

WebX Project

Restaurant Ordering

Web Application

Using Flask and MongoDB

Submitted by: Himesh Pathai

(Roll No: 34)

Division: D15A

Problem Statement

- Create a simple and efficient blog web application where users can:
- Read, search, and manage blog posts.
- Provide a secure admin panel for CRUD operations.
- Securely store data in MongoDB.

Introduction

- Web-based blogging platform using Flask and MongoDB.
- Admins can create, update, or delete blog posts.
- Users can read and search posts.
- Acts as a mini CMS (Content Management System).

Objectives

- Build a dynamic and user-friendly blog system.
- Allow only registered admins to manage blog posts.
- Store data securely in MongoDB.
- Implement a search feature.
- Learn and apply real-world web development skills.

Methodology

- Flask Setup for routing and templates.
- MongoDB Integration using PyMongo.
- User Authentication and Session Management.
- Admin Dashboard for CRUD operations.
- Search functionality.
- UI Design with HTML/CSS (Bootstrap).
- Testing and Debugging.

Tools and Technologies Used

- Python
- Flask
- MongoDB
- HTML/CSS
- VS Code
- Jinja2

Features

- User Registration and Login System.
- Admin Dashboard for blog management.
- Create, Edit, and Delete blog posts.
- Search functionality.
- Responsive and simple UI.

Folder Structure

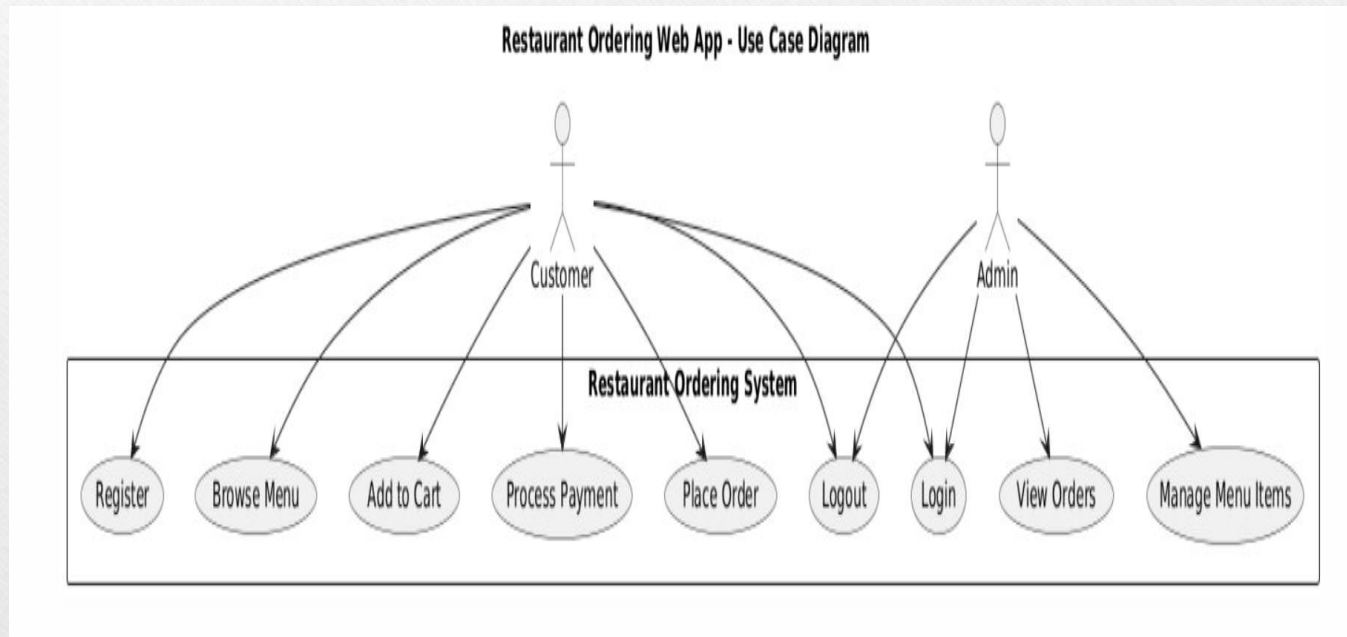
- `__pycache__`/: Compiled Python bytecode.
- `screenshots`/: Project images.
- `static`/: CSS, JavaScript, and image files.
- `templates`/: HTML templates.
- `user`/: User authentication and profiles.
- `app.py`: Main application.
- `stripe_logic.py`: Payment integration.
- `test.py`: Testing scripts.

Working of the Project

- Home Page: Browse posts without logging in.
- User Authentication: Register or login to post or manage content.
- Admin Dashboard: CRUD operations on blog posts.
- Payment Integration: Secure payments using Stripe.
- Profile Management: Update user profiles.

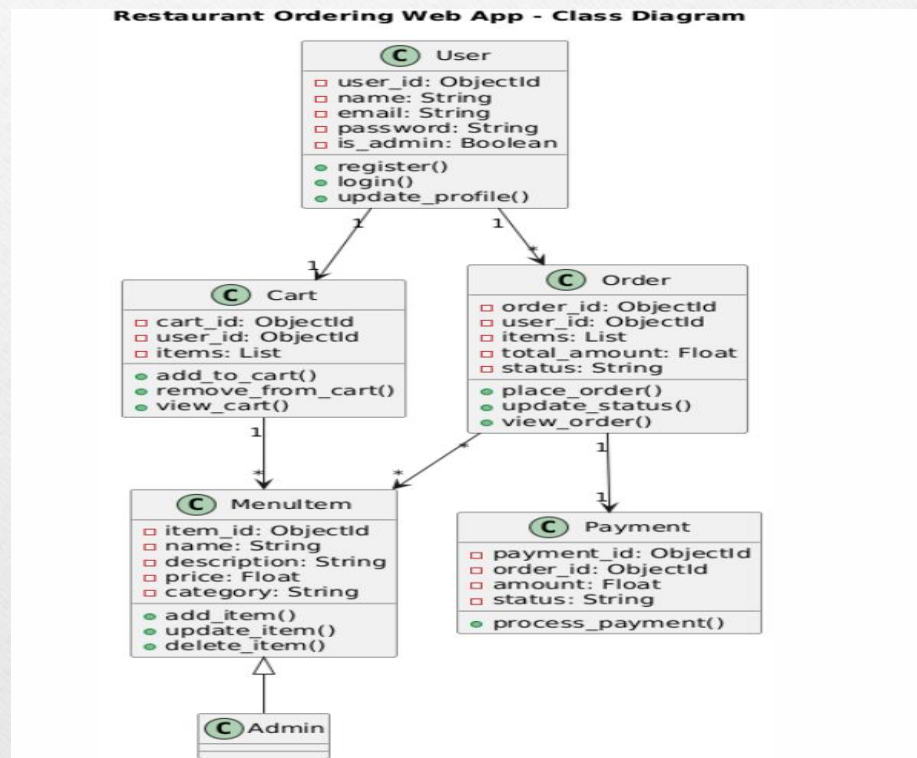
UML Diagrams

- Use Case Diagram



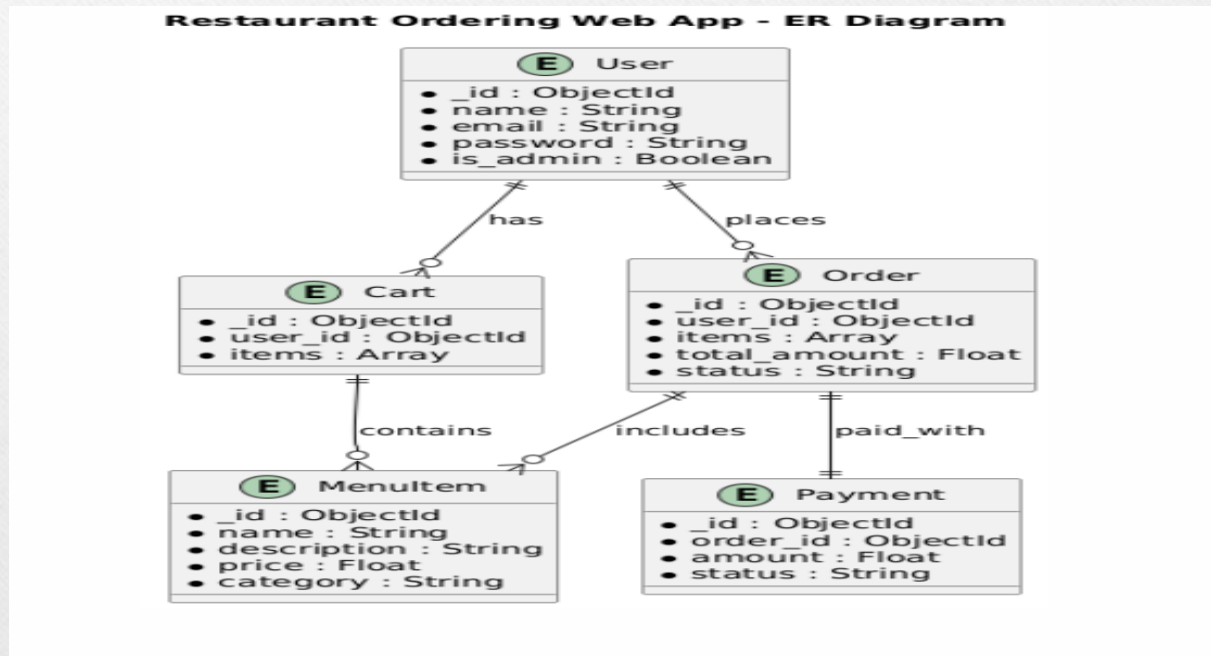
UML Diagrams

- Class Diagram



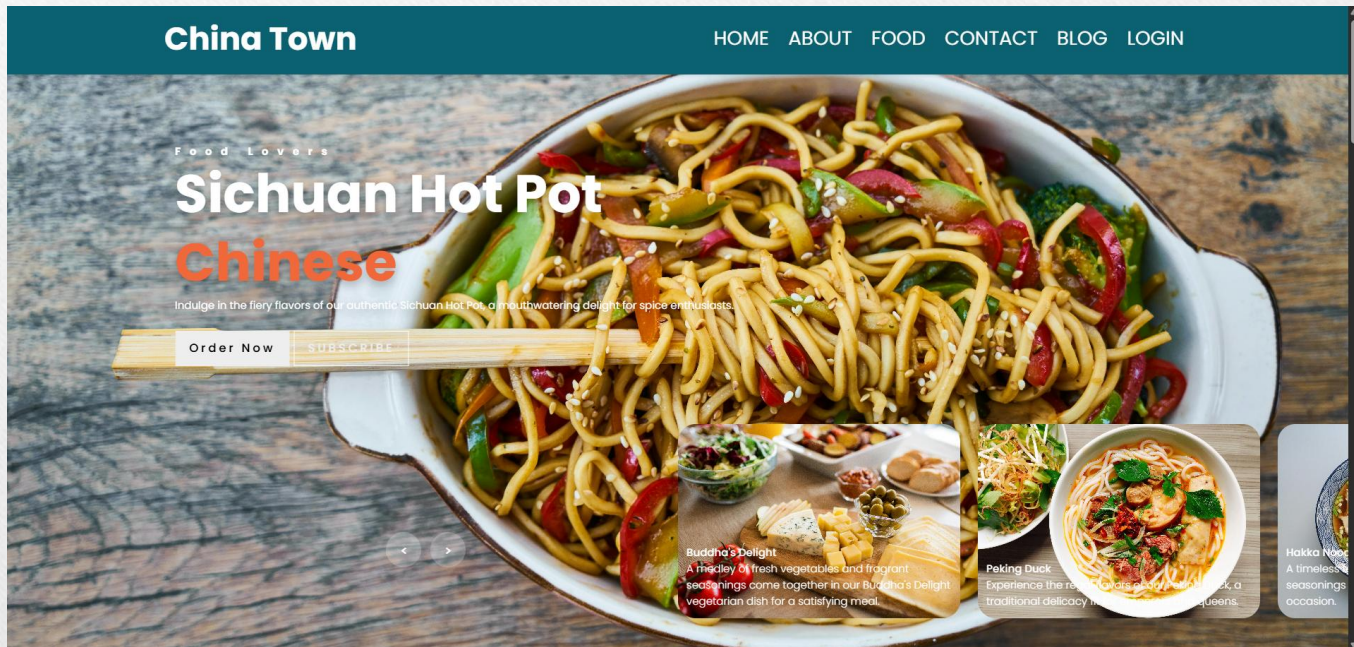
UML Diagrams

- ER Diagram



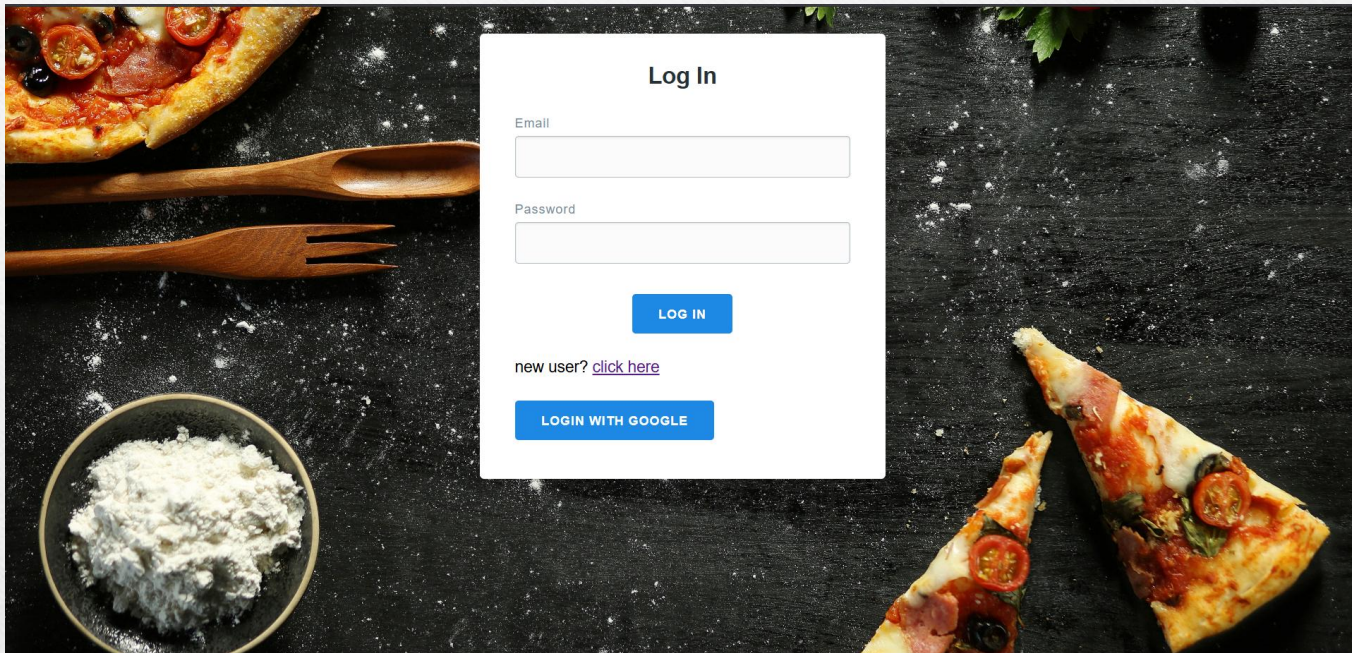
Screenshots

- Home Page



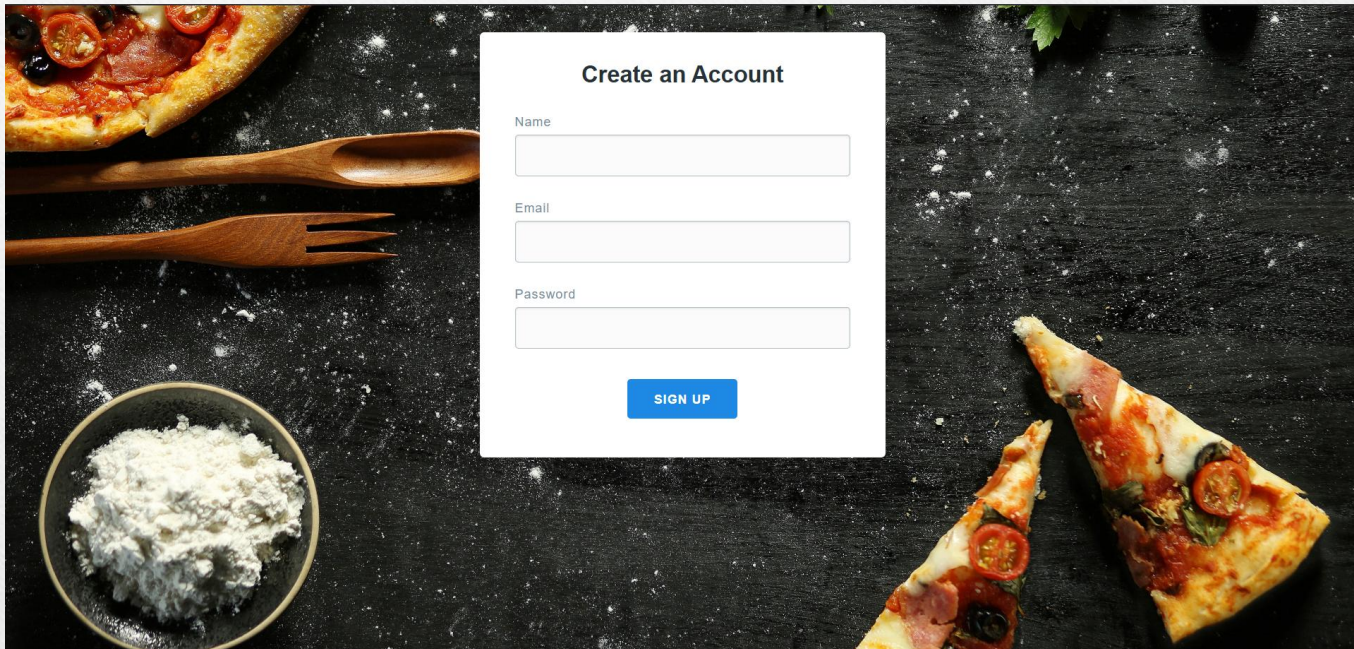
Screenshots

- Login Page



Screenshots

- Signup Page



The background image shows a dark, textured surface with a pizza, a wooden fork and spoon, and a bowl of white cheese. The 'Create an Account' form is centered and contains the following elements:

Create an Account

Name

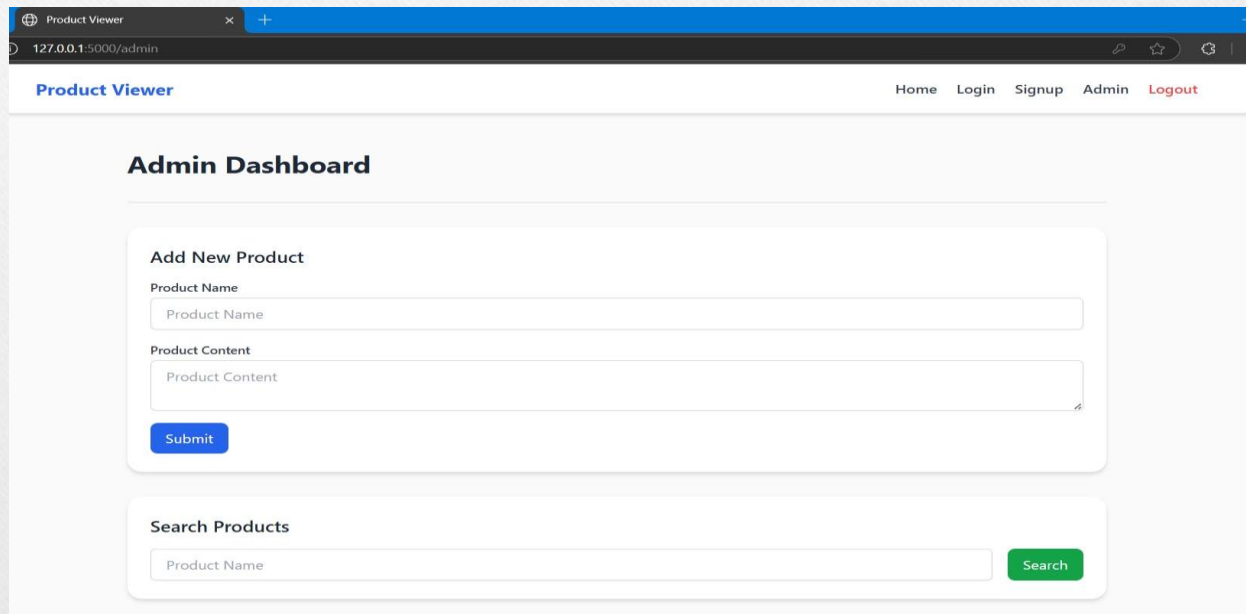
Email

Password

SIGN UP

Screenshots

- Admin Dashboard



The screenshot displays a web browser window with the title 'Product Viewer' and the URL '127.0.0.1:5000/admin'. The browser's address bar and tabs are visible at the top. Below the browser window, the 'Admin Dashboard' is shown. It features a navigation bar with links for 'Home', 'Login', 'Signup', 'Admin', and 'Logout'. The main content area is titled 'Admin Dashboard' and contains two sections: 'Add New Product' and 'Search Products'. The 'Add New Product' section includes a 'Product Name' input field, a 'Product Content' input field, and a 'Submit' button. The 'Search Products' section includes a 'Product Name' input field and a 'Search' button.

Product Viewer

127.0.0.1:5000/admin

Product Viewer

Home Login Signup Admin Logout

Admin Dashboard

Add New Product

Product Name

Product Content

Submit

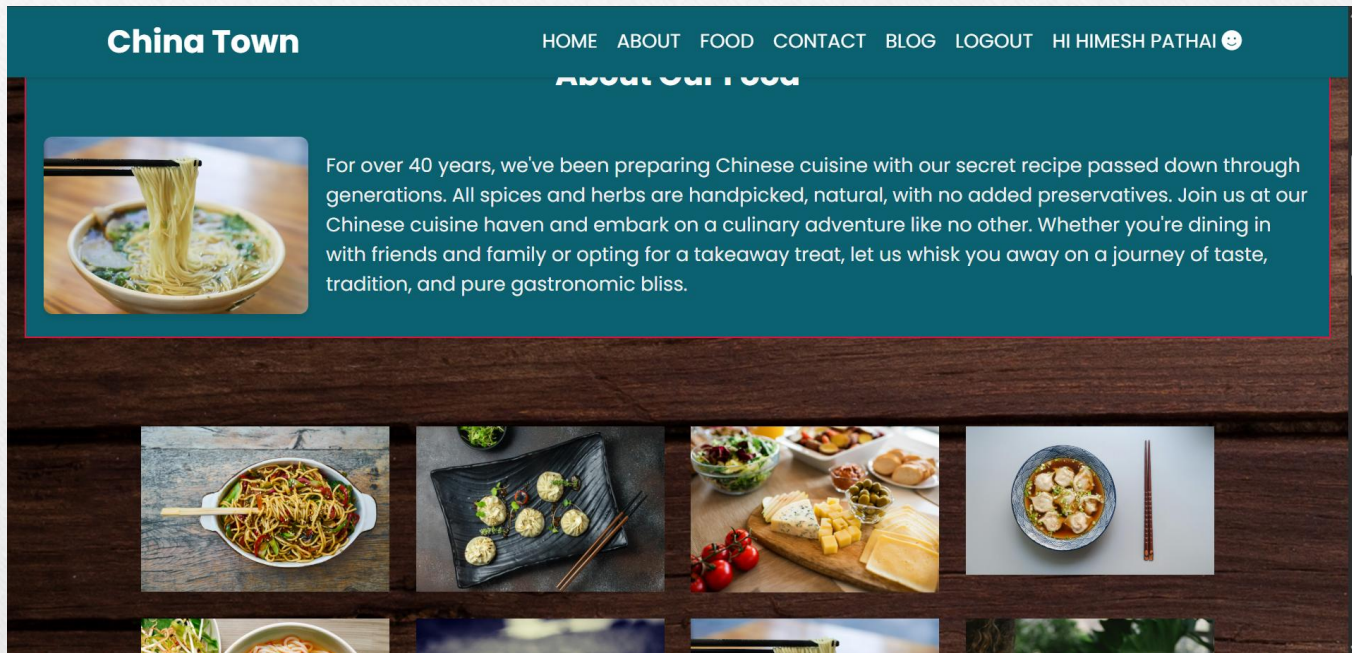
Search Products

Product Name

Search

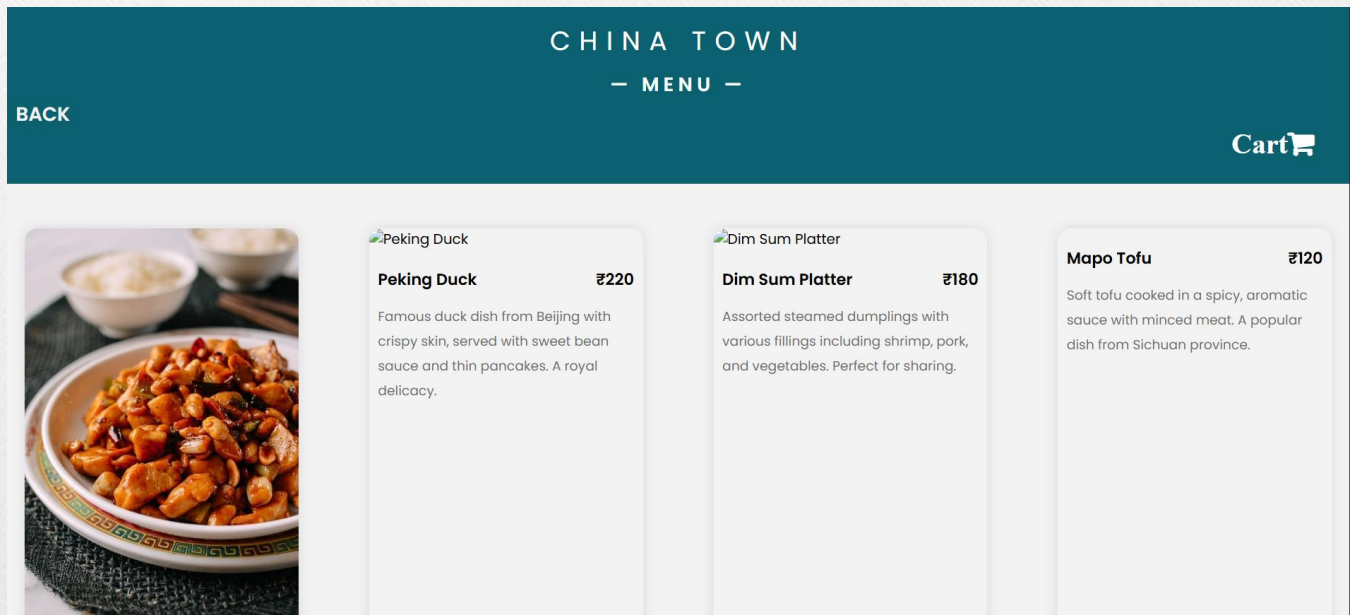
Screenshots

- About Us Page



Screenshots

- Food Menu Page



Sample Code Snippets

- MongoDB Connection:
- `from pymongo import MongoClient`
- `client = MongoClient("mongodb://localhost:27017/")`
- `db = client["blog_app"]`

- Flask Route to Create Post:
- `@app.route('/create', methods=['GET', 'POST'])`
- `def create_post():`
- `if request.method == 'POST':`
- `title = request.form['title']`
- `content = request.form['content']`
- `db.posts.insert_one({'title': title, 'content': content})`
- `return redirect(url_for('admin'))`

Challenges Faced

- Flask to MongoDB connection.
- Session management.
- Dynamic content rendering.
- CRUD operations with form validation.

Conclusion

- Gained hands-on experience with Flask and MongoDB.
- Understood the full-stack development workflow.
- Strengthened knowledge of web app development, data management, and secure authentication.

Future Improvements

- - Add comments section.
- - Image uploads.
- - UI enhancement with Tailwind CSS.
- - Email verification.
- - JWT authentication.

Thank You!