RESTAURANT FOOD ORDERING

WEBSITE USING MEAN STACK

Project Guidelines

Team Members:

1.Chhayani Daksh (2203051051062)

2.Balar Om (2203051050389)

3.Utsav Sangani (2203051050597)

4.Deep Mehta (2203051050147)

5.Rushabh Mistry (2203051050496)

Introduction

Restaurant Food Ordering System Project Using MEAN (MongoDB, Express.js, Angular, Node.js) Stack

---

1. Purpose

The purpose of this project is to develop a **Restaurant Food Ordering System** that allows customers to easily browse the menu, place orders, make payments online, and track the status of their orders, all through a web-based application built using the MEAN stack (MongoDB, Express.js, Angular, and Node.js). This system aims to improve customer convenience, enhance restaurant operations, and provide a secure and efficient platform for food ordering.

---

2. Background

In traditional restaurants, ordering food usually involves physical interaction, leading to possible errors, slower service, and longer wait times. With the rise of digital transformation, restaurants are increasingly moving towards online platforms to offer a more seamless and efficient customer experience. Using the MEAN stack, a full-stack JavaScript solution, enables building a modern, fast, and scalable web application for such use cases.

The **MEAN stack** offers several benefits:

- **MongoDB** for flexible, scalable data storage.
- **Express.js** as a lightweight web framework for handling HTTP requests.
- **Angular** for building dynamic, responsive, and user-friendly frontend interfaces.
- **Node.js** for creating fast and scalable server-side applications.

---

3. Scope

This project will focus on developing a **Restaurant Food Ordering System** with the following functionalities:

- **Customer functionalities:**
  - User authentication (login/registration).
  - View restaurant menu, items, and prices.
  - Add items to the cart and customize orders.
  - Place orders and proceed to secure payment.
  - Track order status (e.g., Pending, In-Progress, Completed).
- **Admin functionalities:**
  - Admin login and authentication.
  - Manage and update restaurant menu items (add, remove, update).
  - View and manage orders (mark as completed, etc.).

- Generate order and payment reports.

This system will be designed to handle a **single restaurant**, but with the potential to scale and adapt for multiple restaurants in future implementations.

---

4. Problem Statement

Many restaurants still rely on outdated methods of taking orders (phone calls, in-person). This leads to common issues such as:

- Miscommunication between customers and staff.
- Long waiting times and customer frustration.
- Errors in the orders placed.
- Lack of efficiency in managing orders and updating the menu.

There is a clear need for a digital solution that streamlines the ordering process, reduces errors, and enhances the overall experience for both customers and staff.

---

5. Solution

The solution is a **Restaurant Food Ordering System** built using the **MEAN stack**, which will:

- Provide a **web-based platform** where customers can browse the menu, place and customize orders, and make secure payments.
- Enable **real-time order tracking** so that customers can monitor their orders' status (pending, in progress, completed).
- Allow restaurant **admins to manage menus, orders, and payments** through a secure admin panel.
- Integrate a **payment gateway** (e.g., Stripe, PayPal) to allow for online payments.
- Ensure scalability, security, and ease of use for both customers and restaurant staff.

---

6. Objectives

The main objectives of the project are:

1. **Develop a user-friendly platform** for customers to easily browse the menu, order food, and make payments online.
2. **Implement secure user authentication** for both customers and restaurant staff.
3. **Create a real-time order tracking system** to allow customers and restaurant staff to monitor the status of orders.

4. **Build an admin dashboard** to manage restaurant operations such as updating the menu, processing orders, and generating reports.
5. **Ensure the system is scalable** to support future expansion to multiple restaurants or locations.

---

7. Methodology

7.1 Data Collection

- **Customer Feedback:** Conduct surveys or interviews with restaurant customers to understand their expectations and preferences regarding the food ordering system.
- **Restaurant Requirements:** Gather requirements from restaurant staff to understand how the current ordering process works and identify inefficiencies.
- **Market Research:** Analyze existing food ordering systems to identify features, pain points, and opportunities for improvement.

7.2 Tools and Technologies

- **Frontend (Angular):** Angular 15+ for building a dynamic and responsive user interface.
- **Backend (Node.js/Express.js):** Node.js for building the server-side application and Express.js for handling routes and API requests.
- **Database (MongoDB):** MongoDB for storing menu items, orders, customer details, and payment records.
- **Authentication:** JWT (JSON Web Tokens) for securing authentication for customers and admins.
- **Payment Gateway:** Integration with services like Stripe or PayPal for processing online payments.
- **Real-time updates:** Socket.io or Firebase for providing real-time order updates to both customers and staff.

7.3 Implementation Process

1. **Requirement Analysis:** Define the system requirements and user stories for both customer and admin functionalities.
2. **System Design:** Design the architecture of the system, including the database schema (MongoDB collections), API endpoints, and UI wireframes.
3. **Frontend Development:** Build the customer-facing frontend using Angular, including pages for menu browsing, cart management, order placement, and payment.
4. **Backend Development:** Set up the backend using Node.js and Express.js. Implement APIs for order management, menu management, and user authentication.
5. **Payment Integration:** Integrate a secure payment gateway (e.g., Stripe).
6. **Admin Panel:** Create an admin dashboard to manage orders, update menu items, and view reports.
7. **Testing:** Conduct unit tests, integration tests, and user acceptance testing (UAT) to ensure the system functions as expected.
8. **Deployment:** Deploy the system to a cloud platform (e.g., AWS, Heroku) for public access.
9. **Documentation:** Document the code, system architecture, and user manuals for both customers and admins.

---

8. Project Plan

8.1 Timeline

- **Week 1-2:** Requirement gathering, design UI/UX, and create system architecture.
- **Week 3-5:** Frontend development (Angular), setting up Node.js/Express backend.
- **Week 6-7:** Integration of MongoDB, APIs, and payment gateway.
- **Week 8-9:** Admin panel development and real-time order tracking.
- **Week 10:** Testing (unit tests, UAT, and integration tests).
- **Week 11:** Deployment and final documentation.
- **Week 12:** Final presentation and review.

8.2 Resources Needed

- **Frontend Developer** for Angular development.
- **Backend Developer** for Node.js/Express and MongoDB integration.
- **UI/UX Designer** for designing the web interface.
- **Testing Tools:** Postman for API testing, Mocha or Jest for unit testing.
- **Cloud Hosting:** AWS, Heroku, or any preferred cloud platform for deployment.

---

9. Expected Deliverables

1. **Web Application** (Restaurant Food Ordering System).
2. **Admin Dashboard** for restaurant staff to manage orders and menu.
3. **Database Schema** and ER diagram for the MongoDB setup.
4. **Payment Integration** setup (Stripe/PayPal).
5. **Codebase**: Well-commented source code with detailed documentation.
6. **User Manuals** for both customers and admins.
7. **Test Reports**: Unit and integration test results.
8. **Final Project Report**: Covering system design, architecture, and implementation details.
9. **Demo Video**: A walkthrough of the system showcasing features.

---

10. Evaluation Criteria

The project will be evaluated based on the following criteria:

1. **Functionality**: Does the system meet the core requirements of the project (menu browsing, order placement, payment processing, and order tracking)?
2. **Usability**: Is the user interface easy to navigate and intuitive for both customers and restaurant staff?
3. **Performance**: Does the system work efficiently even under heavy traffic? Are real-time updates provided in a timely manner?

4. **Security**: Is user data (e.g., payment information) handled securely? Is authentication properly implemented?
5. **Scalability**: Can the system scale to handle multiple restaurants in the future?
6. **Code Quality**: Is the code well-structured, modular, and maintainable?
7. **Testing**: Has sufficient testing been done to ensure reliability and correctness?

---

11. Submission Guidelines

1. **Source Code**: Submit the complete source code with proper documentation.
2. **Project Report**: A detailed report outlining the project objectives, design, implementation, and testing.
3. **Demo Video**: A short video demonstrating the functionality of the system.
4. **Final Presentation**: A slide deck summarizing the project's goals, development process, and outcomes.
5. **Deployment Link**: A live demo link to the deployed application (e.g., on Heroku or AWS).

---

12. References

1. **Books:**
   - *"Learning Angular"* by Brad Green and Shyam Seshadri.
   - *"Node.js Design Patterns"* by Mario Casciaro.
   - *"MongoDB: The Definitive Guide"* by Kristina Chodorow.
2. **Websites:**
   - MongoDB Documentation
   - Angular Official Documentation
   - Node.js Official Documentation
   - Stripe API Documentation
   - [Express.js