Uttara University
Department of Computer Science and Engineering
Holding 77, Beribadh Road, Turag, Uttara, Dhaka 1230, Bangladesh

Database Management System Lab
(CSE0612302)

Academic Session: 2025-2026
Project Proposal
Restaurant Management System

Submitted by:

Group Member	ID
Fariha Hasnat	2233081250
Md Maharab Shuvo	2233081278
Clement Raka De Costa	2233081243
Debottom Chakma	2233081529
Afroza Hossin Sorno	2233081249
Md. Mustafijur Rahman	2233081453

Submitted to:
Md. Ashraful Kabir
Lecturer , Department of CSE
Uttara University

Date of Submission: November 10, 2025

Contents

1 Project Overview

1.1 Project Title

Restaurant Management System

1.2 Project Summary

The Restaurant Management System (RMS) operates as a database-based software solution which unifies restaurant operations through one unified platform. The system enables restaurants to handle their core operations including order management and table booking and billing and inventory control and customer loyalty programs across different locations.

The system bases its operations on three fundamental elements which include data integrity and optimized database design and efficient query processing for reliable system performance. The RMS database system will process extensive amounts of customer information and transaction records while using normalization and indexing and constraints and referential integrity to protect data accuracy and security.

The system implements sophisticated SQL capabilities through triggers and stored procedures and relational models to deliver immediate data access and analytical reports and secure access controls based on user roles. The system allows restaurant managers to use current data for making better business choices.

The backend of the system runs on Python for developing secure business operations and strong server-side functionality. The system uses HTML and CSS and JavaScript for building its user-friendly interface which staff members and administrators can access easily.

The RMS system proves how database management and full-stack development works in business settings through its implementation of data-based choices and contemporary software development methodologies.

1.3 Project Scope

The Restaurant Management System (RMS) functions as a complete web-based platform which unites all fundamental restaurant operations into one digital platform. The system will unite vital operational components through its customer and membership management system and order and billing processing system and table reservation system and inventory and supplier coordination system and multi-branch administration system. The system aims to improve operational efficiency and data consistency and branch-to-branch communication so restaurant managers can base their decisions on data analysis.

The project will create a relational database structure with complete ER diagrams and relational schemas and data normalization to the Third Normal Form (3NF). The system will use SQL-based CRUD operations and stored procedures and triggers to execute automated business processes for order management and billing and inventory updates. The system generates analytical reports through SQL JOIN and GROUP BY operations to help managers track performance and gain valuable insights.

The RMS system will achieve scalability and technological flexibility through its design allows for easy integration with external platforms. The system will integrate with bKash and Rocket and Nagad payment gateways and Foodpanda and Pathao Food and FOODI delivery platforms to automate order processing and delivery coordination. The system will implement SMS and Email notification services to deliver immediate updates about orders and reservations and promotional offers to customers.

1.4 Background and Motivation

The Restaurant Management System (RMS) development stems from Bangladesh restaurants needing digital solutions to optimize their operational processes. Small and medium-sized restaurants in Bangladesh continue using traditional manual systems for order processing and billing and inventory tracking and table management which creates errors and delays and reduces operational efficiency. The practice of using paper-based billing systems together with handwritten order slips creates multiple errors which result in lost orders and miscommunication between kitchen staff and service personnel thus decreasing service quality and customer satisfaction.

The management of inventory presents a significant operational problem for businesses. Restaurants that lack real-time stock monitoring systems end up with either excessive inventory or insufficient stock which causes product waste and lost business opportunities. The process of manual reservation management produces double bookings and underused tables which negatively affects business revenue. Restaurants operating multiple locations need to handle sales data and inventory records and employee performance metrics through centralized systems which creates delays in decision-making and reduces accuracy.

The RMS system solves operational problems by uniting order processing with billing functions and inventory management and reservation systems into a single web-based platform. The system delivers quick service delivery which leads to enhanced customer satisfaction because market competition demands it.

The RMS development work enables team members to learn software engineering fundamentals through database creation and front-end and back-end programming and system deployment. The project helps Bangladesh's restaurant industry transition to digital operations which enables businesses to scale up while improving operational efficiency and service delivery quality.

2 Objectives and Goals

- **Objective 1:** Develop a secure, web-based platform for managing restaurant operations, ensuring role-based access for staff, managers, and administrators to protect sensitive business information.
- **Objective 2:** Automate core processes such as order taking, billing, inventory management, and table reservations to reduce manual workload, minimize errors, and improve efficiency, especially during peak hours.
- **Objective 3:** Provide real-time reporting and analytics on sales, inventory, and customer trends, enabling data-driven decisions for stock management, menu planning, and staffing.
- **Objective 4:** Ensure a user-friendly interface that can be easily used by staff with limited technical experience, accessible across desktops, tablets, and smartphones.
- **Objective 5:** Support scalable deployment for multi-branch operations, allowing small and medium restaurants to expand operations digitally without major infrastructure changes.

3 Technical Requirements

3.1 Functional Requirements

1.User authentication and role-based access for staff, managers, and administrators with secure login.

2.Order management and billing system with automated invoice generation and support for multiple payment methods commonly used in Bangladesh (cash, bKash, Rocket, Nagad).

3.Table reservation system to prevent double bookings and track upcoming reservations.

4.Inventory tracking with alerts for low stock, supplier information, and integration for manual updates.

5.Reporting module to generate daily, weekly, and monthly reports for sales, inventory, and customer visits.

6.Notification system (SMS or email) for order confirmation, reservation reminders, and promotions.

3.2 Non-Functional Requirements

System response time: Under 3 seconds for 90% of requests, considering internet speed variability in Bangladesh.

- **Uptime:** 99% availability during business hours.
- **Data security:** AES-256 encryption for sensitive information and secure backups.

- **Cross-device support:** Compatible with desktops, tablets, and smartphones; works on Chrome and Firefox.
- **Scalability:** Support for 50–200 concurrent users per branch, depending on restaurant size.
- **Offline mode:** Basic offline functionality to allow order taking during temporary internet outages.

3.3 Technologies and Tools

- **Backend:** Python for robust server-side business logic and secure integration with payment APIs.
- **Frontend:** HTML, CSS, and JavaScript for a responsive, user-friendly interface.
- **Database:** MySQL for efficient storage and management of structured data.
- **Version Control:** Git for collaborative development and version tracking.
- **Containerization:** Docker to ensure consistent deployment across multiple environments and simplify scaling.

3.4 System Architecture

The Restaurant Management System (RMS) adopts a three-tier architecture comprising the presentation layer, application layer, and data layer to ensure scalability, maintainability, and secure operations.

The presentation layer, developed using HTML, CSS, and JavaScript, delivers a responsive and user-friendly interface accessible from desktops, tablets, and mobile devices. It allows restaurant staff and administrators to manage orders, reservations, and inventory through an intuitive dashboard.

The application layer, powered by Python, handles all core business logic, including order processing, billing, and data synchronization across branches. It also manages integrations with payment gateways such as bKash, Rocket, and Nagad, as well as future connectivity with POS and delivery platforms.

The data layer, built on MySQL, securely stores structured information such as menu items, customer records, orders, inventory details, and user credentials. Database normalization and indexing are implemented to ensure efficient query performance and data integrity.

Additionally, Docker containerization is used to maintain consistent deployment environments, simplifying updates and scalability across multiple branches. This architecture enables smooth operation, enhances security, and provides a strong foundation for future expansion, including mobile app integration and analytics dashboards.

4 Methodology

4.1 Development Methodology

The Restaurant Management System (RMS) will implement a **Hybrid Agile–Incremental Development Model** to achieve flexible progress and structured development throughout the software development process. The project uses a combined development method which combines incremental modeling with Agile principles to support teams working on parallel tasks and changing project requirements.

The project consists of three parallel working subgroups which handle Database Management (SQL) and Frontend Development and Backend Development. The system development process will follow Agile principles through daily stand-ups and bi-weekly sprint reviews and retrospectives to maintain synchronization between the three subgroups. The SQL team functions as the central data coordination point will maintain application layer consistency through database interaction validation during each sprint.

The system development will follow an incremental approach which delivers operational system parts at each development stage. The system will start with basic order management and billing functions before implementing inventory tracking and reservation systems and report production capabilities. The development team will use Agile sprints to execute planning and development and testing and improvement of system features while receiving user feedback and performance assessment results.

The system will deliver operational modules at an early stage through this hybrid method which enables continuous development with reduced potential risks. The software reliability improves through continuous integration and iterative testing while the modular design makes it easier to debug and perform future updates. The model enables businesses to scale their operations because it provides a straightforward method to add new features in future versions including POS integration and digital payment systems and third-party food delivery platform connections..

4.2 Project Phases

Phase 1: Planning and Requirements Analysis

This phase focuses on identifying restaurant-specific operational challenges and defining the system's functional and non-functional requirements. The team will conduct discussions with restaurant managers, staff, and potential users to understand needs related to order management, billing, inventory, and reservations. The scope, user roles, and system objectives will be clearly outlined. Preliminary wireframes and a project roadmap will be developed.
Tangible outcomes: Requirements Specification Document, Project Plan, Initial Wireframes.

Phase 2: System Design and Prototyping

In this phase, the system architecture will be finalized following a three-tier model — presentation, application, and data layers. UI/UX prototypes for the frontend will be created to ensure user-friendly navigation, while the backend team designs business logic workflows and the SQL team builds the database schema. This stage will also define integration points between modules and APIs for future scalability (e.g., POS and payment gateways).
Tangible outcomes: System Architecture Diagram, Database Schema, UI/UX Prototypes.

Phase 3: Incremental Implementation and Testing

The development process will proceed in multiple Agile sprints, each delivering a functional increment of the RMS. Core modules like billing and order processing will be developed first, followed by inventory, reservations, and reporting. Continuous integration and testing (unit, integration, and user acceptance) will be carried out after each increment to ensure quality and stability.
Tangible outcomes:Working RMS Modules, Test Reports, Integration Logs.

Phase 4: Deployment, Training, and Maintenance

The final system will be deployed in a controlled environment for pilot testing at selected restaurant branches. User training sessions will be conducted for staff to ensure smooth adoption. Post-deployment, maintenance will focus on bug fixes, performance optimization, and preparation for future feature integration (e.g., digital payment, delivery platform connectivity).
Tangible outcomes:Deployed System, User Manuals, Training Materials, Maintenance Plan.

4.3 Risk Management

Effective risk management is essential to ensure the timely and successful delivery of the Restaurant Management System (RMS). Since the project involves multiple parallel development teams—Frontend, Backend, and SQL—working on interdependent modules, several potential risks may arise during the project lifecycle. These include technical, organizational, and operational challenges that could affect performance, schedule, or quality.

Key Risks and Safeguard Strategies:
• Integration Challenges: The system will experience compatibility problems because of poor coordination between backend logic and frontend interface and SQL database.

Safeguard : The team should perform integration tests after each development stage and keep API documentation available for all team members to follow.

• Scope Creep: The project scope will expand past its original boundaries because users request new features which include POS system integration and delivery service automation.

Safeguard : The team needs to establish a formal change management system which requires all new feature additions to get approval from stakeholders.

• Team Coordination Delays: The project timeline will experience delays because team members lack clear understanding of their responsibilities and tasks.

Safeguard : The team should hold weekly sprint meetings and daily stand-up sessions to maintain task understanding and team member responsibility.

• Technical Limitations: The system testing and deployment process will face obstacles because of slow internet speeds and hardware problems and local environment deployment issues.

Safeguard : The system will operate correctly in low-connectivity situations because the development team used lightweight code and developed offline simulation capabilities.

• Data Security Risks: The system faces security threats because unauthorized users can access sensitive customer information and financial data.

Safeguard : The system uses AES-256 encryption and secure authentication methods and performs regular database backups to protect sensitive information.

• Timeline and Resource Constraints: The project schedule will experience delays because of unexpected technical problems and staff member absences.

Safeguard : The project schedule includes extra time for unexpected events and the team receives training to perform each other's duties when needed.

• User Adoption and Training Risks: End users may face difficulties adapting to the new digital system.
Safeguard : Provide training sessions, detailed user manuals, and responsive technical support during deployment.

A risk register will be maintained throughout the project lifecycle to record, monitor, and address risks systematically. Regular reviews will ensure proactive mitigation and project continuity.

5.Team Roles and Responsibilities

Member Name	Role and Responsibilities
Fariha Hasnat	**CTO & Database Administrator :** Leads overall project planning and coordination, manages timelines, ensures collaboration among all teams, and maintains quality standards. Designs and manages the SQL database, optimizes queries, and validates data integrity. Reviews frontend and backend modules for consistency and has final authority to approve or modify components before deployment.
Clement Raka De Costa	**Backend Lead Developer:** Oversees the development of server-side logic using Python, ensuring secure, efficient, and scalable API design. Manages integration between the database and frontend, reviews backend code, and enforces coding standards and best practices. Coordinates with the SQL team to validate data handling and business logic.
Debottom Chakma	**Backend Developer:** Implements backend modules and APIs based on specifications provided by the backend lead. Handles debugging, testing, and optimization of server-side functions. Works closely with the database and frontend teams to ensure smooth system functionality and data consistency.
Afroza Hossin Sorno	**Frontend Lead Developer :** Oversees the design and implementation of responsive user interfaces using HTML, CSS, and JavaScript, ensuring usability, cross-browser compatibility, and adherence to design standards.

	Coordinates with the CTO and ensures their team's code meets standards and deadlines.
Md. Mustafijur Rahman	**Frontend Developer:**Implements assigned UI components and interactive features under the supervision of the frontend lead. Assists in debugging, styling, and improving user experience. Collaborates with backend developers to integrate APIs and ensure smooth data rendering on the client side.
Md Maharab Shuvo	The **QA Engineer** is responsible for ensuring the RMS meets quality and performance standards through rigorous unit, integration, and user acceptance testing. They identify, document, and track bugs, verify fixes, and conduct regression testing after updates. By maintaining detailed test reports and ensuring system reliability, the QA Engineer guarantees a stable and user-friendly platform.

6.Timeline and Milestones

Milestone	Description	Deadline
Requirements Analysis	Conduct stakeholder meetings, finalize requirements, and define project scope for frontend, backend, and database modules.	October 15, 2025
System Design	Develop UI/UX wireframes, database schema, and system architecture based on incremental modules.	October 25, 2025
Incremental Implementation	Frontend team develops responsive interfaces, backend team implements core logic and integrations, and SQL team manages data flow and queries.	November 10, 2025
Testing & Integration	Perform unit, integration, and user acceptance testing; ensure smooth communication between frontend, backend, and database.	November 20, 2025
Deployment & Training	Deploy the complete system, provide staff training, and prepare documentation for maintenance and future updates.	November 25, 2025

7.Expected Challenges and Solutions

Developing the Restaurant Management System (RMS) may present several challenges, including system integration, data consistency, and performance optimization across modules. Solutions include:
• **Coordinated integration testing** between frontend, backend, and SQL teams to ensure smooth data flow and functional compatibility.
• **Implementing AES-256 encryption and secure authentication** to protect sensitive customer and transaction data.
• **Optimizing database queries and indexing** to improve performance during peak business hours.
• **Ensuring responsive, cross-device design** for smooth operation on desktops, tablets, and POS terminals.
• **Conducting iterative user testing** with restaurant staff to enhance usability and reduce training time.

These strategies will ensure that the RMS remains secure, efficient, and adaptable to real-world restaurant operations.

8.Stakeholder Analysis

The key stakeholders of the Restaurant Management System (RMS) for **'Dhakaiya Bites'** include the management team, restaurant staff, and customers, each playing a vital role in the project's success. The management team requires real-time access to analytics on sales, inventory levels, reservations, and branch performance to make data-driven decisions and streamline operations. The staff depend on an intuitive interface for managing orders, billing, table reservations, and customer interactions efficiently, reducing manual workload and service delays. The customers, representing the end users, expect faster service, accurate order tracking, and seamless digital experiences for payments and delivery coordination.

Under the Hybrid Agile–Incremental Development Model, these stakeholders will remain actively engaged throughout the project lifecycle. Weekly feedback meetings, sprint reviews, and prototype demonstrations will be conducted to gather insights and validate progress. This collaborative approach ensures the system continuously evolves based on real operational challenges and customer expectations—ultimately enhancing service quality, productivity, and customer satisfaction across all restaurant operations.

9.Evaluation Plan

• **Usability:** Conduct staff and management surveys to achieve at least 90% satisfaction regarding system navigation, interface clarity, and ease of use.
• **Scalability:** Assess the system's ability to handle up to 200 concurrent users per branch without performance degradation.
• **Performance:** Maintain response times under 3 seconds for 95% of operations, ensuring smooth order processing even during peak hours.
• **Reliability:** Achieve 99% uptime during business hours through optimized server configuration and routine maintenance.
• **Data Accuracy:** Validate billing, order, and inventory records to ensure 100% consistency between frontend and database modules.
• **Integration Efficiency:** Test seamless communication among frontend, backend, and SQL components after each sprint to confirm proper synchronization.
• **Security:** Evaluate the implementation of AES-256 encryption, authentication protocols, and secure payment integration (bKash, Rocket, Nagad).
• **User Feedback and Continuous Improvement:** Weekly feedback sessions and post-deployment analysis of system logs, error reports, and usage statistics will guide incremental updates and future feature enhancements.

This structured evaluation approach ensures that the RMS is secure, efficient, user-friendly, and scalable, aligning with both restaurant operations and long-term digital transformation goals.

10.References

1.Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.

2.Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited.

3.Dennis, A., Wixom, B. H., & Roth, R. M. (2018). *Systems Analysis and Design* (7th ed.). John Wiley & Sons.
4.Shelly, G. B., & Rosenblatt, H. J. (2012). *Systems Analysis and Design* (9th ed.). Cengage Learning.

5.Fowler, M. (2019). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (4th ed.). Addison-Wesley.

6.Agile Alliance. (2023). *What is Agile Software Development?* Retrieved from https://www.agilealliance.org

7.IBM Cloud Education. (2022). *What is the Incremental Model?* IBM Developer. Retrieved from https://developer.ibm.com

8.Docker Inc. (2023). *Containerization Explained: Benefits and Use Cases.* Retrieved from https://www.docker.com/resources/what-containerization/

9.Microsoft. (2024). *Best Practices for Web Application Performance Optimization.* Retrieved from https://learn.microsoft.com

10.MySQL Documentation. (2024). *MySQL 8.0 Reference Manual.* Oracle Corporation. Retrieved from https://dev.mysql.com/doc/

11.Django Software Foundation. (2024). *Django Documentation – Secure and Scalable Web Framework.* Retrieved from https://www.djangoproject.com/

12.Bangladesh Bank. (2024). *Overview of Digital Payment Systems in Bangladesh.* Retrieved from https://www.bb.org.bd

13.Foodpanda Bangladesh. (2024). *Digital Integration and Delivery Management for Restaurants.* Retrieved from https://www.foodpanda.com.bd

14.Pathao Ltd. (2024). *Pathao Food Partner Integration Overview.* Retrieved from https://pathao.com

15.bKash Ltd. (2024). *Secure Digital Payment Gateway Documentation.* Retrieved from https://www.bkash.com

11.Appendix

- Wireframe Mockups: Preliminary UI designs for the customer, staff, and admin dashboards, including pages for order management, billing, reservations, and inventory tracking (to be included in the final submission).
- Database Schema Diagram: Detailed relational schema for core entities such as customers, orders, menu items, inventory, reservations, and staff, illustrating primary and foreign key relationships (to be included in the final submission).
- Use Case Diagrams: Visual representations of key interactions among users and system modules, including order processing, billing, inventory updates, and reservation management.
- System Architecture Diagram: Overview of the three-tier architecture showing interaction among the frontend (HTML, CSS, JavaScript), backend (Python), and database (SQL) layers.
- Data Flow Diagram (DFD): Illustrates the movement of data between components such as user interfaces, application logic, and the central database.

12.Supplementary Information

12.1 System Features in Detail

The Restaurant Management System (RMS) integrates multiple functional modules to streamline restaurant operations, improve efficiency, and ensure accurate record-keeping:

- **Homepage:** Serves as the main dashboard, offering navigation to all system modules and providing summary analytics.
- **Admin Page:** Enables administrators to manage staff details, assign roles, and monitor performance. Includes options for managing employee salaries, tracking payments, and generating financial summaries.
- **Customer Page:** Maintains customer records, membership details, and purchase history, supporting loyalty programs and personalized service.
- **Staff Page:** Provides authorized staff access to edit the menu, update prices, and view current menu listings.
- **Billing Module:** Automatically generates bills based on orders, applies discounts and taxes, and records payment details securely.
- **Orders Module:** Allows staff to take, update, and track customer orders in real time, synchronizing with the kitchen and billing sections for smooth operations.
- **Login/Logout Module:** Implements secure role-based authentication for administrators, staff, and managers, ensuring controlled access to different parts of the system.

Each feature will be developed iteratively under the Agile–Incremental model, incorporating feedback after each sprint to ensure performance, accuracy, and usability.

12.2 User Interface Design Principles

The design focuses on clarity, speed, and user comfort, ensuring that both staff and administrators can operate the system efficiently even during peak business hours.

- **Intuitive Navigation:** A clean dashboard layout with well-organized menus will allow users to quickly switch between modules such as Orders, Billing, Staff Management, and Inventory. Icons, quick-access buttons, and shortcuts will reduce the number of clicks required for common tasks.
- **Real-Time Interaction:** Dynamic updates will allow users to see order status, billing changes, or inventory updates instantly without page reloads, creating a smooth, app-like experience
- **Visual Feedback:** Clear color cues and visual highlights will indicate system states — such as pending orders, completed payments, or low-stock alerts — helping staff make quick decisions under pressure.
- **Accessibility & Readability:** Large, legible fonts, balanced color contrast, and touch-friendly controls will ensure that the interface remains easy to use on desktops, tablets, and POS terminals.
- **Responsiveness & Adaptability:** The interface will automatically adjust to different screen sizes, providing consistent functionality across devices, from management desktops to handheld staff devices.

- **User Engagement:** The system will include subtle animations and confirmation prompts to make interactions feel smoother and more professional while reducing accidental errors.

13.Implementation Details

13.1 Database Design

The MySQL RMS will be designed using a relational and normalized schema to ensure high performance, scalability, and data consistency across all system modules. The database will store and manage information for customers, staff, orders, billing, and menu operations.

Key tables include:

Customers: Stores customer ID, name, contact information, and membership details for tracking orders and reservations.

Orders: Records order ID, customer reference, ordered items, quantities, prices, total amount, and order status.

Billing: Manages bill numbers, payment methods, amounts, discounts, and timestamps, linking directly to the corresponding order.

Payments: Tracks all financial transactions, including customer payments and staff salary disbursements.

Staff: Contains staff ID, name, assigned role, salary, and login credentials for access management.

Salaries: Records salary details, including employee ID, base pay, bonuses, and payment history.

Menu: Stores menu item ID, name, category, description, price, and availability status. Used in modules like Edit Menu, Edit Prices, and View Menu.

Inventory: Tracks ingredient names, current stock, supplier details, and reorder levels for seamless kitchen operations.

Reservations: Maintains reservation ID, customer reference, table number, date, time, and confirmation status.

The Admin Page will pull data from multiple tables (Staff, Roles, Salaries, Payments) to enable management oversight and system control.
All relationships will be established using foreign keys to ensure referential integrity. Stored procedures and triggers will automate repetitive tasks such as updating stock levels after order completion or salary adjustments after payment confirmation.

13.2 API Design

The Restaurant Management System will use RESTful APIs to facilitate communication between the frontend (HTML, CSS, JavaScript) and backend (Python) layers, ensuring smooth data exchange with the SQL database. Each module will have defined API endpoints with secure, role-based access controls.

Key API endpoints include:

/api/menu – Allows admins and authorized staff to add, edit, delete, and view menu items and prices.

/api/customers – Manages customer profiles, order history, and membership details.

/api/staff – Retrieves and updates staff information, including roles, salaries, and attendance logs.

/api/orders – Handles order placement, updates, cancellations, and retrieval for kitchen and billing synchronization.

/api/billing – Generates bills, calculates totals, applies discounts/taxes, and updates payment records.

/api/payments – Processes payments, records transactions, and updates billing and salary modules.

/api/salaries – Handles staff salary records, including base pay, bonuses, deductions, and payment history. I

/api/inventory – Tracks ingredient quantities, updates stock after orders, and sends low-stock alerts.

/api/reservations – Manages table booking requests, updates status, and prevents scheduling conflicts.

All APIs will follow the OpenAPI (Swagger) documentation standard for clarity and maintainability. Security will be enforced using JWT (JSON Web Tokens) for authentication, and HTTPS for encrypted communication between client and server.

14.Additional Notes

The Restaurant Management System (RMS) is designed with scalability and long-term adaptability in mind, ensuring that future technological and operational advancements can be easily incorporated. While the initial version focuses on order management, billing, reservations, and inventory tracking, the following updates are planned for future releases to enhance functionality and business value:

Planned Future Enhancements

AI-Based Demand Forecasting: Implement predictive analytics to estimate customer demand and optimize ingredient purchases.

Smart Inventory Management: Introduce automated reordering alerts and supplier integration for seamless stock replenishment.

Multi-Branch Synchronization: Enable centralized control for franchise or branch-based restaurants, allowing real-time data sharing.

Customer Loyalty & Membership Programs: Integrate digital reward points, referral systems, and personalized offers.

Mobile App Integration: Develop Android/iOS apps for customers to order online, track deliveries, and make reservations.

POS and Payment Expansion: Add integration with digital payment platforms like bKash, Nagad, Rocket, and card gateways.

Delivery Platform APIs: Connect with Foodpanda, Pathao Food, HungryNaki, and FOODI for automated order and delivery coordination.

Cloud Deployment: Migrate to a cloud infrastructure (AWS, Google Cloud) for scalability and remote management.

Multi-Language Interface: Add Bangla and English language options for better accessibility.

Chatbot Support: Introduce AI-driven chat assistants for customer service and order queries.

Advanced Reporting Dashboard: Integrate visual analytics tools for sales insights, staff performance, and expense tracking.

IoT Integration: Connect smart kitchen devices (e.g., temperature sensors, inventory trackers) for operational efficiency.

Maintenance and Continuous Improvement

Regular maintenance cycles will be followed, including database optimization, performance tuning, and security audits. Weekly feedback sessions with restaurant staff will guide usability improvements and feature prioritization.