**Final Project Report**

**for**



**THE SHARP HOTEL MANAGEMENT SYSTEM**

**Prepared by**

**Group Name: Group F**

**Members name** **Student ID** **Student email**

**Pardeep Kaur** **100404907** [**pardeepkaur7@student.kpu.ca**](mailto:pardeepkaur7@student.kpu.ca)

**Harmanpreet Kaur**  **100421083** [**harmanpreetkaur19@student.kpu.ca**](mailto:harmanpreetkaur19@student.kpu.ca)

**Amandeep Singh** **100420043** [**amandeepsingh38@student.kpu.ca**](mailto:amandeepsingh38@student.kpu.ca)

**Randeep Singh** **100424984** [**randeep.singh9@student.kpu.ca**](mailto:randeep.singh9@student.kpu.ca)

**Shilani Sharma** **100415600** [**shilani.sharma@student.kpu.ca**](mailto:shilani.sharma@student.kpu.ca)

**Instructor: Nukhet Tuncbilek**

**Course: INFO 2413**

**Section: S50**

**Date: 15 August 2023**

**Table of Contents:**

[PLANNING (PHASE I) 2](#_Toc2003527504)

[REQUIREMENT ANALYSIS (PHASE II) 3](#_Toc1220536276)

[FUNCTIONALITY OF THE SYSTEM: 4](#_Toc1931762108)

[PROJECT DESIGN (PHASE III) 4](#_Toc1914846096)

[IMPLEMENTATION (PHASE IV) 7](#_Toc780295109)

[INTRODUCTION: 7](#_Toc1298560237)

[PROJECT STRUCTURE: 7](#_Toc1541744974)

[UI IMPLEMENTATION: 9](#_Toc322169273)

[DATABASE IMPLEMENTATION: 10](#_Toc707534783)

[BACK-END IMPLEMENTATION: 15](#_Toc1838573595)

[TESTING: 18](#_Toc930788090)

[USER MANUAL 20](#_Toc1160593309)

[TOOLS 23](#_Toc13127099)

[REFERENCES 23](#_Toc41505470)

# **PLANNING (PHASE I)**

A detailed Hotel Management System will be created for The SHARP hotel, according to The Sharp Project Plan. The goal of the plan is to build an understanding among the project's main stakeholders, including the project sponsor, project manager, project team, and other individuals engaged, regarding the project's scope, goals, objectives, and deliverables.

Background:

The SHARP hotel is a recently opened business looking for a well-structured website to manage its activities. Customers should be able to conveniently book rooms and make payments on the website thanks to a seamless user experience.

Goals and Objectives:

For The SHARP hotel, the project's goals and objectives include automating hotel operations to streamline them, improving customer satisfaction with a user-friendly system, increasing staff productivity, ensuring scalability and adaptability, offering training and user adoption materials, and achieving project success and client satisfaction.

Scope of the Project:

The project's scope includes creating a website to handle reservations, payments, check-in and check-out procedures, room details, and reservation fees. A mobile application, any future requirements, and reporting and analytics tools are expressly excluded from the project.

Risk evaluation:

Risks are recognized, described, and prioritized, and documented strategies for mitigation are used. The evaluation will be followed and updated continuously. The leadership group will frequently be informed of changes and emergencies.

Feasibility Assessment:

The project's economic, technical, operational, legal, and political viability are all assessed. Return on investment, technical capabilities, user acceptance, system performance, regulatory compliance, and stakeholder alignment are important factors to consider.

Roles and responsibilities for the project:

There are definitions for roles like project manager, members of the project development team, and experts. Resource management, decision-making, supervision, development, coordination, expertise lending, and quality assurance are among the responsibilities.

Issue Management:

Managing issues: A thorough strategy to managing issues involves recording, prioritizing, resolving, communicating, documenting, and ongoing quality control checks.

Communication Plan:

A thorough communication plan outlines interactions between the project manager, experts, project participants, and project sponsors. The use of regular meetings, reports, updates, and documentation as communication channels helps to promote effective collaboration and a common understanding.

**REQUIREMENT ANALYSIS (PHASE II)**

The SHARP Hotel Management System Software Requirement Specification is a detailed guide that outlines the software requirements for the hotel management system. The document provides an overview of the subject matter, target audience, definitions, formatting guidelines, references, and acknowledgments. It gives information on the entire hotel management system and outlines the functional and non-functional requirements that the programme must satisfy to support hotels' operations in an efficient manner. The document provides a thorough overview of the functional requirements for the system, as well as use examples and scenarios that illustrate interactions between users, outside entities, and the system. Detailed descriptions, acceptance standards, and any relevant limitations or dependencies are included for each functional requirement. Non-functional requirements are also stated in the paper, outlining the desired characteristics and qualities of the program. Performance requirements, such as response times or throughput rates, reliability and availability requirements, security and data privacy guidelines, usability and accessibility guidelines, and other quality attributes deemed important for the hotel management system are examples of these. The document gives a clear set of instructions and standards for the development team to follow, ensuring that the program meets the expectations of stakeholders. It acts as a reference throughout the project lifecycle for confirming the software's compliance, supporting effective communication between the development team and stakeholders, and providing a foundation for future system maintenance and modifications.

In summary, the SHARP Hotel Management System Software Requirement Specification is a comprehensive document that outlines the software requirements for the hotel management system. It provides a comprehensive overview of the system's functionality, non-functional requirements, and requirements, ensuring that the software meets the expectations of stakeholders and serves as a reference throughout the project lifecycle. The SHARP hotel management system aims to create a comprehensive and fully functional solution for effective day-to-day hotel management. It involves building a dynamic website connected to a highly secure database, which automates and streamlines important hotel processes, enabling staff members to work efficiently online.

## ***FUNCTIONALITY OF THE SYSTEM:***

The system features a system for booking rooms, front desk operations, customer profile management, mobile accessibility, accounting management, staff record management, employee scheduling, reporting and feedback, and security. The SHARP hotel management system can benefit hotels in several ways, including streamlined operations, improved efficiency, enhanced guest experience, accurate financial management, real-time information, and enhanced security. It includes modules such as room booking, front-desk operations, customer profile management, accounting management, staff record management, and employee scheduling, enhancing the guest experience and building customer loyalty. The SRS document is intended for several types of readers, including the client, who orders the hotel management system's development, and the professor, who reviews the document as part of an academic exercise. The client can provide feedback, check requirements, and ensure the final product meets their expectations.

# **PROJECT DESIGN (PHASE III)**

The Design Document for the SHARP Hotel Management System is a document that aims to design the features of the website, aiming to enhance the hotel's online presence, customer experiences, and effective management of hotel operations and services. The document focuses on high-level system design, system architecture, database design, user interface design, and integration needs. The purpose of the Design Document is to highlight key features that will be performed by the website. Key content included in this document are the conceptual data model, which helps understand the relationships within the system, the logical data model, attributes, physical data model, and user interface.

A conceptual data model for a hotel management system depicts a high-level view of the system's data requirements, concentrating on the key entities and relationships between them. It helps in understanding the system's overall data organization and structure. The Conceptual Entity Relationship Diagram for the SHARP hotel management project captures the overall structure of organizational data. Entities are the fundamental building blocks of a database system. They stand in for unique and recognisable goods, concepts, or other entities that are used to store and manage data inside of an organisation. They act as the fundamental building blocks that organise and structure information. The following strong entities have been found within the framework of this database design; each one holds certain data points and contributes to the efficient operation of a hotel system:

Employee: Maintains and tracks information about the employer, such as roles, duties, and contact information.

Employee Type: classifies employees depending on their job duties, including front desk agents, housekeepers, food and beverage staff, and managers.

Housekeeping: Organizes and keeps an eye on the hotel's housekeeping activities and duties.

Guest: The term "guest" refers to those who are staying at the hotel or using its amenities.

Feedback: Gathers opinions and remarks from guests regarding their stay.

Card: Represents the credit or debit cards that visitors use to make purchases.

Payment: Keeps track of all financial transactions involving payments from visitors.

Booking: This term refers to hotel reservations made by guests.

Room: Refers to individual hotel rooms, each of which has unique characteristics.

Room Type: Shows the different categories or types of rooms the hotel offers.

Check-in/check-out: A feature related to the interaction between staff and reservations.

State\_code: An object that includes information about a state and a country.

City\_code: Saves information about the cities of consumers.

These entities have one-to-many relationships: employee\_type – employee, employee – housekeeping, employee – feedback, guest – feedback, guest – card, guest – payment, guest – booking, room\_type – room, state\_code - city\_code, city\_code – guest, City\_code – employee. Only these two entities i.e., employee – booking, booking – room have many to many relationships.

A logical ERD is an extension of a conceptual ERD by introducing operational and transactional entities and describing their columns. This logical ER model enhances the conceptual model by describing the columns in each entity. The attributes for the entities include employee, guest, card, payment, booking, room, state\_code, and city\_code. Each attribute is single-valued, while the actual database system generates the logical data model independently. The attributes include employee\_id, phone\_number, employee\_type, feedback, housekeeping, guest, card, payment, amount, booking, room, state\_code, and city\_code. The logical data model is independent of the database system, allowing for better understanding and management of the database.

A logical ERD is an extension of a conceptual ERD, introducing operational and transactional entities and describing their columns. It enhances the conceptual model by describing the columns in each entity. Here are the attributes for the entities mentioned:

**employee:** employee\_id, employee\_name, employee\_type, hire\_date, salary− phone\_number.

**employee\_type:** employee\_type\_id, employee\_type\_title, employee\_type\_desc.

**feedback:** feedback\_id, comments, feedback\_date.

**housekeeping:** request\_id, details.

**guest:** guest\_id, guest\_fname, guest\_lname, email, phone.

**card:** card\_id, card\_number, card\_holder\_name, expiration\_date, cvv.

**payment:** payment\_id, payment\_date, payment\_method, amount.

**booking:** booking\_id, booking\_date, booking\_time, arrival\_date, departure\_date, no\_of\_adults, no\_of\_children.

**room:** room\_no, room\_type, room\_price, room\_description.

**room\_type:** room\_type\_id, room\_type, max\_occupancy, price.

**state\_code:** state\_id, state, country.

**city\_code:** city\_code, city.

With a special emphasis on user interactions and the manager's user interface, the data describes many facets of the design process for an extensive hotel management website. A guest login page, new guest account creation, guest profile excerpt, profile editing, password change, booking history, reservation creation, room offering display, offer selection, payment method choice, card addition, booking details, housekeeping request, and feedback form are just a few of the features included in the design.

Managing and viewing bookings, housekeeping requests, and feedback are among the features of the manager's interface. Overview information is available on the manager's homepage, which also allows for the addition of new staff and the viewing of feedback submissions. This methodical technique makes sure that booking, managing, and communicating inside the hotel system is done efficiently for both guests and managers.

# **IMPLEMENTATION (PHASE IV)**

## ***INTRODUCTION:***

We will go into details about the installation process for our hotel management website in this section. The conversion of the conceptual design into a practical and interactive system occurs during the implementation phase, which is an important stage. This section seeks to give a thorough explanation of how each of the components are arranged within the framework of our project. We will highlight each file's important features and explain how they contribute to a seamless user experience for both visitors and hotel staff.

## ***PROJECT STRUCTURE:***

The implementation of our hotel management website is divided into various folders, each of which addresses a different component of the system's functionality. The development team's collaboration and modularity are improved by this well-organized method. Here is a description of the high-level organisation and function of each significant folder:

**User Interface:** The files are linked to the website's user interface. It includes several HTML files, each of which controls a different user-facing page. These files specify how the front-end of the website will look, function, and interact:

* about.html: Offers details on the hotel and its amenities.
* add\_employee.html: It is the manager interface for adding new employees to the database.
* blog-standard.html: Displays postings from blogs about the hotel.
* index.html: The hotel's main landing page.
* contact.html: Users can contact the hotel using the provided contact information or the contact form on this page.
* housekeeper\_index.html: A dashboard for housekeeping personnel to supervise room cleaning.
* login.html: System login page with user authentication.
* manager\_index.html: A dashboard with administrative features for hotel managers.
* receptionist\_index.html: A receptionist home page that shows check-in/check-out details and room assignments.
* room-list.html: Lists the available rooms so that guests might select one.
* services-team.html: Displays details about the hotel's services.
* signup.html: Account registration page for new users.

Note: please note that some of the files in frontend are in the backend section. (Some files with .php extension)

**DataBase:** pables regarding database management can be found here. For executing queries for creating database connections, managing tables, and carrying out create, read, update, delete commands on data, these tables will be directly affected.

**‘guest’ table:** This table includes all the guest related information which comes from sign-up and used for login and other tasks.

**‘employee’ table:** This table includes information related to employee as their personal information and their hiring date etc. This data comes from manager dashboard where manager create a new employee, entries from this tables like email will be used for employee’s login and other tasks.

**‘room’ table:** This table includes all the room numbers linked with their room\_tye\_id’s.

**‘room\_type’ table:** This table involves the type of rooms linked with their room number, maximum occupancy, and price from where our site is fetching data to offer available rooms to customer based on guest’s search criteria. Manager can also edit specific information of a room\_type.

**‘employee\_type****’:** This table includes the type of employees in hotel. This is linked with employee\_type\_id which is used for different employees to perform their tasks.

**‘booking’** **Table:** This table contains the all the required booking information with their guest id. The entries will be created in this table when a booking will be done .it is linked with booking\_room id from where room will be updated on booking.

**‘booking\_room’ table:**  This table is linked with booking and room table where it creates link from booking to rooms, meaning rooms related with a booking will be updated.

**‘feedback’ table:** This table contains the feedback related information that is given by guests about hotel. This information will be fetched by manager to view feedbacks.

**Back-end:** The files that handle the system's logic for business and server-side operations are kept in the back-end folder. These files handle user requests, interact with the database, and help the user interface and underlying data communicate.

* add\_employee.php: Addition of new employees to the system is handled by this file.
* all\_bookings.php: this script shows all the bookings to the manager.
* booking.php: Controls room reservation procedures.
* booking\_combinations.php: It deals with room booking combinations.
* edit\_profile.php: This page enables users to change the details of their profiles.
* employee\_list.php: Retrieves and presents an employee list.
* guest\_index.php: The back-end logic for the landing page for guests.
* guest\_bookings.php: this page shows all bookings of the logged in guest to them.
* guest\_profile.php: Information about guests' profiles is managed via this file.
* login.php: User authentication backend logic.
* report.php:this file generates and displays reports to the manager.
* receptionist\_index.php: Dashboard backend logic for the receptionist.
* room\_type\_info.php: Backend logic used to present information about the type of room.
* update\_room\_type.php manages changes to the information about the room type.

This well-organized structure simplifies overall development, encourages collaboration, and makes it possible to maintain and improve our hotel management website in the future. It guarantees that team members may quickly find, alter, and expand components without compromising the functionality of the overall system.

**Other Assets:** In addition to the key components of our hotel management project, we have also made use of a variety of different resources to improve the application's general usability and attractiveness. These assets, which enhance the website's aesthetic appeal and usefulness, include photos, fonts, and other graphic components.

Logo Images: The SHARP hotel's visual depiction is represented by its logo. Its evident appearance on the website's header gives the user interface more branding and awareness.

Room Images: A collection of high-quality images showcasing the different types of hotel rooms offered. Potential guests can see a visual representation of the lodging options from these photographs, which enables them to make well-informed reservations.

Feature Icon: Icons for the different services and facilities provided by The SHARP hotel, including a gym, a library, room service, a pool, and more. The user interface makes use of these icons to graphically represent the hotel's amenities.

Font Selection: For text elements on the website, a specific font family has been used. The chosen typeface improves readability and exudes an air of sophistication and modernity along with the identity of The SHARP hotel.

Colour Palette: To keep the website's design unified and aesthetically appealing throughout, a carefully chosen colour scheme has been devised. The colours help to a cohesive user experience while reflecting the hotel's brand identity.

## ***UI IMPLEMENTATION:***

We will go into our hotel management website's user interface elements in this section. Each UI file serves a distinct purpose in shaping the visual and interactive aspects of the website. We will provide an overview of each UI file, its role in the project, and how it interacts with other parts of the system.

**about.html:**

The information about the hotel and its services is displayed in the file called about.html. It provides a visually appealing layout with text and images, enabling users to learn more about the hotel's offerings. It interacts with the backend to fetch relevant information and present it to users.

**add\_employee.html:**

The manager can add information to this page anytime he wants to add a new employee. The backend (add\_employee.php) receives the form data after submission and processes it before inserting it into the database.

**blog\_standard.html:**

This page is used for displaying the blogs about the Hotel such as the rooms that are available, the taxi fares that are available. This blog page also retrieves data from the backend using the specific file.

**housekeeper\_index.html:**

Housekeeping workers can oversee room cleaning and maintenance on this page, housekeeper\_index.html, which was created just for them. To retrieve room information and modify room statuses, it communicates with the backend.

**index.html:**

Serves as the main landing page for the hotel's website. It introduces users to the hotel's services, amenities, and provides navigation to other sections of the website. This page interacts with the backend (index.php) for displaying dynamic content.

**manager\_index.html:**

Provides a dashboard with administrative features for hotel managers. Managers can read reports, monitor activities, and make critical choices. For the purposes of managing and retrieving data, this page communicates with the backend (manager\_index.php).

**room\_list.html:**

This provides the list of the available rooms so that customers can select one during the reservation process. Depending on their preferences, users can sort and filter rooms. To retrieve and present room information, this page communicates with the backend (filter\_rooms.php).

**services\_team.html:**

It displays members of our team who made the website The SHARP hotel management. In this there is a detail about each team member along with their photos.

**signup.html:**

A registration form is made available for new users to create accounts. It gathers user information, converses with the backend, and verifies inputs before storing user data in the database.

These UI assets work together to make our hotel management website entertaining and simple to use. To make it possible to generate dynamic content, retrieve data, and engage users in conversation, they communicate with the backend elements, like PHP files.

## ***DATABASE IMPLEMENTATION:***

The database architecture of our project is a key element that stores and handles different kinds of data associated with our hotel management website. An overview of the database's structure, tables, relationships, and how our project interacts with the database will be given in this part.

**Architecture and interactions of databases:**

Using MySQL, a well-liked relational database management system, we constructed our database system.

The database houses important data like employee profiles, guest profiles, and accommodation reservations.

Back-end files used by the project run SQL queries that communicate with the database. When necessary, these queries obtain, insert, update, and delete data.

**Database Schema and Tables:**

1. **‘guest’ table:** It contains data about hotel guests. Guest ID, first and last names, emails, phone numbers, addresses in cities, states, and countries, passwords, and cpasswords are among the columns. guest\_id serves as the primary key.

guest\_id: An auto-incrementing integer that is used as the guest table's primary key. Each guest entry is given a unique identification by it.

guest\_fname: The first name of the guest is stored in a variable-length string column (VARCHAR). It can only be up to 50 characters long. Since the column supports NULL values, a value might not always be present.

guest\_lname: The guest's last name is kept in the guest\_lname column, a variable-length string (VARCHAR). It can also be up to 50 characters long and accept NULL values.

email: A VARCHAR column with a length limit of 100 characters that is used to hold the visitor's email address. It accepts NULL as a value.

phone: A VARCHAR column with a maximum character length of 20 used to store the visitor's phone number. It accepts NULL as a value.

address: The guest's address will be kept in a VARCHAR column with a maximum character length of 200. It accepts NULL as a value.

city: city is a VARCHAR column with a 100-character maximum length that is used to hold the guest's home city. It accepts NULL as a value.

state: state is a VARCHAR column with a 100-character maximum length that is used to store the guest's home state. It accepts NULL as a value.

zip: It is a VARCHAR column with a 20-character maximum length that is used to hold the zip code of the visitor's residence. It accepts NULL as a value.

country: country is a VARCHAR column that can contain up to 100 characters and is used to store the visitor's home country. It accepts NULL as a value.

password: A VARCHAR column with a length limit of 255 characters that is used to record the guest's hashed password. Because this field is set as NOT NULL, a password value is required at all times.

cpassword: A VARCHAR column with a length limit of 255 characters is designated as cpassword and is used to store the hashed confirmation password. This column is designated as NOT NULL, much like password.

1. ‘**room\_type’ table:** In this the types of rooms available at the hotel are represented. There are four types of room in the hotel. Following are the columns of room\_type table:

room\_type\_id: Each room type's unique identity is kept in this column. It has a maximum length of 20 characters and is of the VARCHAR type. It functions as the table's main key and cannot be null.

occupancy: The maximum occupancy for the room type is kept in this column. It cannot be null because it is of type INT.

room\_price: The cost of the room type is kept in this column. It has a scale of 2 and is of the DECIMAL type, indicating that it has up to 8 digits before the decimal point and 2 digits after. It has a default value of NULL.

room\_description: The room type description is kept in this column. It is of the TEXT type, allowing for larger amounts of text to be stored. It also has a default value of NULL.

1. **‘room’ table:** Individual hotel rooms are represented by the room table. Following are the columns of room table:

room\_no: An automatically increasing integer column that acts as the room table's main key. Each room is identified separately by it.

Room type ID is kept in the string column room\_type\_id, which is part of the room's data structure. It can only be up to 20 characters long.

Based on the room\_type\_id column, the room table creates a foreign key relationship with the room\_type table. The fk\_room\_type\_id foreign key constraint establishes this link. The room table's room\_type\_id must relate to a valid room\_type\_id in the room\_type table for the foreign key relationship to be legitimate.

1. **‘employee’ table:** This table contains information about hotel employees. Below are the column:

employee\_id: An automatically increasing integer column that acts as the employee table's primary key. Each employee is identified individually.

employee\_fname: A string column that contains the employee's first name. It has a NOT NULL designation.

employee\_lname is a string column that holds the employee's last name. It has a NOT NULL designation.

gender: The employee's gender (M or F) is kept in a character column. It has a NOT NULL designation.

employee\_type\_id: An employee type ID is stored in a string column called employee\_type\_id. It establishes a connection with the employee\_type table and is flagged as NOT NULL.

hire\_date: A date column used to record the hiring date of a new employee. It is flagged as NOT NULL and has the current timestamp as its default value.

end\_date: A date column used to record an employee's termination date. It accepts NULL as a value.

salary: The employee's salary is kept in a decimal column. It accepts NULL as a value.

address: The employee's address is kept in a string column. It accepts NULL as a value.

phone\_number: A string column that contains the employee's phone number. It accepts NULL as a value.

email: An employee's email address is kept in a string column called "email." It has a NOT NULL designation.

city: A string column that contains the employee's home city. It has a NOT NULL designation.

state: A string column that records the employee's home state. It has a NOT NULL designation.

zip: A string column that contains the employee's home zip code. It has a NOT NULL designation.

country: A string column that records the employee's home nation. It has a NOT NULL designation.

password: A string column that contains the employee's hashed password. It has a NOT NULL designation.

cpassword: A string column used to record the hashed confirmation password. It has a NOT NULL designation.

Based on the employee\_type\_id field, the employee table creates a foreign key relationship with the employee\_type table. Employee\_employee\_type\_id\_fkey is a foreign key constraint that describes this relationship. The employee\_type\_id in the employee table must correspond to a valid employee\_type\_id in the employee\_type table, and this is ensured by the foreign key relationship.

1. **‘employee\_type’ table:** It stores information about different types of employees in the hotel. Columns included in this table:

employee\_type\_id: A string column that acts as the employee\_type table's primary key. Each employee kind is identified specifically by it.

employee\_type\_title: A string column used to hold the employee type's title. It can be up to 40 characters long and accepts NULL values.

employee\_type\_desc: A text column used to hold an employee type description. It accepts NULL as a value.

Based on the employee\_type\_id column, the employee table (if defined in your database) might be able to connect to the employee\_type table. Each employee would be connected to a certain employee category thanks to this relationship.

1. **‘booking’ table:** Information regarding hotel reservations made by visitors is kept in the booking table. Columns included in this table:

booking\_id: An auto-incrementing integer column that functions as the booking table's primary key. It gives each booking entry a special identification.

booking\_date\_time: a datetime column that keeps track of the time and date the reservation was made. The timestamp as of now serves as the default.

intended\_check\_in: This date field contains the reservation's planned check-in date. It has a NOT NULL designation.

intended\_check\_out: This date column contains the reservation's planned check-out date. It has a NOT NULL designation.

check\_in: The actual check-in date and time for the reservation are stored in the check\_in datetime field. It accepts NULL as a value.

check\_out: The actual check-out date and time for the reservation are kept in the check\_out datetime column. It accepts NULL as a value.

no\_of\_adults: An integer column that contains the total number of adults related to the reservation. It accepts NULL as a value.

guest\_id: An integer column that serves as a pointer to the visitor who made the reservation. It creates a connection to the guest table and accepts NULL data.

Based on the guest\_id field, the booking table creates a foreign key relationship with the guest table. The booking\_guest\_id\_fkey foreign key restriction establishes this link. The guest\_id in the booking table refers a legitimate guest\_id in the guest table thanks to the foreign key relationship.

1. **‘booking\_room’ table:** This table provides the details of rooms booked to the users such as room booking no., booking id and room no. This table is illustrated with the following structure of SQL:

booking\_room\_id: This column has an 11-digit length and has an INT data type. It serves as the table's main key and is set to auto-increment, which means a new value will be created automatically for each new row.

booking\_id: This column has an 11-digit length and is an INT data type. It saves the booking ID connected to the room.

room\_no: This column has an 11-digit length and is an INT data type. It keeps track of the reserved room's room number.

Here are the following foreign keys that are used in this table:

fk\_booking\_id: The constraint fk\_booking\_id creates a foreign key relationship between the booking\_id column of the booking\_room table and the booking\_id column of the booking table. This guarantees that a booking room access is linked to a legitimate booking.

fk\_room\_no: This constraint creates a foreign key relationship between the room\_no column of the booking\_room table and the room table. This guarantees that a booking room entry is linked to a legitimate room.

The DEFAULT CHARSET specification specifies the character set used for the table as utf8mb4 for compatibility with a wide variety of characters, and the ENGINE specification specifies InnoDB as the storage engine for the table.

## ***BACK-END IMPLEMENTATION:***

The back-end logic of our hotel management website consists of a wide range of activities and functionalities that are essential to the efficient running of the system. In this section, we'll examine each back-end file's function as well as how it interacts with the user interface (UI) and the database.

**1.** **add\_employee.php:**

Function: Manages the system's ability to add new employees.

User interface interaction: Gathers employee information from a form on the UI.

Database interaction: Adds new employee data to the database.

**2. booking.php:**

Purpose: Manages the process of booking a room.

Interaction with UI: Booking information is received.

Database interaction: Database entry of booking information.

**3. booking\_combinations.php:**

Function: Handles booking combinations, like bookings for a room.

UI Interaction: Coordinates and manages combined booking requests received from the UI.

Database interaction: Stores combined booking data in the database.

**4. edit\_profile.php:**

Purpose: It enable users to change the details of their profile.

Interaction with UI: Accepts updated profile data from the UI.

Interaction with Database: Updates the user's profile information in the database.

**5. employee\_list.php:**

Uses a list of employees to be retrieved and displayed.

User interface interaction: Gives the UI a list of employees to display.

Database interaction: Gets employee information out of the database.

Database interaction: Uses a query to find rooms that meet the given requirements.

**6. guest\_index.php:**

Purpose: Backend reasoning behind the landing page for visitors.

Interaction with UI: Retrieving pertinent information to fill the visitor landing page.

Database interaction: Obtains the required data from the database.

**7. guest\_profile.php:**

Purpose: Managing guest profile data.

UI Interaction: Giving the UI access to visitor profile information so it may be shown and edited.

Database interaction: Accesses and modifies the database's visitor profile information.

**8. login.php:**

Function: User authentication backend logic.

UI Interaction: Receives user login credentials from the UI during interaction.

Database interaction: Checks user credentials against the database.

**9. receptionist\_index.php:**

Function: Backend logic for the dashboard of the receptionist.

Interaction with UI: Gets information for the receptionist's dashboard and processes it.

Interaction with the database: Gets relevant data out of the database and updates it.

**10. room\_type\_info.php:**

Function: Backend logic for presenting room type information. Shows different types of room type such as double, king, queen and penthouse.

UI interaction: Gives the UI information about the type of room.

Database interaction: Gets information about the type of room from the database.

**11. update\_room\_type.php:**

Function: Manages updates to the information about the room type.

UI Interaction: Receives updated information on room type.

Database interaction: Modifies the database's entry for the room type.

**12. all\_booking.php:**

Function: Manages and retrieves data regarding all reservations made at the hotel.

UI Interaction: Receives requests from the user interface to provide a list of all reservations.

Database Interaction: Obtaining booking information from the database, processing it, and producing the necessary data.

**13. feedback.php:**

Function: Oversees the submission and storing of guest feedback forms.

UI Interaction: Data from the UI's submitted feedback form is received.

Database Interaction: Enters the feedback information into the database for analysis and record-keeping.

**14. guest\_bookings.php:**

Function: Manages and gives data from the database about visitor reservations for the UI.

UI Interaction: Receives requests from guests or authorised users to reveal their booking histories.

Database Interaction: By using the guest's identification, the database's booking information is retrieved.

**15. report.php:**

Purpose: Generates reports and analytics data.

UI Interaction: Its functionality includes gathering data from the database, processing it, and producing reports for authorised users.

Database Interaction: Responds to UI queries for report generation and talks with the database to retrieve the required data.

**16. view\_rooms.php:**

Function: Displays the data about the rooms such as room type, room no., occupancy, price and description of the rooms.

UI Interaction: Displays room information such as room type, room no., occupancy, price and description of the rooms by retrieving it from the database.

Database Interaction: Responds to requests from the user interface (UI) to display room details and contacts the database to retrieve room data.

**17. viewfeedbacks.php:**

Function: Displays comments made by visitors.

UI Interaction: Presents feedback data on the UI after retrieving it from the database.

Database Interaction: Communicates with the database to retrieve feedback data and interacts with the user interface to reply to requests for seeing feedback entries.

**18. viewguest.php:**

Function: Displays guest details and booking history.

UI Interaction: Takes data from the database and displays it on the user interface, including guest information and booking details.

Database Interaction: handles UI requests to display visitor details and interacts with the database to retrieve pertinent information.

The functionality of the Hotel Management Website is impacted by each of these back-end files in a different way. They handle interactions, process data, and make sure the database and user interface are always in sync. Our system's performance and user experience can be successfully maintained, enhanced, and improved by developers by knowing the function and interactions of these files.

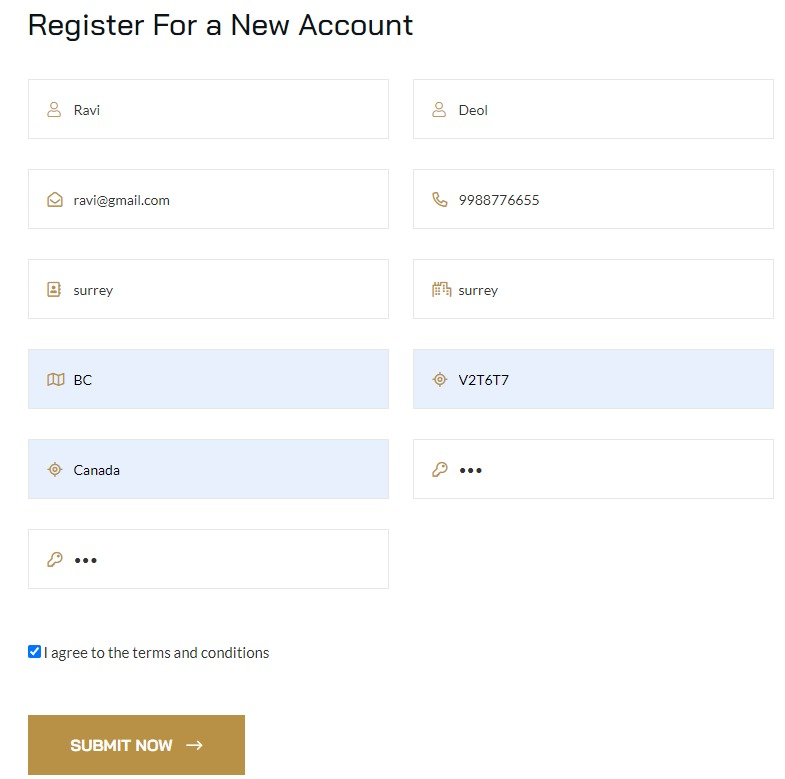
The development tools, languages, frameworks, and libraries utilised in various parts of our website for hotel management are listed in the following table:

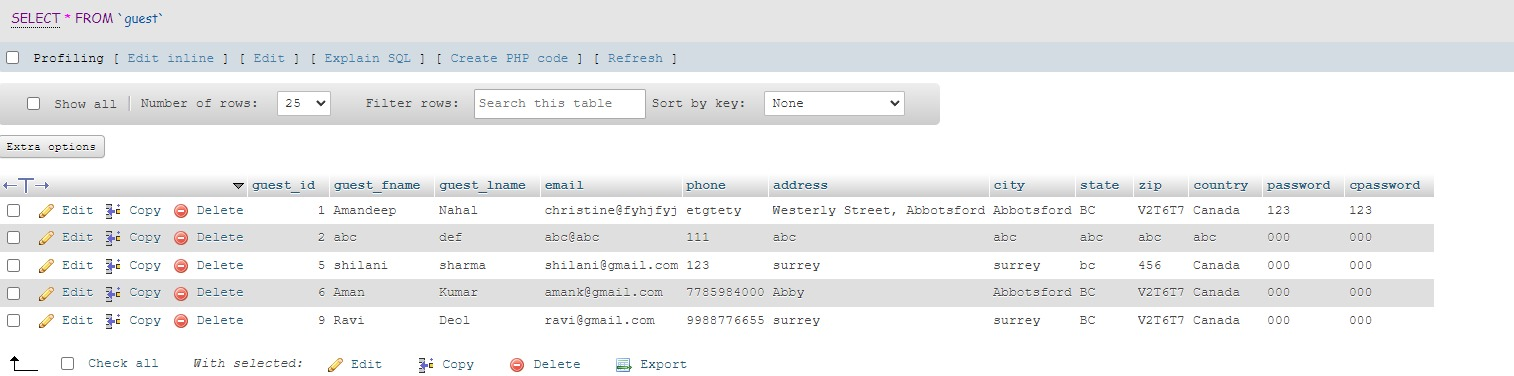
We used XAMPP for connection.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Development tools | Languages | Frameworks and libraries |
| User-interface | Visual studio code | HTML, CSS, JavaScript | Bootstrap, jQuery |
| Back-end | Visual studio code | PHP |  |
| Database | Xampp | SQL |  |

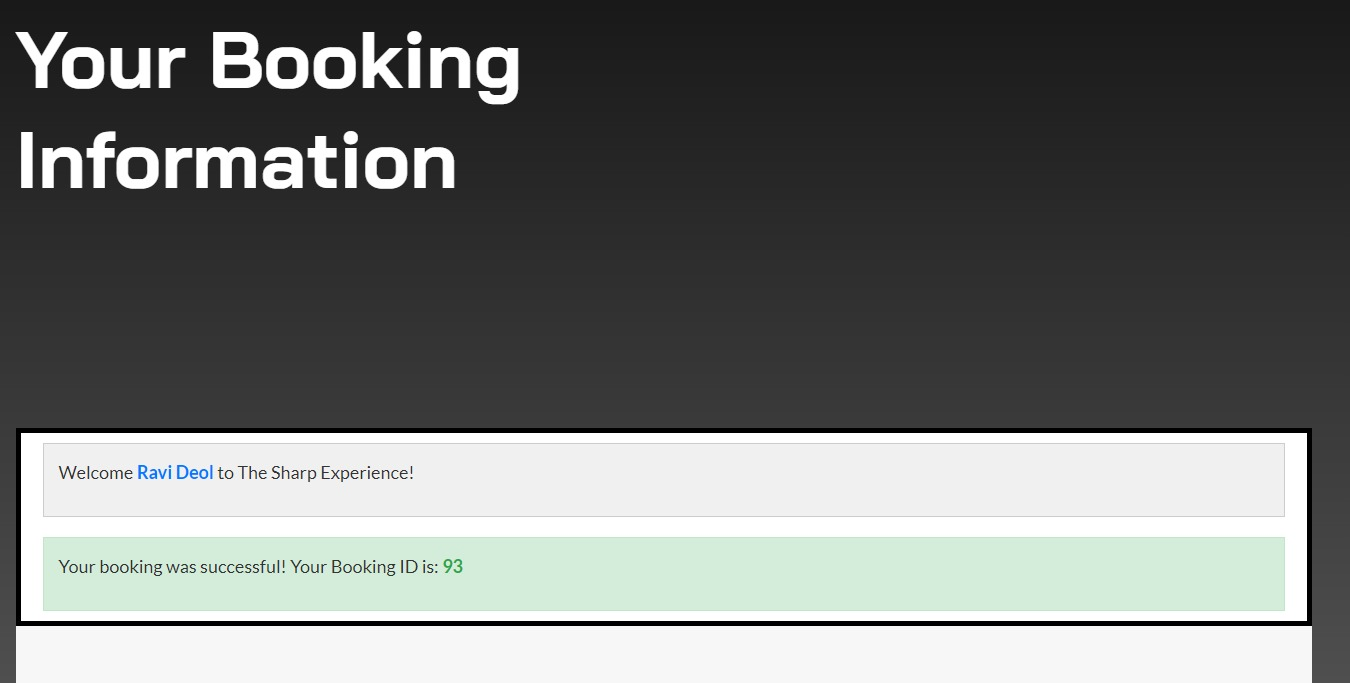
# **TESTING:**

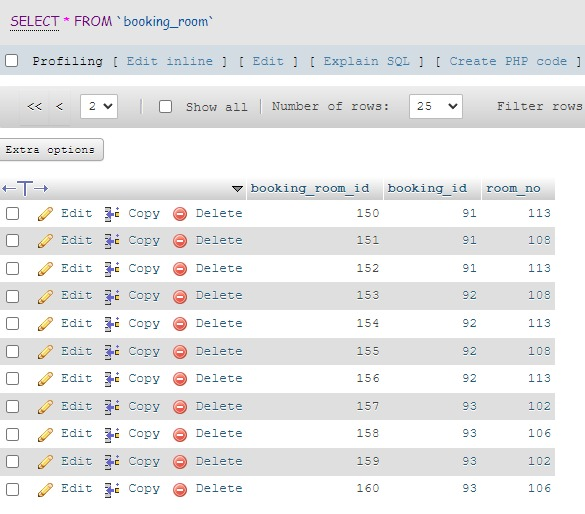
Creating a guest account and verifying it in the database:



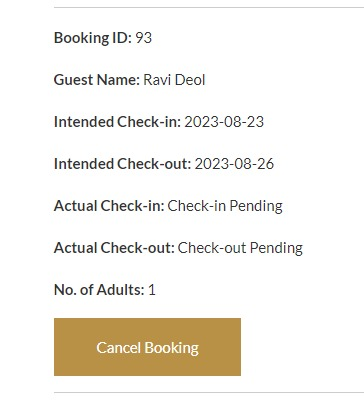


Making a booking and verifying in database:





Cancelling a booking:



# **USER MANUAL**

1. Introduction

Welcome to the SHARP Hotel Management Project User Manual! This guide is designed to provide you with a comprehensive understanding of how to navigate and utilize the features of our hotel management system. whether you're a guest trying to make a reservation, a registered user managing your reservations, or someone looking for information about our hotel facilities and services.

2. User Registration and Login

Creating an Account

To create an account, click on the "Sign Up" or "Register" button. Provide your personal details, such as your name, email address, and password. After successfully registering, you can log in with your credentials.

Logging In:

If you already have an account, click on the "Log In" button. Enter your registered email and password to access your account.

**3. Guest Profile**

Viewing Profile Information:

After logging in, you'll be directed to your profile page. Here, you can view your personal information, including your name, email address, and contact details.

Editing Personal Information:

If your details change, you can edit your profile information. Click on the "Edit Personal Information" button, make the necessary changes, and save your updated details.

**4. Booking a Hotel Room**

Exploring Available Rooms:

Navigate to the "Booking" section to explore available hotel rooms. Filter rooms based on your preferences, such as check-in and check-out dates, number of adults, and room type.

Selecting Dates and Preferences:

Choose your desired check-in and check-out dates, specify the number of adults, and select room preferences. The available rooms will be displayed based on your selections.

Confirming Booking:

Select your preferred room and review the booking details. Once satisfied, proceed to confirm your booking. You may need to provide payment details to secure your reservation.

**5. Managing Bookings**

Viewing Your Bookings

In the "Guest Bookings" section, you can view your booked hotel rooms. Information such as booking ID, check-in and check-out dates, and booking status will be displayed.

Cancelling Bookings

For bookings with pending check-in dates, you'll have the option to cancel the booking. Click on the "Cancel Booking" button to cancel a booking. Note that cancellation policies may apply.

**6. Hotel Information**

Browsing Hotel Amenities:

Explore the amenities offered by each hotel. From swimming pools to fitness gyms, you can learn about the facilities available for your comfort and enjoyment.

Viewing Location Details:

Get information about the hotel's location, including its address and contact details. Use this information to plan your trip and navigate to the hotel.

**7. Contact and Support:**

Getting in Touch with Support:

If you have questions or need assistance, you can contact our support team. Use the provided contact information to reach out for help.

Frequently Asked Questions:

Check the FAQ section for answers to common queries. This section may address topics related to booking, payment, amenities, and more.

**8. Logout**

Signing Out:

When you're finished using the website, remember to log out of your account to ensure your information remains secure.

# **TOOLS**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tool / platform name | Your review for the tool. | Would you use it in the future again? |
| Communication among team members | Whatsapp, Email | Efficient for real-time team communication. | YES |
| Project management |  |  |  |
| Issue / task management |  |  |  |
| File sharing | Google Drive | Easy document sharing and collaboration. | YES |
| Other | Zoom | Reliable for virtual communication. | YES |

# REFERENCES

- Template. (n.d.). Retrieved from [Main Menu Template](https://colorlib.mystagingwebsite.com/download/)

- Hernansartorio. (n.d.). jQuery Nice Select. Retrieved from [jQuery Nice Select](https://github.com/hernansartorio/jquery-nice-select)

- Swiper. (n.d.). Retrieved from [Swiper](https://swiperjs.com)

- The MIT License. (n.d.). Retrieved from [Popper](http://opensource.org/licenses/MIT)

- Imakewebthings. (n.d.). Waypoints. Retrieved from [Waypoints](https://github.com/imakewebthings/waypoints/tree/master/lib)

- Hernansartorio. (n.d.). jQuery Nice Select. Retrieved from [jQuery Nice Select](https://github.com/hernansartorio/jquery-nice-select)

- Dimsemenov. (n.d.). Magnific Popup. Retrieved from [Magnific Popup](http://dimsemenov.com/plugins/magnific-popup/)

- OpenJS Foundation. (2023). jQuery v3.6.0. Retrieved from [jQuery](https://jquery.org/license)

- Isotope by Metafizzy. (n.d.). Retrieved from [Isotope](https://isotope.metafizzy.co)

- The Bootstrap Authors. (2023). Bootstrap v5.1.3. Retrieved from [Bootstrap](https://getbootstrap.com/)

- The Bootstrap Authors. (2023). Bootstrap License. Retrieved from [MIT License](https://github.com/twbs/bootstrap/blob/main/LICENSE)

Google Fonts. (n.d.). Roboto Font. Retrieved from [Roboto](https://fonts.googleapis.com/css2?family=Roboto&display=swap)

Google Fonts. (n.d.). Open Sans Font. Retrieved from [Open Sans](https://fonts.googleapis.com/css2?family=Open+Sans&display=swap)

Google Fonts. (n.d.). Lato Font. Retrieved from [Lato](https://fonts.googleapis.com/css2?family=Lato&display=swap)

Google Fonts. (n.d.). Montserrat Font. Retrieved from [Montserrat](https://fonts.googleapis.com/css2?family=Montserrat&display=swap)

Google Fonts. (n.d.). Poppins Font. Retrieved from [Poppins](https://fonts.googleapis.com/css2?family=Poppins&display=swap)

Google Fonts. (n.d.). Raleway Font. Retrieved from [Raleway](https://fonts.googleapis.com/css2?family=Raleway&display=swap)

Google Fonts. (n.d.). Nunito Font. Retrieved from [Nunito](https://fonts.googleapis.com/css2?family=Nunito&display=swap)

Google Fonts. (n.d.). Ubuntu Font. Retrieved from [Ubuntu](https://fonts.googleapis.com/css2?family=Ubuntu&display=swap)

Google Fonts. (n.d.). Playfair Display Font. Retrieved from [Playfair Display](https://fonts.googleapis.com/css2?family=Playfair+Display&display=swap)