

VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM 590014



A DBMS Mini-Project Report On

“COVID BED SLOT BOOKING SYSTEM”

A Mini-project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum.

Submitted by:

AMOGH S BHARADWAJ(1DT19CS013)

AND

VISHWAJIT H (1DT19CS051)

Under the Guidance of:

Prof. Keerthana Shankar

Asst. Professor, Dept of CSE



DAYANANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(Affiliated to Visvesvaraya Technological University, Belagavi and Approved by AICTE, New Delhi)

CE, CSE, ECE, EEE, ISE, ME Courses Accredited by NBA, New Delhi, NAAC A+



DAYANANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(Affiliated to Visvesvaraya Technological University, Belagavi and Approved by AICTE, New Delhi)

CE, CSE, ECE, EEE, ISE, ME Courses Accredited by NBA, New Delhi, NAAC A+

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project on Database Management System (DBMS) entitled “COVID BED SLOT BOOKING SYSTEM” has been successfully carried out by AMOGH S BHARADWAJ(1DT19CS013) and VISHWAJIT H(1DT19CS051) a Bonafede students of

Dayananda Sagar academy of technology and management in partial fulfilment of the requirements for the award of degree in Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during academic year 2018. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Prof. Keerthana Shankar

Asst. Professor, Dept of CSE

Dr. C. NANDINI

(Vice Principal & HOD, Dept. of CSE)

Examiners:

Signature with Date

1:

2:

ACKNOWLEDGEMENT

It gives us immense pleasure to present before you our project titled “**COVID BED SLOT MANAGEMENT SYSTEM**”. The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of those who made it possible. We are glad to express our gratitude towards our prestigious institution **DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT** for providing us with utmost knowledge, encouragement and the maximum facilities in undertaking this project.

We wish to express a sincere thanks to our respected principal **Dr.RAVISHANKAR** for all their support.

We express our deepest gratitude and special thanks to **Dr.C.Nandini**, Vice Principal & H.O.D, Dept. Of Computer Science Engineering, for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini- project guide Assistant professor **Prof. Keerthana Shankar**, Asst. Professor, Dept of CSE and Mini Project Coordinator **Dr.Prashanth C M**, Professor , Dept of CSE

AMOGH S BHARADWAJ(1DT19CS013)

VISHWAJIT H(1DT19CS051)

ABSTARCT

COVID BED SLOT BOOKING SYSTEM effectively manages to solve the real-world problem of booking bed in hospital. It integrates database and simpler UI to effectively manage and store the data of Hospitals and Patients.

It stores the Hospital data consisting of details related to hospital and number of beds available. This will be displayed to patients while booking a slot of bed and they can select the bed in hospital and thus data will be updated using triggers.

From this Admin, Government can effectively keep track number of beds available, number of patients admitted and their details such as SPO2 levels and derive a solution to control the spread of corona effectively

Table of Contents

CHAPTER 1: INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 PURPOSE	1
1.3 SCOPE	1
1.4 REFERENCES	1
CHAPTER 2: REQUIREMENTS	2
2.1. HARDWARE REQUIREMENTS	2
2.2. SOFTWARE REQUIRMENTS	2
2.3. FUNCTIONAL REQUIRMENTS	2
2.4. NON-FUNCTIONAL REQUIRMENTS	3
CHAPTER 3: DESIGN	4
3.1. SCHEMA DIAGRAM	4
3.2. E-R DIAGRAM	4
3.3. TABLE DESCRIPTION	5
3.3.1 <i>HOSPITAL USER</i>	5
3.3.2 <i>HOSPITAL DATA</i>	5
3.3.3 <i>TRIG</i>	5
3.3.4 <i>USER</i>	6
3.3.2 <i>BOOKING PATIENT</i>	6
CHAPTER 4: IMPLEMENTATION	7
4.1. FLOWCHART	7
4.2. ALGORITHM	8
4.3. FRONT-END DEVELOPMENT	8
4.3.1 <i>Hypertext Markup Language (HTML)</i>	8
4.3.2 <i>Cascading Style Sheets (CSS)</i>	8
4.3.3 <i>Bootstrap</i>	8
4.1.4 <i>JavaScript</i>	9
4.4. BACK-END DEVELOPMENT	9
4.4.1 <i>Python(flask)</i>	9
4.4.2 <i>Web Server APACHE</i>	9
4.4.3 <i>Database – MySQL</i>	9
CHAPTER 5: SOURCE CODE	10

5.1 PYTHON SOURECODE	10
5.2 HTML SOURECODE	22
CHAPTER 6: SNAPSHOT RESULT	27
6.1. HOME	27
6.2. ADMIN LOGIN	28
6.3. ADD HOSPITAL USER	29
6.4. EMAIL AUTHORIZATION	29
6.5. HOSPITAL LOGIN	29
6.6. ADD HOSPITAL DATA	30
6.7. PATIENT SIGNUP	30
6.8. PATIENT LOGIN	31
6.9. BOOK BED SLOT	31
6.10. TRIGGERS	31
CHAPTER 7: CONCLUSION AND MERITS	32
7.1. CONCLUSION	32
7.2. MERITS	32
7.3. FUTURE WORKS	32
CHAPTER 8: BILBOGRAPHY	33
8.1 BOOK REFERENCES:	33
8.2 WEBSITE REFERENCES:	33
1. <i>HTML LEARNING:</i>	33
2. <i>PYTHON LEARNING:</i>	33
PERSONAL DETAILS	34

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

This project is designed so as to be used by Patient to book bed in the hospital. It is an online system through which patients can register, view available hospitals, view available beds and book them.

1.2 PURPOSE

- The purpose of this project is to provide a friendly environment to maintain the details of patients and hospital.
- The main purpose of this project is to show the availability of beds to public sector to overcome the pandemic
- This is an attempt to stop people from jumping queue and bringing transparency to bed allocating beds.
- Easy accessibility for hospital management as data is available as a point source, so the decision can be made faster and effective in admission of covid patients.

1.3 SCOPE

- The document only covers the requirements specifications for the Covid Bed Slot Booking System.
- This document does not provide any references to the other component of the Covid Bed Slot Booking System.
- All the external interfaces and the dependencies are also identified in this document.
- The overall scope of the feasibility study is to provide sufficient information to allow a decision to be made as to whether the Covid Bed Slot Booking project should proceed and if so, its relative priority in the context of other existing Covid Bed Slot Booking Technology

1.4 REFERENCES

- Fundamentals of Database System by Elmasri
- Fundamentals of Software Engineering by Rajib Mall

CHAPTER 2: REQUIREMENTS

2.1. HARDWARE REQUIREMENTS

- Processor - Intel 486/Pentium processor or better
- Processor Speed - 500 MHz or above
- Hard Disk - 20GB(approx.)
- RAM - 64MB or above
- Storage Space - Approx. 2MB

2.2. SOFTWARE REQUIRMENTS

- Technology Implemented : Apache Server, MySQL Server
- Language Used : Python
- Database : My SQL
- User Interface Design : HTML, CSS, Bootstrap.
- Web Browser : Google Chrome, Firefox
- Software : XAMPP Version: 7.1.10

2.3. FUNCTIONAL REQUIRMENTS

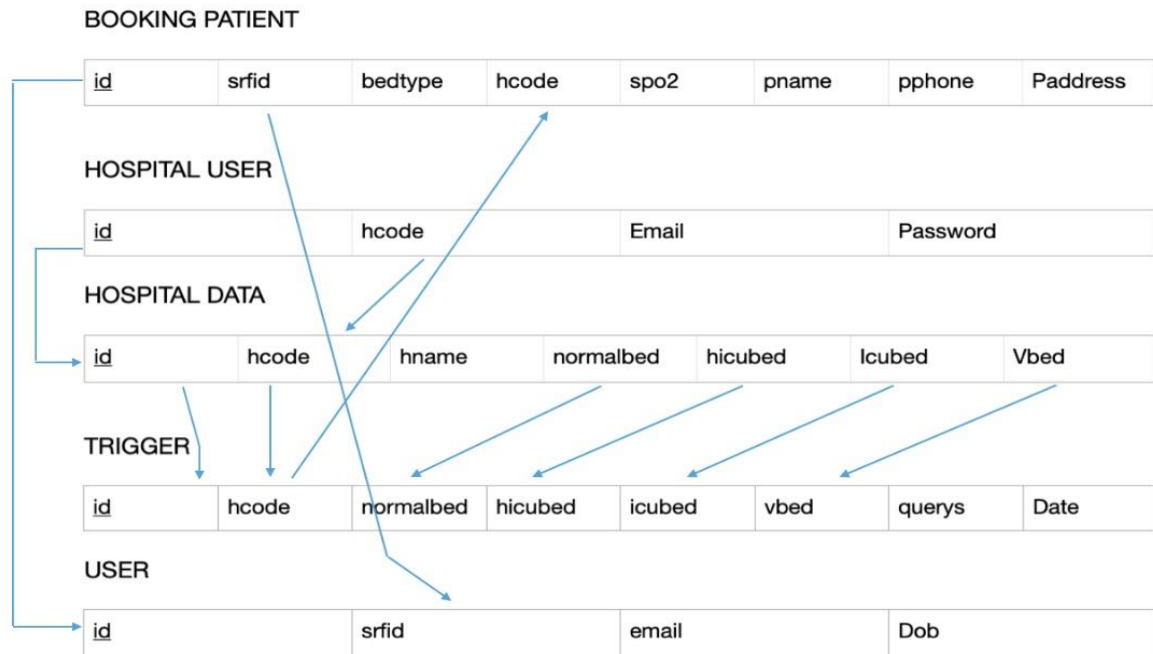
1. The CBSBS should store Login Information of:
 - Admin
 - Hospital User
 - User
2. The CBSBS should able to create new Hospital User through Admin Portal
3. The CBSBS should store all information about Hospital Data such as Hospital Code, Hospital Name, Number of Beds from Hospital User Portal
4. The CBSBS should enable Hospital user should be able to update their data.
5. The CBSBS should enable user to create account based on their SRFID and login.
6. The CBSBS should allow users to view the list of Hospital and beds available and book the bed slot from any mentioned hospitals.
7. The admin should be able to view recent operations performed.

2.4. NON-FUNCTIONAL REQUIRMENTS

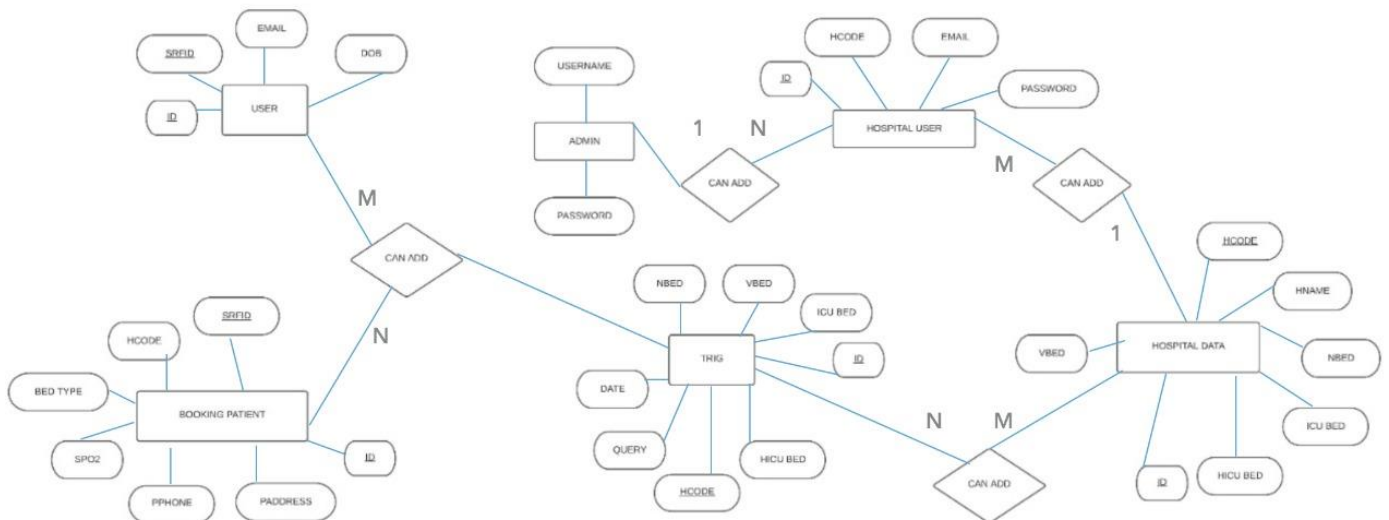
- 1. Usability Requirements** - The user interface should be interactive, simple and easy to understand. The system should prompt for the user and administrator to login to the application for proper input criteria.
- 2. Error Handling** - CBSBS shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period.
- 3. Security Requirements** - CBSBS should provide databases modification only for the ADMIN and HOSPITAL USER after proper authorization.
- 4. Performance and Response time** -The system should have high performance rate when executing user's input and should be able to provide feedback or response within a short time span usually 50 seconds for highly complicated task and 20 to 25 seconds for less complicated task.
- 5. Availability** - This system should always be available for access at 24 hours, 7 days a week. Also, in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.
- 6. Ease of use** - Considered the level of knowledge possessed by the users of this system, a simple but quality user interface should be developed to make it easy to understand and required less training.

CHAPTER 3: DESIGN

3.1. SCHEMA DIAGRAM




3.2. E-R DIAGRAM





3.3. TABLE DESCRIPTION


3.3.1 HOSPITAL USER

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<u>id</u> 	int(11)			No	None	PRIMARY KEY	AUTO_INCREMENT
2	hcode	varchar(20)	utf8mb4_general_ci		No	None		
3	email	varchar(100)	utf8mb4_general_ci		No	None		
4	password	varchar(1000)	utf8mb4_general_ci		No	None		


3.3.2 HOSPITAL DATA

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<u>id</u> 	int(11)			No	None	PRIMARY KEY	AUTO_INCREMENT
2	hcode 	varchar(200)	utf8mb4_general_ci		No	None		
3	hname	varchar(200)	utf8mb4_general_ci		No	None		
4	normalbed	int(11)			No	None		
5	hicubed	int(11)			No	None		
6	icubed	int(11)			No	None		
7	vbed	int(11)			No	None		



3.3.3 TRIG

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<u>id</u> 	int(11)			No	None	PRIMARY KEY	AUTO_INCREMENT
2	hcode	varchar(50)	utf8mb4_general_ci		No	None		
3	normalbed	int(11)			No	None		
4	hicubed	int(11)			No	None		
5	icubed	int(11)			No	None		
6	vbed	int(11)			No	None		
7	querys	varchar(50)	utf8mb4_general_ci		No	None		
8	date	date			No	None		

3.3.4 USER

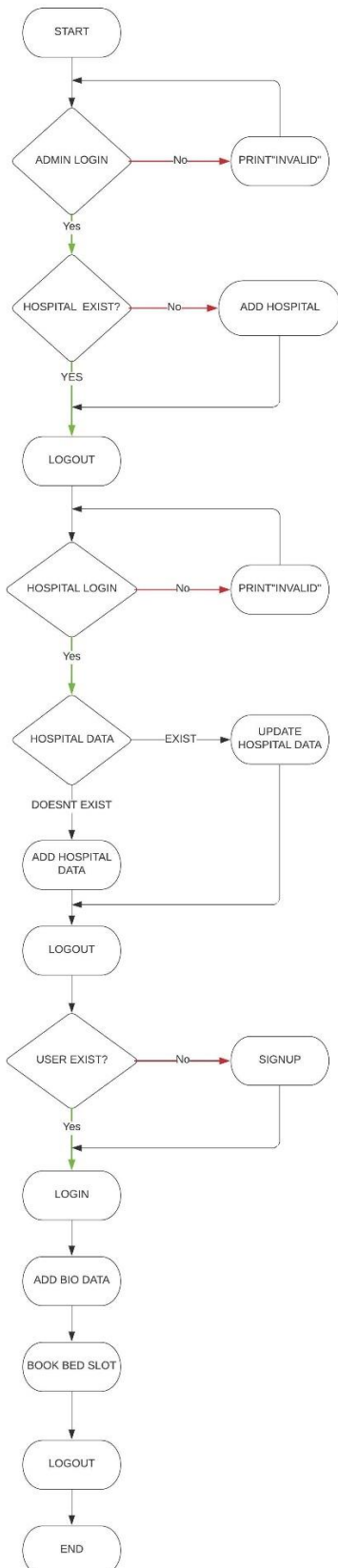
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<u>id</u> 	int(11)			No	None	PRIMARY KEY	AUTO_INCREMENT
2	srfid 	varchar(20)	utf8mb4_general_ci		No	None		
3	email	varchar(100)	utf8mb4_general_ci		No	None		
4	dob	varchar(1000)	utf8mb4_general_ci		No	None		

3.3.2 BOOKING PATIENT

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<u>id</u> 	int(11)			No	None	PRIMARY KEY	AUTO_INCREMENT
2	srfid 	varchar(50)	utf8mb4_general_ci		No	None		
3	bedtype	varchar(50)	utf8mb4_general_ci		No	None		
4	hcode	varchar(50)	utf8mb4_general_ci		No	None		
5	spo2	int(11)			No	None		
6	pname	varchar(50)	utf8mb4_general_ci		No	None		
7	pphone	varchar(12)	utf8mb4_general_ci		No	None		
8	paddress	text	utf8mb4_general_ci		No	None		

CHAPTER 4: IMPLEMENTATION

4.1. FLOWCHART



4.2. ALGORITHM

Step 1: Login with admin credentials

Step 2: Add a hospital and their credentials

Step 3: Login with Hospital Credentials

Step 4: Add Hospital data such as no of beds, hospital name and hospital address

Step 5: Create a login credentials for new user using SRFID

Step 6: Login with created credentials

Step 7: Enter the bio data

Step 8: Select a bed from given list of Hospitals

4.3. FRONT-END DEVELOPMENT

The front-end is built using a combination of technologies such as Hypertext Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS). Front-end developers design and construct the user experience elements on the web page or app including buttons, menus, pages, links, graphics and more.

4.3.1 Hypertext Markup Language (HTML)

HTML is a computer language devised to allow website creation. These websites can then be viewed by anyone else connected to the Internet. HTML is the standard markup language for creating Web pages. It stands for Hyper Text Markup Language. It describes the structure of a Web page. It consists of a series of elements. Its elements tell the browser how to display the content. HTML elements are the building blocks of HTML pages.

4.3.2 Cascading Style Sheets (CSS)

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications.

4.3.3 Bootstrap

Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML- and CSS based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. The bootstrap framework aims to ease web development. Bootstrap is a front end, that is an interface between the user and the server-side code which resides on the "back end" or server. And it is a web application framework, that is a software framework which is designed to support the development of dynamic websites and web applications.

4.1.4 JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. Client-side JavaScript is the most common form of the language.

4.4. BACK-END DEVELOPMENT

Backend is server side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users.

4.4.1 Python(flask)

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. We use jinja in this project. Jinja is a template engine for the Python programming language and is licensed under a BSD License

4.4.2 Web Server APACHE

Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. It allows website owners to serve content on the web — hence the name “web server”. Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server. Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). Apache is a cross-platform software; therefore, it works on both UNIX and Windows servers.

4.4.3 Database – MySQL

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. It is released under an open-source license. It handles a large subset of the functionality of the most expensive and powerful database packages. It uses a standard form of the well-known SQL data language. It works on many operating systems. It works very quickly and works well even with large data sets. MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this.

CHAPTER 5: SOURCE CODE

5.1 PYTHON SOURCECODE

```
#WELCOME TO COVID BED SLOT ALLOTMENT PROJECT
#PROJECT CREATED BY
# 1DT19CS013 AMOGH S BHARADWAJ
# 1DT19CS051 VISHWAJIT H
#Importing libraries
from enum import unique
from flask import Flask,redirect,render_template,request,json,flash
from flask.globals import request,session
from flask_login.utils import login_user
from flask.helpers import url_for
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from flask_login import
login_required,logout_user,login_manager,LoginManager,current_user
from sqlalchemy import engine
from sqlalchemy.engine.base import Connection
from werkzeug.security import generate_password_hash,check_password_hash
from flask_mail import Mail
import json
# mydatabase connection
local_server=True
app=Flask(__name__)
app.secret_key="amoghvishwajit"
```

```
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/covid'

db=SQLAlchemy(app)

#config.json integration

with open('config.json','r') as c:

    params=json.load(c)["params"]

#sendingmail

app.config.update(

    MAIL_SERVER='smtp.gmail.com',

    MAIL_PORT='465',

    MAIL_USE_SSL=True,

    MAIL_USERNAME=params['gmail-user'],

    MAIL_PASSWORD=params['gmail-password']

)

mail = Mail(app)

# this is for getting the unique user access

login_manager=LoginManager(app)

login_manager.login_view='login'

@login_manager.user_loader

def load_user(user_id):

    return User.query.get(int(user_id)) or Hospitaluser.query.get(int(user_id))

#patient database model

class User(UserMixin,db.Model):

    id=db.Column(db.Integer,primary_key=True)

    srfid=db.Column(db.String(20),unique=True)

    email=db.Column(db.String(50))

    dob=db.Column(db.String(1000))
```

#hospitaluser database model

```
class Hospitaluser(UserMixin,db.Model):  
    id=db.Column(db.Integer,primary_key=True)  
    hcode=db.Column(db.String(200))  
    email=db.Column(db.String(100))  
    password=db.Column(db.String(1000))
```

#hospital database model

```
class Hospitaldata(db.Model):  
    id=db.Column(db.Integer,primary_key=True)  
    hcode=db.Column(db.String(50),unique=True)  
    hname=db.Column(db.String(50))  
    normalbed=db.Column(db.Integer)  
    hicubed=db.Column(db.Integer)  
    icubed=db.Column(db.Integer)  
    vbed=db.Column(db.Integer)
```

#trigger databasemodel

```
class Trig(db.Model):  
    id=db.Column(db.Integer,primary_key=True)  
    hcode=db.Column(db.String(20))  
    normalbed=db.Column(db.Integer)  
    hicubed=db.Column(db.Integer)  
    icubed=db.Column(db.Integer)  
    vbed=db.Column(db.Integer)  
    querys=db.Column(db.String(50))  
    date=db.Column(db.String(50))
```

#bookingpatient databasemodel

```
class Bookingpatient(db.Model):
```

```
id=db.Column(db.Integer,primary_key=True)
srfid=db.Column(db.String(20),unique=True)
bedtype=db.Column(db.String(100))
hcode=db.Column(db.String(20))
spo2=db.Column(db.Integer)
pname=db.Column(db.String(100))
pphone=db.Column(db.String(100))
paddress=db.Column(db.String(100))

#routing

#home

@app.route("/")

def home():

    return render_template("index.html")

#signup

@app.route('/signup',methods=['POST','GET'])

def signup():

    if request.method=="POST":

        srfid=request.form.get('srf')
        email=request.form.get('email')
        dob=request.form.get('DOB')
        encpassword=generate_password_hash(dob)
        user=User.query.filter_by(srfid=srfid).first()
        emailUser=User.query.filter_by(email=email).first()
        if user or emailUser:

            flash("Email or srfid is already taken","warning")

            return render_template("usersignup.html")

        new_user=db.engine.execute(f"INSERT INTO `user` (`srfid`,`email`,`dob`)
VALUES ('{srfid}','{email}','{encpassword}') ")
```

```
flash("SignUp Success Please Login","success")
return render_template("userlogin.html")
return render_template("usersignup.html")
#login
@app.route('/login',methods=['POST','GET'])
def login():
    if request.method=="POST":
        srfid=request.form.get('srf')
        dob=request.form.get('DOB')
        user=User.query.filter_by(srfid=srfid).first()
        if user and check_password_hash(user.dob,dob):
            login_user(user)
            flash("Login Success","info")
            return render_template("index.html")
        else:
            flash("Invalid Credentials","danger")
            return render_template("userlogin.html")
    return render_template("userlogin.html")
#HospitalUserlogin
@app.route('/hospitallogin',methods=['POST','GET'])
def hospitallogin():
    if request.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=Hospitaluser.query.filter_by(email=email).first()
        if user and check_password_hash(user.password,password):
            login_user(user)
```

```
        flash("Login Success","info")
        return render_template("index.html")
    else:
        flash("Invalid Credentials","danger")
        return render_template("hospitallogin.html")
    return render_template("hospitallogin.html")
#addhospitalinfo
@app.route("/addhospitalinfo",methods=['POST','GET'])
def addhospitalinfo():
    email=current_user.email
    posts=Hospitaluser.query.filter_by(email=email).first()
    code=posts.hcode
    postsdata=Hospitaldata.query.filter_by(hcode=code).first()
    if request.method=="POST":
        hcode=request.form.get('hcode')
        hname=request.form.get('hname')
        nbed=request.form.get('normalbed')
        hbed=request.form.get('hicubed')
        ibed=request.form.get('icubed')
        vbed=request.form.get('vbed')
        hcode=hcode.upper()
        huser=Hospitaluser.query.filter_by(hcode=hcode).first()
        hduser=Hospitaldata.query.filter_by(hcode=hcode).first()
        if hduser:
            flash("Data is already Present you can update it..","primary")
            return render_template("hospitaldata.html")
        if huser:
```

```
db.engine.execute(f"INSERT INTO `hospitaldata`  
(`hcode`,`hname`,`normalbed`,`hicubed`,`icubed`,`vbed`) VALUES  
('{hcode}','{hname}','{nbed}','{hbed}','{ibed}','{vbed}')
```

```
flash("Data Is Added","primary")
```

```
else:
```

```
flash("Hospital Code not Exist","warning")
```

```
return render_template("hospitaldata.html",postsdata=postsdata)
```

```
#adminlogin
```

```
@app.route('/admin',methods=['POST','GET'])
```

```
def admin():
```

```
    if request.method=="POST":
```

```
        username=request.form.get('username')
```

```
        password=request.form.get('password')
```

```
        if(username==params['user'] and password==params['password']):
```

```
            session['user']=username
```

```
            flash("login success","info")
```

```
            return render_template("addHosUser.html")
```

```
        else:
```

```
            flash("Invalid Credentials","danger")
```

```
        return render_template("admin.html")
```

```
#addhospitaluser
```

```
@app.route('/addHospitalUser',methods=['POST','GET'])
```

```
def hospitalUser():
```

```
    if('user' in session and session['user']==params['user']):
```

```
        if request.method=="POST":
```

```
            hcode=request.form.get('hcode')
```

```
            email=request.form.get('email')
```

```
            password=request.form.get('password')
```

```
encpassword=generate_password_hash(password)

hcode=hcode.upper()

emailUser=Hospitaluser.query.filter_by(email=email).first()

if emailUser:

    flash("Email or srif is already taken","warning")

    db.engine.execute(f"INSERT INTO `hospitaluser`
(`hcode`,`email`,`password`) VALUES ('{hcode}','{email}','{encpassword}') ")

    mail.send_message('COVID CARE CENTER',sender=params['gmail-
user'],recipients=[email],body=f"Welcome thanks for choosing us\nYour Login
Credentials Are:\n Email Address: {email}\nPassword: {password}\n\nHospital
Code {hcode}\n\n Do not share your password\n\nThank You..." )

    flash("Data Sent and Inserted Successfully","warning")

    return render_template("addHosUser.html")

else:

    flash("Login and try Again","warning")

    return render_template("addHosUser.html")

#adminlogout

@app.route("/logoutadmin")

def logoutadmin():

    session.pop('user')

    flash("You are logout admin", "primary")

    return redirect('/admin')

#logout

@app.route('/logout')

@login_required

def logout():

    logout_user()

    flash("Logout SuccessFul","warning")

    return redirect(url_for('login'))
```

```
#triggers
```

```
@app.route("/triggers")
```

```
def triggers():
```

```
    query=Trig.query.all()
```

```
    return render_template("triggers.html",query=query)
```

```
#update
```

```
@app.route("/hedit/<string:id>",methods=['POST','GET'])
```

```
@login_required
```

```
def hedit(id):
```

```
    posts=Hospitaldata.query.filter_by(id=id).first()
```

```
    if request.method=="POST":
```

```
        hcode=request.form.get('hcode')
```

```
        hname=request.form.get('hname')
```

```
        nbed=request.form.get('normalbed')
```

```
        hbed=request.form.get('hicubed')
```

```
        ibed=request.form.get('icubed')
```

```
        vbed=request.form.get('vbed')
```

```
        hcode=hcode.upper()
```

```
        db.engine.execute(f"UPDATE `hospitaldata` SET `hcode`  
='{hcode}',`hname`='{hname}',`normalbed`='{nbed}',`hicubed`='{hbed}',`icubed`  
='{ibed}',`vbed`='{vbed}' WHERE `hospitaldata`.`id`={id}")
```

```
        flash("Slot Updated","info")
```

```
        return redirect("/addhospitalinfo")
```

```
# posts=Hospitaldata.query.filter_by(id=id).first()
```

```
return render_template("hedit.html",posts=posts)
```

```
#deletion
```

```
@app.route("/hdelete/<string:id>",methods=['POST','GET'])
```

```
@login_required
```

```
def hdelete(id):
```

```
    db.engine.execute(f"DELETE FROM `hospitaldata` WHERE  
`hospitaldata`.`id`={id}")
```

```
    flash("Date Deleted","danger")
```

```
    return redirect("/addhospitalinfo")
```

```
#slotbooking
```

```
@app.route("/slotbooking",methods=['POST','GET'])
```

```
@login_required
```

```
def slotbooking():
```

```
    query=db.engine.execute(f"SELECT * FROM `hospitaldata` ")
```

```
    if request.method=="POST":
```

```
        srfid=request.form.get('srfid')
```

```
        bedtype=request.form.get('bedtype')
```

```
        hcode=request.form.get('hcode')
```

```
        spo2=request.form.get('spo2')
```

```
        pname=request.form.get('pname')
```

```
        pphone=request.form.get('pphone')
```

```
        paddress=request.form.get('paddress')
```

```
        check2=Hospitaldata.query.filter_by(hcode=hcode).first()
```

```
        if not check2:
```

```
            flash("Hospital Code not exist","warning")
```

```
        code=hcode
```

```
        dbb=db.engine.execute(f"SELECT * FROM `hospitaldata` WHERE  
`hospitaldata`.`hcode`='{code}' ")
```

```
        bedtype=bedtype
```

```
        if bedtype=="NormalBed":
```

```
    for d in dbb:
        seat=d.normalbed
        print(seat)
        ar=Hospitaldata.query.filter_by(hcode=code).first()
        ar.normalbed=seat-1
        db.session.commit()
elif bedtype=="HICUBed":
    for d in dbb:
        seat=d.hicubed
        print(seat)
        ar=Hospitaldata.query.filter_by(hcode=code).first()
        ar.hicubed=seat-1
        db.session.commit()
elif bedtype=="ICUBed":
    for d in dbb:
        seat=d.icubed
        print(seat)
        ar=Hospitaldata.query.filter_by(hcode=code).first()
        ar.icubed=seat-1
        db.session.commit()
elif bedtype=="VENTILATORBed":
    for d in dbb:
        seat=d.vbed
        ar=Hospitaldata.query.filter_by(hcode=code).first()
        ar.vbed=seat-1
        db.session.commit()
else:
```

```
pass

check=Hospitaldata.query.filter_by(hcode=hcode).first()

if(seat>0 and check):
res=Bookingpatient(srfid=srfid,bedtype=bedtype,hcode=hcode,spo2=spo2,pname
=pname,pphone=pphone,paddress=paddress)

    db.session.add(res)

    db.session.commit()

    flash("Slot is Booked kindly Visit Hospital for Further
Procedure","success")

else:

    flash("Something Went Wrong","danger")

return render_template("booking.html",query=query)

# patient details

@app.route("/pdetails",methods=['GET'])

@login_required

def pdetails():

    code=current_user.srfid

    print(code)

    data=Bookingpatient.query.filter_by(srfid=code).first()

    return render_template("details.html")

# testing wheather db is connected or not

#mydatabase models

class Test(db.Model):

    id=db.Column(db.Integer,primary_key=True)

    name=db.Column(db.String(50))

    db.create_all()

@app.route("/test")

def test():
```

```
try:
    a=Test.query.all()
    print(a)
    return f'MY DATABASE IS CONNECTED'
except Exception as e:
    print(e)
    return f'MY DATABASE IS NOT CONNECTED {e}'
app.run(debug=True)
```

5.2 HTML SOURECODE

```
<!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,60
```

```
0,600i,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 6969696969
</div>
</div>
</div>
<!-- ===== Header ===== -->
<header id="header" class="fixed-top">
  <div class="container d-flex align-items-center">
    <a href="index.html" class="logo me-auto"></a>

    <nav id="navbar" class="navbar order-last order-lg-0">
      <ul>
        <li><a class="nav-link scrollto " href="/">Home</a></li>
        <li><a class="nav-link scrollto "
href="/slotbooking">AvailableBeds</a></li>
        { % if current_user.is_authenticated % }
        <li class="dropdown"><a href="/login"><span>Welcome
{ {current_user.email} }</span> <i class="bi bi-chevron-down"></i></a>
          <ul>
            <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 %}

</ul>

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

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

    <!--<a href="#appointment" class="appointment-btn scrollto"><span
class="d-none d-md-inline">Bed Booking Slot</span> Book 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">

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

                <div class="container">

{% block form %}

{% endblock form %}

                </div>

            </div>

        </div>

    </div>

```

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

<main id="main">

</main><!-- End #main -->


<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>
```


CHAPTER 6: SNAPSHOT RESULT

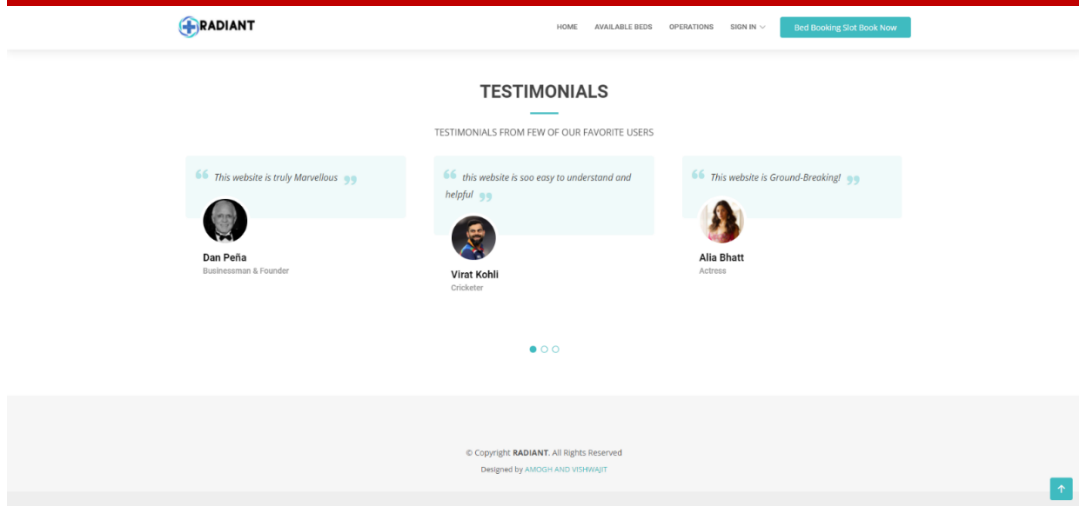
6.1. HOME

The image displays three sequential screenshots of the RADIANT website's home page, illustrating its layout and content.

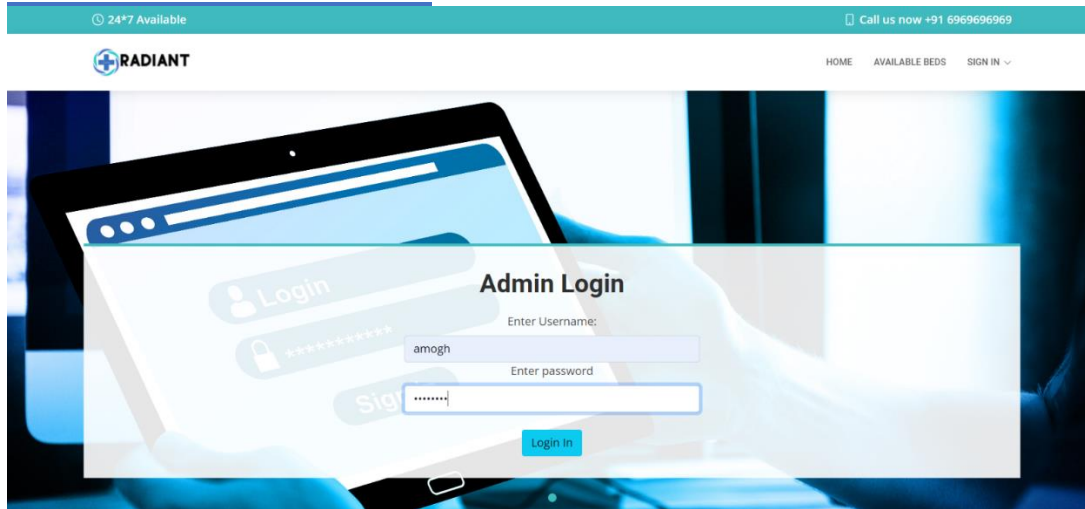
Top Screenshot (Home Section): The header features the RADIANT logo, navigation links (HOME, AVAILABLE BEDS, OPERATIONS, SIGN IN), and a "Bed Booking Slot Book Now" button. The main content area includes four cards with icons and text: "MAINTAIN SOCIAL DISTANCING" (2 meters), "WEAR MASK" (all the time), "SANITIZE HANDS" (avoid touching nose, eyes, ears), and "FOLLOW COVID RULES" (follow government regulations).

Middle Screenshot (Frequently Asked Questions Section): This section is titled "FREQUENTLY ASKED QUESTIONS" and includes a sub-header "Here are Frequently asked Questions Regarding COVID-19 and Radiant website". It lists several questions, such as "What is Covid-19?", "How long do COVID-19 symptoms take to appear?", "How long do Covid symptoms last?", "What are the symptoms of Covid-19?", "Does Booking Bed through Radiant website incurs any charges?", and "What are Requirements to book Bed Slot Through Radiant?".

Bottom Screenshot (About Us Section): This section is titled "ABOUT US" and includes a sub-header "This Website is Designed By Vishwajit.H and Amogh.S.Bharadwaj, students of DSATM". It features a large black square with the RADIANT logo and tagline "SENDING LIGHT TO LIVES". To the right, it provides "SOME INSIGHT ABOUT THIS" project, stating it aims to solve the real-world problem of bed slot allotment. It lists three user types: "USER WHO WANT TO VIEW/BOOK BEDS(PATIENT)", "USER WHO WANTS TO UPDATE BED(HOSPITAL)", and "USER WHO MANAGES (ADMIN)". It also mentions database operations: "AUTHENTICATION, CRUD OPERATIONS(CREATION,RENAMING,UPDATION,DELETION) AND ALSO TRIGGERS TO AUTOMATICALLY UPDATE THE DATABASE".



6.2. ADMIN LOGIN



6.3. ADD HOSPITAL USER

24*7 Available Call us now +91 6969696969

RADIANT HOME AVAILABLE BEDS SIGN IN

Add Hospital User

login success

Logout Login


Enter HCODE:
BLORE003

Enter Email:
amoghrock16@gmail.com

Enter Password:

Submit

6.4. EMAIL AUTHORIZATION

 **amoghvishwajit@gmail.com** to me Sat, 5 Feb, 20:49 (3 days ago) ☆ ↶ ⋮

Welcome thanks for choosing us
Your Login Credentials Are:
Email Address: amoghsai1665@gmail.com
Password: AMOGH

Hospital Code BLORE001

Do not share your password

Thank You...

↶ Reply ➦ Forward

6.5. HOSPITAL LOGIN

24*7 Available Call us now +91 6969696969

RADIANT HOME AVAILABLE BEDS SIGN IN

Hospital Login

Enter EMAIL ID:
amoghrock16@gmail.com

Enter Password:

Login

6.6. ADD HOSPITAL DATA


24*7 Available
Call us now +919986786453

RADIANT
HOME AVAILABLE BEDS WELCOME AMOGHROCK16@GMAIL.COM Bed Booking Slot Book Now

Add Hospital Data

Covid Care Center

Hospital Code
BLORE003
Hospital name
sagar
Normal beds
60
HICU beds
50
ICU beds
20
Ventilator beds
06
Add



Hospital Data

Hospital Code :
Hospital Name :
Normal Beds Available :
H.I.C.U Beds Available :
I.C.U Beds Available :
Ventilators Beds Available :


24*7 Available
Call us now +919986786453

RADIANT
HOME AVAILABLE BEDS WELCOME AMOGHROCK16@GMAIL.COM Bed Booking Slot Book Now

Add Hospital Data

Covid Care Center

Hospital Code
BLORE003
Hospital name
Enter the hospital name
Normal beds
Enter the no of Normal Beds
HICU beds
Enter the no of HICU Beds
ICU beds
Enter the no of ICU Beds
Ventilator beds
Enter the no of Ventilator Beds
Add



Hospital Data

Hospital Code :BLORE003
Hospital Name :sagar
Normal Beds Available :60
H.I.C.U Beds Available :50
I.C.U Beds Available :20
Ventilators Beds Available : 6

6.7. PATIENT SIGNUP

24*7 Available
Call us now +91 6969696969

RADIANT
HOME AVAILABLE BEDS SIGN IN

Sign Up

Enter SRF ID:
1dt19cs054
Enter EMAIL ID:
Hrzc204@gmail.com
Enter DOB:
11/16/2000
Signin
Click Here to Login

6.8. PATIENT LOGIN

24*7 Available
Call us now +91 6969696969

RADIANT
HOME AVAILABLE BEDS SIGN IN

Login

Enter SRF ID:

Enter DOB:

New User? Signup

6.9. BOOK BED SLOT

24*7 Available
Call us now +919986786453

RADIANT
HOME AVAILABLE BEDS WELCOME HRZC204@GMAIL.COM
[Bed Booking Slot Book Now](#)

Book Bed Slot

Covid Care Center

1dt19cs054
VENTILATOR Bed
BLORE003
90
harsh
198745612
MANGALORE

Available Beds					
Hospital Code	Hospital Name	Normal Bed	HICU Bed	I.C.U Bed	Ventilator Bed
BLORE001	victoria	19	30	40	49
BLORE002	kims	49	60	40	80
BLORE003	sagar	60	50	20	6

6.10. TRIGGERS

24*7 Available
Call us now +91 6969696969

RADIANT
HOME AVAILABLE BEDS SIGN IN

BLORE001	50	60	50	20	INSERTED	2022-02-05
BLORE001	50	0	0	0	UPDATED	2022-02-05
BLORE001	50	0	0	0	UPDATED	2022-02-05
BLORE001	50	0	0	0	UPDATED	2022-02-05
BLORE001	50	0	0	0	DELETED	2022-02-05
BLORE001	20	30	40	50	INSERTED	2022-02-05
BLORE001	20	30	40	49	UPDATED	2022-02-05
BLORE001	19	30	40	49	UPDATED	2022-02-08
BLORE002	50	60	40	80	INSERTED	2022-02-08
BLORE002	49	60	40	80	UPDATED	2022-02-08

CHAPTER 7: CONCLUSION AND MERITS

7.1. CONCLUSION

The Covid Bed Slot Booking is a great improvement over the manual system which uses lots of manual work. The computerization of the system speeds up the process. This system was thoroughly checked and tested with dummy data and found to be reliable.

7.2. MERITS

1. The COVID BED SLOT BOOKING SYSTEM is
 - Fast
 - Efficient
 - Reliable
2. Avoids data redundancy and inconsistency.
3. Web-based.
4. Any number of users can use it.
5. Provides more security and integrity to data.
6. Provides authorization

7.3. FUTURE WORKS

The COVID BED SLOT BOOKING SYSTEM can be enhanced by including more functionality like suggesting the patient their nearest hospital, allocation based on severity of Spo2.

We can further add an improvised COVID BED SLOT BOOKING SYSTEM which far more efficient and reliable.

CHAPTER 8: BIBLIOGRAPHY

8.1 BOOK REFERENCES:

1. Learn to Code HTML and CSS: Develop and Style Websites (Web Design Courses) 1st, Kindle Edition by Shay Howe
2. Learn Web Development with Python: Get hands-on with Python Programming and Django web development Paperback – Import, 21 December 2018 by Fabrizio Romano (Author), Gaston C. Hillar (Author), Arun Ravindran (Author)

8.2 WEBSITE REFERENCES:

1. HTML LEARNING:

1. <https://www.codecademy.com/>
2. <https://dash.generalassemb.ly/>
3. <https://www.w3schools.com/>

2. PYTHON LEARNING:

1. <http://www.tutorialspoint.com/>
2. <https://www.codecademy.com/>
3. <https://www.w3schools.com/>

PERSONAL DETAILS

NAME: AMOGH S BHARADWAJ

USN: 1DT19CS013

SEMESTER: 5TH SEM, 'A' SECTION

COLLEGE: DAYANANDA SAGAR ACADEMY OF
TECHNOLOGY AND MANAGEMENT

CONTACT INFORMATION:

EMAIL-ID: amoghsai1665@gmail.com

PHONE NO: 9108130261

NAME: VISHWAJIT H

USN: 1DT19CS051

SEMESTER: 5TH SEM, 'A' SECTION

COLLEGE: DAYANANDA SAGAR ACADEMY OF
TECHNOLOGY AND MANAGEMENT

CONTACT INFORMATION:

EMAIL-ID: [reddyvishwajit2255@gmail.com](mailto:reddivishwajit2255@gmail.com)

PHONE NO: 95915 72556