**Chapter 3**

**3.1 Overview of the proposed system**

As we have discuss so far in the previous chapters, we have a good understanding of how the current system works.

On the basis of that, we have proposed how our system is intended to address the problems identified in the manual system at this stage. During our observations and interactions with some stakeholders of the existing system, the project team discovered some issues with their manual-based system. As a result, the project team proposed developing an automated hotel booking management system to solve the problem of the existing system.

The need for an automated solution arises from the limitations of the manual system in order to address various challenges and enhance operational efficiency. The manual system often involves errors and manual efforts in handling reservations, which can lead to inaccuracies and delays. By implementing an automated system, these issues can be significantly reduced, resulting in improved customer satisfaction and streamlined processes.

Additionally, the centralized nature of the proposed system allows for: efficient management of reservations, room availability, and special requests. This centralized approach enables hotel managers to have access to real-time information, empowering them to make informed decisions and optimize resource allocation. The system's robust reporting capabilities provide valuable insights into occupancy rates, guest preferences, and other key performance indicators, facilitating strategic planning and performance evaluation.

In addition to the mentioned features, the proposed system also offers users the ability to make payments for other services provided by the hotel. Users can conveniently settle their bills for additional services within the system, ensuring a seamless and integrated payment process.

Furthermore, the proposed system allows users to order food directly through the system. Guests can view the hotel's food menu, place orders, and make payments, all within a user-friendly interface. This feature enhances convenience and efficiency, eliminating the need for guests to physically visit or call the restaurant to place their orders.

Moreover, the system enables real-time communication between users and hotel staff, including the ability to connect with designated servants. This direct communication channel facilitates prompt assistance and personalized service, enhancing guest satisfaction and ensuring efficient handling of guest requests.

Overall, the automated solution addresses the limitations of the manual system by providing improved reservation management, streamlined processes, integrated payment options, online food ordering, and efficient communication channels. By incorporating these features, the proposed system aims to optimize the guest experience, enhance operational performance, and ultimately elevate customer satisfaction in the realm of hotel booking management.

**3.2 Functional requirements**

**User Registration**

The system shall provide a user registration feature, allowing guests to create accounts by providing necessary information such as name, contact details, and preferences. users should be able to securely store and manage their login credentials to access the system.

**Room Availability and Booking**

system shall enable users to check the availability of rooms for their desired dates, displaying relevant information such as room types, descriptions, amenities, and prices.users should be able to make reservations for available rooms, specifying their desired check-in and check-out dates, room preferences, and any additional requirements.

**Reservation Modification and Cancellation**

The system shall allow users to modify or cancel their reservations, adhering to the hotel's policies and guidelines. users should be provided with a user-friendly interface to easily update reservation details, such as changing check-in or check-out dates, room type, or other associated information.

**User Profile Management:**

The system shall provide functionality for users to manage their profile information, including personal details, contact information, and preferences. users should be able to update their profiles as needed, ensuring accurate and up-to-date information.

**Payment Processing**

The system shall support secure payment processing, providing users secure way to settle their reservation and service bills. users' payment information should be handled securely, following industry-standard encryption and compliance with relevant security regulations.

**Order Food and Services**

The system shall allow users to view the hotel's food menu, place orders, and make payments for their food orders within the system. users may be provided with the ability to request additional services offered by the hotel, such as spa treatments or transportation, and make necessary payments.

**Communication and Support**

The system shall facilitate communication between users and hotel staff, including designated servants, to address specific requests or provide assistance during the guests' stay.

**Check-In and Check-Out**

The system shall streamline the check-in process, allowing users to request room keys upon arrival and ensuring a smooth and expedited check-in experience. users should be able to initiate the check-out process through the system, notifying hotel staff and facilitating a seamless departure.

**Reporting and Analytics:**

The system shall generate reports and provide analytics capabilities to hotel management, offering insights into occupancy rates, revenue, guest preferences, and other key performance indicators.

**Integration with External Systems**

The system shall support seamless integration with external systems, such as payment gateways, property management systems, or channel managers, to ensure accurate data flow and synchronization.

The integration should enable efficient data exchange and enhance the overall functionality and performance of the system.

**3.3. Non-functional Requirements**

**3.3.1. Performance**

To ensure optimal performance of the Hotel Booking Management System, several non-functional requirements should be considered. The utilization of Node.js and React plays a crucial role in achieving a high-performance system.

Node.js Backend: The Node.js backend should be leveraged to handle concurrent requests efficiently. For example, by utilizing its event-driven, non-blocking I/O model, Node.js allows for scalable and performant handling of multiple requests simultaneously, resulting in improved response times and enhanced system performance.

React Frontend: React, as a JavaScript library for building user interfaces, can significantly contribute to the system's performance. For instance, React's virtual DOM and efficient rendering algorithm minimize unnecessary updates and optimize rendering, leading to better user experience, reduced rendering times, and improved overall system performance.

**3.3.2 Reliability**

Error Handling: The system should implement robust error handling mechanisms on the Node.js backend and the React frontend. Comprehensive error handling can prevent system failures, minimize disruptions, and improve overall reliability. For example, error logging and appropriate error messages can help identify and resolve issues promptly.

Data Integrity: The system should prioritize data integrity by implementing measures such as data validation, consistency checks, and appropriate error handling during data transactions. Ensuring that data stored in the database adheres to predefined rules and constraints can enhance reliability and prevent data corruption.

**3.3.3** **Security**

The security of the Hotel Booking Management System is of utmost importance. Consider the following non-functional requirements:

Secure Communication: All communications between the client (React frontend) and the server (Node.js backend) should be encrypted using secure protocols such as HTTPS. This ensures the confidentiality and integrity of data during transmission.

Authentication and Authorization: The system should implement secure authentication and authorization mechanisms to ensure that only authorized users can access and perform actions within the system. For example, implementing JSON Web Tokens (JWT) for authentication and Role-Based Access Control (RBAC) for authorization can enhance security.

**3.3.4 User interface**

A user-friendly and intuitive interface is essential for the success of the Hotel Booking Management System. Consider the following non-functional requirements:

Responsive Design: The user interface should be responsive and adaptable to different screen sizes and devices. Implementing responsive design principles using React can ensure a consistent and user-friendly experience across various platforms.

Intuitive Navigation: The system should provide clear and intuitive navigation, enabling users to access different features and functionalities effortlessly. Well-organized menus and user-friendly controls can enhance usability.

Consistent Branding: The system's user interface should maintain consistent branding elements, such as logos, colors, and typography, to provide a cohesive and professional appearance.

**3.3.5 Hardware consideration**

To maximize the system's reach and accessibility, compatibility with various platforms and technologies should be ensured. Consider the following non-functional requirements:

Cross-Browser Compatibility: The system should be compatible with major web browsers such as Chrome, Firefox, Safari, and Edge, ensuring consistent functionality and appearance across different browser environments.

Device Compatibility: The system should be designed to work seamlessly on different devices, including desktops, laptops, tablets, and mobile phones. Implementing responsive design practices using React can facilitate compatibility across various devices.

**3.3.6** **Maintainability**

The maintainability of the Hotel Booking Management System is vital for long-term success. Consider the following non-functional requirements:

Modularity: The system should be built with modular components, allowing for easier maintenance, updates, and enhancements. Organizing backend code into reusable modules and employing React components can promote maintainability.

Documentation: Comprehensive documentation, including code comments, API documentation, and system architecture diagrams, should be provided to facilitate understand ability and future maintenance by developers and administrators.

**3.3.7**  **Error Handling and Validation**

To ensure data integrity and a reliable user experience, the Hotel Booking Management System should incorporate robust error handling and validation mechanisms:

Form Validation: The system should perform client-side and server-side form validation to ensure that users enter valid and properly formatted data. React's form validation libraries, such as Formik or Yup, can be utilized to streamline and simplify this process.

Input Sanitization: User input should be properly sanitized to prevent security vulnerabilities, such as cross-site scripting (XSS) attacks and SQL injections. Node.js can utilize security libraries like Helmet or Express Validator to sanitize and validate user input effectively.

Error Logging and Reporting: The system should log and report errors systematically to aid in troubleshooting and issue resolution. Node.js provides logging frameworks like Winston or Bunyan that can be used to capture and record errors, facilitating efficient debugging.

**3.3.8**  **Backup and Recovery**

To safeguard data and ensure system resilience, the Hotel Booking Management System should implement robust backup and recovery mechanisms:

Regular Database Backups: The system's MySQL database should be backed up regularly to prevent data loss. Automated backup scripts or tools like mysqldump can be utilized to create scheduled backups of the database.

**3.3.9** **Quality Issues**

To ensure high-quality software, the Hotel Booking Management System should address the following quality issues:

Code Quality and Testing: The system's codebase should adhere to industry best practices, including modular and maintainable code structures. Automated testing frameworks like Jest can be used to write unit tests and integration tests to verify the system's functionality.

Performance Testing: Load testing tools like Artillery or Apache JMeter can be employed to simulate high traffic scenarios and evaluate the system's performance under stress. This helps identify performance bottlenecks and ensures the system can handle expected user loads.

Security Auditing: Regular security audits and vulnerability assessments should be conducted to identify and mitigate potential security risks. Tools like OWASP ZAP or Retire.js can be utilized to scan for common vulnerabilities and outdated dependencies.

**3.3.10**  **Physical Environment**

Server Infrastructure: The system should be hosted on reliable server infrastructure that meets the performance and availability requirements. Factors like server location, network connectivity, and server specifications should be considered to ensure optimal system operation.

Power Backup: Adequate power backup solutions, such as uninterruptible power supply (UPS) or backup generators, should be implemented to protect against power outages and ensure system availability.

Temperature and Humidity Control: The server room or data center housing the system should maintain suitable temperature and humidity levels to prevent hardware damage and ensure the stable operation of the infrastructure.

**3.3.11 Documentation**

Code Documentation: The system's source code should be well-documented, following established documentation standards and best practices. Inline comments, README files, and code documentation tools like JSDoc can be used to provide clear explanations of the code's functionality, usage, and dependencies.

**3.3.12 Resource Issues**

Efficient resource management is crucial for the Hotel Booking Management System to optimize system performance and utilization. Consider the following non-functional requirements:

Memory Management: Proper memory allocation and garbage collection in Node.js to prevent memory leaks and optimize resource usage.

Database Optimization: Optimizing MySQL database performance through indexing, query optimization, and caching mechanisms.