Chapter 1

Introduction

Human Body is a very complex and sophisticated structure and comprises of millions of functions. All these complicated functions have been understood by man him, part-by-part their research and experiments. As science and technology progressed, medicine became an integral part of the research. Gradually, medical science became an entirely new branch of science. As of today, the Health Sector comprises of Medical institutions i.e. Hospitals, HOSPITALs etc. research and development institutions and medical colleges. Thus the Health sector aims at providing the best medical facilities to the common man

1.1 Background Study

System Analysis is a separation of a substance into parts for study and their implementation and detailed examination. Before designing any system it is important that the nature of the business and the way it currently operates are clearly understood. The detailed examination provides the specific data required during designing in order to ensure that all the client's requirements are fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study. Rather it would not be wrong to say that the analysis and feasibility phases overlap. High-level analysis begins during the feasibility study. Though analysis is represented as one phase of the system development life cycle (SDLC), this is not true. Analysis begins with system initialization and continues until its maintenance. Even after successful implementation of the system, analysis may play its role for periodic maintenance and up gradation of the system. One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding of the requirements is the poor planning of system analysis.

1.2 Problem Statement

Since Hospital is associated with the lives of common people and their day-to-day routines so I decided to work on this project.

The manual handling of the record is time consuming and highly prone to error. The purpose of this project is to automate or make online, the process of day-to-day activities like Room activities, Admission of New Patient, Discharge of Patient, Assign a Doctor, and finally compute the bill etc. I have tried my best to make the complicated process Hospital Management System as simple as possible using Structured & Modular technique & Menu oriented interface. I have tried to design the software in such a way that user may not have any difficulty in using this package & further expansion is possible without much effort. Even though I cannot claim that this work to be entirely exhaustive, the main purpose of my exercise is perform each Hospital's activity in computerized way rather than manually which is time consuming.

I am confident that this software package can be readily used by non-programming personal avoiding human handled chance of error.

1.3 Motivation and Objectives of the Project

Hospital are the essential part of our lives, providing best medical facilities to people suffering from various ailments, which may be due to change in climatic conditions, increased work-load, emotional trauma stress etc. It is necessary for the hospitals to keep track of its day-to-day activities & records of its patients, doctors, nurses, ward boys and other staff personals that keep the hospital running smoothly & successfully.

But keeping track of all the activities and their records on paper is very cumbersome and error prone. It also is very inefficient and a time-consuming process Observing the continuous increase in population and number of people visiting the hospital. Recording and maintaining all these records is highly unreliable, inefficient and error-prone. It is also not economically & technically feasible to maintain these records on paper. Thus keeping the working of the manual system as the basis of our project. We have developed an automated version of the manual system, named as "Administration support system for medical institutions".

The main aim of our project is to provide a paper-less hospital up to 90%. It also aims at providing low-cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user-system interaction and also provides robust & reliable storage and backup facilities.

1.4 Advantages

The software helps to handle the entire administration of hospitals and healthcare facilities. Typically, such a software includes various modules that help doctors manage their assignments and schedules, carry out patient registration, maintain store inventory records, keep track of medicine, administration, maintain blood bank (with available blood type) details, individual record of patients with their test reports, nursing and housekeeping service details, financial information, including final billing & payments, insurance details and much more. After the customized software is implemented and integrated into the system, patient care and hospital administration becomes an easy job.

Chapter 2

REQUIREMENT AND SPECIFICATON

2.1 Hardware Requirements

Processor: Intel core i3/i5

RAM : 2 GB(minimum)

Hard disk : 1TB

Input device : Mouse and Key board

Output device : Monitor

2.2 Software Requirements

Front End : PHP

Back End : MySQL

Operating System : windows 10

Browser : Google chrome

Chapter 3

SYSTEM DESIGN

3.1 ER Diagram

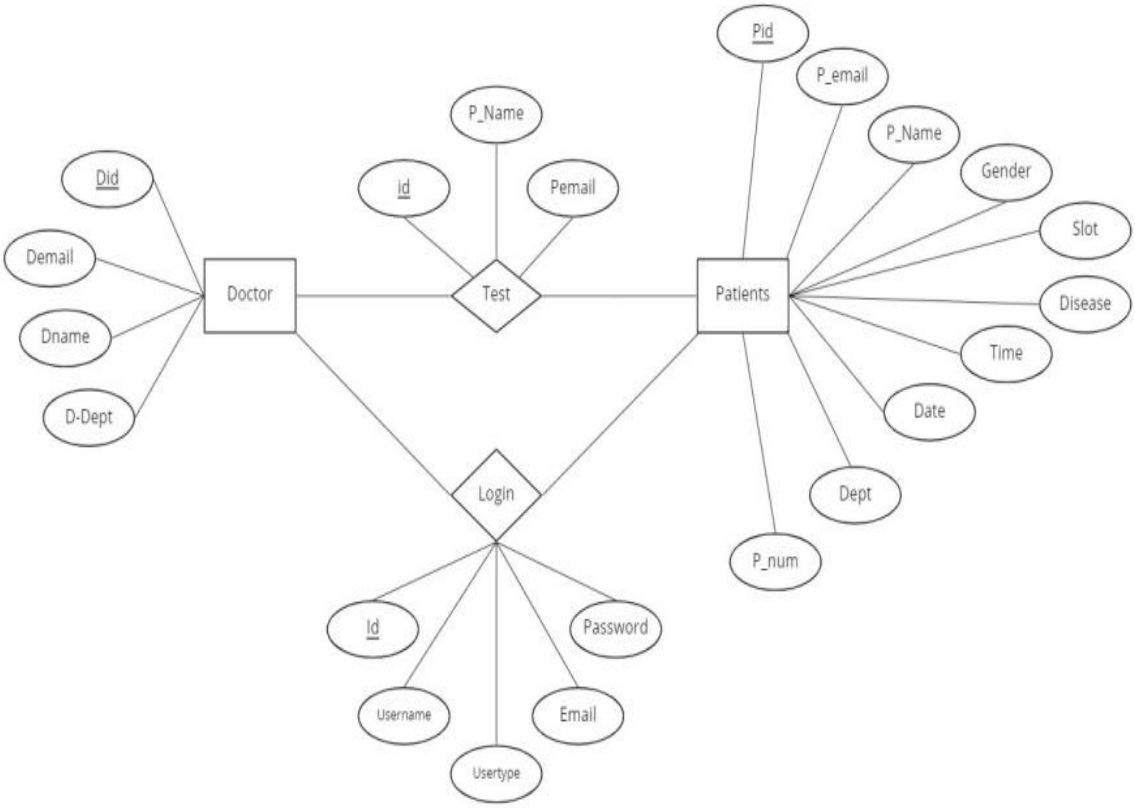


Fig 3.1:ER Diagram

The Fig 3.1 shows , ER DIAGRAM describes inter related things of interest in a specific domain of knowledge.

3.2 Schema diagram

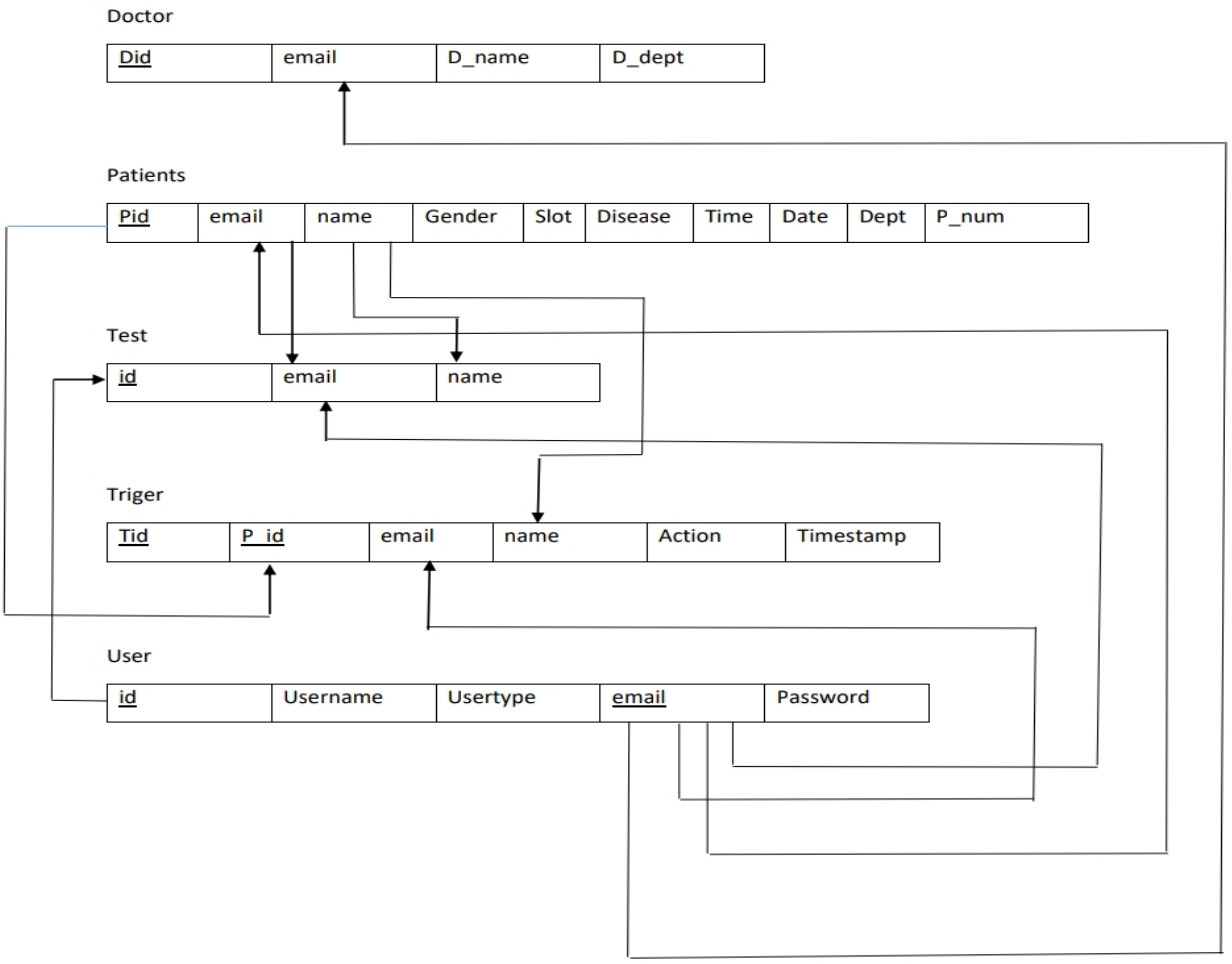


Fig 3.2: Schema Diagram

The Fig 3.2 shows, the schema diagram of a database system is its structure describe in formal language supported by the database management system. The formal definition of data base schema is set of formulas integrity constraints imposed on the database.

3.3 Implementation

TABLE 3.3.1: DOCTORS TABLE

Column Name	Data type and Size	Constraint	Description
did	Int(11)	Primary key	It uniquely identifies the doctor id
email	Varchar(50)		Represent email
doctorname	Varchar(50)		Represent Doctor name
dept	Varchar(100)		Represent Doctor's department

TABLE 3.3.2: PATIENTS TABLE

Column Name	Data type and Size	Constraint	Description
Pid	Int(11)	Primary key	It uniquely identifies the patient id
Email	Varchar(50)		Represent email
Name	Varchar(50)		Represent name
Gender	Varchar(50)		Represent gender
Slot	Varchar(50)		Represent slots
Disease	Varchar(50)		Represent disease
Time	Time		Represent time
Date	Date		Represent date
Dept	Varchar(50)		Represent department
number	Varchar(10)		Represent number

TABLE 3.3.3: TEST TABLE

Column Name	**Data type and Size**	**Constraint**	**Description**
Id name email	Int(11) Varchar(20) Varchar(20)	Primary key	It uniquely identifies the patient id Represent name Represent email

TABLE 3.3.4: TRIGER TABLE

Column Name	Data type and Size	Constraint	Description
Tid Pid Email Name Action timestamp	Int(11) Int(11) Varchar(50) Varchar(20) Varchar(20) datetime	Primary key Primary key	It uniquely identifies the trigr id It uniquely identifies the patirnt id Represent email Represent name Represent action Represent timestamp

TABLE 3.3.5: USER TABLE

Column Name	Data type and Size	Constraint	Description
Id Username Usertype Email password	Int(11) Varchar(50) Varchar(50) Varchar(50) Varchar(1000)	Primary key	It uniquely identifies the user id Represent user name Represent user type Represent email Represent password

Chapter 4

TESTING

4.1 Integration Testing

Integration testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each modules effect on the entire program model. After integration testing the project works successfully.

4.2 Unit Testing

Unit testing performed on each module or block of code during development. Unit testing is normally done by the programmer who writes the code.

4.3 System Testing

System testing done by a professional testing agent on the completed software product before it is introduced to the market**.**

4.4 Acceptance Testing

Acceptance testing is a beta testing of the product done by the actual end user.

4.5 Recovery Testing

Recovery testing is done to demonstrate a software salutation is reliable, trustworthy and can successfully recoup form possible crashes.

4.6 Functional Testing

Functional Testing also known as functional completeness testing. Functional Testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could to improve it during functional testing.

4.7 Hardware/Software Testing

IBM refers to Hardware/Software testing as “HW/SW Testing”. This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

4.8 Security Testing

Security Testing is a variant of Software Testing which ensures, that system and applications in an organization, are free from any loopholes that may cause a big loss. Security testing of any system is about finding all possible loopholes and weaknesses of the system which might result into a loss of information at the hands of the employees or outsiders of the Organization.

Chapter 5

5.1 Snapshots

Sign up page

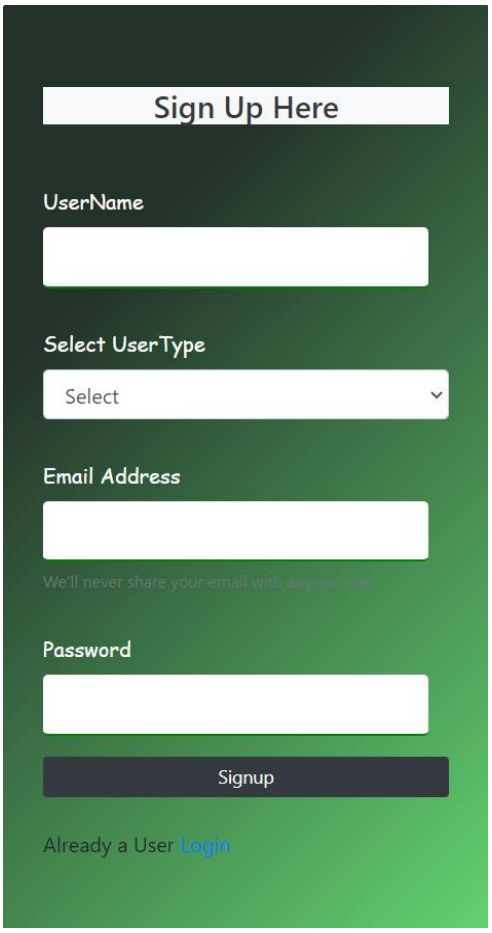


Fig 5.1.1: Sign up page

The Fig. 5.1.1 shows Sign up page, using Username, User type, Email, Password user can sign up.

Login page

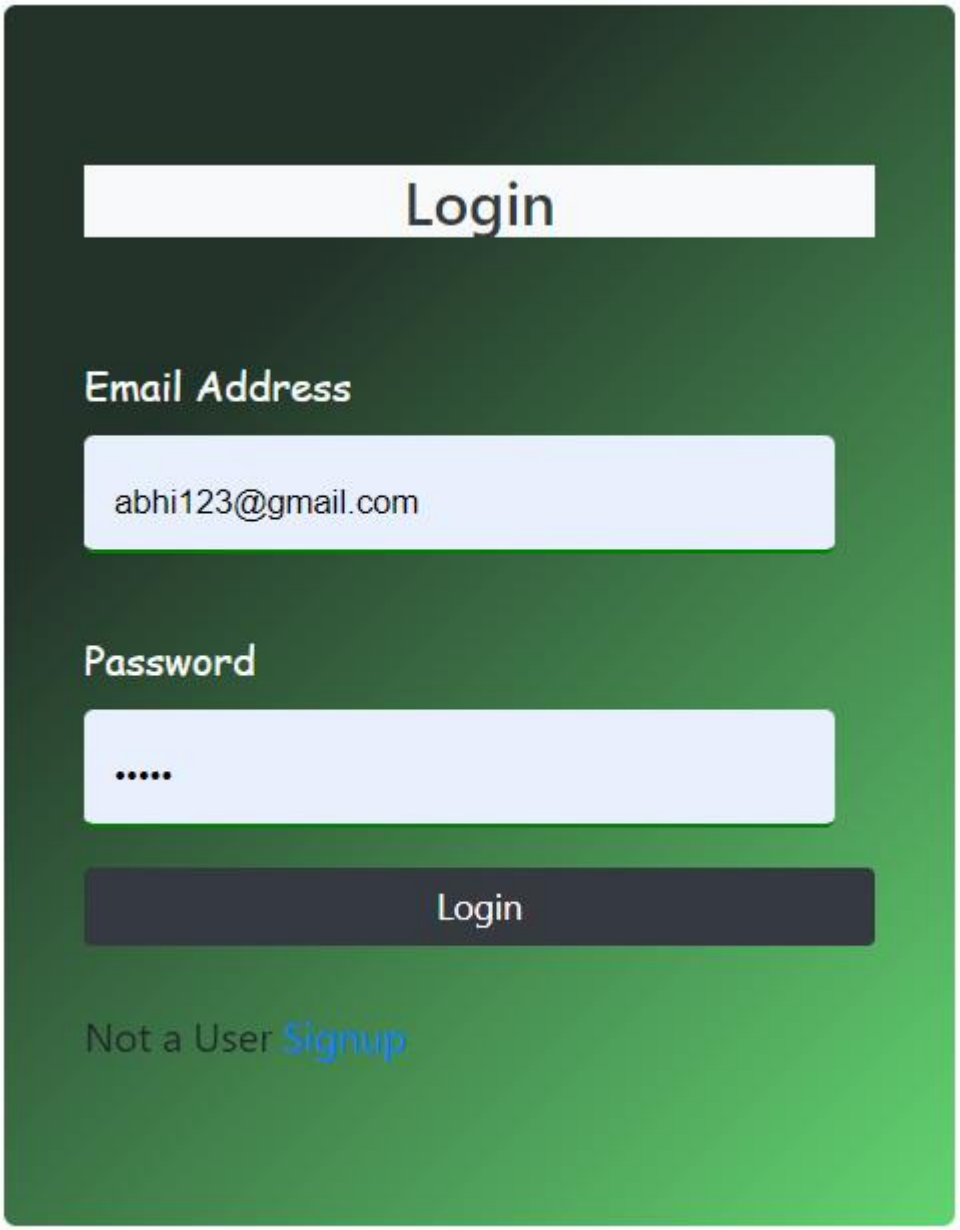


Fig 5.1.2: Login page

The Fig. 5.1.2 shows Login page, using User Email and Password user can Login.

Patients Booking

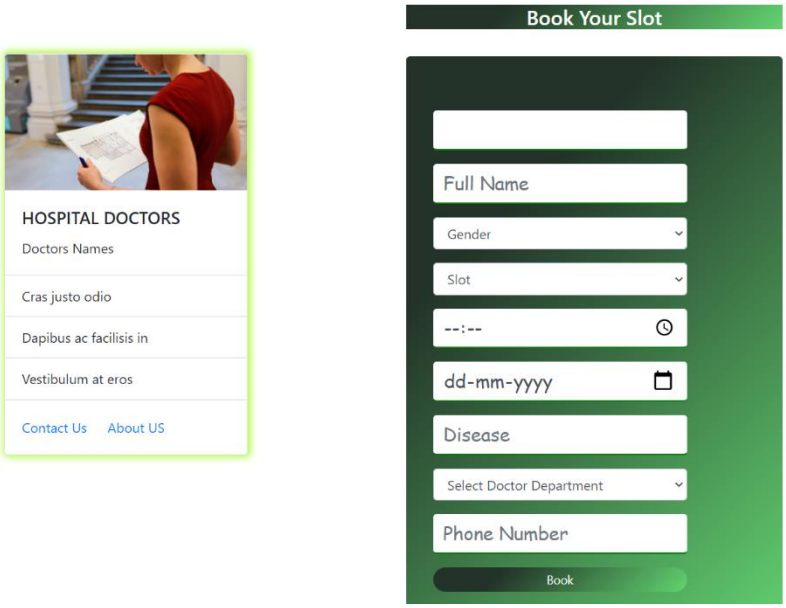


Fig 5.1.3: Patients booking

The Fig. 5.1.3, Patients can book their slots using email, full name, Gender, slot timing, date, disease, doctor department and phone number.

Patients Booking Details

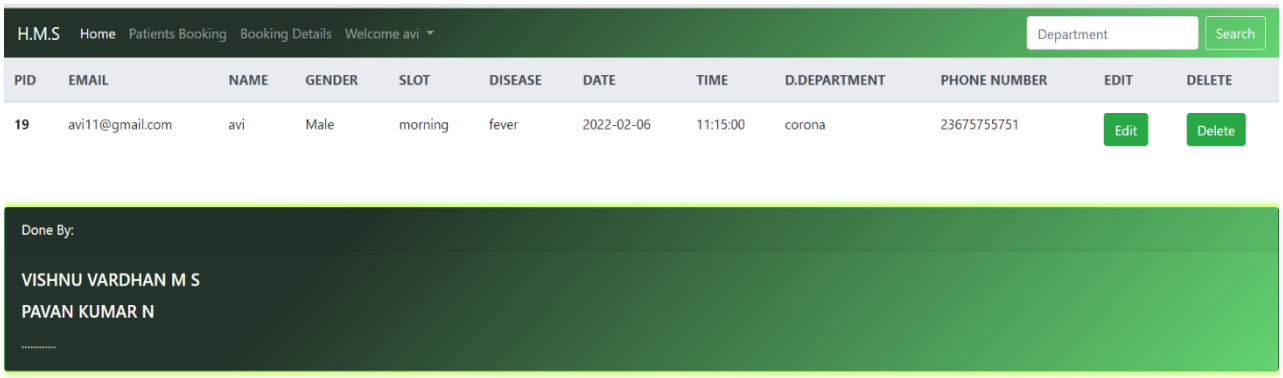


Fig 5.1.4: Patients booking details

The Fig. 5.1.4, Details of the patient who booked the slot,the patient can the edit or delete the details.

Doctors Not Availability

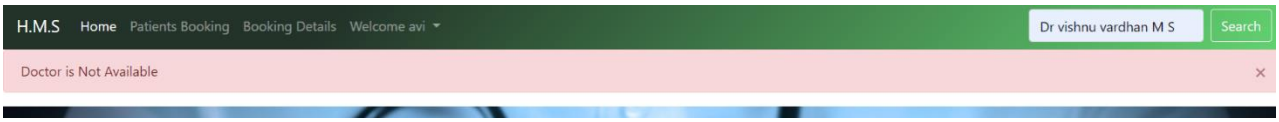


Fig 5.1.5: Doctors not availability

The Fig. 5.1.5 shows whether the Doctor is available. In this case the Doctor is not available.

Doctors Availability

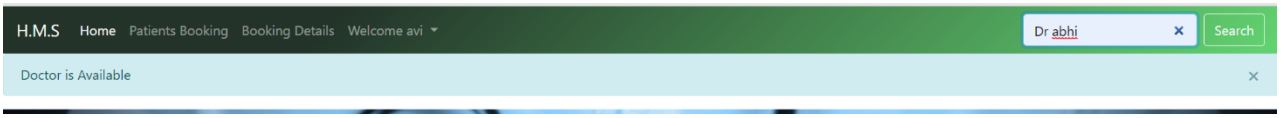


Fig 5.1.6: Doctors availability

The Fig. 5.1.6 Shows the user that the Doctor is available.

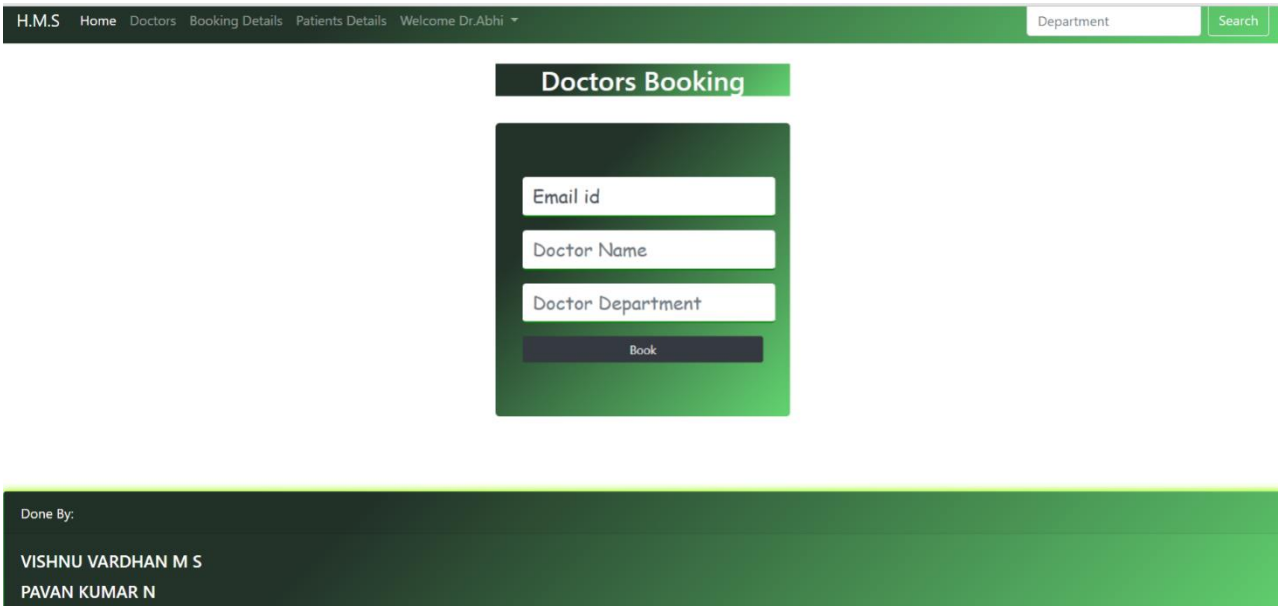
Home page

Fig 5.1.7: Home page

The Fig. 5.1.7, Users can access to other features from the home page.

Doctors Booking

Fig 5.1.8: Doctors booking



The Fig. 5.1.8, Doctors books the open slots by entering Email id, Doctor's name, Doctor's Department.

Patients Details

PID	EMAIL	NAME	GENDER	SLOT	DISEASE	DATE	TIME	D.DEPARTMENT	PHONE NUMBER	EDIT	DELETE
2	anees1@gmail.com	anees1 rehman khan	Male1	evening1	cold1	2020-02-02	21:20:00	ortho11predict	9874561110	Edit	Delete
5	patient@gmail.com	patien	Male	morning	fevr	2020-11-18	18:06:00	Cardiologists	9874563210	Edit	Delete
7	patient@gmail.com	anees	Male	evening	cold	2020-11-05	22:18:00	Dermatologists	9874563210	Edit	Delete
8	patient@gmail.com	anees	Male	evening	cold	2020-11-05	22:18:00	Dermatologists	9874563210	Edit	Delete
9	aneesurrehman423@gmail.com	anees	Male	morning	cold	2020-11-26	17:27:00	Anesthesiologists	9874563210	Edit	Delete
10	anees@gmail.com	anees	Male	evening	fever	2020-12-09	16:25:00	Cardiologists	9874589654	Edit	Delete
15	khushi@gmail.com	khushi	Female	morning	corona	2021-01-23	20:42:00	Anesthesiologists	9874563210	Edit	Delete
16	khushi@gmail.com	khushi	Female	evening	fever	2021-01-31	15:46:00	Endocrinologists	9874587496	Edit	Delete
17	aneeqah@gmail.com	aneeqah	Female	evening	fever	2021-01-23	15:48:00	Endocrinologists	9874563210	Edit	Delete
18	ravi12@gmail.com	ravi	Male	morning	headache	2022-01-20	11:45:00	corona	88617704901	Edit	Delete

127.0.0.1:5000/bookings

Fig 5.1.9 Patients details

The Fig. 5.1.9, Doctor can view all the details the patients and edit.

Patients update the slot details

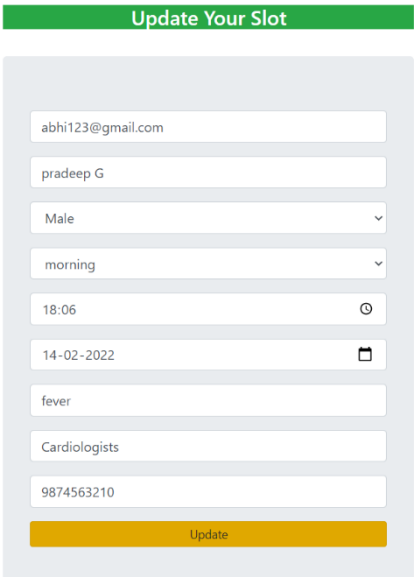


Fig 5.2.0: Patients update the slot details

The Fig. 5.2.0, Patient slot update form, patient can edit the details.

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

This project has been a rewarding experience in more than one way. The entire project work has enlightened us in the following areas. a) We have gained an insight into the working of the HOSPITAL. This represents a typical real world situation. b) Our understanding of database design has been strengthened this is because in order to generate the final reports of database designing has to be properly followed. c) Scheduling a project and adhering to that schedule creates a strong sense of time management. d) Sense of teamwork has developed and confidence of handling real life project has increased to a great extent. e) Initially, there were problem with the validation but with discussions, we were to implement validations

6.2 Limitations of the system

o Online payment is not available at this version.

o Data delete & edit system is not available for all section.

o User account not verified by Mobile SMS not available in this system.

o Loss of data due to mismanagement.

APPENDIX

Main.py

```
fromflaskimportFlask,render_template,request,session,redirect,url_for,flash
fromflask_sqlalchemyimportSQLAlchemy
fromflask_loginimportUserMixin
fromwerkzeug.securityimportgenerate_password_hash,check_password_hash
fromflask_loginimportlogin_user,logout_user,login_manager,LoginManager
fromflask_loginimportlogin_required,current_user
fromflask_mailimportMail
importjson

withopen('config.json','r') asc:
    params=json.load(c)["params"]

# MY db connection
local_server=True
app=Flask(__name__)
app.secret_key='aneeqah'

# this is for getting unique user access
login_manager=LoginManager(app)
login_manager.login_view='login'

# SMTP MAIL SERVER SETTINGS

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)


@login_manager.user_loader
defload_user(user_id):
    returnUser.query.get(int(user_id))




#
app.config['SQLALCHEMY_DATABASE_URL']='mysql://username:password@localhost/databas_table_name'
app.config['SQLALCHEMY_DATABASE_URI']='mysql://root:@localhost/hmdbms1'
db=SQLAlchemy(app)



# here we will create db models that is tables
classTest(db.Model):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(100))
    email=db.Column(db.String(100))

classUser(UserMixin,db.Model):
    id=db.Column(db.Integer,primary_key=True)
    username=db.Column(db.String(50))
    usertype=db.Column(db.String(50))
    email=db.Column(db.String(50),unique=True)
    password=db.Column(db.String(1000))

classPatients(db.Model):
    pid=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
```

```
    gender=db.Column(db.String(50))
    slot=db.Column(db.String(50))
    disease=db.Column(db.String(50))
    time=db.Column(db.String(50),nullable=False)
    date=db.Column(db.String(50),nullable=False)
    dept=db.Column(db.String(50))
    number=db.Column(db.String(50))

classDoctors(db.Model):
    did=db.Column(db.Integer,primary_key=True)
    email=db.Column(db.String(50))
    doctorname=db.Column(db.String(50))
    dept=db.Column(db.String(50))

classTrigr(db.Model):
    tid=db.Column(db.Integer,primary_key=True)
    pid=db.Column(db.Integer)
    email=db.Column(db.String(50))
    name=db.Column(db.String(50))
    action=db.Column(db.String(50))
    timestamp=db.Column(db.String(50))



# here we will pass endpoints and run the fuction
@app.route('/')
defindex():
    a=params['gmail-user']
    returnrender_template('index.html')
```

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

    ifrequest.method=="POST":

        email=request.form.get('email')
        doctorname=request.form.get('doctorname')
        dept=request.form.get('dept')

                query=db.engine.execute(f"INSERT INTO `doctors` (`email`,`doctorname`,`dept`) VALUES ('{email}','{doctorname}','{dept}')")
        flash("Information is Stored","primary")

    returnrender_template('doctor.html')



@app.route('/patients',methods=['POST','GET'])
@login_required
defpatient():
    doct=db.engine.execute("SELECT * FROM `doctors`")

    ifrequest.method=="POST":
        email=request.form.get('email')
        name=request.form.get('name')
        gender=request.form.get('gender')
        slot=request.form.get('slot')
        disease=request.form.get('disease')
        time=request.form.get('time')
        date=request.form.get('date')
        dept=request.form.get('dept')
        number=request.form.get('number')
```

```
        subject="HOSPITAL MANAGEMENT SYSTEM"
                                                query=db.engine.execute(f"INSERT INTO `patients`
(`email`,`name`,                          `gender`,`slot`,`disease`,`time`,`date`,`dept`,`number`)          VALUES
('{email}','{name}','{gender}','{slot}','{disease}','{time}','{date}','{dept}','{number}')")


# mail starts from here


       #mail.send_message(subject,  sender=params['gmail-user'],  recipients=[email],body=f"YOURbOOKING  IS
CONFIRMED THANKS FOR CHOOSING US \nYour Entered Details are :\nName: {name}\nSlot: {slot}")



        flash("Booking Confirmed","info")


    returnrender_template('patient.html',doct=doct)


@app.route('/bookings')
@login_required
defbookings():
    em=current_user.email
    ifcurrent_user.usertype=="Doctor":
        query=db.engine.execute(f"SELECT * FROM `patients`")
        returnrender_template('booking.html',query=query)
    else:
        query=db.engine.execute(f"SELECT * FROM `patients` WHERE email='{em}'")
        returnrender_template('booking.html',query=query)



@app.route("/edit/<string:pid>",methods=['POST','GET'])
@login_required
defedit(pid):
    posts=Patients.query.filter_by(pid=pid).first()
    ifrequest.method=="POST":
```

```
        email=request.form.get('email')
        name=request.form.get('name')
        gender=request.form.get('gender')
        slot=request.form.get('slot')
        disease=request.form.get('disease')
        time=request.form.get('time')
        date=request.form.get('date')
        dept=request.form.get('dept')
        number=request.form.get('number')
        db.engine.execute(f"UPDATE `patients` SET `email` = '{email}', `name` = '{name}', `gender` = '{gender}', `slot` = '{slot}', `disease` = '{disease}', `time` = '{time}', `date` = '{date}', `dept` = '{dept}', `number` = '{number}' WHERE `patients`.`pid` = {pid}")
        flash("Slot is Updates","success")
        returnredirect('/bookings')

    returnrender_template('edit.html',posts=posts)

@app.route("/delete/<string:pid>",methods=['POST','GET'])
@login_required
defdelete(pid):
    db.engine.execute(f"DELETE FROM `patients` WHERE `patients`.`pid`={pid}")
    flash("Slot Deleted Successful","danger")
    returnredirect('/bookings')

@app.route('/signup',methods=['POST','GET'])
defsignup():
    ifrequest.method=="POST":
        username=request.form.get('username')
        usertype=request.form.get('usertype')
```

```
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()
        ifuser:
            flash("Email Already Exist","warning")
            returnrender_template('/signup.html')
        encpassword=generate_password_hash(password)

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

        # this is method 2 to save data in db
        # newuser=User(username=username,email=email,password=encpassword)
        # db.session.add(newuser)
        # db.session.commit()
        flash("Signup Succes Please Login","success")
        returnrender_template('login.html')




    returnrender_template('signup.html')

@app.route('/login',methods=['POST','GET'])
deflogin():
    ifrequest.method=="POST":
        email=request.form.get('email')
        password=request.form.get('password')
        user=User.query.filter_by(email=email).first()

        ifuserandcheck_password_hash(user.password,password):
            login_user(user)
```

```
            flash("Login Success","primary")
            returnredirect(url_for('index'))
        else:
            flash("invalid credentials","danger")
            returnrender_template('login.html')




    returnrender_template('login.html')

@app.route('/logout')
@login_required
deflogout():
    logout_user()
    flash("Logout SuccessFul","warning")
    returnredirect(url_for('login'))



@app.route('/test')
deftest():
    try:
        Test.query.all()
        return'My database is Connected'
    except:
        return'My db is not Connected'



@app.route('/details')
@login_required
defdetails():
    # posts=Trigr.query.all()
```

```
    posts=db.engine.execute("SELECT * FROM `trigr`")
    returnrender_template('trigers.html',posts=posts)


@app.route('/search',methods=['POST','GET'])
@login_required
defsearch():
    ifrequest.method=="POST":
        query=request.form.get('search')
        dept=Doctors.query.filter_by(dept=query).first()
        name=Doctors.query.filter_by(doctorname=query).first()
        ifname:

            flash("Doctor is Available","info")
        else:

            flash("Doctor is Not Available","danger")
    returnrender_template('index.html')





app.run(debug=True)
```

Booking.html

```
{% extends 'base.html' %}


{% block title %}
Booking
{% endblock title %}


{% block body %}
{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}

<divclass="alert alert-{{category}} alert-dismissible fade show"role="alert">
   {{message}}

  <buttontype="button"class="close"data-dismiss="alert"aria-label="Close">
    <spanaria-hidden="true">&times;</span>
  </button>
</div>


  {% endfor %}
  {% endif %}
  {% endwith %}
<tableclass="table">
  <theadclass="thead-light">
    <tr>
      <thscope="col">PID</th>
      <thscope="col">EMAIL</th>
      <thscope="col">NAME</th>
      <thscope="col">GENDER</th>
      <thscope="col">SLOT</th>
```

```
    <thscope="col">DISEASE</th>
    <thscope="col">DATE</th>
    <thscope="col">TIME</th>
    <thscope="col">D.DEPARTMENT</th>
    <thscope="col">PHONE NUMBER</th>
    <thscope="col">EDIT</th>
    <thscope="col">DELETE</th>

  </tr>
 </thead>
 <tbody>
 {% for post in query %}
  <tr>
    <thscope="row">{{post.pid}}</th>
    <td>{{post.email}}</td>
    <td>{{post.name}}</td>
    <td>{{post.gender}}</td>
    <td>{{post.slot}}</td>
    <td>{{post.disease}}</td>
    <td>{{post.date}}</td>
    <td>{{post.time}}</td>
    <td>{{post.dept}}</td>
    <td>{{post.number}}</td>
   <td><ahref="/edit/{{post.pid}}"><buttonclass="btnbtn-success">Edit </button></a></td>
          <td><ahref="/delete/{{post.pid}}"><buttononclick="returnconfirm('Are you sure to Delete
data');"class="btnbtn-success">Delete </button></a></td>
   </tr>
{% endfor %}

 </tbody>
</table>
```

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

Doctor.html

```
{% extends 'base.html' %}

{% block title %}
Doctors
{% endblock title %}

{% block body %}

<divclass="container">

<divclass="row">

<divclass="col-md-4"></div>
<divclass="col-md-4">
{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}

<divclass="alert alert-{{category}} alert-dismissible fade show"role="alert">
    {{message}}

  <buttontype="button"class="close"data-dismiss="alert"aria-label="Close">
    <spanaria-hidden="true">&times;</span>
```

```
 </button>
</div>

 {% endfor %}
 {% endif %}
 {% endwith %}
<br>
<h2class="text-center text-white bg-dark">Doctors Booking</h2>

<br>
<formaction="/doctors"method="post"class="jumbotron">

<divclass="form-group">

<inputtype="email"class="form-control"name="email"value={{current_user.email}}  required>

</div>

<divclass="form-group">

<inputtype="text"class="form-control"name="doctorname"placeholder="Doctor Name"required>

</div>

<divclass="form-group">

<inputtype="text"class="form-control"name="dept"placeholder="Doctor Department"required>

</div>
```

```
 <buttontype="submit"class="btnbtn-dark btn-smbtn-block">Book</button>


</form>


</div>
<divclass="col-md-4"></div>


</div>


</div>


{% endblock body %}
```

Index.html

```
{% extends 'base.html' %}


{% block title %}
HOME
{% endblock title %}


{% block body %}


{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}


<divclass="alert alert-{{category}} alert-dismissible fade show"role="alert">
   {{message}}


 <buttontype="button"class="close"data-dismiss="alert"aria-label="Close">
```

```
  <spanaria-hidden="true">&times;</span>
 </button>
</div>


 {% endfor %}
 {% endif %}
 {% endwith %}



 <divid="carouselExampleCaptions"class="carousel slide"data-ride="carousel">
 <olclass="carousel-indicators">
  <lidata-target="#carouselExampleCaptions"data-slide-to="0"class="active"></li>
  <lidata-target="#carouselExampleCaptions"data-slide-to="1"></li>
  <lidata-target="#carouselExampleCaptions"data-slide-to="2"></li>
 </ol>
 <divclass="carousel-inner">
  <divclass="carousel-item active">
   <imgsrc="static/bg1.jpg"class="d-block w-100"alt="...">
   <divclass="carousel-caption d-none d-md-block">
    <h5>WELCOME TO H M S</h5>
    <p>Nulla vitae elitlibero, a pharetraauguemollisinterdum.</p>
   </div>
  </div>
  <divclass="carousel-item">
   <imgsrc="static/d2.jpg"class="d-block w-100"alt="...">
   <divclass="carousel-caption d-none d-md-block">
    <h5>WELCOME TO H M S</h5>
    <p>Loremipsumdolor sit amet, consecteturadipiscingelit.</p>
   </div>
  </div>
  <divclass="carousel-item">
```

```
    <imgsrc="static/d3.jpg"class="d-block w-100"alt="...">
    <divclass="carousel-caption d-none d-md-block">
      <h5>WELCOME TO H M S</h5>
      <p>Praesentcommodocursus magna, velscelerisquenislconsectetur.</p>
    </div>
  </div>
 </div>
 <aclass="carousel-control-prev"href="#carouselExampleCaptions"role="button"data-slide="prev">
  <spanclass="carousel-control-prev-icon"aria-hidden="true"></span>
  <spanclass="sr-only">Previous</span>
 </a>
 <aclass="carousel-control-next"href="#carouselExampleCaptions"role="button"data-slide="next">
  <spanclass="carousel-control-next-icon"aria-hidden="true"></span>
  <spanclass="sr-only">Next</span>
 </a>
</div>

{% endblock body %}
```

Login.html

```
{% extends 'base.html' %}

{% block title %}
Login
{% endblock title %}

{% block body %}
<divclass="container mt-3">

<divclass="row">
```

```
<divclass="col-md-4">

</div>

<divclass="col-md-4 jumbotron">

<h4class="text-centerbg-light text-dark">Login</h4>
<br>

{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}

<divclass="alert alert-{{category}} alert-dismissible fade show"role="alert">
   {{message}}

  <buttontype="button"class="close"data-dismiss="alert"aria-label="Close">
   <spanaria-hidden="true">&times;</span>
  </button>
</div>

  {% endfor %}
  {% endif %}
  {% endwith %}

<formaction="/login"method="post">
```

```
 <divclass="form-group">
   <labelfor="email">Email Address</label>
   <inputtype="email"class="form-control"id="email"name="email"required>

 </div>

 <divclass="form-group">
   <labelfor="password">Password</label>
   <inputtype="password"class="form-control"id="password"name="password"required>
 </div>

 <buttontype="submit"class="btnbtn-dark btn-smbtn-block">Login</button>
</form>

<br>
Not a User <ahref="/signup">Signup</a>

</div>

<divclass="col-md-4">

</div>

</div>

</div>

{% endblock body %}
```

Signup.html

```
{% extends 'base.html' %}

{% block title %}
Signup
{% endblock title %}

{% block body %}
<divclass="container mt-3">

<divclass="row">


<divclass="col-md-4">

</div>

<divclass="col-md-4 jumbotron">
<h4class="text-centerbg-light text-dark">Sign Up Here</h4>
<br>


{% with messages=get_flashed_messages(with_categories=true) %}
{% if messages %}
{% for category, message in messages %}

<divclass="alert alert-{{category}} alert-dismissible fade show"role="alert">
   {{message}}

  <buttontype="button"class="close"data-dismiss="alert"aria-label="Close">
    <spanaria-hidden="true">&times;</span>
  </button>
</div>


  {% endfor %}
  {% endif %}
  {% endwith %}


<formaction="/signup"method="POST">

  <divclass="form-group">
    <labelfor="username">UserName</label>
    <inputtype="text"class="form-control"id="username"name="username"required>
  </div>
```

```
 <divclass="form-group">

  <labelfor="usertype">Select UserType</label>

<selectclass="form-control"id="usertype"name="usertype"  required>
       <optionselected>Select</option>
       <optionvalue="Patient">Patient</option>
       <optionvalue="Doctor">Doctor</option>
     </select>
 </div>

  <divclass="form-group">
    <labelfor="email">Email Address</label>
    <inputtype="email"class="form-control"id="email"name="email"required>
    <smallid="emailHelp"class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>


  <divclass="form-group">
    <labelfor="password">Password</label>
    <inputtype="password"class="form-control"id="password"name="password"required>
  </div>

  <buttontype="submit"class="btnbtn-dark btn-smbtn-block">Signup</button>
</form>
<br>
Already a User <ahref="/login">Login</a>

</div>

<divclass="col-md-4">

</div>


</div>

</div>



{% endblock body %}
```

References

[1] ”Beginning PHP 4 Databases”, Wrox Press Ltd. Paperback-17, October, 2002.70-130 pp.

[2] Matt Doyle, “Beginning PHP 5.3, 2 ndedition”, October 2009. 150-270 pp.

[3] Luke Welling, Laura Thomson. Sams PHP and MySQL Web Development, 2nd edition, Paperback- 20 February, 2003. 105-209 pp.

[4] W. Jason Gilmore “Beginning PHP 5 and MySQL 5 from Novice to Professional SECOND EDITION”, Jul 9, 2008.100-150 pp.

[5] Abraham Silberschatz, Henry F. Korth and S. Sudarshan “Sixth Edition Database System Conceptsreleased”, January 28, 2010. 206-253 pp.

[6] Server-Side Scriptinghttp://php.net/manual/en/index.php, Last accessed on 12/15/2017at 2:33pm

[7] HTML &CSS https://www.w3schools.com/, Last accessed on 10/21/2017at 1:33pm.

[8] Bootstraphttp://getbootstrap.com/, last accessed on 09/30/2017at 10:10pm.

[9] https://stackoverflow.com/, last accessed on 11/07/2017 at 2:20am.