Web Development for a Restaurant Management System

AN INTERNSHIP REPORT

Submitted by

AFIYA SAINU T S

(RRN: 220071601016)

In partial fulfilment for the award of the degree of

B.Tech

In

COMPUTER SCIENCE AND ENGINEERING

B.S. Abdur Rahman® Crescent Institute of Science & Technology

Deemed to be University u/s 3 of the UGC Act, 1956

DECEMBER 2024

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

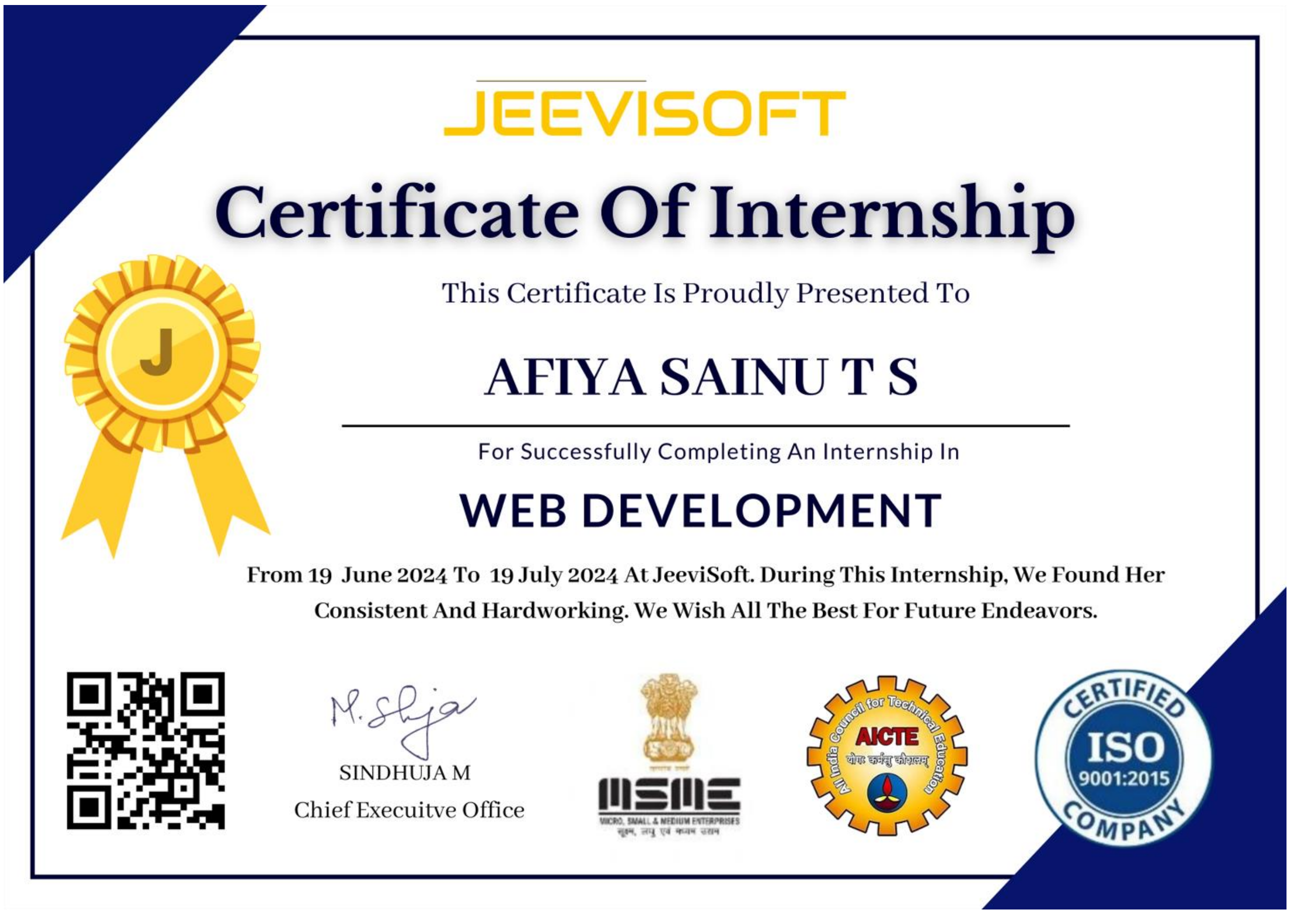
BONAFIDE CERTIFICATE

Certified that this Internship report “WEB DEVELOPMENT FOR A RESTAURANT MANAGEMENT SYSTEM ” is the bonafide work of “AFIYA SAINU T S (220071601016)” who carried out the internship under my supervision. Certified further, that to the best of our knowledge the work reported herein does not form part of any other internship report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr. V. BALAJI
Assistant Professor
Department of CSE
B.S. Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600 048.

INTERNSHIP CERTIFICATE



JEEVISOFT

Certificate Of Internship

This Certificate Is Proudly Presented To

AFIYA SAINU T S

For Successfully Completing An Internship In

WEB DEVELOPMENT

From 19 June 2024 To 19 July 2024 At JeeviSoft. During This Internship, We Found Her Consistent And Hardworking. We Wish All The Best For Future Endeavors.

[illegible]

SINDHUJA M
Chief Execuitve Office

J
MSME
MICRO, SMALL & MEDIUM ENTERPRISES
All India Council for Technical Education
AICTE
CERTIFIED
ISO
9001:2015
COMPANY

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

VIVA VOCE EXAMINATION

The viva voce examination of the Internship titled **“FULL STACK WEB DEVELOPMENT”,** submitted by **AFIYA SAINU T S (220071601016)** is held on ________________.

INTERNAL EXAMINER

ACKNOWLEDGEMENT

I sincerely express my heartfelt gratitude to **Prof. Dr. T. MURUGESAN**, Vice Chancellor and **Dr. N. THAJUDDIN**, Pro-Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology, for providing me an environment to carry out my internship successfully.

I sincerely thank **Dr. N. RAJA HUSSAIN**, Registrar, for furnishing every essential facility for doing my internship.

I thank **Dr. SHARMILA SANKAR**, Dean, School of Computer Information and Mathematical Sciences for her motivation and support.

I thank **Dr. W. AISHA BANU**, Professor and Head, Department of Computer Science and Engineering, for providing strong oversight of vision, strategic direction and valuable suggestions.

I obliged my class advisor **Mr. V BALAJI,** Assistant Professor, Department of Computer Science and Engineering for his professional guidance and continued assistance throughout the internship period.

I thank **K PRAVEEN KUMAR**, project manager, Plasmid Innovation Ltd for his guidance and encouragement throughout the internship period and without his guidance this project would not have been a successful one.

I thank all the **Faculty members** and the **System Staff** of the Department of Computer Science and Engineering for their valuable support and assistance at various stages of internship completion.

(AFIYA SAINU T S)

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

S. NO.	ABBREVIATION	DESCRIPTION
1.	UI	User Interface
2.	UX	User Experience
3.	JWT	JSON Web Token
4.	RBAC	Role-Based Access Control
5.	XSS	Cross-Site Scripting
6.	CORS	Cross-Origin Resource Sharing
7.	API	Application Programming Interface
8.	HTML	HyperText Markup Language
9.	CSS	Cascading Style Sheets
10.	HTTPS	Hypertext Transfer Protocol Secure
11.	AWS	Amazon Web Services

LIST OF FIGURES

ABSTRACT

This internship project presents a comprehensive web-based Restaurant Management System developed using full-stack web technologies to optimize operations and enhance customer engagement. The system is implemented using the MERN (MongoDB, Express.js, React.js, Node.js) stack for backend and frontend development, along with HTML, CSS, and JavaScript for creating a dynamic and intuitive user interface.

The platform is designed to streamline essential restaurant functionalities, enabling users to explore an interactive digital menu, make table reservations with real-time availability, provide feedback to improve service quality, and access contact information for inquiries. The backend, powered by Node.js and Express.js, ensures efficient handling of data operations, while MongoDB offers a scalable database solution. React.js forms the foundation of the responsive front end, providing a seamless user experience across various devices.

This system demonstrates a robust application of full-stack development principles, integrating modern web technologies to deliver a solution that addresses the practical needs of restaurant operations while improving customer satisfaction.

CHAPTER 1

COMPANY OVERVIEW

Jeevisoft Solutions Pvt. Ltd

Fig 1: jeevisoft logo

1.1 COMPANY OVERVIEW

Jeevisoft Solutions Pvt. Ltd. is a dynamic IT solutions provider established with a vision to empower businesses through innovative and efficient software solutions. Headquartered in Chennai, India, the company specializes in crafting cutting-edge technology solutions tailored to meet diverse industry needs. With a commitment to quality, user-centric design, and technological innovation, Jeevisoft has emerged as a trusted partner for businesses seeking comprehensive software systems and IT solutions.

Vision & Mission

- **Vision**: To revolutionize industries with intuitive, reliable, and scalable technology solutions that drive growth and customer satisfaction.
- **Mission**: To deliver value-driven software solutions by combining innovation, expertise, and exceptional customer service.

Service Portfolio

Jeevisoft Solutions offers a diverse range of products and services designed to streamline operations and enhance efficiency across industries:

1. **Point of Sale (POS) Systems**

   - User-friendly POS systems for seamless transactions.

2. **Integration with payment gateways and loyalty programs to enhance customer convenience.**
3. **Custom Software Development**
   - Tailored software solutions to address unique business requirements.
   - Expertise in integrating emerging technologies to optimize operational efficiency.
4. **Customer Relationship Management (CRM)**
   - Tools to manage customer feedback, preferences, and engagement strategies.
   - Features that promote personalized marketing campaigns and loyalty initiatives.
5. **Inventory and Supply Chain Management**
   - Comprehensive systems to monitor and manage inventory levels and streamline supply chains.
   - Real-time tracking and analytics for improved decision-making.
6. **Enterprise Solutions**
   - Scalable and robust solutions to meet the operational demands of large-scale businesses.
   - Emphasis on system integration and process automation for enhanced productivity.

Market Segments

Jeevisoft serves a wide array of industries, including:

- **Retail:** Tools to optimize sales, inventory management, and customer engagement.

- **Healthcare:** Solutions for efficient patient data management and operational support.
- **Manufacturing:** Systems to enhance productivity, manage resources, and track performance.
- **Education:** Software for learning management, student data tracking, and administrative efficiency.

1.2 COMPANY WEBSITE

Jeevisoft Solutions Pvt. Ltd. specializes in delivering innovative software and IT solutions tailored to the needs of businesses across various sectors. The company focuses on providing scalable, user-friendly, and reliable systems that drive operational efficiency and business growth.

Headquartered in Chennai, India, Jeevisoft serves clients globally, leveraging advanced technologies to deliver cutting-edge solutions. With a strong commitment to quality, innovation, and customer satisfaction, Jeevisoft continues to establish itself as a trusted technology partner.

For further details, you can visit their official website: Jeevisoft Solutions Pvt. Ltd.

CHAPTER 2

ROLES AND RESPONSIBILITIES

2.1 ROLES

1. **Frontend Developer:**
   - Assist in designing and developing the user interface (UI) for the restaurant management system using HTML, CSS, and JavaScript.
   - Ensure the frontend is responsive and visually appealing across different devices and screen sizes.
   - Integrate interactive elements for functionalities like menu viewing, table reservations, and customer feedback.
2. **Backend Developer:**
   - Work with Node.js, Express.js, and MongoDB to develop the backend functionalities of the system.
   - Implement RESTful APIs and handle the business logic for the system, including order management, reservation systems, and user authentication.
3. **Database Administrator:**
   - Design and maintain the database schema using MongoDB.
   - Handle CRUD operations related to menu items, reservations, customer feedback, and orders.
   - Optimize database queries and ensure the data integrity and scalability of the system.
4. **Quality Assurance:**
   - Test both frontend and backend components for functionality, performance, and security.
   - Ensure the system meets the required specifications and user needs by performing integration and unit testing.

- Identify, troubleshoot, and resolve issues that affect the user experience or system performance.

5. **Collaboration & Communication:**
    - Work closely with the development team and report regularly to the supervisor or mentor.
    - Participate in code reviews, contribute to team discussions, and assist in the implementation of improvements.
    - Coordinate with cross-functional teams to gather feedback and implement updates to the system.

Responsibilities:

1. **Frontend Development:**
    - Develop the user interface using HTML, CSS, and JavaScript, ensuring responsiveness and interactivity.
    - Work with React.js or similar JavaScript frameworks to implement dynamic features such as menu navigation, order placement, and feedback collection.
    - Ensure the frontend is user-friendly and consistent across all devices.
2. **Backend Development:**
    - Implement APIs and backend logic using Node.js and Express.js to manage restaurant-related data.
    - Handle business operations such as customer orders, table reservations, and customer feedback.
    - Set up and manage the server-side functionality, including authentication, session management, and routing.
3. **Database Management:**

- Design and manage the MongoDB database to store and retrieve data for menu items, customer details, orders, reservations, and feedback.
- Create efficient database queries and optimize performance to handle large-scale data.
- Perform database backups and ensure data security and integrity.

4. **Testing & Debugging:**
   - Perform unit testing and integration testing to verify that both the frontend and backend work as intended.
   - Debug issues in both the frontend and backend, ensuring that all system components function smoothly.
   - Ensure cross-browser compatibility and fix any issues related to different environments.
5. **Version Control & Collaboration:**
   - Use Git for version control, manage branches, and collaborate effectively with team members.
   - Participate in code reviews and contribute suggestions for improving the quality and efficiency of the code.
   - Communicate progress, challenges, and ideas with the team and mentor regularly.
6. **Deployment & Maintenance:**
   - Assist in deploying the web application to staging or production environments using tools such as AWS, Heroku, or similar cloud platforms.
   - Monitor the live application, ensuring uptime and troubleshooting issues that arise.
   - Collaborate with the team to implement updates and new features based on user feedback and system requirements.
7. **Documentation:**

- Document the development process, including code structure, APIs, and database schema.
- Write user guides and technical documentation to support system deployment, maintenance, and future updates.
- Maintain detailed records of the tasks completed, challenges faced, and solutions implemented.

8. **Learning & Development:**
    - Stay updated on the latest trends and technologies in full-stack development, including new tools and best practices.
    - Enhance skills in web development, databases, and cloud platforms through hands-on experience and learning opportunities.

This role provides a comprehensive learning experience in full-stack web development, involving both frontend and backend tasks while contributing to the development of a restaurant management system.

CHAPTER 3

PROJECT OVERVIEW

The Restaurant Management System aims to create a comprehensive software platform to manage all aspects of restaurant operations. It addresses the need for modern businesses to streamline day-to-day tasks such as managing the menu, processing orders, handling customer feedback, and reserving tables. The system is developed using full-stack technologies to ensure a scalable and maintainable solution.

Technologies and Tools

- Frontend: The user interface (UI) is developed using HTML, CSS, and JavaScript. The UI is further enhanced using React.js, which enables dynamic updates to the interface and improves user experience through seamless rendering. This is especially important for real-time features like table reservations and menu updates.
- Backend: The backend is built using Node.js and Express.js. Node.js is a runtime environment that allows JavaScript to run server-side, enabling fast and efficient handling of requests. Express.js is a framework for building RESTful APIs, which allows the frontend to communicate with the backend smoothly.
- Database: Data is stored and managed using MongoDB, a NoSQL database. MongoDB is chosen due to its scalability, flexibility, and ability to handle unstructured data, which is important for managing restaurant menus, customer orders, and feedback efficiently.

Key Features and Functionalities

1. Frontend Development

- User Interface Design:
  - The system's frontend is designed to be visually appealing and user-friendly. It's responsive, meaning it adapts to different screen sizes and devices, such as mobile phones, tablets, and desktops.

    - Aesthetic aspects like layout, typography, and color schemes are carefully chosen to create an inviting and functional interface for both customers and restaurant staff.
- Dynamic Interaction:
    - React.js is used to build the frontend, allowing real-time updates. For example, when a user selects a menu item, the page can dynamically update to show the selected item’s details (e.g., price, description) without reloading the page.

2. Backend Development

- API Design:
    - RESTful APIs are designed to handle different operations such as adding a reservation, updating menu items, or processing customer feedback. APIs act as a bridge between the frontend and the database.
    - These APIs are designed to be efficient, fast, and secure, ensuring smooth communication across the system.
- Authentication:
    - Secure login systems are implemented to ensure that only authorized users (e.g., customers, restaurant administrators) can access sensitive data.
    - Role-based access control is used, meaning administrators have more privileges, such as adding new menu items or reviewing customer feedback, while customers can only book reservations or leave feedback.

3. Database Management

- MongoDB Integration:
    - MongoDB stores all restaurant-related data, such as customer profiles, menu items, reservations, and feedback.
    - It uses collections (similar to tables in relational databases) to organize data. For instance, there might be collections for Menu Items, Customers, Reservations, and Feedback.
- Data Integrity:

    - MongoDB ensures that all data remains consistent and accurate. Operations like adding or updating menu items, making reservations, and submitting feedback are all handled in a manner that avoids data conflicts.

4. Core Functionalities

- Menu Management:
    - The system allows administrators to dynamically update the menu, adding or removing items as needed. Each menu item can include an image, description, and pricing information.
    - This functionality is critical for restaurants with frequent menu changes or specials.
- Table Reservation System:
    - Customers can view available tables and book them in real-time. The system will show table availability and update it in real-time to avoid double-booking.
    - The status of each table (reserved, available, or occupied) is displayed dynamically.
- Feedback System:
    - Customers can rate their dining experience and provide feedback. The system collects this data, which can be used by restaurant managers to improve services.
    - Administrators can access a dashboard to review feedback and analyze trends to identify areas for improvement.

5. Testing and Debugging

- Unit Testing:
    - Each component or module of the system (e.g., the reservation system, feedback system) is tested in isolation to ensure it functions correctly. Unit testing helps catch small bugs early.
- Integration Testing:
    - The system is tested as a whole to ensure that the frontend, backend, and database work seamlessly together. For example, when a

customer submits a reservation, the frontend should update correctly, and the backend should store the reservation data in the database.

6. Deployment and Maintenance

- Cloud Hosting:
    - Once the system is developed and tested, it will be deployed to a cloud hosting platform like AWS or Heroku. Cloud hosting ensures that the system can scale easily to handle varying loads (e.g., during peak dining times).
- Monitoring:
    - After deployment, continuous monitoring is conducted to ensure that the system runs smoothly. Performance metrics, user feedback, and error logs are regularly reviewed to ensure reliability.

7. Version Control and Collaboration

- Git Integration:
    - Version control with Git is used to track changes to the codebase. This is crucial for team collaboration, as it allows developers to work on different parts of the system without interfering with each other’s work.
    - GitHub or GitLab repositories are used to manage and collaborate on the project.

Project Outcomes and Achievements

By the end of the project, the Restaurant Management System will achieve the following objectives:

- A fully functional full-stack web application that integrates the frontend, backend, and database for seamless operation.
- Real-time functionalities that enhance customer experience, such as dynamic menu updates and live table reservation availability.
- A feedback system that helps restaurant administrators understand customer satisfaction and make improvements.

- Scalable and secure design that ensures both customers' and administrators' data are kept safe and the system can grow

CHAPTER 4

THEORETICAL ANALYSIS

This chapter delves into the architectural design, user interface considerations, security measures, scalability strategies, and testing protocols used to build and deploy the Restaurant Management System (RMS). Each section will provide a thorough understanding of how the system is structured to deliver an efficient, secure, and scalable web application.

4.1 Web Application Architecture

The **RMS** is built on a modern web application architecture that emphasizes **efficiency**, **scalability**, and **user interactivity**. Below are the primary layers of the architecture:

Frontend

- **HTML/CSS/JavaScript**: These are the core building blocks for structuring, styling, and adding interactivity to the frontend of the application.
    - **HTML** is used for creating the structure of the pages (e.g., headings, tables, buttons).
    - **CSS** provides styling to ensure the application is visually appealing and consistent across devices.
    - **JavaScript** handles dynamic behavior, such as responding to user actions (e.g., button clicks, form submissions) and updating the UI without requiring a page reload.
- **React.js**:
    - React.js is used to build the UI with reusable components that efficiently render changes. For example, a menu item or table availability can be dynamically updated using React, providing an interactive user experience.
    - React helps with performance by re-rendering only the components that have changed, making the app more efficient.

Backend

- **Node.js**:
    - Node.js is used to execute the server-side logic of the RMS. It is built on the **V8 JavaScript engine**, making it highly efficient for handling numerous simultaneous client requests in a non-blocking manner. This is important for real-time features like table reservations and menu updates.
- **Express.js**:
    - Express.js is a lightweight framework built on top of Node.js that simplifies routing and handling HTTP requests. It helps in creating RESTful APIs that interact with the frontend.
    - The **API endpoints** might include operations like:
        - POST /reservation: To book a table.
        - GET /menu: To retrieve the menu data.
        - POST /feedback: To submit customer feedback.
- **APIs**:
    - APIs serve as the intermediary between the frontend and backend, allowing data to be exchanged between the two systems. For example, the frontend may send a request to the API to fetch available tables or submit an order.

Database

- **MongoDB**:
    - MongoDB is a **NoSQL** database that stores data in a **document-oriented format** (JSON-like documents). It is highly flexible, making it an ideal choice for RMS because:
        - The data structure can easily evolve. For instance, if new menu features or customer information fields are added, it is easy to adjust the schema.

        - It stores **structured and semi-structured data**, such as menu items, customer details, reservations, and feedback.

4.2 User Interface and Experience

The success of RMS depends heavily on its **User Interface (UI)** and **User Experience (UX)**. The system is designed with the following principles:

Design Principles

- **Mobile-first approach**: This ensures that the application looks and works well across all devices, from desktops to smartphones. With more people using mobile devices for reservations and browsing, optimizing the experience for smaller screens is critical.
    - A responsive layout adapts the content to various screen sizes (e.g., columns stacking on smaller screens or images resizing for mobile).
- **Intuitive Navigation**:
    - The UI is designed to be user-friendly, with clear navigation options. Features like dropdown menus, search bars, and quick links improve usability.
    - The system's core functionality, such as viewing the menu, making reservations, or submitting feedback, is easily accessible and clearly labeled.

Interactivity

- **Dynamic Content Rendering**:
    - React.js enables dynamic content updates. For example, when a customer reserves a table, the frontend will update in real-time without reloading the entire page.
    - This creates a more seamless and fluid experience, where users can interact with the system continuously without interruptions.

4.3 Security and Authentication

Given the nature of the data (e.g., personal customer information, financial details), **security** is a primary concern for the RMS.

Authentication

- **JSON Web Tokens (JWT)**:
    - JWT is used for secure session management. When a user logs in, the server generates a token that stores user information (e.g., user ID, role). This token is sent to the client and included in future requests to validate the user's identity.
    - This ensures that only authorized users can access certain resources (e.g., admins can manage the menu, while customers can only make reservations).

Data Protection

- **Role-based Access Control (RBAC)**:
    - RBAC restricts system access based on a user's role. For example:
        - **Admins** can add/edit menu items, manage reservations, and view feedback.
        - **Customers** can only view the menu, make reservations, and submit feedback.
- **Data Encryption**:
    - All communications between the client and server are encrypted using **HTTPS**, preventing sensitive data from being intercepted.
    - **Password Hashing**: Passwords are stored securely using encryption algorithms like **bcrypt**, ensuring that even if the database is compromised, user passwords remain protected.

Preventive Measures

- **Cross-Site Scripting (XSS)**:
    - Input sanitization is applied to ensure that malicious scripts are not executed when a user submits data (e.g., customer feedback).

- **SQL Injection**:
    - Although MongoDB (a NoSQL database) is less vulnerable to SQL injection, preventive measures, such as parameterized queries and validating user input, are still used to avoid potential security risks.

4.4 Scalability and Optimization

The RMS is built to handle growth in both data and user base while maintaining high performance.

Scalability

- **Modular Architecture**:
    - The system is designed with scalability in mind. New features, such as integrating an online ordering system, can be added without major changes to the core system.
- **Load Balancing**:
    - Load balancers are used to distribute traffic evenly across multiple servers, ensuring that no single server is overwhelmed. This helps the system handle a large number of simultaneous users, especially during peak hours.
- **Auto-scaling**:
    - The system can automatically adjust server capacity based on demand. For instance, during a busy holiday season, the system can scale up resources to accommodate increased traffic.

Performance Optimization

- **Database Indexing**:
    - Indexes are created on frequently queried fields, such as the reservation date or menu item, to speed up query performance.
- **Caching**:

    - Frequently accessed data, such as the menu or popular reservations, is stored in a cache for quick retrieval, reducing database load and improving response times.

- **Performance Tools**:
    - Tools like **Lighthouse** are used to measure performance metrics such as page load times, accessibility, and SEO. The results help identify areas where the system can be optimized for better speed and responsiveness.

4.5 Testing and Deployment

Testing

- **Unit Testing**:
    - Each individual component (e.g., API endpoints, UI elements) is tested to ensure it functions correctly in isolation.
- **Integration Testing**:
    - This testing ensures that all parts of the system (frontend, backend, and database) work together seamlessly. For example, when a user makes a reservation, it should reflect correctly in the database.
- **Load Testing**:
    - The system is tested under high traffic conditions to ensure it can handle peak usage. This simulates multiple users interacting with the system simultaneously, ensuring that it doesn’t crash or slow down.

Deployment

- **Cloud Platforms (AWS, Heroku)**:
    - The RMS is deployed to cloud hosting platforms like **AWS** or **Heroku** to ensure it is scalable, cost-effective, and reliable. These platforms offer robust infrastructure for handling large-scale applications.
- **Git for Version Control**:

- Git is used for tracking code changes and managing different versions of the system. Developers use Git to collaborate, handle code merges, and roll back to previous versions if needed.

CHAPTER 5

DESIGN AND IMPLEMENTATION

This chapter outlines the design and implementation processes of the **Restaurant Management System (RMS)**, including architecture, tools, technologies, and step-by-step execution. Additionally, it highlights the usage of **port 3000** during development and testing.

5.1 System Architecture

The Restaurant Management System (RMS) is built using a three-tier architecture, which separates the system into three distinct layers: Presentation, Business Logic, and Data. This modular approach allows for better organization, scalability, maintainability, and flexibility. Let's dive into each of the three layers in detail:

1. Presentation Layer (Frontend)

The Presentation Layer is responsible for handling the user interface (UI) and user experience (UX) of the application. It's the part that users directly interact with when using the system, whether they are customers or administrators.

Frontend Technologies

- React.js:
    - The frontend is developed using React.js, a powerful JavaScript library for building user interfaces. React allows developers to create reusable UI components, which can be easily maintained and updated without the need for full page reloads.
    - React Components: In a React-based system, different UI elements (like forms, buttons, tables) are broken down into components, which can be reused throughout the application.
    - State Management: React uses a virtual DOM to efficiently update and render only the parts of the UI that change when data is modified. This ensures that interactions like making a reservation or adding a menu item are smooth and responsive.

Responsive and Interactive UI

- The frontend is built to be responsive, meaning it adapts to different screen sizes, from large desktop monitors to small mobile screens. This is achieved using a mobile-first design approach.
    - On mobile, the user interface adjusts to display content efficiently, such as switching to a vertical layout for menus and tables.
- Interactive Features:
    - Real-time features, such as table reservations and dynamic menu updates, are enabled through JavaScript. React's state management ensures that when a customer interacts with the application (e.g., making a reservation), the UI updates without needing to refresh the page.

Development and Local Access

- During development, the frontend application is typically run on port 3000 by default, which allows developers to access the application locally using http://localhost:3000. This is where they can test and debug the frontend components.
    - Once deployed, the frontend is usually hosted on cloud platforms (e.g., AWS, Heroku), which will serve the application over a public URL.

2. Business Logic Layer (Backend)

The Business Logic Layer is responsible for handling the server-side logic, such as processing requests, handling authentication, managing data flow, and interacting with the database. The backend operates as the intermediary between the frontend and the database, ensuring that the system runs as expected.

Backend Technologies

- Node.js:
    - Node.js is used for the backend because it allows JavaScript to be executed on the server. It’s known for its event-driven and non-blocking I/O model, making it suitable for handling many simultaneous requests, which is crucial in a system like RMS where multiple users may interact

with the application at the same time (e.g., making reservations, viewing menus).

  - Node.js is also scalable and can handle a large number of concurrent requests, making it ideal for handling traffic spikes during busy hours in the restaurant.

- Express.js:

  - Express.js is a web application framework built on top of Node.js that simplifies routing and request handling. It allows developers to define RESTful API endpoints for the frontend to interact with.

    - For example:

      - POST /reservation: To create a new reservation.

      - GET /menu: To retrieve the restaurant’s menu.

      - POST /feedback: To submit customer feedback.

Authentication and Authorization

- Authentication:

  - The backend is responsible for verifying users' identity. For example, customers or restaurant admins need to log in to access certain features. The backend uses technologies like JWT (JSON Web Tokens) to manage user sessions securely.

  - When a user logs in, the backend generates a token that stores their session details. This token is then sent to the frontend, where it’s stored (usually in cookies or local storage) and used to validate future requests.

- Role-based Access Control (RBAC):

  - Different users have different roles (e.g., customers, restaurant administrators). The backend ensures that each user’s access is limited to only what’s necessary for their role:

        - Customers: Can make reservations, view the menu, and submit feedback.
        - Admins: Have elevated privileges, such as the ability to add/edit menu items or view detailed reports.

Ports for Local Development

- During local development, the backend usually runs on port 5000 or port 8000. This allows the frontend to interact with the backend via the designated port, making API requests such as booking a table or submitting feedback.
    - The API endpoints are accessible locally through http://localhost:5000 (or http://localhost:8000 depending on configuration).
    - The backend and frontend typically run on different ports during development (e.g., 3000 for frontend, 5000/8000 for backend), but CORS (Cross-Origin Resource Sharing) configurations ensure that requests from the frontend can be accepted by the backend.

3. Data Layer (Database)

The Data Layer is responsible for storing and retrieving data from a database. In the case of RMS, a NoSQL database like MongoDB is used because it offers flexibility and scalability, which are crucial for the evolving needs of a restaurant management system.

Database Technologies

- MongoDB:
    - MongoDB is a NoSQL database that stores data in a document-oriented format. Instead of storing data in rows and tables (like traditional relational databases), MongoDB stores data in collections of JSON-like documents (BSON format).
    - This structure allows for schema flexibility, meaning that different records (e.g., reservations or menu items) can have different structures or fields, which is beneficial for evolving applications.

Types of Data Stored

- Menu Details:
    - The menu, including item names, descriptions, prices, and images, is stored in MongoDB. This allows the restaurant to update menu items dynamically through the system's backend.
- Reservations:
    - Customer reservations, including details like the customer’s name, contact information, date, and time, are stored in the database. This helps in tracking bookings and table availability.
- Feedback:
    - Customer feedback, including ratings and comments, is collected and stored in MongoDB. This data can be used to generate reports or identify areas of improvement in the restaurant’s service.

Scalability and Data Flexibility

- Scalability:
    - MongoDB is designed to scale horizontally, meaning the system can handle increased data volume (e.g., customer feedback, orders, reservations) by distributing data across multiple servers.
- Data Integrity:
    - MongoDB ensures that data remains consistent and reliable. For example, when a customer makes a reservation, the system ensures that the table's availability is accurately updated, preventing double-booking.

    - Built with React.js for responsive and interactive UIs, providing a seamless experience for users. It runs locally on port 3000 during development.

5.2 Design Details

5.2.1 Database Design

The database structure supports flexibility and scalability:

- **Collections**:
    - users: Stores user data and roles (e.g., customer or admin).
    - menu: Contains dish names, prices, and categories.
    - reservations: Tracks table bookings and customer details.
    - feedback: Stores customer reviews and ratings.

5.2.2 Frontend Design

- **Home Page**: Displays the restaurant overview and links to menu, reservations, and feedback.
- **Menu Page**: Showcases available dishes with search and filter options.
- **Reservation Form**: Enables users to select a date, time, and party size for booking.

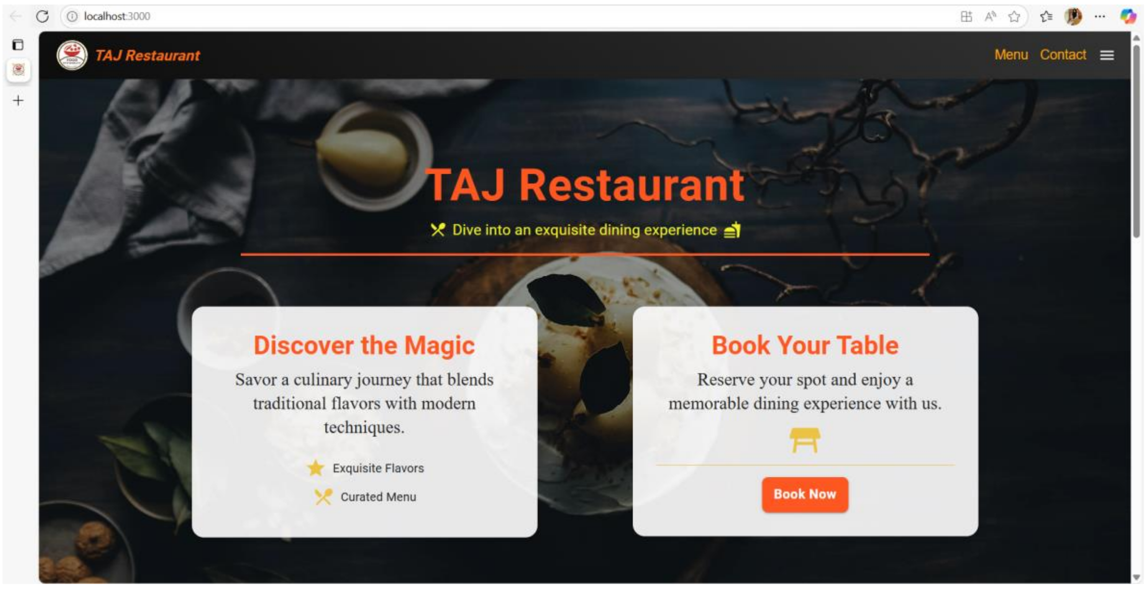

localhost:3000
TAJ Restaurant
Menu
Contact
TAJ Restaurant
Dive into an exquisite dining experience
Discover the Magic
Savor a culinary journey that blends traditional flavors with modern techniques.
Exquisite Flavors
Curated Menu
Book Your Table
Reserve your spot and enjoy a memorable dining experience with us.
Book Now

Fig.5.1. User Interface

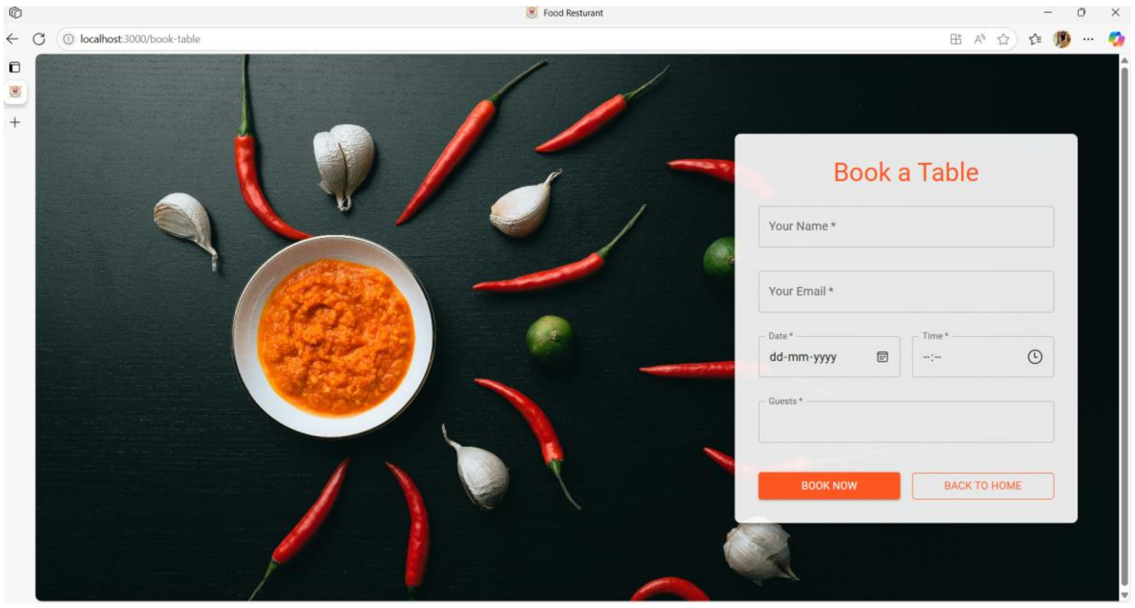
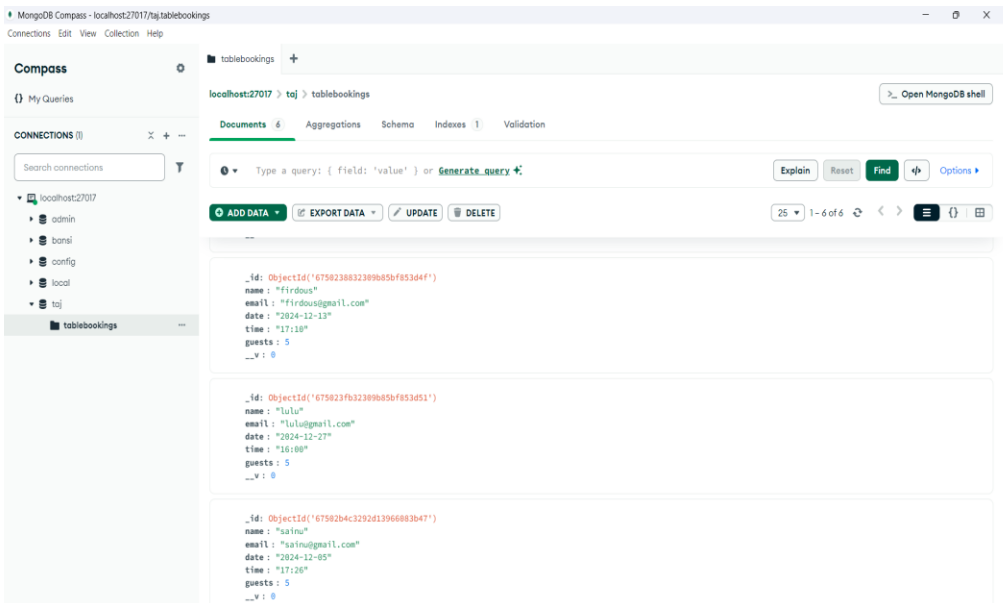


Fig.5.2. Reservation Form

5.2.3 Backend Design

- **API Endpoints**:
    - GET /menu: Fetches menu items.
    - POST /reservations: Saves table booking data.
    - GET /feedback: Retrieves customer feedback.
    - POST /auth/login: Handles user authentication.



```
_id: ObjectId('6750238832309b85bf853d4f')
name : "firdous"
email : "firdous@gmail.com"
date : "2024-12-13"
time : "17:10"
guests : 5
__v : 0

_id: ObjectId('675023fb32309b85bf853d51')
name : "lulu"
email : "lulu@gmail.com"
date : "2024-12-27"
time : "16:00"
guests : 5
__v : 0

_id: ObjectId('67502b4c3292d13966083b47')
name : "sainu"
email : "sainu@gmail.com"
date : "2024-12-05"
time : "17:26"
guests : 5
__v : 0
```

Fig.5.3 MongoDB

5.3 Implementation

5.3.1 Tools and Technologies

- **Frontend**: React.js (port 3000) and Bootstrap for styling.
- **Backend**: Node.js and Express.js (port 5000) with JWT for authentication.
- **Database**: MongoDB Atlas for cloud-based storage.
- **Version Control**: Git and GitHub for collaborative development.

5.3.2 Step-by-Step Execution

1. **Environment Setup**:
    - Installed required dependencies for React, Node.js, and MongoDB.
    - Configured the frontend to run on **port 3000** and the backend on **port 5000**.
2. **Frontend Development**:
    - Created React components for user interface elements.
    - Tested and debugged components using http://localhost:3000.
3. **Backend Development**:
    - Developed RESTful APIs for CRUD operations and authentication.

    - Connected to MongoDB and implemented middleware for security.

4. **Integration**:
    - Configured a proxy in the React package.json file to route frontend requests to the backend.
5. **Testing and Debugging**:
    - Verified seamless communication between frontend and backend through the configured ports.
6. **Deployment**:
    - The application was deployed to **Heroku**, with MongoDB Atlas managing the database.

5.4 Challenges and Solutions

1. **Port Conflicts**: Resolved by assigning separate ports to the frontend (3000) and backend (5000).
2. **API Testing**: Used Postman to debug API endpoints during integration.

CHAPTER 6

RESULT AND ANALYSIS

This chapter presents the results of the **Restaurant Management System (RMS)** and analyzes its performance, emphasizing the role of port configurations in development and testing.

Results

Functional Results

The RMS met all functional requirements:

- **Menu Management**: Administrators can manage dishes through a user-friendly interface.

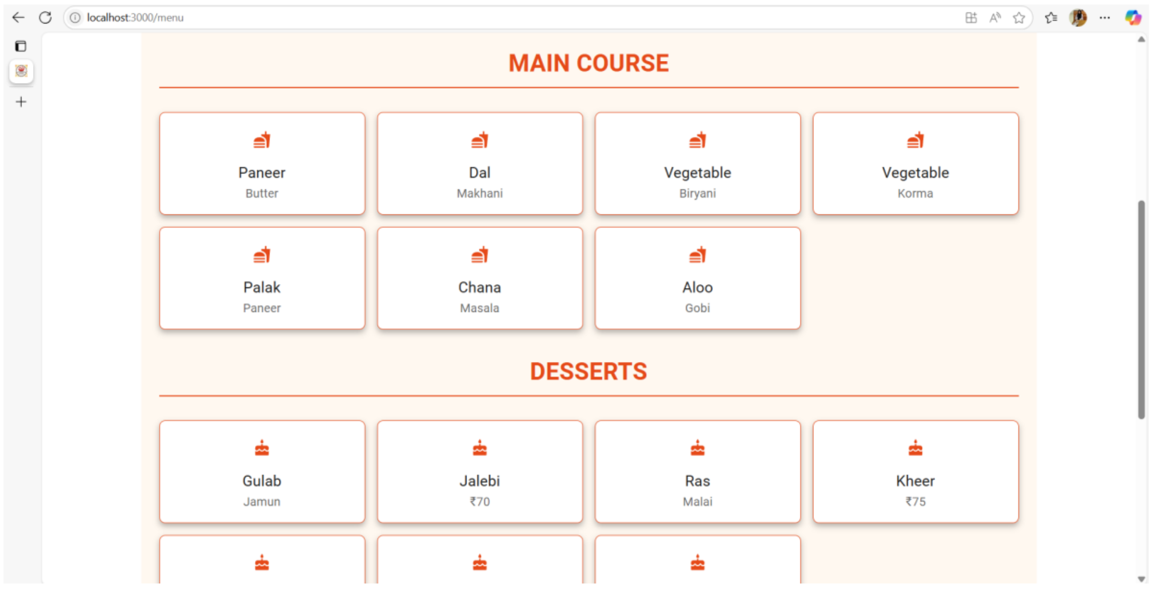


Fig.6.1. Menu page

- **Table Reservations**: Customers can reserve tables efficiently, with real-time availability updates.
- **Customer Feedback**: Users can submit reviews and ratings for restaurant services.
- **Port Usage**: During development, **port 3000** ensured smooth frontend testing, while the backend used **port 5000** for API communication.

Deployment

The RMS was deployed on **Heroku**, with frontend and backend seamlessly integrated. The production environment does not directly expose port 3000, as the application is served as a single unit.

Analysis

Performance Analysis

- API requests responded within an average of 300ms.
- The use of **port 3000** during development enabled efficient testing and debugging.

User Experience

- Responsive design ensured compatibility with various devices.

Scalability

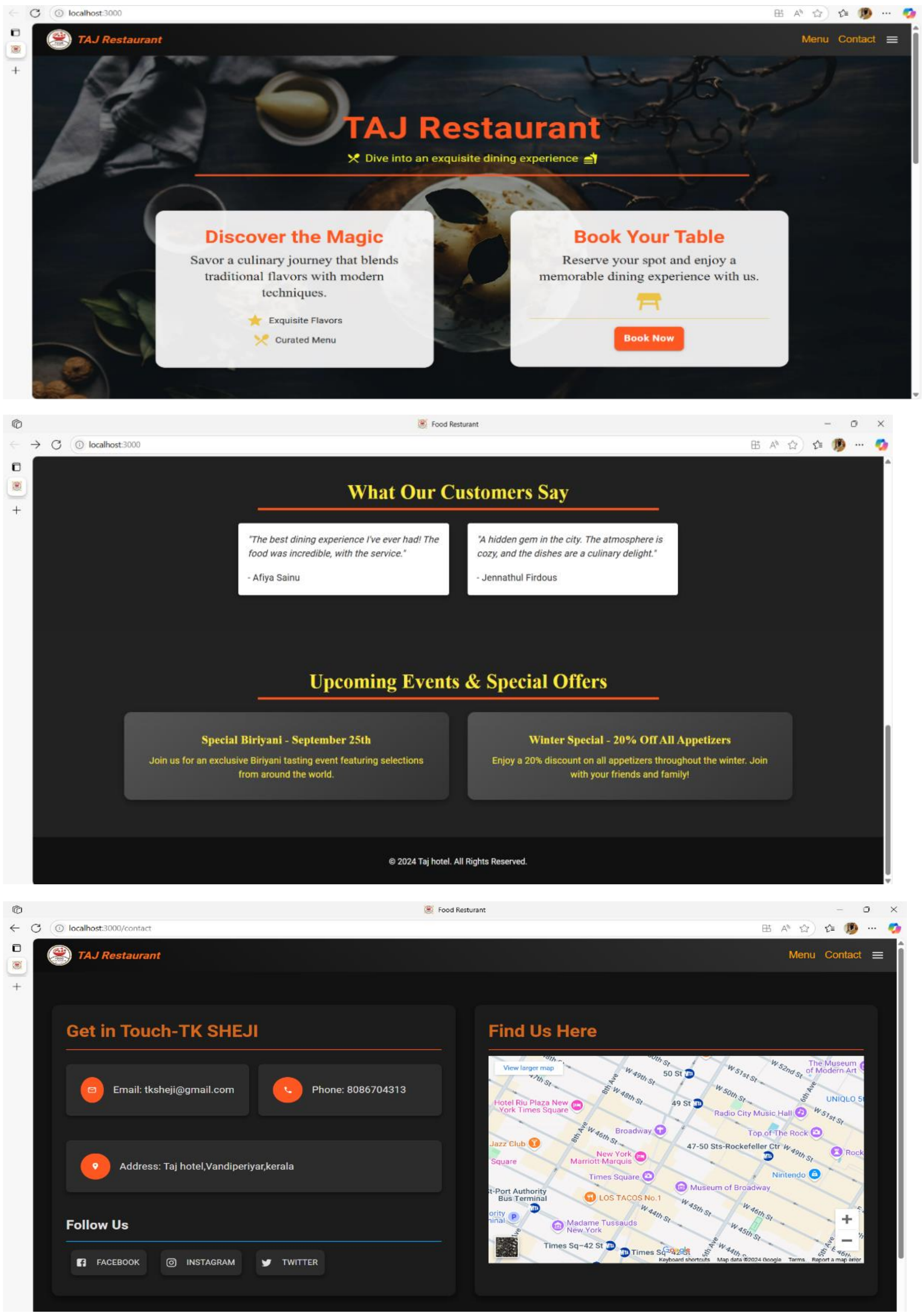
- The architecture supports additional features, with ports configurable for future needs.

Challenges Observed

- Minor port conflicts during initial setup were resolved through configuration adjustments.

Comparison with Objectives

Objective	**Result**	**Status**
Develop a dynamic menu management system	Fully functional	Achieved
Enable online table booking	Seamless reservation system	Achieved
Ensure secure and scalable architecture	Secure and scalable system	Achieved



TAJ Restaurant

Menu Contact

TAJ Restaurant

Dive into an exquisite dining experience

Discover the Magic

Savor a culinary journey that blends traditional flavors with modern techniques.

Exquisite Flavors

Curated Menu

Book Your Table

Reserve your spot and enjoy a memorable dining experience with us.

Book Now

What Our Customers Say

"The best dining experience I've ever had! The food was incredible, with the service."

- Afiya Sainu

"A hidden gem in the city. The atmosphere is cozy, and the dishes are a culinary delight."

- Jennathul Firdous

Upcoming Events & Special Offers

Special Biriyani - September 25th

Join us for an exclusive Biriyani tasting event featuring selections from around the world.

Winter Special - 20% Off All Appetizers

Enjoy a 20% discount on all appetizers throughout the winter. Join with your friends and family!

© 2024 Taj hotel. All Rights Reserved.

TAJ Restaurant

Menu Contact

Get in Touch-TK SHEJI

Email: tksheji@gmail.com

Phone: 8086704313

Address: Taj hotel,Vandiperiyar,kerala

Follow Us

FACEBOOK INSTAGRAM TWITTER

Find Us Here

Fig.6.2. Website pages

CHAPTER 7

CONCLUSION

The **Restaurant Management System (RMS)** project successfully demonstrates the integration of modern web development technologies and cloud platforms to address real-world challenges in the restaurant industry. By utilizing **React.js** for the frontend, **Node.js** and **Express.js** for the backend, and **MongoDB** for data management, the system offers a scalable and flexible solution that can grow with the business.

The system enables efficient management of key operations such as menu management, table reservations, and customer feedback. It provides a user-friendly interface for customers to easily navigate the restaurant's offerings, make reservations, and submit feedback, while administrators can manage these operations seamlessly through an intuitive backend interface.

The frontend was developed and tested using **port 3000**, ensuring a smooth and responsive user experience. The backend, running on separate ports like **port 5000**, facilitated clear communication and easy debugging between the frontend and backend. The use of **MongoDB Atlas** for database management allows the system to handle a large volume of dynamic data with high availability and low latency.

Security was prioritized with **JWT** authentication and role-based access management, ensuring that sensitive information is protected and only authorized users have access to critical functions. The deployment on **Heroku** provides automatic scaling, ensuring that the system can handle varying traffic loads without compromising performance.

In conclusion, the **RMS** project has successfully created a flexible, scalable, and secure system that streamlines restaurant operations and enhances the customer experience. This solution is adaptable for future enhancements, including integration with AI-driven recommendations or loyalty programs, making it a valuable tool for the restaurant industry.

CHAPTER 8

REFERENCES

REFERENCES:

https://developer.mozilla.org/en-US/

https://www.w3schools.com/

https://www.mongodb.com/docs/

https://expressjs.com/

APPENDIX

A1- Source Code

Front-end:

index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Food Resturant</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

<u>package.json</u>

```
{
  "name": "rest-frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.13.0",
    "@emotion/styled": "^11.13.0",
    "@mui/icons-material": "^6.0.1",
    "@mui/lab": "^6.0.0-beta.8",
    "@mui/material": "^6.0.1",
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.7.4",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-router-dom": "^6.26.1",
    "react-scripts": "5.0.1",
    "react-scroll-trigger": "^0.6.14",
    "react-toastify": "^10.0.5",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Back-end:

Package.js

```
{
  "name": "rest-backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "dotenv": "^16.4.5",
    "express": "^4.19.2",
    "mongoose": "^8.5.3"
  }
}
```

Server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const app = express();
const PORT = 5000;
const bookTableRoutes = require('./routes/bookTable');
require('dotenv').config();


mongoose.connect('mongodb://localhost:27017/taj', {
    useNewUrlParser: true,
    useUnifiedTopology: true,
}).then(() => {
    console.log('Connected to MongoDB');
}).catch((err) => {
    console.error('Error connecting to MongoDB', err);
});


app.use(cors());
app.use(express.json());
app.use('/api/book-table', bookTableRoutes);


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

A2 – Technical Bibliography

https://css-tricks.com/

https://docs.npmjs.com/

https://www.djangoproject.com/start/

https://www.python.org/doc/

https://www.freecodecamp.org/