СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної випускної роботи

освітній ступінь бакалавр

спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

на тему “Розробка веб-додатку для закладів харчування на базі Flask та PostgreSQL”

Виконав: студент групи ІПЗ-20д _____ А.П. Полєнов
(підпис) (ініціали і прізвище)

Керівник _____ В.О. Лифар
(підпис) (ініціали і прізвище)

Завідувач кафедри _____ О.І. Захожай
(підпис) (ініціали і прізвище)

Рецензент Д.В. Ратов

Київ- 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітній ступінь бакалавр

спеціальність 121 „Інженерія програмного забезпечення”
(шифр і назва спеціальності)

спеціалізація „Інженерія програмного забезпечення”
(назва спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____Захожай О.І._
“____” _____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ ВИПУСКНУ РОБОТУ СТУДЕНТУ

_____Полєнов Андрій Петрович_____
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка веб-додатку для закладів харчування на базі Flask та PostgreSQL_____
_____

Керівник роботи____доцент д.т.н. Лифар Володимир Олексійович_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджений наказом університету від “_06_”_травня_2024 року №171/15.15-С

2. Строк подання студентом роботи__14.06.24р._____

3. Вихідні дані до роботи___Об’єктом даної роботи є процес розробки веб-додатку._____
_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)__Вступ. Аналітичний огляд, з висвітленням наступних питань: аналіз потреб цільової аудиторії, огляд функціональних вимог, технічний аналіз. Основна частина, в якої висвітлити: збір вимог та проектування, етап реалізації проекту . Висновки. Список використаних джерел._____

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)
_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання__30.03.24р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1	Одержання завдання на виконання роботи	30.03.24	виконано
2	Укладання і погодження з керівником плану і етапів виконання роботи	06.04.24	виконано
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	13.04.24	виконано
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	20.04.24	виконано
5	Укладання та тестування програмного продукту	27.04.24	виконано
6	Укладання, оформлення та погодження пояснювальної записки з керівником	04.05.24	виконано
7	Здача готової пояснювальної записки на кафедру	14.06.24	виконано
8	Укладання доповіді і презентації	16.06.24	виконано

Студент _____ __А.П. Полєнов___
(підпис) (ініціали і прізвище)

Керівник роботи _____ ___В.О. Лифар____
(підпис) (ініціали і прізвище)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІПЗ-20д Полєнов А.П.

Науковий керівник

Доцент, к.т.н. _____ Лифар В.О.

Оцінка наукового керівника: _____

Рецензент Ратов Денис Валентинович к.т.н., доцент кафедри ІТП СНУ ім.В.Даля

ПІБ, місто роботи, посада

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:

_____

Голова ЕК

Професор кафедри ІТП д.т.н. _____ Меняйленко О.С.

підпис

РЕФЕРАТ

Робота містить: 48 сторінок основного тексту, 44 сторінки додатків, 10 рисунків, 10 використаних джерел.

Метою випускної кваліфікаційної роботи є процес розробки веб-додатку для закладів харчування з використанням фреймворка Flask та фронтенду на базі Bootstrap. Дослідження зосереджувалося на розробці повноцінної системи, яка могла б надати користувачам можливість здійснювати замовлення їжі та адміністраторам - управляти меню та замовленнями.

Було проаналізовано багато сайтів-аналогів, їх принцип роботи, перелік наданих можливостей і кращі рішення взаємодії з користувачами. Багато часу було приділено технічній частині, зокрема підбору продуктивного фреймворка Flask, і бібліотек для аутентифікації та роботи з формами, а також бази даних PostgreSQL, яка найкраще підходить для даного проекту.

В результаті виконаної роботи було реалізовано веб-додаток, що дозволяє користувачам реєструватися, входити в систему, переглядати та замовляти страви, а адміністраторам – управляти замовленнями та меню. Веб-додаток є безпечним завдяки використанню Flask-Bcrypt для хешування паролів та забезпечує ефективну взаємодію з базою даних через Flask-SQLAlchemy.

Система може бути легко масштабована для подальшої розробки завдяки використанню шаблону MVC та міграціям бази даних через Fkask-Migrate. Реалізація системи відповідає всім вимогам технічного завдання.

ЗМІСТ

ВСТУП

Останнім часом спостерігається вражаюче зростання популярності веб-додатків для закладів харчування, що виконують функції агрегаторів. Ці додатки надають зручний доступ до різноманітних послуг, таких як замовлення їжі з різних ресторанів, кафе, або навіть доставка продуктів. Їх головна мета полягає в автоматизації процесу замовлення та вибору необхідних страв, щоб забезпечити максимальну зручність та швидкість обслуговування клієнтів.

Сучасна концепція гастрономічного бізнесу визнається як інформаційно-насичена сфера. Власники закладів харчування усвідомлюють необхідність використання новітніх технологій для оптимізації роботи та покращення обслуговування клієнтів. Створення веб-додатку для закладів харчування спрямоване на забезпечення максимальної зручності та ефективності замовлення їжі, а також підвищення ефективності управління закладом.

Метою даного веб-додатку є створення програмної системи для організації онлайн-замовлень їжі у закладах харчування. Для досягнення цієї мети необхідно розв'язати такі завдання:

1. Детально проаналізувати особливості організації роботи закладів харчування в умовах інтернет-середовища, враховуючи специфіку їхньої діяльності та потреб клієнтів.
2. Провести глибокий аналіз існуючих систем автоматизації для закладів харчування, виявити переваги та недоліки кожної з них.
3. Визначити конкретні потреби та характеристики цільового закладу харчування для розробки програмної системи, яка оптимально відповідає їхнім потребам.
4. Поставити перед собою чіткі завдання, що вирішуються веб-додатком, визначивши пріоритетні напрямки його розвитку.

5. Вибрати та описати методологію проектування, яка найкращим чином підходить для розробки програмного забезпечення для закладів харчування.
6. Розробити детальну схему бази даних та структуру веб-додатка, включаючи всі необхідні функціональні модулі та взаємозв'язки між ними.
7. Створити імплементацію клієнтської частини веб-додатка, забезпечивши максимальну зручність та інтуїтивність інтерфейсу для користувачів.
8. Обгрунтувати вибір інструментальних засобів для розробки, забезпечивши оптимальну швидкість та ефективність роботи веб-додатка.
9. Реалізувати веб-додаток відповідно до встановлених завдань та технічних вимог.
10. Описати інтерфейс користувача, зосереджуючись на його зручності та функціональності, та виконати всі необхідні дії для тестування додатка перед випуском його в експлуатацію.

Головною метою при реалізації цього проекту є поліпшення ефективності управління закладом харчування та забезпечення зручності та задоволення потреб клієнтів. Такий веб-додаток повинен забезпечити оптимальні умови для замовлення та отримання їжі з мінімальними витратами часу та зусиль як для клієнтів, так і для власників закладу.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Аналіз потреб цільової аудиторії

Перед розробкою такого веб-додатку важливо провести аналіз потреб цільової аудиторії, щоб зрозуміти їхні вимоги та очікування. У цій статті ми розглянемо потреби власників закладів харчування та їх клієнтів, а також визначимо ключові функції, які має включати веб-додаток для задоволення цих потреб.

Власники закладів харчування – це люди, які керують ресторанами, кафе, пекарнями та іншими закладами громадського харчування. Для них ефективне управління бізнесом та задоволення потреб клієнтів є ключовими аспектами. Ось деякі з основних потреб цієї аудиторії:

Власники закладів харчування потребують зручного інструменту для управління меню. Це включає можливість додавати, редагувати та видаляти страви, встановлювати ціни та описи. Для ефективного ведення бізнесу власникам необхідно відстежувати та керувати замовленнями. Це означає можливість приймати замовлення, відправляти їх до кухні або бару, відстежувати статус замовлення та опрацьовувати оплату.

Контроль за запасами є важливим аспектом управління закладом харчування. Власники потребують зручного інструменту для відстеження рівня запасів, отримання сповіщень про необхідність поповнення та замовлення нових продуктів.

Отримання даних щодо продажів, популярності страв та інших показників дозволяє власникам зробити обґрунтовані рішення для покращення бізнесу. Тому важливо мати можливість генерувати звіти та аналізувати дані. Забезпечення зручного способу комунікації з клієнтами є ще однією важливою потребою. Це може включати функції замовлення онлайн, обмін

повідомленнями щодо замовлень або запитань, а також можливість надсилання спеціальних пропозицій або акцій.

Окрім потреб власників, важливо також зрозуміти очікування та вимоги клієнтів закладів харчування. Це група людей, які зазвичай користуються послугами ресторанів, кафе та інших закладів. Деякі з їхніх ключових потреб включають:

Клієнти цінують можливість швидко та зручно замовляти їжу онлайн. Це означає простий та інтуїтивно зрозумілий інтерфейс додатку, можливість перегляду меню, швидке оформлення замовлення та оплата. Для багатьох клієнтів важливо, щоб їх замовлення було оброблено якнайшвидше. Вони не хочуть чекати довго на свою їжу або напої, тому швидкість обслуговування важлива.

Клієнти також цінують можливість попереднього ознайомлення з меню та перегляду інформації про страви, включаючи опис, фотографії та ціни. Це дозволяє їм зробити обдуманий вибір та підготуватися до замовлення.

Оплата є ще одним важливим аспектом для клієнтів. Вони хочуть мати можливість використовувати різні методи оплати, такі як кредитні картки, мобільні платежі або онлайн-платежі.

Клієнти також можуть мати запитання, пропозиції або скарги, тому важливо мати зручну систему зворотного зв'язку, через яку вони можуть звертатися до адміністрації закладу.

1.2 Огляд функціональних вимог

Управління меню є однією з найважливіших функцій для власників закладу харчування. Додаток повинен надавати можливість додавати нові страви, редагувати та видаляти існуючі, встановлювати ціни, описи та

фотографії. Важливо, щоб ця функція була простою та інтуїтивно зрозумілою для власників, щоб вони могли швидко оновлювати своє меню.

Функція обробки замовлень дозволяє власникам ефективно керувати потоком замовлень. Це включає отримання нових замовлень в реальному часі, відправлення їх до кухні або бару для приготування, відстеження статусу замовлення та опрацювання оплати. Потужна система обробки замовлень дозволяє підтримувати високу швидкість обслуговування та задовольняти потреби клієнтів.

Для забезпечення ефективного ведення бізнесу, власникам потрібний доступ до даних про своїх клієнтів та їхні операції. Це може включати інформацію про історію замовлень, уподобання та дані про оплату. Функції керування клієнтськими даними повинні забезпечити безпеку та конфіденційність даних, а також можливість аналізу цих даних для покращення обслуговування.

Функція резервування столиків дозволяє клієнтам забронювати місце у закладі заздалегідь. Це особливо важливо для ресторанів та кафе з великим потоком клієнтів. Додаток повинен надати можливість перегляду доступних часів та дат, вибору кількості місць та підтвердження бронювання.

Для клієнтів, зручність замовлення їжі онлайн є важливою функцією. Додаток повинен надати зручний інтерфейс для перегляду меню, вибору страв, додавання до кошика та оформлення замовлення. Крім того, можливість вибору способу оплати та доставки допомагає забезпечити максимальний комфорт для клієнтів.

Функціональні вимоги до веб-додатку для закладу харчування включають широкий спектр функцій, які забезпечують як ефективне управління бізнесом, так і зручне обслуговування клієнтів. Важливо, щоб додаток був інтуїтивно зрозумілим для використання як для власників, так і для клієнтів, а також забезпечував безпеку та конфіденційність даних. Ретельне врахування цих

вимог допоможе створити успішний та конкурентоспроможний веб-додаток для закладу харчування.

1.3 Технічний аналіз

Розробка веб-додатку на базі Django та використання Bootstrap для фронтенду є перспективним підходом, оскільки обидва ці інструменти мають свої унікальні переваги та забезпечують ефективну розробку та високу якість продукту. Давайте розглянемо основні переваги і можливості кожного з цих інструментів і їх взаємодію.

Django - це високорівневий Python-фреймворк, спеціально розроблений для швидкої розробки веб-додатків. Він має численні переваги:

1. Швидкий розвиток: Django надає ряд вбудованих функцій та структур, що дозволяють розробникам швидко створювати функціонал веб-додатків без необхідності повторного винаходу колеса.
2. Ефективне управління базами даних: Django має вбудовану ORM (об'єктно-реляційне відображення), що спрощує взаємодію з базами даних. Це дозволяє розробникам працювати з базами даних за допомогою Python-коду, що робить розробку більш зрозумілою та ефективною.
3. Безпека: Django має вбудовані механізми захисту від таких загроз, як SQL-ін'єкції, Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF), що дозволяє створювати безпечні веб-додатки з меншим ризиком для даних користувачів.

Bootstrap - це один з найпопулярніших фреймворків для розробки веб-додатків та сайтів. Деякі з його переваг включають:

1. Респонсивний дизайн: Bootstrap надає набір готових компонентів та стилів, які дозволяють легко створювати респонсивний та адаптивний дизайн, що працює на різних пристроях та розмірах екрану.
2. Швидкий розвиток: Bootstrap містить готові CSS-класи та компоненти, які значно прискорюють розробку веб-інтерфейсу. Завдяки цьому, розробники можуть швидко створювати стильні та сучасні дизайни без великого обсягу власного CSS-коду.
3. Спільнота та документація: Bootstrap має велику та активну спільноту користувачів, а також докладну документацію, що дозволяє розробникам швидко знаходити відповіді на свої питання та вирішувати проблеми.

Використання Django для бекенду та Bootstrap для фронтенду забезпечує ефективну та продуктивну розробку веб-додатків. Django забезпечує потужні можливості для обробки бізнес-логіки, взаємодії з базою даних та безпеки, тоді як Bootstrap допомагає створювати естетичний та респонсивний інтерфейс для користувачів.

Зазвичай взаємодія Django та Bootstrap відбувається шляхом використання шаблонів Django для генерації HTML-коду, який потім стилізується та оформлюється за допомогою Bootstrap. Це дозволяє розробникам створювати динамічні та привабливі веб-додатки, забезпечуючи зручну та злагоджену взаємодію з користувачем.

Розробка веб-додатку на базі Django з використанням Bootstrap для фронтенду є ефективним та перспективним підходом, що дозволяє швидко створювати потужні та стильні веб-додатки. Django забезпечує швидку розробку та ефективне управління базами даних, тоді як Bootstrap допомагає створювати респонсивний та привабливий дизайн. Використання цих інструментів разом дозволяє розробникам створювати високоякісні та конкурентоспроможні веб-додатки для різних потреб та галузей.

Цей аналітичний огляд не лише надає важливу інформацію про контекст та потреби, але й створює базу для подальшого розвитку та вдосконалення веб-додатку для закладу харчування. Розуміння цих ключових аспектів дозволяє зорієнтуватися на головні напрямки розвитку та визначити необхідні функціональність, яка відповідатиме вимогам цільової аудиторії.

На основі аналізу потреб цільової аудиторії можна визначити ключові функції, які має мати веб-додаток. Наприклад, можливість управління меню, обробка замовлень, керування клієнтськими даними та оплатою, а також функції резервування столиків та зручний інтерфейс для клієнтів для замовлення онлайн. Ці функції є основою для подальшого технічного проектування та розробки веб-додатку.

Розробка веб-додатку на базі Django та використання Bootstrap для фронтенду також має великий потенціал для успішного створення продукту. Django забезпечує швидкий розвиток, ефективне управління базами даних та безпеку, в той час як Bootstrap допомагає створювати естетичний та респонсивний дизайн. Це можливості, які важливі для створення зручного та привабливого веб-додатку, що задовольнить потреби користувачів.

У цілому, цей аналітичний огляд створює фундаментальне розуміння контексту, потреб та можливостей, що є важливими для подальшого успішного розвитку та впровадження веб-додатку для закладу харчування. Він надає важливі вказівки для технічного проектування та впровадження, що допоможе створити конкурентоспроможний та привабливий продукт на ринку.

2 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ ЗАКЛАДІВ ХАРЧУВАННЯ

2.1 Обгрунтування актуальності теми

В сучасному світі, гостинна галузь харчування відіграє ключову роль у задоволенні потреб споживачів і впливає на їхні враження та задоволення від переживання кулінарних насолод. Однак, зростаюча складність і вимоги споживачів створюють різноманітні виклики для галузі харчування. Технології та інновації стають важливими інструментами для розв'язання цих проблем і підвищення якості обслуговування. В цьому контексті, розробка веб-додатків для закладів харчування стає актуальною та значущою темою для дослідження та впровадження. Далі у тексті ми розглянемо ряд факторів, які підкреслюють актуальність даної теми.

Зростаюча конкуренція в галузі харчування ставить підвищені вимоги до якості обслуговування. Веб-додатки можуть допомогти закладам харчування не лише привернути увагу клієнтів, але і покращити їхні враження від обслуговування. Інтерактивність, персоналізація та зручність у користуванні стають ключовими факторами у виборі клієнтів. Веб-додатки для закладів харчування можуть значно полегшити процеси управління. Вони надають можливість вести облік замовлень, контролювати запаси, керувати персоналом та аналізувати даними для прийняття стратегічних рішень. Це дозволяє знижувати витрати та оптимізувати бізнес-процеси, що є важливим аспектом в умовах зростаючої конкуренції.

Веб-додатки можуть стати потужним інструментом для підвищення лояльності клієнтів. Програми лояльності, персоналізовані пропозиції та знижки, повідомлення про акції та події - усе це можна ефективно реалізувати

через веб-додаток, що сприяє покращенню відносин з клієнтами та збільшенню їхньої активності. Веб-додатки дозволяють закладам харчування розширити свою аудиторію і відкрити нові ринки. Завдяки онлайн-присутності, ресторани, кафе та інші заклади можуть привернути увагу не лише місцевих мешканців, але й туристів, людей з інших міст або країн. Це відкриває нові можливості для розвитку бізнесу та збільшення прибутків.

Зростання популярності онлайн-сервісів у всіх сферах життя, включаючи галузь харчування, є фундаментальним фактором, що підкреслює актуальність розробки веб-додатку для закладів харчування. Спостерігається тенденція до все більшого використання інтернет-технологій та мобільних пристроїв у процесі замовлення їжі. Це обумовлено зростанням швидкості й доступності Інтернету, а також зручністю використання мобільних додатків. Клієнти, надзвичайно цінуючи свій час та комфорт, все частіше обирають замовлення їжі онлайн, замість традиційних способів, таких як відвідування ресторану або телефонний зв'язок.

Розробка веб-додатку для закладів харчування стає ключовою для їх успішної конкурентоспроможності та відповіді на змінні вимоги сучасного споживача. Ці додатки не лише надають зручний канал для замовлення їжі, а й виконують ряд інших корисних функцій. Наприклад, вони можуть включати в себе можливість перегляду меню з подробним описом страв та цін, можливість вибору додаткових опцій чи налаштувань (таких як додаткові інгредієнти, розмір порції, рівень готування тощо), а також можливість відстежування статусу замовлення. Ці функції дозволяють клієнтам зручно і ефективно вибирати та налаштовувати свої замовлення, що відповідає сучасним вимогам до обслуговування.

Ще одним фактором, що підкреслює актуальність розробки веб-додатку для закладів харчування, є необхідність забезпечення зручності та швидкості обслуговування. Сучасний споживач дуже цінує свій час і очікує, що процес замовлення та отримання їжі буде якомога швидшим і безпечнішим. Веб-додатки дозволяють клієнтам замовляти їжу в будь-який зручний для них час,

не зважаючи на робочий графік або місцезнаходження. Це особливо актуально у сучасному ритмі життя, коли люди можуть бути зайняті цілий день і не мати можливості відвідати ресторан чи кафе особисто. Замовлення через веб-додаток дозволяє їм зекономити час, який витрачався б на поїздку до закладу та очікування на обслуговування, і замість цього сконцентруватися на інших справах.

Пандемія COVID-19 суттєво змінила умови функціонування закладів громадського харчування. Обмеження на відвідування ресторанів та кафе спонукають заклади шукати нові способи привернення клієнтів та забезпечення послуг на високому рівні. Розробка веб-додатку, що дозволяє клієнтам замовляти їжу онлайн, може стати ключовим рішенням для забезпечення безпеки та зручності як для клієнтів, так і для закладів харчування.

Галузь харчового обслуговування характеризується високим рівнем конкуренції, що вимагає від закладів постійного вдосконалення та розробки нових стратегій залучення клієнтів. Веб-додаток, який пропонує зручне та ефективне замовлення їжі, може стати суттєвою конкурентною перевагою для закладів харчування, дозволяючи їм привертати нових клієнтів та зберігати вже існуючу базу.

Подібно до розробки веб-додатків у всіх сферах діяльності, створення програмного забезпечення для закладів харчування відкриває широкі можливості для персоналізації та аналітики. Це означає, що ресторани, кафе та інші заклади можуть впроваджувати різноманітні функції та фічі, які забезпечать індивідуалізований підхід до кожного клієнта.

З одного боку, веб-додатки можуть збирати та аналізувати дані про споживацькі преференції. Це може включати інформацію про типи страв, які клієнти замовляють найчастіше, їхні уподобання до певних інгредієнтів чи категорій кухні, а також інші деталі, що допомагають розуміти вподобання кожного клієнта окремо.

На основі цих даних веб-додаток може пропонувати персоналізовані рекомендації щодо страв та напоїв. Наприклад, якщо клієнт зазвичай замовляє страви певної кухні, система може рекомендувати йому нові страви з цієї кухні або навіть пропонувати індивідуально підготовлені пропозиції, адаптовані до його уподобань.

З іншого боку, аналітика дозволяє закладам харчування збирати цінну інформацію про своїх клієнтів і їхні замовлення. Це може включати дані про частоту замовлень, середній чек, популярність окремих страв у певні дні тижня або часи доби. Аналіз цих даних дозволяє закладам ліпше розуміти потреби своїх клієнтів та адаптувати свою стратегію обслуговування для максимальної ефективності і задоволення клієнтів.

За допомогою цієї інформації, заклади можуть приймати рішення щодо асортименту страв, ціноутворення, акцій та промоцій. Наприклад, якщо аналітика показує, що певна страва має низьку популярність, заклад може вирішити вилучити її з меню або надати знижку на її замовлення, щоб стимулювати продажі.

Крім того, дані про замовлення можуть допомогти в плануванні запасів та оптимізації робочих процесів. Наприклад, якщо аналітика показує, що певна страва часто замовляється в певний час дня, заклад може підготувати відповідні інгредієнти заздалегідь та зменшити час очікування для клієнтів.

Таким чином, персоналізація та аналітика, які надає розроблений веб-додаток, не лише підвищують задоволення клієнтів від обслуговування, але і дозволяють закладам харчування оптимізувати свою діяльність та збільшити ефективність.

Стрімкий розвиток технологій, які надають можливості у сфері веб-розробки та програмування, відкриває безліч нових перспектив для створення потужних та ефективних веб-додатків. На сьогоднішній день, інструменти розробки, такі як Flask та Bootstrap, дозволяють максимально використовувати переваги останніх технологічних досягнень.

Flask - це мікрофреймворк для Python, який дозволяє розробникам створювати веб-додатки швидко та ефективно. Його легка вага та простота використання роблять його ідеальним інструментом для створення веб-додатків, зокрема для ресторанів та інших закладів харчування. Використання Flask дозволяє розробникам швидко реалізувати основні функції, такі як обробка замовлень та управління меню.

Bootstrap, у свою чергу, є одним з найпопулярніших фронтенд-фреймворків, який забезпечує швидкий та простий спосіб створення сучасних та адаптивних веб-інтерфейсів. Використання Bootstrap дозволяє розробникам ефективно працювати над дизайном та розміщенням елементів веб-сторінок, забезпечуючи їм адаптивність для різних пристроїв, включаючи комп'ютери, планшети та смартфони.

Завдяки поєднанню Flask та Bootstrap, розробники можуть створювати високоякісні та функціональні веб-додатки для закладів харчування з урахуванням найновіших технологічних тенденцій. Це дозволяє створити інтерактивні та зручні інтерфейси для користувачів, які можуть замовляти їжу та отримувати доступ до різноманітної інформації про меню, акції та спеціальні пропозиції.

Стрімкий розвиток технологій у сфері веб-розробки також означає постійне оновлення та вдосконалення інструментів, що надає нові можливості для створення веб-додатків, які є більш ефективними, швидкими та безпечними. Наприклад, поява нових версій мов програмування, фреймворків та бібліотек дозволяє розробникам вдосконалювати функціональність своїх додатків та забезпечувати їх збільшеною продуктивністю та стабільністю.

У підсумку, розробка веб-додатку для закладів харчування є актуальною та перспективною темою, що відповідає вимогам сучасного ринку та потребам споживачів. Впровадження такого додатку може значно поліпшити якість обслуговування, зробити процес замовлення їжі більш зручним та ефективним,

а також забезпечити конкурентні переваги для закладів харчування у сучасному цифровому світі.

2.2 Мета та завдання дослідження

Мета даної дипломної роботи полягає у розробці веб-додатку для закладів харчування з використанням Flask та Bootstrap. Основною метою є створення функціонального, ефективного та інноваційного інструменту, який сприятиме оптимізації процесів управління замовленнями, обслуговуванням клієнтів та управлінням бізнесом для закладів громадського харчування.

- Завдання дослідження: Аналіз потреб та вимог клієнтів та закладів харчування: Перший етап дослідження передбачає збір та аналіз вимог різних сегментів клієнтів закладів харчування. Важливо визначити потреби різних груп клієнтів, щоб зрозуміти, які функціональні та нефункціональні вимоги повинна враховувати система.
- Огляд існуючих систем управління замовленнями та аналіз ринку: Для успішної розробки веб-додатку важливо провести огляд існуючих систем управління замовленнями в закладах харчування. Це дозволить зрозуміти сучасні тенденції, визначити недоліки та переваги існуючих рішень та знайти можливості для покращення.
- Проектування архітектури та інтерфейсу користувача: На цьому етапі виконується розробка архітектури системи та проектування інтерфейсу користувача. Важливо ретельно продумати структуру бази даних для зберігання інформації про меню, замовлення, користувачів тощо. Також потрібно розробити зручний і привабливий інтерфейс користувача, щоб забезпечити комфортне використання системи.
- Розробка серверної та клієнтської частин: На цьому етапі виконується розробка серверної та клієнтської частин веб-додатку. Для серверної частини використовується фреймворк Flask, а для клієнтської частини

- Bootstrap. Важливо забезпечити стабільну та безпечну роботу системи, а також зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Ці завдання дослідження дозволять систематично та ефективно реалізувати цілі проекту та досягти мети розробки веб-додатку для закладів харчування з використанням Flask та Bootstrap. Кожен етап є важливим для успішної реалізації проекту та забезпечення задоволення від користування системою як з боку клієнтів, так і з боку закладів харчування.

2.3 Об'єкт та предмет дослідження

Дослідження об'єкта та предмета є важливим етапом в науковій роботі. Об'єкт дослідження - це та сфера або область, яка вивчається, а предмет - це конкретний аспект цієї області, який є об'єктом уваги дослідника. У нашому випадку, об'єктом є заклади громадського харчування, а предметом - розробка веб-додатку для їхньої оптимізації та покращення сервісу.

Об'єкт дослідження: Заклади громадського харчування

Заклади громадського харчування охоплюють широкий спектр закладів, від ресторанів і кафе до швидкого харчування та кав'ярень. Ці заклади надають послуги готування та подачі їжі споживачам у публічних місцях, таких як ресторани, кафе, бари, закусочні тощо. Вони є важливою частиною громадського життя та культури, і забезпечують не лише харчування, а й соціальні функції, такі як зустрічі з друзями, бізнес-зустрічі тощо.

Об'єкт дослідження включає в себе різні типи закладів харчування, що працюють у різних сегментах ринку та мають відмінні характеристики. Це можуть бути елітні ресторани зі стравами вищого класу, затишні кафе з домашньою атмосферою, фаст-фуд заклади зі швидким обслуговуванням,

кав'ярні зі спеціальною кавою та інші. Такий широкий спектр закладів потребує різноманітних підходів до оптимізації та покращення їхньої роботи.

Предмет дослідження: Розробка веб-додатку для оптимізації закладів харчування

Предмет дослідження - це конкретний аспект об'єкта, який є об'єктом уваги дослідника. У нашому випадку, предметом дослідження є розробка веб-додатку, спрямованого на покращення та оптимізацію роботи закладів громадського харчування.

Аналіз об'єкта та предмета дослідження:

Аналізуючи об'єкт дослідження, ми розглядаємо різноманітні аспекти функціонування закладів харчування, такі як їхні структури, типи послуг, цільова аудиторія, конкурентні переваги та виклики, з якими вони зіштовхуються.

Щодо предмета дослідження, розробка веб-додатку для оптимізації закладів харчування, ми зосереджуємося на розробці інструменту, який допоможе автоматизувати та полегшити ряд процесів у цих закладах. Це може включати в себе замовлення їжі онлайн, управління меню, управління замовленнями, клієнтський облік, відгуки та рейтинги, акції та знижки тощо.

Значення дослідження:

Розробка веб-додатку для оптимізації закладів харчування має велике значення як для самих закладів, так і для їхніх клієнтів. Для закладів це можливість покращити ефективність своєї роботи, залучити більше клієнтів, забезпечити їм кращий сервіс та створити конкурентні переваги на ринку. Для клієнтів це можливість замовляти їжу зручним способом, швидко отримувати послуги, мати доступ до актуальної інформації про заклади та отримувати знижки та пропозиції.

Отже, дослідження об'єкта та предмета надає нам можливість глибше зрозуміти потреби та проблеми закладів харчування та розробити ефективні інструменти для їхнього вирішення.

3 ТЕОРЕТИЧНІ ВІДОМОСТІ

3.1 Огляд існуючих систем управління замовленнями в закладах харчування

Сфера громадського харчування надзвичайно динамічна та конкурентна, і сьогоднішні заклади харчування мають багато вимог щодо оптимізації своєї роботи та покращення обслуговування клієнтів. Це призводить до появи і розвитку різноманітних систем управління замовленнями, які надають закладам харчування інструменти для ефективного управління процесами приготування страв, обслуговування клієнтів, управління персоналом та інше. У цьому розділі ми розглянемо деякі існуючі системи управління замовленнями в закладах харчування, їх функціональні можливості, переваги та недоліки.

1. Toast POS

Toast POS - це комплексна платформа управління рестораном, яка надає закладам харчування набір інструментів для автоматизації бізнес-процесів. Система включає в себе модулі для замовлення їжі, управління залишками товарів, облік фінансів, управління клієнтськими відгуками та багато іншого. Toast POS дозволяє підвищити ефективність роботи закладу, зменшити час обробки замовлень та покращити обслуговування клієнтів.

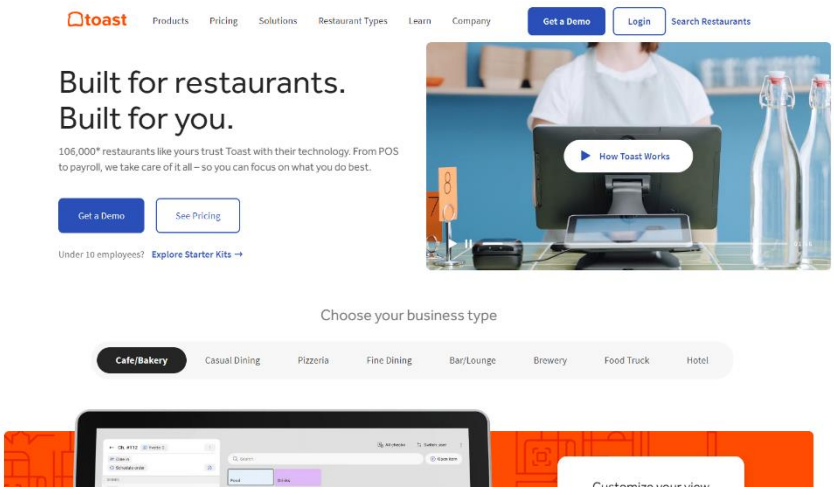

Рисунок 3.1.1 – Інтерфейс

2. Square for Restaurants

Square for Restaurants - це інтегрована система управління рестораном від компанії Square. Вона надає ресторанам можливість приймати замовлення, керувати робочим часом персоналу, вести облік фінансів та багато іншого. Система має зручний інтерфейс та широкі можливості налаштування, що дозволяє відповідати потребам різних типів закладів харчування.

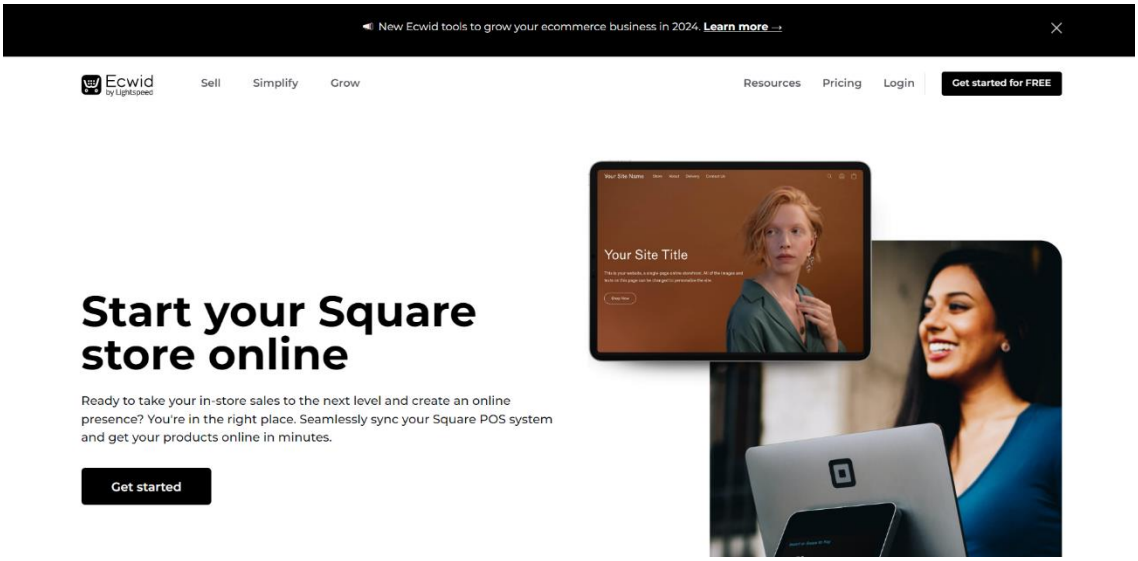

Рисунок 3.1.2 – Інтерфейс

3. Poster

Poster - це інтегрована система управління рестораном, яка надає широкий спектр функцій, включаючи прийом замовлень, керування залишками товарів, аналітику продажів, управління персоналом та багато іншого. Система має інтуїтивний інтерфейс та швидку реакцію, що дозволяє зручно та ефективно керувати роботою закладу.

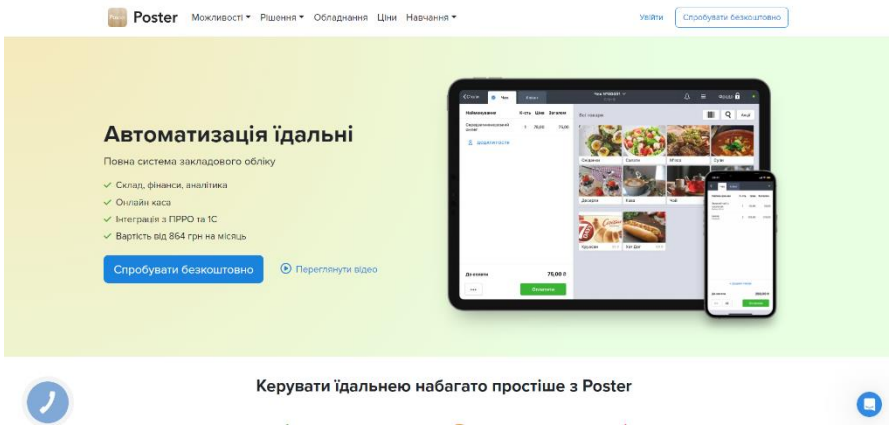

Рисунок 3.1.3 – Інтерфейс

4. TouchBistro

TouchBistro - це система управління рестораном, спеціально розроблена для закладів харчування з обслуговуванням за столиками. Вона має широкий спектр функцій, включаючи керування меню, замовлення страв, управління столиками, облік фінансів та інші. TouchBistro дозволяє ресторанам оптимізувати процес обслуговування клієнтів та підвищити ефективність роботи персоналу.

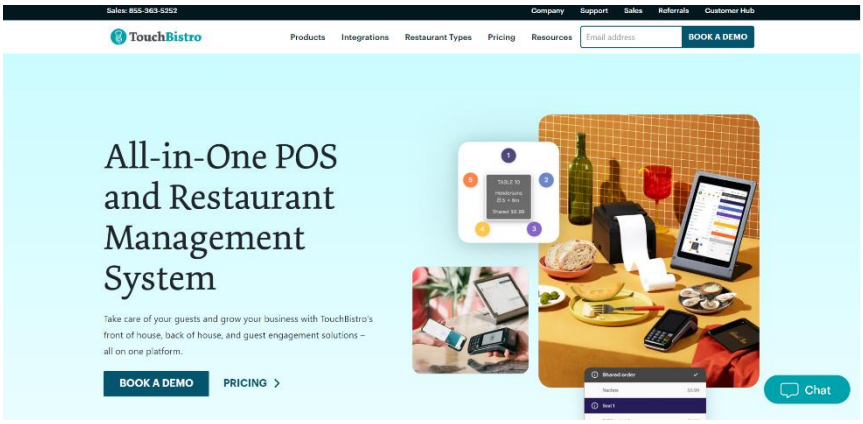

Рисунок 3.1.4 – Інтерфейс

5. Revel Systems

Revel Systems - це харчова система управління, яка надає ресторанам і кафе широкий набір інструментів для ефективного управління бізнесом. Система включає в себе модулі для замовлення їжі, управління складом, облік фінансів, аналітику продажів та інше. Revel Systems має зручний інтерфейс та гнучкі налаштування, що дозволяють адаптувати систему під потреби конкретного закладу харчування.

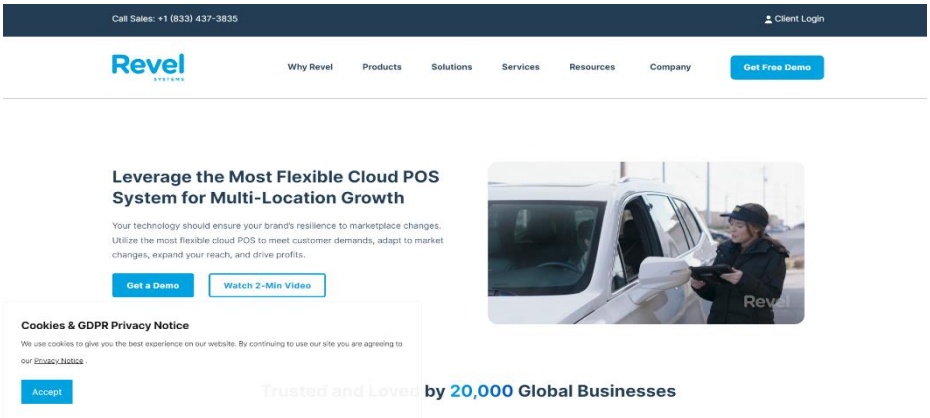

Рисунок 3.1.5 – Інтерфейс

Аналіз існуючих систем:

Існуючі системи управління замовленнями в закладах харчування надають широкий спектр функціональних можливостей та дозволяють ресторанам та кафе оптимізувати свою роботу, підвищити ефективність обслуговування клієнтів та покращити управління бізнесом. Однак кожна система має свої переваги та недоліки, і вибір конкретної системи повинен залежати від потреб та характеристик конкретного закладу харчування.

Деякі з переваг існуючих систем включають в себе широкий спектр функцій, зручний інтерфейс, можливість інтеграції з іншими системами та багато іншого. Однак деякі системи можуть бути складними у використанні, мати обмежені можливості налаштування, або бути недостатньо гнучкими для задоволення потреб конкретного закладу харчування.

Враховуючи ці фактори, при виборі системи управління замовленнями важливо провести детальний аналіз потреб та характеристик свого закладу харчування, а також оцінити переваги та недоліки доступних на ринку рішень. Інвестування у правильну систему управління замовленнями може значно підвищити ефективність та конкурентоспроможність вашого бізнесу в сфері громадського харчування.

3.2 Опис основних принципів веб-додатків

Веб-додатки стали необхідною складовою сучасного цифрового світу, їх популярність постійно зростає завдяки швидкому розвитку технологій та зростаючій доступності Інтернету. Основні принципи веб-додатків визначають їхні можливості, ефективність та зручність для користувачів. У цьому розділі ми розглянемо основні принципи веб-додатків та їхнє значення для розробки та використання.

3.2.1. Клієнт-серверна архітектура

Одним із ключових принципів, що лежить в основі сучасних веб-додатків, є використання клієнт-серверної архітектури. Згідно з цим принципом, веб-додаток розбивається на дві основні складові: клієнтську та серверну частини, кожна з яких відповідає за певні аспекти функціонування та взаємодії з користувачем.

- Роль клієнтської частини: Клієнтська частина веб-додатку функціонує у веб-браузері або мобільному додатку і відповідає за відображення інтерфейсу користувача, а також за обробку подій, спричинених користувачем, таких як кліки, натискання кнопок або введення даних. Вона відповідає за створення зручного та інтуїтивно зрозумілого інтерфейсу, що спрощує взаємодію з користувачем.
- Роль серверної частини: Серверна частина веб-додатку, навпаки, функціонує на сервері та відповідає за обробку запитів користувачів, доступ до бази даних, аутентифікацію користувачів та інші функції, які вимагають обробки на сервері. Вона забезпечує логіку додатка, збереження даних та взаємодію з базою даних чи іншими зовнішніми ресурсами.
- Вигоди клієнт-серверної архітектури: Ця архітектурна модель дозволяє досягти ефективності роботи веб-додатку та розподілити навантаження між клієнтською та серверною сторонами. Це дозволяє покращити продуктивність, масштабованість та забезпечити кращу контрольну точку для розробки та підтримки додатку.

Цей підрозділ спростежує важливість та функціональність клієнт-серверної архітектури, що є важливим аспектом для розуміння та ефективної реалізації веб-додатків.

3.2.2. Кросплатформенність та доступність

Ще одним ключовим аспектом розробки веб-додатків є їхня кросплатформенність, яка визначається здатністю додатка працювати на різних операційних системах та пристроях без необхідності великих модифікацій. Кросплатформенність надає веб-додаткам більшу гнучкість та універсальність, що дозволяє їм залучати більшу аудиторію користувачів.

Доступність веб-додатків є ще одним важливим аспектом, який визначає їхню успішність. Цей аспект описує здатність додатків працювати на різних пристроях та забезпечувати коректне відображення та функціональність, навіть на пристроях з низькими технічними характеристиками або обмеженим Інтернет-з'єднанням. Важливою є також здатність додатків адаптуватися до різних контекстів використання, включаючи особливості взаємодії з користувачем та відображення інформації на екрані.

Для досягнення кросплатформенності та доступності веб-додатків розробники використовують різні підходи, такі як використання веб-стандартів, адаптивного дизайну, розробки мобільних додатків за допомогою кросплатформених фреймворків та інші методи. Ці підходи дозволяють забезпечити максимальну сумісність та зручність використання додатків для різних категорій користувачів та їхніх пристроїв.

Кросплатформенність та доступність є важливими аспектами розробки веб-додатків, оскільки вони дозволяють залучати широку аудиторію користувачів та забезпечують зручність та доступність використання додатків незалежно від технічних характеристик пристроїв та умов використання.

3.2.3. Масштабованість та продуктивність

Ще одним ключовим принципом успішної розробки веб-додатків є їхня масштабованість, яка визначається здатністю додатків працювати з великою

кількістю користувачів або обсягами даних без втрати продуктивності. Масштабованість дозволяє забезпечити стабільну роботу додатку навіть у періоди пікового навантаження, коли кількість користувачів або обсяги даних можуть бути значно збільшені.

Продуктивність веб-додатків визначається їхньою швидкодією та ефективністю роботи, яка включає швидкість відповіді на запити користувачів, час завантаження сторінок та обробки даних. Забезпечення високої продуктивності додатків є ключовим завданням для забезпечення задоволення користувачів та збереження їхньої уваги.

Для досягнення масштабованості та продуктивності веб-додатків розробники використовують різні стратегії та техніки, такі як кешування даних, паралельне виконання завдань, оптимізація запитів до бази даних, використання CDN (Content Delivery Network) для швидкої доставки контенту користувачам та інші методи. Ці стратегії дозволяють оптимізувати роботу додатків та забезпечити їхню стабільну роботу навіть при великому обсязі взаємодії з користувачами та даними.

Масштабованість та продуктивність є критично важливими аспектами для успішної експлуатації веб-додатків, особливо у сучасному середовищі зростаючого обсягу даних та користувачів. Забезпечення високої масштабованості та продуктивності дозволяє забезпечити задоволення потреб користувачів та конкурентоспроможність додатків на ринку.

3.2.4. Безпека та конфіденційність

Іншим ключовим принципом розробки веб-додатків є їхня безпека та конфіденційність. Безпека веб-додатків охоплює захист від різноманітних видів кібератак, таких як витік даних, вторгнення в систему або вірусні атаки. Забезпечення безпеки веб-додатків вимагає розробки та впровадження

різноманітних заходів безпеки, таких як шифрування даних, аутентифікація користувачів та моніторинг системи на предмет потенційних загроз.

Конфіденційність веб-додатків означає забезпечення захисту конфіденційної інформації користувачів, такої як особисті дані, фінансові дані або комерційна інформація. Це важливо для забезпечення довіри користувачів та забезпечення їхньої конфіденційності та безпеки під час користування веб-додатком.

Для забезпечення безпеки та конфіденційності веб-додатків розробники використовують різноманітні технології та практики. Це може включати в себе шифрування даних під час передачі, використання захищених протоколів зв'язку, регулярне оновлення системи з метою закриття вразливостей та реалізацію механізмів аутентифікації та авторизації користувачів.

Забезпечення безпеки та конфіденційності веб-додатків є важливим аспектом, оскільки це впливає на довіру користувачів до додатку та його успішність на ринку. Недостатня безпека може призвести до серйозних наслідків, таких як витік конфіденційної інформації, фінансові втрати або пошкодження репутації компанії. Тому розробники повинні приділяти особливу увагу цим аспектам під час розробки та підтримки веб-додатків.

3.2.5. Гнучкість та розширюваність

Останнім, але не менш важливим принципом розробки веб-додатків є їхня гнучкість, яка визначається здатністю додатків адаптуватися до змін у вимогах користувачів та ринкових умов. Це означає, що веб-додатки повинні бути готові до змін у функціональності, додавання нових можливостей або підтримки нових технологій для забезпечення задоволення потреб користувачів та відповіді на вимоги ринку.

Розширюваність веб-додатків визначається їхньою здатністю розширювати свої можливості та обсяги при необхідності. Це може включати в себе підтримку нових типів пристроїв, збільшення масштабу обробки даних або підтримку нових функцій та сервісів для забезпечення росту та розвитку веб-додатку з плином часу.

Для забезпечення гнучкості та розширюваності веб-додатків розробники використовують архітектурні підходи, такі як мікросервісна архітектура, використання API для розширення функціональності, розробка з використанням модульного підходу та інші методи. Ці підходи дозволяють забезпечити гнучкість та розширюваність додатків для відповіді на змінні потреби користувачів та вимоги ринку.

Гнучкість та розширюваність є критично важливими аспектами для успішної експлуатації веб-додатків у сучасному технологічному середовищі. Забезпечення гнучкості дозволяє додаткам залишатися актуальними та конкурентоспроможними у змінному середовищі, тоді як розширюваність дозволяє їм рости та розвиватися для задоволення потреб користувачів та відповідати на зміни у вимогах ринку.

Основні принципи веб-додатків визначають їхні можливості, ефективність та зручність для користувачів. Клієнт-серверна архітектура, кросплатформенність та доступність, масштабованість та продуктивність, безпека та конфіденційність, гнучкість та розширюваність - це ключові принципи, які дозволяють веб-додаткам успішно функціонувати та задовольняти потреби користувачів у сучасному цифровому світі.

3.3. Визначення основних концепцій Flask, PostgreSQL та Bootstrap

Flask, PostgreSQL та Bootstrap є ключовими інструментами у розробці веб-додатків. Flask - це легкий та гнучкий мікрофреймворк для Python, який використовується для створення веб-додатків та веб-сайтів. PostgreSQL – вільна об'єктно-реляційна система управління базами даних. Bootstrap - це фреймворк фронтенду, який надає набір готових до використання компонентів та стилів для швидкого розроблення зручного та сучасного веб-інтерфейсу. У цьому розділі ми розглянемо основні концепції та функціональність Flask, PostgreSQL та Bootstrap.

3.3.1. Опис Flask:

1. **Мікрофреймворк:** Flask відомий як мікрофреймворк, що означає, що він має мінімальний набір функцій, необхідних для створення веб-додатку. Це робить його легким для вивчення та використання, особливо для початківців. Flask дозволяє розробникам використовувати тільки ті компоненти, які необхідні для конкретного проекту, що робить його більш гнучким та налаштованим під потреби розробника.
2. **Вбудований сервер розробки:** Flask постачається з вбудованим сервером розробки, який дозволяє розробникам тестувати свій веб-додаток без необхідності налаштування окремого веб-сервера. Це робить процес розробки більш простим та ефективним.
3. **URL-шаблони:** Flask використовує URL-шаблони для визначення адрес сторінок у веб-додатку. Це дозволяє розробникам створювати динамічні URL-адреси, які можуть залежати від параметрів запиту. URL-шаблони роблять структуру веб-додатка більш організованою та легко змінюваною.

4. **Виготовлення відповідей:** Flask дозволяє розробникам легко створювати відповіді на HTTP-запити. Це може бути HTML-сторінка, JSON-відповідь, файл або будь-який інший контент, який необхідно повернути користувачеві.
5. **Розширення:** Flask має велику кількість розширень, які допомагають розширити його базовий функціонал. Ці розширення охоплюють такі аспекти як аутентифікація, кешування, форми, бази даних тощо. Вони дозволяють розробникам швидко та легко додавати новий функціонал до їхніх веб-додатків.
6. **Спрощене управління конфігураціями:** Flask має просту систему управління конфігураціями, що дозволяє визначати параметри конфігурації за допомогою змінних середовища або файлів конфігурації. Це робить керування конфігураціями більш простим та зручним для розробників.

3.3.2. Опис PostgreSQL:

1. **Масштабованість:** Одним з найбільших плюсів PostgreSQL є те що вона може легко керувати великим обсягом даних. Це стосується не тільки обсягу даних, якими може керувати ця БД, але і числа користувачів, що одночасно працюють у ній.
2. **Гнучкість:** Ця база даних PostgreSQL підтримує як реляційні, і нереляційні запити.

- Нереляційна база даних — база даних, у якій на відміну від більшості традиційних систем баз даних не використовується таблична схема рядків і стовпців. У цих базах даних застосовується модель зберігання, оптимізована під конкретні вимоги типу даних, що зберігаються. Наприклад, дані можуть зберігатися як прості пари "ключ - значення", документи JSON або граф, що складається з ребер та вершин.

- Реляційна база даних – це набір даних із заданими взаємозв'язками.

Реляційна модель об'єднує дані в таблиці, де кожен рядок є окремим записом, а кожен стовпець складається з атрибутів, що містять значення. Табличний формат дозволяє легко встановлювати зв'язки між точками даних та отримувати доступ до інформації будь-яким необхідним способом, не реорганізовуючи дані.

Також PostgreSQL сумісна з низкою найважливіших мов програмування і протоколів, включаючи C, C++, Python, Perl, Java, .Net, Ruby, ODBC і Tcl. Що дозволяє користувачам працювати використовуючи мову яка для них назручніша.

3. **Кросплатформність:** Окрім того що PostgreSQL є досить гнучкою базою даних яка задовольняє більшість потреб вона сумісна з усіма основними операційними системами, включаючи Linux, Windows та Macintosh. Це робить PostgreSQL більш популярним та розповсюдженим рішенням для створення та керування базами даних.

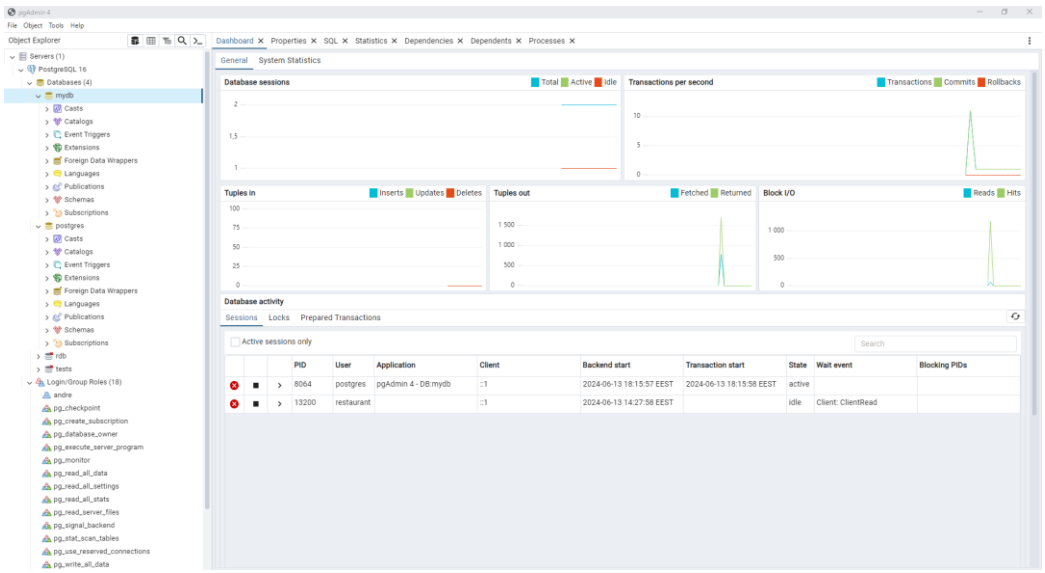

Рисунок 3.3.2 – pgAdmin 4

3.3.3. Опис Bootstrap:

1. **Готові компоненти:** Bootstrap надає велику кількість готових до використання компонентів, таких як кнопки, форми, навігаційні панелі, каруселі та багато інших. Ці компоненти дозволяють розробникам швидко створювати зручний та стильний веб-інтерфейс без необхідності писати власний CSS-код.
2. **Система сітки:** Bootstrap базується на системі сітки, яка дозволяє розробникам легко розміщувати елементи на сторінці та робити їх адаптивними до різних розмірів екранів. Це дозволяє створювати веб-сторінки, які виглядають зручно на будь-яких пристроях, від комп'ютерів до мобільних телефонів.
3. **Стилізація за допомогою CSS:** Bootstrap надає набір готових стилів, які дозволяють розробникам швидко стилізувати свій веб-додаток без необхідності великих CSS-знань. Це робить процес створення стильного веб-інтерфейсу більш простим та ефективним.
4. **JavaScript компоненти:** Bootstrap має також набір готових JavaScript компонентів, таких як модальні вікна, випадаючі меню та підказки, які доповнюють функціонал стандартних HTML-елементів. Ці компоненти дозволяють розробникам додавати інтерактивність та функціональність до своїх веб-додатків з мінімальними зусиллями.
5. **Кастомізація:** Bootstrap дозволяє розробникам легко кастомізувати вигляд та поведінку своїх компонентів за допомогою власних CSS-стилів та JavaScript-коду. Це дозволяє створювати унікальні та індивідуальні веб-додатки, які відповідають потребам конкретного проекту.

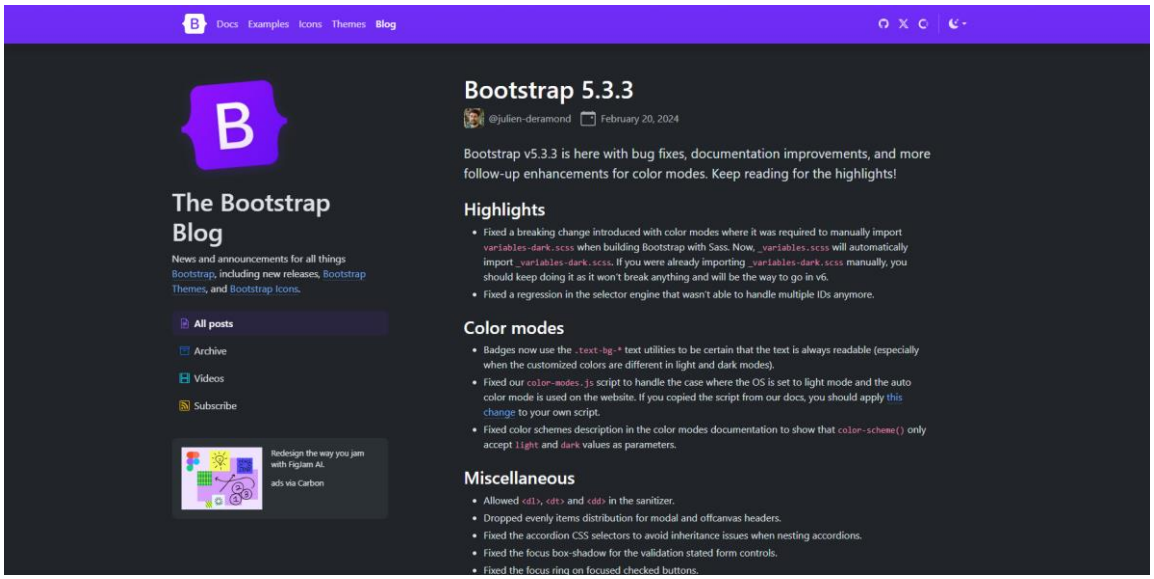

Рисунок 3.3.3 – сторінка Bootstrap

Flask, PostgreSQL та Bootstrap - це потужні інструменти, які дозволяють розробникам швидко та ефективно створювати сучасні веб-додатки. Flask надає зручну і гнучку основу для створення веб-серверів та логіки додатків, PostgreSQL дозволяє легко керувати БД, тоді як Bootstrap надає набір готових компонентів та стилів для швидкої розробки стильного та зручного веб-інтерфейсу. Використання цих двох інструментів разом дозволяє розробникам створювати потужні та естетично приємні веб-додатки з мінімальними зусиллями.

Після розгляду основних переваг Flask, PostgreSQL та Bootstrap, важливо також звернути увагу на їхні недоліки. Хоча ці інструменти мають численні переваги і широко використовуються у веб-розробці, вони також мають певні обмеження і потенційні проблеми. Розуміння цих мінусів допоможе прийняти більш обґрунтоване рішення щодо їх використання в залежності від вимог проекту.

3.4.1. Недоліки Flask

1. **Обмежений набір вбудованих функцій**: Flask, будучи мікрофреймворком, має обмежений набір вбудованих функцій. Це означає, що багато функціональності, яка є у більш великих фреймворках, таких як Django, потрібно додавати вручну через розширення або написання власного коду.
2. **Масштабованість**: Flask може бути складнішим для масштабування в порівнянні з більш комплексними фреймворками. При зростанні проекту і кількості користувачів може знадобитися більше зусиль для оптимізації та підтримки продуктивності.
3. **Відсутність вбудованої ORM**: Flask не має вбудованої системи управління базами даних (ORM). Для роботи з базами даних необхідно використовувати зовнішні бібліотеки, такі як SQLAlchemy, що додає додатковий шар складності.

Альтернативи Flask:

1. **Django**

   Переваги:

   - Велика кількість вбудованих функцій, таких як ORM, аутентифікація, адміністрування тощо.
   - Структурованість і стандартизація, що полегшує управління великими проектами.

   Недоліки:

   - Велика кількість вбудованих функцій може ускладнити розробку невеликих проектів.
   - Важче вивчити і використовувати для простих проектів порівняно з Flask.

2. **FastAPI**

   Переваги:

   - Висока продуктивність завдяки асинхронній обробці запитів.
   - Вбудована підтримка OpenAPI та автоматична генерація документації.

   Недоліки:

   - Новіший фреймворк, менша кількість ресурсів і прикладів у порівнянні з Flask.
   - Може бути складніше для новачків через використання асинхронного програмування.

3.4.2. Недоліки PostgreSQL

1. **Складність налаштування**: PostgreSQL може бути складнішим у налаштуванні та адмініструванні порівняно з іншими СУБД, такими як MySQL.
2. **Ресурсоємність**: PostgreSQL може споживати більше ресурсів системи (пам'яті та процесорного часу), що може бути проблемою для малих проектів або обмежених ресурсів серверів.
3. **Можливі проблеми з сумісністю**: Деякі старіші або менш поширені інструменти та бібліотеки можуть не мати повної підтримки PostgreSQL.

   Альтернативи PostgreSQL:

1. **MySQL**

   Переваги:

   - Легше налаштування та адміністрування.
   - Широко використовується та має велику спільноту підтримки.

Недоліки:

- Менш потужні можливості роботи з транзакціями та розширеннями у порівнянні з PostgreSQL.

Менш гнучкий у роботі з складними запитами та обробкою великих обсягів даних.

2. **SQLite**

Переваги:

- Дуже легка установка та використання, відмінний вибір для малих проектів або прототипів.
- Не потребує налаштування серверної частини.

Недоліки:

- Не підходить для великих або навантажених додатків через обмеження продуктивності та одночасного доступу.
- Менші можливості для масштабування та складної обробки даних.

3.4.3. Недоліки Bootstrap

1. **Обмеженість дизайну**: Використання стандартних компонентів Bootstrap може призвести до того, що веб-сайти виглядають схожими один на одного, що може бути проблемою для проектів, які потребують унікального дизайну.
2. **Роздутий CSS та JavaScript**: Bootstrap містить багато коду, який може не використовуватися у проекті, що збільшує розмір файлів CSS та JavaScript і може вплинути на продуктивність завантаження сторінки.

3. **Важкість кастомізації**: Хоча Bootstrap надає можливість кастомізації, для глибоких змін необхідно мати гарні знання CSS та JavaScript, що може ускладнити процес розробки.

Альтернативи Bootstrap:

1. **Foundation**

Переваги:

- Гнучка система сітки та більші можливості кастомізації.
- Добре підходить для створення адаптивних та інклюзивних дизайнів.

Недоліки:

- Менша спільнота та кількість готових прикладів у порівнянні з Bootstrap.
- Може бути складнішим у вивченні для новачків.

2. **Tailwind CSS**

Переваги:

- Високий рівень кастомізації через утилітарний підхід до CSS-класів.
- Менше залежність від готових компонентів, що дозволяє створювати унікальні дизайни.

Недоліки:

- Вимагає більше ручної роботи з CSS-класами, що може ускладнити процес розробки для новачків.
- Може призвести до «роздутих» HTML-шаблонів через велику кількість класів.

Висновок

Кожен з інструментів – Flask, PostgreSQL та Bootstrap – має свої переваги і недоліки. Вибір конкретного інструменту залежить від вимог проекту та рівня досвіду розробника. Flask підходить для легких та гнучких проектів, але може бути обмежений для великих додатків. PostgreSQL є потужним інструментом для роботи з великими даними, але може вимагати більше ресурсів і складності в налаштуванні. Bootstrap дозволяє швидко створювати адаптивні інтерфейси, але може призводити до схожості дизайнів. Розглянуті альтернативи пропонують інші підходи та можливості, які можуть бути більш відповідними в певних ситуаціях, але також мають свої обмеження.

4 АНАЛІЗ ВИМОГ ДО СИСТЕМИ

4.1. Визначення функціональних та нефункціональних вимог

Аналіз вимог до системи - це ключовий етап у процесі розробки будь-якого програмного продукту, включаючи веб-додаток для закладів харчування на базі Flask та Bootstrap. У цьому розділі ми розглянемо процес визначення функціональних та нефункціональних вимог до системи, а також визначимо ключові характеристики, які необхідно врахувати при розробці додатку.

Функціональні вимоги:

1. **Реєстрація та авторизація користувачів:**

- Система повинна надавати можливість користувачам створювати облікові записи та входити до системи за допомогою електронної пошти або інших ідентифікаторів.
- Потрібна можливість відновлення пароля у випадку забутого доступу до облікового запису.

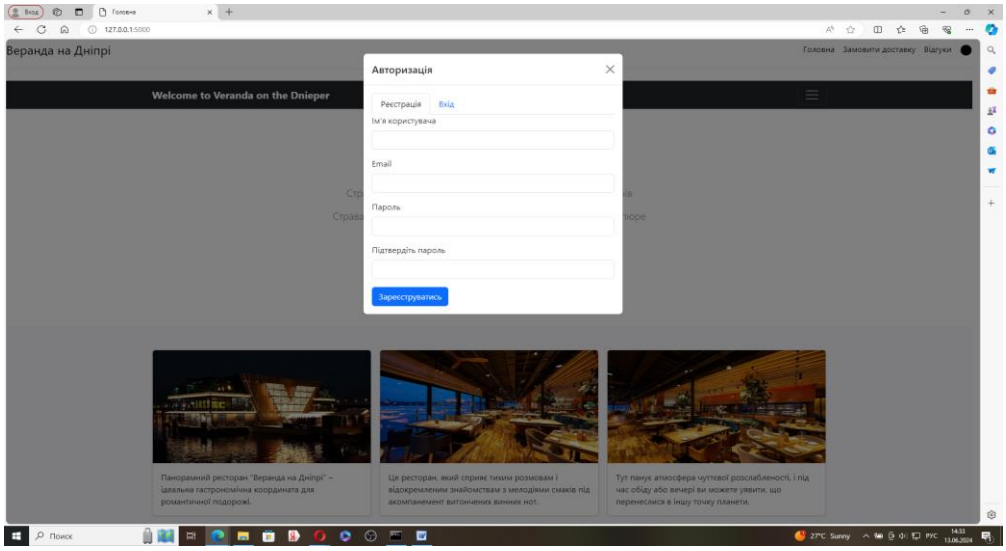

Рисунок 4.1 – вікно авторизації

2. **Замовлення їжі:**

- Користувачі повинні мати можливість переглядати меню закладу та робити замовлення онлайн.
- Додаток повинен підтримувати різні методи оплати, такі як кредитні картки, електронні гаманці, платіжні системи тощо.

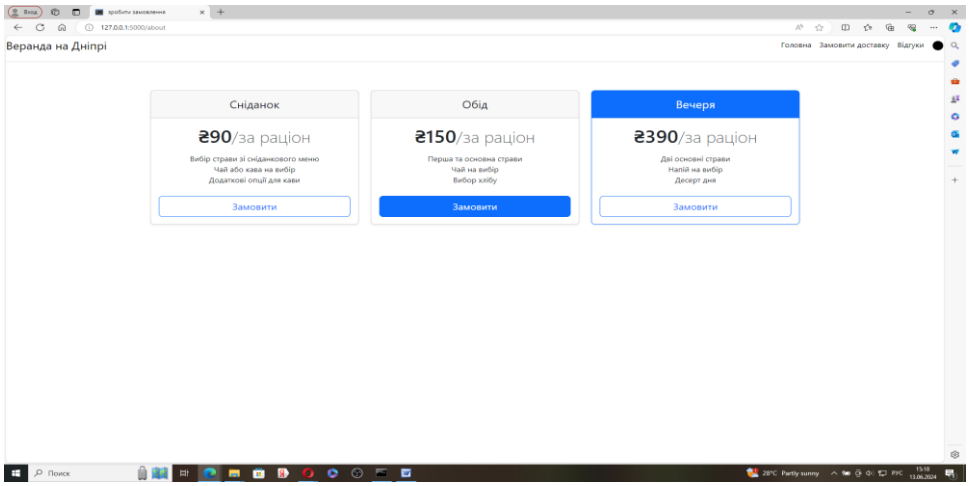

Рисунок 4.2 – сторінка замовлень

3. **Управління замовленнями:**

- Адміністратори повинні мати можливість переглядати та керувати замовленнями, змінювати їх статус, відміняти або підтверджувати.
- Користувачам необхідно надавати зручний інтерфейс для відстеження статусу їхніх замовлень.

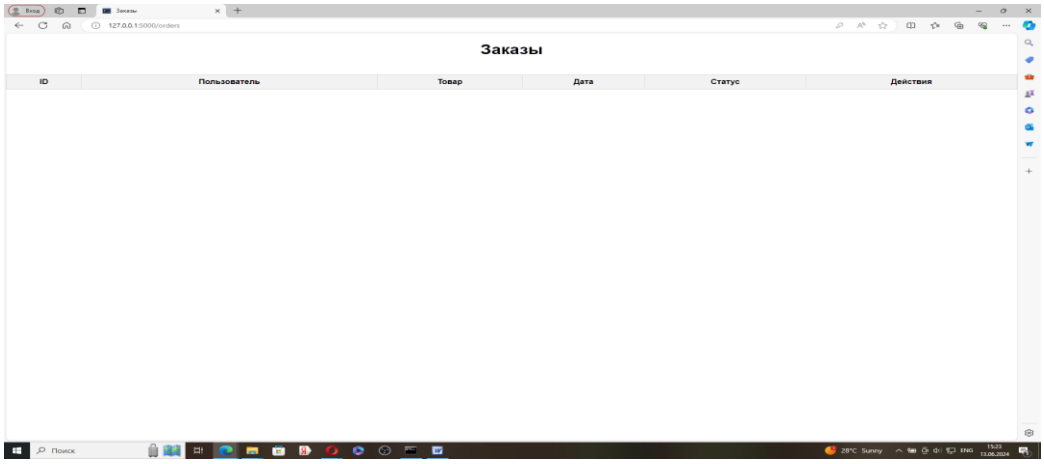

Рисунок 4.3 – сторінка управління замовленнями

4. **Управління меню:**

- Адміністраторам повинна бути можливість додавати, видаляти та редагувати страви у меню, встановлювати їхню ціну та опис.
- Потрібна підтримка категоризації страв для зручного перегляду користувачами.

5. **Управління користувачами:**

- Адміністраторам повинна бути надана можливість керувати обліковими записами користувачів, блокувати або видаляти їхні профілі.

Нефункціональні вимоги:

1. **Зручність використання:**

- Веб-додаток повинен мати інтуїтивно зрозумілий та легкий у використанні інтерфейс для забезпечення комфортного користування як адміністраторами, так і звичайними користувачами.

2. **Швидкодія та продуктивність:**

- Додаток повинен бути швидким у відповіді на запити користувачів та має високу продуктивність навіть при великому обсязі даних.

3. **Безпека:**

- Важливо забезпечити високий рівень безпеки для захисту конфіденційності та цілісності даних користувачів, а також уникнути можливих атак на систему.
- Необхідно використовувати надійні методи аутентифікації та авторизації для запобігання несанкціонованому доступу.

4. **Адаптивний дизайн:**

- Веб-додаток повинен мати адаптивний дизайн, який забезпечить коректне відображення на різних пристроях та розмірах екранів.

5. **Масштабованість:**

- Додаток повинен бути легко масштабованим для забезпечення стабільної роботи навіть при збільшенні обсягу користувацької бази та трафіку.

Аналіз вимог до системи визначає ключові характеристики та функціональність, які необхідно врахувати при розробці веб-додатку для закладів харчування на базі Flask та Bootstrap. Визначення як функціональних, так і нефункціональних вимог допоможе забезпечити успішну реалізацію проекту та задоволення потреб його користувачів.

4.2 Створення сценаріїв використання

Створення сценаріїв використання є важливим етапом у розробці веб-додатку для закладів харчування на базі Flask та Bootstrap. Ці сценарії допомагають розробникам та команді проекту краще зрозуміти, як користувачі будуть взаємодіяти з додатком та які функціональні можливості потрібно реалізувати. У цьому розділі ми розглянемо кілька основних сценаріїв використання для веб-додатку закладу харчування.

1. Реєстрація та вхід до системи:

Сценарій: Реєстрація нового користувача

1. Користувач відкриває веб-додаток та переходить на сторінку реєстрації.
2. Він вводить свої особисті дані, такі як ім'я, прізвище, електронну пошту та пароль.
3. Після заповнення форми, користувач підтверджує свої дані та натискає кнопку "Зареєструватися".
4. Система перевіряє введені дані та реєструє нового користувача.

5. Після успішної реєстрації, користувач автоматично авторизується в системі та переходить на головну сторінку додатку.

Сценарій: Вхід існуючого користувача

1. Користувач відкриває веб-додаток та переходить на сторінку входу.
2. Він вводить свою електронну пошту та пароль, які він використовував під час реєстрації.
3. Після введення даних, користувач натискає кнопку "Увійти".
4. Система перевіряє введені дані та авторизує користувача в системі.
5. Після успішного входу, користувач переходить на головну сторінку додатку та отримує доступ до всіх його функцій.

2. Замовлення їжі:

Сценарій: Перегляд меню та замовлення страв

1. Після входу до системи, користувач переходить на сторінку з меню.
2. Він переглядає список доступних страв разом з їхніми цінами та описами.
3. Після вибору бажаних страв, користувач додає їх до кошика.
4. Після завершення вибору, користувач переходить до кошика та перевіряє замовлення.
5. Після перевірки, він натискає кнопку "Оформити замовлення".
6. Система перевіряє замовлення, обробляє платіж та підтверджує успішну операцію.
7. Користувач отримує підтвердження замовлення та інформацію про очікуваний час доставки.

3. Управління замовленнями:

Сценарій: Перегляд та керування замовленнями адміністратором

1. Адміністратор входить до системи та переходить на сторінку управління замовленнями.
2. Він переглядає список усіх активних замовлень, включаючи їхній статус та деталі.
3. Адміністратор має можливість змінювати статус замовлення, відміняти його або підтверджувати.
4. Після обробки замовлення, адміністратор позначає його як оброблене та надсилає повідомлення користувачеві про статус.

4. Управління меню:

Сценарій: Додавання нових страв до меню

1. Адміністратор входить до системи та переходить на сторінку управління меню.
2. Він обирає опцію "Додати нову страву" та заповнює форму з інформацією про страву, включаючи назву, опис та ціну.
3. Після введення даних, адміністратор натискає кнопку "Зберегти".
4. Система перевіряє введені дані та додає нову страву до меню.
5. Після успішного додавання, адміністратор отримує підтвердження та переходить на сторінку зі списком усіх страв.

5. Управління користувачами:

Сценарій: Блокування облікового запису користувача

1. Адміністратор входить до системи та переходить на сторінку управління користувачами.

2. Він знаходить обліковий запис користувача, який потрібно заблокувати.

3. Адміністратор обирає опцію "Блокувати" та підтверджує свою дію.

4. Система блокує доступ облікового запису користувача та надсилає повідомлення про це.

Створення сценаріїв використання дозволяє визначити, як користувачі будуть взаємодіяти з веб-додатком для закладів харчування. Ці сценарії включають в себе основні дії та операції, які будуть виконувати як звичайні користувачі, так і адміністратори системи. Врахування цих сценаріїв у процесі розробки допоможе створити зручний та функціональний веб-додаток, який задовольнить потреби користувачів.

5 РЕАЛІЗАЦІЯ СИСТЕМИ

5.1 Налаштування середовища розробки

Налаштування середовища розробки є першим і важливим кроком у створенні веб-додатку для закладів харчування на базі Flask та Bootstrap. Ефективне середовище розробки допомагає збільшити продуктивність, полегшує відлагодження та сприяє успішній реалізації проекту. У цьому розділі ми розглянемо ключові аспекти налаштування середовища розробки для створення веб-додатку з використанням Flask та Bootstrap.

1. Вибір IDE або текстового редактора:

Першим кроком є вибір інтегрованого середовища розробки (IDE) або текстового редактора, який ви будете використовувати для написання коду. Деякі популярні вибори включають PyCharm, Visual Studio Code, Sublime Text, або Atom. Кожен з них має свої переваги та можливості, тому важливо обрати той, який найбільше відповідає вашим потребам та особистому вподобанню.

2. Встановлення Python та Flask:

Переконайтеся, що ви маєте встановлену актуальну версію Python на вашому комп'ютері. Flask можна легко встановити за допомогою pip, менеджера пакетів Python. Ви можете встановити Flask, виконавши наступну команду в командному рядку:

pip install Flask

Також вам може знадобитися встановити додаткові пакети, такі як:

- Flask-SQLAlchemy є розширенням для Flask, яке інтегрує SQLAlchemy — потужний інструмент ORM (Object-Relational Mapping) для роботи з базами даних у Python. Він дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід.
- Flask-WTF забезпечує просте створення та валідацію форм у веб-додатках.
- Flask-Login потрібен для управління сесіями користувачів та забезпечення аутентифікації. Надає функціонал для входу, виходу та обмеження доступу до певних частин додатка, забезпечуючи безпеку та контроль доступу.
- WTForms є бібліотекою для створення та валідації форм у Python. Вона забезпечує потужний та гнучкий інструмент для роботи з формами, дозволяючи легко створювати та перевіряти форми в додатку.
- Flask-Bcrypt використовується для хешування паролів з використанням алгоритму Bcrypt, що забезпечує високу безпеку збереження паролів.
- Flask-Migrate є розширенням для Flask, яке інтегрує Alembic для управління міграціями бази даних. Це дозволяє легко керувати змінами в схемі бази даних протягом життєвого циклу проекту.
- cloudipsp є клієнтською бібліотекою для інтеграції з CloudPayments, що дозволяє обробляти онлайн-платежі у веб-додатках. Це забезпечує можливість швидко та безпечно здійснювати платежі онлайн.

Вибір цих та інших пакетів залежить від функціональності, яку ви плануєте реалізувати у своєму додатку.

3. Робоче середовище віртуального середовища:

Рекомендується використовувати віртуальне середовище Python для ізоляції вашого проекту від інших програм на вашому комп'ютері та уникнення конфліктів залежностей. Ви можете створити віртуальне середовище, використовуючи модуль venv:

python -m venv myenv

А потім активувати його:

source myenv/bin/activate (на Linux/Mac)

myenv\Scripts\activate (на Windows)

4. Використання системи контролю версій:

Система контролю версій (VCS), така як Git, допоможе вам ефективно керувати змінами в коді та спільно працювати з іншими членами команди. Створіть репозиторій Git для вашого проекту та регулярно вносьте зміни в нього, забезпечуючи безпеку та контроль над розробкою.

5. Налаштування сервера розробки:

Flask має вбудований веб-сервер, який можна використовувати для розробки та тестування вашого додатку. Це дозволяє вам запускати ваш додаток локально на вашому комп'ютері для тестування перед розгортанням на виробничий сервер. Для запуску сервера виконайте команду:

flask run

6. Використання Bootstrap:

Для використання Bootstrap у вашому проекті ви можете підключити його стилі та скрипти безпосередньо до веб-сторінок вашого додатку. Ви також можете використовувати сторонні CDN або встановити Bootstrap через npm або

yarn. Завдяки цьому ваші сторінки будуть мати сучасний та адаптивний дизайн, що полегшить їхнє використання для користувачів на різних пристроях.

7. Налаштування розробки бази даних:

Якщо ваш веб-додаток потребує бази даних, вам знадобиться налаштувати з'єднання та роботу з нею. Використовуйте SQLAlchemy або Flask-SQLAlchemy для спрощення взаємодії з базою даних у вашому додатку. Налаштуйте вашу базу даних та моделі, щоб забезпечити збереження та обробку даних у вашому додатку.

Налаштування середовища розробки - це ключовий етап у процесі створення веб-додатку для закладів харчування на базі Flask та Bootstrap. Важливо обрати правильні інструменти та налаштувати їх таким чином, щоб забезпечити ефективну та продуктивну розробку. Грамотне налаштування середовища розробки допоможе вам швидше досягти своїх цілей та створити високоякісний веб-додаток.

ВИСНОВОК

У цій дипломній роботі ми розглянули процес розробки веб-додатку для закладів харчування з використанням фреймворка Flask та фронтенду на базі Bootstrap. Дослідження зосереджувалося на розробці повноцінної системи, яка могла б надати користувачам можливість здійснювати замовлення їжі та адміністраторам - управляти меню та замовленнями.

На перших етапах дослідження ми провели аналіз вимог до системи, де чітко визначили функціональні та нефункціональні вимоги, а також обґрунтували актуальність теми. Після цього ми розробили архітектуру системи, вибравши Flask як основний інструмент для реалізації серверної частини та Bootstrap для клієнтського інтерфейсу.

У процесі роботи над дипломною роботою ми розглянули та реалізували різні компоненти системи, такі як аутентифікація користувачів, управління меню та замовленнями, а також взаємодію з базою даних. Ми також вивчили основні принципи веб-додатків та використання фреймворків Flask та Bootstrap для їх розробки.

Результатом нашої роботи став реалізований прототип веб-додатку для закладів харчування, який надає користувачам можливість здійснювати замовлення та адміністраторам - управляти роботою системи. Ми успішно впровадили основні функціональність, такі як автентифікація, управління меню та замовленнями, а також забезпечили зв'язок з базою даних для зберігання та обробки інформації. Крім того, наш додаток має адаптивний дизайн, що робить його доступним на різних пристроях.

Однак, слід відзначити, що наш прототип має обмежений функціонал та може бути розширений та вдосконалений. Для подальшого розвитку системи необхідно розглянути додавання нових функціональних можливостей, таких як система оцінок та відгуків, можливість поповнення балансу та інші. Також

варто удосконалити інтерфейс користувача, зробити його більш інтуїтивно зрозумілим та привабливим для користувачів.

Для подальшого розвитку системи рекомендується провести докладний аналіз потреб користувачів та внести необхідні зміни у функціонал додатку відповідно до цих потреб. Також варто розглянути можливість інтеграції додаткових сервісів та API, які можуть поліпшити функціональність системи та зробити її більш привабливою для користувачів.

Крім того, рекомендується провести тестування системи на різних етапах розробки, щоб виявити та виправити можливі помилки та недоліки. Тестування повинно включати як модульні, так і інтеграційні тести, а також тести на відповідність вимогам користувача. Такий підхід дозволить забезпечити високу якість та надійність системи.

Узагальнюючи, розробка веб-додатку для закладів харчування є складним та важливим завданням, яке вимагає уважного підходу та ретельного планування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flask Documentation. (https://flask.palletsprojects.com/)
2. Bootstrap Documentation. (https://getbootstrap.com/docs/)
3. "Learning Flask" by Miguel Grinberg. O'Reilly Media, 2015.
4. "Flask Web Development: Developing Web Applications with Python" by Miguel Grinberg. O'Reilly Media, 2018.
5. "Bootstrap 4: Exploring New Features" by Ray Villalobos. LinkedIn Learning, 2020.
6. "Bootstrap 4 Essential Training" by Ray Villalobos. LinkedIn Learning, 2018.
7. "SQLAlchemy Documentation." (https://www.sqlalchemy.org/)
8. "Introduction to Database Systems" by C.J. Date. Addison Wesley, 2004.
9. "Software Engineering: A Practitioner's Approach" by Roger S. Pressman. McGraw-Hill Education, 2014.
10. "Flask Mega-Tutorial" by Miguel Grinberg. (https://blog.miguelgrinberg.com/category/Flask)

ДОДАТКИ

ДОДАТОК А

КОД ФАЙЛУ APP.PY

```
from flask import Flask, render_template, url_for, request, redirect, flash, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, SelectField
from wtforms.validators import DataRequired, Length, Email, EqualTo
from flask_bcrypt import Bcrypt
from flask_migrate import Migrate
from datetime import datetime
from cloudipsp import Api, Checkout
from getpass import getpass


app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://restaurant:resthq07@localhost/mydb'
app.config['SECRET_KEY'] = 'you-will-never-guess'
db = SQLAlchemy(app)
migrate = Migrate(app, db)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
```

```
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'


class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(150), nullable=False, unique=True)
    email = db.Column(db.String(150), nullable=False, unique=True)
    password = db.Column(db.String(150), nullable=False)
    is_staff = db.Column(db.Boolean, default=False)


class Review(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.Text, nullable=False)
    date = db.Column(db.DateTime, default=datetime.utcnow)
    last_edited = db.Column(db.DateTime, nullable=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

    def __repr__(self):
        return f'<Review {self.id}>'
```

```
class Item(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    price = db.Column(db.Integer, nullable=False)
    description = db.Column(db.Text, nullable=True)

    def __repr__(self):
        return self.name


class Order(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    item_id = db.Column(db.Integer, db.ForeignKey('item.id'), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    status = db.Column(db.String(50), nullable=False, default='new')

    item = db.relationship('Item', backref=db.backref('orders', lazy=True))
    user = db.relationship('User', backref=db.backref('orders', lazy=True))


@login_manager.user_loader
def load_user(user_id):
```

```
    return User.query.get(int(user_id))


class RegistrationForm(FlaskForm):
    username = StringField('Ім\'я користувача', validators=[DataRequired(),
Length(min=2, max=20)])
    email = StringField('Електронна пошта', validators=[DataRequired(), Email()])
    password = PasswordField('Пароль', validators=[DataRequired(), Length(min=6)])
    confirm_password = PasswordField('Підтвердьте пароль',
validators=[DataRequired(), EqualTo('password')])
    submit = SubmitField('Зареєструватися')


class LoginForm(FlaskForm):
    email = StringField('Електронна пошта', validators=[DataRequired(), Email()])
    password = PasswordField('Пароль', validators=[DataRequired()])
    submit = SubmitField('Увійти')


@app.route('/')
def index():
    return render_template("index.html")
```

```
@app.route('/about')
def about():
    items = Item.query.order_by(Item.price).all()
    return render_template("about.html", data=items)



@app.route('/buy/<int:id>')
def item_buy(id):
    item = Item.query.get(id)

    if item is None:
        flash('Товар не знайдено', 'danger')
        return redirect(url_for('index'))

    new_order = Order(item_id=item.id, user_id=current_user.id)
    db.session.add(new_order)
    db.session.commit()

    api = Api(merchant_id=1396424, secret_key='test')
    checkout = Checkout(api=api)
    data = {
        "currency": "UAH",
        "amount": str(item.price) + "00"
```

```
    }
    url = checkout.url(data).get('checkout_url')
    return redirect(url)


@app.route('/reviews', methods=['POST', 'GET'])
def reviews():
    if request.method == 'POST':
        text = request.form['text']
        review = Review(text=text, user_id=current_user.id)

        try:
            db.session.add(review)
            db.session.commit()
            return jsonify(success=True, message='Відгук успішно додано!')
        except:
            return jsonify(success=False, message='Сталася помилка під час додавання відгуку')

    reviews = Review.query.order_by(Review.date.desc()).all()
    return render_template("reviews.html", reviews=reviews)
```

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    form = RegistrationForm()
    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
        user = User(username=form.username.data, email=form.email.data, password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash('Ваш обліковий запис було створено! Тепер ви можете увійти', 'success')
        return redirect(url_for('login'))
    return render_template('register.html', form=form)


@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    form = LoginForm()
    if form.validate_on_submit():
```

```
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('index'))
        else:
            flash('Вхід не вдався. Перевірте електронну пошту та пароль', 'danger')
    return render_template('login.html', form=form)


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))


@app.route('/delete_review/<int:id>', methods=['POST'])
@login_required
def delete_review(id):
    review = Review.query.get_or_404(id)
    if review.user_id != current_user.id:
```

```
        return jsonify({'success': False, 'message': 'Ви не можете видалити цей відгук'}), 403

    try:
        db.session.delete(review)
        db.session.commit()
        return jsonify({'success': True})
    except:
        return jsonify({'success': False, 'message': 'Помилка при видаленні відгуку'}), 500


@app.route('/edit_review/<int:id>', methods=['POST'])
@login_required
def edit_review(id):
    review = Review.query.get_or_404(id)
    if review.user_id != current_user.id:
        return jsonify({'success': False, 'message': 'Ви не можете редагувати цей відгук'}), 403

    new_text = request.form['text']
    review.text = new_text
    review.last_edited = datetime.utcnow()
    try:
```

```
        db.session.commit()
        return jsonify({'success': True})
    except:
        return jsonify({'success': False, 'message': 'Помилка при редагуванні відгуку'}), 500


@app.route('/orders')
@login_required
def view_orders():
    if current_user.is_staff:  # Перевірка на адміністратора
        orders = Order.query.order_by(Order.date.desc()).all()
        return render_template('orders.html', orders=orders)
    else:
        flash('У вас немає доступу до цієї сторінки', 'danger')
        return redirect(url_for('index'))


@app.route('/update_status/<int:id>', methods=['POST'])
@login_required
def update_status(id):
    if not current_user.is_staff:
        return jsonify({'success': False, 'message': 'У вас немає прав для виконання цієї дії'}), 403
```

```
    order = Order.query.get_or_404(id)
    if order:
        new_status = request.form.get('status')
        if new_status in ['очікування', 'виконується', 'завершено']:
            order.status = new_status
            try:
                db.session.commit()
                return jsonify({'success': True})
            except:
                return jsonify({'success': False, 'message': 'Помилка при оновленні статусу замовлення'}), 500
        else:
            return jsonify({'success': False, 'message': 'Недопустимий статус замовлення'}), 400
    return jsonify({'success': False, 'message': 'Замовлення не знайдено'}), 404



@app.route('/delete_order/<int:id>', methods=['POST'])
@login_required
def delete_order(id):
    if not current_user.is_staff:
        return jsonify({'success': False, 'message': 'У вас немає прав для виконання цієї дії'}), 403
```

```
    order = Order.query.get_or_404(id)
    if order:
        try:
            db.session.delete(order)
            db.session.commit()
            return jsonify({'success': True})
        except:
            return jsonify({'success': False, 'message': 'Помилка при видаленні замовлення'}), 500
    return jsonify({'success': False, 'message': 'Замовлення не знайдено'}), 404


@app.route('/delete_account', methods=['GET', 'POST'])
@login_required
def delete_account():
    if request.method == 'POST':
        user = User.query.filter_by(email=current_user.email).first()
        if user and bcrypt.check_password_hash(user.password, request.form['password']):
            logout_user()
            try:
                db.session.delete(user)
                db.session.commit()
```

```
                return jsonify({'success': True, 'message': 'Обліковий запис успішно видалено'})
            except Exception as e:
                return jsonify({'success': False, 'message': 'Помилка при видаленні облікового запису'}), 500
        else:
            return jsonify({'success': False, 'message': 'Невірний пароль'}), 400
    return render_template('delete_account.html')


if __name__ == "__main__":
    app.run(debug=True)
```

ДОДАТОК Б

КОД ФАЙЛУ BASE.HTML

```
<!doctype html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></
script>

    <title>{% block title %}{% endblock %}</title>

</head>

<body>

    <div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-
bottom">

        <a href="/" class="d-flex align-items-center link-body-emphasis text-
decoration-none">

            <span class="fs-4">Веранда на Дніпрі</span>

        </a>

        <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">

            <a class="me-3 py-2 link-body-emphasis text-decoration-none"
href="/">Головна</a>

            <a class="me-3 py-2 link-body-emphasis text-decoration-none"
href="/about">Замовити доставку</a>
```

```
        <a class="me-3 py-2 link-body-emphasis text-decoration-none" href="/reviews">Відгуки</a>

        {% if current_user.is_authenticated %}

          <a class="btn btn-danger" href="{{ url_for('logout') }}">Вийти</a>

        {% else %}

          <a class="py-2 link-body-emphasis text-decoration-none" href="#" data-bs-toggle="modal" data-bs-target="#authModal">

            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="currentColor" class="bi bi-person-circle" viewBox="0 0 16 16">

              <path d="M13.468 12.37C12.758 11.226 11.43 10.5s-4.758.726-5.468 1.87A6.994 6.994 0 0 0 8 16a6.994 6.994 0 0 0 5.468-3.63zM8 9a3 3 0 1 0 0-6 3 3 0 0 0 0 6z"/>

              <path fill-rule="evenodd" d="M8 1a7 7 0 1 1 0 14A7 7 0 0 1 8 1z"/>

            </svg>

          </a>

        {% endif %}

      </nav>

    </div>

    {% block body %}{% endblock %}


    <!-- Модальное окно для авторизации -->

    <div class="modal fade" id="authModal" tabindex="-1" aria-labelledby="authModalLabel" aria-hidden="true">

      <div class="modal-dialog">

        <div class="modal-content">

          <div class="modal-header">

            <h5 class="modal-title" id="authModalLabel">Авторизація</h5>
```

```
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body">

                <ul class="nav nav-tabs" id="authTab" role="tablist">

                    <li class="nav-item" role="presentation">

                        <a class="nav-link active" id="register-tab" data-bs-toggle="tab" href="#register" role="tab" aria-controls="register" aria-selected="true">Реєстрація</a>

                    </li>

                    <li class="nav-item" role="presentation">

                        <a class="nav-link" id="login-tab" data-bs-toggle="tab" href="#login" role="tab" aria-controls="login" aria-selected="false">Вхід</a>

                    </li>

                </ul>

                <div class="tab-content" id="authTabContent">

                    <div class="tab-pane fade show active" id="register" role="tabpanel" aria-labelledby="register-tab">

                        <form action="{{ url_for('register') }}" method="POST">

                            <div class="mb-3">

                                <label for="username" class="form-label">Ім'я користувача</label>

                                <input type="text" class="form-control" id="username" name="username" required>

                            </div>

                            <div class="mb-3">

                                <label for="email" class="form-label">Email</label>
```

```
                    <input type="email" class="form-control" id="email"
name="email" required>

                </div>

                <div class="mb-3">

                    <label for="password" class="form-label">Пароль</label>

                    <input type="password" class="form-control" id="password"
name="password" required>

                </div>

                <div class="mb-3">

                    <label for="confirm_password" class="form-
label">Підтвердіть пароль</label>

                    <input type="password" class="form-control"
id="confirm_password" name="confirm_password" required>

                </div>

                <button type="submit" class="btn btn-
primary">Зареєструватись</button>

            </form>

        </div>

        <div class="tab-pane fade" id="login" role="tabpanel" aria-
labelledby="login-tab">

            <form action="{{ url_for('login') }}" method="POST">

                <div class="mb-3">

                    <label for="login-email" class="form-label">Email</label>

                    <input type="email" class="form-control" id="login-email"
name="email" required>

                </div>

                <div class="mb-3">
```

```
                        <label for="login-password" class="form-
label">Пароль</label>

                        <input type="password" class="form-control" id="login-
password" name="password" required>

                    </div>

                    <button type="submit" class="btn btn-
primary">Увійти</button>

                </form>

            </div>

        </div>

    </div>

</body>

</html>
```

ДОДАТОК В

КОД ФАЙЛУ INDEX.HTML

```
{% extends 'base.html' %}


{% block title %}
Головна
{% endblock %}


{% block body %}
    <div class="dropdown position-fixed bottom-0 end-0 mb-3 me-3 bd-mode-toggle"></div>
    <header data-bs-theme="dark">
        <div class="collapse text-bg-dark" id="navbarHeader">
            <div class="container">
                <div class="row">
                    <div class="col-sm-8 col-md-7 py-4">
                        <h5>
                            <p class="text-body-secondary">У нас ви знайдете найкращі страви та затишну атмосферу.</p>
                            <p class="text-body-secondary">Приходьте та насолоджуйтеся смачною їжею та відмінним обслуговуванням!</p>
                        </h5>
                    </div>
                    <div class="col-sm-4 offset-md-1 py-4">
                        <h4>Contact</h4>
                        <ul class="list-unstyled">
                            <li><a href="#" class="text-white">Follow on Twitter</a></li>
                            <li><a href="#" class="text-white">Like on Facebook</a></li>
                            <li><a href="#" class="text-white">Email me</a></li>
```

```
                </ul>
            </div>
        </div>
    </div>
    <div class="navbar navbar-dark bg-dark shadow-sm">
        <div class="container">
            <a href="/about" class="navbar-brand d-flex align-items-center">
                <strong>Welcome to Veranda on the Dnieper</strong>
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarHeader" aria-controls="navbarHeader" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
        </div>
    </div>
</header>

<main>
    <section class="py-5 text-center container">
        <div class="row py-lg-5">
            <div class="col-lg-6 col-md-8 mx-auto">
                <h1 class="fw-light">Фірмові страви</h1>
                <p class="lead text-body-secondary">Страва №1 - Стейк з дикого сибаса на подушці з теплих овочів</p>
                <p class="lead text-body-secondary">Страва №2 - іле дорадо з в'яленими томатами та картопляним пюре</p>
                <p class="lead text-body-secondary">Страва №3 - Соте з морепродуктів в азіатському соусі</p>
                <p class="lead text-body-secondary">
```

```
                <a href="URL_TO_MENU" class="text-primary">Переглянути все меню</a>
            </p>
        </div>
    </div>
</section>

<div class="album py-5 bg-body-tertiary">
    <div class="container">
        <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
            <div class="col">
                <div class="card shadow-sm">
                    <img src="https://reston.ua/runtime/cache/248x142/58cc53e.jpg" class="bd-placeholder-img card-img-top" width="100%" height="225" alt="Ресторан Веренда на Днепре 1">
                    <div class="card-body">
                        <p class="card-text">Панорамний ресторан “Веранда на Дніпрі” – ідеальна гастрономічна координата для романтичної подорожі.</p>
                    </div>
                </div>
            </div>
            <div class="col">
                <div class="card shadow-sm">
                    <img src="https://reston.ua/runtime/cache/248x142/285a13b.jpg" class="bd-placeholder-img card-img-top" width="100%" height="225" alt="Ресторан Веренда на Днепре 2">
                    <div class="card-body">
                        <p class="card-text">Це ресторан, який сприяє тихим розмовам і відокремленим знайомствам з мелодіями смаків під акомпанемент витончених винних нот.</p>
                    </div>
                </div>
```

```
            </div>
            <div class="col">
              <div class="card shadow-sm">
                <img src="https://reston.ua/runtime/cache/248x142/7f89474.jpg" class="bd-placeholder-img card-img-top" width="100%" height="225" alt="Ресторан Веренда на Днепре 3">
                <div class="card-body">
                  <p class="card-text">Тут панує атмосфера чуттєвої розслабленості, і під час обіду або вечері ви можете уявити, що перенеслися в іншу точку планети.</p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </main>

      <footer class="text-body-secondary py-5">
      <div class="container">
        <p class="mb-1">
          Телефони для бронювання столів:
          <a href="tel:+1234567890">+1 (234) 567-890</a>
          <a href="tel:+0987654321">+0 (987) 654-321</a>
        </p>
        <p class="mb-0">
          Телефони гарячої лінії:
          <a href="tel:+1122334455">+1 (122) 334-455</a>
          <a href="tel:+5566778899">+5 (566) 778-899</a>
        </p>
```

```
    </div>
</footer>
{% endblock %}
```

ДОДАТОК Г

КОД ФАЙЛУ ABOUT.HTML

```
{% extends 'base.html' %}


{% block title %}
зробити замовлення
{% endblock %}


{% block body %}
<div class="container py-3">
  <main>
    <br>
    <div class="row row-cols-1 row-cols-md-3 mb-3 text-center">
      <div class="col">
        <div class="card mb-4 rounded-3 shadow-sm">
          <div class="card-header py-3">
            <h4 class="my-0 fw-normal">Сніданок</h4>
          </div>
          <div class="card-body">
            <h1 class="card-title pricing-card-title">₴90<small class="text-body-secondary fw-light">/за раціон</small></h1>
            <ul class="list-unstyled mt-3 mb-4">
```

Вибір страви зі сніданкового меню

Чай або кава на вибір

Додаткові опції для кави

Замовити

<h4 class="my-0 fw-normal">Обід</h4>

<h1 class="card-title pricing-card-title">₴150<small class="text-body-secondary fw-light">/за раціон</small></h1>

<ul class="list-unstyled mt-3 mb-4">

Перша та основна страви

Чай на вибір

Вибор хлібу

Замовити

```
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card mb-4 rounded-3 shadow-sm border-primary">
        <div class="card-header py-3 text-bg-primary border-primary">
          <h4 class="my-0 fw-normal">Вечеря</h4>
        </div>
        <div class="card-body">
          <h1 class="card-title pricing-card-title">₴390<small class="text-body-secondary fw-light">/за раціон</small></h1>
          <ul class="list-unstyled mt-3 mb-4">
            <li>Дві основні страви</li>
            <li>Напій на вибір</li>
            <li>Десерт дня</li>
          </ul>
          <a href="{{ url_for('item_buy', id=9, user_id=current_user.id) }}" class="w-100 btn btn-lg btn-outline-primary">Замовити</a>
          <br>
        </div>
      </div>
    </div>
  </div>
```

```
  </main>


</div>

{% endblock %}
```

ДОДАТОК Д

КОД ФАЙЛУ REVIEWS.HTML

```
{% extends 'base.html' %}


{% block title %}
Відгуки
{% endblock %}


{% block body %}
<h1>Залиште відгук</h1>
{% if current_user.is_authenticated %}
<form id="reviewForm" method="post" action="/reviews">
    <textarea name="text" id="text" class="form-control" rows="4"
placeholder="Ваш відгук"></textarea><br>
    <input type="submit" class="btn btn-success" value="Відправити">
</form>
{% else %}
<p>Увійдіть до облікового запису, щоб залишити відгук.</p>
<a class="btn btn-primary" href="{{ url_for('login') }}">Увійти</a>
{% endif %}


<div>
    {% for review in reviews %}
```

```
<div class="review" style="position: relative; padding: 10px; border: 1px solid #ccc; margin-bottom: 10px;">
    <p id="review-text-{{ review.id }}">{{ review.text }}</p>
    <small>
        Створено: {{ review.date.strftime('%Y-%m-%d %H:%M:%S') }}
        {% if review.last_edited %}
        <br>Відредаговано: {{ review.last_edited.strftime('%Y-%m-%d %H:%M:%S') }}
        {% endif %}
    </small>
    {% if current_user.is_authenticated and current_user.id == review.user_id %}
    <div style="position: absolute; top: 10px; right: 10px;">
        <div class="dropdown">
            <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton{{ review.id }}" data-bs-toggle="dropdown" aria-expanded="false">
                &#x22EE;
            </button>
            <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton{{ review.id }}">
                <li><a class="dropdown-item" href="#" onclick="openEditModal({{ review.id }})">Редагувати</a></li>
                <li><a class="dropdown-item" href="#" onclick="deleteReview({{ review.id }})">Видалити</a></li>
            </ul>
```

```
            </div>
        </div>
        {% endif %}
    </div>
  {% endfor %}
</div>

<!-- Модальне вікно для редагування -->
<div class="modal fade" id="editReviewModal" tabindex="-1" aria-labelledby="editReviewModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="editReviewModalLabel">Редагувати відгук</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <textarea id="editReviewText" class="form-control" rows="4"></textarea>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Відміна</button>
```

```
            <button type="button" class="btn btn-success"
onclick="saveEdit()">Зберегти</button>
        </div>
    </div>
  </div>
</div>

<script>
  document.getElementById('reviewForm').addEventListener('submit',
function(event) {
    event.preventDefault(); // Запобігаємо відправці форми
    const formData = new FormData(this);
    fetch('/reviews', {
      method: 'POST',
      body: formData
    })
    .then(response => response.json())
    .then(data => {
      if (data.success) {
        location.reload(); // Перезавантаження сторінки після успішного
додавання відгуку
      } else {
        alert(data.message || 'Помилка при додаванні відгуку');
      }
```

```
        })
        .catch(error => {
            console.error('Помилка:', error);
        });
    });

    let currentReviewId;

    function openEditModal(id) {
        currentReviewId = id;
        const reviewText = document.getElementById(`review-text-${id}`).innerText;
        document.getElementById('editReviewText').value = reviewText;
        const editReviewModal = new
bootstrap.Modal(document.getElementById('editReviewModal'));
        editReviewModal.show();
    }

    function saveEdit() {
        const newText = document.getElementById('editReviewText').value;
        fetch(`/edit_review/${currentReviewId}`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-form-urlencoded'
```

```
            },
            body: `text=${encodeURIComponent(newText)}`
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                location.reload(); // Перезавантаження сторінки після успішного редагування відгуку
            } else {
                alert('Помилка при редагуванні відгуку');
            }
        })
        .catch(error => {
            console.error('Помилка:', error);
        });
    }

    function deleteReview(id) {
        if (confirm('Ви впевнені, що хочете видалити цей відгук?'))
            {
            fetch(`/delete_review/${id}`, {
                method: 'POST'
            })
```

```
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    location.reload(); // Перезавантаження сторінки після успішного видалення відгуку
                } else {
                    alert('Помилка при видаленні відгуку');
                }
            })
            .catch(error => {
                console.error('Помилка:', error);
            });
        }
    }
</script>
{% endblock %}
```

ДОДАТОК Е

КОД ФАЙЛУ LOGIN.HTML

```
{% extends "base.html" %}


{% block title %}
Login
{% endblock %}


{% block body %}
<h1>Login</h1>
<form method="POST">
    {{ form.hidden_tag() }}
    <div class="form-group">
        {{ form.email.label }}<br>
        {{ form.email(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.password.label }}<br>
        {{ form.password(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.submit(class="btn btn-primary") }}
```

```
    </div>
</form>
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        {% for category, message in messages %}
            <div class="alert alert-{{ category }}">{{ message }}</div>
        {% endfor %}
    {% endif %}
{% endwith %}
{% endblock %}
```

ДОДАТОК Ж

КОД ФАЙЛУ REGISTER.HTML

```
{% extends "base.html" %}


{% block title %}
Register
{% endblock %}


{% block body %}
<h1>Register</h1>
<form method="POST">
    {{ form.hidden_tag() }}
    <div class="form-group">
        {{ form.username.label }}<br>
        {{ form.username(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.email.label }}<br>
        {{ form.email(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.password.label }}<br>
```

```
        {{ form.password(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.confirm_password.label }}<br>
        {{ form.confirm_password(size=32, class="form-control") }}
    </div>
    <div class="form-group">
        {{ form.submit(class="btn btn-primary") }}
    </div>
</form>
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        {% for category, message in messages %}
            <div class="alert alert-{{ category }}">{{ message }}</div>
        {% endfor %}
    {% endif %}
{% endwith %}
{% endblock %}
```

ДОДАТОК И

КОД ФАЙЛУ DELETE_ACCOUNT.HTML

```
{% extends 'base.html' %}


{% block title %}
Видалення облікового запису
{% endblock %}


{% block body %}
<h1>Видалення облікового запису</h1>
<form method="POST">
    <div class="form-group">
        <label for="password">Введіть ваш пароль для підтвердження:</label>
        <input type="password" name="password" class="form-control" id="password" required>
    </div>
    <button type="submit" class="btn btn-danger">Видалити обліковий запис</button>
</form>
{% endblock %}
```

ДОДАТОК К

КОД ФАЙЛУ ORDERS.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Заказы</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
    <style>
        body {
            display: flex;
            flex-direction: column;
            min-height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }
        h1 {
            text-align: center;
            margin: 20px 0;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
        }
        th, td {
            border: 1px solid #ddd;
```

```
        padding: 8px;
        text-align: center;
    }
    th {
        background-color: #f2f2f2;
    }
    form {
        display: inline-block;
    }
    select, button {
        margin: 5px 0;
    }
</style>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function() {
        $('form.update-status').on('submit', function(event) {
            event.preventDefault();
            var $form = $(this);
            $.ajax({
                type: $form.attr('method'),
                url: $form.attr('action'),
                data: $form.serialize(),
                success: function(response) {
                    if (response.success) {
                        location.reload();
                    } else {
                        alert('Помилка: ' + response.message);
                    }
```

```
                },
                error: function() {
                    alert('Сталася помилка під час оновлення статусу');
                }
            });
        });

        $('form.delete-order').on('submit', function(event) {
            event.preventDefault();
            var $form = $(this);
            $.ajax({
                type: $form.attr('method'),
                url: $form.attr('action'),
                data: $form.serialize(),
                success: function(response) {
                    if (response.success) {
                        location.reload();
                    } else {
                        alert('Помилка: ' + response.message);
                    }
                },
                error: function() {
                    alert('Виникла помилка при видаленні замовлення');
                }
            });
        });
    });
    </script>
</head>
```

```
<body>
    <h1>Замовлення</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Користувач</th>
                <th>Товар</th>
                <th>Дата</th>
                <th>Статус</th>
                <th>Управління</th>
            </tr>
        </thead>
        <tbody>
            {% for order in orders %}
            <tr>
                <td>{{ order.id }}</td>
                <td>{{ order.user.username }}</td>
                <td>{{ order.item.name }}</td>
                <td>{{ order.date }}</td>
                <td>{{ order.status }}</td>
                <td>
                    <form class="update-status" action="{{ url_for('update_status',
id=order.id) }}" method="post">
                        <select name="status">
                            <option value="очікування" {% if order.status == 'очікування'
%}selected{% endif %}>очікування</option>
                            <option value="виконується" {% if order.status == 'виконується'
%}selected{% endif %}>виконується</option>
```

```
                    <option value="завершено" {% if order.status == 'завершено' %}selected{% endif %}>завершено</option>
                </select>
                <button type="submit">Оновити статус</button>
            </form>
            <form class="delete-order" action="{{ url_for('delete_order', id=order.id) }}" method="post">
                <button type="submit">Видалити</button>
            </form>
        </td>
    </tr>
    {% endfor %}
  </tbody>
 </table>
</body>
</html>
```