Haramaya University

Building the Basis for Development

College of Computing and Informatics

Department of Software Engineering

SOFTWARE DESIGN DOCUMENT (SDD)

FOR

Feven HOTEL/RESTAURANT MANAGEMENT SYSTEM

Group members	ID
1. ABENEZER YOHANNES	861/13
2. EDEN BIRHANU	882/13
3. GETHAUN TAMIRAT	421/12
4. HEAVEN ENDRIS	889/13
5. NIYA MURAD	901/13

Advisor name

Mr. Dita A.

Date:

Signature

Submission date: MAY , 2025

Submitted to:DEPARTMENT OF SOFTWARE ENGINEERING

Table of content

Table 1: Table of contents

Lists of Tables pages

Lists of Figures

Definition, Abbreviation and Acronyms

Definitions

- **SDD (Software Design Document)**: A document that describes the architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements.
- **SRS (Software Requirements Specification)**: A document that provides a detailed description of the intended purpose and environment for software under development.
- **UI (User Interface)**: The means by which a user interacts with a computer system or application.
- **API (Application Programming Interface)**: A set of protocols for building and interacting with software applications.
- **ORM (Object-Relational Mapping)**: A programming technique used to convert data between incompatible type systems using object-oriented programming languages.

Abbreviations

- **HMS (Hotel Management System)**: A software application designed to manage hotel operations.
- **NFR (Non-Functional Requirement)**: Criteria that can be used to judge the operation of a system, rather than specific behaviors.
- **DBMS (Database Management System)**: Software for creating and managing databases.
- **HTML (HyperText Markup Language)**: The standard markup language for documents designed to be displayed in a web browser.
- **CSS (Cascading Style Sheets)**: A style sheet language used for describing the presentation of a document written in HTML or XML.

Acronyms

- **PHP (Hypertext Preprocessor)**: A widely-used open-source scripting language suited for web development.
- **MySQL**: An open-source relational database management system.
- **CRUD (Create, Read, Update, Delete)**: The four basic functions of persistent storage.
- **ERD (Entity-Relationship Diagram)**: A visual representation of different data entities and their relationships.
- **GUI (Graphical User Interface)**: A user interface that includes graphical elements, such as windows, icons, and buttons.

CHAPTER ONE : INTRODUCTION

1.1 document SCOPE

This complete SDD will contain the general definition and features of the project, design constraints, the overall system architecture and data architecture, a brief explanation about our current progress and schedule of the project. With the help of UML diagrams, design of the system and subsystems/modules will be explained visually in order to help the programmer to understand all information stated in this document correctly and easily.

This document outlines the system modules, their decomposition and interrelationships, data structures and dependencies, interfaces (including user interfaces and module interfaces), and potentially detailed design elements for specific components like login and registration.

The organization of this SDD is structured to guide readers logically through the design process. Beginning with this Introduction where, readers will find a comprehensive overview of the system in Chapter one , followed by details on the system architecture in Chapter two, data design and component design in Chapter three, human interface design in Chapter four, and a requirements matrix in Chapter five. Appendices are included for supplementary information such as diagrams.

This SDD aims to provide the system with a modern and efficient tool, directly addressing the operational challenges identified in the existing manual system. By detailing the design of an automated system, this document supports the project's goal to streamline operations, reduce errors, save time, improve staff productivity, and enhance convenience for both staff and customers.

1.2 PURPOSE

This SDD is intended to provide a software system design which will satisfy functional and nonfunctional requirements stated in SRS Document of Web based hotel management system . Purpose of this document is serving as a guideline throughout development phase of the project for developers.
This document is the primary reference for all software development activities, providing the necessary information for programmers to write code that accurately reflects the intended system.

The SDD provides a stable and comprehensive reference, ensuring that all team members involved in the development process are aligned under a single vision. It outlines all parts of the software and describes how they will work together. Specifically, the document details the system's architecture, module decomposition, data structures, interfaces, and algorithmic approaches.

This document supports both the preliminary and detailed design stages of the software development lifecycle. It captures high-level architectural decisions in the preliminary stage

and provides sufficient detail for individual components and data structures during the detailed design stage. By clearly defining the interactions between system components, the SDD ensures that the system will achieve the complete functionality of the system as outlined in the requirements.

Furthermore, this SDD serves multiple crucial objectives within the context of the Feven Hotel/Restaurant Management System project:

* Guiding Development: It provides the software development team with explicit instructions and specifications on how to build the system, minimizing ambiguity and potential misinterpretations of requirements.
* Facilitating Communication: It serves as a centralized source of truth for communication among designers, developers, testers, and project managers.
* Supporting Verification and Validation: It lays the foundation for verification and validation activities, enabling the development of comprehensive test plans and procedures to ensure the implemented system meets the design specifications and, consequently, the original requirements.
* Aiding Maintenance and Evolution: For future maintenance and updates, the SDD acts as a blueprint, providing developers and maintenance teams with a detailed understanding of the system's structure and components, facilitating changes and scalability enhancements.
* Ensuring Requirement Traceability: Along with the requirements matrix, the SDD helps ensure traceability, establishing clear links between requirements and design choices.

In essence, the purpose of this SDD is to provide a thorough, accurate, and easy-to-understand description of the software design, empowering the development team to efficiently build the Hotel Management System and contribute to the project's overall success in modernizing operations and improving service delivery.

1.3 Document Convention

This Software Design Document (SDD) for the Hotel Management System adheres to specific standards and typographical conventions to ensure consistency, clarity, and readability. These conventions are implemented throughout the document to facilitate easy interpretation and navigation for all intended audiences.

This document follows the Software Engineering Project standards provided by the College of Computing and Informatics, Haramaya University, which likely align with widely accepted software engineering documentation standards. Specific standards, such as the IEEE Recommended Practice for Software Design Descriptions, have also been considered in shaping the structure and content presented here.

Typographical Conventions:

* Section Headings: Chapter and section titles are presented in a larger, bold font to clearly delineate the document structure. Subsections use progressively smaller or less emphasized fonts.

* Bold Text: Bold font is used to highlight key terms, concepts, or important phrases within the text, drawing the reader's attention to critical information. This convention is used extensively to emphasize the most important parts of the response, as requested.
* Italic Text: "*Italic font"* may be used for emphasis on certain words or phrases, or to denote titles of other documents or specific system elements where appropriate.
* Source Citations: Information directly supported by the provided sources is followed by a citation in square brackets, e.g., [i] or [j, k], indicating the source number(s). When a statement is based on multiple sources, all supporting sources are listed.
* Code/Technical Terms: Specific technical terms, commands, or names of programming languages/frameworks (e.g., HTML, PHP, React, Node.js, MySQL) may be presented in a distinct font or style (though plain text is used here for consistency in this format).
* Lists: Bullet points or numbered lists are used to present items in a clear, organized manner, especially when detailing features, objectives, or lists of components.
* Diagrams and Figures: references are made to conceptual diagrams (e.g., context diagrams, UML diagrams, ER diagrams) that would typically be included in a physical SDD document to visually represent system architecture, data relationships, or workflow.
* Cross-referencing: References to other sections within this document or to external documents (like the SRS) are made using section numbers or document titles to facilitate navigation and traceability.

1.4 Intended Audience

This Software Design Document (SDD) is intended for a diverse group of stakeholders involved in the Feven Hotel/Restaurant Management System project. Each group will utilize this document for different purposes, focusing on the sections most relevant to their roles and responsibilities. Understanding the needs of this varied audience has influenced the level of detail and organization within this document.

The primary intended audiences for this SDD are:

* Software Designers: These individuals were involved in creating the design and will use this document to ensure the captured design is accurate and complete. They might also refer back to it during subsequent design stages or modifications.
* Software Development Team: This is a key audience. Developers will use this SDD as the primary reference for writing the code that implements the system. They require detailed information about component design, module interfaces, data structures, and potentially algorithms to translate the design into functional software.
They will consult sections on system architecture, component design, and data design.
* Project Managers: Project managers use the SDD to gain an overview of what needs to be built and how. They focus on the system overview, architectural design, and overall structure to monitor project progress, manage resources, and ensure the development aligns with the project plan and scope.
* Quality Assurance (QA) Team / Testers: The QA team will refer to this document, alongside the SRS, to design test cases and plans. They use the design details, particularly the functional descriptions and expected system behavior outlined in the design, to verify that the implemented system functions correctly and meets the specified requirements. They

might be interested in the requirements matrix to ensure all requirements are addressed by the design and implementation.
* Maintenance and Support Teams: Once the system is deployed, maintenance and support personnel will rely on the SDD to understand the system's structure, components, and dependencies when performing updates, bug fixes, or enhancements. The detailed design information is crucial for efficiently diagnosing issues and implementing changes.
* Documentation Writers: Individuals responsible for creating user manuals and help documentation for the Feven Hotel/Restaurant staff and managers will use the SDD to understand the system's functionalities and how different components interact. They will draw from the descriptions of user interfaces and system processes.
* Future Developers/Teams: If new developers join the project or if the system is handed over to a different team, the SDD serves as essential documentation to quickly bring them up to speed on the system's design and architecture.

While End Users (e.g., hotel managers, receptionists, customers) are not typically direct readers of the SDD, their requirements and characteristics as defined in the SRS have been the driving force behind the design decisions documented here. They are more likely to interact with user documentation derived from this SDD.

CHAPTER TWO: SYSTEM ARCHITECTURAL DESIGN

This chapter details the architectural design of the Feven Hotel/Restaurant Management System. It begins with a high-level overview of the system, summarizing its purpose, scope, and significance, and outlining the technologies used. Subsequently, it discusses the non-functional requirements that guide the design and establishes the key design goals. The chapter then delves into the decomposition of the system into manageable subsystems, defining their roles and relationships. Finally, it describes the system architecture using the 4+1 View Model, presenting different perspectives suitable for various stakeholders.

2.1. System Overview

This section provides a summary of the critical information regarding the Feven Hotel/Restaurant Management System project, including its objectives, the problem it addresses, its scope, its significance, and the technologies and tools that will be employed in its development. It also explains the overall workflow of the system.

Project Objectives: The general objective of this project is to develop and implement a web-based system that streamlines operations, improves efficiency, and enhances customer satisfaction at Feven Hotel/Restaurant. Specifically, the system aims to automate key processes, reduce errors, save time, improve staff productivity, and offer greater convenience for customers through online services. It is intended to provide Feven Hotel/Restaurant with a modern and efficient tool for managing their business.

Problem Statement: Feven Hotel/Restaurant currently faces significant operational challenges due to its reliance on inefficient manual processes for managing reservations, orders, inventory, and customer information. This manual approach leads to inefficiencies, errors, difficulties in tracking performance, and issues associated with handling large physical files and calculation errors. Manual record-keeping can also lead to a high risk of confidential data loss. These issues negatively impact service delivery and potentially result in revenue loss and reduced staff productivity. The proposed system is motivated by these challenges and seeks to overcome the problems associated with the outdated manual process.

Scope: The Feven Hotel/Restaurant Management System is designed as a web-based platform accessible via the internet through standard web browsers. Its scope is defined by the functionalities needed to automate and streamline major operations specifically for Feven Hotel/Restaurant. The key functionalities included within the scope are:

* Online table reservations
* Online ordering for both dine-in and takeout
* Menu management
* Order management
* Inventory management

* Sales reporting
* Customer management

The system is intended to replace or significantly improve upon the current manual system, which primarily relies on phone calls and spreadsheets.

Limitations (as part of the scope): It is important to note the boundaries and limitations of this project as defined in the proposal. The system will not include integration with third-party accounting software. Payment gateway integration will be limited to a single specified payment method or gateway. The project will not include the development of a mobile application at this stage. The project timeline is limited to six months.

Significance: This project holds significant importance as it provides Feven Hotel/Restaurant with a modern and efficient tool for managing their operations. By automating key processes, it will reduce errors, save time, and improve staff productivity. The enhanced customer experience through online reservations and ordering is expected to increase sales. Furthermore, this project serves as a practical example of a web-based management system suitable for small businesses, contributing to the field of software engineering. Stakeholders, including the owner, manager, staff, and customers, will benefit from increased efficiency, improved service, and enhanced convenience.

Technologies and Tools: The system will be developed using a specific technology stack. This includes HTML, PHP, React, Node.js, and MySQL. This combination is considered suitable for developing a scalable and reliable web application. PHP and MySQL are commonly used for online hotel management systems and databases. MySQL is suitable for storing data. React and Node.js offer modern frameworks for building responsive and efficient web interfaces and backend services. The system will require hardware such as a computer with sufficient RAM (e.g., 512MB or more), monitor, keyboard, and mouse, and software including an operating system (Windows or Linux), a web server (WAMP or XAMPP), and a web browser. A database backend like MySQL or SQL Server is needed.

Overall Workflow: The overall workflow of the system involves two primary groups of users: customers and hotel/restaurant staff (including managers, receptionists, etc.). Customers interact with the online interface to perform actions like making online table reservations and placing online food/drink orders. Staff and administrators utilize the system through a potentially different interface or restricted views to manage these customer activities, update system data (such as menu items, inventory levels, room statuses), and generate various reports (including sales reports). The system acts as a central hub for all operational data and processes, facilitating communication and data flow between different functions like reservations, ordering, inventory, and reporting. The system is designed to handle transactions automatically and use a database to retrieve needed information.

showing major data flows to and from key external actors.

* System Boundary: The core web-based Feven Hotel/Restaurant Management System.
* External Entities/Actors:

* Customers: Interact with the system for online reservations and ordering. They send reservation requests, order details, and potentially payment information (if a payment gateway is integrated within the single allowed method). They receive confirmation, order status updates, and potentially access hotel/restaurant information.
* Hotel/Restaurant Staff (including Manager, Receptionist): Interact with the system to manage operations. They input/update data (room status, menu, inventory), process orders and reservations, manage customer information, and generate reports. They receive system notifications, reports, and operational data views.
* Chapa is the selected external payment gateway integrated into the Hotel Management System (HMS) to securely process online payments. The HMS sends payment requests to Chapa via its public API, which redirects users to a secure hosted payment page. After the customer completes the payment, Chapa returns transaction status and details to the HMS through a secure callback (webhook). The system uses this confirmation to update the reservation or order status accordingly.
* Administrator (Manager/Owner): Overlaps with staff but has higher privileges, including full access, viewing financial reports, making decisions based on data. Interacts for overall system configuration, user role management, and comprehensive reporting.

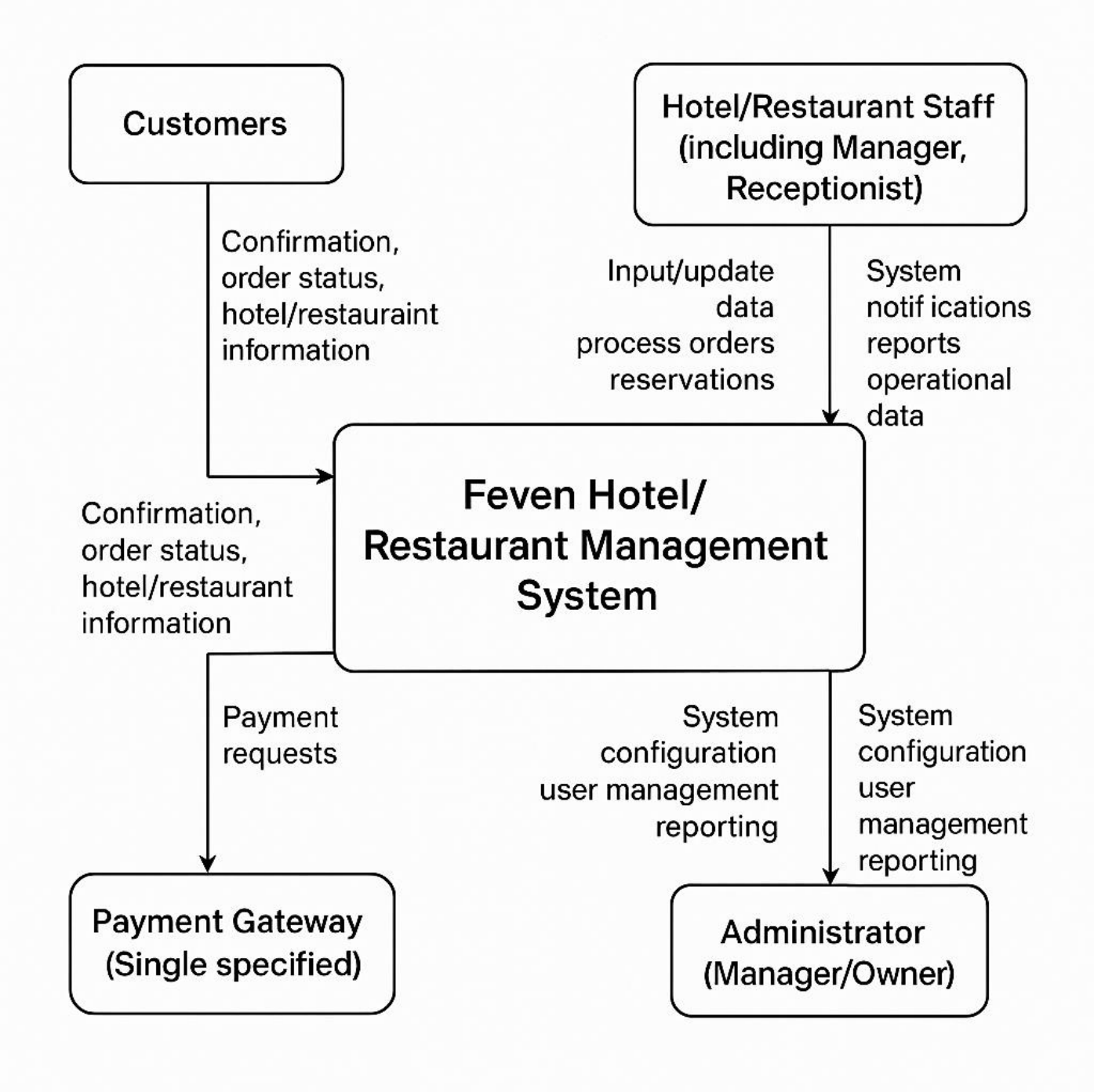


Figure 1: subsystem relationship of Feven hotel/Restaurant Management system

2.2 Design Goals

The design goals for the Feven Hotel/Restaurant Management System are derived from the non-functional requirements (NFRs) specified or implied for the system. These goals explain how the design intends to achieve these requirements and the rationale behind key design decisions. While the SRS document outlines the detailed requirements, including NFRs, the design goals articulate the targets for the system's quality attributes and operational characteristics.

Key Design Goals Based on Implied and Explicit NFRs:

1. Usability: The system aims to be user-friendly and easy to navigate for both customers and staff.

* Rationale: A user-friendly interface is crucial for encouraging customer adoption of online services and improving staff productivity by reducing the learning curve and errors associated with the previous manual system. The design prioritizes intuitive layouts and clear workflows. Screen images and descriptions are planned in the Human Interface Design section to ensure a positive user experience.

2. Performance: The system should provide fast response times for user interactions, such as searching room availability, making reservations, placing orders, and generating reports.

* Rationale: Slow performance can lead to customer frustration and reduced staff efficiency. The design leverages technologies like React for a responsive frontend and aims for efficient database queries using MySQL to retrieve information quickly. The choice of a multi-tiered architecture (as described in 2.4) separates concerns and can help optimize performance.

3. Reliability: The system must be reliable, ensuring that transactions are processed accurately and consistently, and data is maintained without loss.

* Rationale: Accuracy is paramount in managing reservations, orders, billing, and inventory. The manual system was prone to errors. The design incorporates data validation rules and aims for robust backend logic using technologies like PHP to handle operations reliably. A well-designed and properly maintained database (MySQL) is the backbone for data reliability. Fault tolerance might be considered if redundant servers are part of the environment.

4. Availability: The web-based system should be available to customers and staff whenever needed, ideally 24/7 for online customer access.

* Rationale: Continuous availability is essential for online booking and ordering services. While acknowledging potential limitations like power outages, the design aims for high uptime based on the hosting environment. Constant system flow checkup is a security goal, but also supports availability.

5. Scalability: The system should be able to handle an increasing number of users and data volume as Feven Hotel/Restaurant grows.

* Rationale: Scalability ensures the system can support future business expansion without significant re-architecture. The chosen technologies and the multi-tiered architecture are considered suitable for building a scalable web application. Separating presentation, application, and data layers allows for scaling individual components as needed.

6. Security: Protecting sensitive customer and business data (like personal information, transaction details, inventory data) is a critical goal.

* Rationale: The manual system had a high risk of data loss. The design incorporates security measures such as user authentication (secure login/registration with different user roles/authorizations), role-based access control, and aims to protect against common web vulnerabilities Data security is a key aspect of database design. Constant security updates are a related goal.

7. Maintainability: The system should be designed in a modular and well-structured way to facilitate future maintenance, updates, and enhancements.

* Rationale: A maintainable system reduces the cost and effort required for bug fixes, feature additions, and adaptations. Modular design with clear interfaces helps developers

understand and modify the system. Using established frameworks and following coding conventions contributes to maintainability.

2.3. Subsystem Decomposition

The Feven Hotel/Restaurant Management System is a complex application, and its design involves decomposing the large system into smaller, more manageable subsystems or modules. Each subsystem is a collection of related functionalities and data that can potentially be developed and tested with a degree of independence. This decomposition clarifies the system's structure, facilitates parallel development, and enhances maintainability. The relationships and dependencies between these subsystems are also defined.

Based on the scope and required functionalities, the system can be decomposed into the following major subsystems:

1. **User and Authentication Management Subsystem:**
   * Responsibilities: Handling user registration, login, authentication, session management, and managing different user roles (Customer, Staff, Manager/Admin, Owner) and their respective access permissions.
   * Key Functions: User registration, login verification, password management (forgot/reset), session creation/validation, role assignment, permission checks.
2. **Reservation Management Subsystem:**
   * Responsibilities: Managing all aspects of room and table reservations, including checking availability, booking, modifying, and canceling reservations. It also handles room status updates.
   * Key Functions: Search availability (by date, room type, capacity, table size), create reservation, retrieve reservation details, modify reservation, cancel reservation, assign rooms, check-in, check-out, update room status (e.g., 'Needs Cleaning'). Interacts with User, Room (part of Inventory/Data Management), and potentially Payment subsystems.
3. **Order Management Subsystem:**
   * Responsibilities: Handling online food/drink orders for dine-in and takeout, including placing orders, managing order status, and processing order details.
   * Key Functions: View menu, add items to cart, place order, retrieve order details, update order status (e.g., 'Received', 'Preparing', 'Ready', 'Served', 'Completed'), associate orders with tables/reservations or takeout customers. Interacts with Menu (part of Inventory/Data Management) and potentially Payment subsystems.
4. **Inventory Management Subsystem:**
   * Responsibilities: Managing the hotel's inventory, including rooms, menu items, and supplies (like utensils or food ingredients if detailed tracking is needed). It also involves maintaining details about these items.
   * Key Functions: Add/edit/delete room details (type, capacity, price, status), add/edit/delete menu items (name, description, price, category), update stock levels for supplies (if applicable), monitor low stock alerts. Interacts with Reservation (for room status/availability), Order (for menu details and deducting stock), and Reporting subsystems.
5. **Customer Management Subsystem:**

* Responsibilities: Storing and managing customer details and historical data.
* Key Functions: Create customer profile, view customer details, update customer information, search customer records. Linked to Reservation and Order subsystems to associate activities with customers.

6. **Reporting Subsystem:**
* Responsibilities: Generating various reports for administrators and managers based on the data in the system.
* Key Functions: Generate sales reports (daily income, periodic sales), occupancy reports, reservation reports, order reports, potentially inventory reports. Accesses data from Reservation, Order, and Inventory subsystems. Provides data analysis for decision-making.
7. **System Administration Subsystem:**
* Responsibilities: Providing tools for administrators to configure system settings, manage users (beyond basic authentication), perform backups, and monitor system health.
* Key Functions: User account management (create, edit, delete staff/admin accounts), system configuration settings, access logs, backup/restore functions. Overlaps with User Management but focuses on administrative tasks.

Relationships and Dependencies:
These subsystems interact closely:
* Authentication Management is depended upon by all other subsystems requiring user access (Reservation, Order, Inventory, Customer, Reporting, System Admin).
* Reservation Management depends on User Management (to link reservations to users) and Inventory Management (for room availability and status).
* Order Management depends on User Management (to link orders to users), Inventory Management (for menu details and stock), and potentially Reservation Management (for dine-in orders linked to tables/reservations).
* Inventory Management is accessed by Reservation Management (room data), Order Management (menu data), and Reporting Subsystem.
* Customer Management is accessed by Reservation Management and Order Management to retrieve or update customer profiles.
* Reporting Subsystem depends on data from Reservation, Order, and Inventory Management subsystems.
* System Administration interacts with User Management and other subsystems for configuration and monitoring.

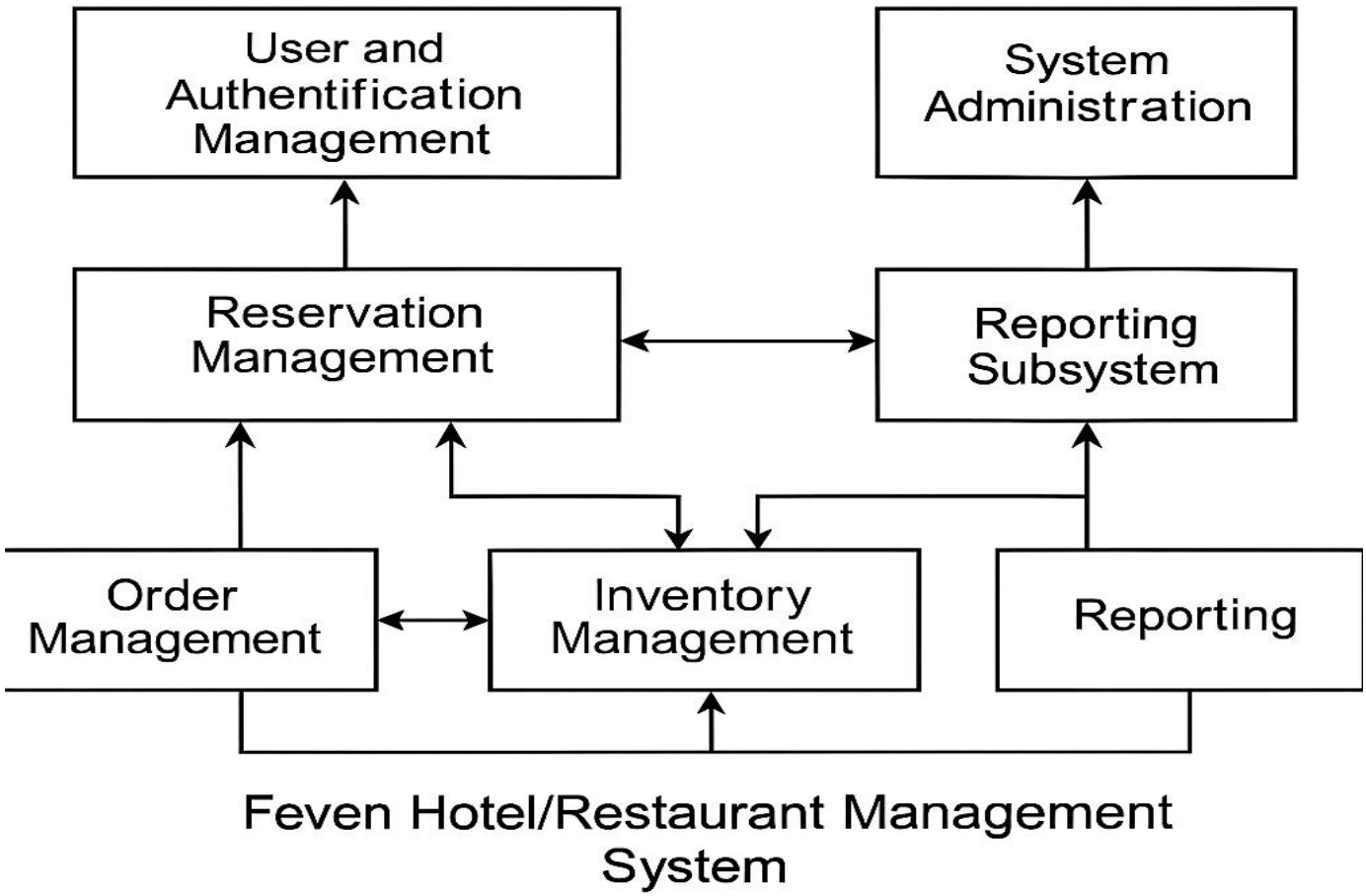


Figure 2 : package diagram for subsystem decompostion

2.4. 4+1 Architecture View Model

The 4+1 View Model is used to describe the architecture of software-intensive systems from the viewpoint of different stakeholders. It consists of four main views: Logical, Process, Development, and Physical, plus selected use cases or scenarios (+1 view) which instantiate and validate the architecture. This section describes the Feven Hotel/Restaurant Management System using these views.

2.4.1. Logical View

The Logical View is concerned with the system's functionality as it pertains to end-users and the main structural elements that provide this functionality. It represents the object model of the design, identifying major classes, their attributes, relationships, and significant methods. This view decomposes the system into key abstractions that are important from a functional perspective.

* Key Abstractions (Conceptual Classes):
    * User: Represents any user of the system (Customer, Staff, Manager, Owner). Attributes might include userID, username, password , role, contact information. Methods might include login, logout, update profile. (Relates to User Management subsystem).

* Customer: Extends User or is linked to a User. Represents a customer making reservations or placing orders. Additional attributes: name, address, nationality, ID proof details. Methods: make reservation, place order, view history. (Relates to Customer Management subsystem).
    * Room: Represents a hotel room. Attributes: roomNumber, roomType, capacity, price, status (Available, Reserved, Occupied, Needs Cleaning), description, picture. Methods: check status, update status. (Part of Inventory Management).
    * Reservation: Represents a booking for a room or table. Attributes: reservationID, userID (linking to Customer), roomID/tableID, checkInDate, checkOutDate, numberOfGuests, status (Confirmed, Pending, Cancelled), bookingDate, totalCost. Methods: create booking, modify booking, cancel booking, confirm check-in, confirm check-out. (Relates to Reservation Management subsystem).
    * Table: Represents a restaurant table (for restaurant aspect). Attributes: tableNumber, seatingCapacity, status (Available, Occupied, Reserved). Methods: check status, update status. (Part of Inventory Management, used by Reservation for tables).
    * MenuItem: Represents an item on the restaurant menu. Attributes: itemID, name, description, price, category. Methods: get details. (Part of Inventory Management).
    * Order: Represents a customer's food/drink order. Attributes: orderID, customerID/reservationID (linking to Customer/Reservation), orderDate/Time, status (Pending, Processing, Completed, Cancelled), orderType (Dine-in, Takeout), totalAmount. Methods: create order, add item to order, update status, calculate total. (Relates to Order Management subsystem).
    * OrderItem: Represents a specific item within an Order. Attributes: orderItemID, orderID (linking to Order), itemID (linking to MenuItem), quantity, unitPrice, subtotal.
    * Bill/Invoice: Represents the final charge for a guest's stay or order. Attributes: billID, reservationID/orderID, issueDate, totalAmount, paymentStatus. Methods: generate bill, add charges, process payment.
    * Employee/Staff: Extends User or is linked to a User. Represents hotel/restaurant staff. Additional attributes: employeeID, role (Receptionist, Housekeeping, Kitchen Staff), contact info. Methods: perform tasks based on role (e.g., Receptionist manages check-in/out, Housekeeping updates room status). (Relates to User/Staff Management).
    * Report: Represents a generated report. Attributes: reportID, reportType (Sales, Occupancy, etc.), creationDate, parameters (date range etc.). Methods: generate report. (Relates to Reporting subsystem).

* Relationships: Relationships between these classes would include associations (e.g., a Customer *makes* a Reservation), aggregations (e.g., an Order *contains* OrderItems), and potentially inheritance (e.g., Customer and Employee inheriting from User). For instance, a User can have one general information (1:1 relation between userRegistrationInfo and userGeneralInfo). A user can book several rooms, and a room can be reserved by one user (referring to data dependency description).

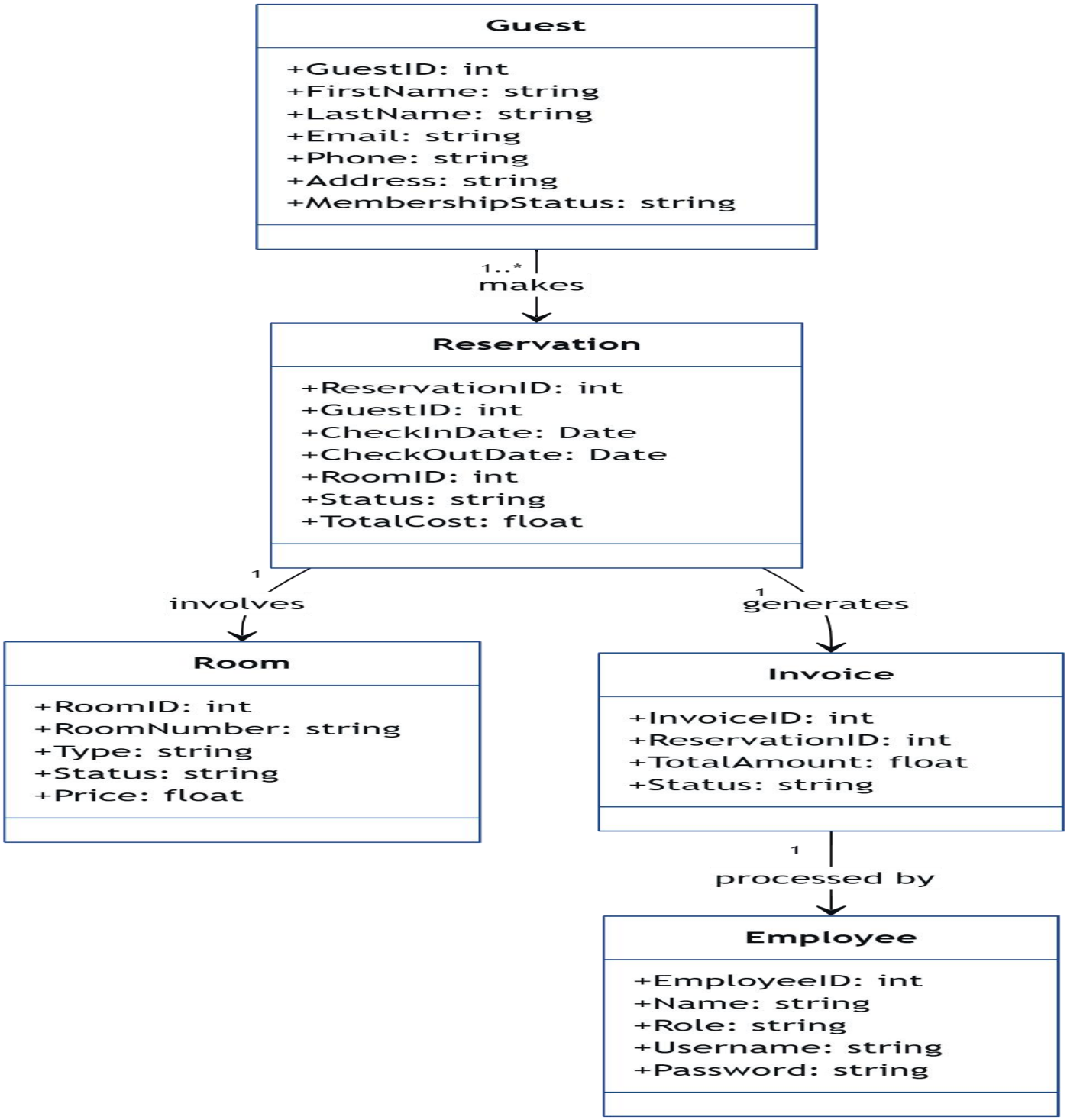
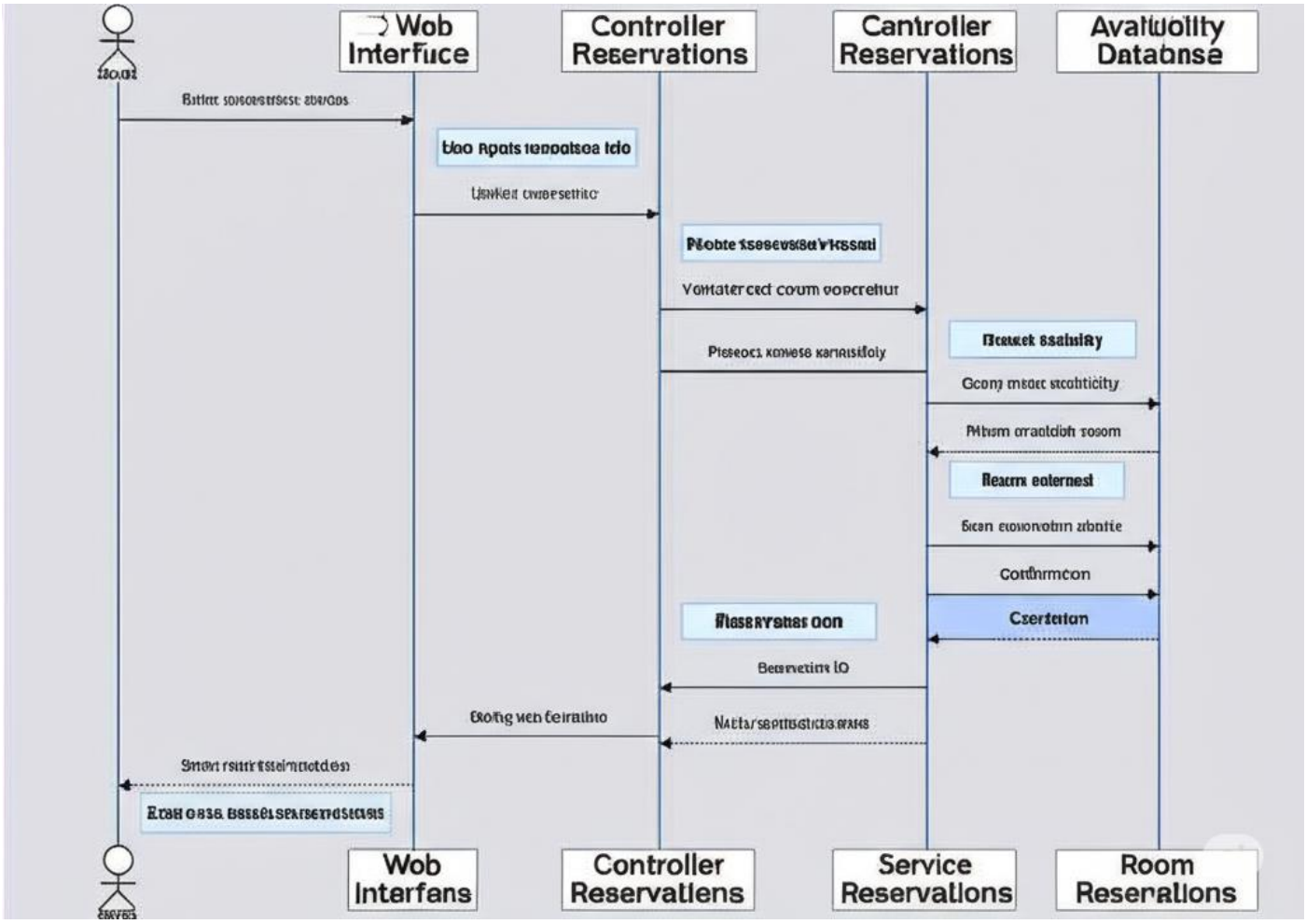


Figure 3 : logical View model

2.4.2. Process View

The Process View describes the system's concurrent processes and how they interact. It focuses on the runtime behavior, illustrating how tasks are executed, how processes communicate, and how they handle synchronization. For a web-based system, this view often relates to how the backend server handles multiple concurrent requests from users and how different parts of the application execute simultaneously.

* Key Processes:
    * User Request Handling Process: This is the main process that receives incoming requests from client browsers (customers and staff). It involves parsing the request, routing it to the appropriate application logic, and preparing a response. The system must be designed to handle many of these processes concurrently to serve multiple users simultaneously.
    * Application Logic Processes: These processes execute the core business logic of the system within the backend (PHP). include:
        * Reservation Processor: Handles the steps involved in checking availability, creating/updating/cancelling reservations.
        * Order Processor: Manages the lifecycle of an order from creation to completion.
        * Inventory Manager: Updates stock levels and item details.
    * Database Interaction Process: This process handles communication with the MySQL database. It receives requests from Application Logic Processes (e.g., retrieve room availability, save a new reservation, update inventory) and executes database operations. This process might involve connection pooling to efficiently manage database connections under heavy load.
    * Background Processes (Potential): Depending on detailed requirements, there might be background processes for tasks like sending automated email confirmations, generating scheduled reports, or performing data backups.
* Interactions: Processes communicate by passing data (e.g., request parameters, database query results, processed data). Synchronization mechanisms are needed to manage shared resources, particularly database access, to prevent data corruption when multiple processes try to modify the same data simultaneously. The multi-tiered architecture naturally separates some processing, with the frontend handling UI logic and the backend managing application and data logic.



.

Figure 4 : Process View

2.4.3. Development View

The Development View (also known as the Implementation View or Module View) describes the static organization of the software modules and components as they would be organized in the development environment. It focuses on the structure that is visible to programmers. This view helps manage code, facilitate build processes, and assign work to development teams.

* Layered Architecture: A common and suitable architectural style for a web-based system like this is a multi-layered architecture. Based on the chosen technologies, the system can be organized into layers:

* Presentation Layer (Frontend): Built using React, HTML, and CSS. This layer is responsible for the user interface and user interaction logic. It runs in the user's web browser. It communicates with the Application Layer via APIs. Components here would include pages (e.g., Home, Rooms, Reservation, Menu, Cart, Login, Dashboard), UI elements (buttons, forms, tables), and frontend logic for validation and interactivity.

* Application Layer (Backend): Built using PHP. This layer contains the core business logic and acts as an intermediary between the Presentation Layer and the Data Layer. It receives requests from the frontend, processes them, interacts with the Data Layer, and sends responses back to the frontend. This layer would be organized into modules corresponding to the subsystems identified in 2.3 (e.g., User Management Module, Reservation Module, Order Module).

* Data Access Layer: This layer is typically part of the backend and is responsible for accessing the database. It contains code that translates requests from the Application Layer into database queries (SQL) and converts database results back into a format usable by the application logic.

Using an Object-Relational Mapper (ORM) could be part of this layer.

* Database Layer: This layer consists of the MySQL database server. It stores all the system's data, including user information, reservation details, room data, menu items, orders, and reports. It is managed by the Database Interaction Process described in the Process View.

* Component Organization: Within these layers, code would be organized into folders, packages, or modules corresponding to specific functionalities or subsystems (e.g., a reservations package in the backend, a components/users folder in the frontend). Libraries and frameworks (React, Node.js, PHP, potentially Express for Node.js) would be included as dependencies.

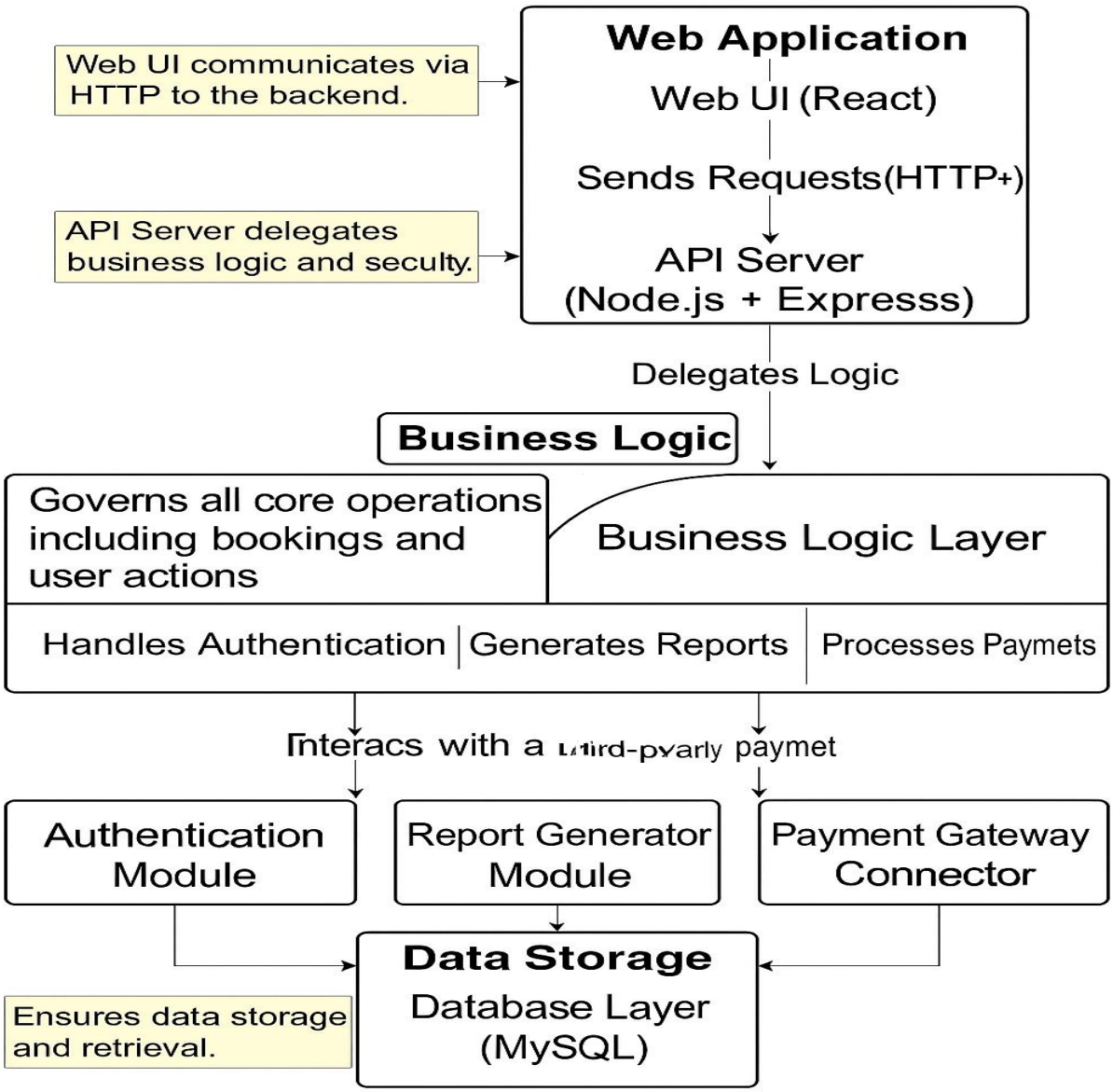


Figure 5 : development View

2.4.4. Physical View

The Physical View describes the mapping of the software components onto the hardware (or virtual hardware) environment. It illustrates the distributed aspects of the system, showing where processes and data are physically located and how they communicate over a network.

* Hardware/Environment Components:
* Client Machines: Desktop computers, laptops, tablets, or smartphones running web browsers. These devices host the Presentation Layer (frontend) and communicate with the server over the internet.
* Web Server(s): Server machine(s) hosting the Application Layer (backend) and serving the frontend files (HTML, CSS, JavaScript) to the client browsers. This could be hosted on a hosting server or a local test server. A web server like Apache or Nginx might be used in conjunction with Node.js or PHP processing. This server handles incoming HTTP requests.

* Database Server(s): Server machine(s) hosting the MySQL database. This server stores and manages the system's persistent data. It communicates with the Web Server (specifically the Data Access Layer within the Application Layer) over a network connection (often within the same data center or local network for performance and security).

* Deployment and Communication:
    * The frontend application (built with React, HTML, CSS) is deployed on the Web Server. When a user accesses the system URL in their browser, the server sends these files to the client machine.
    * The backend application (built with Node.js/PHP) is also deployed on the Web Server or potentially a separate Application Server depending on the setup.
    * The MySQL database is installed and runs on the Database Server.
    * Client browsers communicate with the Web Server using the HTTP/HTTPS protocol to send requests (e.g., login, book room, place order) and receive responses (e.g., user data, availability lists, confirmation messages).
* The Application Layer on the Web Server communicates with the Database Server using database-specific protocols (e.g., TCP/IP with MySQL protocol) to store and retrieve data.

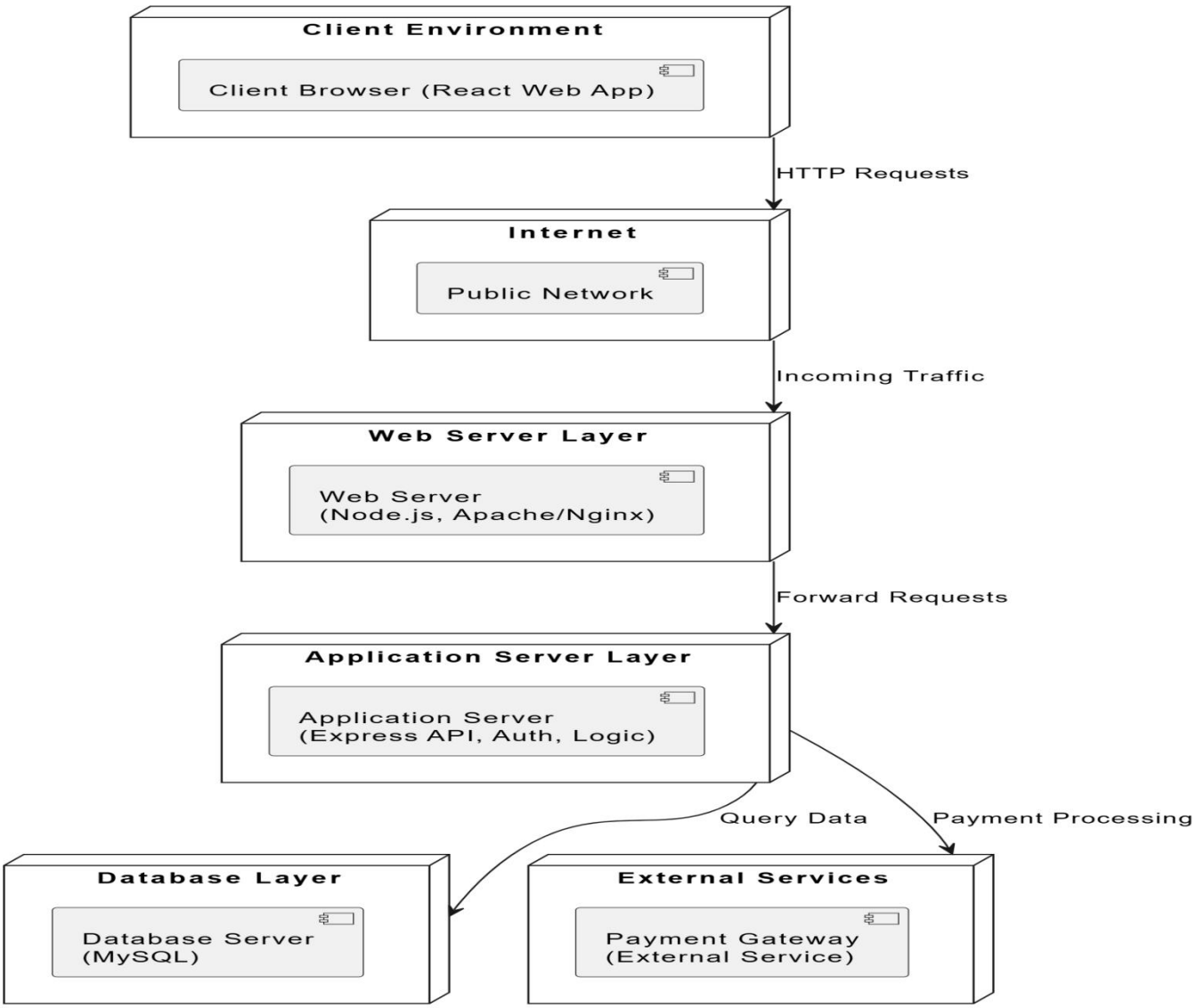


Figure 6 : Physical View

Chapter 3: Data Design

This chapter describes how the information domain of the system is transformed into data structures. It outlines how the major data entities will be stored, processed, and organized. The sections below cover the logical representation of the data, the physical implementation details, and a comprehensive data dictionary.

3.1 Logical Data Model

The logical data model for the Web-Based Hotel Management System defines the abstract structure of the data without considering how it is physically stored. It reflects the business entities, their attributes, and the relationships among them, derived directly from the functional requirements in the SRS.

This model ensures data integrity, supports normalization, and acts as the basis for creating a relational schema in MySQL.

Entity Descriptions and Relationships

Guest: Represents customers interacting with the hotel.

Reservation: Captures booking details including dates and associated guest and room.

Room: Describes each physical room with status, type, and pricing.

RoomType: Categorizes rooms (e.g., Standard, Deluxe).

Employee: Represents system users such as receptionists, housekeepers, and managers.

Invoice: Holds financial summaries per reservation.

InvoiceItem: Contains line-by-line billing details linked to an invoice.

Payment: Stores information about payments made towards invoices.

InventoryItem: Represents consumables in the restaurant or minibar.

Order: Captures food or drink orders linked to a reservation or table.

OrderItem: Details each item ordered.

Feedback: Allows customers to submit complaints or feedback.

ActivityLog: Tracks critical user/system actions for audit and security.

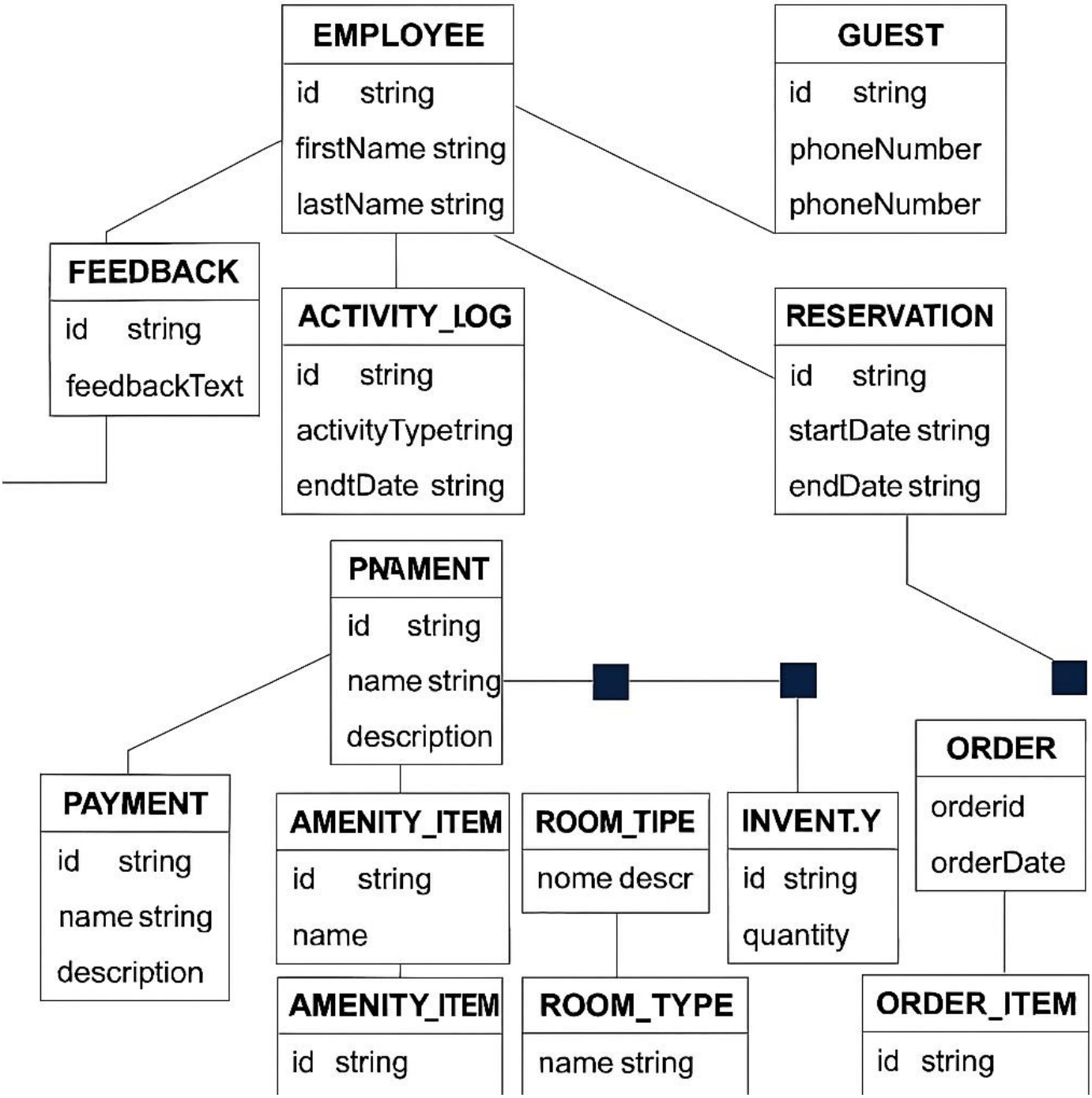


Figure 7: Enhanced Entity-Relationship Diagram

3.2 Physical Data Management

This section details the actual database or data management approach used to store the system's data. For the Feven Hotel/Restaurant Management System, the chosen database technology is MySQL. MySQL is a Relational Database Management System (DBMS).

The physical data model implements the logical model within the constraints and features of MySQL. This involves defining the specific SQL CREATE TABLE statements, choosing appropriate data types (e.g., INT, VARCHAR, DATE), specifying primary and foreign keys to enforce relationships and data integrity, and defining indexes for performance optimization.

Considerations for physical data management include:

* Data Storage: Defining the table schemas and relationships in MySQL based on the logical model.
* Data Access: Designing queries for efficient data retrieval and manipulation to meet performance requirements.
* Security: Implementing database-level security measures, such as user roles and permissions, to protect sensitive data. While the full scope of integration with third-party accounting or payment gateways is limited, securing the stored customer and transaction data in the database is crucial.
* Backup and Recovery: Planning procedures for backing up the database and recovering data in case of system failures (though detailed operational procedures might be outside the SDD scope, the design should accommodate this).

The structure of tables in the database will be created using the MySQL DBMS. This involves translating the entities and attributes from the logical model into physical tables and columns, selecting appropriate data types and constraints specific to MySQL.

3.3 Data Dictionary

The data dictionary provides detailed descriptions of all data elements used within the system. It serves as a central repository for information about the data, including attributes, their data types, memory sizes, and descriptions.
The data dictionary will list each table and its columns, providing details for each column.

Entity/Table	Attribute/Column	Data Type	Size	Description
Guest	GuestID	INT	-11	Primary key, unique identifier for the guest
Guest	EmailAddress	VARCHAR	-255	Guest's email address
Guest	PhoneNumber	VARCHAR	-20	Guest's phone number
Room	RoomNumber	VARCHAR	-50	Unique identifier for the room
Room	RoomType	VARCHAR	-50	Type of room (e.g., Single, Double, Suite)
Room	Status	VARCHAR	-20	Current status (e.g., Available, Occupied, Dirty)
Reservation	ReservationID	INT	-11	Primary key, unique identifier for the reservation
Reservation	GuestID	INT	-11	Foreign key referencing the Guest table
Reservation	RoomNumber	VARCHAR	-50	Foreign key referencing the Room table
Reservation	CheckInDate	DATE		Date of check-in
Reservation	CheckOutDate	DATE		Date of check-out
Reservation	Status	VARCHAR	-20	Status (e.g., Confirmed, Checked-in, Cancelled)

Table 2 : Data dictionary

Chapter 4: Human Interface Design

This chapter describes the design of the user interface (UI) for the Feven Hotel/Restaurant Management System. It outlines how users will interact with the system, the visual appearance of key screens, and the principles guiding the UI design to ensure usability, efficiency, and user satisfaction. The goal is to provide a user-friendly platform accessible via the internet that streamlines operations and enhances customer satisfaction.

4.1 Overview of User Interface

The user interface is the primary point of interaction between the users (Customers, Staff, Administrators) and the Feven Hotel/Restaurant Management System. As a web-based system, the interface will be accessible through standard web browsers on various devices, with consideration for mobile-friendly web design. The design prioritizes ease of use and efficiency to improve staff productivity and provide convenience for customers.

From the user's perspective, the system will provide intuitive workflows for core functionalities such as online table reservations, online food/drink orders, managing menu details, tracking inventory, managing customer information, and generating reports.

* Customers will interact with the public-facing sections of the system to view vacant room information, check price ranges, make and cancel room bookings, and place online food/drink orders (for dine-in or takeout).
* Staff will use the system to manage daily operations, including handling reservations (both online and phone-based inputs), processing orders, checking guests in/out, managing room status, and potentially managing inventory levels. They require an interface that allows efficient tracing and management of transactions and reports.
* Administrators (and potentially General Managers) will have access to broader management functions, such as managing room categories and listings, managing users (staff, customers), updating menu and inventory details, and generating essential sales and operational reports.

The system will provide feedback information to the user at each step, such as confirmation messages for successful actions, error messages for invalid inputs or failures, and real-time updates on status (e.g., reservation status, order status, room availability). Descriptive messages and tooltips will be displayed to help users understand operations and buttons. Immediate and reversible actions, along with feedback, will make the system more forgiving.

Key design goals for the user interface include simplicity, consistency, and predictability across different modules and user roles. Navigation should be easy and sensible, providing meaningful paths and exits. Related elements will be grouped together to improve clarity and usability.

4.2 Screen image

This section presents visual representations of key screens within the system, along with detailed descriptions of their purpose, elements, and associated actions. These images are design mockups illustrating the application of the principles outlined above and demonstrating how functional requirements are met through the interface. Accurate drawings or screenshots of the prototype will be included here.

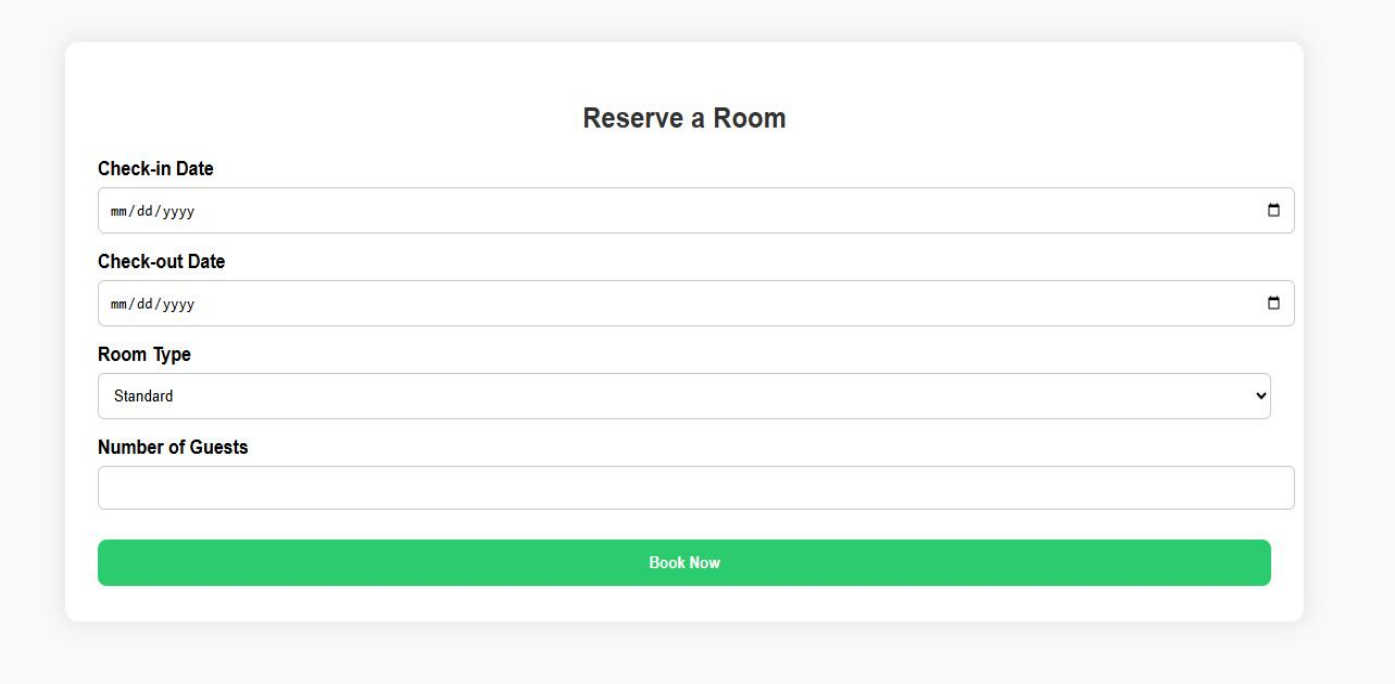


Figure 8 : Guest – Room Booking Page

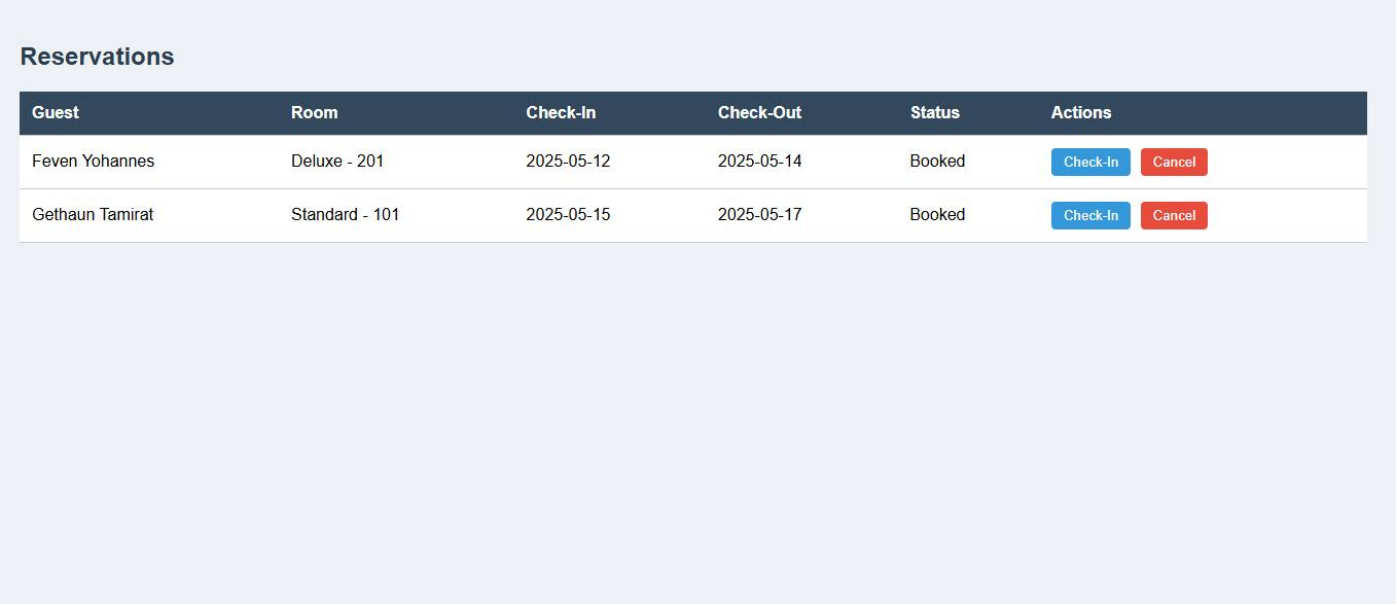


Figure 9 : Receptionist – Reservation Management Panel

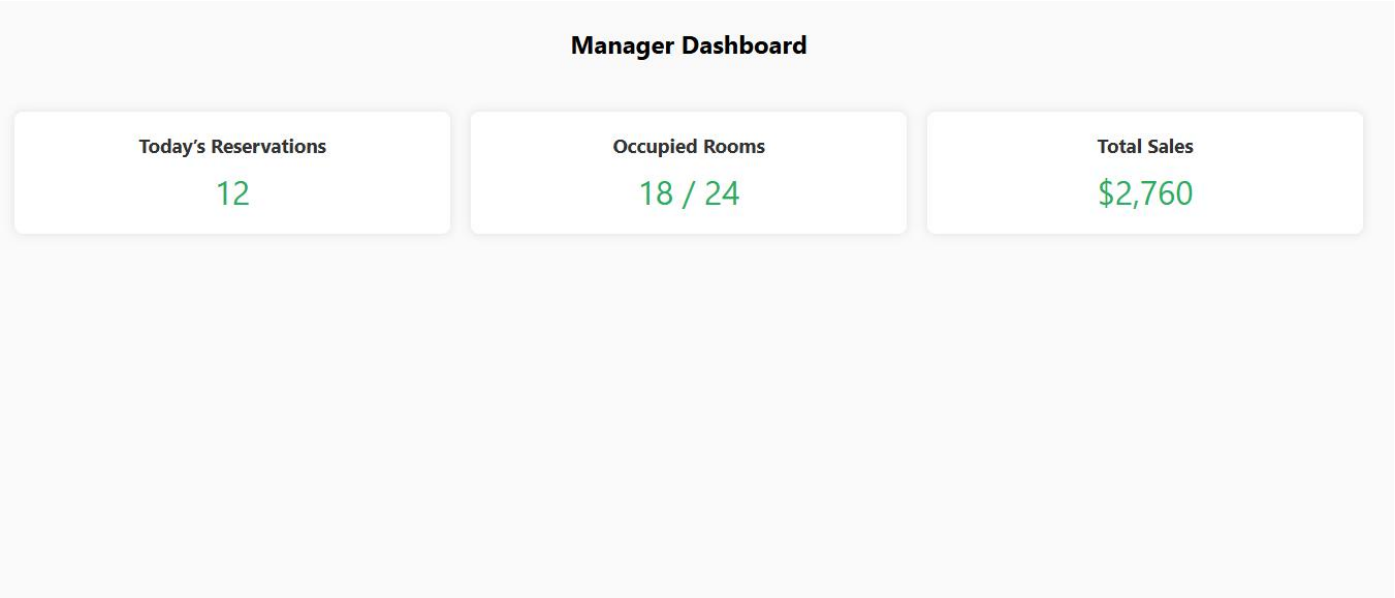


Figure 10 : Manager – Dashboard Overview

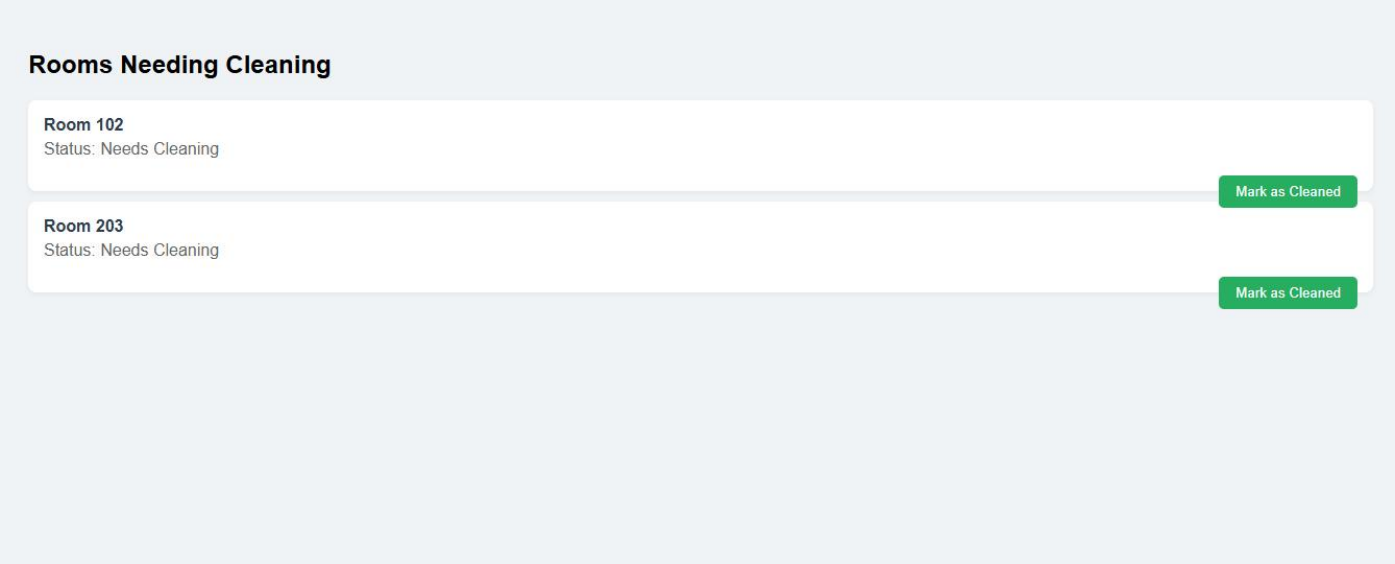


Figure 11 : Housekeeping – Cleaning Task List

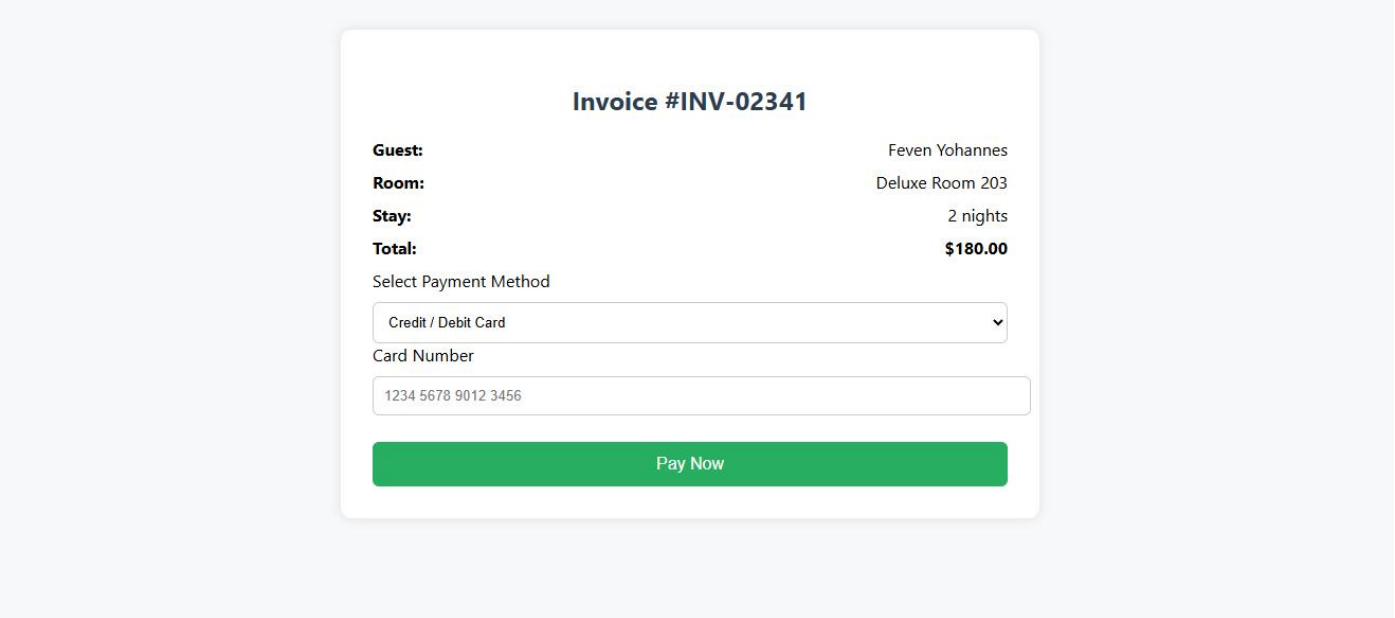


Figure 12 : Guest – Payment Page (Invoice & Checkout)

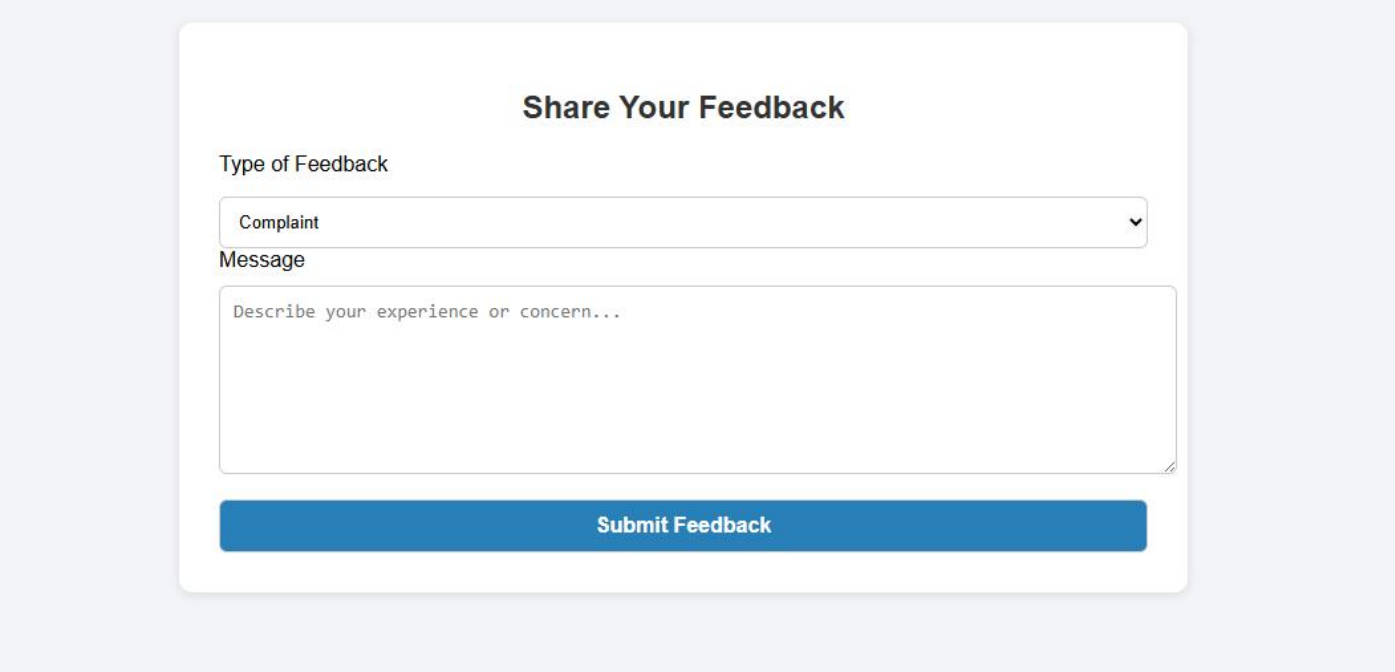


Figure 13 : Guest – Feedback / Complaint Form

Figure 14 : Manager – Sales Report Viewer

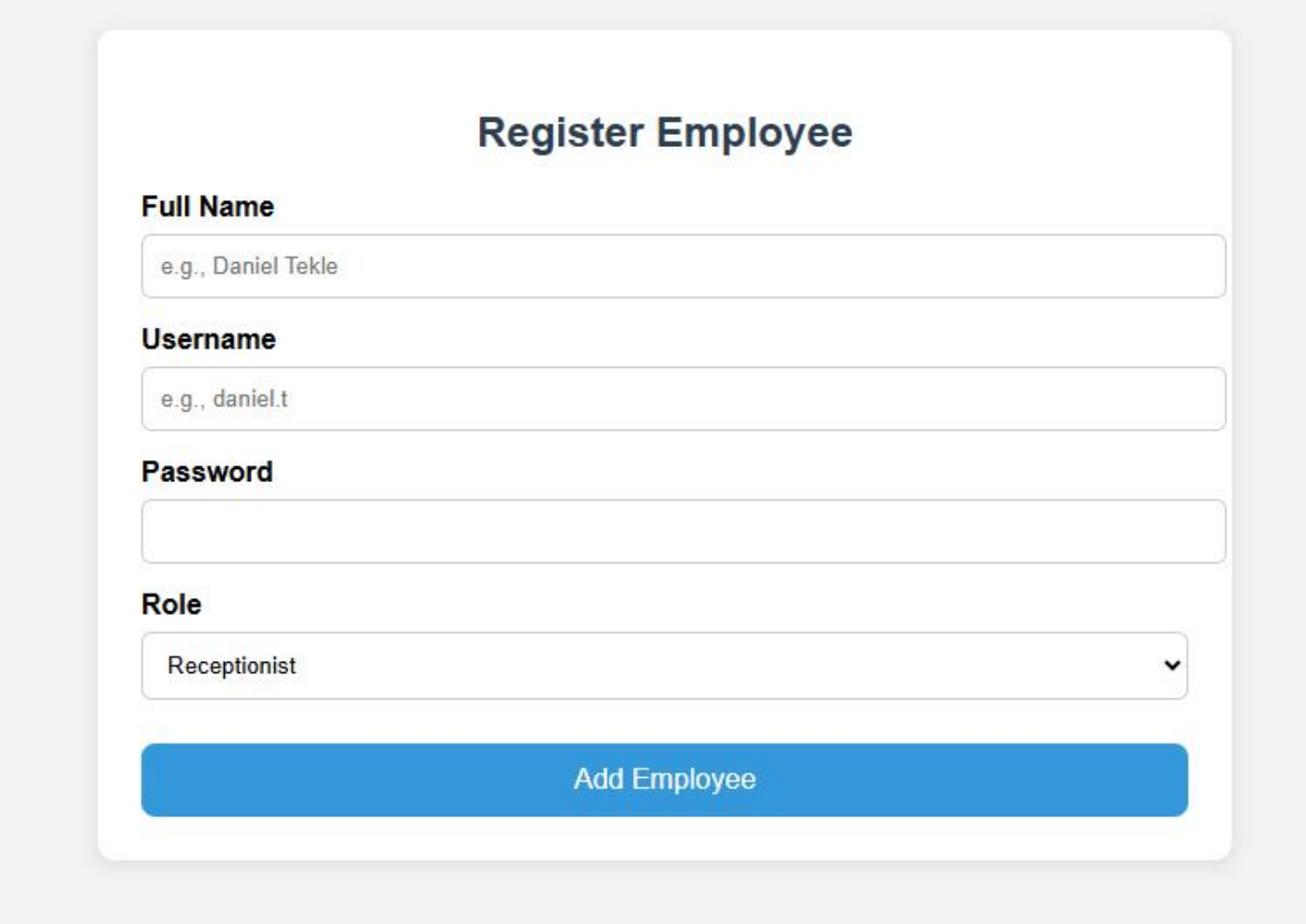


Figure 15 :Manager – Employee Registration Page

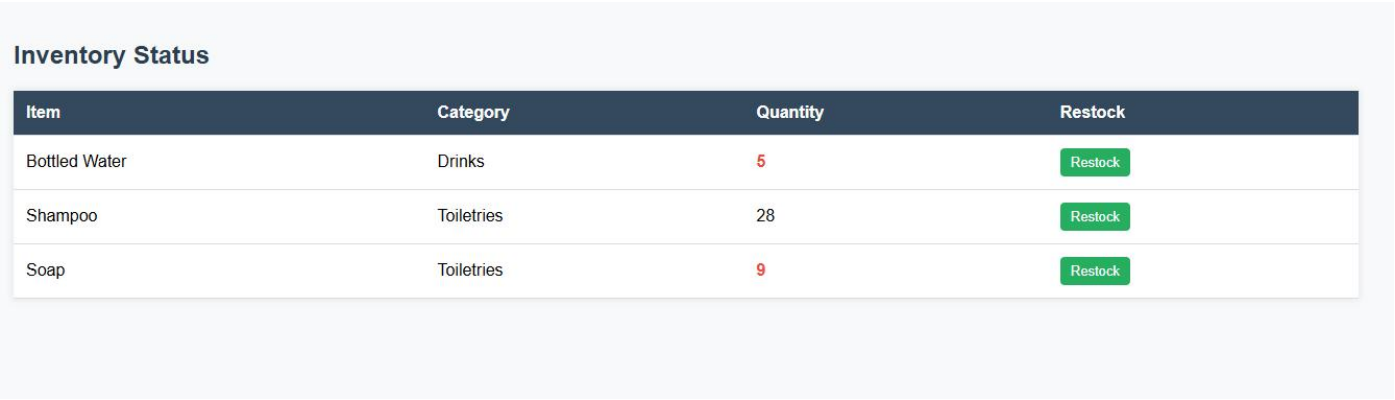


Figure 16 :Manager – Inventory Management Panel

Figure 17 : Manager – Restaurant Menu Editor

Chapter 4.3: Screen Image and Description

Image No.	Screen Title	Purpose	Description of Interface Components and Actions
8	Reserve a Room	To initiate a booking request.	Includes input fields for Check-in/Check-out dates, room type selection, and number of guests. A Book Now button finalizes the reservation request.
9	Reservation Management	To manage current guest reservations.	Shows a table listing Guest Name, room info, check-in/check-out dates, and status. An Actions column includes buttons like "Check-In" and "Cancel" for reservation handling.
10	Manager Dashboard	To provide managerial insight into key hotel operations.	Displays metrics in a grid of cards showing Today's Reservations, Occupied Rooms, and Total Sales, allowing quick monitoring of performance.

11	Room Cleaning Status	To assist housekeeping in identifying and updating rooms that need cleaning.	Lists rooms labeled "Needs Cleaning". Each entry features a "Mark as Cleaned" button for staff to update the cleaning status.
12	Invoice Payment	To facilitate guest invoice settlement.	Displays invoice number, guest name, room details, stay duration, and total amount due. Offers payment method options (Credit/Debit Card, Pay at Hotel, Bank Transfer) with fields for input.
13	Submit Feedback	To collect user opinions, complaints, or suggestions.	Titled "Share Your Feedback", the form includes a dropdown for feedback type, a text area for detailed input, and a Submit button to send the feedback to the system.
14	Sales Report	To review weekly financial performance.	Displays a Weekly Sales Report with columns for date, Room Sales, Restaurant Sales, and Total. A summary row shows the Total Revenue for the reporting period.
15	Add New Employee	To register new employees in the system.	A form labeled "Register Employee" collects full name, username, and password, along with a dropdown to assign roles (Receptionist, Housekeeping, Manager, or Admin).
16	Inventory Control	To manage and monitor stock levels.	A table displays item name, category, and available quantity. Items with low stock are highlighted. Each row includes a "Restock" button for initiating replenishment.
17	Menu Management	To update the restaurant's food and beverage offerings.	Table includes Dish Name, Category, and Price columns. Each row has an "Edit" button to modify item details.

Table 3 : Screen Image and Description

CHAPTER FIVE: REQUIREMENT MATRIX

This chapter maps each functional requirement identified in the SRS to its implementing system component/module. The requirement matrix ensures complete traceability and shows which part of the system satisfies each user need.

5.1 Requirement Traceability Matrix

#	Requirement ID	Requirement Description	Implemented in Component
1	FR01	User registration and login for guests	Authentication Module, Guest UI
2	FR02	Search room availability	Reservation Module, Room Service
3	FR03	Online room reservation	Reservation Controller, Guest UI
4	FR04	Admin can add/edit/delete room types	Room Management Module, Admin UI
5	FR05	View and manage bookings	Reservation Dashboard (Receptionist)
6	FR06	Room check-in and check-out	Reservation Module, Room Status Handler
7	FR07	Invoice generation upon checkout	Invoice Generator, Billing Module
8	FR08	Online payment support	Payment Gateway Integration
9	FR09	Guests can view payment history	Guest Dashboard
10	FR10	Staff account management	Employee Manager (Manager Role)
11	FR11	Inventory management and stock updates	Inventory Controller
12	FR12	Restaurant menu update	Menu Editor
13	FR13	Food ordering and order tracking	Order Processor, Guest Dining UI
14	FR14	Feedback submission and complaints	Feedback Module
15	FR15	Manager dashboard with reports and analytics	Reporting Module
16	FR16	Housekeeping staff updates cleaning status	Housekeeping Dashboard
17	FR17	View list of all employees	Employee Manager
18	FR18	Generate financial reports	Report Generator Module
19	FR19	Role-based access control	Security Module
20	FR20	View and manage food order logs	Restaurant Order Dashboard

Table 4 : Requirement Traceability Matrix

REFERENCES

- Sommerville, Ian. *Software Engineering* (10th ed.). Pearson, 2015.
- Pressman, Roger S. *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill, 2014.
- ISO/IEC/IEEE 42010:2011 - Systems and software engineering — Architecture description.
- Online UML Diagram Tools – draw.io, mermaid.js
- MySQL Documentation – https://dev.mysql.com/doc
- HTML & CSS Standards – W3C
- Payment Gateway API Docs (e.g., Stripe, PayPal)
- SRS Document – Web-Based Hotel Management System, 2025 Edition
- Senior Project Documentation Template – Haramaya University, College of Computing and Informatics and Informatics

APPENDIX (optional)

APPENDIX A: ENTITY-RELATIONSHIP DIAGRAM (ERD)

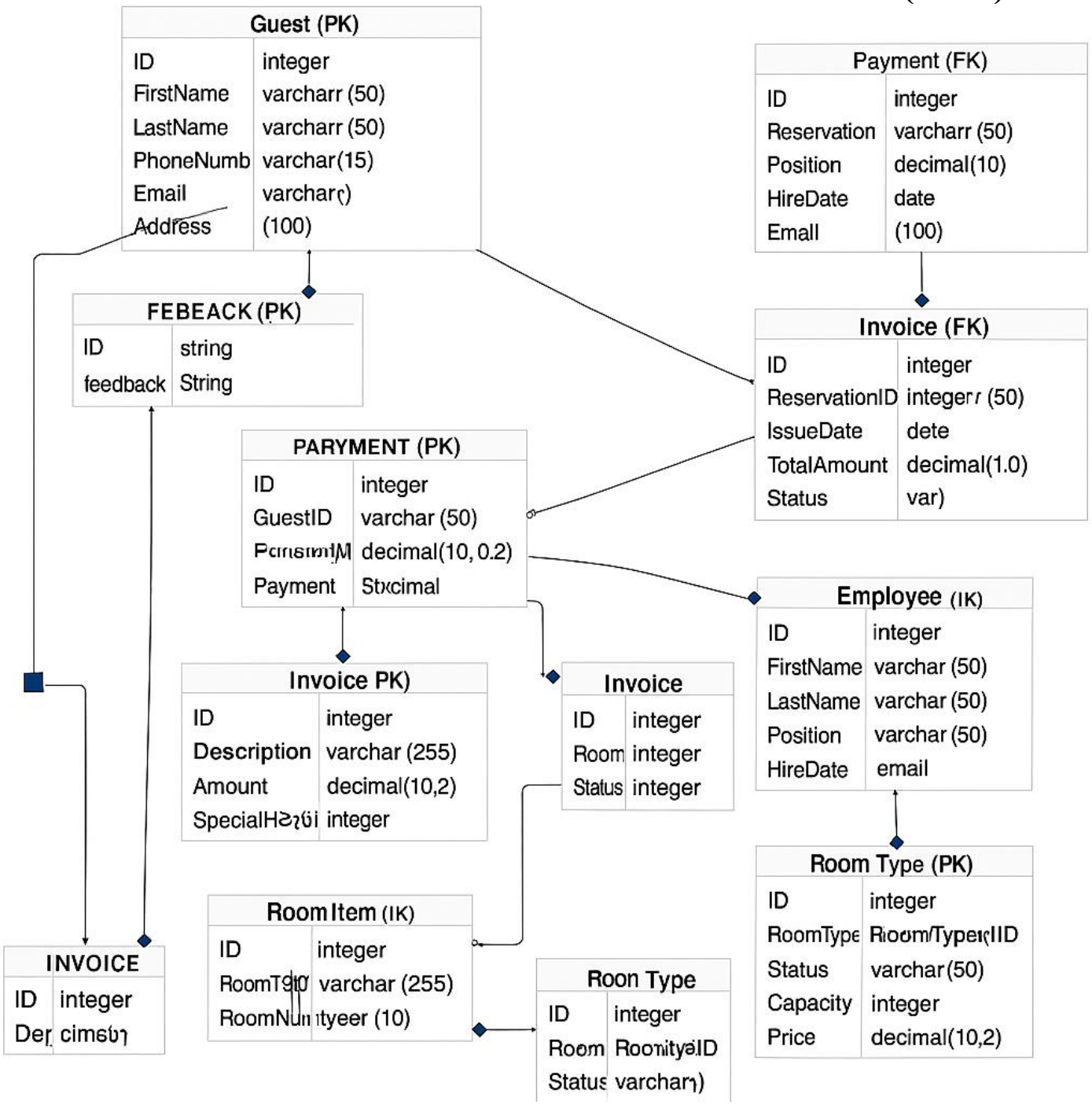


Figure 18 : Er-diagram

APPENDIX B: DATA DICTIONARY

Table	Column Name	Data Type	Constraints	Description
Guest	GuestID	INT	PK, Auto-Increment	Unique guest identifier
Guest	FullName	VARCHAR(100)	NOT NULL	Guest's full name
Guest	Email	VARCHAR(100)	UNIQUE, NOT NULL	Guest email
Guest	Phone	VARCHAR(20)	NOT NULL	Contact number
Guest	MembershipLevel	ENUM	DEFAULT 'Regular'	Guest type: Regular/VIP
Reservation	ReservationID	INT	PK, Auto-Increment	Unique reservation identifier
Reservation	GuestID	INT	FK → Guest	Refers to the booking guest
Reservation	RoomID	INT	FK → Room	Refers to reserved room
Reservation	CheckInDate	DATE	NOT NULL	Date of check-in
Reservation	CheckOutDate	DATE	NOT NULL	Date of check-out
Reservation	Status	ENUM	DEFAULT 'Booked'	Reservation status
Reservation	TotalAmount	DECIMAL(10,2)	NOT NULL	Total cost of reservation

Table 5: DATA DICTIONARY

APPENDIX C: API ENDPOINTS DOCUMENTATION

Method	Endpoint	Description	Required Roles
POST	/api/register	Register a new guest	Guest (Public)
POST	/api/login	Authenticate guest or employee	All Roles
GET	/api/rooms/available	List available rooms	Guest, Reception
POST	/api/reservations	Create reservation	Guest
GET	/api/reservations/{id}	View reservation details	Guest, Reception
PUT	/api/rooms/{id}/status	Update room cleaning or occupancy	Housekeeping, Mgr
POST	/api/payments	Submit payment	Guest, Reception
GET	/api/reports/daily	Retrieve daily financial report	Manager
POST	/api/feedback	Submit feedback or complaint	Guest

Table 6 : API ENDPOINTS DOCUMENTATION

APPENDIX D: UI STYLE GUIDE

Component	Standard Style
Font Family	"Segoe UI", Arial, sans-serif
Primary Color	#2c3e50 (headers, buttons)
Accent Color	#27ae60 (success), #e74c3c (error), #3498db (info buttons)
Border Radius	8px–12px for buttons, forms, cards
Button Style	Solid fill, hover darkens, shadow on focus
Input Fields	Light background, rounded edges, large padding
Spacing	Minimum 20px padding on sides and 15px between elements
Icons	Lucide or Material Icons (used in nav and alerts)

Table 7 : UI STYLE GUIDE