## Chapter 1

* 1. **Overview**

# Introduction

The COVID Bed Slot Booking System is a mini project designed to assist hospitals in managing bed availability and reservation during pandemic conditions like COVID-19. The system aims to reduce manual efforts, avoid overbooking, and improve patient care efficiency. It integrates hospital resource data and enables real-time slot booking through a user-friendly interface.

Developed using **HTML/CSS** for the frontend and **Python with MySQL** for the backend, the system provides an interactive platform for hospital administrators to monitor, update, and allocate bed slots effectively.

## Problem Statement

During pandemics, hospitals face high demand for beds, making manual tracking ineffective Overbooking or miscommunication can lead to patient dissatisfaction or life-threatening delays. There is a need for a reliable, automated system that allows hospitals to track COVID bed availability and allow quick, secure bookings.

This project aims to:

* Enable real-time bed slot bookings and cancellations.
* Maintain records of patients, beds, and allocations.
* Allow secure login for hospital staff.
* Improve coordination and reduce administrative load.

## Database Management System

## A database management system (DBMS) is a system software for creating and managing databases. A DBMS makes it possible for end users to create, protect, read, update, and delete data in a database. The most prevalent type of data management platform, the DBMS essentially serves as an interface between databases and users or application programs, ensuring that data is consistently organized and remains easily accessible.

## SQL

## SQL (Structured Query Language) is a standard programming language used to manage and manipulate relational databases. It allows users to perform tasks such as retrieving, inserting, updating, and deleting data from a database. SQL is used to define the structure of database tables and establish relationships between them. It provides commands like SELECT, INSERT, UPDATE, DELETE, and CREATE for various operations. SQL is widely used in applications involving data storage and processing. It is supported by most relational database systems like MySQL, Oracle, and PostgreSQL. SQL is simple to learn yet powerful enough for advanced data manipulation.

## PYTHON/MySQL

1. PYTHON

Visual Studio Code (VS Code) is a free, lightweight, and powerful source code editor developed by Microsoft used primarily for programming in Python, but it also supports other languages like JavaScript, C++, Java, and PHP through plugins. It helps developers write, test, and debug code efficiently with features like code suggestions, error highlighting, and project management tools.

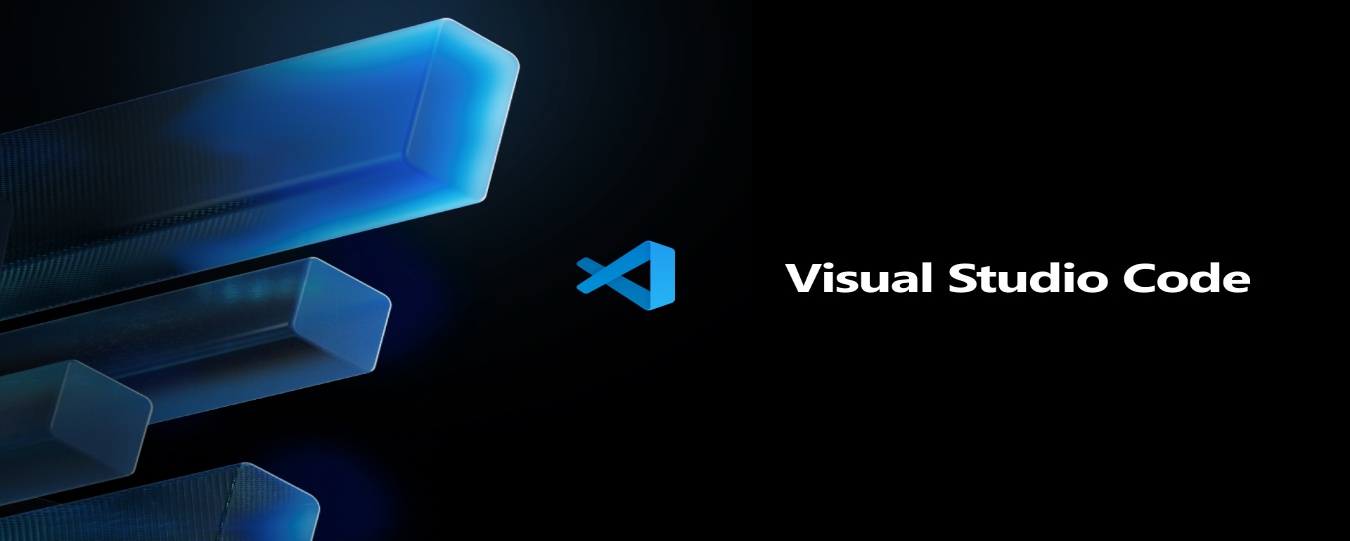


Fig 1.5.1 Visual Studio Code(VS Code)

1. MySQL

MySQL is an open-source Relational Database Management System(RDBMS) that uses SQL (Structured Query Language) to manage data. It stores information in tables and is widely used in web applications for storing structured data like user profiles, medical records, orders, etc. MySQL is developed and supported by Oracle Corporation.

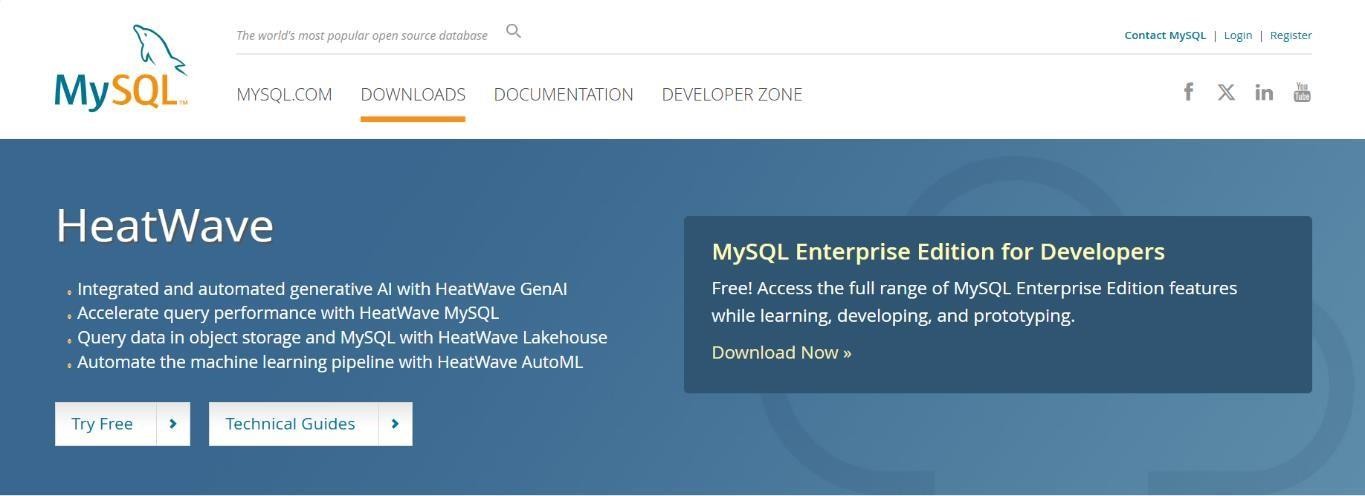


Fig 1.5.2 MySQL

**Chapter 2**

# REQUIREMENTS SPECIFICATION

## Overall Description

**The Covid Bed Slot Booking System** is a Python-based desktop application designed to simplify and digitalize the key operations of a hospital. The main objective of this application is to build an online bed slot booking platform, which allows a user to book a bed in a preferred hospital looking at the number of beds available.

This includes registration of patients, storing their details into the system, and also computerized billing for beds in the hospitals. The software has the facility to give a unique id(srfid) for every patient and stores the details of every patient automatically. It includes a facility to know the status of availability of beds at each listed hospital.

The Covid Bed Slot Booking System can be entered using a username and password. It is only accessible to admin. Only they can add data into the database. The data can be retrieved easily The interface is very user- friendly. The data is well protected for personal use and makes the data processing extremely fast.

## Specific Requirements

* + - **Login Module** : Authenticates users using credentials stored in the database
    - **Hospital Data Management**: Allows adding, viewing, and deleting Hospital records.
    - **Patient Management**: Allows adding, viewing, and deleting patient records like SRFID, E-Mail, Date of Birth
    - **Bed Booking**: Books Required type of Beds for patients with specific Oxygen Level
    - **Database Integration**: All data is securely stored and managed in MySQL using XAMPP Server

## 2.3 Software Requirements

* + - **Operating System**: Windows 11
    - **Programming Language**: Python(Version 3.13.3)
    - **IDE**: VS Code
    - **Database**: MySQL Server
    - **Database Management Tool**: MySQL Workbench
    - **Database Driver**: MySQL-Connector-Python using XAMPP Server
    - **GUI Framework**: Python
    - **Build Tool**: Not required

## 2.4 Hardware Requirements

* + - **Processor**: AM AMD Ryzen 7 4800H Processor (8-core/16-thread, 12MB Cache, 4.2 GHz max boost)
    - **RAM**: 16.00 GB
    - **Storage**:512 GB
    - **Internet Connection**: Required only for downloading tools / dependencies
    - **Graphics:** NVIDIA GeForce RTX 3050 4GB GDDR6 VRAM with up to 1600Mhz at 60W (75W with Dynamic Boost)
    - **Operating System:** Windows 11
    - **Display**: 15.6-inch (39.62 cms) FHD (1920 x 1080) 16:9, 250nits, 144Hz Refresh rate.

## Chapter 3

**3.1 System Design**

# DETAILED DESIGN

The Covid Bed Slot Booking System includes registration of patients, storing their details into the system, and also computerized billing for beds in the hospitals. The software has the facility to give a unique id(srfid) for every patient and stores the details of every patient automatically. It includes a facility to know the current status of availability of beds at each listed hospital.

The Covid Bed Slot Booking System can be entered using a username and password. It is only accessible to admin. Only they can add data into the database. The data can be retrieved easily. The interface is very user- friendly. The data is well protected for personal use and makes the data processing very fast.

The Covid Bed Slot Booking System is powerful, flexible, and easy to use and is designed and developed to deliver real conceivable benefits to hospitals and patients.

Bed Slot Booking System is designed for multispecialty hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting, in a seamless flow.

Bed Slot Booking System is a software product suite designed to improve the quality and management of hospitals in the areas of clinical process analysis and activity-based costing. It enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital and helps you manage your processes.

## Entity Relationship Diagram

## 

Fig 3.2.1 ER Diagram

### Relational Schema

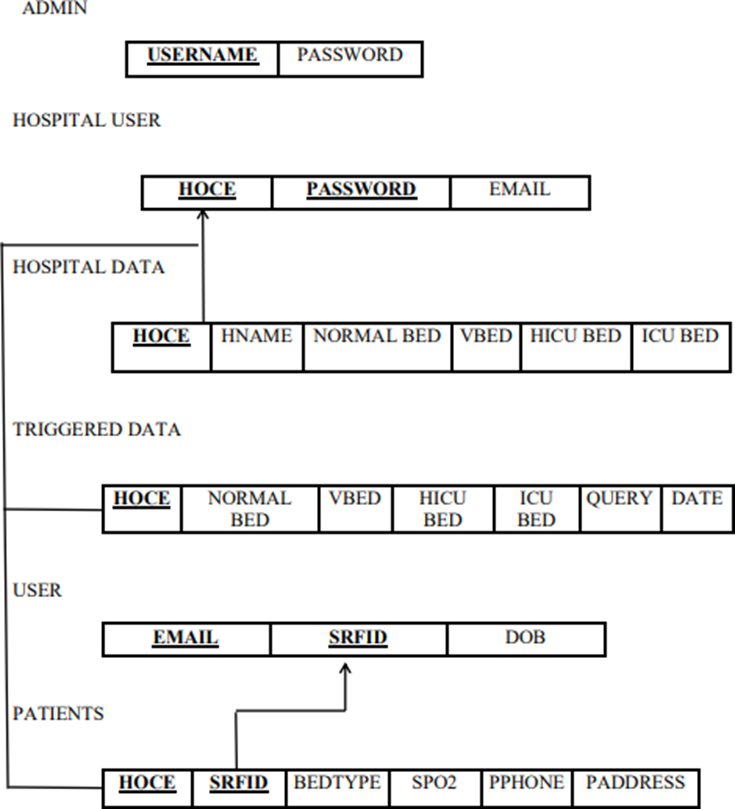
****

Fig 3.2.2 Relational Schema

## Description of Tables

1. Booking Patient

Books the required type of Bed for patients.

Columns:

* PID (int, Primary key)
* SRFID (varchar)
* Bed Type (varchar)
* Hcode(varchar)
* Spo2(int)
* Pname(varchar)
* Pphone(varchar)
* Paddress(varchar)

1. Hospital Data

Stores the Bed details in Hospital Columns:

* HDID (int, Primary key)
* Hcode (int)
* Hname (varchar)
* Normalbed(varchar)
* HICUbed(varchar)
* ICUbed(varchar)
* Vbed(varchar)

1. Hospital User

Stores Hospital user information. Columns:

* HUID (int, Primary key)
* Hcode(varchar)
* Email (varchar)
* Password(varchar)

1. Triggered Data

Stores updated information.

Columns:

* TDID (int, Primary key)
* Hcode (varchar)
* Normalbed (int)
* HICUbed (int)
* ICUbed(int)
* Vbed (int)
* Querys(varchar)
* Date (int)

1. User

Stores Users information.

Columns:

* UID (int, primary key)
* SRFID (varchar)
* E-mail(varchar)
* DOB (varchar)

**Chapter 4**

# IMPLEMENTATION

## Module and Their Rules

### 1. Admin module

In this module Admin can add list of hospitals and send email to the hospital containing hospital code and password to login. Admin adds various hospital data using his credentials Only Admin has the right to add hospital lists. Admin module has several sub-modules in it. They are:

* **Login**: Admin uses user\_id and password to login.
* **Add Hospital user**: Adds hospital user consisting of email, hospital code, password.
* **Logout**: Admin logs out of the page.

### 2.Hospital module

In Hospital module, listed hospitals can add hospital details like various beds (ICU, Normal, Ventilator etc..) available in their hospital. The entered details can be modified (update, delete) and can be viewed using triggered data section. If invalid credentials are entered error message is displayed. Hospital module has several sub-modules in it.

They are:

* **Login**: Hospital user uses email id and password to login.
* **Add Hospital Data**: Adds hospital data consisting of hospital bed details.
* **View Triggered data**: Hospital user can view triggered data like update, insert, delete operations.
* **Logout**: Hospital user logs out of the page.

### 3.Patient module

Patients can login/sign-in using unique SRFID (given for covid +ve patients) and email. Based on available beds in various hospitals they can book the type of bed they want. The details entered by the patients can be seen in patient detail section. Hospital module has several sub-modules in it.

They are:

* **Login**: User uses srf\_id and date-of-birth to login.
* B**ooking Bed Slot**: User can book slot among the available beds in the listed hospital.
* **View Available beds**: View the beds available in the listed hospital.
* **Patient Details**: User can view the entered patient details.
* **Logout**: Patient logs out of the page.

## Triggers and Stored Procedures

Triggers in the COVID Bed Slot Booking System are employed to enforce business rules automatically and ensure data integrity in response to specific database actions such as insertions, updates, or deletions.

One key trigger implemented is the **“after delete on patients”** trigger, which automatically logs the details of any deleted patient record into a separate **patient\_deletion\_log** table. This functionality is crucial for maintaining a history of patient data, supporting audit trails, and enabling future recovery or analysis of deleted records.

Another essential trigger is the **“prevent double booking”** trigger. This is a **“before insert”** trigger on the **bed\_bookings** table. It checks whether a bed is already booked for the specified hospital, date, and time slot. If a conflict is detected—i.e., the same bed is already reserved during that time—the trigger cancels the insertion and raises an error. This helps avoid overbooking and ensures fair allocation of critical resources, especially during peak COVID periods.

Stored procedures in the system help modularize and standardize recurring database operations, reducing code duplication and enhancing maintainability.

For instance, the **Add Bed Booking** stored procedure takes patient details, hospital ID, bed type, and booking date as input parameters and inserts the booking into the **bed\_bookings** table only after verifying bed availability. This encapsulates complex logic in a single call and ensures consistency in how bookings are made.

Another stored procedure, **Get Available Beds By Hospital**, is used to retrieve all currently unbooked beds in a given hospital. It queries the **beds** table and filters out beds that are already booked for the specified time window. This procedure simplifies the frontend logic by returning just the relevant data for slot selection, thereby improving performance and reducing database load.

By leveraging triggers and stored procedures, the COVID Bed Slot Booking System automates critical checks, enforces booking rules, and optimizes performance, all while reducing the need for repetitive SQL logic in the backend application code.

## Result

The successful development and implementation of the COVID Bed Slot Booking System has resulted in a functional, user-friendly software solution designed to manage critical healthcare resources efficiently during the pandemic. The system allows hospital administrators and staff to securely log in, manage bed inventories, handle patient bookings, and monitor slot availability in real-time.

By integrating a Python-based backend with a MySQL database and a responsive HTML/CSS frontend, the application ensures data consistency, up-to-date availability, and ease of use. The system has significantly streamlined the bed allocation process by reducing manual effort, preventing double bookings through automated trigger-based checks, and enabling fast retrieval of bed and patient records.

Features such as validation rules, database-level enforcement of business logic via triggers, and reusable stored procedures contribute to the system’s reliability and maintainability. Administrators can now perform tasks such as:

* Adding or removing bed records
* Booking and cancelling COVID patient bed slots
* Viewing available beds by hospital and bed type
* Reviewing historical booking data for audit or analysis

All of these operations are accessible from a centralized, easy-to-navigate dashboard.

Overall, the system achieves its primary goal of digitizing and automating the COVID bed allocation process, improving operational efficiency, reducing errors, and ensuring fair and effective use of limited healthcare resources.

**Chapter 5**

* 1. **Software Testing**

# Testing

Software testing was conducted throughout the development of the COVID Bed Slot Booking System to ensure that each module functioned as intended and that the system performed reliably under various scenarios. Both manual and functional testing approaches were used to verify the core functionalities, including user login, patient and bed management, and slot booking operations.

The **Login Module** was tested with valid and invalid credentials to ensure proper authentication and redirection. Error handling was verified by entering incorrect login details and leaving input fields blank to check for appropriate error messages and system behaviour.

The **Patient and Bed Management Modules** underwent input validation testing, such as detecting empty fields, verifying correct data formats (e.g., phone numbers), and preventing duplicate entries. Tests were also conducted to ensure that beds marked as “occupied” could not be reassigned without first being released or updated.

The **Bed Booking Module** was tested for multiple scenarios, including booking with existing and non-existent patient IDs, ensuring referential integrity with foreign key constraints. Double-booking tests were carried out to confirm that triggers correctly prevented the same bed from being reserved more than once for overlapping times.

The **View Bookings Module** was tested for accuracy in displaying patient and bed assignment information, correct date-based sorting, and responsiveness. Search and filter functionalities (if implemented) were verified to help users easily locate specific bookings.

Integration testing validated smooth data flow between modules, such as ensuring beds can only be booked for existing patients and available slots. Backend operations including insertions, updates, deletions, and joins were tested to confirm consistent and accurate data handling.

Overall, the testing phase confirmed that the system met its functional requirements, gracefully handled invalid input, and maintained reliable data consistency across all modules.

* 1. **Module Testing and Integration**

Module (unit) testing was conducted on individual components of the COVID Bed Slot Booking System to detect and correct issues early in development. Each module was verified to operate correctly in isolation before integration with other parts of the system.

The **Login Module** was tested to confirm that only valid credentials granted access. Validation of user input ensured that blank or incorrect fields were handled gracefully with proper feedback messages.

The **Manage Patient Module** was tested to verify proper data entry and validation rules. Checks included mandatory field completion, duplicate contact numbers, and age restrictions. Invalid or incomplete data was appropriately rejected.

The **Manage Bed Module** was tested to confirm that bed details such as bed ID, type, and availability status were correctly stored. Attempts to add duplicate beds or update occupied beds without proper release procedures were prevented.

The **Slot Booking Module** was tested by booking available beds for existing patients. Invalid patient IDs and already-occupied beds were tested to verify that foreign key constraints and triggers functioned correctly, especially in preventing double-booking.

The **View Bookings Module** was tested to ensure real-time data retrieval and display. Queries filtered by hospital, patient, or date were verified to show accurate results and reflect changes immediately.

Each module demonstrated robust handling of valid and invalid user inputs, confirming stability and usability.

Integration testing ensured seamless interaction between modules. For instance, only patients registered in the system could book beds, and any updates to patient or bed information were reflected in active bookings. Attempts to delete patients with active bookings were blocked, validating the enforcement of relational constraints.

The integration of the login module with the main dashboard was tested for secure access. Transitions between modules post-login were seamless and maintained session consistency.

Real-time communication between the Python backend and MySQL database was verified during integration, ensuring that all SQL operations executed correctly and returned expected outcomes.

Together, module and integration testing confirmed that the COVID Bed Slot Booking System is a reliable, well-integrated application capable of managing critical hospital operations efficiently.

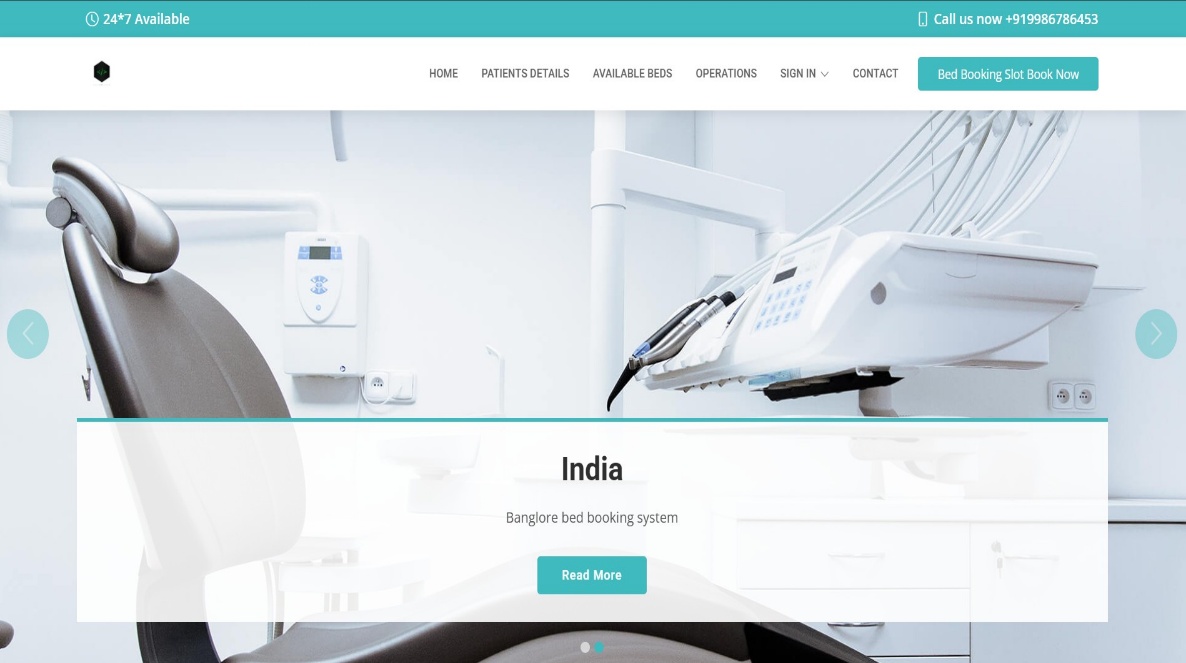
## Limitations

The current version of the COVID Bed Slot Booking System is designed for small to medium-scale hospitals or temporary COVID care centres. It may not scale well for larger healthcare institutions handling thousands of beds or concurrent users.

The system lacks advanced features such as billing and invoicing, electronic medical record (EMR) integration, role-based access control (RBAC), and inventory or oxygen supply management. Security features are basic, with no implementation of data encryption or multi-factor authentication.

Additionally, the system is desktop-based and not optimized for mobile or web platforms, limiting accessibility for users on the go. Future enhancements could focus on expanding functionality, improving scalability, and strengthening security for wider adoption.

**Chapter 6**

****

# SNAP SHOTS

Fig 6.1 Home Page

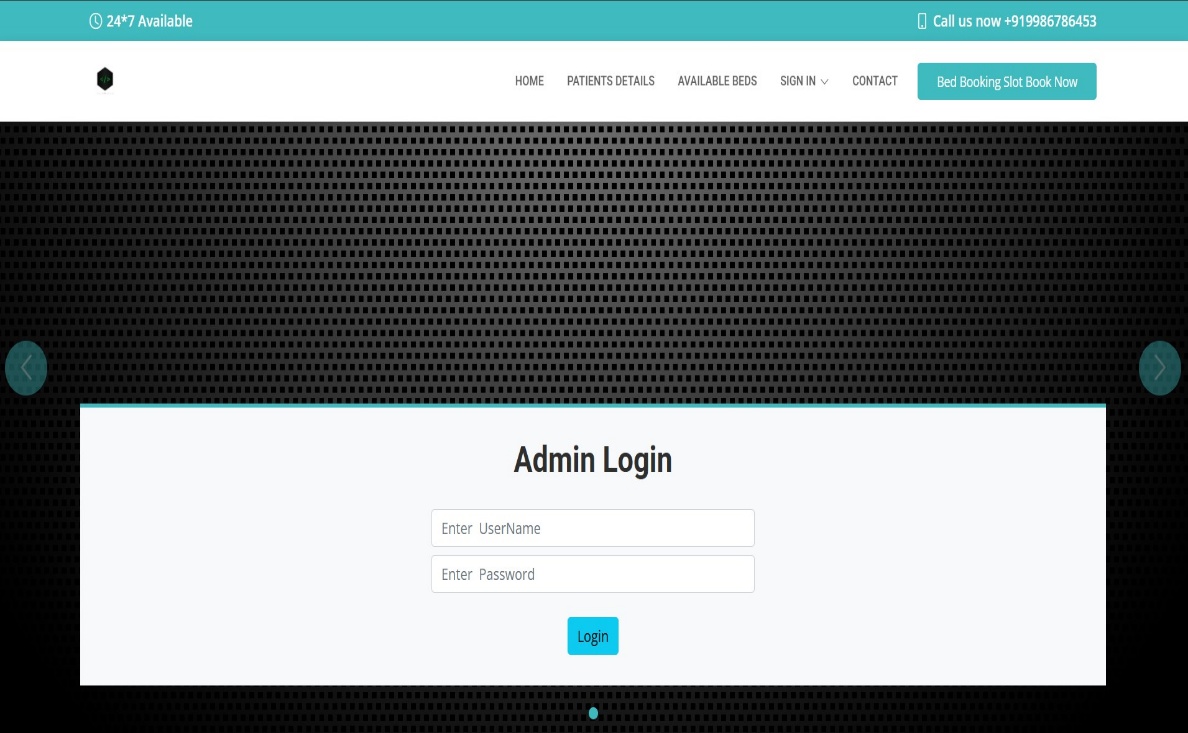
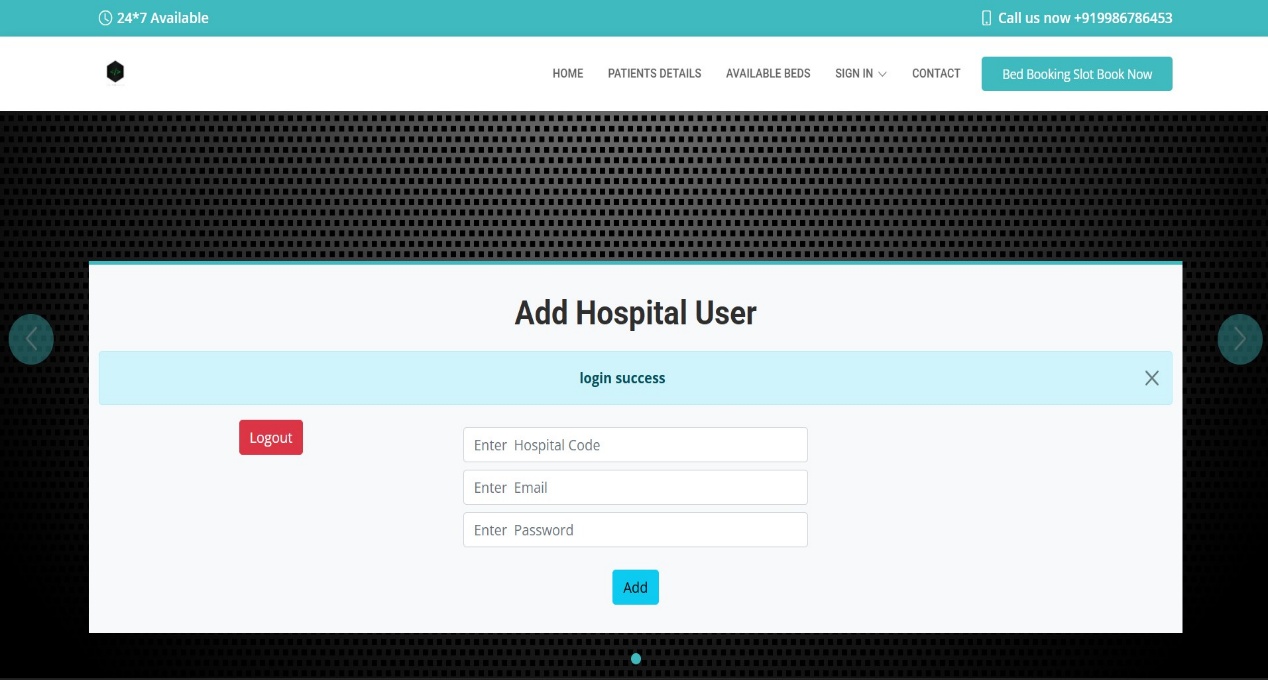


Fig 6.2 Admin Login



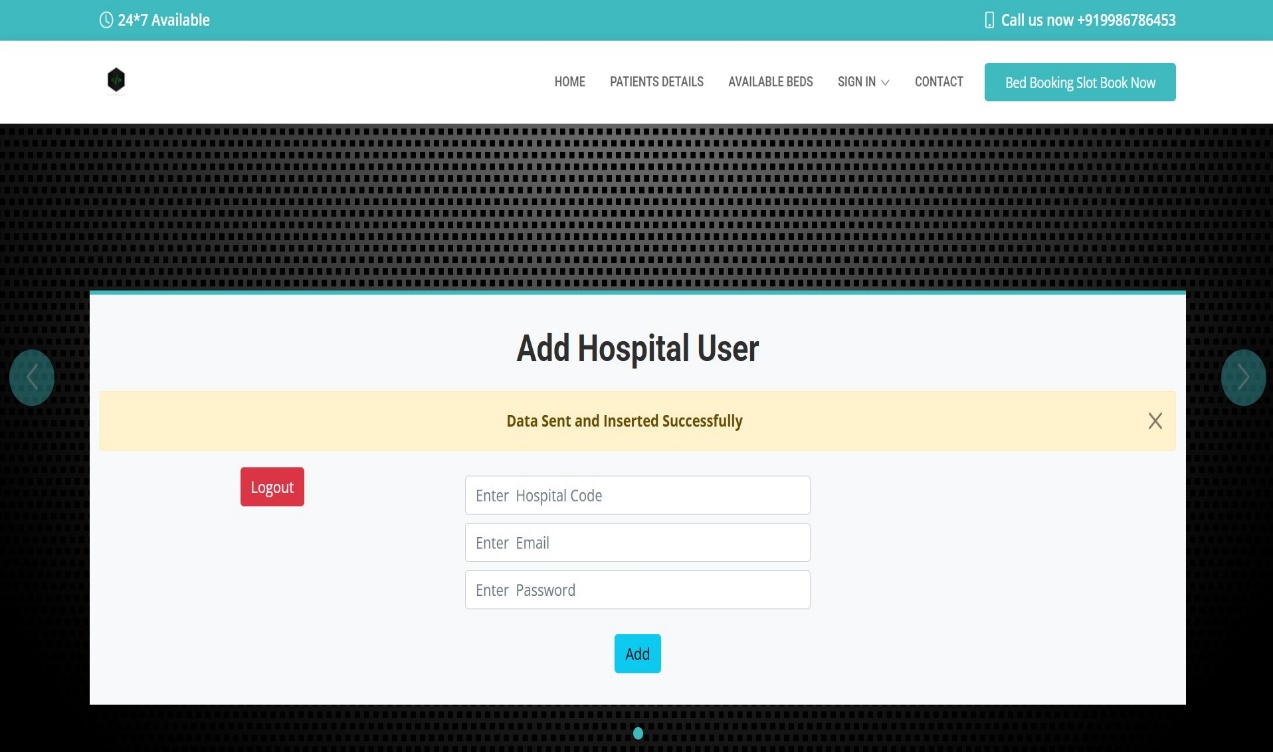


Fig 6.3 Add Hospital User

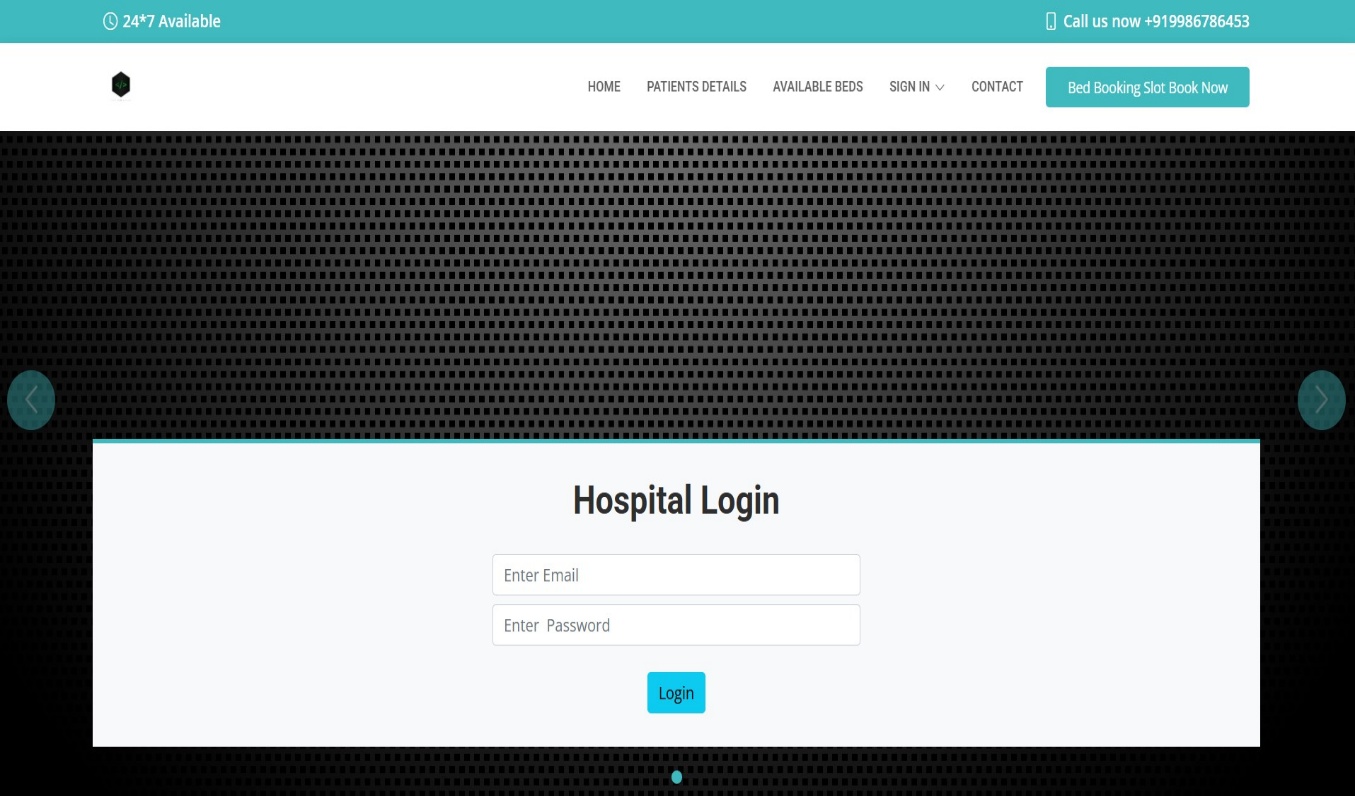


Fig 6.4 Hospital Login

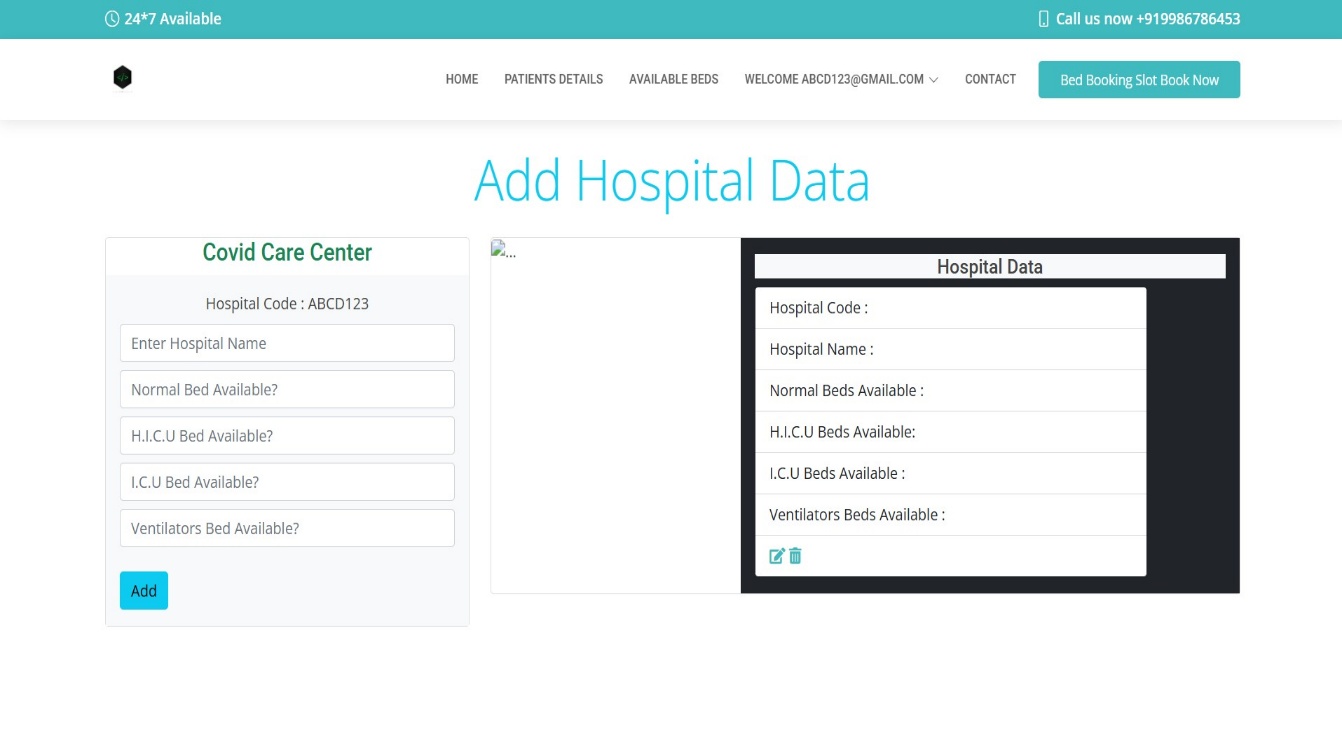


Fig 6.5 Add Hospital Data

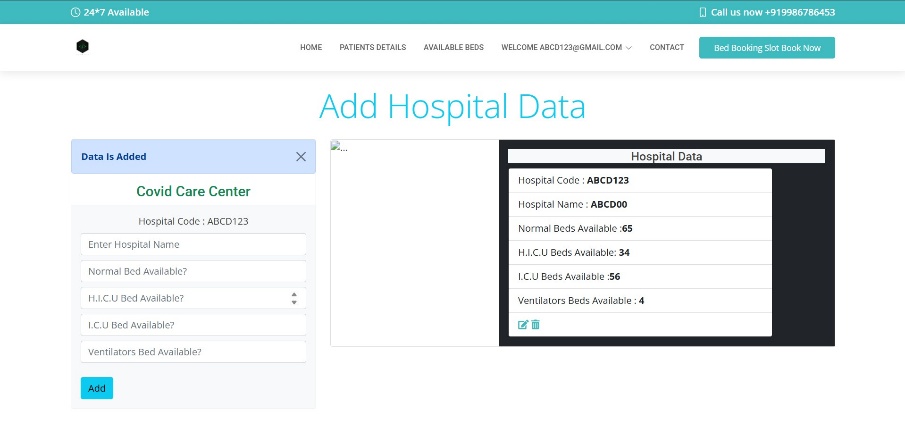


Fig 6.6 Hospital Data Added

A screenshot of a computer

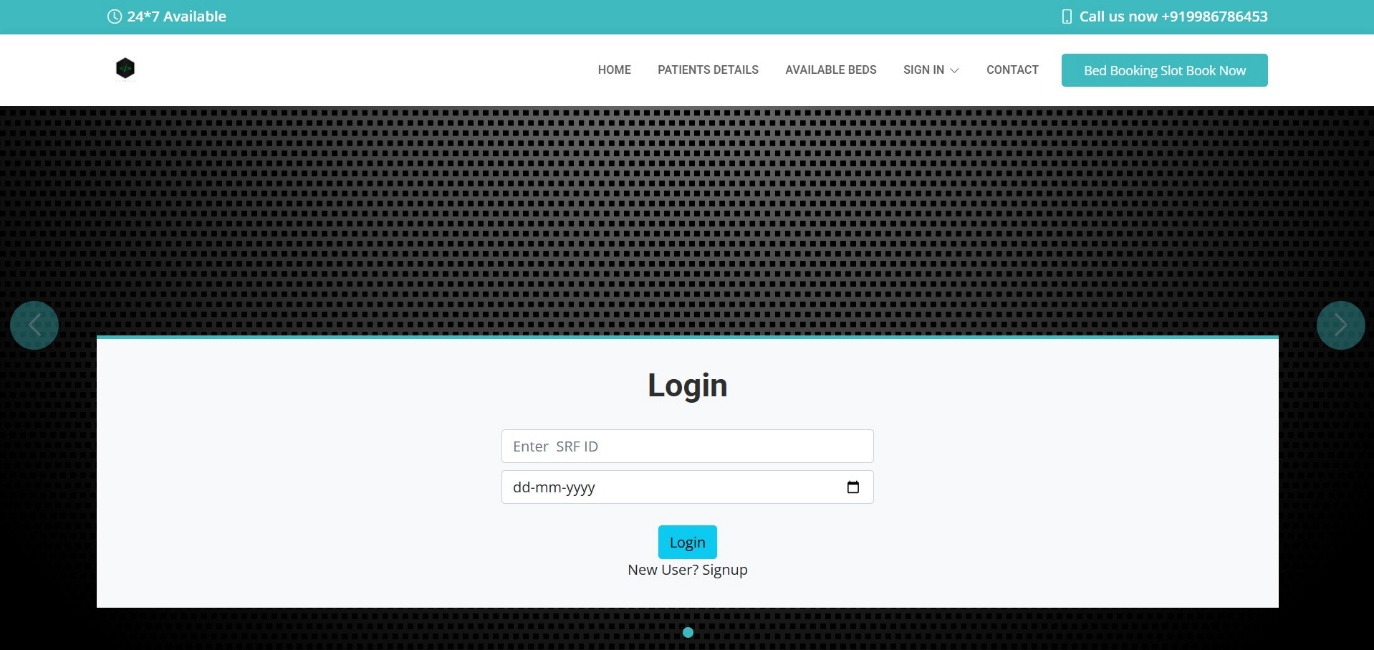
AI-generated content may be incorrect.Fig 6.7 User Login

Fig 6.7 Book Bed Slot

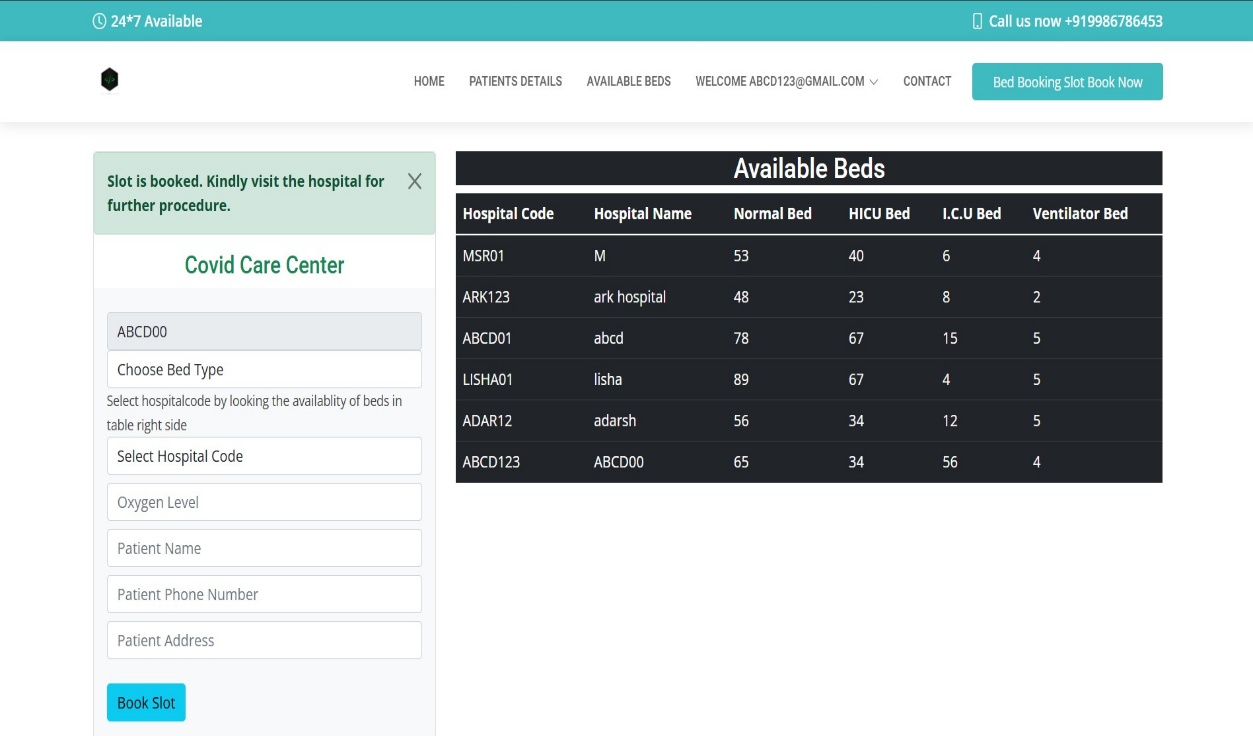


Fig 6.8 Booking Success

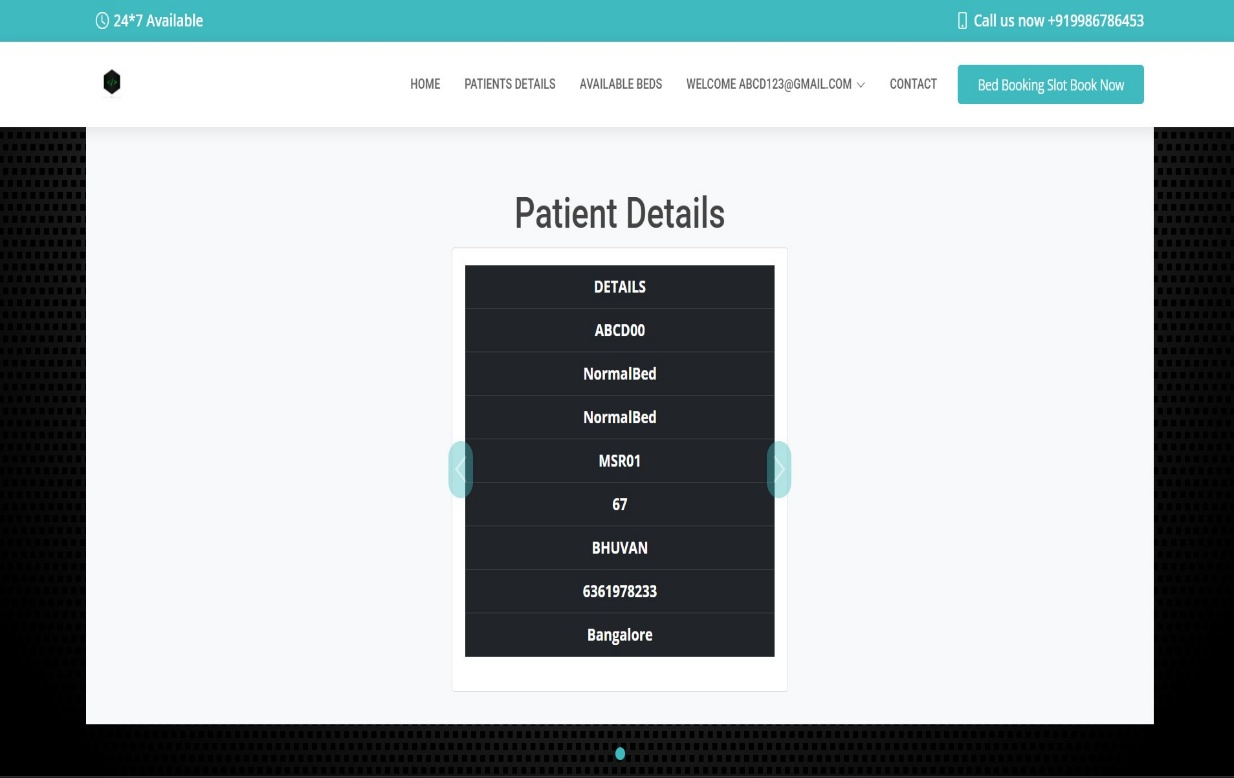


Fig 6.9 Patient Details

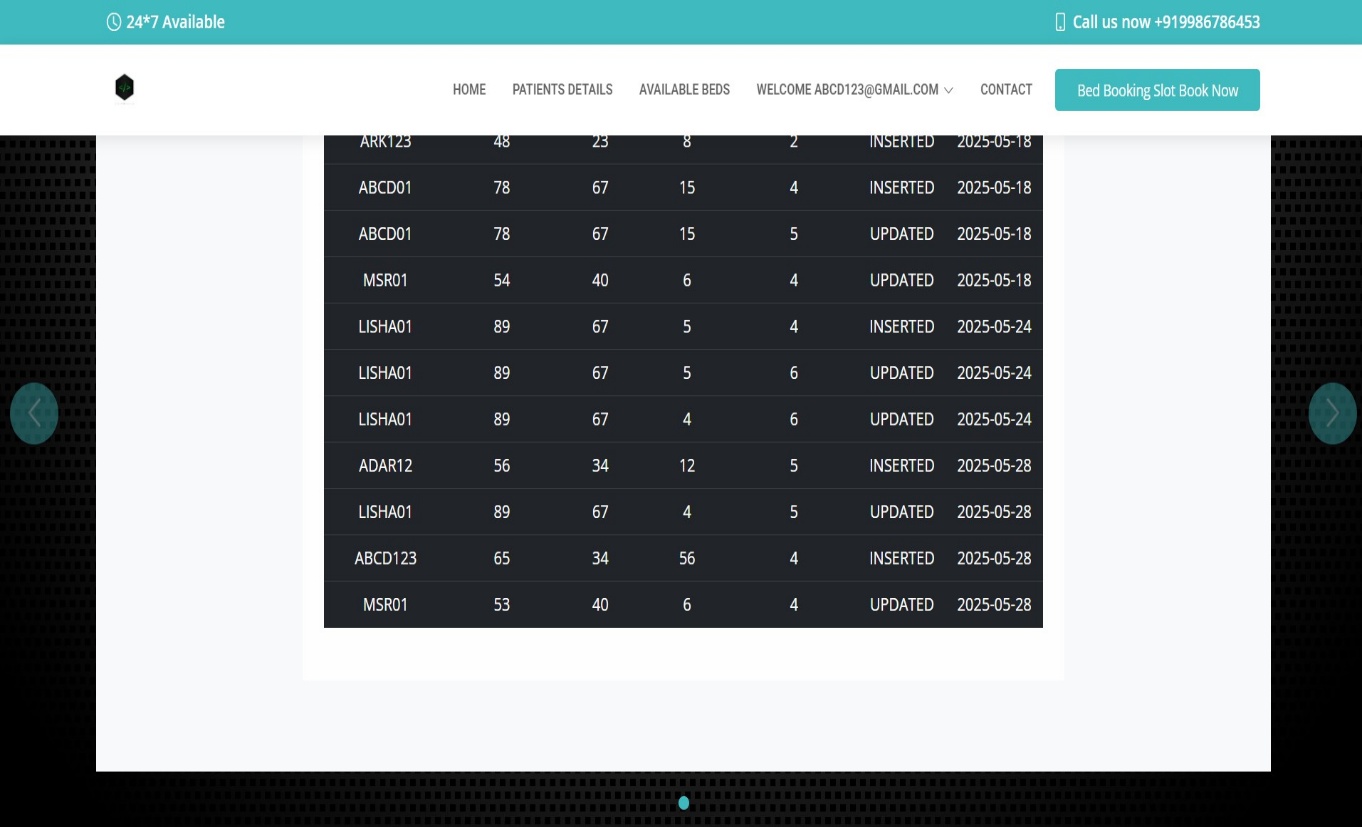


Fig 6.10 Triggered Data

**Chapter 7**

# CONCLUSION

The COVID Bed Slot Booking System project successfully automates critical functions related to hospital bed allocation, patient registration, and slot booking during the pandemic. By integrating a Python backend with a MySQL database and a simple HTML/CSS frontend, the system ensures efficient management of available beds, reliable patient tracking, and timely allocation of resources. The system improves operational accuracy, minimizes manual workload, and provides a user-friendly interface for hospital staff to manage bookings with ease. It enforces data integrity and business rules through the use of database triggers and stored procedures, ensuring consistent and secure operations. While the project meets the essential requirements of a mini system for COVID bed management, it also serves as a foundation for future upgrades such as real-time availability dashboards, mobile access, patient notification systems, and integration with broader hospital information systems. Overall, this project demonstrates the effective application of database management, backend development, and user interface design to solve real-world healthcare challenges, particularly during high-demand situations like a pandemic.

**Chapter 8**

# FUTURE ENHANCEMENTS

To improve the scalability, functionality, System, the following future enhancements are proposed: and security of the COVID Bed Slot Booking

* **Role-Based Access Control (RBAC):** Implement distinct roles such as admin, hospital staff, and receptionists to control access to specific features and data based on user responsibilities.
* **Billing and Payment Integration:** Add support for patient billing, including payment tracking and integration with online payment gateways.
* **Electronic Medical Records (EMR):** Introduce detailed EMR functionality to maintain comprehensive patient history, including previous illnesses, test reports, and treatment logs.
* **Cloud-Based Deployment:** Enable cloud-based access to support multiple hospital locations, remote management, and centralized data storage.
* **Web and Mobile Versions:** Develop responsive web and mobile applications to enhance accessibility and usability on various devices, especially for patients and remote staff.

These enhancements will significantly strengthen the system, making it more comprehensive, secure, and suitable for large-scale or long-term healthcare operations.

**Chapter 9**

# REFERENCES

1. Python (Official Download) – https:/[/www.python.org/downloads/](http://www.python.org/downloads/)
2. MySQL Server & MySQL Workbench – https://dev.mysql.com/downloads/
3. MySQL Connector/Python – https://dev.mysql.com/downloads/connector/python/
4. HTML/CSS Documentation – https://developer.mozilla.org/en-US/docs/Web
5. Flask Framework (optional if used) – https://flask.palletsprojects.com/
6. General Research and Troubleshooting – https:/[/www.google.com/](http://www.google.com/)