Project Report

Online Hotel Reservation

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Understanding/Analysis	7					
Implementation	8					
Report Writing	10					
					Total obtained mark	
Comments						

Semester: Spring 2025 **Group No:- (07)**

Name: Md. Zahidul Islam **ID: 221-35-1082**

Name: Oishi Salowa **ID: 221-35-1032**

Name: Khadiza Islam Punno **ID: 221-35-960**

Batch: 37th **Section: E1** **Course Code: SE331**

Course Name: Software Engineering Design Capstone Project

Course Teacher Name: Mr. Foysal Khandakar Joy **Designation: Lecturer**

Submission Date: 17/04/2025

Software Engineering Design Capstone Project

SE 331 E1

1. **Project Name:** Online Hotel Reservation System

2. **Stack (FrontEnd + BackEnd + DataBase):** HTML-CSS-JS-PHP-MySQL-BOOTSTRAP

3. **Roles of TeamMates:**

Name	Role
Md. Zahidul Islam (221-35-1082)	Backend & Database (Server Side)
Oishi Salowa (221-35-1032)	FrontEnd (Client Side)
Khadiza Islam Punno (221-35-960)	Project Management & System Architecture

Table Of Contents

1. Introduction

1.1 Project Overview

The hotel reservation system is a comprehensive web-based application designed to automate and streamline the process of booking rooms in a hotel. In the traditional method of hotel reservation, a customer has to either call the hotel, physically visit, or send emails to reserve a room. These manual systems are often inefficient, error-prone, and time-consuming for both customers and hotel staff. The purpose of this project is to develop a digital solution that simplifies the process for both hotel management and guests.

This project facilitates customers to easily browse available rooms, check room types and availability, and make reservations through an intuitive web interface. The system maintains a structured database to manage customer details, room bookings, check-in/check-out records, and transaction histories. The system also incorporates a secured admin panel that allows administrators to manage rooms, handle reservations, confirm bookings, process check-ins/check-outs, and view financial transactions.

Overall, the hotel reservation system aims to minimize human errors, increase efficiency, and provide a better experience to the customers while empowering hotel staff with a digital dashboard to manage operations.

1.2 Problem Statement

In many small to medium-sized hotels, the reservation process is still handled manually or via very basic systems. This results in several problems such as:

- **Data redundancy and inaccuracy** due to manual entry.
- **Lack of real-time room availability updates**, which can lead to overbooking or customer dissatisfaction.
- **Limited accessibility** to customers, as reservations often rely on phone calls or physical visits.
- **Inefficiencies in managing check-ins/check-outs**, reservations, and transactions.
- **Difficulty in maintaining customer records** and historical booking data for future use.

Without a proper reservation system, hotel staff are burdened with repetitive administrative tasks, and customers face unnecessary delays and confusion. A digital system is needed to eliminate these challenges and bring modern convenience to both users and administrators.

1.3 Objectives of the System

The primary objectives of the hotel reservation system are as follows:

- To develop an easy-to-use, web-based application for online hotel room reservations.
- To enable hotel customers to view room availability, submit reservation requests, and receive confirmations.
- To allow hotel staff to manage bookings, room availability, customer details, and transaction records effectively.
- To create a secured admin panel for staff with different levels of access (Admin, Staff).
- To automate the check-in and check-out process to improve operational efficiency.
- To generate transaction logs for completed reservations and maintain accurate financial records.
- To improve user satisfaction by providing a smooth and responsive web interface.

1.4 Scope of the Project

The scope of this project covers both the **front-end user interface** and the **back-end admin panel** of the hotel reservation system. The core components include:

- **Customer-facing module**: Includes homepage, about/contact pages, room search, booking forms, confirmation page, and reservation query handling.
- **Admin panel**: Allows hotel management to log in securely and perform tasks like adding/editing/deleting rooms, managing user accounts, confirming reservations, and processing check-in/check-out.
- **Database system**: MySQL-based database to store all data including rooms, customers, transactions, and user accounts.
- **Security features**: Includes authentication, session handling, and SQL injection prevention for data security.
- **Local deployment**: The system is developed and tested using XAMPP server on a local environment but is fully capable of being hosted on a live server.

This system is designed to support a single hotel with multiple rooms, and can be extended in the future to support multi-branch hotel chains, mobile app versions, and integration with online payment gateways.

1.5 Methodology

The development of the hotel reservation system follows the **Waterfall model**, which consists of the following stages:

1. **Requirement Analysis**: Identifying the features, functions, and constraints of the current system and the goals of the new system.
2. **System Design**: Planning the architecture, interface design, database schema, and security measures.
3. **Implementation**: Developing the system using PHP for backend, MySQL for database, HTML/CSS for design, and JavaScript for client-side interactions.

4. **Testing**: Performing unit testing, integration testing, and user acceptance testing to ensure the system is free of bugs and meets requirements.
5. **Deployment**: Deploying the system locally using XAMPP and preparing it for hosting.
6. **Documentation**: Preparing comprehensive documentation for system users, developers, and future maintenance.

This methodology ensures that each stage is completed before moving to the next, allowing thorough planning and a structured development cycle.

1.6 Tools and Technologies Used

The hotel reservation system is developed using modern and widely-used technologies that ensure reliability, security, and maintainability. These include:

- **Programming Languages & Markup**:
  - **PHP** – Server-side scripting language used to create dynamic pages and handle server logic.
  - **HTML** – Markup language for structuring the web pages.
  - **CSS** – Used for styling and layout design of the front-end interface.
  - **JavaScript** – For client-side interactivity and dynamic form validation.
  - **jQuery** – A fast, small JavaScript library to simplify DOM manipulation and AJAX operations.
- **Frameworks & Libraries**:
  - **Bootstrap** – Front-end framework for responsive design and UI components.
  - **DataTables (jQuery Plugin)** – Used in the admin panel to present data in tabular format with features like search, sort, and pagination.
- **Database**:
  - **MySQL** – Used as the relational database management system (RDBMS) to store all backend data securely.
- **Development Environment**:
  - **XAMPP** – A free and open-source cross-platform web server package that includes Apache, MySQL, PHP, and Perl. It is used for local development and testing.

2. System Analysis

2.1 Existing System and its Limitations

In many traditional hotels, the reservation process is still managed manually or using basic spreadsheets and phone records. A customer typically needs to call or visit the hotel to inquire

about room availability, prices, or make a reservation. Staff members then record this information manually, which introduces several inefficiencies and risks.

Key limitations of the existing manual system:

- **Time-consuming reservation process**: Both the customer and the hotel staff must spend time on calls or in-person communication to book rooms, which leads to delays.
- **Lack of real-time availability**: Rooms may appear available during customer inquiries, but due to manual tracking, there's a risk of double bookings or inaccurate information.
- **Data inconsistency**: Since entries are done manually, there's a higher risk of human error, such as incorrect customer details, wrong dates, or mismatched reservations.
- **Difficulty in tracking history**: There is no systematic way to track past customer bookings, payment history, or stay records.
- **No integration**: The absence of an integrated system leads to a disjointed flow of operations—booking, check-in, check-out, and payments are all tracked separately.
- **Limited scalability**: As the business grows, managing reservations manually becomes impractical and inefficient.

This calls for a modern, automated solution that can address all the above pain points with a centralized, digital approach.

2.2 Proposed System Overview

The proposed hotel reservation system is a complete, web-based application that aims to eliminate the drawbacks of the existing manual system. It digitizes the entire reservation lifecycle—from customer search and booking to admin confirmation and financial tracking—within a single, easy-to-use platform.

Key features of the proposed system:

- **Customer Module**: Allows users to search available rooms, view detailed room information, and make online reservations through a clean and responsive interface.
- **Admin Panel**: Secure login-based backend for hotel staff to manage bookings, check-ins, check-outs, and room inventory.
- **Real-time availability tracking**: The system automatically updates room availability once a booking is confirmed or a check-in/check-out is processed.
- **Database-driven architecture**: All customer, room, and transaction data are stored in a MySQL database to ensure consistency, reliability, and easy retrieval.
- **Improved customer experience**: Customers can book rooms from anywhere, anytime, without needing to call or visit the hotel.
- **Efficient operations**: Hotel staff can perform all reservation-related tasks through a centralized dashboard, reducing manual work and errors.

This new system not only streamlines hotel operations but also enhances customer satisfaction by making the booking process faster, easier, and more reliable.

2.3 Functional Requirements

Functional requirements describe the specific behaviors and features that the system must have. These include user interactions, data handling, and system responses.

Customer Side:

- View homepage, about, and contact pages.
- Search for available rooms by date and type.
- Submit reservation requests via reservation form.
- Receive booking confirmation on-screen and through reply handling.

Admin Side:

- Secure login/logout functionality.
- Dashboard displaying total rooms, bookings, and transactions.
- Add, edit, and delete room details.
- Manage user accounts (add/edit/delete).
- View and confirm pending reservations.
- Process check-in and check-out of guests.
- View completed transactions and reservation history.

System-Level Requirements:

- Automatically update room availability upon booking.
- Store and retrieve customer details and reservation records.
- Validate all inputs to ensure clean and accurate data.
- Maintain session-based login for secure admin operations.

2.4 Non-Functional Requirements

These define the overall qualities and constraints of the system. They do not describe specific behaviors but are crucial for performance and reliability.

- **Performance**: The system should load pages within 2 seconds and handle concurrent user requests without delay.
- **Usability**: The interface should be user-friendly, intuitive, and responsive across devices.
- **Security**: User data and admin access must be protected using authentication, session control, and secure database interactions.
- **Scalability**: The system should be capable of expanding to support more rooms, users, or branches in the future.

- **Maintainability**: Code should be modular and well-documented to support future updates.
- **Reliability**: The system must remain stable during normal operations with minimal downtime.
- **Portability**: The system should work across all modern browsers and be easily hosted on standard web servers.

2.5 User Roles and Permissions

To ensure the right level of access and security, the system defines multiple user roles. Each role has a specific set of permissions tailored to its responsibilities.

General User

Role: Customer/Guest
Access and Permissions:

- View available rooms and hotel information.
- Search and filter room listings based on criteria (e.g., room type, date).
- Submit a reservation request via the form.
- Receive confirmation after successful reservation.
- View confirmation details of reservation.

Limitations:

- Cannot view other users' data.
- Cannot modify room details or database entries.
- Cannot access admin panel.

Admin

Role: Hotel staff or management personnel
Access and Permissions:

- Log in to the admin dashboard securely.
- Add new rooms to the inventory with relevant details.
- Edit or delete existing room information.
- Add and manage staff accounts (admin users).
- Confirm or reject pending reservations.
- Process guest check-ins and check-outs.
- Access full transaction records and financial logs.
- Utilize utility scripts for validation and database interactions.

Security Considerations:

- All admin operations are password-protected and session-based.

- Admin access is restricted to verified users only.
- Sensitive actions (like deleting data) are guarded with confirmation prompts.

3. System Design

System design is the blueprint that defines the structure, components, interfaces, and data of a system to satisfy specified requirements. For our hotel reservation system, the design phase involves choosing the right architecture, visualizing data flow, planning database relationships, and modeling user interactions.

3.1 System Architecture

The architecture of the hotel reservation system follows a **three-tier architecture**:

1. Presentation Layer (Front-end)

- Responsible for displaying the UI elements to users.
- Includes HTML, CSS, Bootstrap for layout and styling.
- JavaScript and jQuery handle client-side interactivity.
- Accessible through a web browser on desktop or mobile.

2. Application Layer (Business Logic)

- Built using PHP to handle all business logic.
- Processes data sent from the front end.
- Communicates with the database layer.
- Includes all server-side scripts for room search, reservation, authentication, admin panel functions, etc.

3. Data Layer (Database)

- MySQL database (db_hor) stores all application data.
- Includes tables for users, rooms, reservations, check-in/out records, and transaction logs.
- PHP uses mysqli to connect and interact securely with this data.

Flow Summary:
The user interacts with the UI → PHP scripts process input and perform actions → Data is stored/retrieved from MySQL → Output is sent back to the user.

3.2 ER Diagram and Database Schema

The **Entity-Relationship Diagram (ERD)** models the core database structure. Below is a textual description (and you can create an actual diagram from this structure in a tool like MySQL Workbench or dbdiagram.io):

Entities and Their Relationships:

1. **Users**
    - user_id (PK)
    - name
    - email
    - password
    - role (admin/user)
2. **Rooms**
    - room_id (PK)
    - room_number
    - room_type
    - price_per_night
    - status (available/booked)
    - features
    - image_path
3. **Reservations**
    - reservation_id (PK)
    - user_id (FK)
    - room_id (FK)
    - check_in_date
    - check_out_date
    - status (pending/confirmed)
4. **CheckInOut**
    - record_id (PK)
    - reservation_id (FK)
    - checkin_time
    - checkout_time
5. **Transactions**
    - transaction_id (PK)
    - reservation_id (FK)
    - amount
    - date
    - payment_method

Relationships:

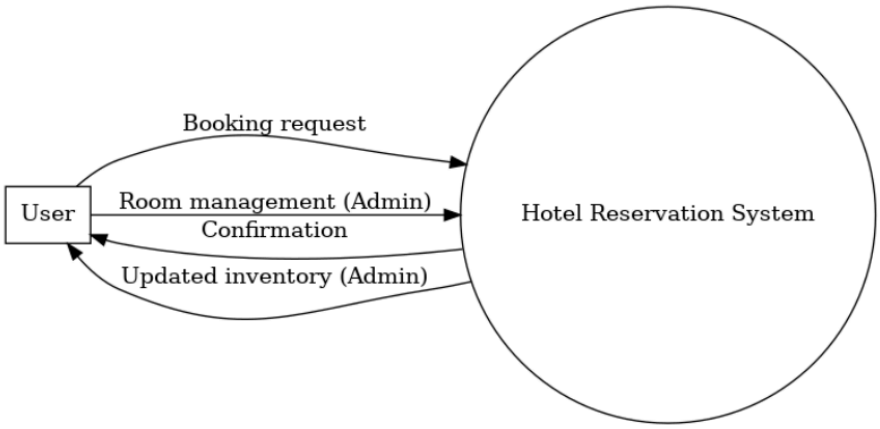
- A User can make multiple Reservations.
- A Room can be linked to many Reservations.
- A Reservation can result in one CheckInOut record.
- A Reservation can lead to one Transaction.

3.3 Data Flow Diagram (DFD)

A **Data Flow Diagram** shows how data moves within the system. Here's a level-wise breakdown:

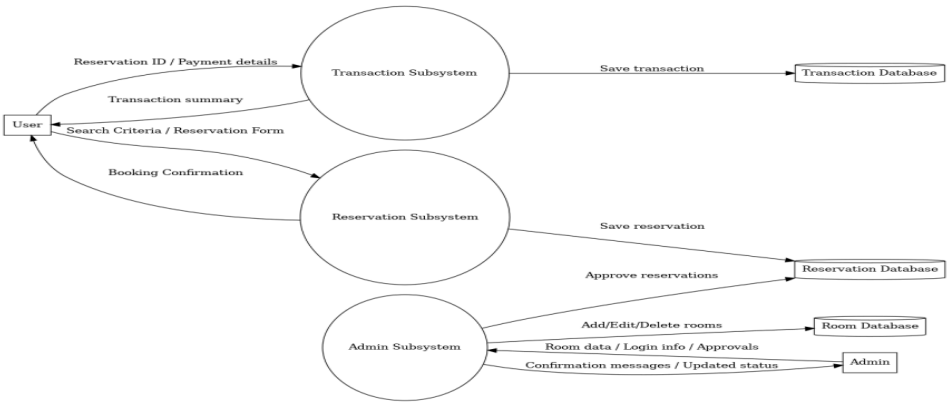
Level 0 (Context Diagram)

- **External Entity**: User
- **System**: Hotel Reservation System
- **Data Flows**:
    - User sends booking request → System stores reservation → System returns confirmation
    - Admin manages rooms → System updates inventory



Level 1 DFD

1. **Reservation Subsystem**
    - Inputs: Search Criteria, Reservation Form
    - Processes: Check availability, Save reservation
    - Output: Booking Confirmation
2. **Admin Subsystem**
    - Inputs: Room data, Login info, Reservation approvals
    - Processes: Add/edit/delete rooms, Approve reservations, Process check-ins/outs
    - Outputs: Updated room status, Confirmation messages
3. **Transaction Subsystem**
    - Inputs: Reservation ID, Payment details
    - Processes: Calculate amount, Save transaction
    - Outputs: Transaction summary



3.4 Use Case Diagrams

Use Case Diagrams provide a high-level visualization of user interactions with the system.

Actors:

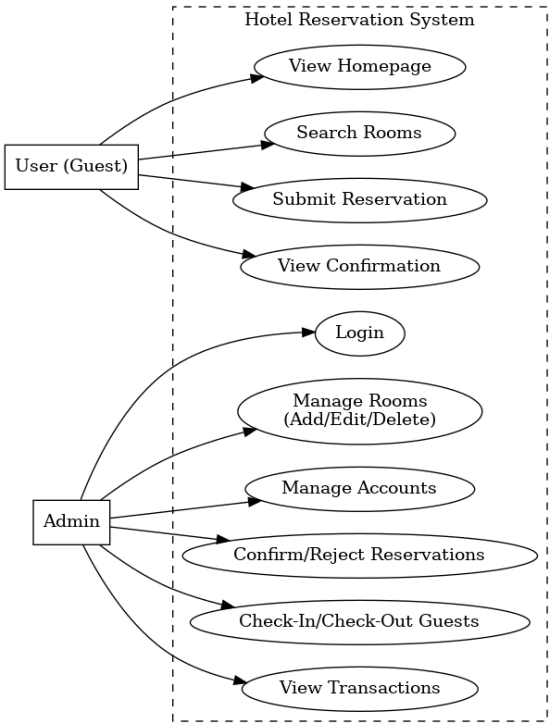
- **User (Guest)**
- **Admin**

User Use Cases:

- View Homepage
- Search Rooms
- Submit Reservation
- View Confirmation

Admin Use Cases:

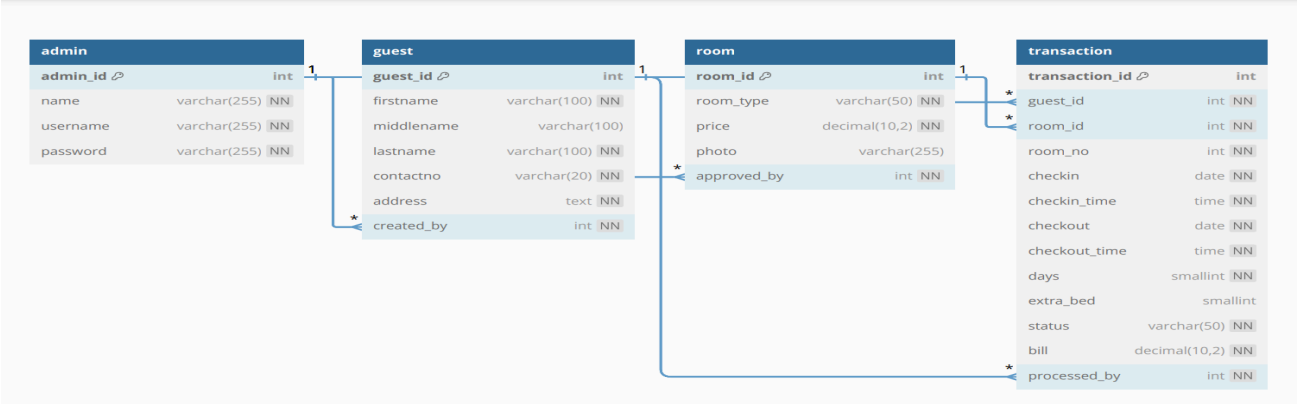
- Login
- Manage Rooms (Add/Edit/Delete)
- Manage Accounts
- Confirm/Reject Reservations
- Check-In/Check-Out Guests
- View Transactions



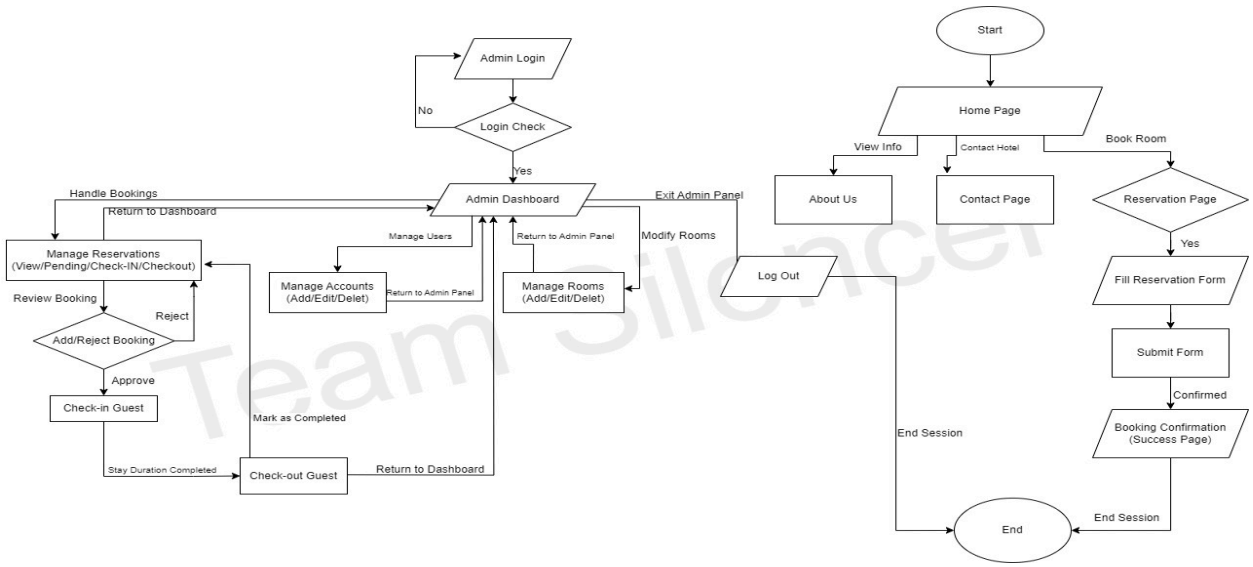
Use Case Example:
Title: *Submit Reservation*
Actor: User
Pre-condition: User selects check-in/check-out dates
Flow:

1. User fills out the reservation form.
2. System checks availability.
3. Reservation is saved as pending.
4. Admin later confirms or rejects the reservation.

3.5 Database Flow Diagrams



3.6 Flow Chart Diagrams



4. Database Design

The database is the backbone of any web-based application. For the hotel reservation system, a well-structured and optimized **MySQL database** ensures consistent data storage, fast queries, and secure access to user, room, and transaction data. This section describes the core schema, tables, their relationships, and key SQL queries used throughout the system.

4.1 MySQL Database Overview (db_hor.sql)

The database used in this system is named **db_hor**, short for "Database - Hotel Reservation". It contains all the necessary tables to support both customer and admin functionalities, and ensures relational consistency through foreign keys.

The schema design supports the following functionalities:

- Storing user login and profile data.
- Room inventory management.
- Reservation handling and status tracking.
- Check-in and check-out process recording.
- Transaction history logging for completed stays.

All tables are normalized and designed using **InnoDB** engine to ensure ACID compliance and foreign key constraints.

4.2 Tables and Relationships

Here's a breakdown of the main tables in db_hor:

1. users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**guest_id**	int(11)			No	*None*		AUTO_INCREMENT
2	**firstname**	varchar(50)	latin1_swedish_ci		No	*None*		
3	**middlename**	varchar(30)	latin1_swedish_ci		No	*None*		
4	**lastname**	varchar(40)	latin1_swedish_ci		No	*None*		
5	**address**	varchar(50)	latin1_swedish_ci		No	*None*		
6	**contactno**	varchar(13)	latin1_swedish_ci		No	*None*		

2. rooms

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**room_id**	int(11)			No	*None*		AUTO_INCREMENT
2	**room_name**	varchar(255)	latin1_swedish_ci		No	*None*		
3	**room_type**	varchar(50)	latin1_swedish_ci		No	*None*		
4	**price**	varchar(11)	latin1_swedish_ci		No	*None*		
5	**photo**	varchar(100)	latin1_swedish_ci		No	*None*		

3. admin

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	admin_id	int(11)			No	None		AUTO_INCREMENT
2	name	varchar(30)	latin1_swedish_ci		No	None		
3	username	varchar(24)	latin1_swedish_ci		No	None		
4	password	varchar(24)	latin1_swedish_ci		No	None		

4. transactions

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	admin_id	int(11)			No	None		AUTO_INCREMENT
2	name	varchar(30)	latin1_swedish_ci		No	None		
3	username	varchar(24)	latin1_swedish_ci		No	None		
4	password	varchar(24)	latin1_swedish_ci		No	None		

Relationships Summary

- One **user** → many **reservations**
- One **room** → many **reservations**
- One **reservation** → one **transaction**
- One **reservation** → one **check-in/out** record

4.3 SQL Queries and Optimization

Common SQL Queries

1. Fetch Available Rooms:

```
SELECT * FROM rooms
WHERE status = 'available'
AND room_id NOT IN (
 SELECT room_id FROM reservations
 WHERE check_in_date <= CURDATE()
 AND check_out_date >= CURDATE()
 AND status = 'confirmed'
);
```

2. Insert a Reservation:

```
INSERT INTO reservations (user_id, room_id, check_in_date, check_out_date, status)
VALUES (?, ?, ?, ?, 'pending');
```

3. Confirm a Reservation (Admin):

```
UPDATE reservations
SET status = 'confirmed'
WHERE reservation_id = ?;
```

4. Add Transaction:

```
INSERT INTO transactions (reservation_id, amount, payment_method, date)
VALUES (?, ?, ?, NOW());
```

5. Record Check-in/Check-out:

```
INSERT INTO checkin_checkout (reservation_id, checkin_time)
VALUES (?, NOW());
-- later update for checkout
UPDATE checkin_checkout
SET checkout_time = NOW()
WHERE reservation_id = ?;
```

Optimization Techniques

- **Indexes**: Applied on foreign keys (user_id, room_id) and frequently searched fields (status, email) to speed up lookups.
- **Prepared Statements**: All SQL interactions use parameterized queries in PHP (mysqli_prepare) to prevent SQL injection.
- **Efficient Joins**: Complex reports (e.g., total revenue per room) are written using JOIN operations between transactions, reservations, and rooms.
- **Data Integrity**: All foreign key constraints are enforced using ON DELETE CASCADE or ON DELETE RESTRICT depending on logical flow.

5. User Side Module

The **User Side Module** is designed for the guest users who interact with the system to view available rooms, make reservations, and manage their bookings. This module is built with a focus on ease of use and seamless interaction between users and the hotel system. All user-facing

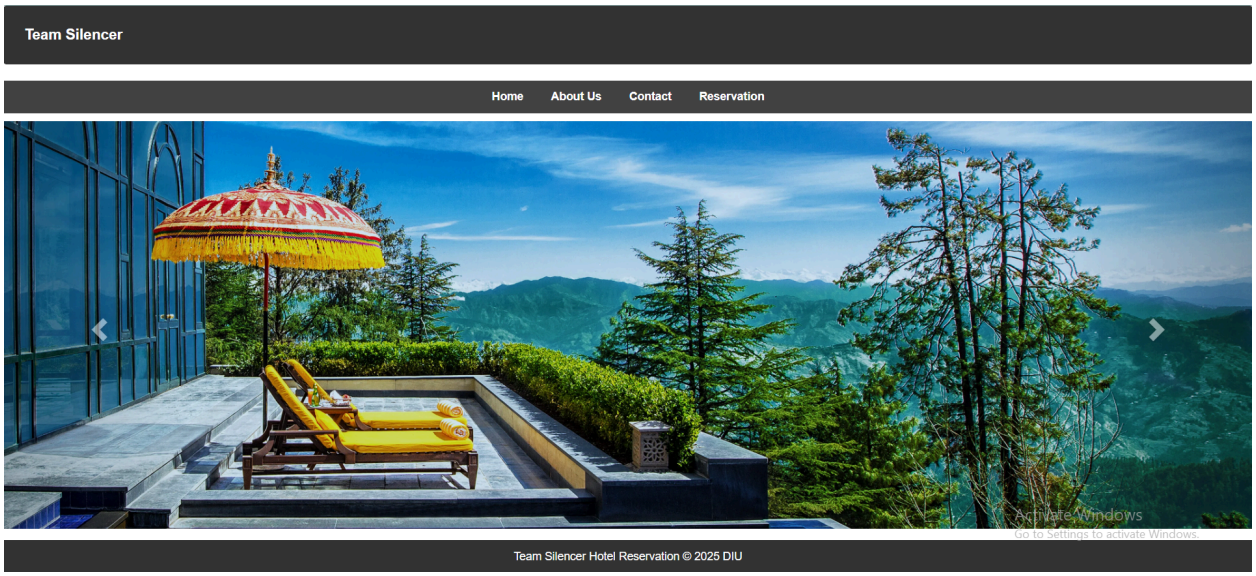
components are hosted on the web front-end, using HTML, CSS, Bootstrap for styling, and JavaScript/jQuery for interactivity.

5.1 Homepage (index.php)

The homepage is the first page users encounter when visiting the website. It is designed to provide an overview of the hotel, showcase the available rooms, and offer essential navigation options for room booking.

Key Features:

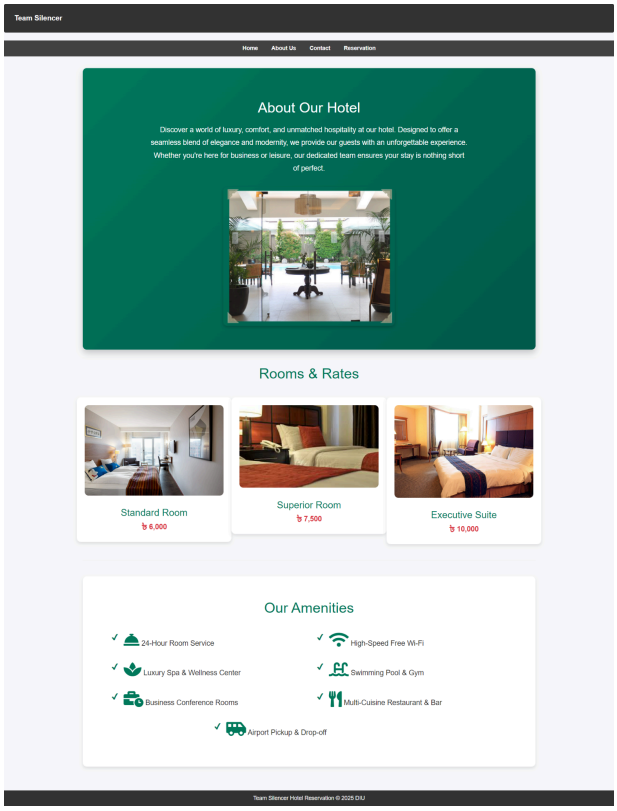
- **Navigation Bar**: Includes links to Home, About Us, Contact Us, and Room Search.
- **Hotel Overview**: Displays a brief introduction to the hotel, its amenities, and any special promotions.
- **Room Categories**: Display a list of room types (e.g., Deluxe, Suite) with attractive images and prices.
- **Room Search**: A search form that allows users to specify their check-in/check-out dates and the number of guests.
- **Footer**: Contains contact information, links to privacy policy, and social media.

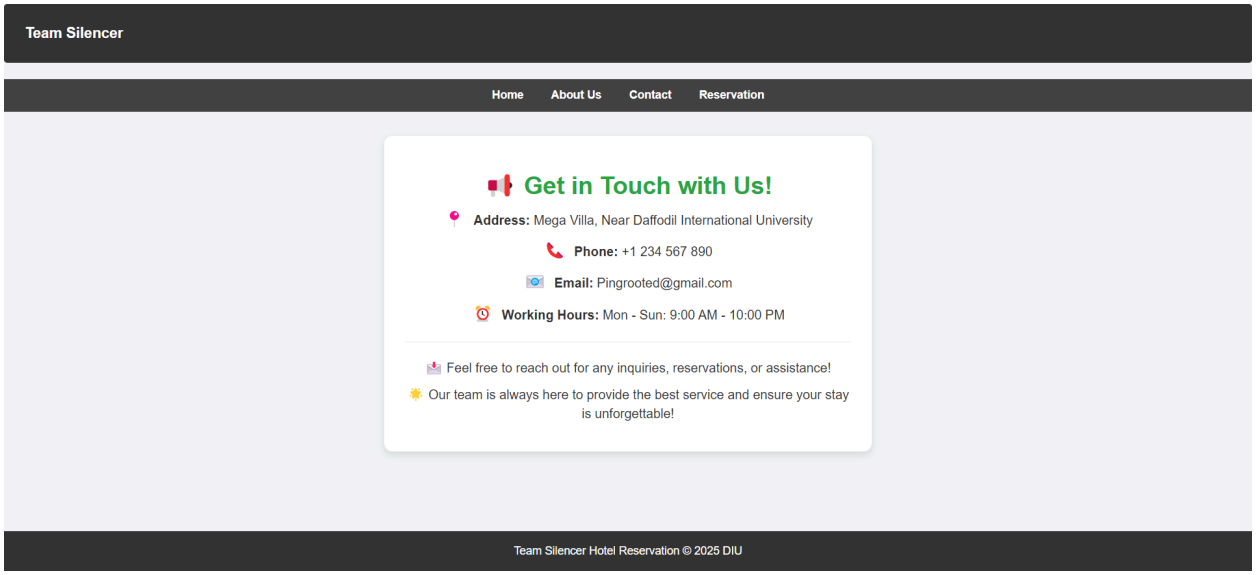


5.2 About and Contact Pages (aboutus.php, contactus.php)

These pages provide additional details about the hotel and allow users to contact the hotel for further inquiries.

- **About Us (aboutus.php)**: Describes the hotel's history, mission, facilities, and customer service ethos.
- **Contact Us (contactus.php)**: Provides a contact form where users can submit inquiries, along with a hotel address, phone number, and email.



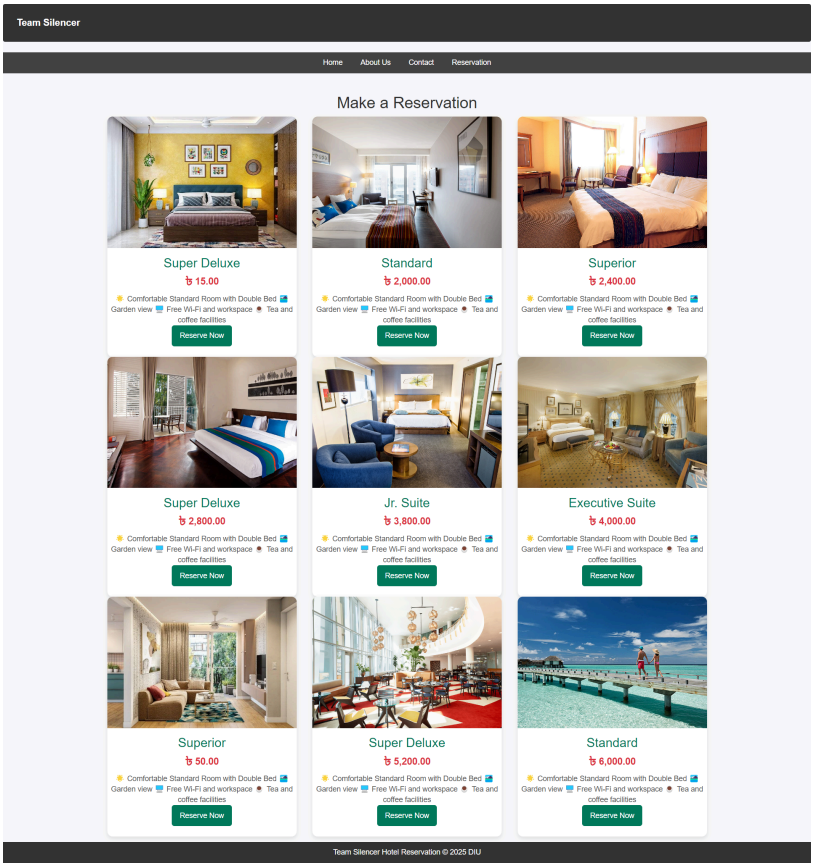


5.3 Room Search and View

The room search and view functionality is the core of the user experience. Users can search for available rooms based on check-in/check-out dates and the number of guests, and view detailed information about each room.

Key Features:

- **Search**: Filters available rooms based on dates and guest count.
- **Room Details**: Displays detailed room information (price, description, amenities, availability).
- **Room Booking**: Direct link to the reservation page for booking the room.

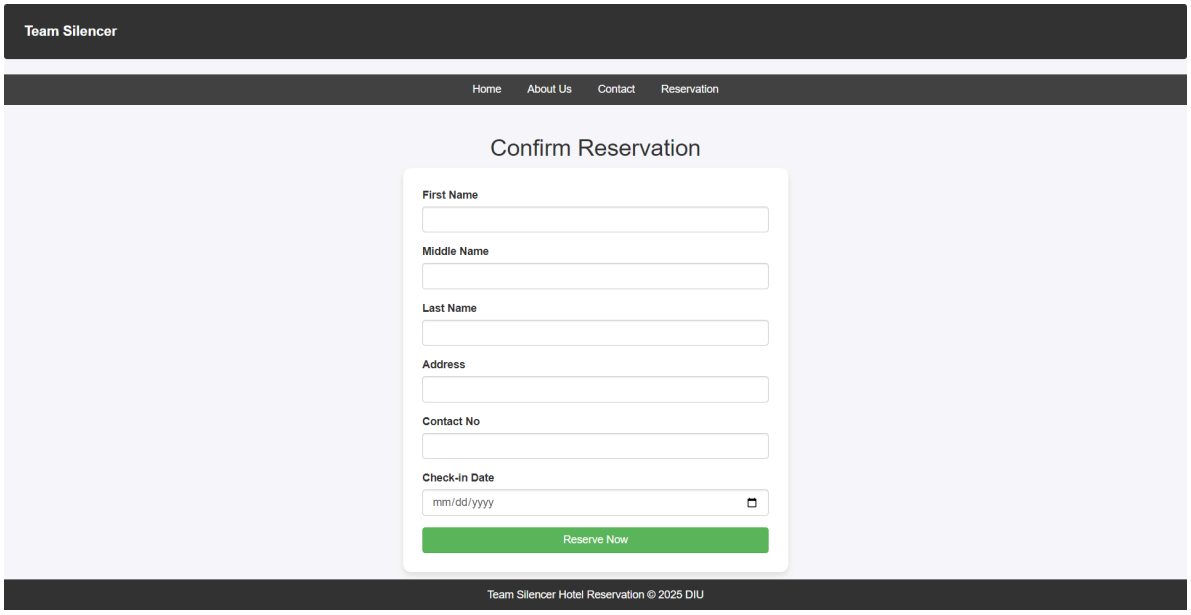


5.4 Reservation Form (reservation.php, add_reserve.php)

The reservation form allows users to book a room by providing essential details such as their personal information, check-in/check-out dates, and payment preferences.

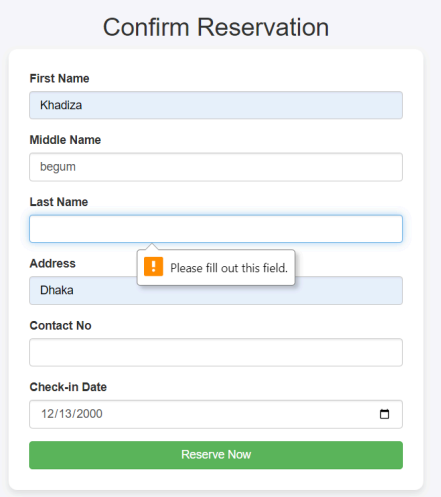
Key Features:

- **Room Selection**: Users choose a room based on availability.
- **Guest Information**: Collects name, email, phone number, and payment details.
- **Reservation Details**: Displays total price, room type, and booking dates.
- **Confirmation**: After submission, users are shown a confirmation page with reservation details.



5.5 Reservation Query Submission (add_query_reserve.php)

This page allows users to submit queries related to their reservations before finalizing the booking. This functionality can handle queries about room availability, special requests, or modifications to an existing reservation.

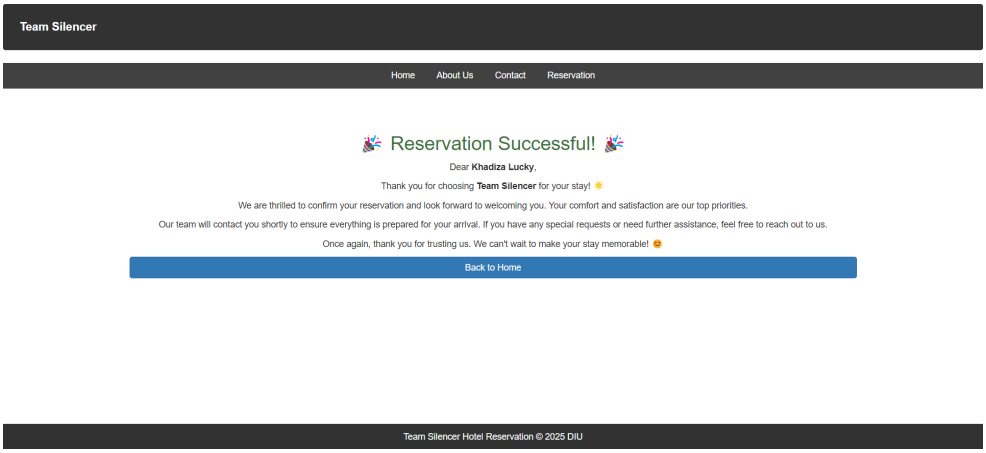


5.6 Reservation Confirmation (confirmation.php)

Once a reservation is successfully submitted, users are redirected to the **Reservation Confirmation** page. This page provides the user with the summary of their booking, including the room details, dates, price, and a unique booking ID.

Confirmation Page Features:

- Displays booking summary.
- Reservation ID for tracking.
- Payment instructions (if needed).



5.7 Reply Handling (reply_reserve.php)

This functionality is for users to receive updates or replies regarding their reservation status or queries. After a reservation is confirmed or any special requests are handled, the system can notify users through this page.

Reply Handling Process:

- Admin replies to queries submitted by users.
- Users can view admin responses to their reservations or inquiries.

6. Admin Panel Module

The **Admin Panel Module** provides the necessary tools for hotel administrators to manage the operations of the system, from handling reservations and rooms to overseeing user accounts and financial transactions. This module is protected by secure authentication and includes a user-friendly interface for smooth day-to-day management.

6.1 Admin Login & Logout (login.php, logout.php)

The **Login** page allows administrators to securely access the admin panel. It uses user authentication (stored in the users table) to verify that the logged-in user has an admin role.

Features:

- **Secure Login**: Admin must provide a valid email and password.

- **Session Management**: Upon successful login, an admin session is created to track the logged-in user.

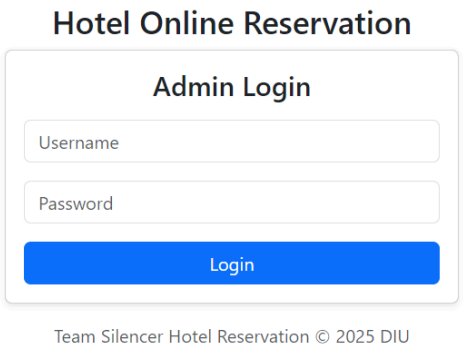
Logout Process:

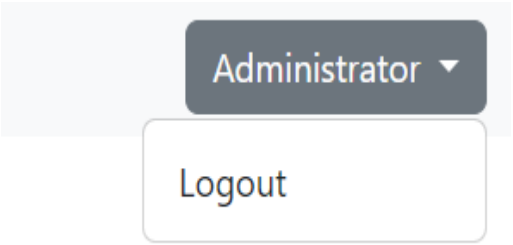
- On **logout**, the system clears the admin session and redirects the user to the login page.

```
session_start();

session_destroy();

header('Location: login.php');
```



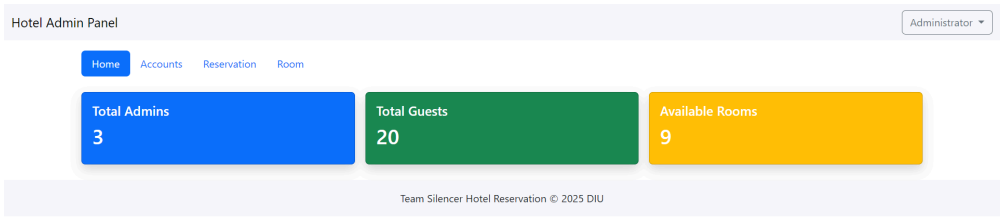


6.2 Admin Dashboard (home.php)

The **Admin Dashboard** serves as the central hub for managing the hotel system. It provides key metrics and quick access to various administrative functions.

Features:

- **Summary Statistics**: Displays information such as total bookings, available rooms, and ongoing transactions.
- **Navigation Panel**: Links to room management, account management, reservation management, and other sections.
- **Quick Actions**: Buttons for adding new rooms, viewing pending reservations, and processing transactions.

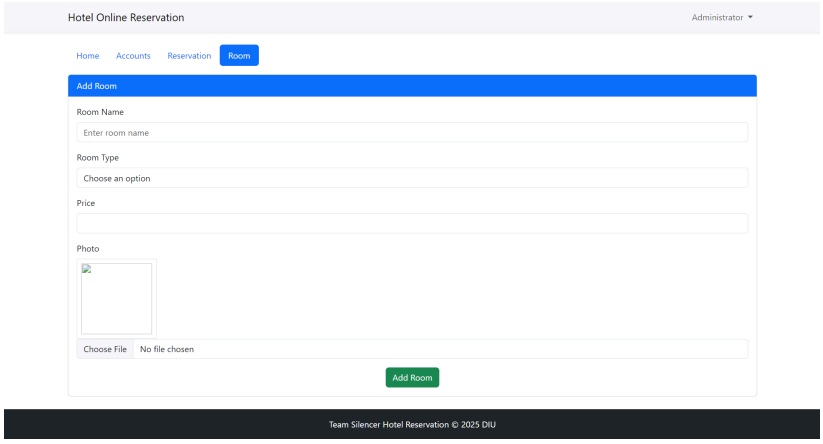


6.3 Room Management

This section is dedicated to managing the rooms available in the hotel. Admins can add, edit, or delete rooms based on hotel needs.

Add Room (add_room.php, add_query_room.php)

- Admin can add new rooms to the system by entering details like room type, price, and amenities.

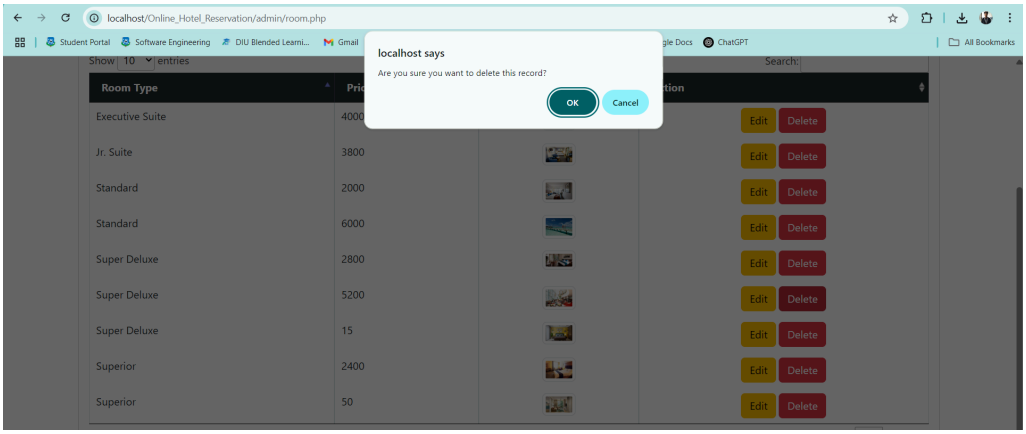


Edit Room (edit_room.php, edit_query_room.php)

- Admin can update the details of existing rooms (e.g., price change, room features update).

Delete Room (delete_room.php)

- Admin can delete rooms that are no longer needed. This will remove the room from the available inventory.

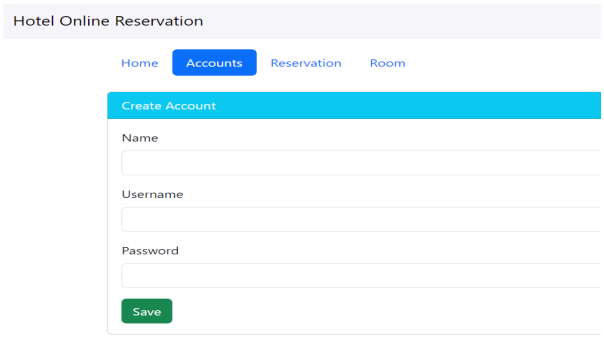


6.4 Account Management

Admin has the ability to manage user accounts for both regular customers and hotel staff.

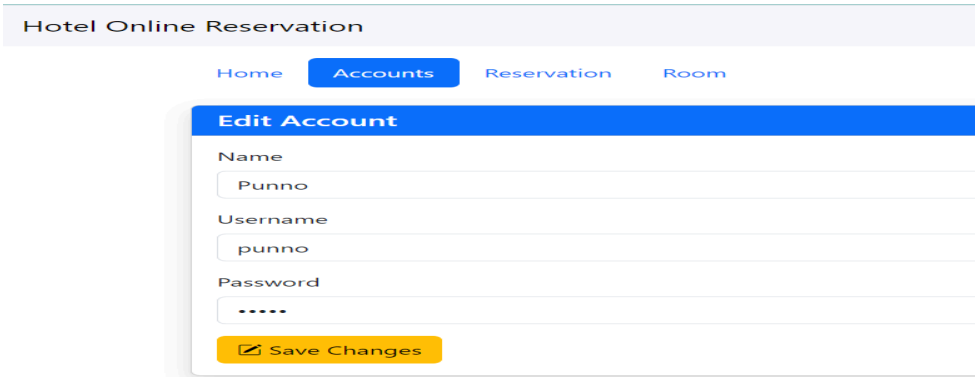
Add Account (add_account.php, add_query_account.php)

- Admin can create new user accounts (e.g., for hotel staff or customers). The system will store the credentials and user roles.



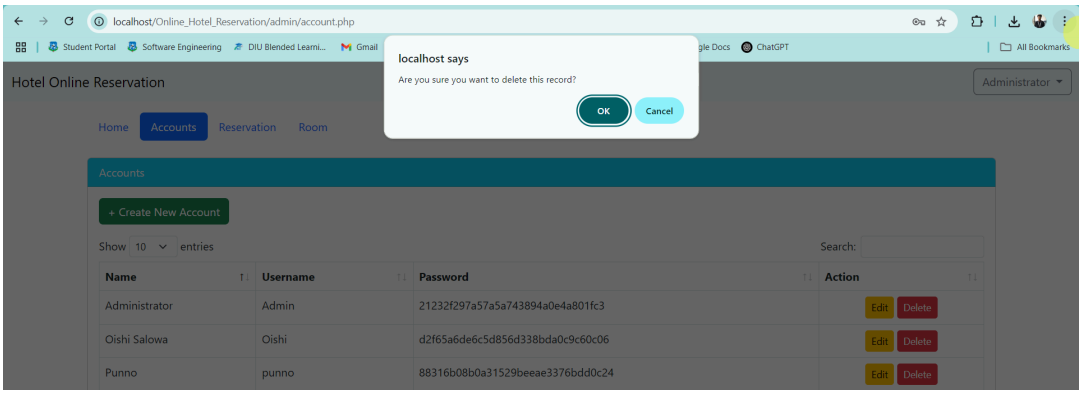
Edit Account (edit_account.php, edit_query_account.php)

- Admin can update existing account details, including changing passwords or roles.



Delete Account (delete_account.php)

- Admin can delete accounts that are no longer needed (e.g., user cancellations or staff removals).

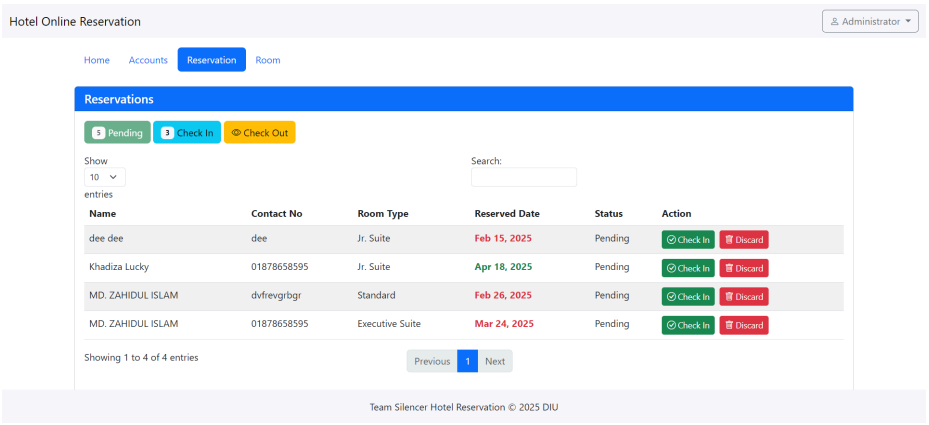


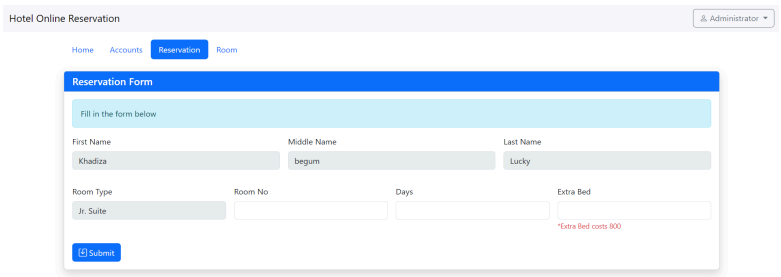
6.5 Reservation Management

Admins can manage the status of reservations and handle customer requests for booking.

View & Confirm Reservations (confirm_reserve.php)

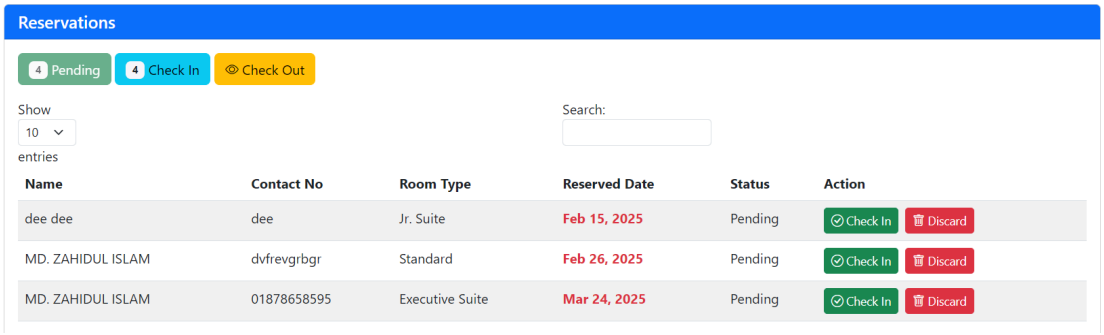
- Admin can view all reservations, confirm pending bookings, or cancel reservations if necessary.





Delete Pending Reservations (delete_pending.php)

- Admin can delete reservations that have not been confirmed or are no longer valid.

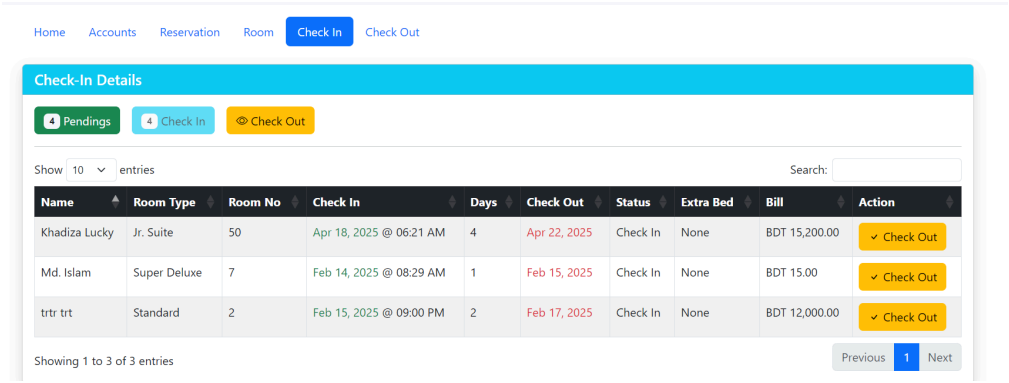


6.6 Check-in and Check-out

Admins handle the check-in and check-out process for guests.

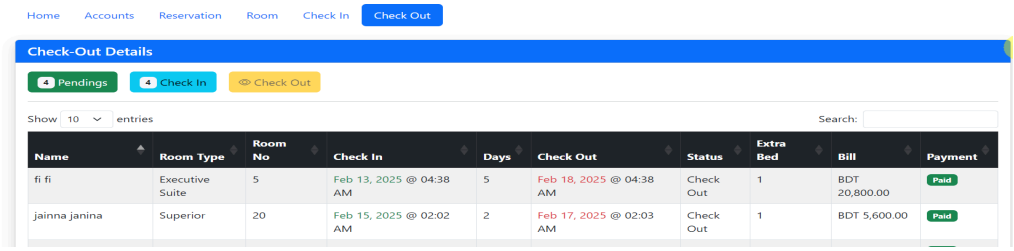
Check-in (checkin.php)

- Admin records the check-in time when the guest arrives at the hotel. This updates the system to mark the room as "occupied."



Check-out (checkout.php, checkout_query.php)

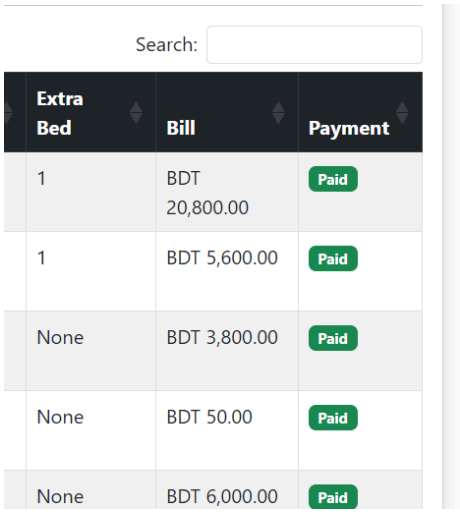
- Admin confirms the guest's departure and calculates the final bill. This operation also marks the room as available for future bookings.



6.7 Transactions (transaction.php)

This section is where the admin can track financial transactions, including payments for reservations, refunds, and adjustments.

- **Transaction Details**: Admin can view all transaction records, including payment methods and amounts.
- **Transaction Summary**: Displays financial data, such as total revenue and pending payments.



6.8 Admin Utility Files

The admin panel relies on several utility files to ensure proper functionality. These files handle database connections, form validations, and security features.

- **connect.php**: Responsible for connecting the admin panel to the database.
- **validate.php**: Handles form validation and checks if user credentials are correct.
- **name.php**: Contains helper functions for generating reports or displaying dynamic data.
- **save_form.php**: Handles data insertion into the database for forms like room addition or reservation confirmation.

7. Front-End Development

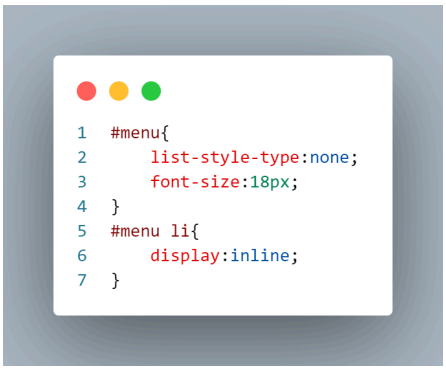
The front-end of the Hotel Reservation System is designed to be clean, responsive, and user-friendly, ensuring seamless interaction for both guests and administrators. It makes use of standard web technologies like HTML, CSS, JavaScript, Bootstrap, and jQuery. These tools together create an interface that is both visually appealing and functionally effective on various devices including desktops, tablets, and smartphones.

7.1 CSS Styling (style.css, bootstrap.css, bootstrap-theme.css)

Styling is a crucial part of the user interface as it defines the visual layout and aesthetics of the system.

Custom CSS (style.css)

- This file includes project-specific styles, such as color schemes, font choices, layout spacing, form elements, and table customizations.
- Ensures consistent branding and design throughout all pages of the system.
    - Helps to override default Bootstrap styles where needed to fit the system's design preference.



```
#menu{
    list-style-type:none;
    font-size:18px;
}
#menu li{
    display:inline;
}
```

Bootstrap CSS (bootstrap.css, bootstrap-theme.css)

- Provides a responsive grid system that automatically adjusts to screen sizes.
- Supplies predefined styles for buttons, tables, forms, alerts, modals, and more.
- Speeds up development and ensures compatibility with all modern browsers.
- Bootstrap's theme file is used to give a more polished look with subtle gradients and shadows.

```
html {
  font-family: sans-serif;
  -webkit-text-size-adjust: 100%;
      -ms-text-size-adjust: 100%;
}
body {
  margin: 0;
}
article,
aside,
details,
figcaption,
figure,
footer,
header,
hgroup,
main,
menu,
nav,
section,
summary {
  display: block;
}
audio,
canvas,
progress,
video {
  display: inline-block;
  vertical-align: baseline;
}
```

7.2 JavaScript and jQuery Integration (jquery.js, jquery.dataTables.js)

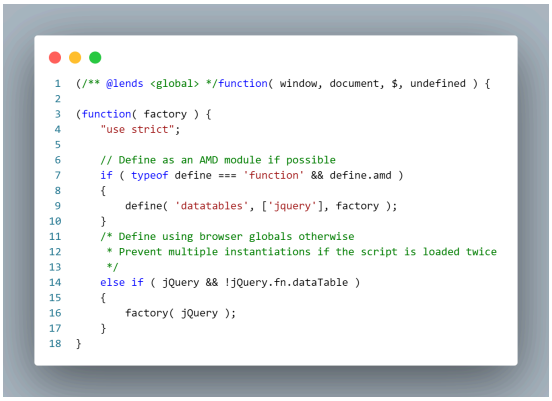
JavaScript enhances the system's interactivity, and jQuery simplifies DOM manipulation and AJAX operations.

jQuery (jquery.js)

- A lightweight JavaScript library that simplifies HTML document traversal and event handling.
- Used for real-time validation, animations, modals, and dynamic content loading without refreshing the page.

jQuery DataTables (jquery.dataTables.js)

- Provides advanced interaction controls for HTML tables, including:
  - Sorting
  - Pagination
  - Searching/filtering
- Used extensively in the Admin Panel to display reservations, room listings, and user data in an efficient and manageable way.



Sample Initialization:

```
$(document).ready(function() {

  $('#roomTable').DataTable();

});
```

7.3 DataTables in Admin Panel (dataTables.bootstrap.js)

To maintain a Bootstrap-styled interface while using DataTables, this integration file applies the Bootstrap theme to the DataTables layout.

Benefits:

- Seamless appearance with existing Bootstrap components.
- Responsive table structure with filter/search box, navigation controls, and dropdowns.
- Supports export functionalities (e.g., print, PDF) with extensions.

```
/* Set the defaults for DataTables initialisation */
$.extend(true, $.fn.dataTable.defaults, {
    "sDom": "<'row'<'col-sm-6'l><'col-sm-6'f>r>" + "t" + "<'row'<'col-sm-6'i><'col-sm-6'p>>",
    "oLanguage": {
        "sLengthMenu": "_MENU_ records per page"
    }
});


/* Default class modification */
$.extend($.fn.dataTableExt.oStdClasses, {
    "sWrapper": "dataTables_wrapper form-inline",
    "sFilterInput": "form-control input-sm",
    "sLengthSelect": "form-control input-sm"
});
```

Typical Use:

```
<link rel="stylesheet" href="css/dataTables.bootstrap.css">

<script src="js/jquery.dataTables.js"></script>

<script src="js/dataTables.bootstrap.js"></script>
```

7.4 Fonts and Icons Used (Glyphicons)

The system utilizes **Glyphicons**, which are icon fonts included in Bootstrap for a clean and lightweight interface.

Usage Areas:

- Navigation icons (dashboard, rooms, reservations)
- Buttons for actions like **Edit**, **Delete**, **Add**
- Visual cues for statuses (confirmed, pending, checked-in, checked-out)

Why Glyphicons?

- Minimal file size compared to image icons.

- Scalable without loss of quality.
- Simple to integrate using only CSS classes.

Responsive Design Implementation

All front-end components are built with responsiveness in mind. This ensures that users can access the system across different screen sizes (mobile, tablet, desktop).

Responsive Techniques Used:

- Bootstrap's grid system (col-md-*, col-sm-*, container-fluid)
- Media queries for fine-tuning layout in smaller screens
- Collapsible navigation menu for mobile view
- Flexible form layouts and table containers

User Experience Enhancements

Several enhancements have been added for better usability:

- **Hover effects** and **button transitions**
- **Loading animations** during data fetch or form submissions
- **Tooltip integrations** to guide users
- **Client-side form validations** using JavaScript for faster feedback

8. Security Implementation

Security is a top priority in any web-based system, especially one that handles sensitive data like customer reservations, personal information, and payment-related records. This section outlines the comprehensive security measures implemented in the Hotel Reservation System to protect both users and administrators.

8.1 Authentication and Session Handling

Authentication ensures that only legitimate users can access protected areas, especially the admin panel.

Login Verification

- Only users with valid credentials (email and password) are allowed access.
- Passwords are hashed using secure hashing algorithms (e.g., password_hash() and password_verify() in PHP) to prevent exposure even if the database is compromised.

```
if (password_verify($entered_password, $stored_hash)) {

  $_SESSION['user_id'] = $user['id'];
```

```
    $_SESSION['role'] = $user['role'];
}
```

Session Management

- PHP sessions track logged-in users and restrict access based on roles.
- Sessions are started on login and destroyed on logout.
- Pages requiring authentication use a session-checking mechanism to ensure the user is logged in.

```
session_start();
if (!isset($_SESSION['user_id']) || $_SESSION['role'] != 'admin') {
    header('Location: login.php');
    exit;
}
```

Session Timeout

- Inactivity-based timeouts can be implemented to automatically log out idle users and protect sessions from hijacking.

8.2 Input Validation

Input validation is crucial to protect the system from malformed or malicious user inputs.

Client-Side Validation

- Implemented using JavaScript and HTML5 form attributes (required, pattern, type).
- Prevents empty or invalid fields from being submitted.

Server-Side Validation

- All inputs are validated again on the server side, regardless of client-side checks.
- Ensures data integrity before insertion into the database.

```
$email = filter_var($_POST['email'], FILTER_VALIDATE_EMAIL);
if (!$email) {
    die("Invalid email format.");
}
```

8.3 SQL Injection Prevention

SQL injection is one of the most dangerous attacks on database-driven applications. The system implements multiple layers of defense:

Prepared Statements (mysqli or PDO)

- All database queries use prepared statements to separate SQL code from data.

```
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");

$stmt->bind_param("s", $email);

$stmt->execute();
```

- This protects against injection even if malicious SQL is entered.

Escaping Special Characters

- All inputs are sanitized using htmlspecialchars(), trim(), and mysqli_real_escape_string() where applicable.

8.4 Secure Admin Access

The admin panel is protected with strict access controls and additional security layers:

Role-Based Access Control

- Only users with the admin role can access admin functions.
- Role verification is done on every admin page to prevent privilege escalation.

URL Access Restriction

- Direct access to admin files is blocked unless the session is active and verified.
- Unauthorized users attempting access are redirected to the login page.

Captcha or 2FA (Optional Enhancement)

- CAPTCHA can be added to the login page to prevent brute force attacks.
- Two-Factor Authentication (2FA) could be added in the future for enhanced admin login security.

8.5 File Upload and Form Security

If any file upload feature is introduced (e.g., profile pictures, documents), these security steps should be followed:

- Only allow specific file types (e.g., .jpg, .png, .pdf)

- Check MIME type and size of uploaded files
- Store files outside the web root and serve via controlled PHP scripts

8.6 Cross-Site Scripting (XSS) Protection

To avoid XSS attacks:

- All dynamic data displayed on the web pages is encoded using htmlspecialchars()
- JavaScript inputs are sanitized and filtered

```
echo htmlspecialchars($user_input, ENT_QUOTES, 'UTF-8');
```

8.7 Cross-Site Request Forgery (CSRF) Protection

To prevent CSRF attacks:

- Tokens are added to all forms that perform critical operations like reservation submission or data modification.
- Tokens are stored in the session and validated upon form submission.

```
// Generate token

$_SESSION['token'] = bin2hex(random_bytes(32));

// Include in form

<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['token']; ?>">

// Verify token on server

if ($_POST['csrf_token'] !== $_SESSION['token']) {

   die("Invalid CSRF token.");

}
```

8.8 HTTPS and Secure Deployment (Deployment-Level Security)

Although the system runs on a local server (XAMPP) during development, secure deployment best practices include:

- Using **HTTPS** (SSL) on live servers
- Ensuring **secure cookie flags** for session cookies (HttpOnly, Secure)
- Periodic **security audits** and vulnerability scans

8.9 Error Handling and Logging

- **Error messages** are suppressed from end-users in production mode to prevent information leakage.

- **Error logs** are maintained to help developers debug issues without exposing sensitive data.

```
ini_set('display_errors', 0);

ini_set('log_errors', 1);

ini_set('error_log', '/var/log/php_errors.log');
```

8.10 Admin and User Password Policy

To ensure strong credentials:

- Passwords must be at least 8 characters long
- Recommended to include a mix of uppercase, lowercase, digits, and symbols
- Password change mechanism requires verifying the current password first

9. Testing and Validation

Testing is a fundamental stage in software development that ensures the system behaves as expected, both functionally and non-functionally. For the Hotel Reservation System, various testing methodologies were applied to validate different modules, from user interactions to backend logic and database integrity.

This section discusses the different levels and types of testing performed throughout the development lifecycle and outlines how bugs were tracked, documented, and resolved.

9.1 Testing Methodologies

To ensure comprehensive quality assurance, a combination of testing methodologies was implemented:

1. Manual Testing

- Developers and testers interact with the system directly to check its functionality.
- Used for UI testing, form submission, login/logout, reservation process, and admin panel usage.

2. Black Box Testing

- Focuses on validating output without knowledge of internal code logic.
- Ideal for testing inputs and outputs of forms (e.g., reservation form, contact form).

3. White Box Testing

- Involves analyzing and testing the internal logic of PHP functions, database queries, and session handling.

- Helps identify logical errors and faulty conditions within the code.

4. Regression Testing

- After any update or bug fix, previous functionalities were re-tested to ensure no new issues were introduced.
- Ensures the stability of the overall system during continuous development.

5. Cross-Browser Testing

- Ensures the UI behaves consistently across major browsers: Chrome, Firefox, Edge, and Safari.
- Responsive checks were also conducted on mobile browsers.

9.2 Functional Testing

This testing focused on verifying that all components and user flows functioned as intended.

User Side Functionalities Tested:

- Room search and view mechanism
- Reservation form validation and submission
- Reservation confirmation page
- Contact form message delivery
- Confirmation and reply handling

Admin Panel Functionalities Tested:

- Login/logout authentication
- Dashboard redirection and statistics loading
- Room management: add, edit, delete rooms
- Account management: create, edit, delete admin accounts
- Reservation handling: view, confirm, or delete pending reservations
- Check-in and check-out processes
- Transaction view and data consistency

Test Case Example:

Test Case	Input	Expected Output	Status
Login as Admin	Email + Password	Redirect to home.php	✅ Passed

Submit Reservation	Valid data	Show confirmation.php	✅ Passed
Delete Room	Existing Room ID	Room removed	✅ Passed

9.3 Unit and Integration Testing

Unit Testing

- Each PHP function (e.g., user validation, database insertion, email sending) was tested in isolation.
- Included tests for:
  - add_query_reserve.php: inserting reservation
  - checkin.php: date validations and update status
  - login.php: login logic and error handling

Integration Testing

- Verified the interaction between multiple components.
- For example:
  - Reservation form → Database → Confirmation page
  - Admin action → Database update → Dashboard summary

These tests ensured that not only individual scripts worked, but also that the data flowed properly between them.

9.4 Bug Tracking and Fixes

During the development and testing phases, a simple bug tracking sheet was maintained (could be done via Excel, Google Sheets, or GitHub Issues). Each bug entry included:

- Bug ID
- Description
- Module affected
- Severity (Low, Medium, High)
- Status (Open, In Progress, Resolved)
- Fix Version

Common Bugs Identified and Fixed:

Bug ID	Description	Severity	Fix Summary
#101	Reservation not recorded for special characters in name	Medium	Applied input sanitization
#108	Admin logout not destroying session completely	High	Fixed session destroy method
#115	Room number field accepted alphabets	Low	Added input type & validation
#124	Double check-in possible due to missing status check	High	Added check to prevent re-checkin

Each issue was fixed, re-tested, and verified before being marked resolved. Bugs were typically fixed within 24–48 hours depending on severity.

9.5 Validation Tools and Approaches

In addition to manual and scripted tests, validation tools were used:

HTML/CSS Validators

- W3C Markup Validator for HTML
- CSS Validation Service for style sheets

JavaScript Testing

- Console debugging and error logging
- Browser dev tools for inspecting AJAX calls

Database Testing

- Checked for:
    - Integrity constraints
    - Null value handling
    - Duplicate entry prevention
    - Cascading delete behavior (e.g., deleting room → remove reservations)

9.6 Security Testing (Basic Level)

While not a full-scale penetration test, some basic security tests were conducted:

- Attempted SQL injections on login and reservation forms
- Cross-site scripting (XSS) attempts on feedback and contact forms
- Checked for session fixation and role-based access violations
- Verified logout functionality destroyed session cookies

9.7 Final Acceptance Testing

Before final deployment, a full walkthrough of the system was done by:

- The development team
- A group of end-user testers (could be friends, staff, or students)

They followed real-life scenarios such as:

- Booking a room from the homepage
- Logging into the admin panel
- Confirming a pending reservation
- Performing a check-in and check-out cycle

Any last-minute suggestions or UX issues were addressed based on their feedback.

10. Deployment and Hosting

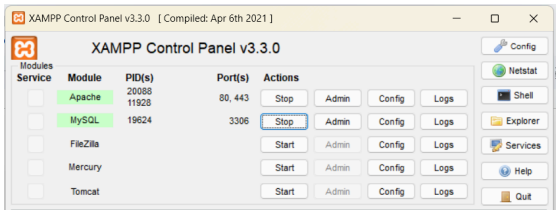
Deployment is the process of delivering a functional and stable version of a web application to a production or testing environment. For the Hotel Reservation System, initial development and testing were conducted locally using XAMPP, and provisions have been outlined for deployment to a live server with security and performance in mind.

10.1 Local Setup Using XAMPP

XAMPP is a free, open-source cross-platform web server solution package that includes Apache, MySQL, PHP, and Perl. It allows developers to build and test PHP-based applications on their own systems before deploying them live.

Installation and Configuration Steps:

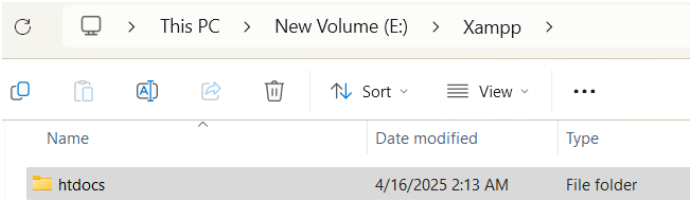
1. **Download and Install XAMPP**:
    - Download the XAMPP installer from https://www.apachefriends.org/



- Install the package and start Apache and MySQL modules from the XAMPP control panel.

2. **Place Files in the htdocs Directory**:

The Hotel Reservation System files are placed in:
C:\xampp\htdocs\hotel_reservation



3. **Launch the System in Browser**:

Access the system by visiting:
http://localhost/hotel_reservation/index.php

4. **Database Configuration**:
   - Import the provided SQL file (db_hor.sql) via phpMyAdmin.

Modify the connect.php file to ensure proper database credentials:
$conn = new mysqli("localhost", "root", "", "db_hor");

10.2 Folder Structure and File Organization

A clean and structured file system is maintained for easy navigation and maintainability.

Folder Hierarchy:

```
hotel_reservation/
│
├── admin/
│   ├── add_room.php
│   ├── edit_room.php
│   ├── confirm_reserve.php
│   ├── checkin.php
│   ├── transaction.php
│   └── ...
│
├── includes/
│   ├── connect.php
│   ├── validate.php
│   ├── name.php
│   └── save_form.php
│
├── css/
│   ├── style.css
│   ├── bootstrap.css
│   └── bootstrap-theme.css
│
├── js/
│   ├── jquery.js
│   ├── dataTables.bootstrap.js
│   └── jquery.dataTables.js
│
├── images/
│   └── logo.png, room1.jpg, etc.
│
├── index.php
├── reservation.php
├── aboutus.php
├── contactus.php
└── readme.txt
```

This logical separation of frontend, backend, and resource files ensures modularity and helps future developers manage or extend the application easily.

10.3 Database Import Instructions

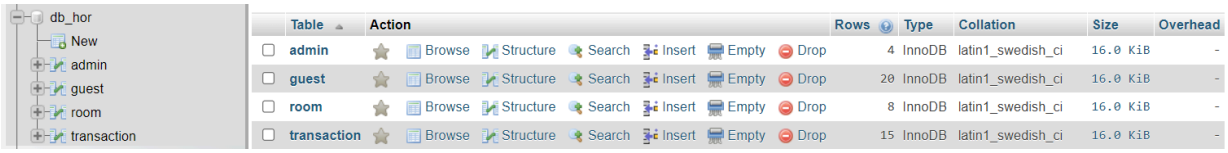
The project includes a .sql dump file named db_hor.sql, which creates and populates all the required tables.

Steps to Import Database:

1. Open http://localhost/phpmyadmin
2. Click **"New"** to create a new database, name it db_hor
3. Go to the **Import** tab
4. Choose the db_hor.sql file from the project directory
5. Click **Go** to execute the import

After successful import:

- All tables (users, rooms, reservations, etc.) will be available
- Default admin credentials can be found in the users table or set manually



10.4 Hosting Recommendations

While XAMPP is ideal for local testing, deploying the project online allows users and staff to access it from anywhere. Below are hosting guidelines:

Shared Hosting

- Ideal for smaller scale deployments with limited budgets
- Choose hosts that support PHP 7.4+ and MySQL (e.g., Hostinger, Bluehost, Namecheap)
- Upload files to the /public_html directory using FTP (e.g., FileZilla)
- Use cPanel's MySQL Database Wizard to set up database and import db_hor.sql

VPS or Dedicated Server

- Recommended for large hotels or systems expecting high traffic
- Provides full root access and flexibility
- Use Apache or Nginx with PHP and MySQL
- Secure the server with firewalls and SSL (Let's Encrypt or paid certificates)

Deployment Steps for Live Server:

1. Upload the entire project folder to the server
2. Import the db_hor.sql file into the hosting control panel's database manager

Edit connect.php to use live database credentials:
$conn = new mysqli("your_host", "your_user", "your_password", "db_hor");

3. Next movement is need to go to domain or subdomain
4. Configure domain or subdomain (e.g., hotel.example.com) to point to the project directory
5. Test all modules to ensure proper functioning

10.5 Optional: Domain, SSL, and Email Integration

- **Domain Name**: Purchase from registrars like GoDaddy or Namecheap.
- **SSL Certificate**: Protect user data and enable HTTPS using Let's Encrypt (free) or paid SSL.
- **Email Integration**: Use SMTP (via PHPMailer or similar) for sending confirmation emails after reservations.

10.6 Deployment Best Practices

- **Backup**: Always take full backups (files + database) before updates.
- **File Permissions**: Set proper file and directory permissions (644 for files, 755 for directories).
- **Error Handling**: Turn off detailed errors in production by updating php.ini and error_reporting() settings.
- **Performance Optimization**:
    - Minify CSS/JS

- Enable caching
- Optimize database indexes

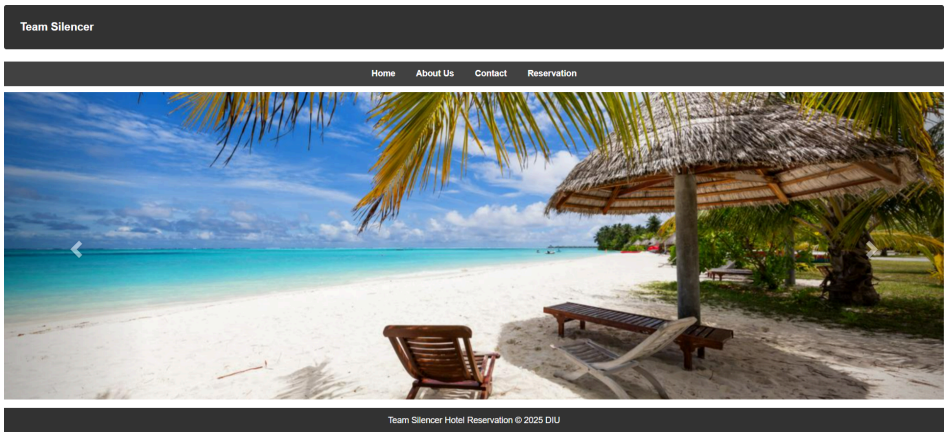
11. Screenshots and User Interface

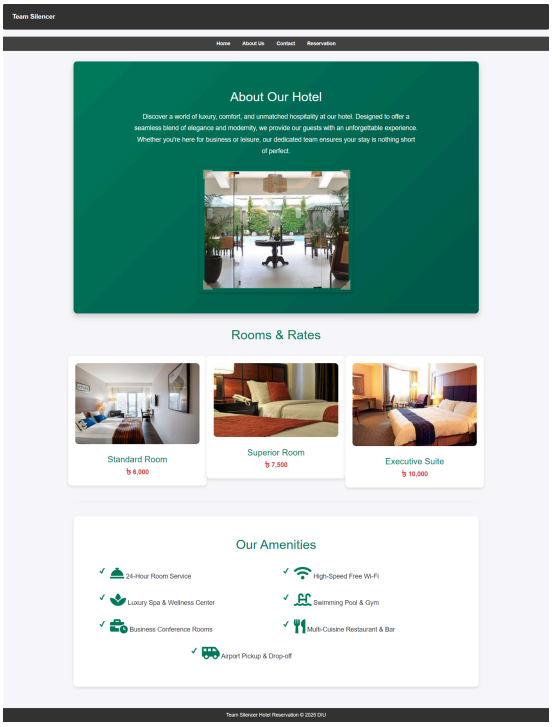
A well-designed user interface (UI) enhances usability, improves user satisfaction, and ensures that the system is intuitive to navigate. The Hotel Reservation System was developed with a clean and responsive interface using Bootstrap and custom CSS. This section provides a detailed visual and descriptive overview of the system's major interface components, from the public-facing homepage to the admin dashboard.

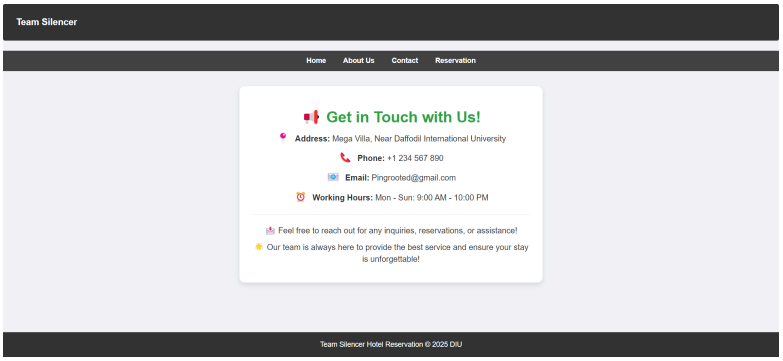
11.1 Homepage (index.php)

Overview:

- The homepage is the first page visitors see.
- Designed with a hotel-themed banner and a welcome message.
- Features navigation to other sections like About Us, Contact, Rooms, and Reservations.







Features Displayed:

- Responsive navigation bar with links
- Hero section with images and introduction text
- Quick search area (optional in future versions)
- Footer with hotel address, contact info, and social links

Design Highlights:

- Bootstrap carousel (optional) for displaying hotel images
- Clean layout with call-to-action buttons like “Book Now”

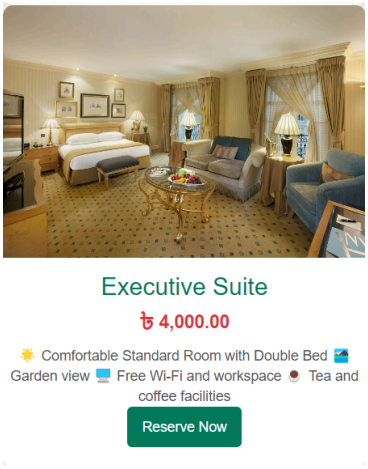
11.2 Booking/Reservation Form (reservation.php)

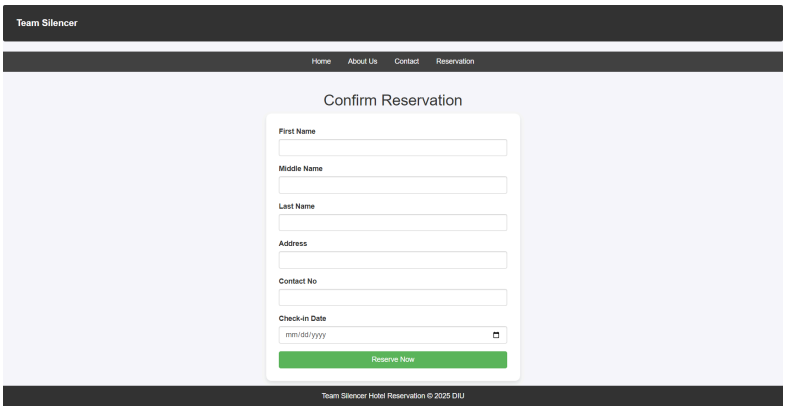
Purpose:

Allows users to select and submit details about their desired room and stay duration.

Form Fields Include:

- Full name
- Email address
- Check-in and Check-out dates
- Number of guests
- Room type
- Additional requests/comments





Features:

- Date pickers for date fields
- Input validation via JavaScript and PHP
- Form submission sends data to add_reserve.php

UI/UX:

- Easy-to-read labels and spacing
- Feedback messages after form submission (e.g., “Reservation Sent Successfully”)

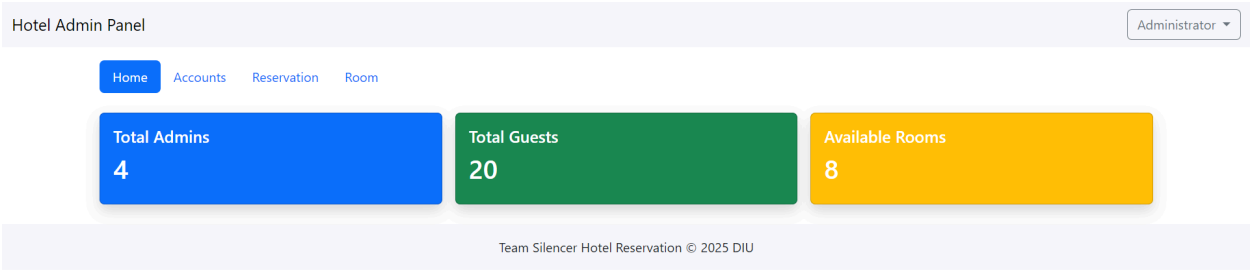
11.3 Admin Dashboard (home.php)

Purpose:

The central management panel for hotel admins to view and manage reservations, rooms, and user accounts.

Dashboard Displays:

- Summary boxes (Total Reservations, Available Rooms, Checked-in Guests, etc.)
- Graphs or tables (optional future enhancement)
- Quick links to major modules (e.g., Add Room, View Reservations)



Interface Design:

- Left-side or top navigation bar with admin options
- Bootstrap cards for data summaries
- Uses DataTables to display reservation and room data in paginated, searchable tables

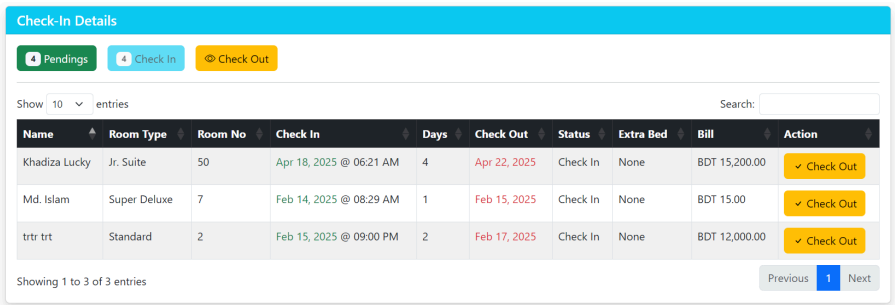
Access Control:

- Only visible to logged-in admins
- Admin sessions are handled securely

11.4 Check-in and Check-out Panels

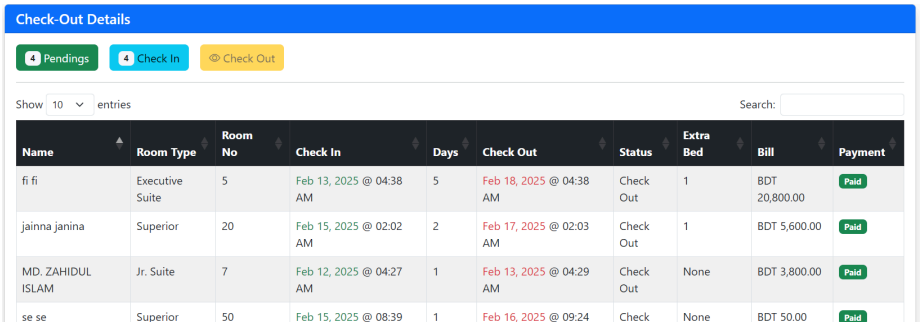
Check-in Panel (checkin.php):

- Displays reservations that are pending check-in
- Admin can mark guests as checked-in with one click
- Uses dropdowns or buttons beside each reservation entry



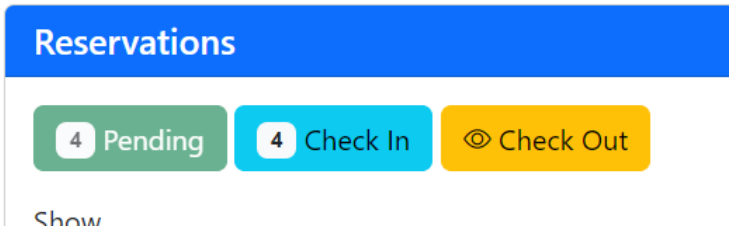
Check-out Panel (checkout.php):

- Shows all checked-in guests
- Admin can select and finalize check-out
- Automatically updates status and optionally generates receipts



UI Elements:

- Table view with status labels (Pending, Checked-in, Checked-out)
- Confirmation dialogs for critical actions



11.5 Transactions View (transaction.php)

Purpose:

Displays a record of all completed check-outs and payments (if integrated).

Displayed Fields:

- Guest name
- Room details
- Duration of stay
- Total amount
- Date of transaction
- Payment status (for future online payment integration)

Name	Room Type	Room No	Check In	Days	Check Out	Status	Extra Bed	Bill	Payment
fi fi	Executive Suite	5	Feb 13, 2025 @ 04:38 AM	5	Feb 18, 2025 @ 04:38 AM	Check Out	1	BDT 20,800.00	Paid
jainna janina	Superior	20	Feb 15, 2025 @ 02:02 AM	2	Feb 17, 2025 @ 02:03 AM	Check Out	1	BDT 5,600.00	Paid
MD. ZAHIDUL ISLAM	Jr. Suite	7	Feb 12, 2025 @ 04:27 AM	1	Feb 13, 2025 @ 04:29 AM	Check Out	None	BDT 3,800.00	Paid

Design Aspects:

- Organized table format using DataTables
- Filters and sorting options
- Export or print functionality (optional feature)

11.6 Additional UI Screens *(Optional but recommended)*

About Us Page (aboutus.php):

- Hotel background, mission, and team introduction
- Styled with images and text blocks

Contact Us Page (contactus.php):

- Contact form with fields: Name, Email, Subject, Message
- Google Map integration (optional)
- Contact details displayed alongside the form

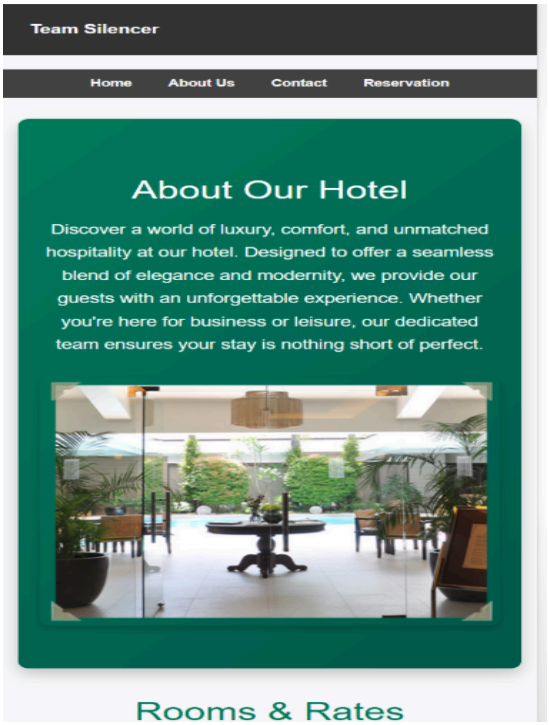
Error Pages (404, Access Denied):

- Custom-designed error pages to handle unauthorized access or incorrect URLs
- Friendly messages and navigation back to the homepage

11.7 Mobile Responsiveness

All user-facing pages and admin modules were tested on mobile devices and are fully responsive. This ensures.

- Proper scaling and alignment on different screen sizes
- Clickable buttons and readable text on smartphones and tablets
- Navigation collapses into a hamburger menu on smaller devices



Design Summary

Module	Design Framework	Key UI Features
Homepage	Bootstrap + Custom CSS	Banner, navbar, footer
Reservation	Form design with validation	Responsive layout, alerts
Admin Dashboard	DataTables, Cards	Summary panels, quick actions
Check-in/Check-out	Tables + status badges	Admin interaction panels
Transactions	Table with filters	Historical data view

12. Conclusion and Future Scope

12.1 Summary of Achievements

The Hotel Reservation System was developed to automate and streamline the entire hotel booking process, benefiting both users and hotel staff. The system provides a web-based interface where customers can book rooms online, and administrators can manage reservations, check-ins, check-outs, transactions, and hotel room data.

Key Achievements:

- **User-friendly reservation interface** for customers to browse rooms and make bookings.
- **Admin panel with full control** over room management, reservation confirmations, check-ins, check-outs, and transaction monitoring.
- **Secure login system** for admin access with session handling.
- **Responsive design** for both desktop and mobile users.
- **Data storage and retrieval** using MySQL database and optimized SQL queries.
- **Implementation of modern web technologies** such as PHP (with mysqli), JavaScript, Bootstrap, and jQuery for both client-side and server-side functionality.
- **Validation and security** features like input sanitization and basic SQL injection protection were applied throughout the system.

Overall, the system successfully automates hotel operations and reduces manual workload, improving operational efficiency and accuracy.

12.2 Challenges Faced During Development

Every project has its obstacles. Here are some challenges encountered and how they were addressed:

1. User Authentication and Session Handling:

- Ensuring secure session-based login/logout functionality required careful handling to prevent unauthorized access.
- PHP sessions were used to maintain state and restrict access to admin modules.

2. Dynamic Room Availability Checking:

- Making sure that rooms are not overbooked required real-time data verification from the database.
- Reservation overlaps were prevented using date-range logic in SQL queries.

3. Frontend Design and Responsiveness:

- Designing a visually appealing yet responsive layout for mobile devices while keeping the user experience simple and elegant.
- Bootstrap was used to maintain a consistent design across all devices.

4. Admin Module Complexity:

- Managing CRUD operations for multiple entities like users, rooms, and reservations while maintaining data integrity.
- DataTables integration and AJAX-based loading were considered (and partially implemented) to handle large datasets more efficiently.

5. Security Concerns:

- Preventing SQL injection, securing admin login, and input validation were vital aspects.
- Prepared statements (mysqli) and form validation were implemented as solutions.

12.3 Future Enhancements

The current version of the Hotel Reservation System provides all the core functionalities needed by a hotel to manage its bookings. However, there are several potential features and enhancements that can significantly improve the system's capabilities in future versions:

1. Online Payment Integration

- **Goal**: Allow customers to make secure payments directly through the website using credit/debit cards or digital wallets.
- **Implementation Suggestion**:
    - Integrate with payment gateways like **PayPal**, **Stripe**, or **SSLCommerz**.
    - Automatically generate invoices and mark transactions as “Paid” upon successful payment.

2. User Reviews and Ratings

- **Goal**: Enable guests to leave feedback about their experience.
- **Benefits**:
    - Builds trust and provides insights to hotel management.
    - Public reviews can attract future customers.
- **Implementation Suggestion**:
    - Add a “Rate Your Stay” section post check-out.
    - Store ratings in the database and display them on room detail pages.

3. Mobile Application Support

- **Goal**: Provide a native mobile experience for both guests and admins.
- **Platforms**: Android and iOS.

- **Implementation Suggestion**:
    - Develop using **Flutter** or **React Native** for cross-platform support.
    - Sync app data with the central MySQL database via a RESTful API built in PHP.

4. Multi-Hotel or Chain Hotel Support

- **Goal**: Enable one system to manage multiple hotel branches.
- **Benefits**:
    - Scales the system for larger operations.
- **Implementation Suggestion**:
    - Add a new "hotel_id" field to associate rooms and reservations with specific branches.

5. Role-Based Access Control

- **Goal**: Implement different permission levels (e.g., receptionist, manager, admin).
- **Example**:
    - A receptionist can check-in guests but cannot delete accounts.
    - An admin has full access to the system.

6. Reporting and Analytics

- **Goal**: Generate reports on income, occupancy rates, booking trends, etc.
- **Types of Reports**:
    - Daily/Monthly reservations
    - Peak seasons
    - Financial summaries

7. Email Notifications and Reminders

- **Goal**: Send automatic emails for booking confirmations, reminders, and feedback requests.
- **Implementation Suggestion**:
    - Use PHPMailer with SMTP support for better reliability and formatting.
    - Schedule reminders using CRON jobs (if hosted on a server).

13. References

The development of this Hotel Reservation System has involved the utilization of various online resources, programming libraries, and developer tools. These references played a critical role in guiding system design, implementing complex functionality, and solving technical challenges.

13.1 Online Resources

These are the primary websites, tutorials, documentation pages, and forums used for research, troubleshooting, and implementation guidance:

Source	Purpose	URL
PHP Manual	Core PHP syntax, functions, mysqli usage	https://www.php.net
W3Schools	HTML, CSS, JavaScript reference and examples	https://www.w3schools.com
MDN Web Docs	JavaScript, CSS, DOM standards and practices	https://developer.mozilla.org
Bootstrap Docs	Front-end layout and responsive design	https://getbootstrap.com
Stack Overflow	Troubleshooting, code suggestions, error handling	https://stackoverflow.com
jQuery Documentation	DOM manipulation, event handling	https://api.jquery.com
DataTables Docs	Integration of dynamic tables in admin panel	https://datatables.net
XAMPP Official	Configuration and setup of local web server	https://www.apachefriends.org
GitHub Repos	Referenced sample CRUD apps and UI elements	https://github.com
SQLBolt	MySQL query syntax and practice problems	https://sqlbolt.com
TutorialsPoint	PHP sessions, form handling, validation tutorials	https://www.tutorialspoint.com

13.2 Libraries and Plugins Used

This system integrates several open-source libraries and plugins that enhance front-end and back-end functionalities. Below is a list of libraries used and their role:

Front-End Libraries

- **Bootstrap (v3.3.x)**
  - *Use:* UI design, layout structure, and responsive elements
  - *CDN / Local Usage*: bootstrap.css, bootstrap-theme.css
- **jQuery (v3.x)**
  - *Use:* DOM manipulation, animations, form validation
  - *File Used:* jquery.js
- **Glyphicons (Bootstrap Icons)**
  - *Use:* UI icons for buttons, navigation, status indicators
- **DataTables.js**
  - *Use:* Admin panel data display, search, sort, and pagination
  - *Files Used:* jquery.dataTables.js, dataTables.bootstrap.js

Back-End Libraries / Tools

- **PHP (7.x or 8.x)**
  - *Use:* Server-side scripting, session management, database connection
  - *Functions Used:* mysqli_connect, session_start, include, etc.
- **MySQL (via phpMyAdmin on XAMPP)**
  - *Use:* Structured database for storing all hotel data
  - *DB Name:* db_hor
- **XAMPP (Apache + MySQL + PHP + phpMyAdmin)**
  - *Use:* Local development environment
  - *Version Used:* Latest compatible with PHP 7/8

Utility Scripts and Helpers

- **validate.php**
  - Handles form input validation and cleaning.
- **connect.php**
  - Establishes and maintains database connection using mysqli.
- **name.php, save_form.php**
  - Helper files to handle modular functions.

Text Editors / IDEs Used

- **Visual Studio Code**
  - *Purpose:* Writing and organizing source code with syntax highlighting and plugin support

- **Sublime Text (Optional)**
    - Lightweight editor for quick edits

13.3 Additional Acknowledgments

We also acknowledge the open-source community, bloggers, and YouTube educators who consistently publish tutorials, demos, and code reviews which serve as vital learning materials and reference sources. This includes:

- Traversy Media (YouTube)
- CodeCourse
- Programming with Mosh
- DevDojo
- CodeAcademy

13.4 Licensing Notice

All third-party libraries used in this project fall under open-source licenses (MIT, GPL, etc.). Appropriate license information should be included with any distribution of this system to ensure compliance.

14. Appendix

The appendix serves as a supplementary section that includes practical materials referenced throughout the project. It provides readers, reviewers, or developers with ready access to source code samples, SQL structure, interface snapshots, and the README file that guides initial setup. These resources make the project easier to understand, test, and deploy.

14.1 Source Code Snippets

Here are essential code snippets from different modules that demonstrate key operations like database connectivity, reservation handling, and admin authentication.

Database Connection - connect.php

```php
<?php
    $conn = new mysqli("localhost", "root", "", "db_hor") or die(mysqli_error());

```

Reservation Query - add_query_reserve.php

```php
<!-- Fetch Data -->
<?php
    $q_p = $conn->query("SELECT COUNT(*) as total FROM `transaction` WHERE `status` = 'Pending'") or die(mysqli_error($conn));
    $f_p = $q_p->fetch_array();
    $q_ci = $conn->query("SELECT COUNT(*) as total FROM `transaction` WHERE `status` = 'Check In'") or die(mysqli_error($conn));
    $f_ci = $q_ci->fetch_array();
?>
```

Admin Login Validation - login.php

```php
<?php
    if(ISSET ($_POST['login'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $query = $conn->query("SELECT * FROM `admin` WHERE `username` = '$username' && `password` = '$password'") or die(mysqli_error());
        $fetch = $query->fetch_array();
        $row = $query->num_rows;

        if($row > 0){
            session_start();
            $_SESSION['admin_id'] = $fetch['admin_id'];
            header('location:home.php');
        }else{
            echo "<center><labe style = 'color:red;'>Invalid username or password</label></center>";
        }
    }
?>
```

14.2 SQL Dump File (db_hor.sql)

This file contains the full database schema for the Hotel Reservation System, including table definitions, foreign key relationships, and sample data for testing.

Sample Tables:

- rooms
- reservations
- admins
- transactions

14.3 README File (readme.txt)

This file provides guidance for setting up the system locally.