аналізувати дані і формувати звіти про продажі, витрати, ефективність роботи персоналу та іншу інформацію, необхідну для прийняття управлінських рішень.

Прогнозування попиту: З використанням аналізу даних та машинного навчання система може прогнозувати майбутній попит на страви і напої, що допоможе оптимізувати запаси і зменшувати витрати.

Управління персоналом: Система може допомагати в плануванні графіків роботи персоналу на основі очікуваного обсягу роботи.

Створення такої системи потребує інтеграції різних технологій, включаючи бази даних, аналітику даних, машинне навчання та інтелектуальний аналіз. Також важливо враховувати вимоги до захисту даних, оскільки інформація про клієнтів та фінансові дані можуть бути чутливими.

Навчання нейронної мережі для аналізу звітів закладів харчування має на меті підвищити ефективність прийняття рішень та оптимізувати процеси управління. Завдяки використанню великих обсягів даних та потужності сучасних обчислювальних систем, нейронні мережі здатні виявляти приховані закономірності та прогнозувати майбутні тенденції. У цьому розділі розглянемо основні етапи навчання нейронної мережі для цього завдання, включаючи збір даних, передобробку, архітектуру мережі, навчання та оцінку моделі.

4.2 Збір та підготовка даних

Збір даних Збір даних є фундаментальним етапом у процесі навчання нейронної мережі для аналізу звітів закладів харчування. Якість даних безпосередньо впливає на точність прогнозів та ефективність моделі.

Передобробка даних є критично важливим кроком для підвищення якості моделей машинного навчання. Передобробка включає декілька основних етапів такі як очищення даних, видалення дублікатів, корекція помилок, нормалізація та масштабування, кодування категоріальних змінних, розділення даних.

Збір та передобробка даних є невід'ємною частиною процесу навчання нейронної мережі для аналізу звітів закладів харчування. Ці етапи забезпечують високу якість вхідних даних, що дозволяє отримати точні та надійні результати. Ретельна передобробка та аналіз даних створюють міцну основу для побудови ефективної та продуктивної моделі.

Для навчання нейроної мережі було використано набір даних, з відкритого джерела «kaggle» [21] на 16 тисяч замовлень, ресурс з таблицею з даними наведено на рисунку 4.1.

# Order Number Order Number	Order Date Date of order	Item Name Name of Product	# Quantity Quantity of product in order	# Product Price Product Price	# Total p Total prod
630 16.1k	13081 unique values	Pilau Rice 6% Plain Naan 5% Other (66344) 89%	1.00 - 3.50 Count: 73,222 1 51	0.5 17.9	1
16118	03/08/2019 20:25	Plain Papadum	2	0.8	6
16118	03/08/2019 20:25	King Prawn Balti	1	12.95	6
16118	03/08/2019 20:25	Garlic Naan	1	2.95	6
16118	03/08/2019 20:25	Mushroom Rice	1	3.95	6
16118	03/08/2019 20:25	Paneer Tikka Masala	1	8.95	6
16118	03/08/2019 20:25	Mango Chutney	1	0.5	6
16117	03/08/2019 20:17	Plain Naan	1	2.6	7
16117	03/08/2019 20:17	Mushroom Rice	1	3.95	7
16117	03/08/2019 20:17	Tandoori Chicken (1/4)	1	4.95	7
16117	03/08/2019 20:17	Vindaloo - Lamb	1	7.95	7
16117	03/08/2019 20:17	Chapati	1	1.95	7
16117	03/08/2019 20:17	Lamb Tikka	1	4.95	7

Рисунок 4.1 – Таблиця з набором даних на сайті «kaggle»

Після оброки даних, видалення дублікатів і виправлення помилок у записах залишилося приблизно 13 тисяч замовлень. Ці дані а саме 9 тисяч замовлень будуть використовуватися для навчання нейронної мережі, а 4 тисячі будуть використані для перевірки роботи нейронної мережі.

4.3 Процес навчання нейронної мережі

Навчання нейронної мережі є критично важливим етапом, який визначає її здатність ефективно вирішувати завдання прогнозування на основі вхідних даних. Процес навчання включає кілька ключових аспектів: підготовку даних, вибір гіперпараметрів, визначення функції втрат, оптимізацію ваг мережі, регуляризацію для запобігання перенавчанню та оцінку продуктивності моделі.

Перед початком навчання нейронної мережі необхідно ретельно підготувати дані. Це включає в себе розподіл даних на навчальний набір, валідаційний набір та тестовий набір. Навчальний набір становитиме 9 тисяч замовлень, валідаційний набір — 2 тисячі, а тестовий набір також 2 тисячі замовлень. Такий підхід дозволяє нейронній мережі навчатися на одному наборі даних, перевіряти свої результати на іншому та оцінювати остаточну продуктивність на тестовому наборі.

Для навчання було обрано кількість епох 100, що є стандартною кількістю для навчання нейронної мережі. Це означає, що мережа пройде по навчальному набору даних 100 разів. Така кількість епох дозволяє мережі поступово коригувати свої ваги і покращувати точність прогнозування. Крім того, розмір батчу було обрано 64. Цей параметр впливає на те, скільки зразків даних буде використовуватися для оновлення ваг мережі за один раз. Вибір розміру батчу 64, що трохи уповільнює процес навчання, але збільшує точність прогнозування.

Швидкість навчання, встановлена на рівні 0.005, визначає, на скільки будуть змінюватися ваги при кожному оновленні. Цей параметр є важливим, оскільки занадто велика швидкість навчання може призвести до нестабільного навчання, а занадто мала — до повільного збирання потрібних знань.

Під час навчання також важливо обрати правильну функцію втрат, яка буде відображати, наскільки добре мережа виконує свої завдання. Оптимізація ваг мережі відбувається за допомогою різних алгоритмів, таких як градієнтний спуск, що дозволяє мінімізувати функцію втрат і покращити продуктивність моделі.

Оцінка продуктивності моделі проводиться на валідаційному та тестовому наборах, щоб зрозуміти, наскільки добре модель навчилася і наскільки вона зможе ефективно застосовувати свої знання на нових даних. Цей етап є вирішальним для визначення успіху всього процесу навчання нейронної мережі.

Під час тестування було встановлено, що система надає позитивні результати у 81% випадків, такий показник вказує на те що нейрона мережа є досить точною і відповідає потребам системи.

4.4 Тестування системи

Тестування, як завершальний етап розробки сайту, відіграє важливу роль у процесі створення якісного програмного забезпечення.

Spring пропонує різноманітні типи тестування для забезпечення надійності та якості розроблюваного програмного забезпечення. Ось кілька основних типів тестування, які підтримуються в Spring:

- Unit тестування: Це тип тестування, який перевіряє окремі компоненти програми (наприклад, класи, методи) без їх залежностей. Для unit тестування в Spring зазвичай використовуються фреймворки, такі як JUnit або TestNG.
- Integration тестування: Інтеграційне тестування перевіряє взаємодію між різними компонентами програми, такими як класи, сервіси, модулі. Spring надає підтримку для інтеграційного тестування за допомогою фреймворка Spring Test, який дозволяє запускати тести в контексті Spring.
- End-to-End (E2E) тестування: Це тестування функціональності програми від початку до кінця, включаючи всі її компоненти та залежності. Для E2E

тестування в Spring можна використовувати різні інструменти, такі як Selenium або Cucumber, які дозволяють автоматизувати тестові сценарії та перевіряти поведінку програми.

- Mock тестування: Mock тестування використовує макети (mock objects) для імітації поведінки залежностей компонентів. Spring надає функціональність для створення макетів об'єктів за допомогою фреймворку Mockito або EasyMock.
- Performance тестування: Це тип тестування, який оцінює продуктивність та швидкодію програми при певних навантаженнях. Для виконання performance тестування в Spring можна використовувати інструменти, такі як JMeter або Apache Bench.

Ці типи тестування дозволяють забезпечити високу якість та надійність програмного забезпечення, розробленого з використанням Java Spring.

Всі результати випробувань заносяться до відповідної документації та аналізуються розробниками для виявлення можливих проблем та покращення якості програмного забезпечення. Звіт про випробування виступає важливим засобом комунікації, що допомагає всім зацікавленим сторонам, включаючи керівників проектів та інших учасників команди, зрозуміти результати тестування та прийняти обґрунтовані рішення щодо готовності програмного забезпечення до випуску [22]. Документація також може включати описи тестових сценаріїв, виявлені помилки, відповідні виправлення та інші важливі дані, які допомагають у підтримці та подальшому розвитку програми.

Отже, етап тестування системи є критично важливим для переконання у її надійності та коректності перед введенням в експлуатацію. Проведене тестування включало реалізацію різноманітних видів тестів, таких як Unit тестування, інтеграційне тестування, End-to-End тестування, Mock тестування та Performance тестування. Кожен з цих видів тестів спрямований на перевірку різних аспектів системи та виявлення можливих проблем або недоліків. Такий комплексний підхід до тестування дозволяє забезпечити високу якість

програмного забезпечення перед його впровадженням та запобігає появленню проблем у подальшому.

4.5 Розробка інструкції користувача

Інструкція користувача складається з докладного опису основних етапів та процедур, які необхідно виконати для успішного використання програмного забезпечення. Ця документація допомагає користувачам зрозуміти, які вимоги повинні бути відповідним чином виконані для оптимальної роботи програми та як правильно налаштувати її на їхньому пристрої. Вона надає корисні вказівки та поради, які сприяють ефективному використанню програми. Вона містить інформацію про мінімальні та рекомендовані конфігурації, детально представлені у таблиці 4.1.

Мінімальні та рекомендовані конфігурації для запуску веб-застосунку можуть бути ключовими для забезпечення оптимальної продуктивності та коректної роботи програми. Мінімальна конфігурація зазвичай визначається як найнижчий рівень обладнання чи програмного середовища, який є необхідним для запуску застосунку, тоді як рекомендована конфігурація вказує на оптимальний рівень ресурсів для забезпечення найкращої продуктивності та виконання завдань. Для веб-застосунків мінімальна конфігурація може включати базовий веб-сервер або хмарне середовище з мінімальним обсягом оперативної пам'яті та процесорною потужністю, необхідною для запуску застосунку та обробки невеликого обсягу даних. У той час як рекомендована конфігурація може включати більш потужні сервери або хмарні ресурси з великим обсягом оперативної пам'яті, швидкими процесорами та можливістю масштабування для обробки великих обсягів трафіку та даних. Забезпечення відповідної конфігурації є важливою складовою успішного використання веб-застосунку і може визначатися відповідно до потреб користувача, обсягу даних, типу додатку та інших факторів.

Таблиця 4.1 – Мінімальна та рекомендована конфігурації для

	Мінімальна конфігурація	Рекомендована конфігурація
Тип процесора	Чотирьохядерний процесор з тактовою частотою близько 1.6 ГГц	Восьмиядерний процесор з тактовою частотою близько 3.0 ГГц
Об’єм оперативної пам’яті	4 ГБ	8 ГБ або більше
Місце на жорсткому диску	256 Мб	1 ГБ або більше
Операційна система	Неактуальна версія Windows, macOS Catalina або Linux	Остання версія Windows, macOS Catalina, або Linux

Після успішного входу в систему користувача передбачається перевірка етапів для забезпечення безпеки та правильності процесу, що включає в себе перевірку коректності введених облікових даних. Користувач може використовувати панель навігації застосунку для переходу між різними розділами веб-застосунку.

Отже, інструкція користувача відіграє важливу роль у запобіганні помилок та неправильного використання програмного забезпечення. Це допомагає уникнути втрати даних та забезпечити безперебійну роботу застосунку. Дотримання інструкцій дозволяє користувачам максимально використовувати можливості програми та підтримує їхню продуктивність.

4.6 Висновки

Навчання нейронної мережі для аналізу звітів закладів харчування є складним, але надзвичайно важливим процесом, який може значно підвищити ефективність управління та прийняття рішень у цій галузі. Завдяки використанню сучасних методів обробки даних та машинного навчання, можна отримати цінні інсайти, які дозволяють досягти значних покращень у роботі закладів харчування.

Цей процес включає збирання великої кількості даних, таких як відгуки клієнтів, дані про продажі, інвентаризацію, і навіть поведінкові патерни відвідувачів. Потім ці дані проходять через етапи очищення та підготовки, щоб бути готовими до аналізу. Після цього нейронна мережа навчається на цих даних, що дозволяє їй виявляти приховані закономірності та робити прогнози.

Застосування нейронних мереж в аналізі звітів допомагає управлінцям швидше і точніше ідентифікувати проблемні ділянки, оптимізувати витрати, покращити якість обслуговування клієнтів і навіть передбачити майбутні тенденції на ринку. Це, в свою чергу, сприяє підвищенню задоволеності клієнтів і збільшенню прибутків закладу.

Крім того, нейронні мережі можуть автоматизувати багато рутинних завдань, таких як обробка відгуків чи прогнозування попиту на різні страви, що дозволяє персоналу більше зосередитися на креативних та стратегічних аспектах управління. Таким чином, впровадження технологій машинного навчання в управління закладами харчування не тільки покращує їх роботу, але й сприяє розвитку індустрії в цілому, надаючи їй нові можливості для зростання та інновацій.

Також було проведено тестування функціональності та візуального оформлення компонентів нашого мобільного додатку. Після завершення тестування було підготовлено ретельне керівництво користувача для даної системи. Це керівництво надає докладні пояснення щодо кожної функції та можливостей програмного продукту, розкриваючи їх функціонал та способи

використання. Такий підхід допомагає користувачам швидко ознайомитися з можливостями додатку та максимально ефективно використовувати його у своїй роботі.

ВИСНОВКИ

У рамках бакалаврської дипломної роботи було розроблено програмні компоненти для інформаційної системи, що призначена для керування закладами громадського харчування. Для створення цих компонентів було використано середовище програмування Intellij Idea, а робота проводилася відповідно до методичних вказівок [23].

У процесі виконання бакалаврської дипломної роботи було проведено аналіз існуючих інформаційних систем для закладів громадського харчування. Були розглянуті основні аналоги програмного продукту, визначено їхні недоліки порівняно з власною реалізацією та сформульовано основні завдання бакалаврської дипломної роботи.

Під час аналізу технологій розробки був обґрунтований вибір мови програмування Java, а також використання Spring Framework та його модулів, зокрема Spring Security, Spring Web, Lombok, PostgreSQL Driver. Також були використані Maven та база даних PostgreSQL.

У процесі виконання бакалаврської дипломної роботи було розроблено модуль для авторизації та реєстрації користувачів, а також сервіс для генерації звітів. Після проведення тестування програмного забезпечення було виявлено, що всі основні функції працюють належним чином та відповідають вимогам, зазначеним у технічному завданні.

Крім того, під час тестування були перевірені різноманітні сценарії використання програми, що дозволило виявити та усунути всі виявлені помилки та недоліки. Отримані результати тестування підтверджують високу якість та готовність програмного продукту до використання користувачами згідно зі створеною інструкцією користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Носирєв О. О. Фактори впливу на розвиток туристичного і готельно-ресторанного бізнесу. Збірка матеріалів Міжнародної науково практичної інтернет-конференції «Актуальні проблеми та перспективи розвитку агропродовольчої сфери, індустрії гостинності та торгівлі», 02 листопада 2022 р. Харків: ДБУ. 2022. С. 278–280. [Електронний ресурс] – Режим доступу до ресурсу: https://tinyurl.com/2snn9vs6 (дата звернення: 15.03.2024).

2. Навіщо Інтернету речей потрібен штучний інтелект [Електронний ресурс] – Режим доступу до ресурсу: https://iotji.io/aiot-navischo-internetu-rechei-potriben-shtuchnyi-intelekt/ (дата звернення: 19.03.2024).

3. Кізін Г. О., Ткаченко О. М. Переваги та недоліки використання мікросервісної архітектури для розробки інформаційної системи для керування закладами громадського харчування (ВНТКП ВНТУ), м.Вінниця, 24-25 березня 2024 р. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20503/16972

4. Офіційний сайт Toast POS [Електронний ресурс] – Режим доступу до ресурсу: https://pos.toasttab.com/ (дата звернення: 22.03.2024).

5. Офіційний сайт Square for Restaurants [Електронний ресурс] – Режим доступу до ресурсу: https://squareup.com/us/en/point-of-sale/restaurants (дата звернення: 22.03.2024).

6. Офіційний сайт Revel Systems [Електронний ресурс] – Режим доступу до ресурсу: https://revelsystems.com/ (дата звернення: 22.03.2024).

7. Офіційний сайт Revel Systems [Електронний ресурс] – Режим доступу до ресурсу: https://www.touchbistro.com/ (дата звернення: 22.03.2024).

8. Особливості функціонування закладів ресторанного господарства в період кризи [Електронний ресурс] – Режим доступу до ресурсу: https://tourlib.net/statti_ukr/girnyak11.htm (дата звернення: 30.03.2024).

9. Автоматизація закладів громадського харчування програмою «Мінісофт Кафе» [Електронний ресурс] – Режим доступу до ресурсу:

https://minisoft.ua/article/avtomatizaciya-zakladiv-gromadskogo-kharchuvannya-programoyu-minisoft-kafe (дата звернення: 01.04.2024).

10. Java Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java (дата звернення: 06.03.2024).

11. Офіційний сайт Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-framework (дата звернення: 10.03.2024).

12. Офіційний сайт PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/ (дата звернення: 12.04.2024).

13. Інтерфейс Користувача (UI): Значення, Складові та Вплив на Продажі [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/chto-takoe-ui-i-kak-polzovatelskij-interfejs-vliyaet-na-prodazhi-internet-magazina (дата звернення: 15.04.2024).

14. Back-end розробка [Електронний ресурс] – Режим доступу до ресурсу: https://brainlab.com.ua/uk/blog-uk/back-end-rozrobka# (дата звернення: 18.04.2024).

15. Офіційний веб сторінка Spring Security [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/projects/spring-security (дата звернення: 20.04.2024).

16. Serving Web Content with Spring MVC [Електронний ресурс] – Режим доступу до ресурсу: https://spring.io/guides/gs/serving-web-content (дата звернення: 22.04.2024).

17. Офіційний сайт Project Lombok [Електронний ресурс] – Режим доступу до ресурсу: https://projectlombok.org/ (дата звернення: 25.04.2024).

18. Офіційний сайт PostgreSQL JDBC Driver [Електронний ресурс] – Режим доступу до ресурсу: https://jdbc.postgresql.org/ (дата звернення: 25.04.2024).

19. Офіційний сайт Apache Maven [Електронний ресурс] – Режим доступу до ресурсу: https://maven.apache.org/ (дата звернення: 27.04.2024).

20. Особливості тестування Spring-застосунків [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/42430/ (дата звернення: 27.04.2024).

21. Kaggle [Електронний ресурс] – https://www.kaggle.com/ (дата звернення: 27.05.2024).

22.Перцептрон [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD#:~:text=%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%CC%81%D0%BD%2C%20%D0%B0%D0%B1%D0%BE%20%D0%BF%D0%B5%D1%80%D1%81%D0%B5%D0%BF%D1%82%D1%80%D0%BE%CC%81%D0%BD%20(%D0%B0%D0%BD%D0%B3%D0%BB.,1%C2%BB%20%D1%83.%201960%20%D1%80%D0%BE%D1%86%D1%96 (дата звернення: 27.05.2024).

23. Романюк О. Н., Чорноволик Г. О., Методичні вказівки до виконання бакалаврських дипломних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення»: методичні – вказівки Вінниця : ВНТУ, 2022. 51 с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка інформаційної системи для керування закладами громадського харчування»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доцент О. М. Ткаченко
«12» березня 2024 р.
Виконав:
_____ студент гр. 1ПІ-20б Г. О. Кізін
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка інформаційної системи для керування закладами громадського харчування».

Галузь застосування – заклади громадського харчування.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою роботи э створення інформаційної системи для закладів громадського харчування для підвищення ефективності роботи за рахунок застосування нейронної мережі

Призначення роботи – інформаційна система для полегшення роботи та збільшення ефективності закладів громадського харчування.

4. Вихідні дані для проведення БДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

Герберт Шилдт, Java. Повне керівництво. 10-е видання (2 томи), 2020р., 730 + 780 с.

Ю. Козима, Р. Харроп, К. Шефер, К. Хо, Spring 5 для професіоналів, 5-е видання, 2019р., 1119с.

5. Технічні вимоги

Базові методи заповнення тексту, методи запису даних до бази даних, методи ефективного пошуку даних в базі даних, методи захисту даних та шифрування, кількість введених даних не обмежена, кінцевий інтерфейс для взаємодії з системою.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку та наявних аналогів інформаційних систем для закладів громадського харчування	13.03.24 –22.03.24
2	Розробка моделей та загальної архітектури інформаційної системи	13.03.24 –22.03.24
3	Розробка алгоритмів роботи та інтерфейсу системи	25.03.24 –19.04.24
4	Навчання нейронної мережі, тестування системи та нейронної мережі	22.04.24 –10.05.24
5	Оформлення матеріалів до захисту БДР	13.05.24 – 24.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інформаційної системи для керування закладами громадського харчування

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ткаченко О. М.

Оригінальність	94.2%
Схожість	5.8%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи Кізін Г. О.

Керівник роботи Ткаченко О. М.

Додаток В – Лістинг програми

```
package com.kurs.kurs.controller;

import com.kurs.kurs.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AuthController {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @PostMapping("/register")
    public ResponseEntity<?> registerUser(@RequestBody UserRegistrationRequest
request) {
        if (userRepository.findByUsername(request.getUsername()) != null) {
            return ResponseEntity.status(HttpStatus.CONFLICT).body("User with this
username already exists.");
        }
```

```
        String hashedPassword =
bCryptPasswordEncoder.encode(request.getPassword());

        User newUser = new User(request.getUsername(), hashedPassword,
request.getEmail());
        userRepository.save(newUser);

        return ResponseEntity.status(HttpStatus.CREATED).body("User registered
successfully.");
    }

    @PostMapping("/login")
    public ResponseEntity<?> loginUser(@RequestBody UserLoginRequest request) {
        User user = userRepository.findByUsername(request.getUsername());

        if (user == null || !bCryptPasswordEncoder.matches(request.getPassword(),
user.getPassword())) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid
username or password.");
        }
        return ResponseEntity.ok("Login successful.");
    }
}

package com.kurs.kurs.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

```
import java.util.Date;
import java.util.List;

@RestController
public class ReportController {

    @Autowired
    private ReportService reportService;

    @GetMapping("/report")
    public List<Sale> generateReport(@RequestParam("startDate") Date startDate,
                                     @RequestParam("endDate") Date endDate)
        return reportService.generateReport(startDate, endDate);
    }
}

package com.kurs.kurs.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.List;

@Service
public class ReportService {

    @Autowired
    private SalesRepository salesRepository ;
```

```
    public List<Sale> generateReport(Date startDate, Date endDate) {
        return salesRepository.findBySaleDateBetween(startDate, endDate);
    }
}

package com.kurs.kurs.controller;

import com.kurs.kurs.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Date;
import java.util.List;

public interface SalesRepository extends JpaRepository<Sale,Long> {
    List<Sale> findBySaleDateBetween(Date startDate, Date endDate);
}

package com.kurs.kurs.controller;

public class UserLoginRequest {
    private String username;
    private String password;

    public UserLoginRequest(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
```

```
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

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

package com.kurs.kurs.controller;

public class UserRegistrationRequest {
    private String username;
    private String password;
    private String email;
    public UserRegistrationRequest(String username, String password, String email) {
        this.username = username;
        this.password = password;
        this.email = email;
    }

    // Гетери та сетери для всіх полів класу
    public String getUsername() {
```

```
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
package com.kurs.kurs.controller;

import com.kurs.kurs.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,Long> {
```

```
    User findByUsername(String username);
}

package com.kurs.kurs;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class KursApplication {

        public static void main(String[] args) {
                SpringApplication.run(KursApplication.class, args);
        }

}
package com.kurs.kurs;

import lombok.Getter;
import lombok.Setter;

import java.util.Date;

@Getter
@Setter
public class User {
    private int id;
    private String username;
    private String password;
    private String email;
```

```
    private Date registrationDate;

    public User(String username, String password, String email) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.email = email;
        this.registrationDate = registrationDate;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", email='" + email + '\'' +
                ", registrationDate=" + registrationDate +
                '}';
    }
}
package com.testtask.testtask.api.service;

import com.testtask.testtask.api.dao.CustomerRepository;
import com.testtask.testtask.api.exeption.ValidationException;
import com.testtask.testtask.api.model.Customer;
import com.testtask.testtask.api.model.CustomerEntity;
import lombok.NoArgsConstructor;
import lombok.SneakyThrows;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.time.Instant;
import java.util.List;
import java.util.Optional;

@Service
@NoArgsConstructor
public class CustomerService {

   String EMAIL = "^(?=.{1,100}$)[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-
Z]{2,}$";
   String FULL_NAME = "^(?=\\s*\\S)([\\s\\S]){2,50}$";
   @Autowired
   private CustomerRepository customerRepository;

   public Customer getCustomer(Long id) {
      Optional<CustomerEntity> customerEntity = customerRepository.findById(id);
      return customerEntity.map(this::mapCustomerEntityToCustomer).orElse(null);
   }

   public List<Customer> getAllCustomers(){
      return
customerRepository.findAllByIsActiveIsTrue().stream().map(this::mapCustomerEnti
tyToCustomer).toList();
   }

   public Customer saveOrUpdateCustomer(CustomerEntity updatedCustomer){
      if(!validationCustomer(updatedCustomer)){
         return null;
      }
```

```
        if(updatedCustomer.getId() == null){
            updatedCustomer.setCreated(Instant.now());
            updatedCustomer.setIsActive(true);
        }
        else {
            return null;
        }
        customerRepository.save(updatedCustomer);
        return mapCustomerEntityToCustomer(updatedCustomer);
    }

    public Customer saveOrUpdateCustomer(Long id,CustomerEntity
updatedCustomer) {
        if(!validationCustomer(updatedCustomer)){
            return null;
        }
        if(customerRepository.findById(id).isPresent()){
            updatedCustomer.setId(id);
            updatedCustomer.setUpdated(Instant.now());

updatedCustomer.setCreated(customerRepository.findById(id).get().getCreated());

updatedCustomer.setIsActive(customerRepository.findById(id).get().getIsActive());
        }
        else
        {
            return null;
        }
        customerRepository.save(updatedCustomer);
        return mapCustomerEntityToCustomer(updatedCustomer);
```

```
    }

    public void deleteCustomer(Long id){
        Optional<CustomerEntity> customer = customerRepository.findById(id);
        if(customer.isPresent()){
            customer.get().setIsActive(false);
            customerRepository.save(customer.get());
        }
    }

    private Customer mapCustomerEntityToCustomer(CustomerEntity
customerEntity){
        if(customerEntity == null)
            return null;
        else
            return new Customer(
                customerEntity.getId(),
                customerEntity.getFullName(),
                customerEntity.getEmail(),
                customerEntity.getPhone());
    }

    @SneakyThrows
    private boolean validationCustomer(CustomerEntity customerEntity){
        if(!customerEntity.getEmail().matches(EMAIL))
            throw new ValidationException("Invalid email.");
        if(!customerRepository.findByEmail(customerEntity.getEmail()).isEmpty())
            throw new ValidationException("Email already used.");
        if(!customerEntity.getFullName().matches(FULL_NAME))
            throw new ValidationException("Invalid full name.");
```

```
        return true;
    }
}
package controller.filter;


import javax.servlet.*;
import java.io.IOException;

public class CharacterSetFilter implements Filter {
    @Override public void init(FilterConfig filterConfig){ }
    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException {
        servletRequest.setCharacterEncoding("UTF-8");
        servletResponse.setContentType("text/html; charset=UTF-8");
        servletResponse.setCharacterEncoding("UTF-8");
        filterChain.doFilter(servletRequest, servletResponse);
    }
}
package model.dao.Connection;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class BasicConnectionPool implements ConnectionPool {
```

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

    private String url;
    private String user;
    private String password;
    private List<Connection> connectionPool;
    private List<Connection> usedConnections = new ArrayList<>();
    private static int INITIAL_POOL_SIZE = 10;
    private BasicConnectionPool(String url, String user, String password,
List<Connection> pool) {
        this.url = url;
        this.user = user;
        this.password = password;
        connectionPool = pool;
    }

    public static BasicConnectionPool create(
            String url, String user,
            String password) throws SQLException {

        List<Connection> pool = new ArrayList<>(INITIAL_POOL_SIZE);
        for (int i = 0; i < INITIAL_POOL_SIZE; i++) {
            pool.add(createConnection(url, user, password));
        }
        return new BasicConnectionPool(url, user, password, pool);
    }
```

```
@Override
public Connection getConnection() {
    Connection connection = connectionPool
            .remove(connectionPool.size() - 1);
    usedConnections.add(connection);
    return connection;
}

@Override
public boolean releaseConnection(Connection connection) {
    connectionPool.add(connection);
    return usedConnections.remove(connection);
}

@Override
public String getUrl() {
    return url;
}

@Override
public String getUser() {
    return user;
}

@Override
public String getPassword() {
    return password;
}

private static Connection createConnection(
```

```
            String url, String user, String password)
            throws SQLException {
        return DriverManager.getConnection(url, user, password);
    }
}
package model.dao.Connection;

import java.sql.Connection;

public interface ConnectionPool {
    Connection getConnection();
    boolean releaseConnection(Connection connection);
    String getUrl();
    String getUser();
    String getPassword();
}
package model.dao.Connection;

import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Properties;

public class DataBaseConnectionManager{
    static Properties property = new Properties();
    private static ConnectionPool connectionPool;
    static {
        ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
        InputStream input = classLoader.getResourceAsStream("app.properties");
```

```
        try {
            property.load(input);
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            Class.forName("org.postgresql.Driver");
            connectionPool = BasicConnectionPool.create(property.getProperty("url"),
                                                property.getProperty("user"),
                                                property.getProperty("password"));
        } catch (SQLException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    public static Connection getConnection() {
        Connection connection;
        connection = connectionPool.getConnection();
        return connection;
    }
    public static void putConnection(Connection connection){
        connectionPool.releaseConnection(connection);
    }
    private DataBaseConnectionManager(){}
}
```

Додаток Г – Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка інформаційної системи для керування закладами громадського харчування

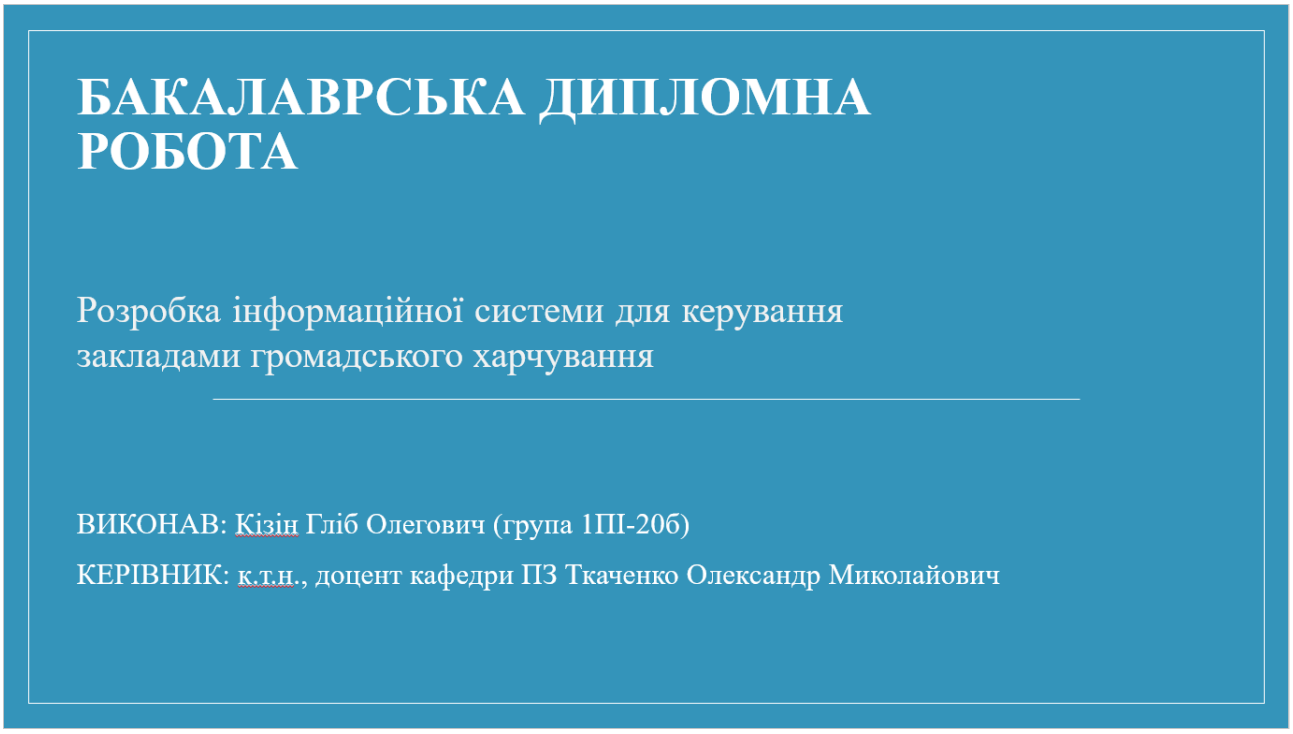

Рисунок Г.1 – Слайд презентації №1

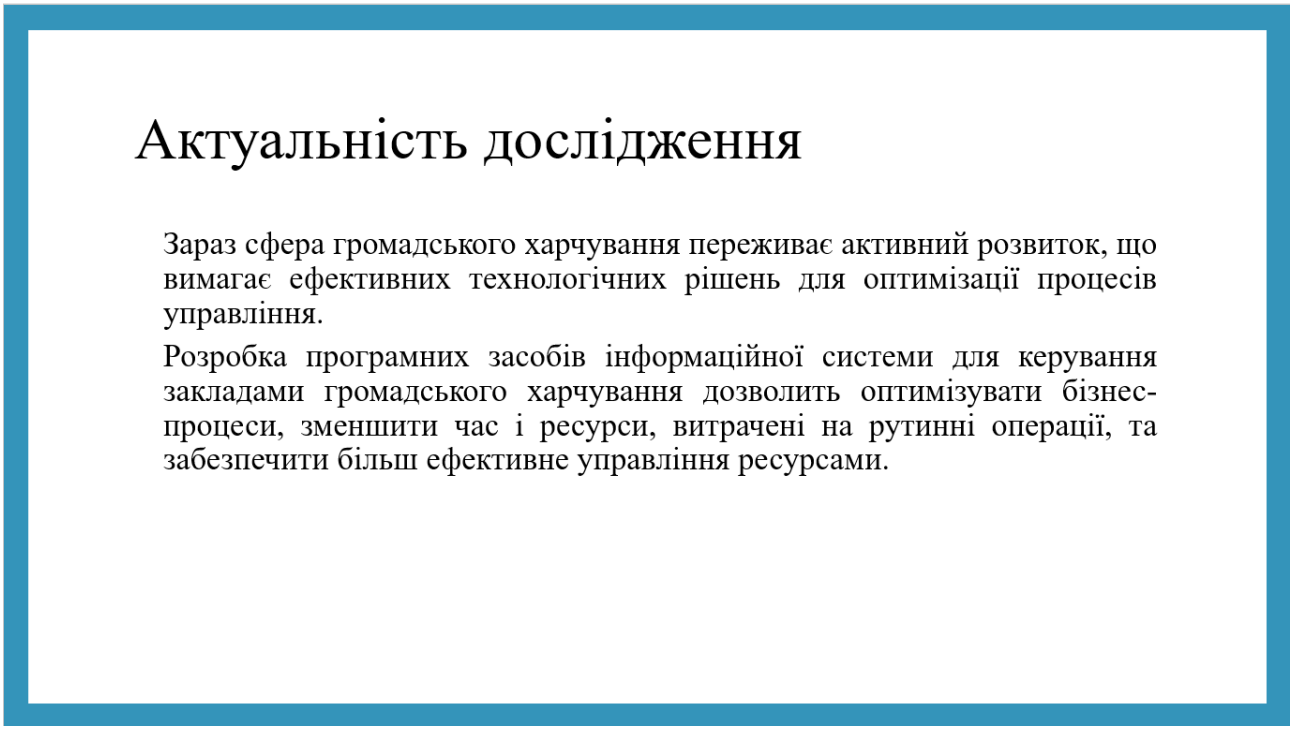

Рисунок Г.2 – Слайд презентації №2

Мета та завдання

Мета бакалаврської дипломної роботи полягає у і підвищення ефективності функціонування бізнес-процесів за рахунок розробки інформаційної системи для управління закладами громадського харчування.

Основними задачами дослідження є:

Розробка архітектури програмного забезпечення, включаючи вибір технологічних стеків, баз даних та архітектурних патернів, які найбільш відповідають потребам цільових користувачів.

Створення функціональних модулів системи, таких як управління замовленнями, облік запасів, обробка платежів тощо.

Реалізація програмного забезпечення на базі розробленої архітектури та функціональності.

Впровадження програмної системи та проведення тестування для перевірки її працездатності та відповідності вимогам.

Ці завдання спрямовані на створення ефективної та інтуїтивно зрозумілої інформаційної системи, яка відповідає потребам сучасних закладів громадського харчування і сприяє підвищенню їхньої продуктивності та конкурентоспроможності.

Рисунок Г.3 – Слайд презентації №3

Об’єкт, предмет та методи дослідження

Об’єкт дослідження – процес розробки інформаційної системи для керування закладами громадського харчування, стратегії щодо покращення роботи закладів громадського харчування.

Предмет дослідження - методи і засоби розробки програмних засобів для керування закладами громадського харчування мовою програмування Java.

Методи дослідження. У процесі досліджень використовувались теорія інформаційних систем для розробки загальної архітектури інформаційної системи; системний аналіз для визначення вимог до системи, розробки технічного завдання та моделювання бізнес-процесів та ідентифікації ключових елементів системи та взаємозв'язків між ними; теорія проектування баз даних для розробки структури баз даних; об'єктно-орієнтоване програмування для реалізації програмних компонентів системи на основі принципів об'єктно-орієнтованого підходу.

Рисунок Г.4 – Слайд презентації №4

Наукова новизна

1. Запропоновано архітектуру нейронної мережі, особливість якої полягає в можливості на основі проаналізованих даних продажів закладів громадського харчування, що дозволило формування прогнозів та порад для подальшого розвитку закладу та гнучке налаштування за потреб користувачів.

2. Отримала подальшого розвитку архітектура серверної частини у якій на від міну від існуючих, додано модуль спеціально навченої нейроної меріжі, що дозволило отримати можливість використовувати її для формування звітів з пропозиціями щодо покращення роботи закладів.

Рисунок Г.5 – Слайд презентації №5

Практична цінність отриманих результатів

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для управління закладами громадського харчування.

Розроблені програмні модулі дозволяють ефективно відслідковувати та контролювати роботу таких закладів, включаючи управління запасами, аналіз продажів та генерацію звітів.

Рисунок Г.6 – Слайд презентації №6

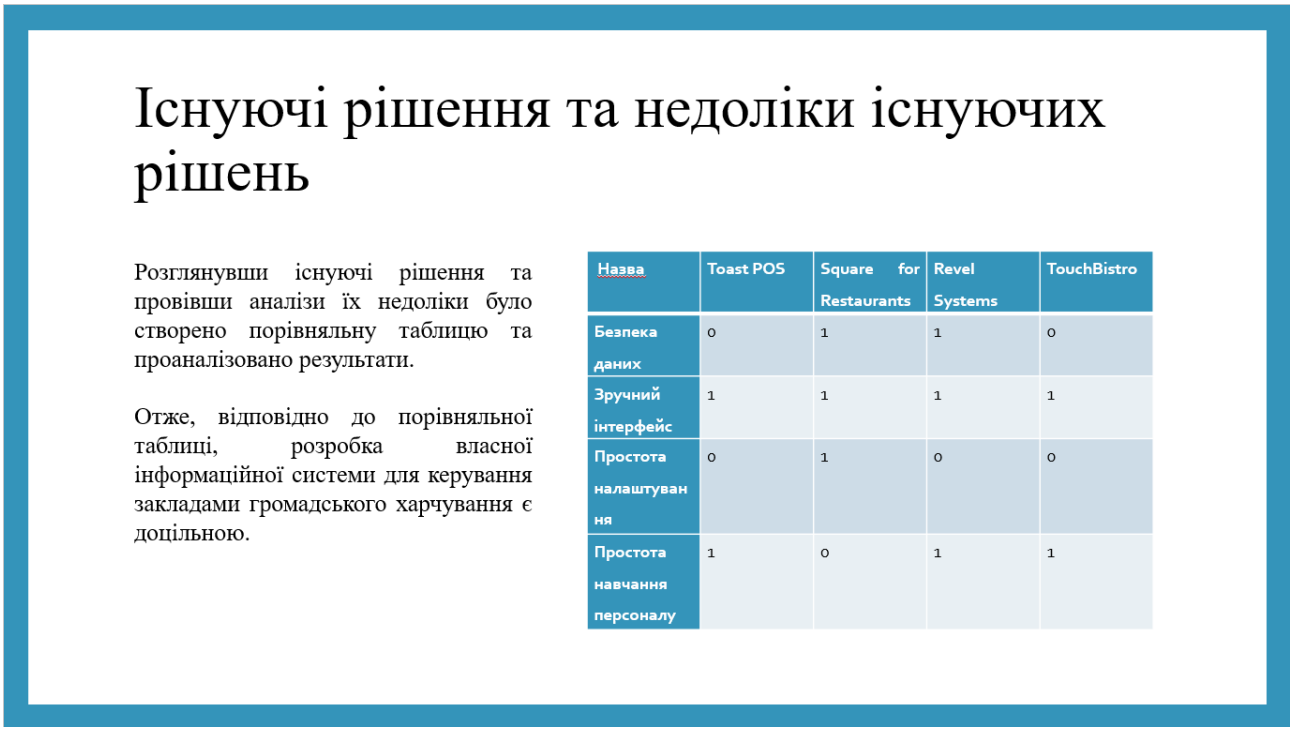

Рисунок Г.7 – Слайд презентації №7

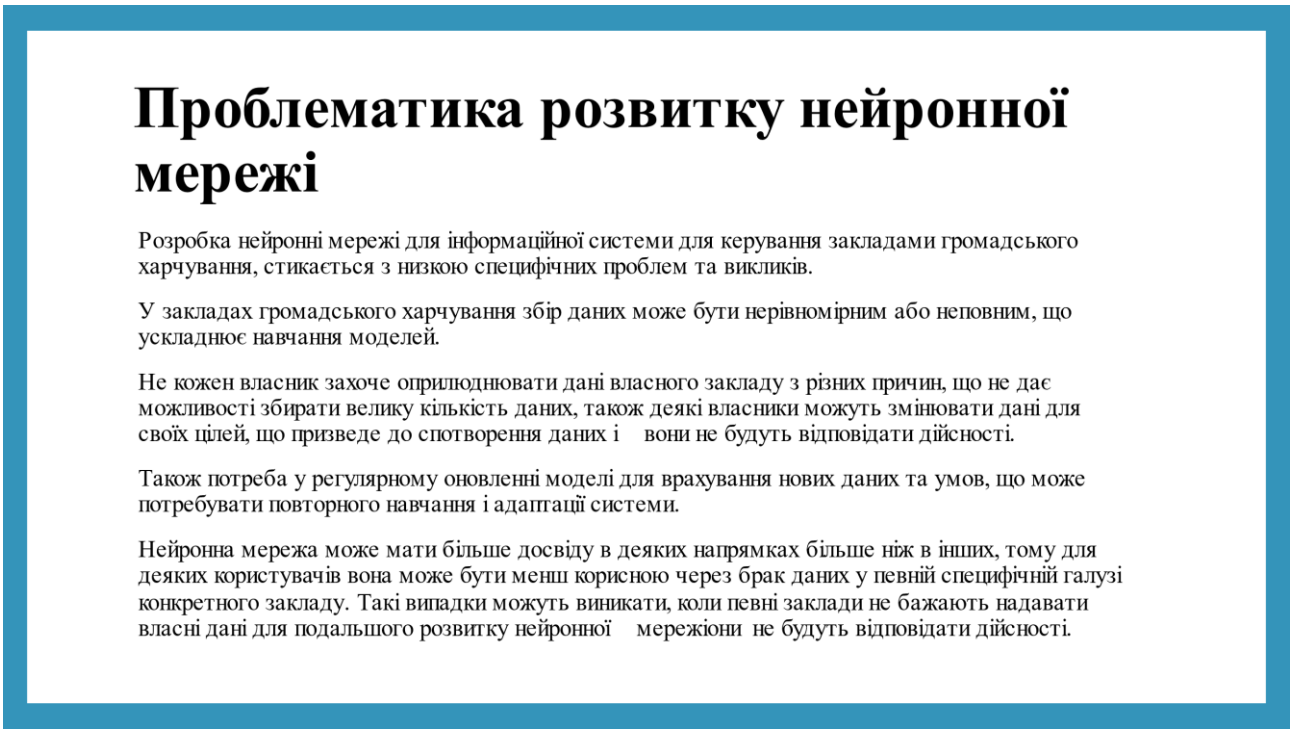

Рисунок Г.8 – Слайд презентації №8

Вдосконалена архітектура серверної частини

Для власної розробки було прийнято рішення модифікувати стандартну архітектуру серверної частини системи, додавши модуль з нейронною мережею.

Цей модуль буде інтегрований в серверну інфраструктуру та працюватиме, аналізуючи дані та надаючи рекомендації. Цей підхід забезпечить кілька ключових переваг.

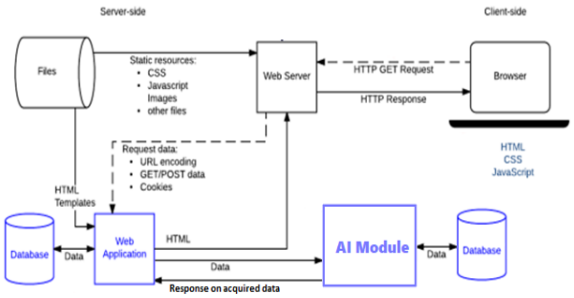

Рисунок Г.9 – Слайд презентації №9

Запропонова архітектура нейронної мережі

Для розробки власної нейронної мережі була обрана архітектура багатошаровий персептрон .
Зважаючи на всі дані, вхідний шар буде мати 6 нейронів, відповідно до характеристик, таких як обсяг продажів, витрати, кількість клієнтів, середній чек тощо. Перший прихований шар буде мати 64 нейрони, другий прихований шар буде мати 32 нейрони, і вихідний шар буде мати 1 нейрон для прогнозування.

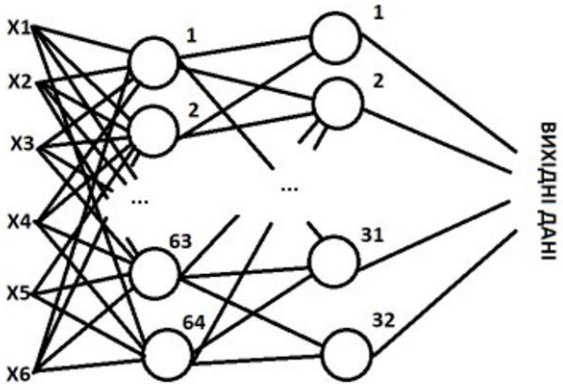

Рисунок Г.10 – Слайд презентації №10

Рисунок Г.11 – Слайд презентації №11

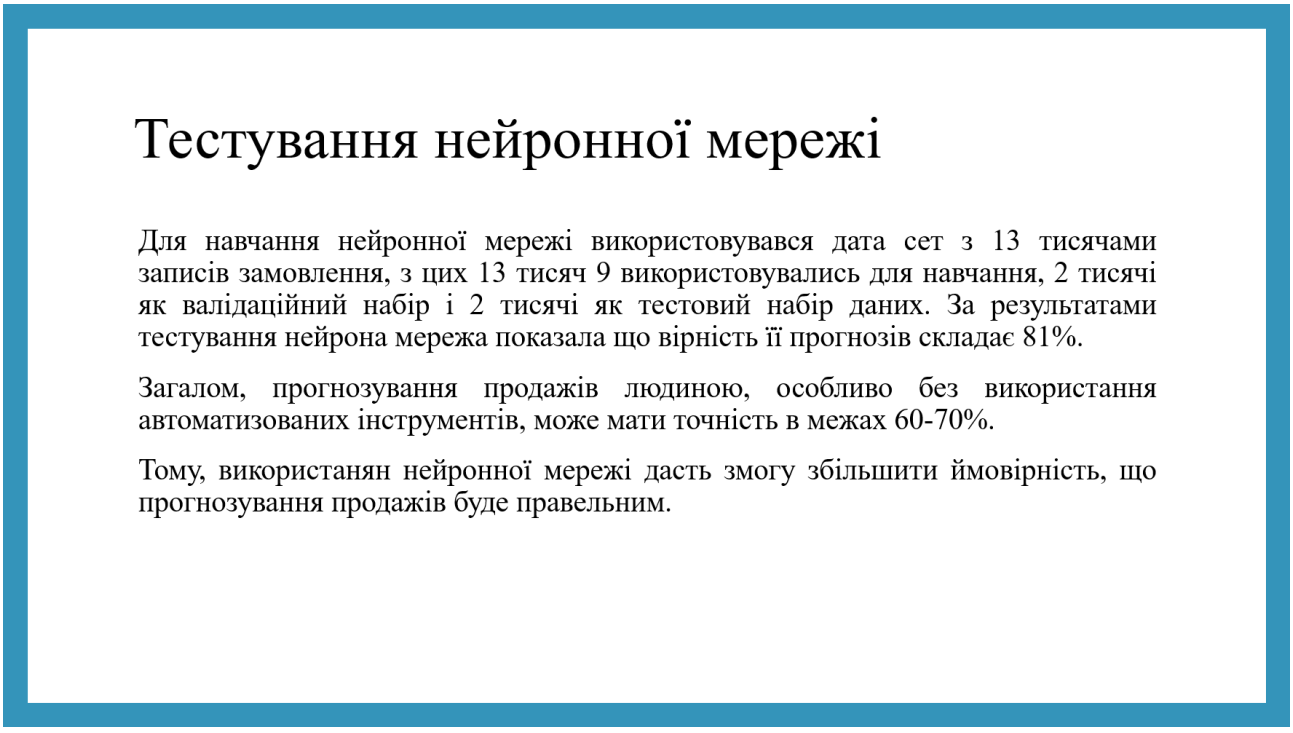

Рисунок Г.12 – Слайд презентації №12

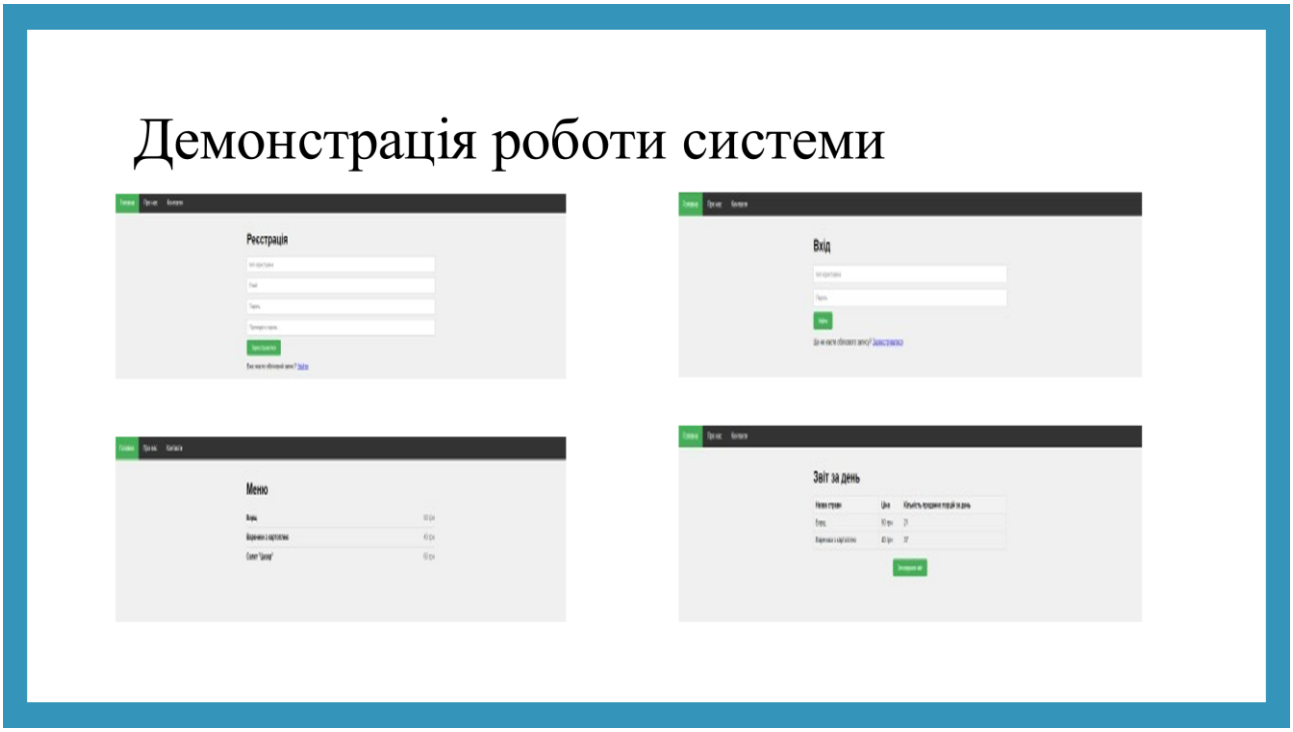

Рисунок Г.13 – Слайд презентації №13

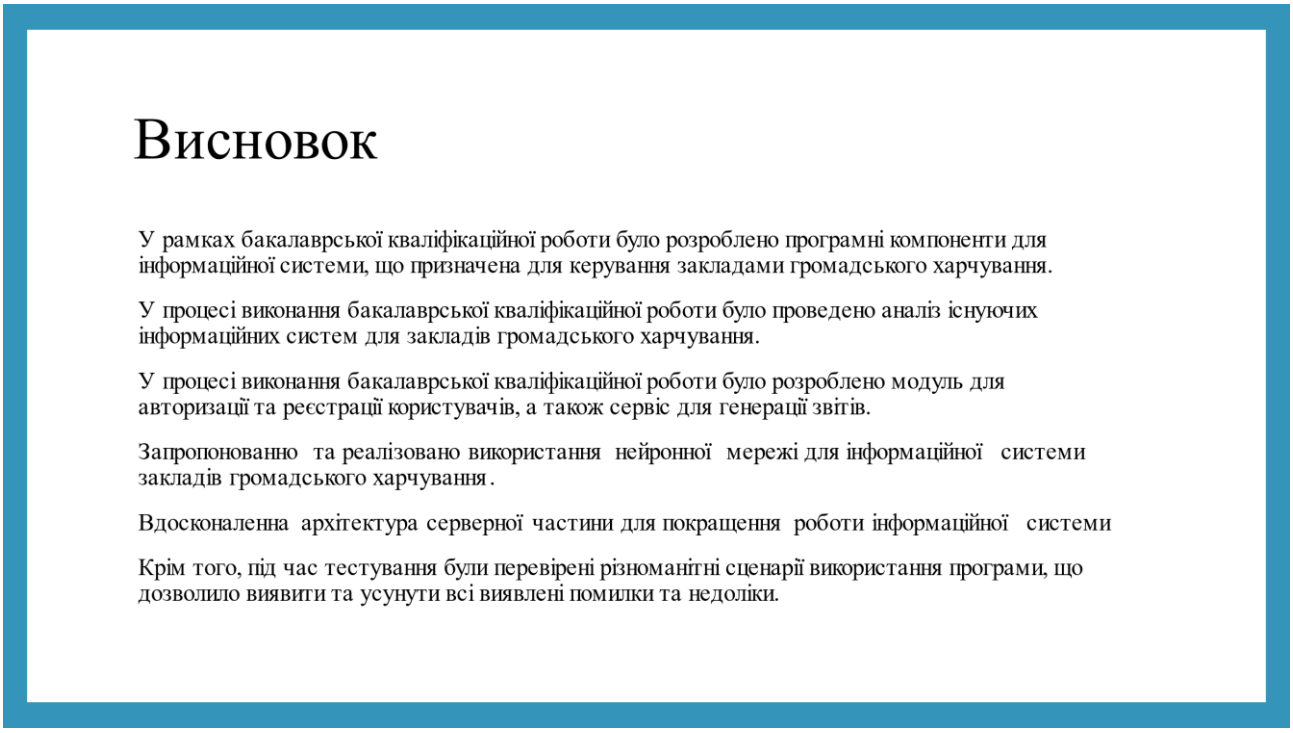

Рисунок Г.14 – Слайд презентації №14

Рисунок Г.15 – Слайд презентації №15