Chapter 1: Introduction

Introduction

The world is becoming more advanced nowadays. Everything that is manual is being transferred digitally for faster speed and better work in every aspect, like education, business, government work, hospitals. Whenever you visit a hospital in a case of emergency or any health-related problem, it seems that you don't get beds on time, the facilities and equipment are not available, and you were not aware of these things previously just because there is no facility that provides you information regarding whether the hospital is available or not. Many patients suffer because of this situation, and so many die also, just because the ICU beds in the hospitals are occupied and they have to wait in the general ward for their turn, and due to the delay in time, many people lose their lives.

The project uses Python, Flask, SQLAlchemy, HTML, CSS, and JavaScript to create this website successfully. We take data from hospitals by providing a registration form which will be fielded by the hospital management department. The data is going to be stored in a database, and then we provide information to users as per their requirements.

1.1 Overview

The project is based on the data fields by hospitals, which we store in our database and provide the user requirements with particular hospital information because hospital management needs to insert the right data and upgrade the data timely so people get the right information. The website is made in such a way that users have the option to choose between private and government hospitals. Users can also search for a hospital directly if they want, and we will provide all the necessary information regarding that particular hospital, including all the equipment facilities, doctor's availability, and all other information.

1.2 Background and Motivation

ICU and ventilator facilities are very important to helping those people who need treatment very quickly, because if any patient arrives at that hospital, its ICU beds are already occupied for operation. So they have to wait in the general ward until the operation is finished, putting their lives in danger; or maybe in a case where the bed is available but the equipment like a ventilator is not available.

Object Detection and recognition becomes a necessity when there is a need of automation, where the identification is done by machines instead of doing it manually for better performance and reliability. Normally, there are people hired specially for counting the number of students and vehicles that enter in a college everyday and maintain their records manually in a register. Automation by this system provides a better way to perform the same work.

1.3 Problem Statement and Objectives

To get the information about the occupancy of beds and necessary equipment for the patients in a particular hospital, especially in emergency cases. An electronic system will be brought into usage to identify these and make patients aware and easy to find.

Thus, the system implemented has the following objectives:
Objective 1: To count the number of beds are present in the hospital.
Objective 2: To provide information about facilities present in the hospitals.

1.4 Scope of the Project

As the project uses hospital information by their own administration we can use this website widely in other city also , Private hospitals and Government hospital both come under this project.

1.5 Team Organization

Ansh Joshi: Working in a team is a fantastic opportunity, and I am grateful that college provided me the opportunity to work on a project. I analyze the project, and as a front-end developer, it's my responsibility to make the website eye-catching and highly interactive so users don't have any problems.

Bhavik Mundra: This is my first project, and I am very excited and enthusiastic to work on it. I overview the topic, and as a backend developer, I have to create all the logic correctly and link the frontend, backend, and database successfully.

Bhavik Sharma: My role is to create a database to collect data and handle the documentation portion of the project while working with a fantastic team to get tremendous knowledge and expertise.

Bhavika Darpe: I oversee the team and manage the backend and database integration for the project. I can state with some confidence that this experience will be useful to me in the future.

1.6 Report Structure

The project ***Hospital Bed Availability Checking*** is primarily concerned with the **Image processing in real-time** and the whole project report is categorized into five chapters.

Chapter 1: Introduction- introduces the background of the problem followed by Smart India Hackathon. The chapter describes the objectives, scope and applications of the project. Further, the chapter gives the details of team members and their contribution in development of the project which is then subsequently ended with a report outline.

Chapter 2: Review of Literature- explores the work done in the area of Project undertaken and discusses the limitations of the existing system and highlights the issues and challenges of the project area. The chapter finally ends up with the requirement identification for present project work based on findings drawn from reviewed literature and end user interactions.

Chapter 3: Proposed System- starts with the project proposal based on requirement identified, followed by benefits of the project. The chapter also illustrates the software engineering paradigm used along with different design representations. The chapter also includes a block diagram and details of major modules of the project. Chapter also gives insights of different types of feasibility study carried out for the project undertaken. Later it gives details of the different deployment requirements for the developed project.

Chapter 4: Implementation- includes the details of different Technology/ Techniques/ Tools/ Programming Languages used in developing the Project. The chapter also includes the different user interfaces designed in the project along with their functionality. Further it discusses the experiment results along with testing of the project. The chapter ends with evaluation of the project on different parameters like accuracy and efficiency.

Chapter 5: Conclusion- Concludes with objective wise analysis of results and limitation of present work which is then followed by suggestions and recommendations for further improvement.

Chapter 2: Review of Literature

Review of Literature

There are various websites already present that also provide information about hospital management and the availability of all things related to health in different cities for different hospitals. The study of these hospitals provides information about the bed facilities available, with websites such as Bhopal, Delhi, and Kokilaben private hospitals serving as examples.

2.1 Preliminary Investigation

2.1.1 Current System

S.No.	Existing System	Source	Strengths	Weaknesses
1.	Delhi Govt. Website	https://coronabeds.jantasamvad.org/	Provides information about bed availability.	The information is not updated.
2.	Bhopal	https://bhopalcovidbeds.in/	Contains the list of govt as well as private hospitals with contact no. available only.	It doesn't show only the numbers of beds available in a particular hospital.

3.	MP Govt.	http://sarthak.nhmmp.gov.in/covid/facility-bed-occupancy-dashboard/	The information is kept updated.	The website doesn’t provide information about the facilities provided by the hospital.

Table: 1

This is the current website, which provides information about bed availability, whether or not necessary equipment is available, how much oxygen is available, and information about doctors.

2.2 Limitations of Current System

The limitation of these current system follows:

- First and foremost, if all of this work is done manually, we must first arrive at the hospital in an emergency condition, so we don't know if the beds are available or not, if the required equipment for an operation is available or not, and we also don't know where the hospitals are located, which is a limitation of manual work in that it takes time, and this problem also leads to premature discharge.
- The Delhi Government website The disadvantage of this website is that it only works during the corona and only provides information about corona beds with oxygen and ventilators; additionally, the updating process on this website is inefficient. The Bhopal Government website for COVID beds is also quite good, but similarly, they are also providing information about COVID-related issues, and this website is showing how many beds are available but not giving information about which hospitals have how many beds, so this is the drawback of this website: they are not providing the correct and complete information.

2.3 Requirement Identification and Analysis for Project

To complete this project, we need to learn about front-end programming languages such as HTML, CSS, and Java script, as well as the Bootstrap framework.

Similarly, for Beckend, we use Python, and for databases, we use Flask and my SQL.

After creating all the requirements in technical terms individually in all the languages in the database, we need to link all these files together to show the website can run successfully as we expected.Then we'll use Github to successfully publish our website.

Chapter 3: Proposed System

Proposed System

3.1 The Proposal

The proposal is to deploy this website for an Indore user on the internet so that anyone in Indore can get information about all of the hospitals in Indore and can track the data of a specific hospital through this website.
The website also describes the types of facilities available in the hospital, such as a canteen, a rest area, OT machines, and other utensils.

3.2 Benefits of the Proposed System

The current system had a lot of challenges that are overcome by this system :

- **Society Help:** In the event of an emergency, the website will assist the human community in searching for hospital and bed availability.
- **Man Power :** It does not require any person or their efforts to search for a hospital.
- **24 x 7 Availability :** Website is Active 24*7 and information also updated by the hospitals

  .

3.3 Block Diagram

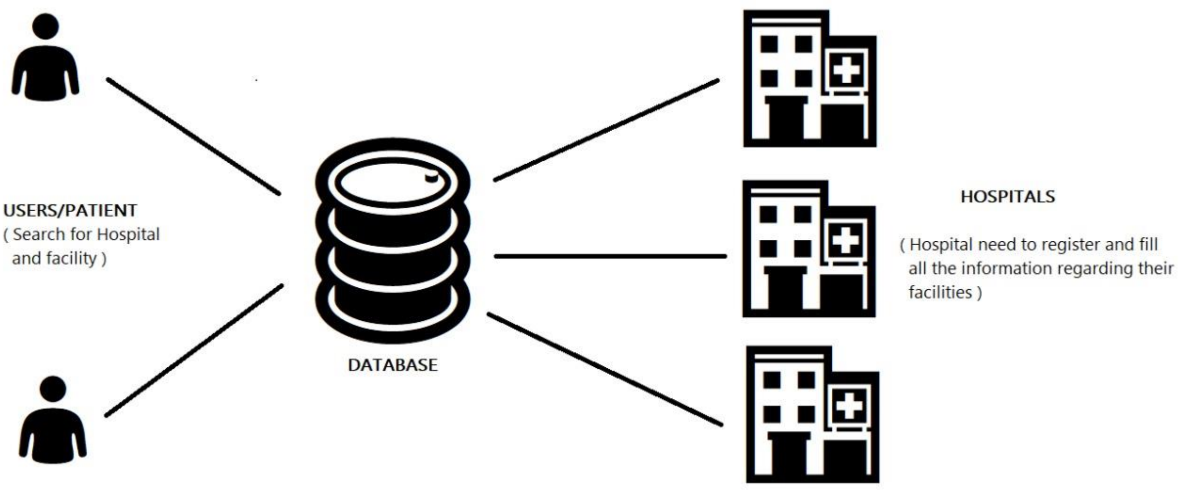

USERS/PATIENT
(Search for Hospital and facility)
DATABASE
HOSPITALS
(Hospital need to register and fill all the information regarding their facilities)

Figure 3.1 : Block Diagram

3.4 Feasibility Study

A feasibility study is an analysis of how successfully a system can be implemented, accounting for factors that affect it such as economic, technical and operational factors to determine its potential positive and negative outcomes before investing a considerable amount of time and money into it.

3.4.1 Technical

Technically, the website helps with all of the points that we have discussed about poor availability of beds facilities in hospitals.The hospital will register a page that we are going to collect in the database and show that data to users on page.

3.4.2 Economical

The website serves the intended purpose without any overhead costs thus it is economical. There is no such hardware implemented or purchased software used for designing the website. The user also just can search for the required details without costing money.

3.4.3 Operational

The main motto of our system is to provide the information regarding hospitals and bed availability in the city of Indore. The system is able to do that accurately and efficiently making the system operationally feasible.

3.5 Design Representation

3.5.1 Data Flow Diagrams

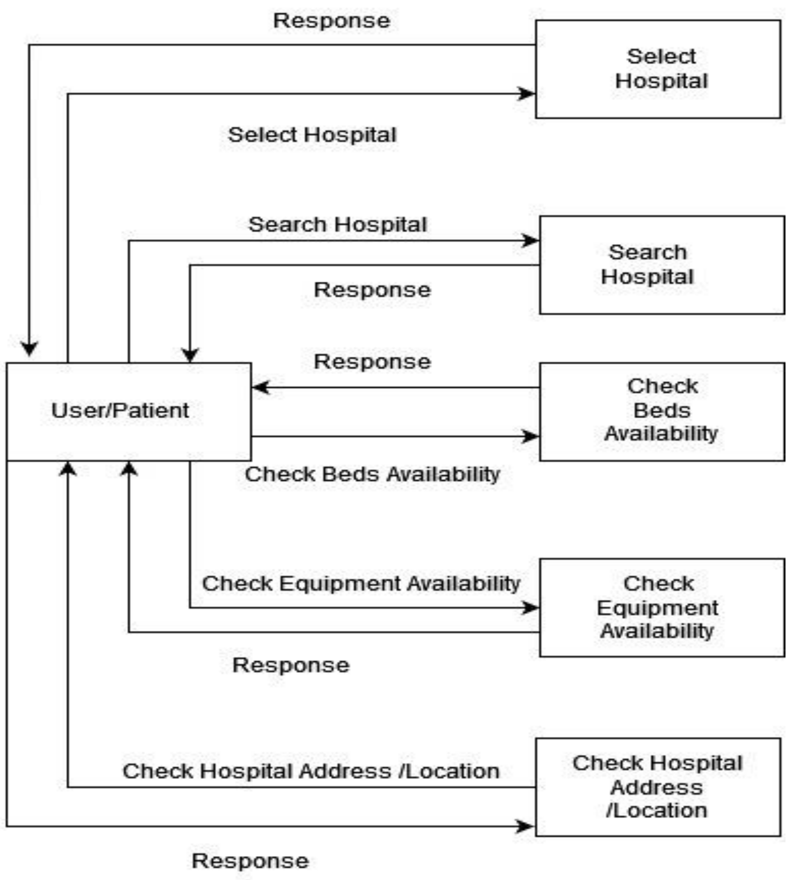


Figure 3.2 : Data Flow Diagram Level 0

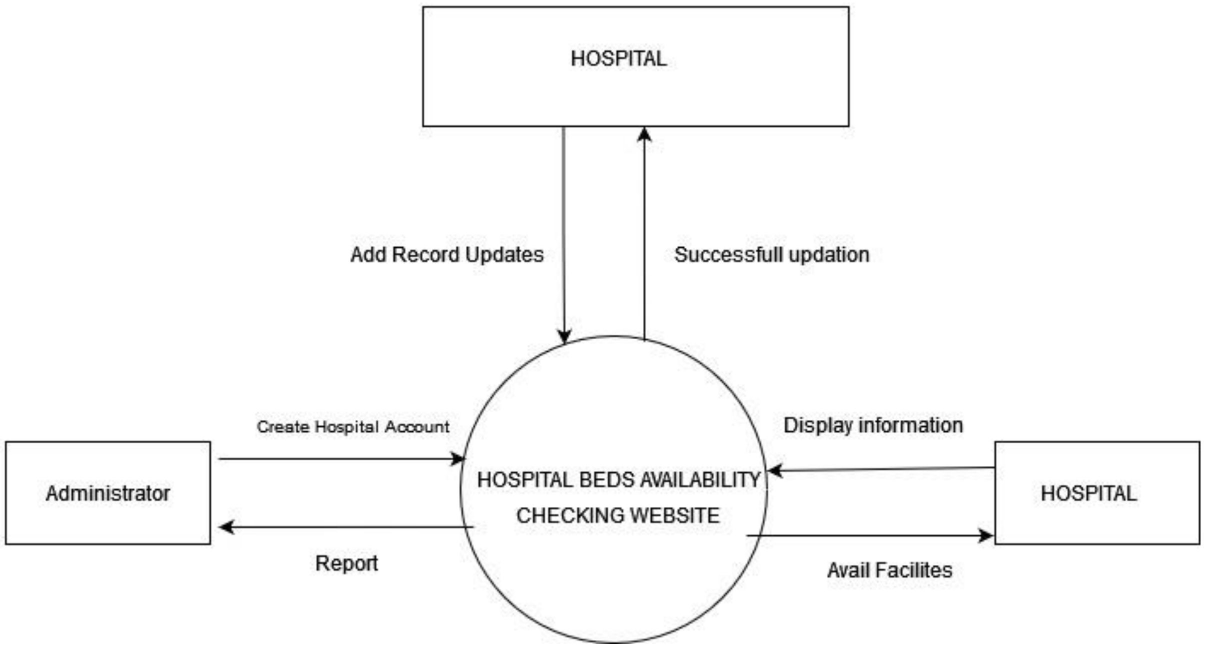


Figure 3.3 : Data Flow Diagram Level 1

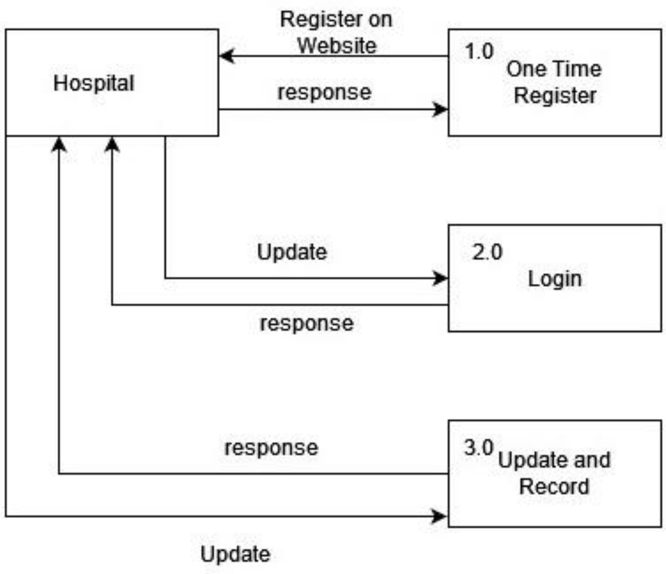


Figure 3.4 : Data Flow Diagram Level 1

3.5.2 Database Structure

Database Creation : We need to construct a database to hold the information about hospitals. To do this, first we install virtual environment and activate it. Then we import flask, flask-mail and flask-SQLAlchemy for Python. The name of database is given as **hospitalinfo.db** using following command:

```
app.config['SQLALCHEMY DATABASE URI'] = "sqlite:/hospitalinfo.db".
```

The database is named hospitalinfo.db and the key to access this database is "db." Next, we import this database into Python and link it to a Python file by using the command below:

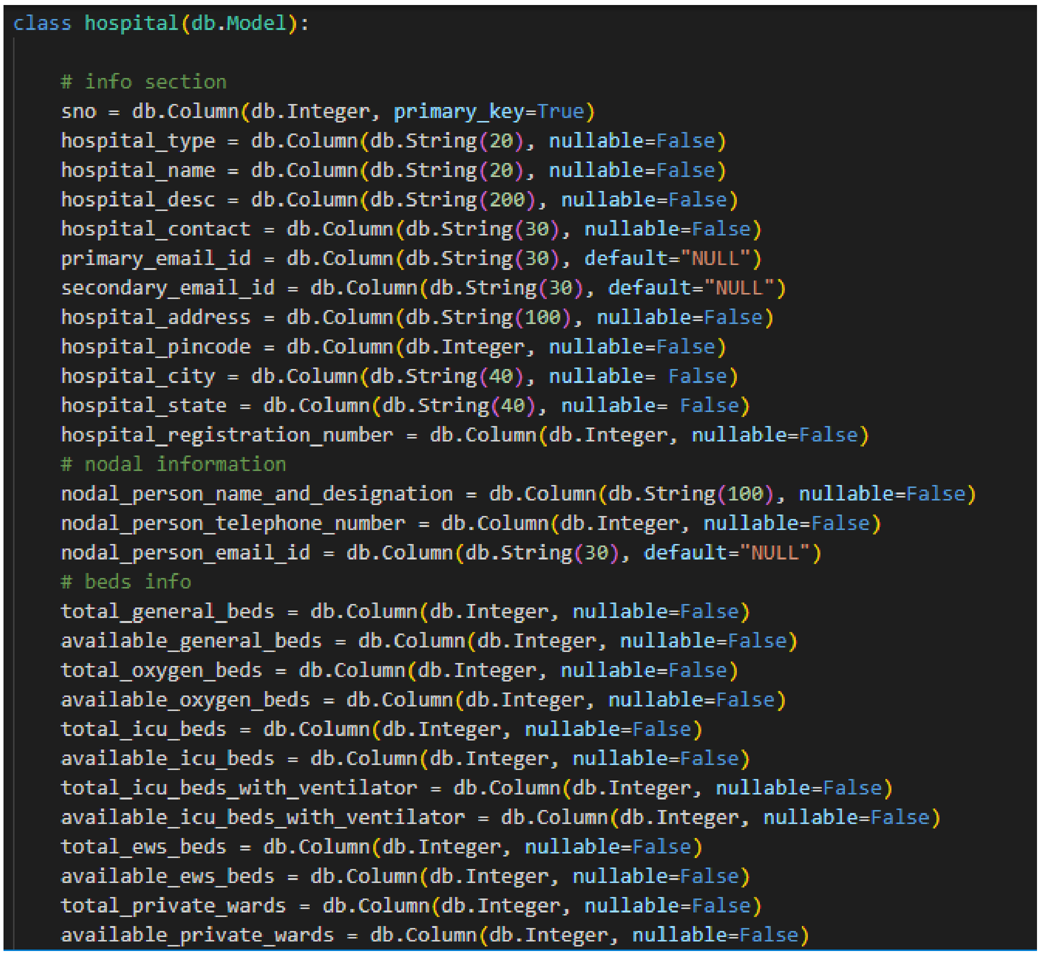
```
python
from app import db
db.create_all ()
```

Table Creation : We create table hospital in hospitalinfo database using sqlalchemy class method :

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

Then we create attributes related to all info that hospitals fill and provide data size and all constraints like PRIMARY KEY , not null = false**.** To store the information of the hospital, we create a table where every column stores different data.

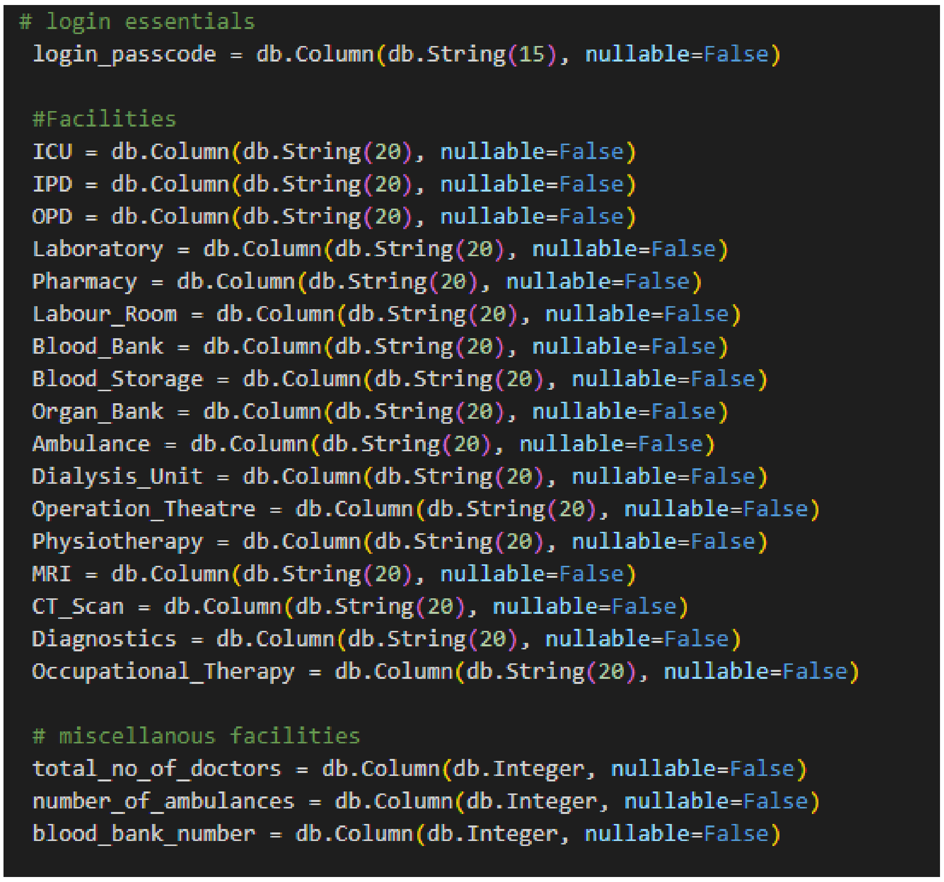
DataBase Collection : Using the function REGISTRATION(), which has numerous variables to store data like hospital name, hospital address, hospital desc, hospital contact and many other attributes that will be entered by hospital staff, we collect data from the hospital administration and pass the details to an object.This object is passed to the class Model where all the information is stored in the database.



```python
class hospital(db.Model):

    # info section
    sno = db.Column(db.Integer, primary_key=True)
    hospital_type = db.Column(db.String(20), nullable=False)
    hospital_name = db.Column(db.String(20), nullable=False)
    hospital_desc = db.Column(db.String(200), nullable=False)
    hospital_contact = db.Column(db.String(30), nullable=False)
    primary_email_id = db.Column(db.String(30), default="NULL")
    secondary_email_id = db.Column(db.String(30), default="NULL")
    hospital_address = db.Column(db.String(100), nullable=False)
    hospital_pincode = db.Column(db.Integer, nullable=False)
    hospital_city = db.Column(db.String(40), nullable= False)
    hospital_state = db.Column(db.String(40), nullable= False)
    hospital_registration_number = db.Column(db.Integer, nullable=False)
    # nodal information
    nodal_person_name_and_designation = db.Column(db.String(100), nullable=False)
    nodal_person_telephone_number = db.Column(db.Integer, nullable=False)
    nodal_person_email_id = db.Column(db.String(30), default="NULL")
    # beds info
    total_general_beds = db.Column(db.Integer, nullable=False)
    available_general_beds = db.Column(db.Integer, nullable=False)
    total_oxygen_beds = db.Column(db.Integer, nullable=False)
    available_oxygen_beds = db.Column(db.Integer, nullable=False)
    total_icu_beds = db.Column(db.Integer, nullable=False)
    available_icu_beds = db.Column(db.Integer, nullable=False)
    total_icu_beds_with_ventilator = db.Column(db.Integer, nullable=False)
    available_icu_beds_with_ventilator = db.Column(db.Integer, nullable=False)
    total_ews_beds = db.Column(db.Integer, nullable=False)
    available_ews_beds = db.Column(db.Integer, nullable=False)
    total_private_wards = db.Column(db.Integer, nullable=False)
    available_private_wards = db.Column(db.Integer, nullable=False)
```

Figure 3.5 : Database Diagram 1



```
# login essentials
 login_passcode = db.Column(db.String(15), nullable=False)

 #Facilities
 ICU = db.Column(db.String(20), nullable=False)
 IPD = db.Column(db.String(20), nullable=False)
 OPD = db.Column(db.String(20), nullable=False)
 Laboratory = db.Column(db.String(20), nullable=False)
 Pharmacy = db.Column(db.String(20), nullable=False)
 Labour_Room = db.Column(db.String(20), nullable=False)
 Blood_Bank = db.Column(db.String(20), nullable=False)
 Blood_Storage = db.Column(db.String(20), nullable=False)
 Organ_Bank = db.Column(db.String(20), nullable=False)
 Ambulance = db.Column(db.String(20), nullable=False)
 Dialysis_Unit = db.Column(db.String(20), nullable=False)
 Operation_Theatre = db.Column(db.String(20), nullable=False)
 Physiotherapy = db.Column(db.String(20), nullable=False)
 MRI = db.Column(db.String(20), nullable=False)
 CT_Scan = db.Column(db.String(20), nullable=False)
 Diagnostics = db.Column(db.String(20), nullable=False)
 Occupational_Therapy = db.Column(db.String(20), nullable=False)

 # miscellanous facilities
 total_no_of_doctors = db.Column(db.Integer, nullable=False)
 number_of_ambulances = db.Column(db.Integer, nullable=False)
 blood_bank_number = db.Column(db.Integer, nullable=False)
```

Figure 3.6 : Database Diagram 2

Chapter 4 : Implementation

Implementation

For the problem of counting the number of students and vehicles entering the college campus manually, the system is designed in such a way so as to automate the process by placing a camera at the entrance gate so that students, bikes and cars getting inside the college campus can be identified and counted.

For the purpose of getting the information, the system is designed in such a way that user can easily view the required hospitals. The user simply needs to view the website and he will get the information like different types of beds, their availability at a particular hospital and facilities available there.

4.1 Tools Used

4.1.1 Flask SQLAlchemy

FLask is a microframework that allows you to build web apps in Python. Flask is easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running. Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Flask-SQLAlchemy is an extension for Flask by setting up common objects and patterns for using those objects, such as a session tied to each web request,models and engines.

4.2 Language Used

Python language is used in the system due to the following Characteristics :

- **Simple:**
  Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English (but very strict English!). This pseudocode nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the syntax i.e. the language itself.

- **Free and Open Source :**
  Python is an example of a FLOSS (Free/Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read the software's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and improved by a community who just want to see a better Python.

- **Object Oriented :**
  Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

- **Extensive Libraries :**
  The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI(graphical user interfaces) using Tk, and also other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the "batteries included" philosophy of Python.

The flask language is used which is a web application framework written in python.
The primary advantage of choosing flask is built-in fast debugger, secure cookies and unit testing support, unicode basis and more.The framework allows developers to decide how exactly to build the application needed.

4.3 Screenshots

The Following are the screenshots of the result of the project :

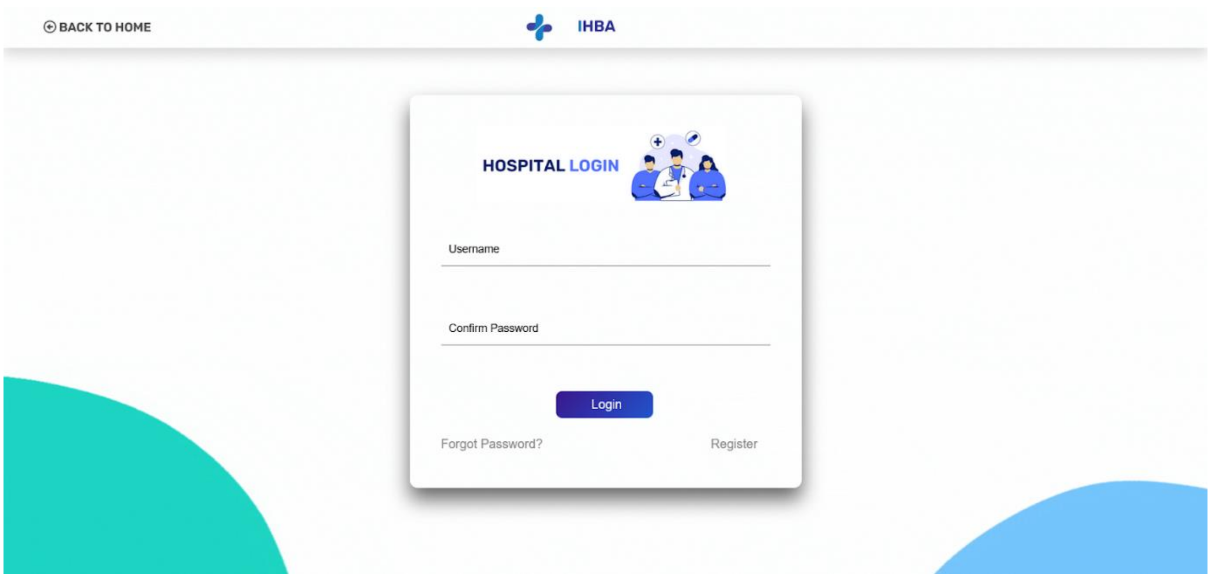
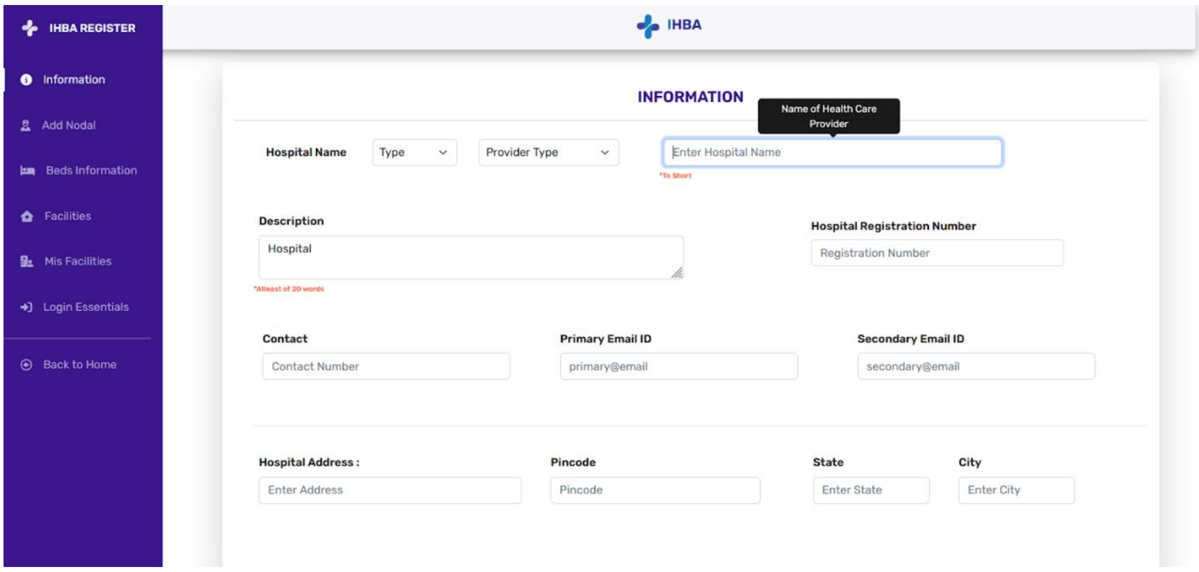


Figure 4.1 : Screenshot 1



Hospital Name: Type, Provider Type, Enter Hospital Name
*To Short
Description: Hospital
*Atleast of 20 words
Hospital Registration Number: Registration Number
Contact: Contact Number
Primary Email ID: primary@email
Secondary Email ID: secondary@email
Hospital Address : Enter Address
Pincode: Pincode
State: Enter State
City: Enter City

Figure 4.2 : Screenshot 2

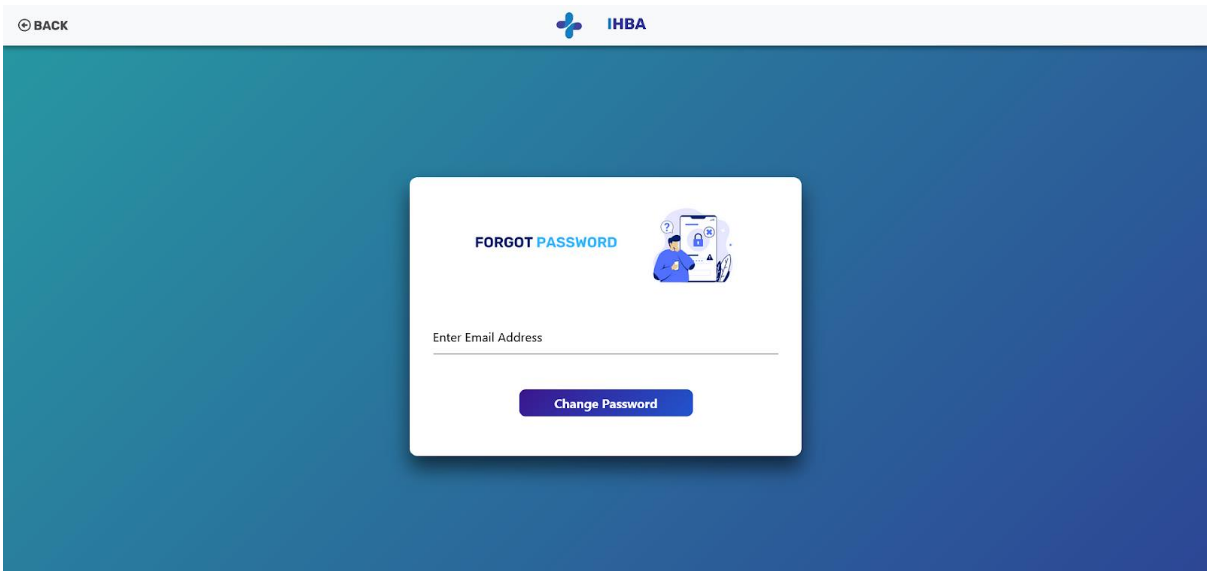


Figure 4.3 : Screenshot 3

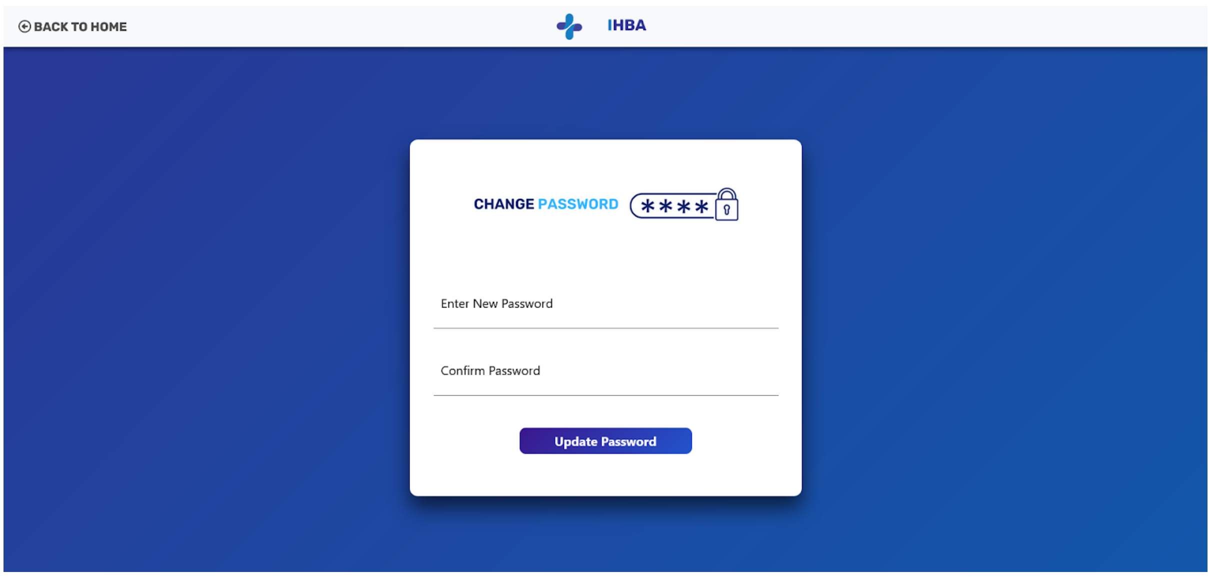
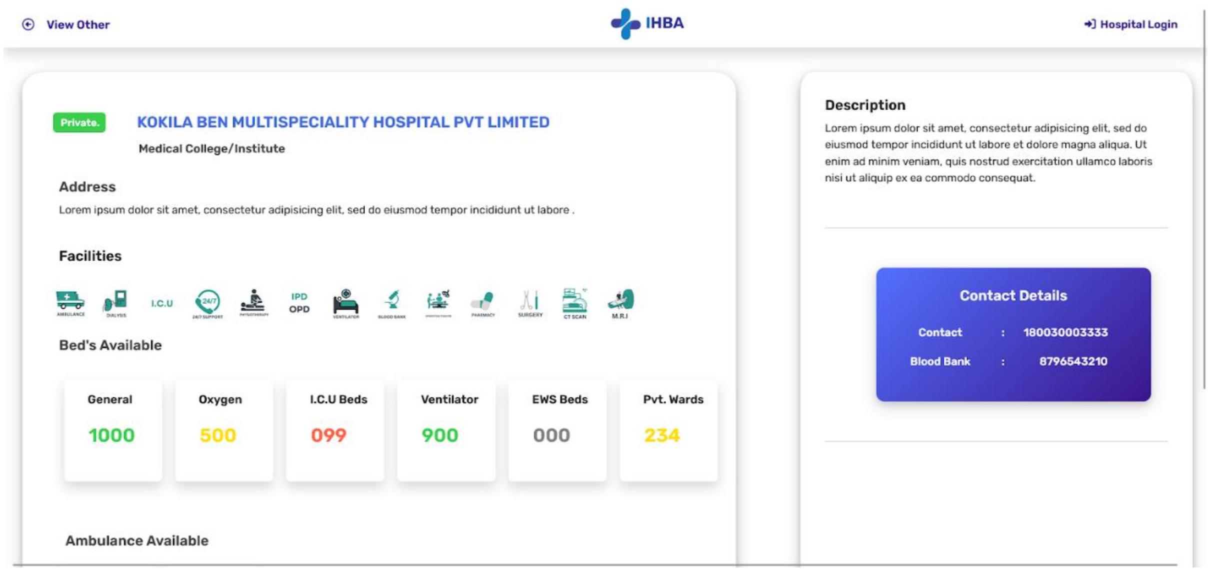


Figure 4.4 : Screenshot 4



KOKILA BEN MULTISPECIALITY HOSPITAL PVT LIMITED

Medical College/Institute

Address

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore .

Facilities

Bed's Available

General	Oxygen	I.C.U Beds	Ventilator	EWS Beds	Pvt. Wards
1000	500	099	900	000	234

Ambulance Available

Description

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Contact Details

Contact	:	180030003333
Blood Bank	:	8796543210

Figure 4.5 : Screenshot 5

4.4 Testing

Testing is the process of evaluation of a system to detect differences between given input and expected output and also to assess the features of the system. Testing assesses the quality of the product. It is a process that is done during the development process. .

4.4.1 Strategy Used

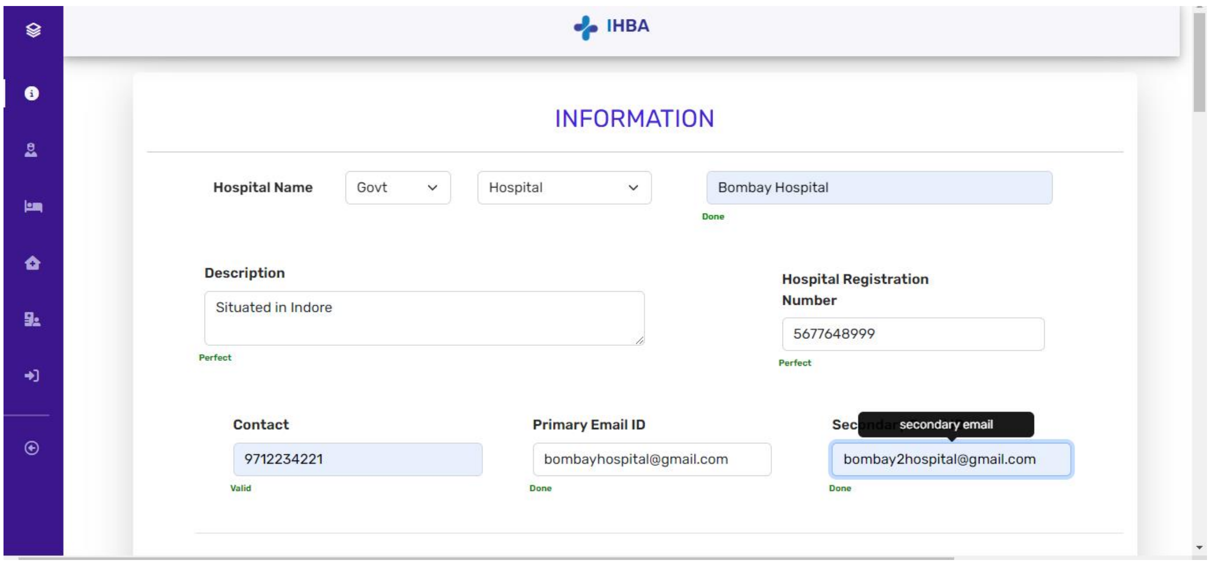
Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

The texting method used here is Black Box Testing. It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic

(i) Data Insertion Test CASE :

In this test case, we will insert hospital data and then determine whether the data is successfully stored in the database.



Hospital Name: Govt | Hospital | Bombay Hospital (Done)

Description: Situated in Indore (Perfect)

Hospital Registration Number: 5677648999 (Perfect)

Contact: 9712234221 (Valid)

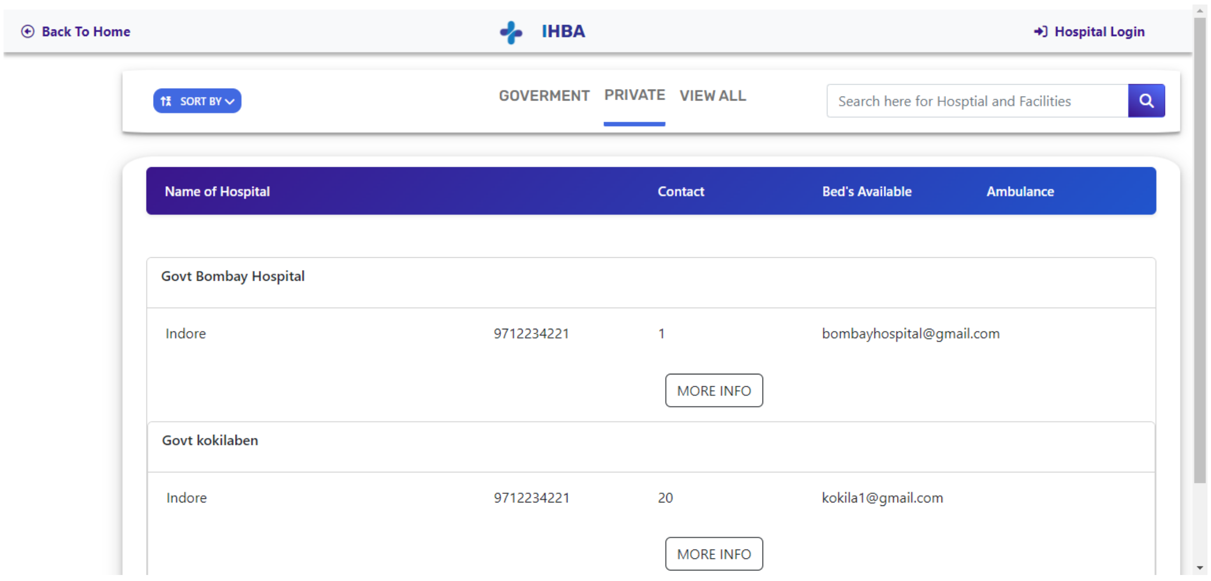
Primary Email ID: bombayhospital@gmail.com (Done)

Sec secondary email: bombay2hospital@gmail.com (Done)

Figure 4.6 : Test Case Data Insertion

This is the registration page where we will enter the hospital's information. Once we have entered all the information, we will submit the form. Next, we must determine if the hospital appears on the index page or not. If the data is there, the test case is successful.

Now in the below image you can see that there is data of “BOMBAY HOSPITAL” present which means we have passed the test case.



Name of Hospital		Contact	Bed's Available	Ambulance
Govt Bombay Hospital				
Indore	9712234221	1	bombayhospital@gmail.com	
Govt kokilaben				
Indore	9712234221	20	kokila1@gmail.com	

Figure 4.7 Test Case Index Page

(ii) Search Tab Test CASE :

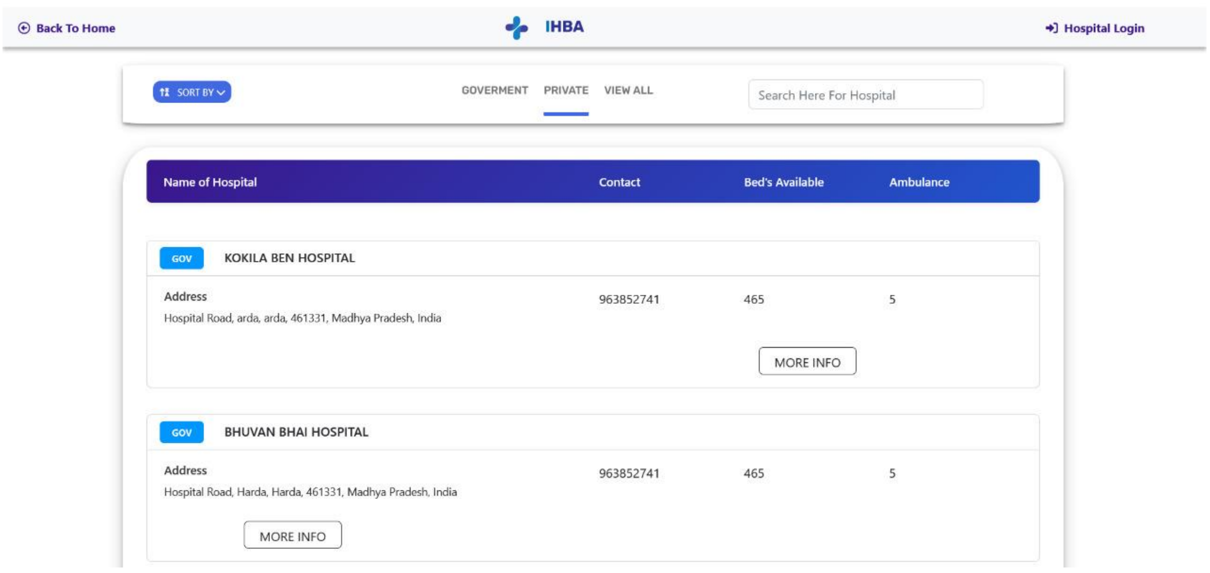
The first space is for the search bar when the user is using the bar to search for a particular hospital, so the output must match the expected result after searching and show details about the hospitals.For example, if a user searches for "Apollo Hospital," it must display information about the hospital to the user.

(iii) Sort According Private and Government Hospitals

The second test case is a sorting test case where the user can sort the list of hospitals according to private hospitals or government hospitals.

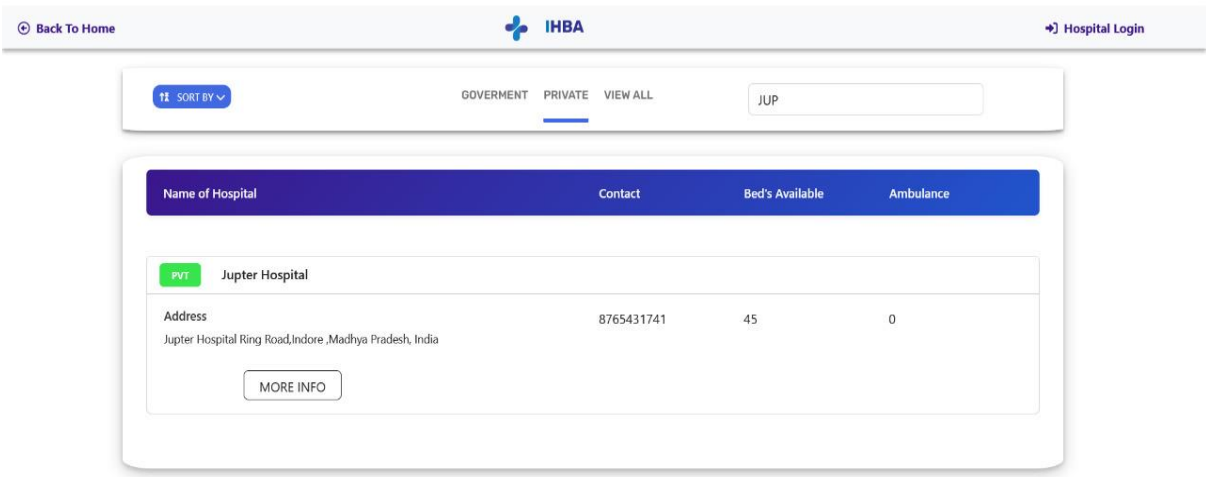
This feature is represented by a radio button with two options: "government" or "private".

If you select the government, then the website provides you with a list of all the government hospitals; if you select a private hospital, then it gives you a list of all private hospitals.



Name of Hospital	Contact	Bed's Available	Ambulance
GOV KOKILA BEN HOSPITAL Address Hospital Road, arda, arda, 461331, Madhya Pradesh, India	963852741	465	5
GOV BHUVAN BHAI HOSPITAL Address Hospital Road, Harda, Harda, 461331, Madhya Pradesh, India	963852741	465	5

Figure 4.8 : Screenshot of Sort Function



Name of Hospital	Contact	Bed's Available	Ambulance
PVT Jupter Hospital Address Jupter Hospital Ring Road,Indore ,Madhya Pradesh, India	8765431741	45	0

Figure 4.9 : Screenshot of Sort Function

Chapter 5: Conclusion

Conclusion

5.1 Conclusion

The goal of the project was to provide information about Indore hospitals' beds availability and facilities available in the hospitals so that people could easily track down all the information they needed in less time and without difficulty, and as a result, we may have saved one life.

5.2 Limitations of the Work

As you know, the website is working very well, but it is limited to Indore City only right now, which means we are only collecting data from Indore Hospital. This means the website is not going to help those people who are searching for hospitals in their city.

We are just taking the data from the hospital and showing it to the user about how many beds are available and what facilities are available in the hospital, but there is a limitation in the functionality in that we are not allowing the user to book any beds for the future.

5.3 Suggestion and Recommendations for Future Work

- Email Reminder If a hospital fails to update information on time or fails to complete the information, it is considered incorrect information and the hospital's data is removed from the website.
- Filter Option is provided for users so they can apply those filters according to their needs like Cancer Specialist or Blood Donation.
- Health Related Camp information near the town.
- We can Expand this website by providing information about clinics , medical stores and pathology.

- [NEAR BY] button, with which we can provide the user with information about hospitals located near his/her location. So if someone from outside came here and, for some reason, they had to search for a hospital, they were free to use this button for help.
- Best Hospital Suggestion based on patient remarks.

Bibliography

1. https://core.ac.uk/download/pdf/11036705.pdf
2. https://www.researchgate.net/publication/265095912_Cross_Hospital_Bed_Management_System
3. https://drive.google.com/file/d/1k9iaKSQ1roUN7lVszyZ3I2GihwAo4xou/view
4. https://excise.wb.gov.in/CHMS/Public/Page/CHMS_Public_Hospital_Bed_Availability.aspx
5. https://coronabeds.jantasamvad.org/
6. https://www.kokilabenhospital.com/
7. http://sarthak.nhmmp.gov.in/covid/facility-bed-occupancy-dashboard/
8. https://bhopalcovidbeds.in

Source Code

Create app.py:

```
from asyncio.windows_events import NULL

from email.policy import default

from flask import Flask, render_template, request, redirect, url_for, flash

from flask_sqlalchemy import SQLAlchemy


app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///hospitalinfo.db"

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False  #set to false to disable
tracking and use less memory

db = SQLAlchemy(app)


class hospital(db.Model):

    # info section

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

    hospital_type = db.Column(db.String(20), nullable=False)

    hospital_name = db.Column(db.String(20), nullable=False)

    hospital_desc = db.Column(db.String(200), nullable=False)

    hospital_contact = db.Column(db.String(30), nullable=False)

    primary_email_id = db.Column(db.String(30), default="NULL")

    secondary_email_id = db.Column(db.String(30), default="NULL")

    hospital_address = db.Column(db.String(100), nullable=False)
```

```
 hospital_pincode = db.Column(db.Integer, nullable=False)

 hospital_city = db.Column(db.String(40), nullable= False)

 hospital_state = db.Column(db.String(40), nullable= False)

 hospital_registration_number = db.Column(db.Integer, nullable=False)

 # nodal information

 nodal_person_name_and_designation = db.Column(db.String(100), nullable=False)

 nodal_person_telephone_number = db.Column(db.Integer, nullable=False)

 nodal_person_email_id = db.Column(db.String(30), default="NULL")

 # beds info

 total_general_beds = db.Column(db.Integer, nullable=False)

 available_general_beds = db.Column(db.Integer, nullable=False)

 total_oxygen_beds = db.Column(db.Integer, nullable=False)

 available_oxygen_beds = db.Column(db.Integer, nullable=False)

 total_icu_beds = db.Column(db.Integer, nullable=False)

 available_icu_beds = db.Column(db.Integer, nullable=False)

 total_icu_beds_with_ventilator = db.Column(db.Integer, nullable=False)

 available_icu_beds_with_ventilator = db.Column(db.Integer, nullable=False)

 total_ews_beds = db.Column(db.Integer, nullable=False)

 available_ews_beds = db.Column(db.Integer, nullable=False)

 total_private_wards = db.Column(db.Integer, nullable=False)

 available_private_wards = db.Column(db.Integer, nullable=False)

# login essentials

 login_passcode = db.Column(db.String(15), nullable=False)
```

```
#Facilities

ICU = db.Column(db.String(20), nullable=False)

IPD = db.Column(db.String(20), nullable=False)

OPD = db.Column(db.String(20), nullable=False)

Laboratory = db.Column(db.String(20), nullable=False)

Pharmacy = db.Column(db.String(20), nullable=False)

Labour_Room = db.Column(db.String(20), nullable=False)

Blood_Bank = db.Column(db.String(20), nullable=False)

Blood_Storage = db.Column(db.String(20), nullable=False)

Organ_Bank = db.Column(db.String(20), nullable=False)

Ambulance = db.Column(db.String(20), nullable=False)

Dialysis_Unit = db.Column(db.String(20), nullable=False)

Operation_Theatre = db.Column(db.String(20), nullable=False)

Physiotherapy = db.Column(db.String(20), nullable=False)

MRI = db.Column(db.String(20), nullable=False)

CT_Scan = db.Column(db.String(20), nullable=False)

Occupational_Therapy = db.Column(db.String(20), nullable=False)

# miscellanous facilities

total_no_of_doctors = db.Column(db.Integer, nullable=False)

number_of_ambulances = db.Column(db.Integer, nullable=False)

blood_bank_number = db.Column(db.Integer, nullable=False)

hospital_provider_type = db.Column(db.String(50), nullable=False)
```

```
    def __repr__(self) -> str:
        return f"{self.sno}-{self.hospital_name}"


@app.route("/", methods=['GET', 'POST'])
def index():
    args = request.args
    arg = args.get("type")
    if arg :
        obj=hospital.query.filter_by(hospital_type=arg).all()
    else :
        obj = hospital.query.all()
    return render_template('index.html', obj=obj)


@app.route("/moreinfo/<int:sno>",methods=['GET','POST'])
def moreinfo(sno):
    obj=hospital.query.filter_by(sno=sno).first()
    return render_template('moreinfo.html',obj=obj)


@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        primary_email_id = request.form['primary_email_id']
        login_passcode = request.form['login_passcode']
        if primary_email_id=='bhavik.mundra1603@gmail.com' and
login_passcode=='Bhavik1603*':
```

```
            obj=hospital.query.all()

            return redirect('/admin')

        if primary_email_id=='bhavika.darpe@gmail.com' and login_passcode=='Bhavika37':

            obj=hospital.query.all()

            return redirect('/admin')

        obj = hospital.query.filter_by(primary_email_id=primary_email_id).first()

        if obj.login_passcode == login_passcode:

            return redirect(f"/update/{obj.sno}")

    return render_template('login.html')


@app.route("/update/<int:sno>", methods=['GET', 'POST'])

def update(sno):

    if request.method == 'POST':

        nodal_person_name_and_designation =
request.form['nodal_person_name_and_designation']

        nodal_person_telephone_number = request.form['nodal_person_telephone_number']

        nodal_person_email_id = request.form['nodal_person_email_id']

        # beds info

        total_general_beds = request.form['total_general_beds']

        available_general_beds = request.form['available_general_beds']

        total_oxygen_beds = request.form['total_oxygen_beds']

        available_oxygen_beds = request.form['available_oxygen_beds']

        total_icu_beds = request.form['total_icu_beds']

        available_icu_beds = request.form['available_icu_beds']

        total_icu_beds_with_ventilator = request.form['total_icu_beds_with_ventilator']
        available_icu_beds_with_ventilator =
request.form['available_icu_beds_with_ventilator']

        total_ews_beds = request.form['total_ews_beds']

        available_ews_beds = request.form['available_ews_beds']

        total_private_wards = request.form['total_private_wards']

        available_private_wards = request.form['available_private_wards']

        # miscellanous facilities

        total_no_of_doctors = request.form['total_no_of_doctors']

        number_of_ambulances = request.form['number_of_ambulances']

        blood_bank_number = request.form['blood_bank_number']

        #facilities

        ICU = request.form['ICU']

        IPD = request.form['IPD']

        OPD = request.form['OPD']

        Laboratory = request.form['Laboratory']

        Pharmacy = request.form['Pharmacy']

        Labour_Room = request.form['Labour_Room']

        Blood_Bank = request.form['Blood_Bank']

        Blood_Storage = request.form['Blood_Storage']

        Organ_Bank = request.form['Organ_Bank']

        Ambulance = request.form['Ambulance']

        Dialysis_Unit = request.form['Dialysis_Unit']

        Operation_Theatre = request.form['Operation_Theatre']

        Physiotherapy = request.form['Physiotherapy']

        MRI = request.form['MRI']
```

```
CT_Scan = request.form['CT_Scan']

Dialysis_Unit = request.form['Dialysis_Unit']

Occupational_Therapy = request.form['Occupational_Therapy']

obj = hospital.query.filter_by(sno=sno).first()

obj.nodal_person_name_and_designation = nodal_person_name_and_designation

obj.nodal_person_telephone_number = nodal_person_telephone_number

obj.nodal_person_email_id = nodal_person_email_id

# beds info

obj.total_general_beds = total_general_beds

obj.available_general_beds = available_general_beds

obj.total_oxygen_beds = total_oxygen_beds

obj.available_oxygen_beds = available_oxygen_beds

obj.total_icu_beds = total_icu_beds

obj.available_icu_beds = available_icu_beds

obj.total_icu_beds_with_ventilator = total_icu_beds_with_ventilator

obj.available_icu_beds_with_ventilator = available_icu_beds_with_ventilator

obj.total_ews_beds = total_ews_beds

obj.available_ews_beds = available_ews_beds

obj.total_private_wards = total_private_wards

obj.available_private_wards = available_private_wards

obj.total_no_of_doctors = total_no_of_doctors

obj.number_of_ambulances = number_of_ambulances

obj.blood_bank_number = blood_bank_number

#facilities
```

```
        obj.ICU = ICU

        obj.IPD = IPD

        obj.OPD = OPD

        obj.Laboratory = Laboratory

        obj.Pharmacy = Pharmacy

        obj.Labour_Room = Labour_Room

        obj.Blood_Bank = Blood_Bank

        obj.Blood_Storage = Blood_Storage

        obj.Organ_Bank = Organ_Bank

        obj.Ambulance = Ambulance

        obj.Dialysis_Unit = Dialysis_Unit

        obj.Operation_Theatre = Operation_Theatre

        obj.Physiotherapy = Physiotherapy

        obj.MRI = MRI

        obj.CT_Scan = CT_Scan

        obj.Occupational_Therapy = Occupational_Therapy

        db.session.add(obj)

        db.session.commit()

    obj = hospital.query.filter_by(sno=sno).first()

    return render_template('update.html', obj=obj)


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

def registration():

    if request.method == 'POST':
        hospital_type = request.form['hospital_type']

        hospital_name = request.form['hospital_name']

        hospital_desc = request.form['hospital_desc']

        hospital_contact = request.form['hospital_contact']

        primary_email_id = request.form['primary_email_id']

        secondary_email_id = request.form['secondary_email_id']

        hospital_address = request.form['hospital_address']

        hospital_pincode = request.form['hospital_pincode']

        hospital_city = request.form['hospital_city']

        hospital_state = request.form['hospital_state']

        hospital_registration_number = request.form['hospital_registration_number']

        # nodal information

        nodal_person_name_and_designation =
request.form['nodal_person_name_and_designation']

        nodal_person_telephone_number = request.form['nodal_person_telephone_number']

        nodal_person_email_id = request.form['nodal_person_email_id']

        # beds info

        total_general_beds = request.form['total_general_beds']

        available_general_beds = request.form['available_general_beds']

        total_oxygen_beds = request.form['total_oxygen_beds']

        available_oxygen_beds = request.form['available_oxygen_beds']

        total_icu_beds = request.form['total_icu_beds']

        available_icu_beds = request.form['available_icu_beds']

        total_icu_beds_with_ventilator = request.form['total_icu_beds_with_ventilator']
        available_icu_beds_with_ventilator =
request.form['available_icu_beds_with_ventilator']

        total_ews_beds = request.form['total_ews_beds']

        available_ews_beds = request.form['available_ews_beds']

        total_private_wards = request.form['total_private_wards']

        available_private_wards = request.form['available_private_wards']

        # login essentials

        login_passcode = request.form['login_passcode']

        #facilities

        ICU = request.form['ICU']

        IPD = request.form['IPD']

        OPD = request.form['OPD']

        Laboratory = request.form['Laboratory']

        Pharmacy = request.form['Pharmacy']

        Labour_Room = request.form['Labour_Room']

        Blood_Bank = request.form['Blood_Bank']

        Blood_Storage = request.form['Blood_Storage']

        Organ_Bank = request.form['Organ_Bank']

        Ambulance = request.form['Ambulance']

        Dialysis_Unit = request.form['Dialysis_Unit']

        Operation_Theatre = request.form['Operation_Theatre']

        Physiotherapy = request.form['Physiotherapy']

        MRI = request.form['MRI']

        CT_Scan = request.form['CT_Scan']

        Occupational_Therapy = request.form['Occupational_Therapy']
        # miscellanous facilities

        total_no_of_doctors = request.form['total_no_of_doctors']

        # contact number of ambulance driver

        number_of_ambulances = request.form['number_of_ambulances']

        blood_bank_number = request.form['blood_bank_number']

        hospital_provider_type =request.form['hospital_provider_type']

        obj =
hospital(hospital_provider_type=hospital_provider_type,total_no_of_doctors=total_no_of_do
ctors, number_of_ambulances=number_of_ambulances,
blood_bank_number=blood_bank_number, hospital_type=hospital_type,
hospital_name=hospital_name, hospital_desc=hospital_desc,
hospital_contact=hospital_contact, primary_email_id=primary_email_id,
secondary_email_id=secondary_email_id,
hospital_address=hospital_address,hospital_city=hospital_city,hospital_state=hospital_state,
hospital_pincode=hospital_pincode,
hospital_registration_number=hospital_registration_number,
nodal_person_email_id=nodal_person_email_id,
nodal_person_name_and_designation=nodal_person_name_and_designation,
nodal_person_telephone_number=nodal_person_telephone_number,
total_ews_beds=total_ews_beds, available_ews_beds=available_ews_beds,
total_general_beds=total_general_beds, available_general_beds=available_general_beds,
total_icu_beds=total_icu_beds, available_icu_beds=available_icu_beds,
total_icu_beds_with_ventilator=total_icu_beds_with_ventilator,
available_icu_beds_with_ventilator=available_icu_beds_with_ventilator,
total_oxygen_beds=total_oxygen_beds, available_oxygen_beds=available_oxygen_beds,
total_private_wards=total_private_wards, available_private_wards=available_private_wards,
login_passcode=login_passcode,ICU=ICU, IPD=IPD, OPD=OPD, Laboratory=Laboratory,
Pharmacy=Pharmacy, Labour_Room=Labour_Room, Blood_Bank=Blood_Bank,
Blood_Storage=Blood_Storage, Organ_Bank=Organ_Bank, Ambulance=Ambulance,
Dialysis_Unit=Dialysis_Unit, Operation_Theatre=Operation_Theatre,
Physiotherapy=Physiotherapy, MRI=MRI, CT_Scan=CT_Scan,
Occupational_Therapy=Occupational_Therapy)
        db.session.add(obj)
        db.session.commit()
        return redirect('/login')
    return render_template('registration.html')


@app.route("/admin", methods=['GET', 'POST'])
def admin():
    obj = hospital.query.all()
    return render_template("admin.html", obj=obj)


@app.route("/deleteHospitalAccountByAdmin/<int:sno>")
def deleteHospitalByAdmin(sno):
    obj_ = hospital.query.filter_by(sno=sno).first()
    db.session.delete(obj_)
    db.session.commit()
    return redirect("/admin")


if __name__ == "__main__":
    app.run(debug=True, port=5050)
```