VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018

DBMS MINI PROJECT

REPORT ON

"HOSPITAL BED MANAGEMENT SYSTEM"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

PAVAN KUMAR C(1KG19CS062)
PUNITH K(1KG19CS068)

Under the Guidance of

Mrs.Amitha S
Assistant Professor
Department of CSE
K.S.S.E.M

Department of Computer Science & Engineering
K. S. SCHOOL OF ENGINEERING AND MANAGEMENT
#15, Mallasandra, off. Kanakapura Road, Bengaluru – 560109

2021-2022

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT
BENGALURU - 560109

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the **DBMS MINI PROJECT** entitled **"HOSPITAL BED MANAGEMENT SYSTEM"** presented by **Mr.Pavan Kumar C, USN: 1KG19CS062, Mr.Punith K, USN:1KG19CS068** of **V semester** in partial fulfillment of the award of **Bachelor of Engineering** in **Computer Science & Engineering** in **Visvesvaraya Technological University**, Belagavi during the academic year **2021-2022**. The **DBMS MINI PROJECT** has been approved as it satisfies the academic requirements in respect of **DBMS Mini Project(18CSL58)** prescribed for the Bachelor of Engineering degree.

| --- | --- | --- |
Mrs. Amitha S .	**Dr. K Venkat Rao**	**Dr.K.RAMANARASIMHA**
Assistant Professor, CSE	**Associate Prof. & Head, CSE**	**Principal / Director**
K.S.S.E.M., Bengaluru	**K.S.S.E.M., Bengaluru**	**K.S.S.E.M., Bengaluru**

Name of the Student: **Pavan Kumar C** **Punith K**

USN: **1KG19CS062** **1KG19CS068**

Signature of the Student:

Name of the examiners **Signature with date**

1. _____ _____

2. _____ _____

ACKNOWLEDGEMENT

The successful presentation of the **DBMS MINI PROJECT** would be incomplete without the mention of the people who made it possible and whose constant guidance crowned our effort with success.

We would like to extend our gratitude to the **MANAGEMENT, KAMMAVARI SANGHAM**, Bengaluru, for providing all the facilities to present the DBMS Mini Project.

We would like to extend our gratitude to **Dr.K.RAMA NARASIMHA,** Principal / Director, K.S.School of Engineering and Management, Bengaluru, for facilitating us to present the DBMS Mini Project.

We thank **Dr.K Venkat Rao**, Associate Professor and Head, Department of Computer Science and Engineering, K. S. School of Engineering and Management, Bengaluru, for his encouragement.

We would like to thank our Project Guide, **Mrs. Amitha S.**, Assistant Professor, Department of Computer Science and Engineering, K. S. School of Engineering and Management, Bengaluru, for her constant guidance and inputs.

We would like to thank all the **Teaching** Staff and **Non-Teaching** Staff of the college for their co-operation.

Finally, We extend our heart-felt gratitude to our family for their encouragement and support without which We would not have come so far. Moreover, We thank all our friends for their invaluable support and cooperation.

PAVAN KUMAR C (1KG19CS062)

PUNITH K (1KG19CS068)

ABSTRACT

Beds4Meds is an integrated platform created to simplify the chaos around finding hospital beds for COVID treatment in urban India. It is made for the benefit of individuals who have just been diagnosed with the COVID-19 virus. It can be thought of as an interface wherein affected people can find medical facilities near them and medical authorities can update the status of bed availability.

It is a collaborative project to document all available medical facilities in a user-friendly format which can be accessed by everyone who needs it. It provides a start-to-end solution that takes care of and helps the affected individual from the moment that he/she is diagnosed to the time he/she is admitted to the hospital for treatment. Beds4Meds acts as the bridge between the afflicted person and the medical authorities involved by providing the individual with a solution that will guarantee minimal risk and quicker treatment without panic and chaos. It also relieves hospitals of the trouble of having to turn away people due to no availability and helps them document their occupancy publicly.

TABLE OF CONTENTS

Chapter No.	Contents	Page No.
	Acknowledgement	I
	Abstract	II
	Table of Contents	III
	List of Figures	V
	List of Tables	VI
Chapter 1	**INTRODUCTION**	1
1.1	OVERVIEW	1
1.2	PROBLEM STATEMENT	1
1.3	DATABSE MANAGEMENT SYSTEM	2
1.4	SQL	2
1.5	HTML / JAVASCRIPT	3
Chapter 2	**REQUIREMENTS SPECIFICATION**	4
2.1	OVERALL DESCRIPTION	4
2.2	SPECIFIC REQUIREMENTS	4
2.3	SOFTWARE REQUIREMENTS	4
2.4	HARDWARE REQUIREMENTS	5
2.5	TECHNOLOGY	6
Chapter 3	**DETAILED DESIGN**	7
3.1	SYSTEM DESIGN	7
3.2	ENTITY RELATIONSHIP DIAGRAM	8
3.3	RELATIONAL SCHEMA	11
3.4	DESCRIPTION OF TABLES	12
Chapter 4	**IMPLEMENTATION**	14
4.1	MODULE AND THEIR ROLES	14
4.2	TRIGGERS AND STORED PROCEDURES	22
4.3	RESULT	23
Chapter 5	**TESTING**	24
5.1	SOFTWARE TESTING	24
5.2	MODULE TESTING AND INTEGRATION	24
5.3	LIMITATIONS	25

Chapter 6 SNAP SHOTS 26
6.1 HOME PAGE 26
6.2 SEARCH MODULE 26
6.3 SEARCH RESULTS 27
6.4 THE FINAL PAGE OF SEARCH MODULE 27
6.5 THE LOGIN FOR HOSPITAL MODULE 28
6.6 HOME PAGE FOR HOSPITALS 28
6.7 ADD A PATIENT 29
6.8 THE PATIENT LIST OF HOSPITALS MODULE 29
Chapter 7 CONCLUSION 30
Chapter 8 FUTURE ENHANCEMENTS 31
REFERENCES 32

LIST OF FIGURES

Figure No.	Figure Name	Page No.
3.1	JSP Architecture	6
3.2	Enhanced ER diagram of Hospital Bed Management System	8
3.3	ER diagram of Hospital Bed Management System	9
3.4	Schema diagram	10
6.1	Home Page	26
6.2	Search Module	26
6.3	Search Results	27
6.4	The Final Page Of Search Module	27
6.5	The Login For Hospital Module	28
6.6	Home Page For Hospitals	28
6.7	Add A Patient	29
6.8	The Patient List Of Hospitals Module	29

LIST OF TABLES

Table No.	Table Name	Page No.
3.4.1	Wards	11
3.4.2	Hospitals	11
3.4.3	Hospitals with ventilators	11
3.4.4	Hospitals without ventilators	11
3.4.5	Patient list	11
3.4.6	Users	11

Chapter 1

INTRODUCTION

1.1 OVERVIEW

The platform maintains a comprehensive collection of data relating to the medical facilities in a particular geographic location. It sorts these details based on many parameters such as location, ambulance availability and vacancies which are then used as constraints when displaying search results to the end user.

Our platform is based around the two groups of end users that are going to be making use of the platform itself. These two groups are the 2 ends of the spectrum in terms of the service we provide. As in, on one hand, we have the patient who will be in search of suitable treatment facilities close to him. On the other side are the hospital staff who can update and control their vacancy/occupancy data. These 2 customer groups are benefited by our platform that helps one reach the other, hence breaching the aforementioned gap between the patient and the medical facility.

1.2 PROBLEM STATEMENT

There is nothing more valuable than human life and to help preserve it, we have developed this interface. Our priority lies strongly with providing the patient with an easy and smooth experience in his/her search for a treatment centre. The communication gap between the patients and the government/medical authorities is solved by our interface that seamlessly connects the two agencies and unites the separate wings of the COVID crisis.

1.3 DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users application programs, ensuring that data is consistently organizedand remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified, and the database schema, which defines the database's logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

1.4 SQL

SQL is a standard language for storing, manipulating and retrieving data in databases. Originally based upon relational algebra and tuple relational calculus, SQL consists ofa data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

1.5 HTML / JavaScript

HTML is a markup language used for structuring and presenting content on the web and the fifth and current major version of the HTML standard.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

Java Script often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

JAVA CONNECTIONS

To connect the database with the front end we use a java connector JDBC (Java Database Connectivity). JDBC is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

To achieve connectivity we use JSPs (Java Server Pages) in this project. Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP is similar to PHP and ASP, but it uses the Java programming language.

Chapter 2

REQUIREMENTS SPECIFICATION

A computerized way of handling information about property and users details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a database driven web application whose requirements are mentioned in this section.

2.1 OVERALL DESCRIPTION

A reliable and scalable database driven web application with security features that is easy to use and maintain is the requisite.

2.2 SPECIFIC REQUIREMENTS

The specific requirements of the Hospital Bed Management System are stated as follows:

2.3 SOFTWARE REQUIREMENTS

1. A working internet connection for the access to the website.

   Bandwidth greater than or equal to 25 MBPS

   Latency lesser than or equal to 100 ms

2. A web browser to access the URL of the website.

   Eg: Internet Explorer, Google Chrome, Mozilla Firefox

3. No specific software required.

4. Operating System: Windows XP equivalent or higher

2.4 HARDWARE REQUIREMENTS

Any basic computer or mobile device that has LAN capability to connect to the Internet.

Processor: Intel Core 2 Duo or better.

Memory: 1GB RAM or higher.

Screen Resolution: 1366 x 768 pixels minimum

Keyboard: To enter the values into the website.

2.5 TECHNOLOGY

□ Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. We have used this language for the creation of elements that are shown in the webpage. Be it the forms that collect data or the section that shows hospital details, we designed it all using HTML.

□ Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology ofthe World Wide Web, alongside HTML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. We have used CSS for the styling of the web pages.

□ My SQL is an open-source relational database management system (RDBMS).The "SQL" is the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language that programmers use to create, modify and extract data from the relational database, as well as control user access to the database. We have used My SQL for the creation, updating and maintenance of the databases which house all the information relating to the project.

□ Jinja is a web template engine for the Python programming language. Jinja is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code.

□ Flask is a micro web framework written in Python. It is classified as a micro framework 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. We have used Flask for the integration of the platform as a whole (database + webpage). It is an application that holds the whole platform together.

Chapter 3

DETAILED DESIGN

3.1 SYSTEM DESIGN

The web server needs a JSP engine, i.e., a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs. This server will act as a mediator between the client browser and a database.

The following diagram shows the JSP architecture.

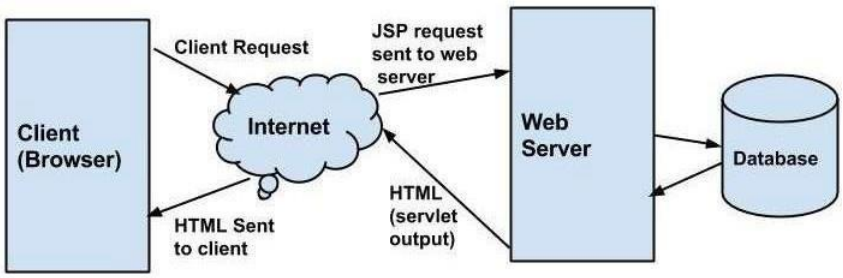

Fig. 3.1: JSP Architecture

Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client. Database architecture focuses on the design, development, implementation and maintenance of computer programsthat store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users. Several types of databases, including relational or multimedia, may be created.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business.

An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three-schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ER Diagramis unlikely to be helpful on its own in integrating data into a pre-existing information system.

Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.

We have prepared an Entity-Relationship Diagram that elucidates the 6 tables that we created inthe databaseand their schema is also specified. The relationships between the entities and the triggers, functions and indexes used are also displayed.

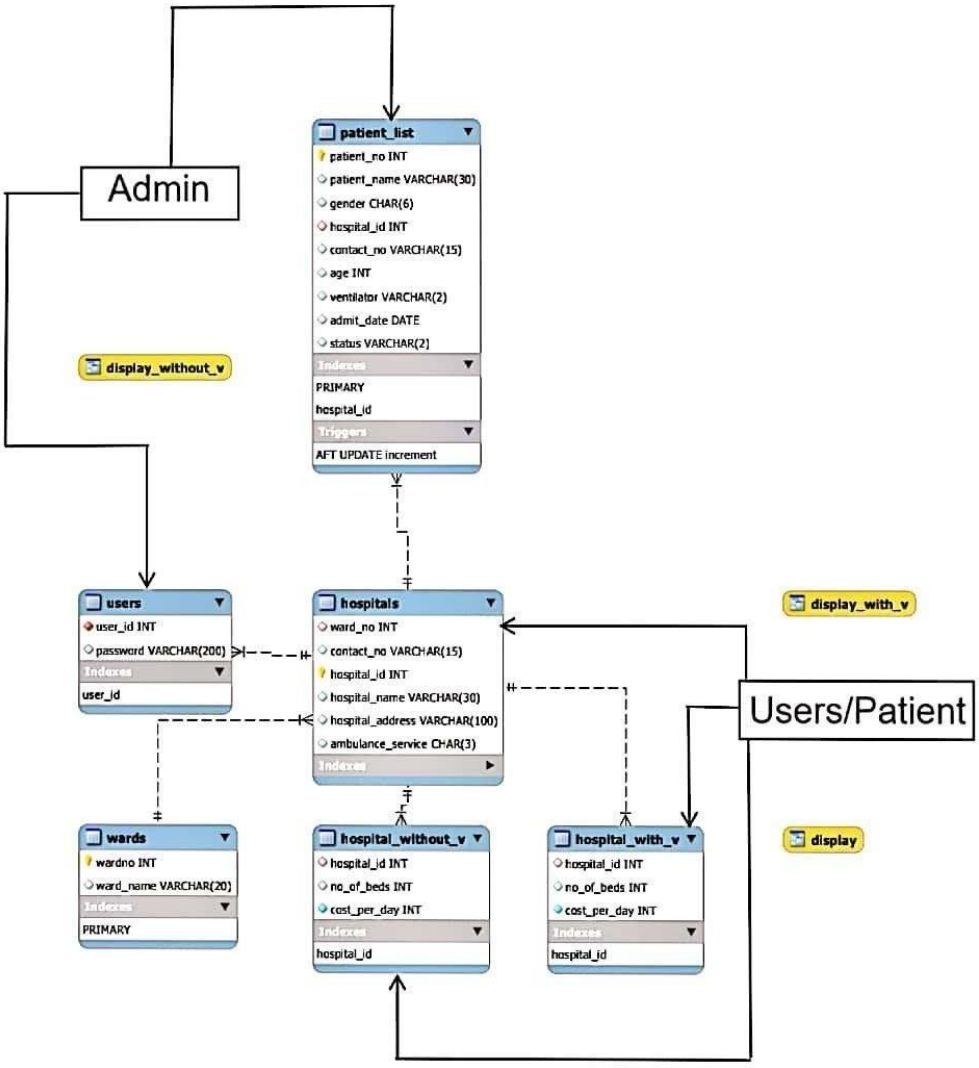

Figure 3.2: Enhanced ER diagram of Hospital Bed Management System

This diagram provides a very clear picture of the architecture of the platform and how the dataworks together with the UI to serve all the requests successfully.

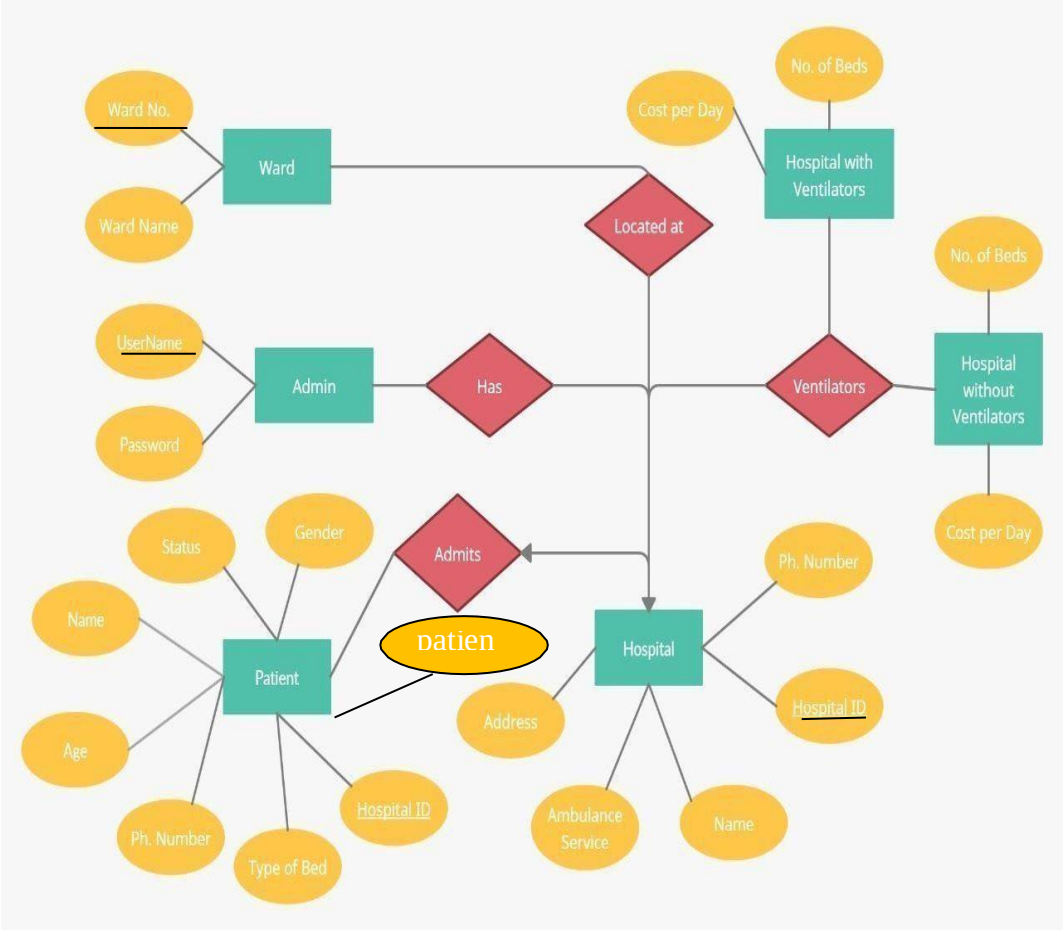

Figure 3.3: ER diagram of Hospital Bed Management System

3.3 RELATIONAL SCHEMA

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

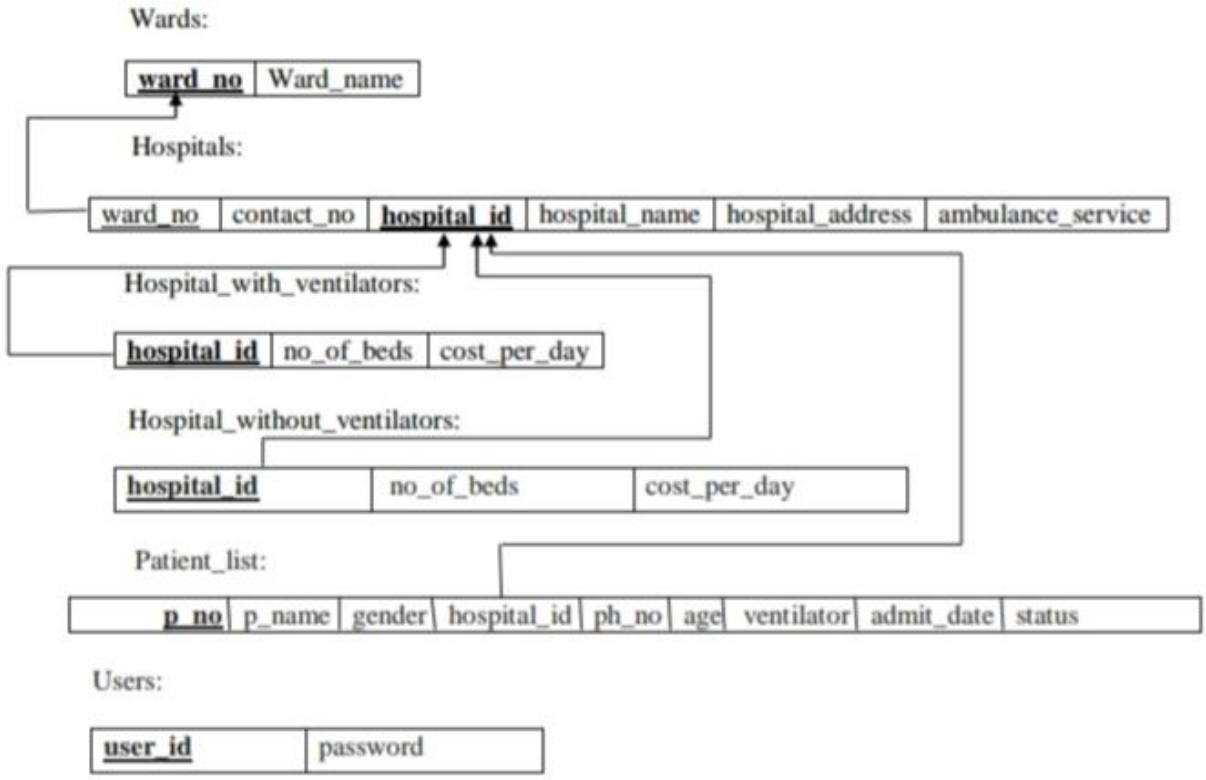

Fig. 3.4: Schema diagram

3.4 DESCRIPTION OF TABLES

The database consists of six tables:

3.4.1. Wards : It stores the ward details.

- Ward_no : Unique Number of the ward
- Ward_name: Name of the ward

3.4.2. Hospitals : Contains the details of the hospital

- Ward_no : Foreign key for the ward number
- Contact_no : Phone number of the hospital
- Hospital_id : Unique Id of the hospital
- Hospital_name : Name of the hospital
- Hospital_address : Address of the hospital
- Ambulance_service : Details of the Ambulance Facilities in Hospital

3.4.3. Hospitals_with_ventilators : Contains the details of the ventilators

- Hospital_id : Foreign key for the Id of the hospital
- No_of_beds: Beds number
- Cost_per_day: Amount received for each day

3.4.4. Hospitals_without_ventilators : Contains the details of hospitals without ventilators

- Hospital_id : Foreign key for the Id of the hospital
- No_of_beds: Beds number
- Cost_per_day: Amount received for each day

3.4.5. Patient_list : Contains the list of the patient

- Patient_no : Unique Id number of the patient
- Patient_name : Name of the patient
- Gender : Knows the gender of the patient
- Hospital_id : Foreign keyfor the Id of the hospital
- Contact_no : Phone number of the patient
- Age : Knows the age of the patient
- Ventilator : Checks the availability of the ventilator
- Admit_date : Provides the admission date of the patient
- Status : Checks the status

3.4.6. Users : Makes the login for the users

- User_id : Unique Login id of the user
- Password : Checks the password of the user

Chapter 4

IMPLEMENTATION

4.1 MODULES AND THEIR ROLES

```
from flask import Flask,render_template,request,g,session,redirect,flash
from datetime import timedelta
import functions
import os
app=Flask(_name_)
app.secret_key=os.urandom(24)
dbconn={"host":"localhost","user":"root","password":"Pavan12!@","database":"mini_
project"} #sql user and
database details
ldetail=[]
@app.route("/",methods=['GET','POST'])
def entry():
return render_template('home.html',the_title='home')
@app.route("/hsearch",methods=['GET','POST'])
def hello():
return render_template('beds4meds.html',the_title='beds4meds')
@app.route("/home",methods=['GET','POST'])
def home():
return redirect('/dropsession')
```

```
@app.route("/login",methods=['GET','POST'])
def login():
if "user" in session:
flash("Already Logged in")
return redirect('/admin')
else:
return render_template('loginpage.html')
@app.route("/register",methods=['GET','POST'])
def register():
return render_template('register.html',the_title='register')
@app.route("/success" ,methods=['post'])
def success():
req=request.form
try:
functions.check_data_from_form(req)
except Exception:
return redirect('/register')
hname=request.form['hname']
haddr=request.form['haddr']
hid=request.form['hid']
hcno=request.form['cno']
hwardno=request.form['wno']
hnowov=request.form['bno_wo_v']
hcowov=request.form['cpd_wo_v']
hnowv=request.form['bno_w_v']
hcowv=request.form['cpd_w_v']
```

```
ambsrv=request.form['amb_srv']
hpasswd=request.form['passwd']
#checking if total no of beds is zero
if int(hnowov)==0 and int(hnowv)==0:
flash("no of beds cant be zero")
return redirect('/login')
#if not zero then add hospitalinto database
functions.add_new_hospital((hwardno,hcno,hid,hname,haddr,ambsrv))
functions.encrypt_password(hid,hpasswd)
functions.add_bed_without_v((hid,hnowov,hcowov))
functions.add_bed_with_v((hid,hnowv,hcowv))
flash("hospital registered")
return redirect('/login')
@app.route("/update",methods=['POST','GET'])
def update():
req=request.form
try:
functions.check_data_from_form(req)
except Exception:
return redirect('/login')
if functions.check_hospital_id(int(request.form['uname']))==0:
flash("invalid user id")
return redirect('/admin')
if functions.decrypt_password(int(request.form['uname']),request.form['psswd']):
session['user']=request.form['uname']
flash("login successful")
return redirect('/admin')
```

```
else:
flash("wrong password/userid")
return redirect('/login')
@app.route('/admin',methods=['POST','GET'])
def admin():
if "user" in session:
return render_template('admin.html')
else:
return redirect('/login')
@app.route('/add_patient')
def add_patient():
if "user" in session:
user=session["user"]
bnowov=functions.bed_count_without_ventilators(user)
bnowv=functions.bed_count_with_ventilators(user)
return render_template('add_patient.html',bwov=bnowov,bwv=bnowv)
@app.route('/success1',methods=['POST','GET'])
def upddate_db():
if "user" in session:
req=request.form
try:
functions.check_data_from_form(req)
except Exception:
return redirect('/add_patient')
user=session["user"]
if functions.check_bed_availability(request.form['bt'],user):
```

```
functions.update_database(request.form['bt'],request.form['name'],request.form['gender'
],request.form['cno'],req uest.form['age'],user)
functions.decrement_bed_count(request.form['bt'],user)
flash("Patient was admitted successfully")
return redirect('/admin')
else:
flash("No bed is available")
return redirect('/admin')
@app.route('/view_patient')
def view():
if "user" in session:
user=session["user"]
content=functions.view_patients(user)
if len(content)==0:
flash("No patients")
return redirect('/admin')
else:
l=['patient_no','patient_name','gender','contact_no','age','ventilator','admit_date']
return render_template('view_patient.html',row_titles=l,the_data=content)
else:
return redirect('/login')
@app.route('/discharge',methods=['POST','GET'])
def discharge():
if "user" in session:
user=session["user"]
req=request.form
```

```
try:
functions.check_data_from_form(req)
except Exception:
return redirect('/view_patient')
if functions.check_patient(request.form['pid'],user):
functions.increment_bed_count(request.form['pid'],user)
content=functions.discharge_patient(request.form['pid'])
flash("Patient was discharged successfully")
return redirect('/admin')
else:
flash("enter correct patient id")
return redirect('/admin')
else:
return redirect('/login')
@app.route('/about_us')
def about_us():
return render_template('about.html')
@app.before_request
def before_request():
g.user=None
if 'user' in session:
g.user=session['user']
@app.route('/dropsession')
def dropsession():
if "user" in session:
flash("You were logged out successfully")
```

```
session.pop('user',None)

return redirect('/login')

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

def hsearch():

l=('hospital_id','ward_no','hospital_name','no_of_beds','cost_per_day')
message="Hurray! There are hospitals in your ward or in nearby wards." message1="we recommend hospital with vetilator considering your age."

ldetail.clear()

req = request.form

try:

functions.check_data_from_form(req)

except Exception:

return redirect('/hsearch')

pname=request.form['name']

Age=int(request.form['age'])

cno=request.form['cno']

wardno=request.form['wardno'] #taking all details filled in form gender=request.form.getlist('gender')

ge =gender[0]

ldetail.extend([pname,Age,cno,wardno,ge])

contents=functions.search_hospital_in_ward_without_v(int(wardno))
contents1=functions.search_hospital_in_ward_with_v(int(wardno))

if len(contents)==0:                    #checking if hospital is present or not

return

render_template('hsearch_no_hospital.html',the_name=pname,the_age=Age,the_wardn
o=wardno,the_gender=g e,the_contno=cno)

elif int(Age)<60:                       #checking age ,if age>60 then recommend
```

hospital withventilators

return
render_template('hsearch_in_ward.html',the_name=pname,the_age=Age,the_wardno=wardno,the_gender=ge,the_contno=cno,row_titles=l,the_data=contents,the_message=message)

else:

if len(contents1)==0: return

render_template('hsearch_in_ward.html',the_name=pname,the_age=Age,the_wardno=wardno,the_gender=ge,the_contno=cno,row_titles=l,the_data=contents,the_message=message)

else:return
render_template('hsearch_in_ward.html',the_name=pname,the_age=Age,the_wardno=wardno,the_gender=ge,the_contno=cno,row_titles=l,the_data=contents1,the_message=message1)

@app.route('/confirm',methods=['GET','POST'])def confirm():

contents=functions.hospital_details(request.form['hid'])

if functions.check_hospital_id(request.form['hid'])==0:

return redirect('/hsearch')

cost_without_v=functions.avg_cost_without_v(request.form['hid'])
cost_with_v=functions.avg_cost_with_v(request.form['hid'])

if cost_with_v!=0:

cost=cost_with_v *15

else:

cost=cost_without_v * 15

return

render_template('final.html',the_name=contents[0][3],the_id=contents[0][2],the_cno=contents[0][1],the_wno=contents[0][0],the_addr=contents[0][4],the_ambs=contents[0][5],the_cost=cost) if_name==”main_”: app.run(debug=True)

4.2 TRIGGERS AND STORED PROCEDURES

We have implemented triggers in this project. Once we update the patient_list table in the database, the bed count in the medical facility is incremented based on the allotment of the bed to the respective patient.

Functions

We have also put functions to use in the implementation of this project. Two functions have been created. One, to obtain the average expenses that might be incurred based on the selection of hospital. The other function achieves the same output but takes into account the ventilator requirement and kicks in only if the patient is a senior citizen.

Indexes

We have 4 main indexes that have been used in the database section.

1. Primary index for the ward number on the wards table.
2. Secondary index for the war number on the hospitals table.
3. Index for the hospital id field on the hospitals table.
4. Index for the patient id field on the patient list table.

4.3 RESULT

We've also used views for the display section on the webpage. These have been created mainly to achieve a difference in the output between the hospitals with or without ventilators. Separate views have been created so that the ventilator availability feature is clearly distinguishable in the hospitals.

The resulting system is able to:

- Minimize the waiting time for patients during an emergency.
- Reduces the waiting time for patients who need immediate surgery.
- Reduces the cost of medication by eliminating the cost of drugs going to empty beds.
- Smooth bed management in hospitals creates a better and safer environment for patients and a more efficient working environment for staff.
- Effective Bed management is critical to enabling the successful flow of patients through a healthcare facility.
- It involves the medical care, physical resources, and internal systems needed to get patients from the point of admission.

Chapter 5

TESTING

5.1 SOFTWARE TESTING

Testing is the process used to help identify correctness, completeness, security and quality of developed software. This includes executing a program with the intent of finding errors. It is important to distinguish between faults and failures. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. Most testing occurs after system requirements have been defined and then implemented in testable programs.

5.2 MODULE TESTING AND INTEGRATION

Module testing is a process of testing the individual subprograms, subroutines, classes, or procedures in a program. Instead of testing whole software program at once, module testing recommend testing the smaller building blocks of the program. It is largely white box oriented. The objective of doing Module testing is not to demonstrate proper functioning of the module but to demonstrate the presence of an error in the module. Module testing allows implementing of parallelism into the testing process by giving the opportunity to test multiple modules simultaneously.

The final integrated system too has been tested for various test cases such as duplicate entries and type mismatch.

5.3 LIMITATIONS

- It would be difficult for the patients to book a bed during emergencies like accidents, heartattacks etc.,
- The patient cannot be offered a bed when all the beds are already booked by others in theparticular hospital.
- It would be difficult for the patient to provide a ventilators when there are no availableventilators facilities in particular hospital for a huge numbers.
- As the functionality of the hospitals does not depends with this platform, hence the patients may face difficulties during the improper hospital functionality.

Chapter 6

SNAPSHOTS

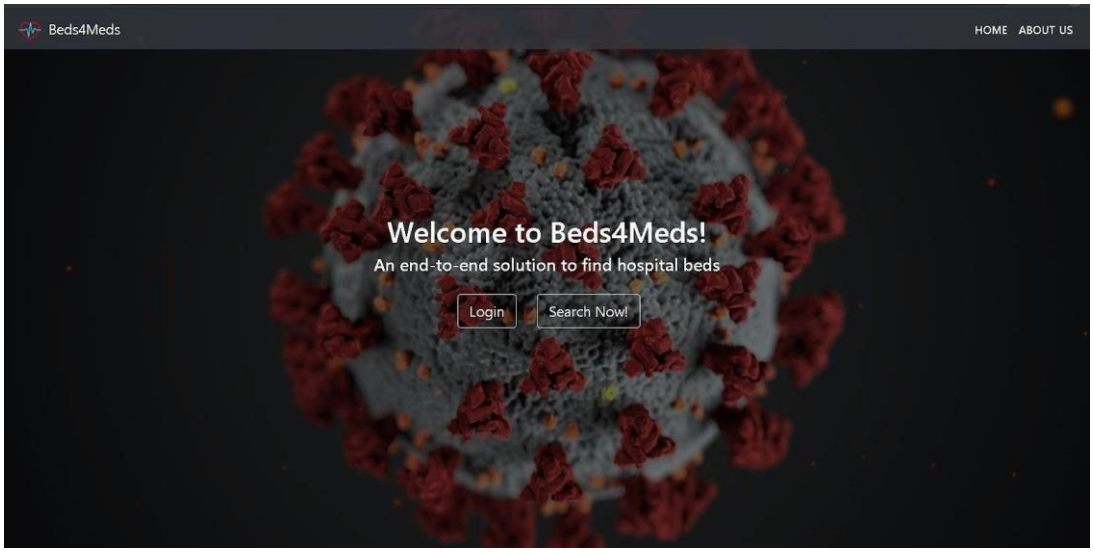

Figure 6.1: Home-Page

Can navigate to either of the modules from here

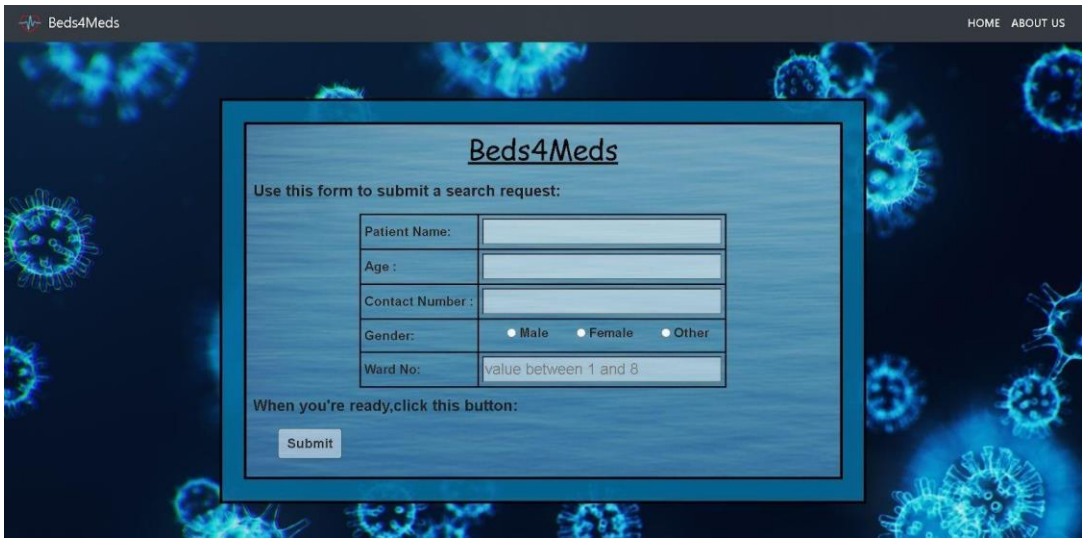

Figure 6.2: Search Module

The patient can search for hospitals here

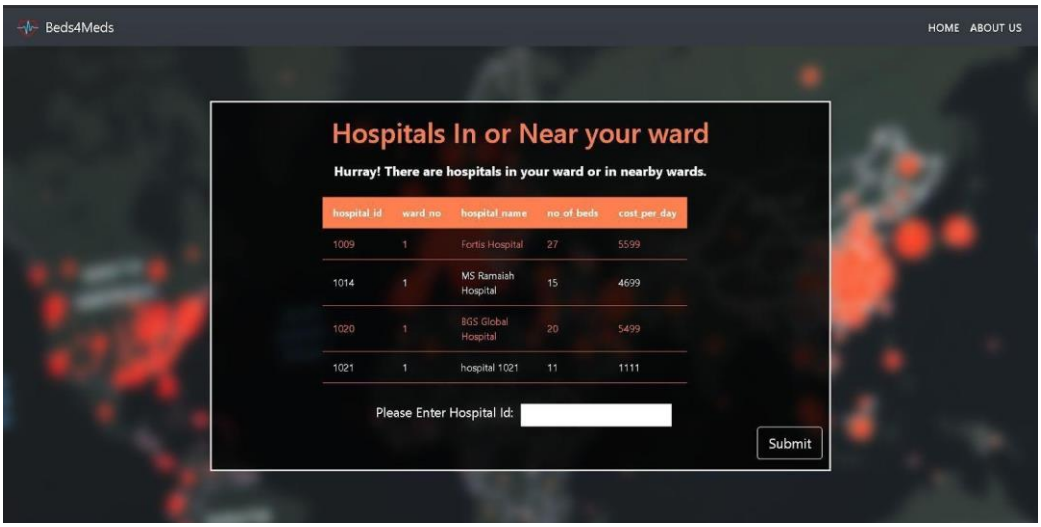

Figure 6.3: Search Results

Based on user-provided criteria and open for selection

Figure 6.4: The Final page of Search Module

Search culminates in the results. Everything you need to know. In 1 Page

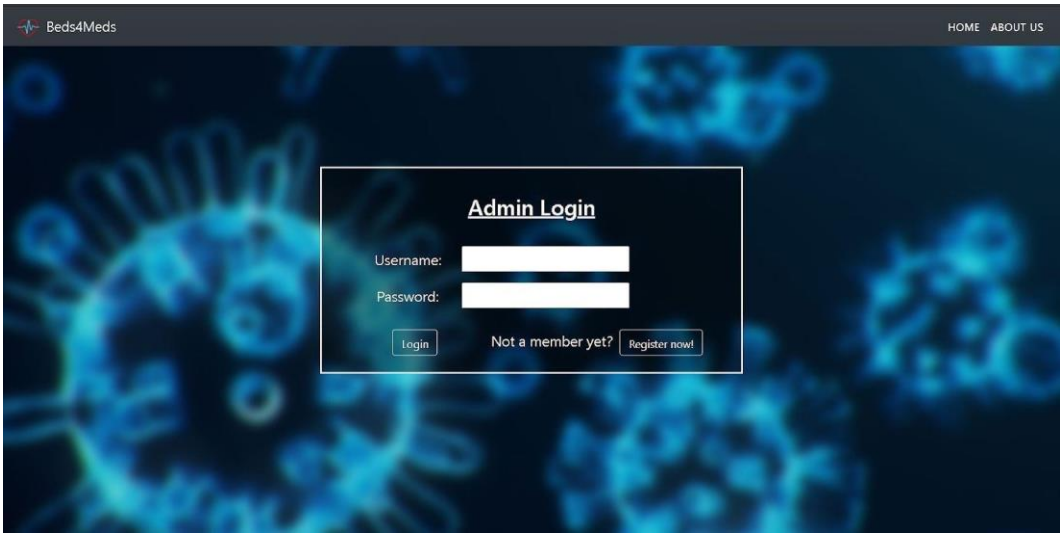

Figure 6.5: The Login for Hospital Module

The staff can login here and gain access to their account

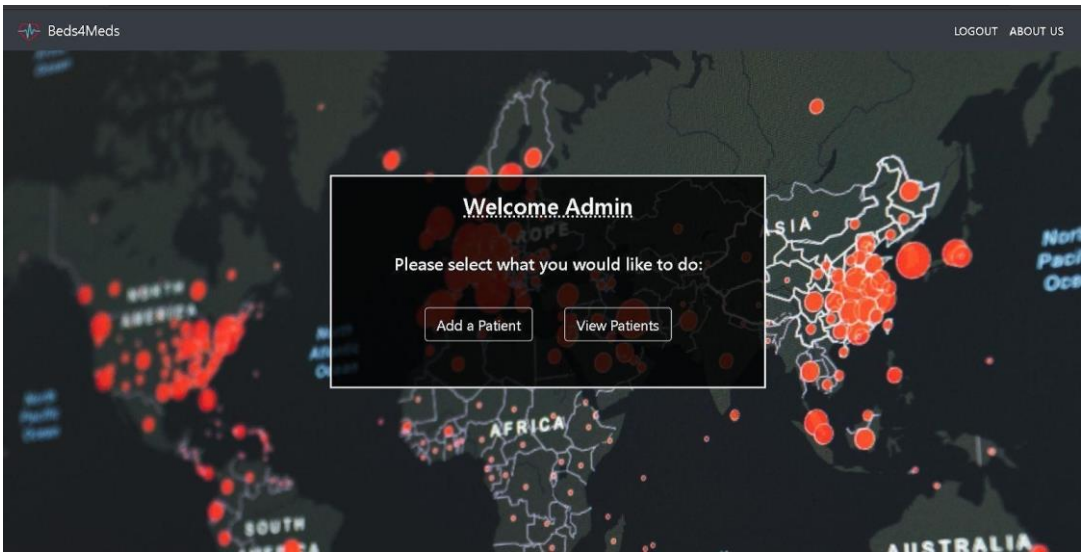

Figure 6.6: Home-Page for Hospitals

Can choose to view/add/discharge patients here

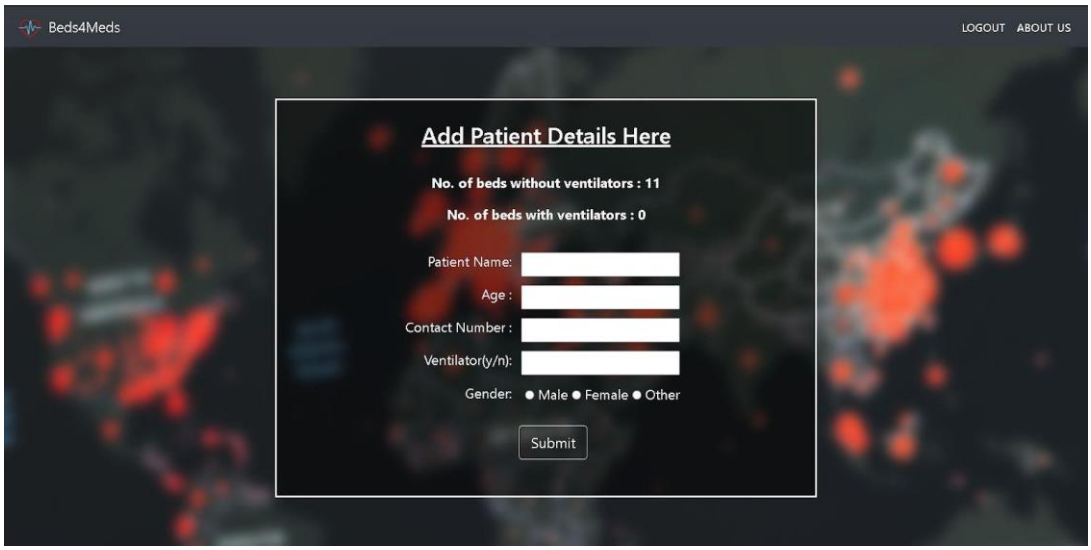

Figure 6.7: Add A Patient

Functionality to admit a new patient at the facility

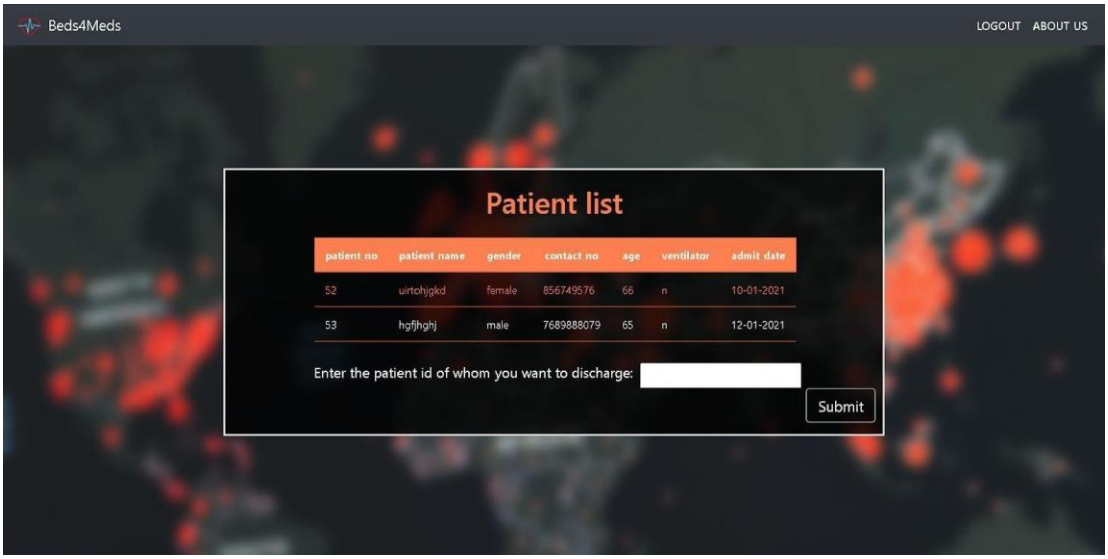

Figure 6.8: The Patient list of Hospitals Module

We can view all the patients currently housed in the hospital

Chapter 7

CONCLUSION

In summary we can conclude that Beds4Meds as a platform is immensely beneficial to both, the patients and the hospitals and provides a much needed connection between the two. It solves countless problems such as finding a hospital, maintaining information publicly, and documenting the medical facilities in a given geographical area for further use.

It can also be concluded that such a portal can serve to bridge the huge communication gap between the hospitals and the patients by making all information transparent like it is in many developed countries.

There exist certain initiatives wherein a small fragment of our platform's functionality exists but those are of very limited use and only such an integrated application like Beds4Meds can boast of having the potential to service citizens and medical facilities all around the nation.

It's decentralised system of database updating and maintenance also ensures higher security of sensitive data and since each account is secured, the risks of information leak is very low if not nullified.

The uncomplicated and simplistic approach that has been taken to design the UI and application as a whole is the reason for the easy and smooth User-experience that the platform provides. This ensures persons of all ages, whether or not internet/technology savvy can make use of and enjoy the benefits of our Beds4Meds application.

All in all, this platform can prove to be a gamechanger in the arena of medical technology as it is an all-new approach to unify India's medical society which is largely divided into Government and Private.

Chapter 8

FUTURE ENHANCEMENTS

Scale-up

Beds4Meds as a platform has immense scope for enhancement. We have built this platform as a not-for-profit method to help and service citizens of a city but this platform can be scaled up and made to be a system that works on a country-wide level.

Booking System

Beds4Meds currently does not have any connection with the hospitals or the patients. It purely provides an interface to connect the two. However, it hosts the potential for the creators to partner with medical facilities around the nation and give birth to a bed-booking service where commissions can be made for profit and monetary gain can be achieved for both hospitals and platforms.

Payment Option

Alternatively, Beds4Meds can also be run as a semi-commercial platform where the details of hospitals can be shown for free but priority access to the hospitals and pre-booking of the rooms can be added on a pay-to-use basis. This allows both a free and a paid service to co-run on the platform. Thus, if a payment gateway can be included in the platform, the potential of commercialising this service is limitless.

Social Media

Social media like Facebook and Instagram can be integrated into the platform in order to automatically notify selected near and dear ones of the infected patient so that he/she doesn't have to waste anytime.

REFERENCES

1. For help in Flask Head First Python, 2nd Editionby Paul Barry Released November 2016 Publisher(s): O'Reilly Media, Inc.ISBN: 9781491919538

2. Relational database concepts from https://www.geeksforgeeks.org/relational-model-in-dbms/

3. Inspiration for solving the crisis from https://www.deccanherald.com/city/top-bengaluru-stories/bengaluru-faces-icu-bed-shortage- after-slight-spike-in-covid-19-cases-887159.html

4. To fix errors https://stackoverflow.com/questions/15831364/flask-error-handler

5. For HTML tips https://www.w3schools.com/html/default.asp

6. For CSS and Bootstrap tagshttps://getbootstrap.com/