

COVID BED SLOT BOOKING

DBMS MINI PROJECT REPORT

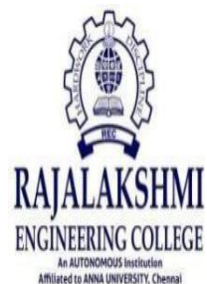
Submitted by :

JASHAREEN J 231801066

MAHALAKSHMI B 231801092

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



**RAJALAKSHMI ENGINEERING COLLEGE,
ANNA UNIVERSITY, CHENNAI: 602 105**

NOV 2024

RAJALAKSHMI ENGINEERING COLLEGE
CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this report title “**COVID BED SLOT BOOKING**” is the Bonafide work of **JASHAREEN J (231801066)** and **MAHALAKSHMI B (231801092)** who carried out the mini project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.J.Gnanasekaran.,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT,
Professor,
Department of AI&DS,
Rajalakshmi Engineering College
Chennai – 602 105.

SIGNATURE

Mrs.M.Thamizharasi.,M.Tech., (Ph.d)
SUPERVISOR,
Assistant Professor,
Department of AI&DS,
Rajalakshmi Engineering College,
Chennai – 602 105.

Submitted for the DBMS Mini project review held on_____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavor to put forth this report.

My sincere thanks to our Chairman **Mr. S. MEGANATHAN, M.E., F.I.E.**, and our Chairperson **Dr. (Mrs.)THANGAM MEGANATHAN, M.E., Ph.D.**, for providing me with the requisite infrastructure and sincere endeavoring educating me in their premier institution.

My sincere thanks to **Dr.S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to **Dr.J.GNANASEKARAN.,M.E.,Ph.D.**, Head of the Department of Artificial Intelligence and Machine Learning and Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, **Mrs.M.THAMIZHARASI.,M.Tech., (Ph.d)...**,Assistant Professor, Department of Artificial Intelligence and Machine Learning, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

ABSTRACT

The COVID-19 Bed Slot Booking System is a web-based application designed to streamline the process of hospital bed allocation for COVID-19 patients, ensuring efficient management during emergencies. The system addresses the challenges of manual bed allocation, such as delays, miscommunication, and data mismanagement, by providing a centralized and automated platform for patients, hospitals, and administrators.

The platform incorporates two main user roles: patients and hospital administrators. Patients can search for available hospital beds in their vicinity, view bed availability in real time, and book a slot based on their requirements. Hospital administrators can manage bed occupancy, update availability statuses, and ensure seamless patient admission. The system also supports emergency prioritization to address critical cases promptly.

The project uses MySQL for database management, ensuring robust and secure storage of patient and hospital data. Java is employed for backend logic, providing reliable and efficient server-side processing, while HTML, CSS, Java and JavaScript are used to create a responsive and user-friendly frontend. The platform is hosted locally using XAMPP, facilitating easy development and deployment.

Key features of the system include:

1. Real-time bed availability updates: Hospitals can dynamically update bed occupancy.
2. Search and filtering options: Patients can filter hospitals by location, bed type, and availability.
3. Secure booking process: Patients can reserve beds with essential personal details.
4. Emergency handling: The system prioritizes bookings based on patient health severity.
5. Data analytics and reporting: Hospital administrators can generate reports for monitoring and decision-making.

This project demonstrates how digital solutions can effectively address healthcare resource challenges during pandemics, ensuring timely care and saving lives.

TABLE OF CONTENTS

- 1. Introduction
 - 1.1 Overview
 - 1.2 Problem Statement
- 2. Backend Design
 - 2.1 ER Diagram
 - 2.2 ER Mapping
 - 2.3 Normalization
- 3. Front End Design
 - 3.1 Screen Layout
 - 3.2 Front End And Back End Conn
- 4. Major Modules
 - 4.1 Admin Module
 - 4.2 Hospital Module
 - 4.3 Patient Module
- 5. Implementation
 - 5.1 Create Statements
 - 5.2 Trigger Statements And Data Insertion
 - 5.3 Java Code
 - 5.4 HTML FILES
- 6. Snapshots
 - 6.1 Discussion
 - 6.2 Result
- 7. Applications
- 8. Conclusion

Introduction

1.1 Overview

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.

The project 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(sr fid) 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.

1.2 Problem Statement

“To design and develop a system that will manage information about list of various hospitals, availability of beds in hospitals, allotment of beds for patients and display patient details”.

Back End Design

2.1 Conceptual Database Design (ER Diagram)

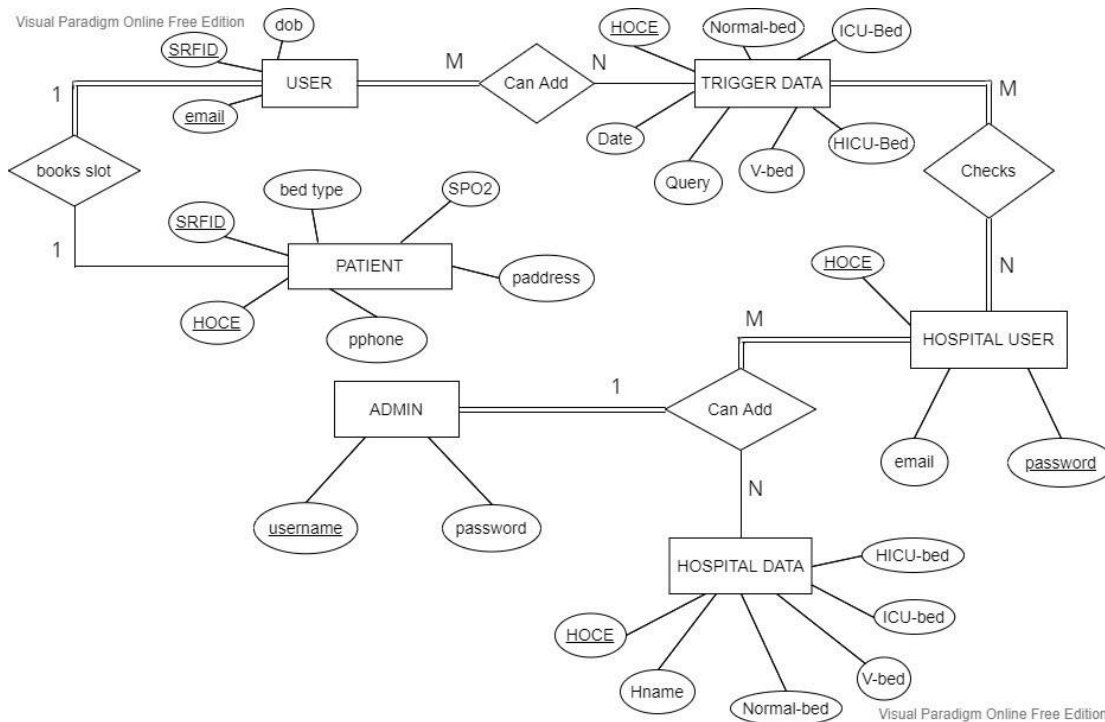


Figure 2.1: ER DIAGRAM

The ER diagram for Covid Bed Slot Booking System is as shown above in the Figure 2.1.

The covid bed slot booking database system mainly consists of 6 entities namely User, Admin, Hospital data, Hospital user, Patient, Triggered data.

- **User** and **patients** have **1:1** cardinality ratio, since 1 user can book 1 slot for patient having **unique SDFID**. User has **total participation** in all relation.
- **User** can *check* **Triggered data** of the Hospital which contains all the details about available beds in the hospital and has **M: N** cardinality ratio
- **Hospital user** can *add or update* the **Hospital data** which consists of available beds in the hospital and will get updated regularly.
- Hospital data and Hospital user has **M: N** cardinality ratio.

- **Hospital user** and **Triggered data** have **M: N** cardinality ratio and both have **total participation**.
- **Admin** *adds* **Hospital user** and in-turn hospital user adds the hospital data. Admin has **total participation** in adding hospital user and has **1: N** cardinality ratio.

2.2 Logical Database Design (ER MAPPING)

A schema diagram can display only some aspects of a schema like the name of record type, data type and constraints. Other aspects can't be specified through the schema diagram.

ADMIN (Username, Password)

HOSPITAL USER (Hoce, Password, Email)

HOSPITAL DATA (Hoce, Hname, Normalbed, Vbed, HICUbed, ICUbed)

TRIGGERED DATA (Hoce, Normalbed, Vbed, HICUbed, ICUbed, Query, Date)

USER (Email, Srfid, Dob)

PATIENTS (Hoce, Srfid, Bedtype, SPO2, Pphone, Paddress)

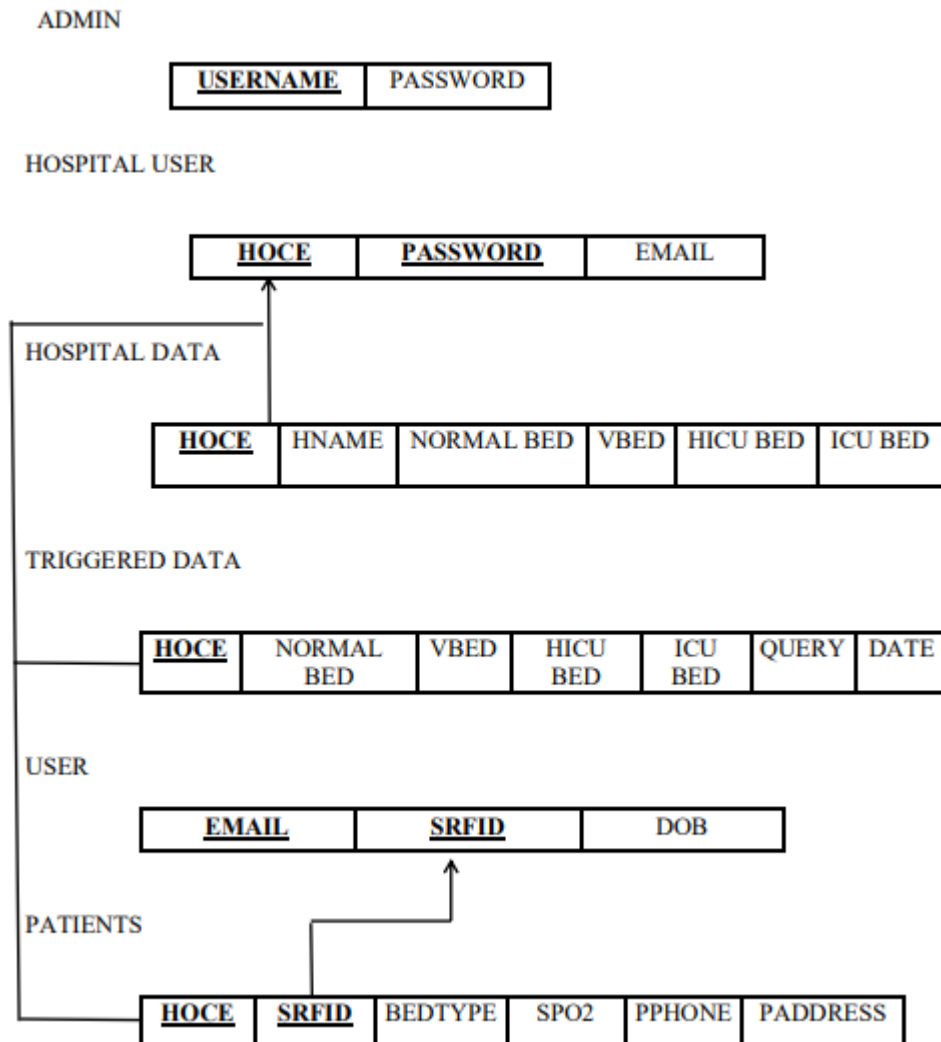


Figure 2.2: ER to Relational Mapping

2.3 Normalization

Normalization is the process of reorganizing data in a database so that it meets two basic requirements:

- There is no redundancy of data (all data is stored in only one place) and
- Data dependencies are logical (all related data items are stored together).

First Normal Form (1NF)

The table is said to be in 1NF if it follows following rules

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be the same domain.
- All the columns in a table should have unique names.
- And the order in which data is stored, does not matter.

Second Normal Form (2NF)

For a table to be in 2NF, following conditions must be followed

- It should be in the First Normal Form
- It should not have partial functional dependency.

Partial dependency means that a non-prime attribute is functionally dependent on part of a candidate key.

Third Normal Form (3NF)

A table is said to be in 3NF when,

- It is in the Second Normal Form (i.e., it should not have partial functional dependency).
- It does not have transitive dependency.

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For example:

$X \rightarrow Z$ is a transitive dependency if the following functional dependencies hold true:

$X \rightarrow Y$

$Y \rightarrow Z$

BOOKING PATIENT

<u>PID</u>	Srfid	Bedtype	Hcode	Spo2	Pname	Pphone	Paddress
------------	-------	---------	-------	------	-------	--------	----------

Functional Dependencies

PID --> (Srfid, Bedtype, Hcode, Spo2, Pname, Pphone, Paddress)

PID+= (PID, Srfid, Bedtype, Hcode, Spo2, Pname, Pphone, Paddress)

Candidate key= PID

PID	Srfid	Bedtype	Hcode	Spo2	Pname	Pphone	Paddress
1	KA12345678	ICUBed	BBH01	89	Rahul	9833276236	Bangalore
2	KA64527123	HICUBed	BBA02	85	Akshay	6453729819	Mysore

Table 2.1: Booking Patient

Justification

- The above-mentioned dependencies we can observe that the BOOKINGPATIENT relation is in 1NF as all the attributes are atomic in nature.
- The relation is in 2NF as all the Non-Prime attributes are fully functionally dependent on the Prime attribute and there is no partial dependency.
- The relation is also in 3NF as there is no Transitive dependency between the Non-Prime attributes and the Prime attribute.

HOSPITAL DATA

<u>HDID</u>	Hcode	Hname	Normalbed	HICUbed	ICUbed	Vbed
-------------	-------	-------	-----------	---------	--------	------

Functional Dependencies

HDID --> (Hcode, Hname, Normalbed, HICUbed, ICUbed, Vbed)

HDID+= (HDID, Hcode, Hname, Normalbed, HICUbed, ICUbed, Vbed)

Candidate key= HDID

HDID	Hcode	Hname	Normalbed	HICUbed	ICUbed	Vbed
1	BBH01	KIMS	15	9	4	2
2	BBA02	Apollo	30	19	5	7

Table 2.2: Hospital Data

Justification

- The above-mentioned dependencies we can observe that the HOSPITALDATA relation is in 1NF as all the attributes are atomic in nature.
- The relation is in 2NF as all the Non-Prime attributes are fully functionally dependent on the Prime attribute and there is no partial dependency.
- The relation is also in 3NF as there is no Transitive dependency between the Non-Prime attributes and the Prime attribute.

HOSPITAL USER

<u>HUID</u>	Hcode	Email	Password
-------------	-------	-------	----------

Functional Dependencies

HUID --> (Hcode, Hname, Email, Password)

HUID+= (HUID, Hcode, Hname, Email, Password)

Candidate key= HUID

HUID	Hcode	Email	Password
2	BBH01	myspace.amd@gmail.com	2022bbh01
5	BBA02	abhishekgowda1906@gmail.com	2022bba02

Table 2.3: Hospital User

Justification

- The above-mentioned dependencies we can observe that the HOSPITALUSER relation is in 1NF as all the attributes are atomic in nature.
- The relation is in 2NF as all the Non-Prime attributes are fully functionally dependent on the Prime attribute and there is no partial dependency.
- The relation is also in 3NF as there is no Transitive dependency between the Non-Prime attributes and the Prime attribute.

TRIGGERED DATA

<u>TDID</u>	Hcode	Normalbed	HICUbed	ICUbed	Vbed	Queries	Date
-------------	-------	-----------	---------	--------	------	---------	------

Functional Dependencies

TDID --> (Hcode, Normalbed, HICUbed, ICUbed, Vbed, Queries, Date)

TDID+= (TDID, Hcode, Normalbed, HICUbed, ICUbed, Vbed, Queries, Date)

Candidate key= TDID

TDID	Hcode	Normalbed	HICUbed	ICUbed	Vbed	Queries	Date
1	BBH01	15	10	4	3	UPDATES	2022-01-26
2	BBH01	15	9	4	2	UPDATES	2022-01-26

Table 2.4: Triggered Data

Justification

- The above-mentioned dependencies we can observe that the TRIGGERDATA relation is in 1NF as all the attributes are atomic in nature.
- The relation is in 2NF as all the Non-Prime attributes are fully functionally dependent on the Prime attribute and there is no partial dependency.
- The relation is also in 3NF as there is no Transitive dependency between the Non-Prime attributes and the Prime attribute.

USER

<u>UID</u>	Srfid	Email	Dob
------------	-------	-------	-----

Functional Dependencies

UID --> (Srfid, Email, Dob)

UID+= (UID, Srfid, Email, Dob)

Candidate key= UID

UID	Srfid	Email	Dob
3	KA12345678	amd@gmail.com	16-06-2001
4	KA64527123	akshay@gmail.com	10-07-2007

Table 2.5: User

Justification

- The above-mentioned dependencies we can observe that the USER relation is in 1NF as all the attributes are atomic in nature.
- The relation is in 2NF as all the Non-Prime attributes are fully functionally dependent on the Prime attribute and there is no partial dependency.
- The relation is also in 3NF as there is no Transitive dependency between the Non-Prime attributes and the Prime attribute.

Front End Design

3.1 Screen Layout Design for Webpages, forms

Screen layout is the part of graphic design that deals with the arrangement of visual elements on a page. Screen layout is used to make the web pages look better. It establishes the overall appearance, relative importance and relationships between the graphic elements to achieve a smooth flow of information and eye movement for maximum effectiveness or impact.

Using visually attractive and simple design forms the basis of all the screen layout design and using these layouts in authentication and login makes it easy to use and user-friendly.

Screen layout determines the overall structure of your screen. They define the structure of a harness included in a composite portal. A harness can contain only one screen layout.

A screen layout is comprised of the following:

- `<header>` - Defines a header for a document or a section
- `<nav>` - Defines a set of navigation links
- `<section>` - Defines a section in a document
- `<article>` - Defines an independent, self-contained content
- `<aside>` - Defines content aside from the content (like a sidebar)
- `<footer>` - Defines a footer for a document or a section



Figure 3.1: Screen Layout Design

3.2 Front End and Back End Connectivity

What is MySQL?

MySQL is a widely used relational database management system developed, distributed, and supported by Oracle Corporation. It is ideal for both small and large applications, highly reliable, fast, and easy to use. It supports standard SQL and can be compiled on various platforms.

What is JPA?

JPA (Java Persistence API) is a specification for managing relational data in Java applications. It enables developers to map Java objects to database tables and execute database operations in a simple and abstract manner.

What is Spring Data JPA?

Spring Data JPA is a Spring framework module that integrates JPA into a Spring application. It simplifies database operations by providing default implementations for repository operations.

Installation and Setup:

Steps to Install and Set Up MySQL:

1. Download and install MySQL or a server like WAMP.
2. Create a database named flaskcodeoop (or choose any name).
3. Add tables directly using JPA Entity classes.

Steps to Set Up Spring Boot:

1. Create a Spring Boot application using Spring Initializr.
2. Add dependencies: Spring Web, Spring Data JPA, MySQL Driver

```
package com.example.demo.model;
import jakarta.persistence.*;
@Entity
public class UserInfo {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```



```

@Column(unique = true, nullable = false)
private String username;
@Column(nullable = false)
private String password;
// Constructors
public UserInfo() {}
public UserInfo(String username, String password) {
    this.username = username;
    this.password = password;
}
// Getters and Setters
public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public String getUsername() {
    return username;
}
public void setUsername(String username) {
    this.username = username;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
}

```

If you check your flaskcodeloop database, you can see that we have our table with the data.

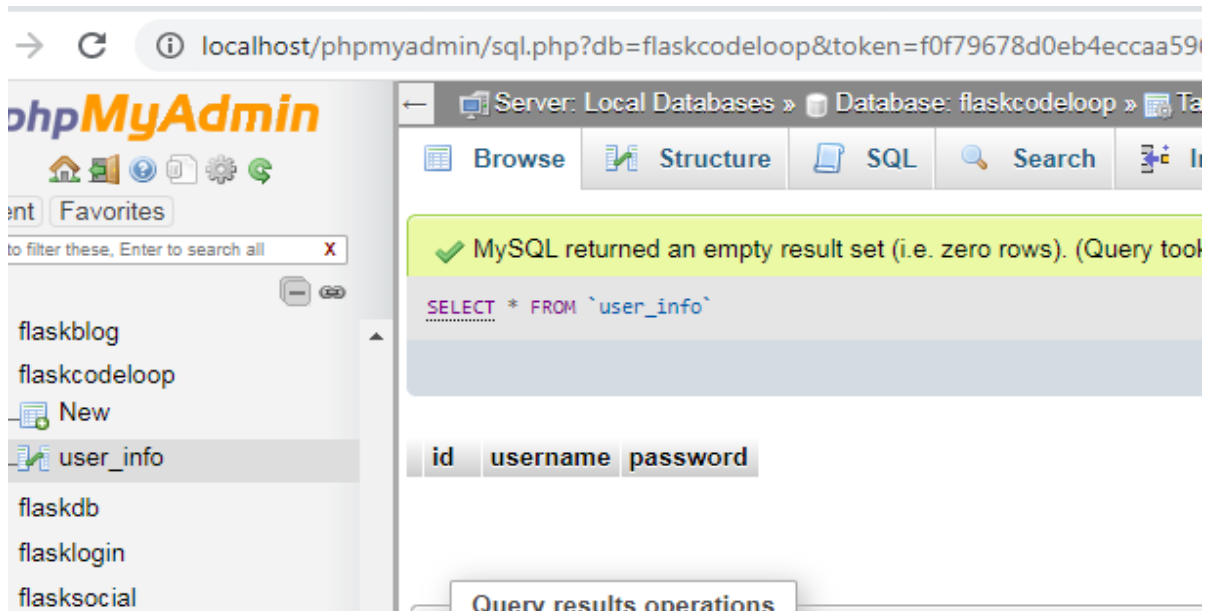


Fig 3.2.1: Back end database

Now let's add some data to our database. basically, we are going to add just two UserInfo to our database table.

```
package com.example.demo;
import com.example.demo.model.UserInfo;
import com.example.demo.repository.UserInfoRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
```

@Component

```
public class DataLoader implements CommandLineRunner {
    @Autowired
    private UserInfoRepository userInfoRepository;
```

@Override

```
public void run(String... args) throws Exception {
    // Adding two UserInfo objects
    UserInfo user1 = new UserInfo("codeloop", "1234");
    UserInfo user2 = new UserInfo("parwiz", "12345");
```

```

// Save users to the database
userInfoRepository.save(user1);
userInfoRepository.save(user2);
}
}

```

Check your database table, you will have two UserInfo data.

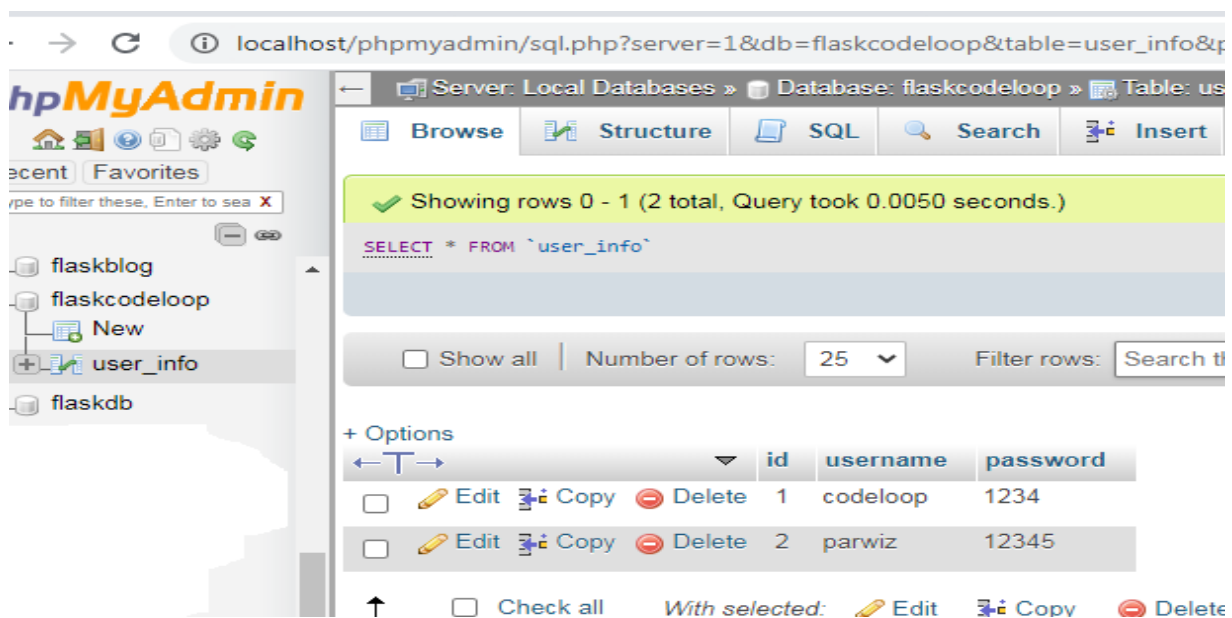


Fig 3.2.2: Back end database table values

Major Modules

4.1 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

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

4.2 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

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

4.3 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

1. **Login:** User uses srf_id and date-of-birth to login.
2. **Booking Bed Slot:** User can book slot among the available beds in the listed hospital.
3. **View Available beds:** View the beds available in the listed hospital.
4. **Patient Details:** User can view the entered patient details.
5. **Logout:** Patient logs out of the page.

Implementation

5.1 Create Statements

```
CREATE TABLE `bookingpatient` (  
  `id` int(11) NOT NULL,  
  `srfid` varchar(20) NOT NULL,  
  `bedtype` varchar(50) NOT NULL,  
  `hcode` varchar(50) NOT NULL,  
  `spo2` int(11) NOT NULL,  
  `pname` varchar(50) NOT NULL,  
  `pphone` varchar(12) NOT NULL,  
  `paddress` text NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ALTER TABLE `bookingpatient`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `srfid` (`srfid`);
```

```
ALTER TABLE `bookingpatient`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
```

```
CREATE TABLE `hospitaldata` (  
  `id` int(11) NOT NULL,  
  `hcode` varchar(200) NOT NULL,  
  `hname` varchar(200) NOT NULL,  
  `normalbed` int(11) NOT NULL,  
  `hicubed` int(11) NOT NULL,  
  `icubed` int(11) NOT NULL,  
  `vbed` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ALTER TABLE `hospitaldata`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `hcode` (`hcode`);
```

```
ALTER TABLE `hospitaldata`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

```
CREATE TABLE `hospitaluser` (  
  `id` int(11) NOT NULL,  
  `hcode` varchar(20) NOT NULL,  
  `email` varchar(100) NOT NULL,  
  `password` varchar(1000) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ALTER TABLE `hospitaluser`  
  ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `hospitaluser`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;
```

```
CREATE TABLE `trig` (  
  `id` int(11) NOT NULL,  
  `hcode` varchar(50) NOT NULL,  
  `normalbed` int(11) NOT NULL,  
  `hicubed` int(11) NOT NULL,  
  `icubed` int(11) NOT NULL,  
  `vbed` int(11) NOT NULL,  
  `querys` varchar(50) NOT NULL,  
  `date` date NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ALTER TABLE `trig`  
  ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `trig`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
```

```
CREATE TABLE `user` (  
  `id` int(11) NOT NULL,  
  `srfid` varchar(20) NOT NULL,
```

```
`email` varchar(50) NOT NULL,  
`dob` varchar(1000) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
ALTER TABLE `user`  
  ADD PRIMARY KEY (`id`),  
  ADD UNIQUE KEY `srfid` (`srfid`);  
ALTER TABLE `user`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

5.2 Trigger statements and Data insertion

```
CREATE TRIGGER `Delete` BEFORE DELETE ON `hospitaldata` FOR EACH ROW INSERT INTO trig  
VALUES(null,OLD.hcode,OLD.normalbed,OLD.hicubed,OLD.icubed,OLD.vbed,'DELETED',NOW())
```

```
CREATE TRIGGER `Insert` AFTER INSERT ON `hospitaldata` FOR EACH ROW INSERT INTO trig  
VALUES(null,NEW.hcode,NEW.normalbed,NEW.hicubed,NEW.icubed,NEW.vbed,'INSERTED',NOW())
```

```
CREATE TRIGGER `Update` AFTER UPDATE ON `hospitaldata` FOR EACH ROW INSERT INTO trig  
VALUES(null,NEW.hcode,NEW.normalbed,NEW.hicubed,NEW.icubed,NEW.vbed,'UPDATED',NOW())
```

```
INSERT INTO `bookingpatient` (`id`, `srfid`, `bedtype`, `hcode`, `spo2`, `pname`, `pphone`, `paddress`)  
VALUES
```

```
(1, 'KA12345678', 'ICUBed', 'BBH01', 89, 'Rahul', '9833276236', 'Bangalore');
```

```
INSERT INTO `hospitaldata` (`id`, `hcode`, `hname`, `normalbed`, `hicubed`, `icubed`, `vbed`) VALUES
```

```
(2, 'BBH01', 'KIMS', 15, 9, 4, 2),
```

```
(3, 'BBA02', 'bangalore', 30, 19, 5, 7);
```

```
INSERT INTO `hospitaluser` (`id`, `hcode`, `email`, `password`) VALUES
```

```
(2, 'BBH01', 'myspace.amd@gmail.com,
```

```
'pbkdf2:sha256:260000$M1iXPteF8RJ2qbM3$622f475ce5dd3de53f383487c0415624da68d2a203b38bd0efa7  
e6c803a0b632'),
```

```
(5, 'BBA02', 'abhishekgowda1906@gmail.com,
```

```
'pbkdf2:sha256:260000$QuPrZOfiOdQlAaSh$612549985607d0805543f933ab8ed3215993f7165b6a506fce61  
636127f90fcd');
```

```
INSERT INTO `trig` (`id`, `hcode`, `normalbed`, `hicubed`, `icubed`, `vbed`, `querys`, `date`) VALUES
(1, 'BBH01', 15, 10, 4, 3, 'UPDATED', '2022-01-26'),
(2, 'BBH01', 15, 9, 4, 2, 'UPDATED', '2022-01-26');
```

```
INSERT INTO `user` (`id`, `srfid`, `email`, `dob`) VALUES
(3, 'KA12345678', 'amd@gmail.com',
'pbkdf2:sha256:260000$UeXj8AGlpZ050nCv$c7f0f357f3db0211c9a984db4e305b88de8722f21d354425e62c
303a816220c3');
```

5.3 JAVA Code

5.3.1 Packages

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.web.bind.annotation.*;
import org.springframework.stereotype.Service;

import jakarta.persistence.*;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Controller;

import java.util.List;

@SpringBootApplication
public class CovidCareApplication {
    public static void main(String[] args) {
        SpringApplication.run(CovidCareApplication.class, args);
    }
}

// --- Entities ---
```


@Entity

```
class User {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private int id;  
    private String srfid;  
    private String email;  
    private String dob;  
  
    // Getters and Setters  
    public int getId() { return id; }  
    public void setId(int id) { this.id = id; }  
    public String getSrfid() { return srfid; }  
    public void setSrfid(String srfid) { this.srfid = srfid; }  
    public String getEmail() { return email; }  
    public void setEmail(String email) { this.email = email; }  
    public String getDob() { return dob; }  
    public void setDob(String dob) { this.dob = dob; }  
}
```

@Entity

```
class HospitalUser {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private int id;  
    private String hcode;  
    private String email;  
    private String password;  
  
    // Getters and Setters  
    public int getId() { return id; }  
    public void setId(int id) { this.id = id; }  
    public String getHcode() { return hcode; }  
    public void setHcode(String hcode) { this.hcode = hcode; }  
    public String getEmail() { return email; }  
    public void setEmail(String email) { this.email = email; }
```

```

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }
}

```

@Entity

```

class HospitalData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String hcode;
    private String hname;
    private int normalbed;
    private int hicubed;
    private int icubed;
    private int vbed;

    // Getters and Setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getHcode() { return hcode; }
    public void setHcode(String hcode) { this.hcode = hcode; }
    public String getHname() { return hname; }
    public void setHname(String hname) { this.hname = hname; }
    public int getNormalbed() { return normalbed; }
    public void setNormalbed(int normalbed) { this.normalbed = normalbed; }
    public int getHicubed() { return hicubed; }
    public void setHicubed(int hicubed) { this.hicubed = hicubed; }
    public int getIcubed() { return icubed; }
    public void setIcubed(int icubed) { this.icubed = icubed; }
    public int getVbed() { return vbed; }
    public void setVbed(int vbed) { this.vbed = vbed; }
}

```

@Entity

```

class BookingPatient {
    @Id

```

```

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;
private String srfid;
private String bedtype;
private String hcode;
private int spo2;
private String pname;
private String pphone;
private String paddress;

// Getters and Setters
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getSrfid() { return srfid; }
public void setSrfid(String srfid) { this.srfid = srfid; }
public String getBedtype() { return bedtype; }
public void setBedtype(String bedtype) { this.bedtype = bedtype; }
public String getHcode() { return hcode; }
public void setHcode(String hcode) { this.hcode = hcode; }
public int getSpo2() { return spo2; }
public void setSpo2(int spo2) { this.spo2 = spo2; }
public String getPname() { return pname; }
public void setPname(String pname) { this.pname = pname; }
public String getPphone() { return pphone; }
public void setPphone(String pphone) { this.pphone = pphone; }
public String getPaddress() { return paddress; }
public void setPaddress(String paddress) { this.paddress = paddress; }
}

interface UserRepository extends JpaRepository<User, Integer> {
    User findBySrfid(String srfid);
    User findByEmail(String email);
}

interface HospitalUserRepository extends JpaRepository<HospitalUser, Integer> {
    HospitalUser findByEmail(String email);
}

```

```

interface HospitalDataRepository extends JpaRepository<HospitalData, Integer> {
    HospitalData findByHcode(String hcode);
}

interface BookingPatientRepository extends JpaRepository<BookingPatient, Integer> {
    BookingPatient findBySrfid(String srfid);
}

@RestController
@RequestMapping("/api")
class MainController {
    @Autowired
    private UserRepository userRepository;
    @Autowired
    private HospitalUserRepository hospitalUserRepository;
    @Autowired
    private HospitalDataRepository hospitalDataRepository;
    @Autowired
    private BookingPatientRepository bookingPatientRepository;
    @Autowired
    private JavaMailSender mailSender;
    @PostMapping("/signup")
    public String signup(@RequestParam String srfid, @RequestParam String email, @RequestParam String
dob) {
        if (userRepository.findBySrfid(srfid) != null || userRepository.findByEmail(email) != null) {
            return "Email or SRF ID already exists!";
        }
        User user = new User();
        user.setSrfid(srfid);
        user.setEmail(email);
        user.setDob(dob); // In real-world apps, hash sensitive data.
        userRepository.save(user);
        return "Signup Successful!";
    }
    @PostMapping("/login")
    public String login(@RequestParam String srfid, @RequestParam String dob) {
        User user = userRepository.findBySrfid(srfid);

```

```

        if (user != null && user.getDob().equals(dob)) {
            return "Login Successful!";
        }
        return "Invalid credentials!";
    }

    @PostMapping("/addHospitalData")
    public String addHospitalData(@RequestParam String hcode, @RequestParam String hname,
                                   @RequestParam int normalbed, @RequestParam int hicubed,
                                   @RequestParam int icubed, @RequestParam int vbed) {
        if (hospitalDataRepository.findByHcode(hcode) != null) {
            return "Hospital code already exists!";
        }
        HospitalData hospitalData = new HospitalData();
        hospitalData.setHcode(hcode.toUpperCase());
        hospitalData.setHname(hname);
        hospitalData.setNormalbed(normalbed);
        hospitalData.setHicubed(hicubed);
        hospitalData.setIcubed(icubed);
        hospitalData.setVbed(vbed);
        hospitalDataRepository.save(hospitalData);
        return "Hospital Data Added!";
    }

    @PostMapping("/bookBed")
    public String bookBed(@RequestParam String srfid, @RequestParam String bedtype, @RequestParam
String hcode) {
        HospitalData hospital = hospitalDataRepository.findByHcode(hcode);
        if (hospital == null) return "Hospital not found!";
        switch (bedtype) {
            case "NormalBed":
                if (hospital.getNormalbed() > 0) hospital.setNormalbed(hospital.getNormalbed() - 1);
                else return "No Normal Beds available!";
                break;
            case "HICUBed":
                if (hospital.getHicubed() > 0) hospital.setHicubed(hospital.getHicubed() - 1);
                else return "No HICU Beds available!";
                break;
        }
    }

```

```

    case "ICUBed":
        if (hospital.getIcubed() > 0) hospital.setIcubed(hospital.getIcubed() - 1);
        else return "No ICU Beds available!";
        break;
    case "VentilatorBed":
        if (hospital.getVbed() > 0) hospital.setVbed(hospital.getVbed() - 1);
        else return "No Ventilator Beds available!";
        break;
    default:
        return "Invalid bed type!";
}
hospitalDataRepository.save(hospital);
// Save booking details
BookingPatient booking = new BookingPatient();
booking.setSrfid(srfid);
booking.setBedtype(bedtype);
booking.setHcode(hcode);
booking.setBookingDate(new Date()); // Assuming there is a booking date field
bookingPatientRepository.save(booking);
// (Optional) Send confirmation email
User user = userRepository.findBySrfid(srfid);
if (user != null) {
    String emailContent = String.format(
        "Dear %s,\n\nYour booking for a %s at hospital %s has been successfully confirmed.\n\nThank
you.",
        user.getSrfid(), bedtype, hcode
    );
    SimpleMailMessage message = new SimpleMailMessage();
    message.setTo(user.getEmail());
    message.setSubject("Hospital Bed Booking Confirmation");
    message.setText(emailContent);
    mailSender.send(message);
}

return "Bed booked successfully!";
}

```

5.4 HTML FILES

5.4.1 Base.html (Home page)

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="utf-8">
```

```
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
```

```
    <title>{ % block title % } { % endblock title % } </title>
```

```
    <meta content="" name="description">
```

```
    <meta content="" name="keywords">
```

```
    <!-- Favicons -->
```

```
    <link href="static/assets/img/favicon.png" rel="icon">
```

```
    <link href="static/assets/img/apple-touch-icon.png" rel="apple-touch-icon">
```

```
    <!-- Google Fonts -->
```

```
    <link
```

```
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i"
```

```
    rel="stylesheet">
```

```
    <!-- Vendor CSS Files -->
```

```
    <link href="static/assets/vendor/fontawesome-free/css/all.min.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/animate.css/animate.min.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/aos/aos.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
```

```
    <link href="static/assets/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
```

<!-- Template Main CSS File -->

<link href="static/assets/css/style.css" rel="stylesheet">

</head>

<body>

<!-- ===== Top Bar ===== -->

<div id="topbar" class="d-flex align-items-center fixed-top">

<div class="container d-flex align-items-center justify-content-center justify-content-md-between">

<div class="align-items-center d-none d-md-flex">

<i class="bi bi-clock"></i> 24*7 Available

</div>

<div class="d-flex align-items-center">

<i class="bi bi-phone"></i> Call us now +91-9558955488

</div>

</div>

</div>

<!-- ===== Header ===== -->

<header id="header" class="fixed-top">

<div class="container d-flex align-items-center">

<!-- Uncomment below if you prefer to use an image logo -->

<!-- <h1 class="logo me-auto">Medicio</h1> -->

<nav id="navbar" class="navbar order-last order-lg-0">

Home

Patient Details

Available Beds


```

        <!-- <li><a class="nav-link scrollto" href="/triggers">Operations</a></li> -->

{ % if current_user.is_authenticated and current_user.hcode% }

<li><a class="nav-link scrollto" href="/triggers">Operations</a></li>

<li class="dropdown"><a href="/login"><span>Welcome { { current_user.email } }</span> <i
class="bi bi-chevron-down"></i></a>

    <ul>

        <li><a href="/addhospitalinfo">Add Hospital Data</a></li>

        <li><a href="/logout">Logout</a></li>

    </ul>

</li>

{ % elif current_user.is_authenticated and current_user.srfid % }

<li class="dropdown"><a href="/login"><span>Welcome { { current_user.email } }</span> <i
    class="bi bi-chevron-down"></i></a>

    <ul>

        <li><a href="/slotbooking">Book Slot</a></li>

        <li><a href="/logout">Logout</a></li>

    </ul>

</li>

{ % else % }

<li class="dropdown"><a href="/login"><span>Sign In</span> <i class="bi bi-chevron-
down"></i></a>

    <ul>

        <li><a href="/login">User Login</a></li>

        <li><a href="/hospitallogin">Hospital Login</a></li>

        <li><a href="/admin">Admin Login</a></li>

    </ul>

</li>

```

```

    { % endif % }

    <li><a class="nav-link scrollto" href="#contact">Contact</a></li>

</ul>

<i class="bi bi-list mobile-nav-toggle"></i>

</nav><!-- .navbar -->

<a href="/slotbooking" class="appointment-btn scrollto"><span class="d-none d-md-inline">Book
Bed</span>

Slot Now</a>

</div>

</header><!-- End Header -->

<!-- ===== Hero Section ===== -->

<section id="hero">

    <div id="heroCarousel" data-bs-interval="5000" class="carousel slide carousel-fade" data-bs-
ride="carousel">

        <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

        <div class="carousel-inner" role="listbox">

            <!-- Slide 1 -->

            <div class="carousel-item active" style="background-image: url(static/assets/img/slide/slide-
1.jpg)">

                <div class="container">

                    <h2>Welcome to <span>Covid center</span></h2>

                    <p>Bangalore bed booking System</p>

                    <a href="#about" class="btn-get-started scrollto">Read More</a>

                </div>

            </div>

            <!-- Slide 2 -->

            <div class="carousel-item" style="background-image: url(static/assets/img/slide/slide-2.jpg)">

```

```
<div class="container">
```

```
<h2>India</h2>
```

```
<p>Bangalore bed booking System.</p>
```

```
<a href="#about" class="btn-get-started scrollto">Read More</a>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<a class="carousel-control-prev" href="#heroCarousel" role="button" data-bs-slide="prev">
```

```
<span class="carousel-control-prev-icon bi bi-chevron-left" aria-hidden="true"></span>
```

```
</a>
```

```
<a class="carousel-control-next" href="#heroCarousel" role="button" data-bs-slide="next">
```

```
<span class="carousel-control-next-icon bi bi-chevron-right" aria-hidden="true"></span>
```

```
</a>
```

```
</div>
```

```
</section><!-- End Hero -->
```

```
<main id="main">
```

```
{% block body %}
```

```
{% endblock body %}
```

```
<footer id="footer">
```

```
<div class="container">
```

```
<div class="copyright">
```

```
&copy; Copyright <strong><span>AM Darshan</span></strong>. All Rights Reserved
```

```
</div>
```

```
<div class="credits">
```

```
Designed by <a href="https://github.com/AM-Darshan">AM Darshan</a>
```

```
</div>
```

```
</div>

</footer><!-- End Footer -->

<div id="preloader"></div>

<a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i

    class="bi bi-arrow-up-short"></i></a>

<!-- Vendor JS Files -->

<script src="static/assets/vendor/purecounter/purecounter.js"></script>

<script src="static/assets/vendor/aos/aos.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script src="static/assets/vendor/glightbox/js/glightbox.min.js"></script>

<script src="static/assets/vendor/swiper/swiper-bundle.min.js"></script>

<script src="static/assets/vendor/php-email-form/validate.js"></script>

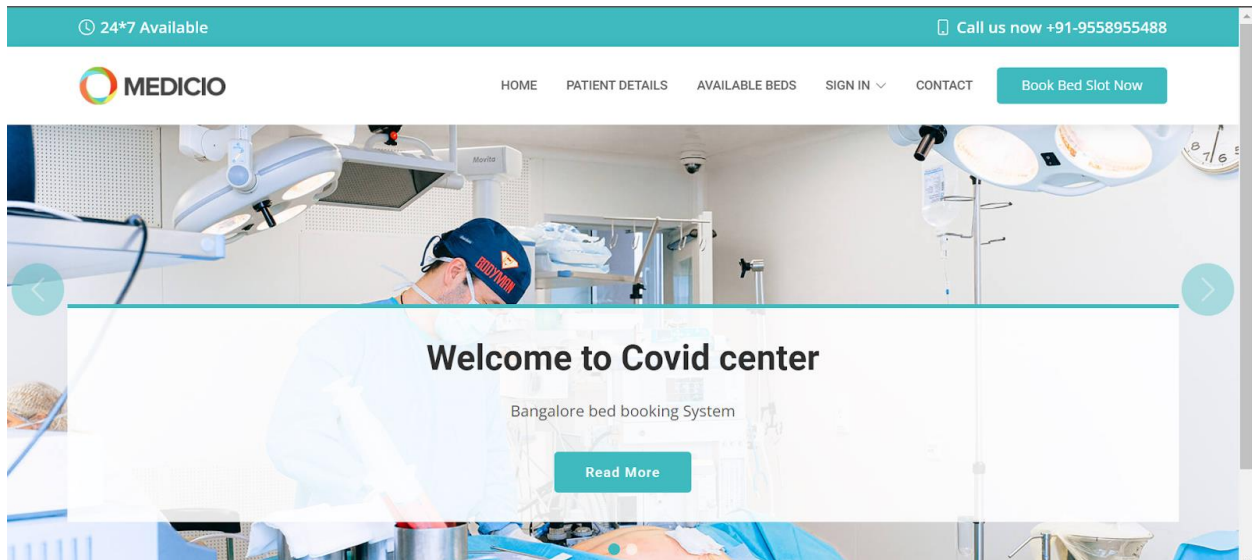
<!-- Template Main JS File -->

<script src="static/assets/js/main.js"></script>

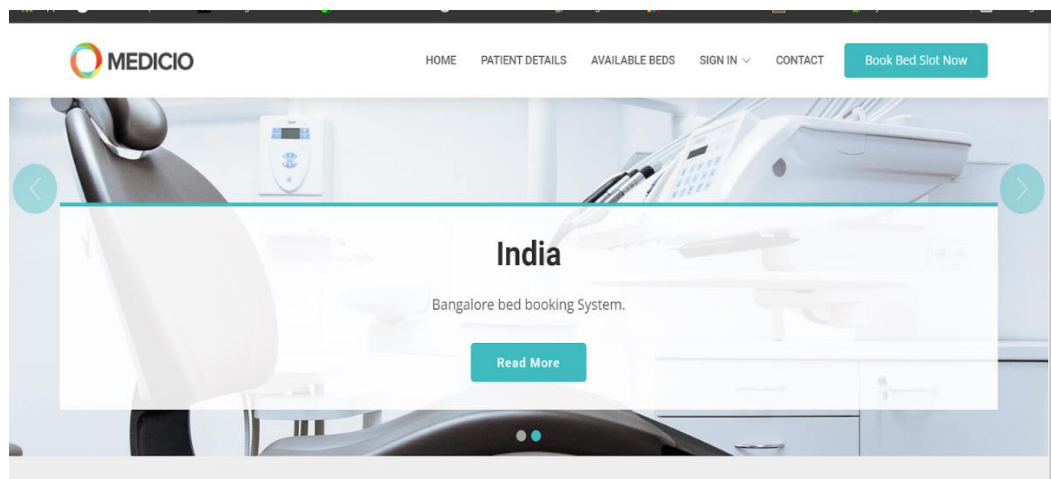
</body>

</html>
```

Snapshots




6.1 Home Page



6.2 Home Page Slider

🕒 24*7 Available

📞 Call us now +91-9558955488




HOME PATIENT DETAILS AVAILABLE BEDS SIGN IN ▾ CONTACT

Bed Slot Book Now

Sign Up

Enter the SRF ID

Enter email

dd-mm-yyyy 


SignIn

Click here to Login

6.3 User Sign Page

🕒 24*7 Available

📞 Call us now +91-9558955488




HOME PATIENT DETAILS AVAILABLE BEDS SIGN IN ▾ CONTACT

Bed Slot Book Now

Login

Enter the SRF ID

dd-mm-yyyy 


Login

New User ? SignUp

6.4 User login page

24*7 Available

Call us now +91-9558955488



HOME PATIENT DETAILS AVAILABLE BEDS SIGN IN CONTACT

Bed Slot Book Now

Hospital Login

Enter Email


Enter Password

Login

6.5 Hospital Login page

24*7 Available

Call us now +91-9558955488



HOME PATIENT DETAILS AVAILABLE BEDS SIGN IN CONTACT

Bed Slot Book Now

Admin Login

Enter the UserName

Enter password

Login

6.6 Admin Login Page

24*7 Available
Call us now +91-9558955488

HOME
PATIENT DETAILS
AVAILABLE BEDS
SIGN IN
CONTACT
Bed Slot Book Now

Add Hospital User

Logout

Enter Hospital Code

Enter Email

Enter Password

Add

6.7 Add Hospital user page

24*7 Available
Call us now +919986786453

HOME
PATIENTS DETAILS
AVAILABLE BEDS
WELCOME MYSpace.AMD@GMAIL.COM
CONTACT
Bed Booking Slot Book Now

Add Hospital Data

Covid Care Center

BBH01

Enter Hospital Name

Normal Bed Available?

H.I.C.U Bed Available?

I.C.U Bed Available?

Ventilators Bed Available?

Add

Hospital Data

Hospital Code : **BBH01**

Hospital Name : **KIMS**

Normal Beds Available :**15**

H.I.C.U Beds Available: **9**

I.C.U Beds Available :**4**


Ventilators Beds Available : **2**

6.8 Add Hospital Data page

Update Hospital Data

Covid Care Center

6.9 Hospital Data Update page

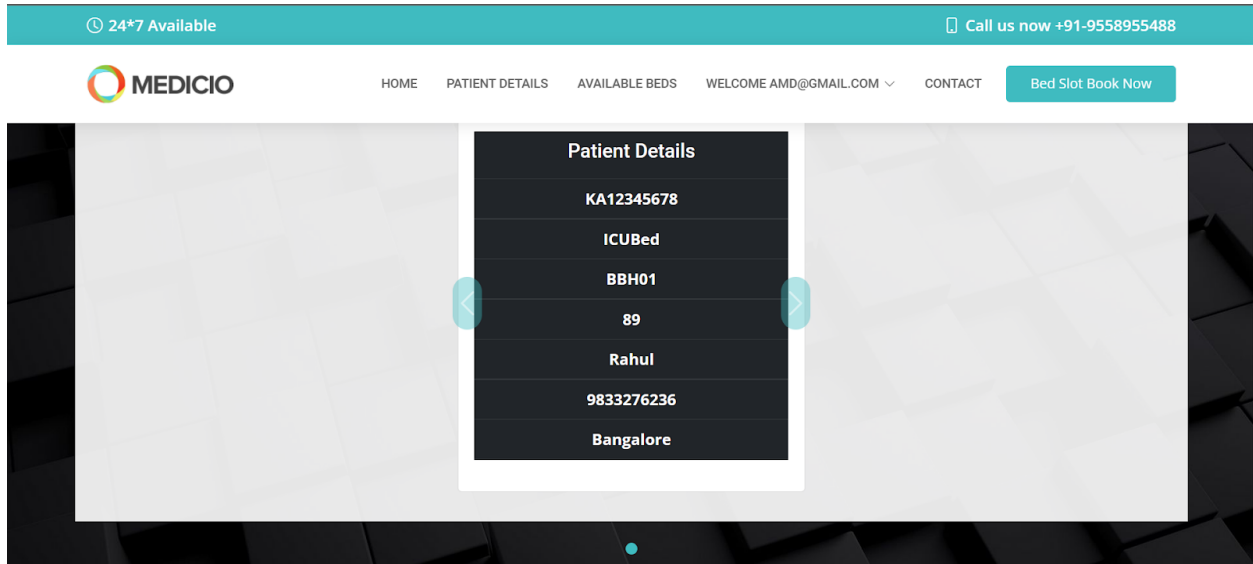
 [HOME](#) [PATIENTS DETAILS](#) [AVAILABLE BEDS](#) [WELCOME AMD@GMAIL.COM](#) [CONTACT](#) [Bed Booking Slot Book Now](#)

Bed Slot Booking

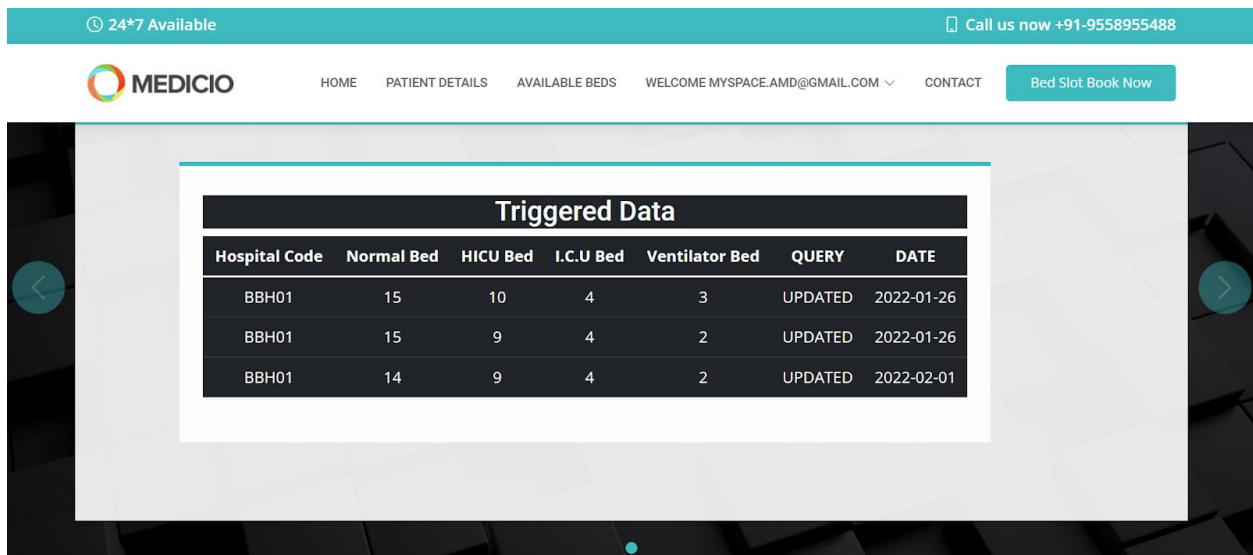
Covid Care Center

Available Beds					
Hospital Code	Hospital Name	Normal Bed	HICU Bed	I.C.U Bed	Ventilator Bed
BBH01	KIMS	14	9	4	2
BBA02	bangalore	30	19	5	7

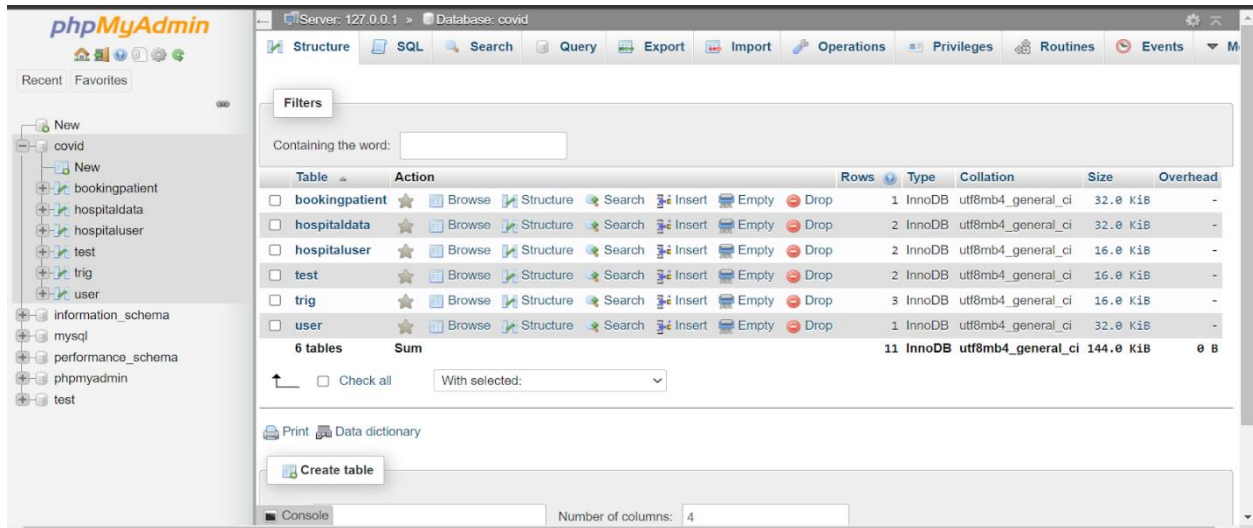
6.10 Bed Slot Booking page



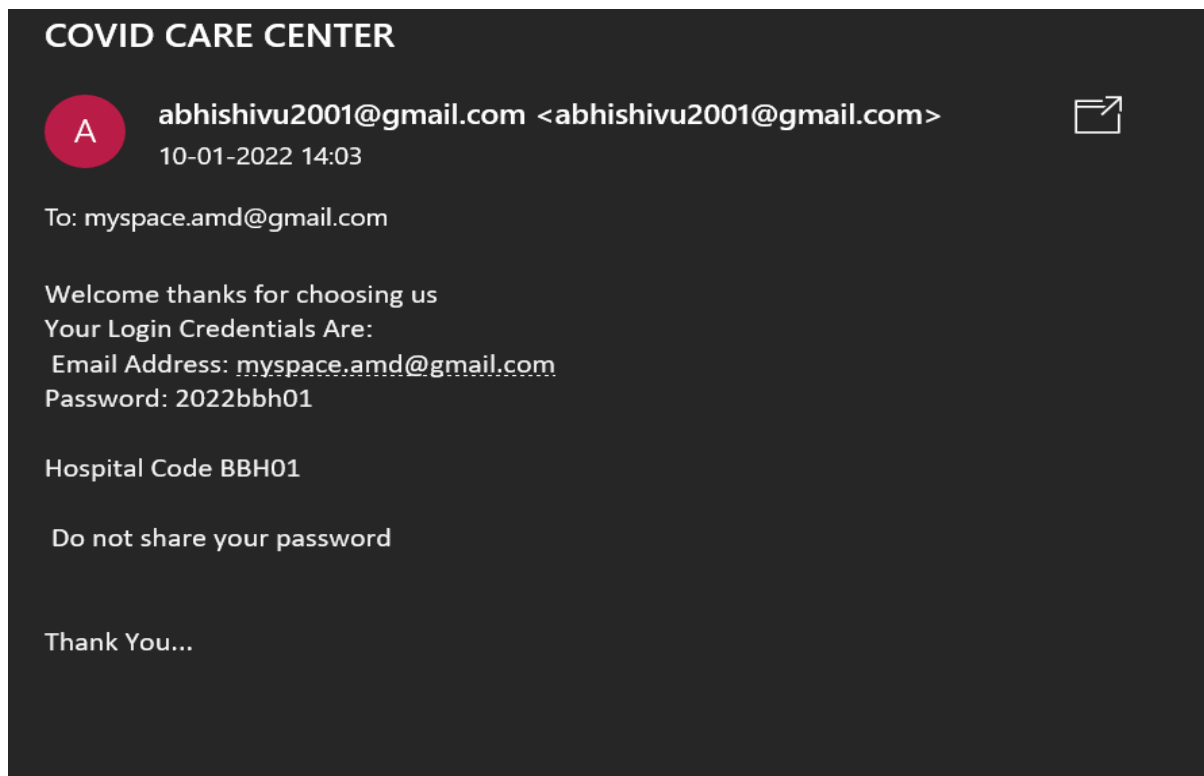
6.11 Patient Detail page



6.12 Triggered Data page



6.13 Xampp server Database page



6.14 Email

Applications

- Patients can schedule services in comfort of their home.
- Quick and easy booking of beds for patients.
- Online scheduling system operate 24/7.
- Online Booking reduces the appointment gaps and hence immediate concern can be shown for the patient.
- User-friendly and error free experience for the user booking the bed slot

Conclusion

The project work titled “Covid Bed Slot Booking System” has been designed using Python wherein many user-friendly form controls have been added in order to make it a user interactive application. The system is developed in such a way that the user with common knowledge of computers can handle it easily. The system forms a general platform for building a most advanced bed slot booking systems.

So, the Covid Bed Slot Booking System is mainly used to list the various hospitals, to store the hospital details such as types of beds available, slot bookings, triggered data and patient details.

REFERENCES

COVID-19 and Hospital Resource Management

- Source: World Health Organization (WHO)
- Link: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019>

Database Management for Healthcare Systems

- Source: MySQL Documentation
- Link: <https://dev.mysql.com/doc/>

Java for Web Development

- Source: Oracle Java Documentation
- Link: <https://docs.oracle.com/javase/>

Web Technologies for Responsive Applications

- Source: Mozilla Developer Network (MDN)
- Link: <https://developer.mozilla.org/en-US/>

XAMPP for Local Development

- Source: Apache Friends
- Link: <https://www.apachefriends.org/>

Emergency Management Systems during COVID-19

- Source: National Center for Biotechnology Information (NCBI)
- Link: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7199664/>

Digital Health Innovations in Pandemics

- Source: SpringerLink
- Link: <https://link.springer.com/>