Project Report

On

Hospital Management System

Submitted by:

Soham Bapusaheb Waghmare (2020BCS035)(A17)

Chaitanya Rajendra Talware (2020BCS008)(B04)

Aniket Krishna Sontakke (2020BCS024)(B12)

Under the guidance of:

Dr. Jaishri Waghmare

Mrs. Swapnaja Moralwar

Shri Guru Gobind Singhji Institute of Engineering and Technology

Vishnupuri, Nanded (Maharashtra State) INDIA PIN 431606

Government Aided Autonomous Institute DTE Code: 2020

NAAC Accredited institute GRADE B++, CGPA 2.91 (2020 -2025)

Vision Statement: Education of Human Power for Technological Excellence

Department of Computer of Computer Science and Engineering

CERTIFICATE

Certified that this is bonafide record pf project work titled

Hospital management system

Done By:

Soham B. Waghmare (2020BCS035) (A17)

Chaitanya R. Talware (2020BCS008) (B04)

Aniket K. Sontakke (2020BCS024) (B12)

IN PARTIAL FULFILMENT FOR THE REQUIREMENT OF THIRD YEAR, PROJECT,

Mrs. Swapnaja Moralwar
(Project Guide)

Dr. Jaishri Waghmare
(Head of Department)

Acknowledgement

Apart from the efforts of team, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

The completion of any inter-disciplinary project depends upon cooperation, co-ordination, and combined efforts of several sources of knowledge.

We are eternally grateful to our teacher Mrs. Swapnaja Moralwar for her even willingness to give us valuable advice and direction under which we executed this project. Her constant guidance and willingness to share her vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks.

Table of Content

Abstract

Our project Hospital Management system includes registration of patients, storing their details into the system, and booking their appointments with doctors. Our software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. User can search availability of a doctor and the details of a patient using the id. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the database. The data can be retrieved easily. The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast.

It is having mainly two modules. One is at Administration Level and other one is of user I.e., of patients and doctors. The Application maintains authentication to access the application. Administrator task includes managing doctors' information, patient's information. To achieve this, aim a database was designed one for the patient and other for the doctors which the admin can access. The complaints which are given by user will be referred by authorities. The Patient modules include checking appointments, prescription. User can also pay doctor's Fee online.

Problem Statement

In this busy world we don't have the time to wait in infamously long hospital queues. The problem is, queuing at hospital is often managed manually by administrative staff, then take a token there and then wait for our turn then ask for the doctor and the most frustrating thing - we went there by traveling a long distance and then we come to know the doctor is on leave or the doctor can't take appointments.

HMS will help us overcome all these problems because now patients can book their appointments at home, they can check whether the doctor they want to meet is available or not. Doctors can also confirm or decline appointments, this help both patient and the doctor because if the doctor declines' appointment, then patient will know this in advance and patient will visit hospital only when the doctor confirms' the appointment this will save time and money of the patient. Patients can also pay the doctor's consultant fee online to save their time.

HMS is essential for all healthcare establishments, be it hospitals, nursing homes, health clinics, rehabilitation centres, dispensaries, or clinics. The main goal is to computerize all the details regarding the patient and the hospital. The installation of this healthcare software results in improvement in administrative functions and hence better patient care, which is the prime focus of any healthcare unit.

Chapter 1.

Introduction

1.1 PURPOSE

This software will help the company to be more efficient in registration of their patients and manage appointments, records of patients. It enables doctors and admin to view and modify appointments schedules if required. The purpose of this project is to computerize all details regarding patient details and hospital details.

1.2 SCOPE

The system will be used as the application that serves hospitals, clinic, dispensaries, or other health institutions. The intention of the system is to increase the number of patients that can be treated and managed properly.

If the hospital management system is file based, management of the hospital must put much effort on securing the files. They can be easily damaged by fire, insects, and natural disasters. Also, could be misplaced by losing data and information.

1.3 DEFINITIONS, ACRONYMS, and ABBREVIATIONS

1. **Cardiologist** - treats heart disease.
2. **Pediatrician** - treats infants, toddlers, children, and teenagers.
3. **Plastic Surgeon** - restores, reconstructs, corrects, or improves in the shape and appearance of damaged body structures, especially the face.
4. **Psychiatrist** - treats patients with mental and emotional disorders.
5. **Ophthalmologist** - treats eye defects, injuries, and diseases.
6. **ENT**- Ear, Nose and Throat Specialist.
7. **Appt –** Appointment.
8. **Sign up -** Creating New User.
9. **Log in -** Logging in Existing User.
10. **Ph No** - Mobile number.

1.4 OVERVIEW

Our application contains two modules – the admin module and the user module. Our application will not only help the admin to preview the monthly and/or yearly data but it will also allow them to edit, add or update records. The software will also help the admin to monitor the transactions made by the patients and generate confirmations for the same. The admin will be able to manage and update information about doctors.

The user module can be accessed by both the doctors and the patients. The doctor can confirm and/or cancel appointments. The doctors can even add prescriptions for their patients using our application. The patients will be able to apply for the appointment and make transaction for the same and can even cancel appointments with the doctors. They can track details about the previous transactions made by them.

Advantages

- The system automates the manual procedure of managing hospital activities.
- Doctors can view their patients' treatment records and details easily.
- It even generates an instant bill.
- The system is convenient and flexible to be used.
- It saves their time, efforts, money and resources.

Disadvantages

- Requires large database.
- The admin has to manually keep updating the information by entering the details in the system.
- Need Internet connection.

1.5 SOFTWARE TOOLS USED

HTML-

HTML or Hyper Text Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content. HTML tags most come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, further tags, comments, and other types of text-based content. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags but uses the tags to interpret the content of the page.HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

CSS-

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind

of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification.

JAVA SCRIPT-

JavaScript (JS) is a dynamic computer programming language. It is most used as part of web browsers, whose implementations allow client- side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications. JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from

the Self and Scheme programming languages. It is a multi- paradigm language, supporting object-oriented, imperative,

and functional programming styles. The application of JavaScript to use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language, but just-in-time compilation is now performed by recent (post-2012) browsers.

PHP-

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page, it now stands for PHP: Hypertext Pre-processor, a recursive backronym. PHP code is interpreted by a webserver with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly

into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used

in standalone graphical applications. PHP is free software released under the PHP License. PHP can be deployed on most web servers and as a standalone shell on almost every operating system and platform, free of charge.

MYSQL-

MySQL ("My S-Q-L", officially, but also called "My Sequel") is (as of July 2013) the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius daughter, My. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety

of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation

MySQL is a popular choice of database for use in web applications and is a central component of the widely used LAMP open-source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open-source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality.

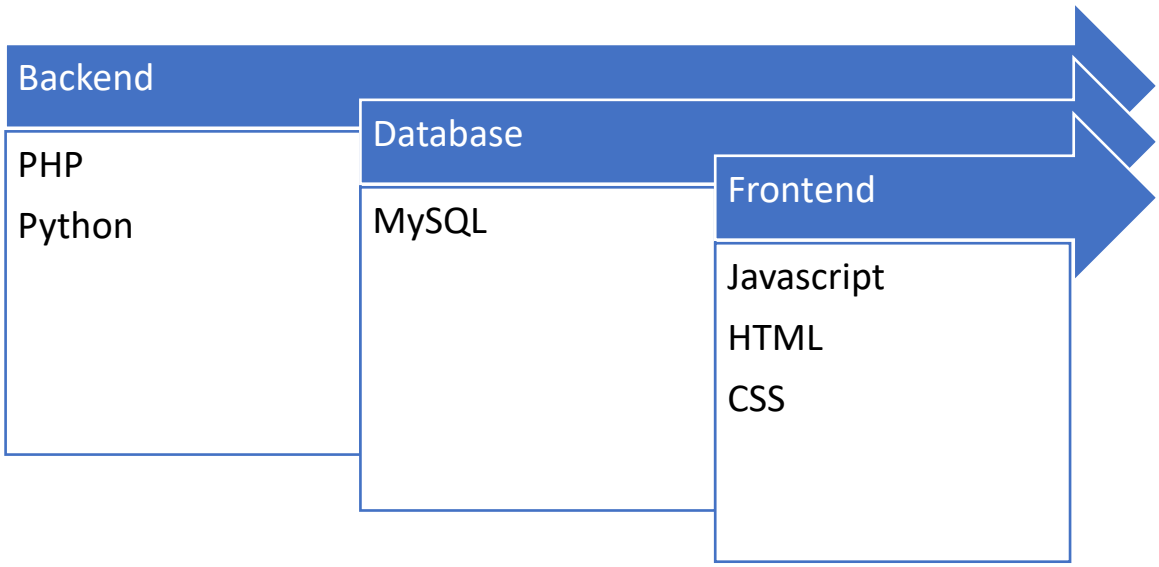
Python-

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

XAMPP-

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

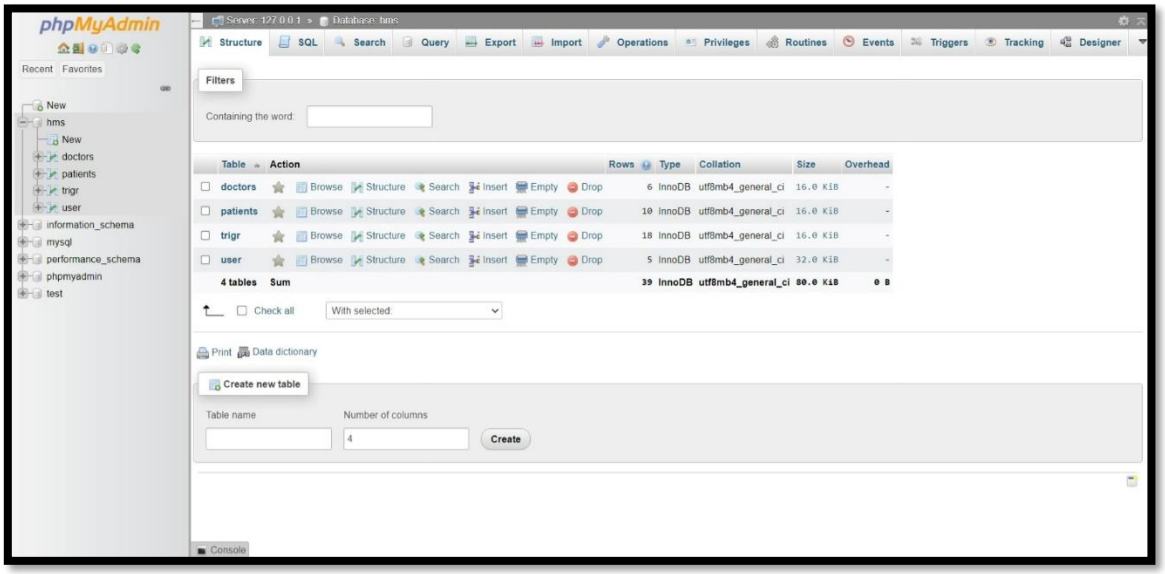
1.6 TECHNOLOGY STACK:



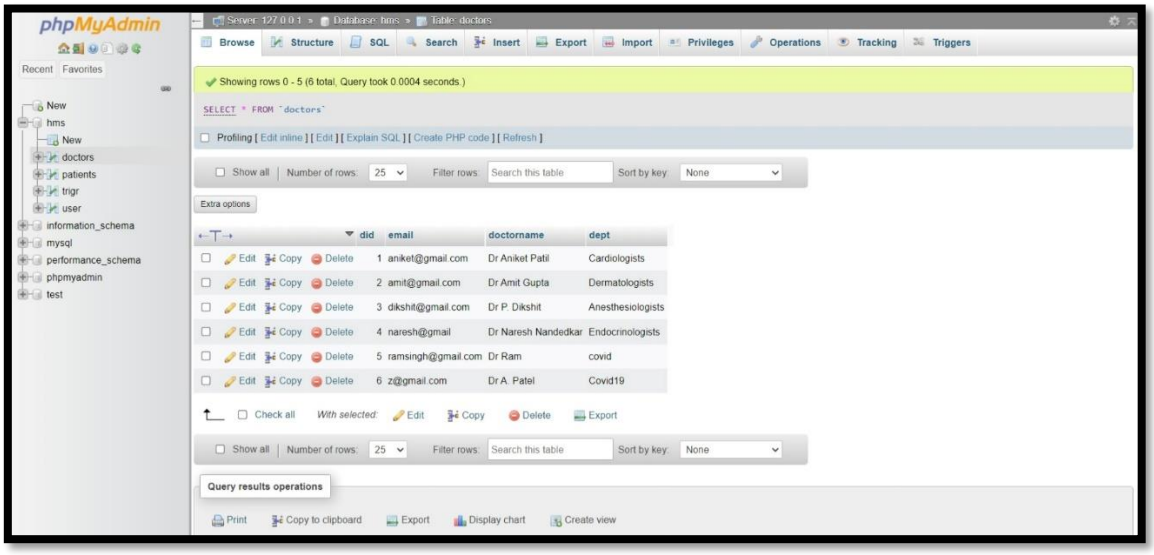
Chapter 2.
System Design

2.1 Table Design

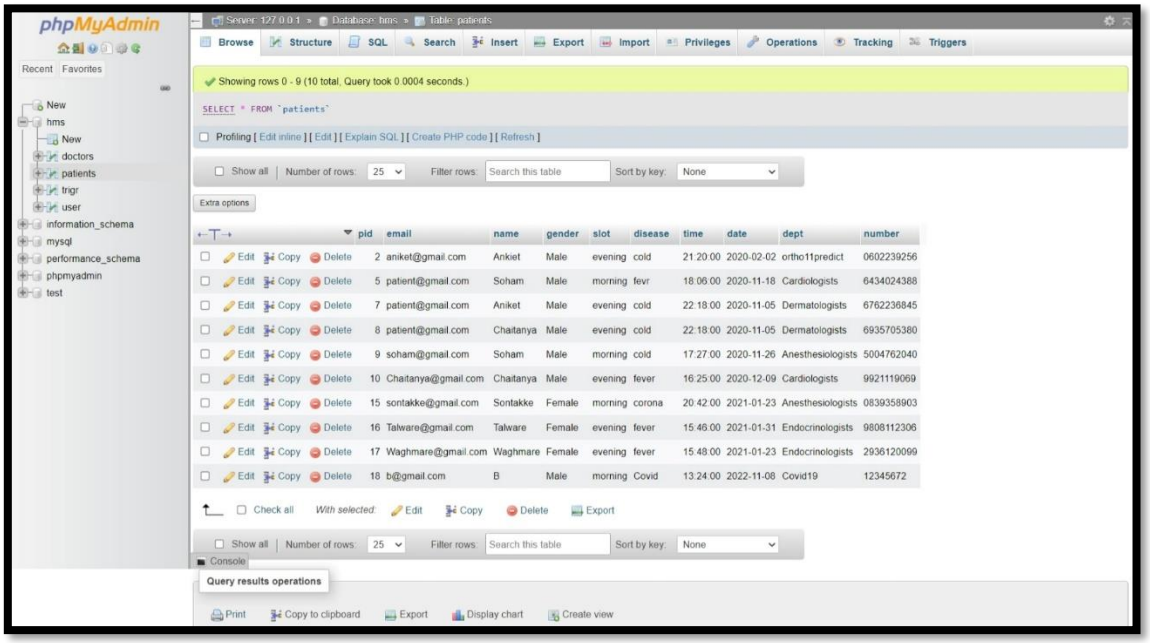
2.1.1 Tables in the database:



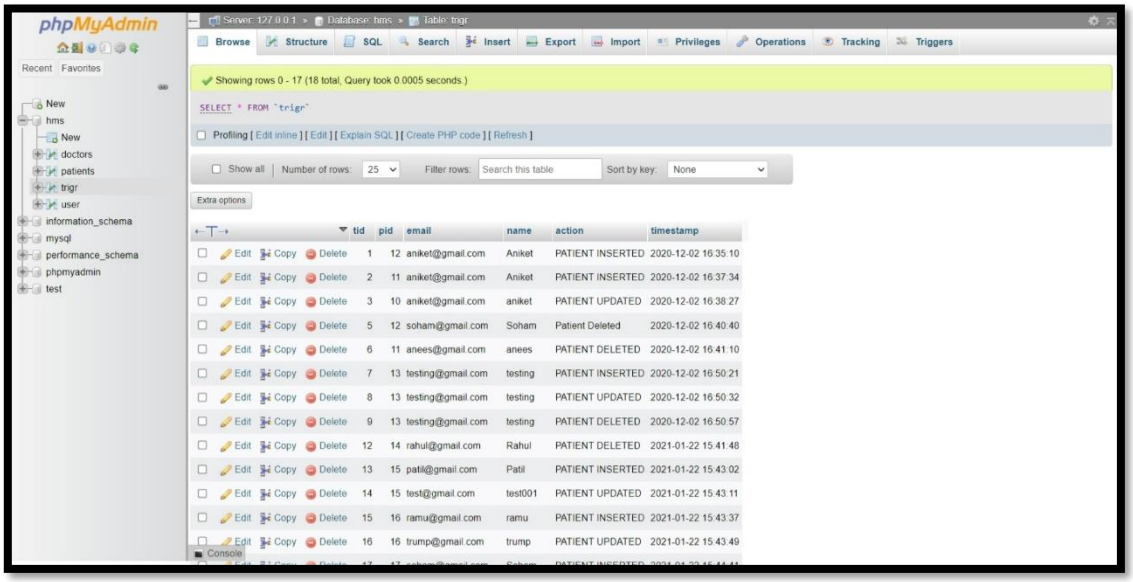
2.1.2 Table of doctors:



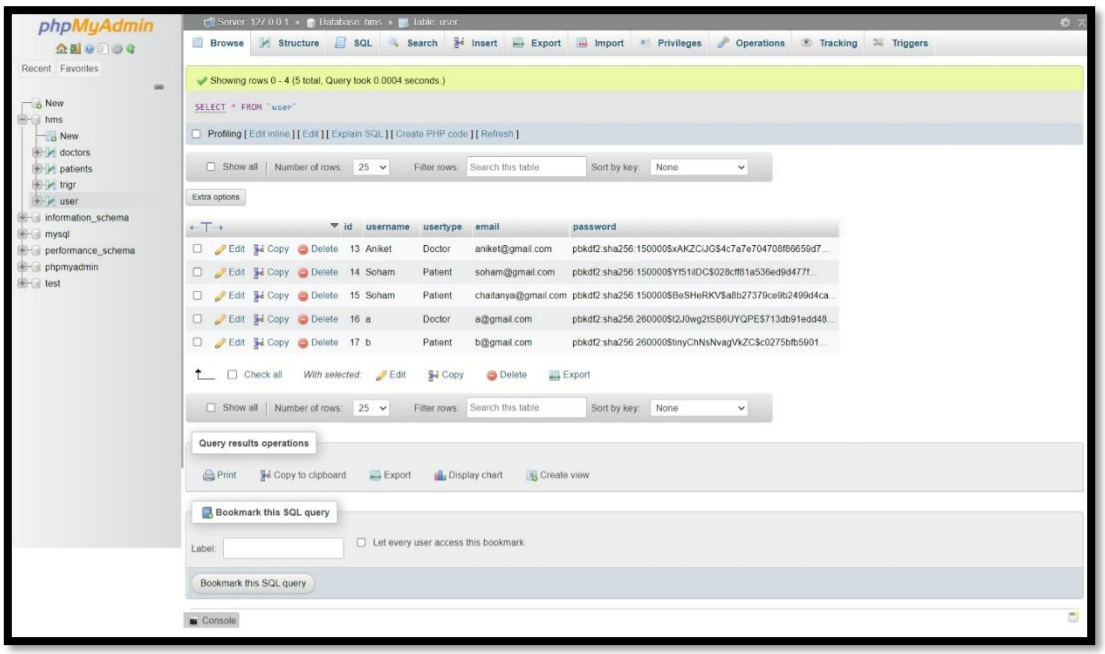
2.1.3 Table of patients:



2.1.4 Table showing Record of all activities:



2.1.5 Table showing users in the system:



2.2 ER Diagram

Chapter 3.

Python Code

```python
from flask import Flask, render_template, request, session, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import login_user, logout_user, login_manager, LoginManager
from flask_login import login_required, current_user
from flask_mail import Mail
import json

with open('config.json', 'r') as c:
    params = json.load(c)["params"]

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


# 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
def load_user(user_id):
    return User.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/hms'
db = SQLAlchemy(app)


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


class User(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))

```

```
class Patients(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))


class Doctors(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))


class Trigr(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('/')
def index():
    a = params['gmail-user']
    return render_template('index.html')


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

    if request.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")

    return render_template('doctor.html')


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

    if request.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}','{t

# mail starts from here

        # mail.send_message(subject, sender=params['gmail-user'], recipients=[email],body=f"YOUR bOOKING IS CONFIRMED THANKS FOR CHOOSING US \nYour Entered Details are :\nName

        flash("Booking Confirmed", "info")

    return render_template('patient.html', doct=doct)


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


@app.route("/edit/<string:pid>", methods=['POST', 'GET'])
@login_required
def edit(pid):
    posts = Patients.query.filter_by(pid=pid).first()
    if request.method == "POST":
        email = request.form.get('email')
        name = request.form.get('name')
        gender = request.form.get('gender')
        slot = request.form.get('slot')
```

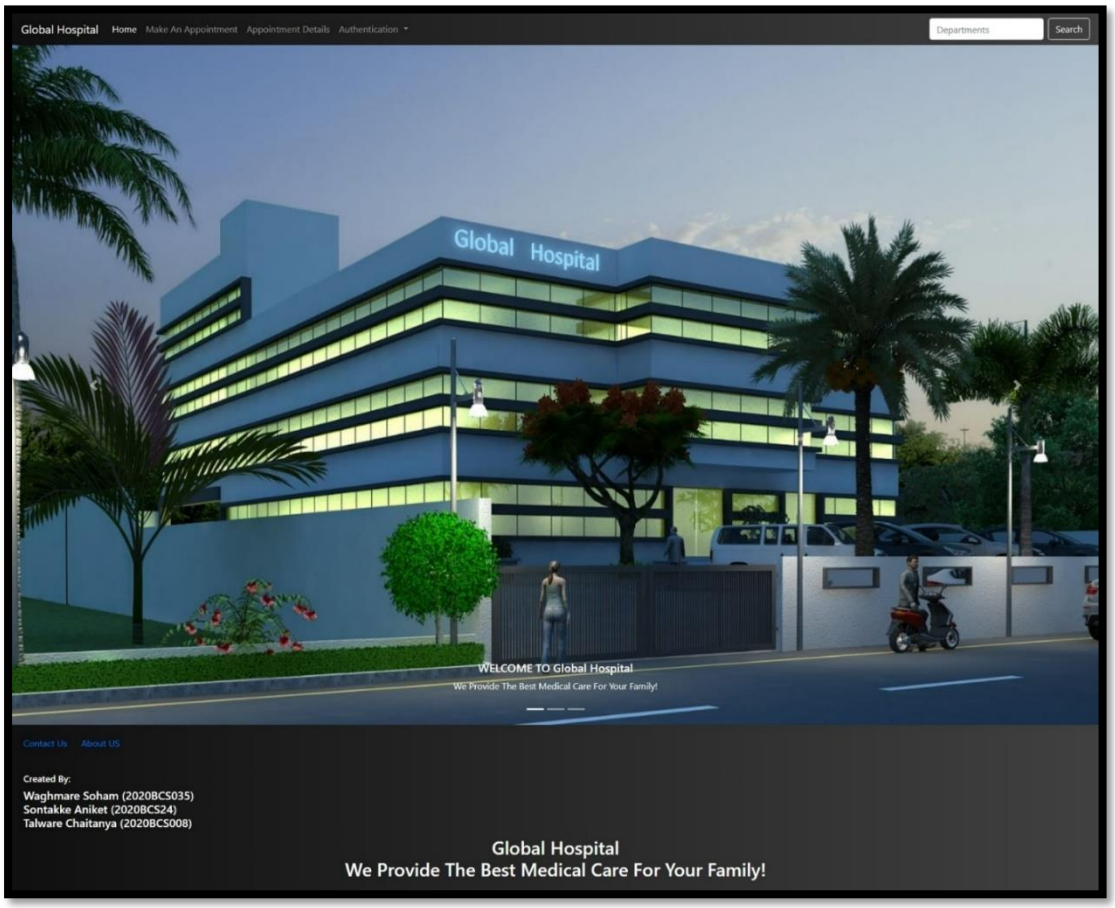
```
        disease = request.form.get('disease')
        time = request.form.get('time')
        date = request.form.get('date')
        dept = request.form.get('dept')
        number = request.form.get('number')
        db.engine.execute(
            f"UPDATE `patients` SET `email` = '{email}', `name`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

    return render_template('edit.html', posts=posts)


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


@app.route('/signup', methods=['POST', 'GET'])
def signup():
    if request.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()
        if user:
            flash("Email Already Exist", "warning")
            return render_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")
        return render_template('login.html')

    return render_template('signup.html')


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

        if user and check_password_hash(user.password, password):
            login_user(user)
            flash("Login Success", "primary")
            return redirect(url_for('index'))
        else:
            flash("invalid credentials", "danger")
            return render_template('login.html')

    return render_template('login.html')


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


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


@app.route('/details')
@login_required
def details():
    # posts=Trigr.query.all()
    posts = db.engine.execute("SELECT * FROM `trigr`")
    return render_template('trigers.html', posts=posts)


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

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

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


app.run(debug=True)
```

Chapter 4.

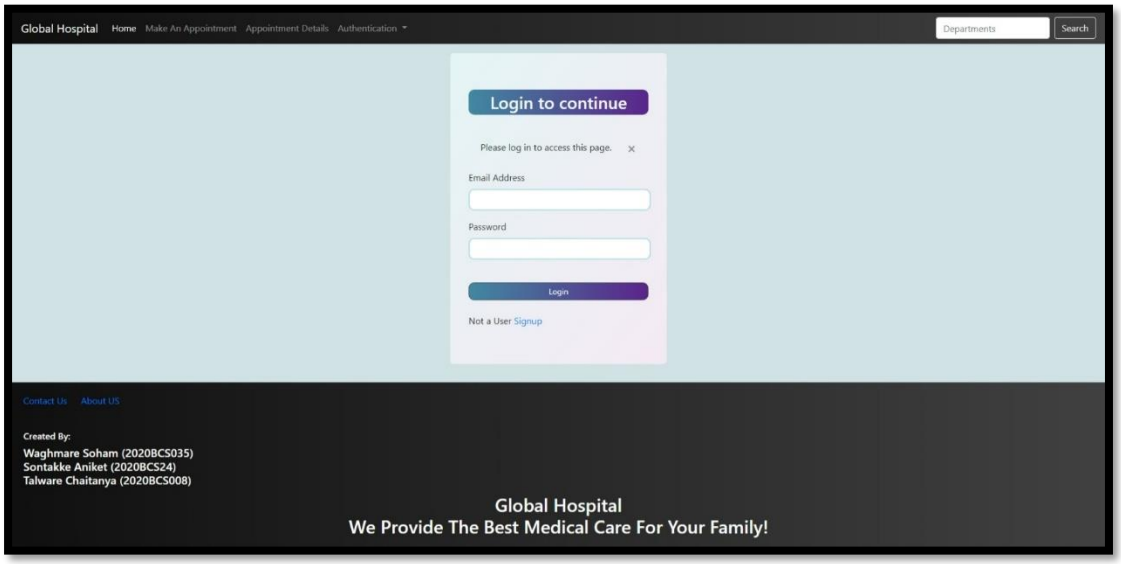
System Implementation

4.1 Screenshots

Home page

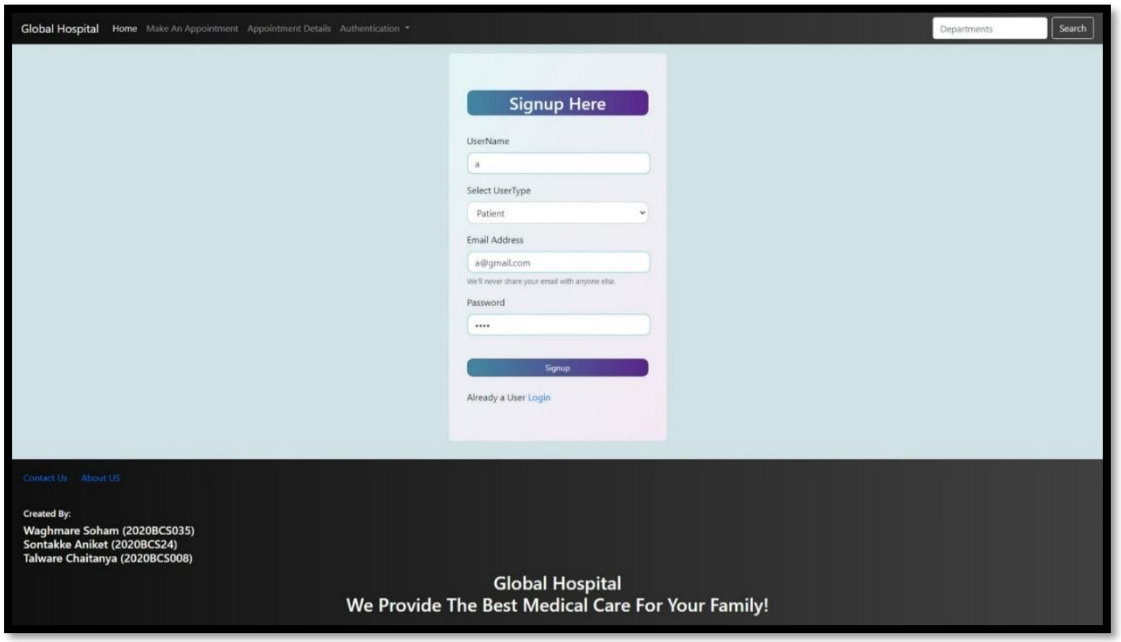


Login page



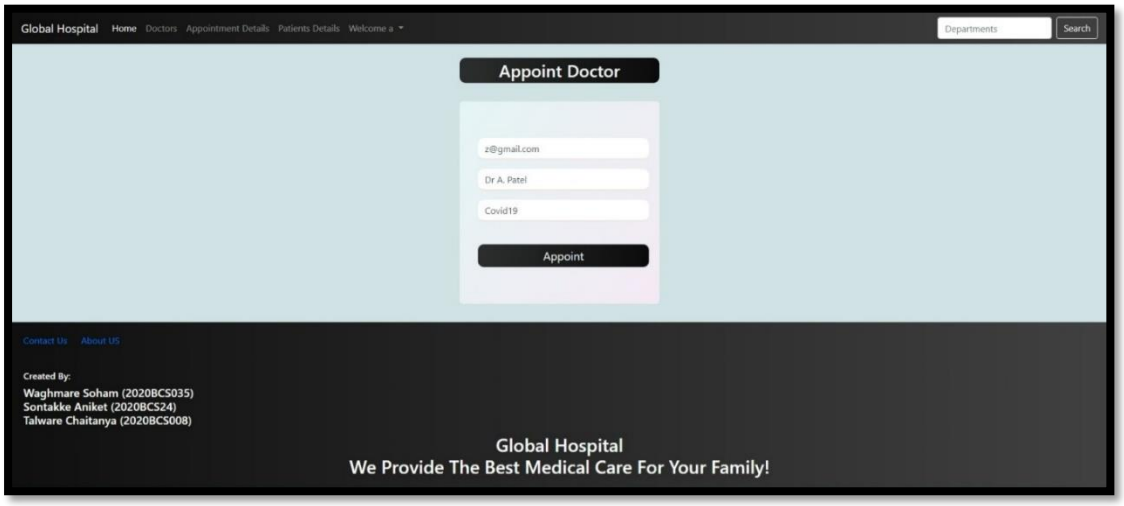
Page for already signup users to login to the system.

Signup page



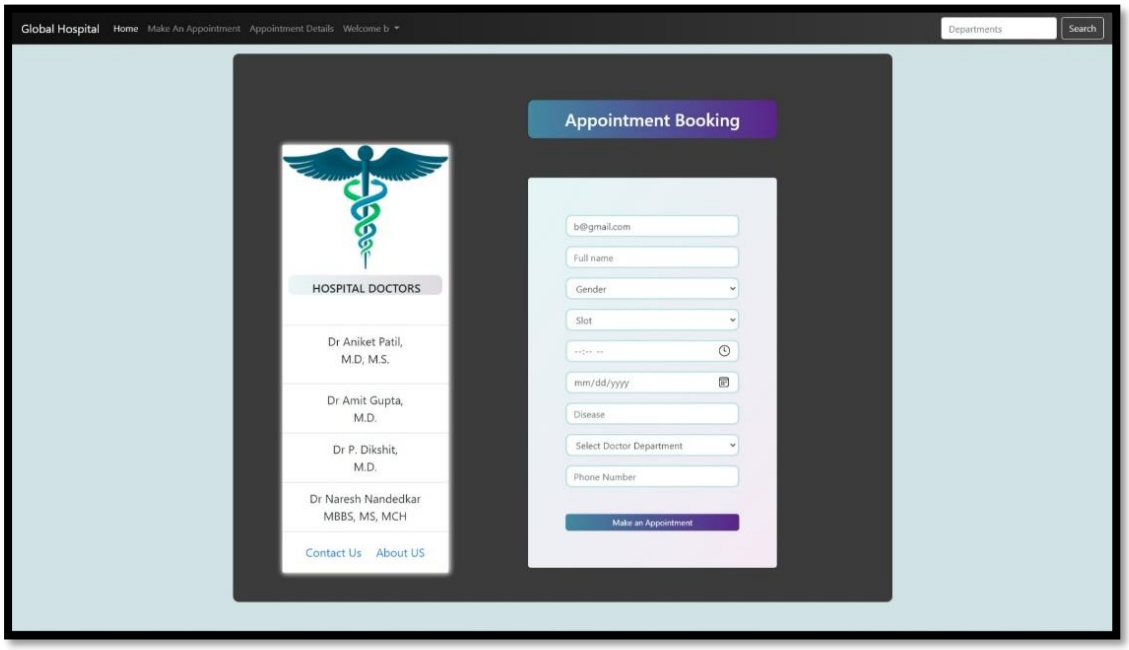
Page for new users to signup. Both patients and Doctors can signup here.

Page for appointing a doctor



More doctors can be appointed here. Can only be done by other doctors.

Page for booking an appointment

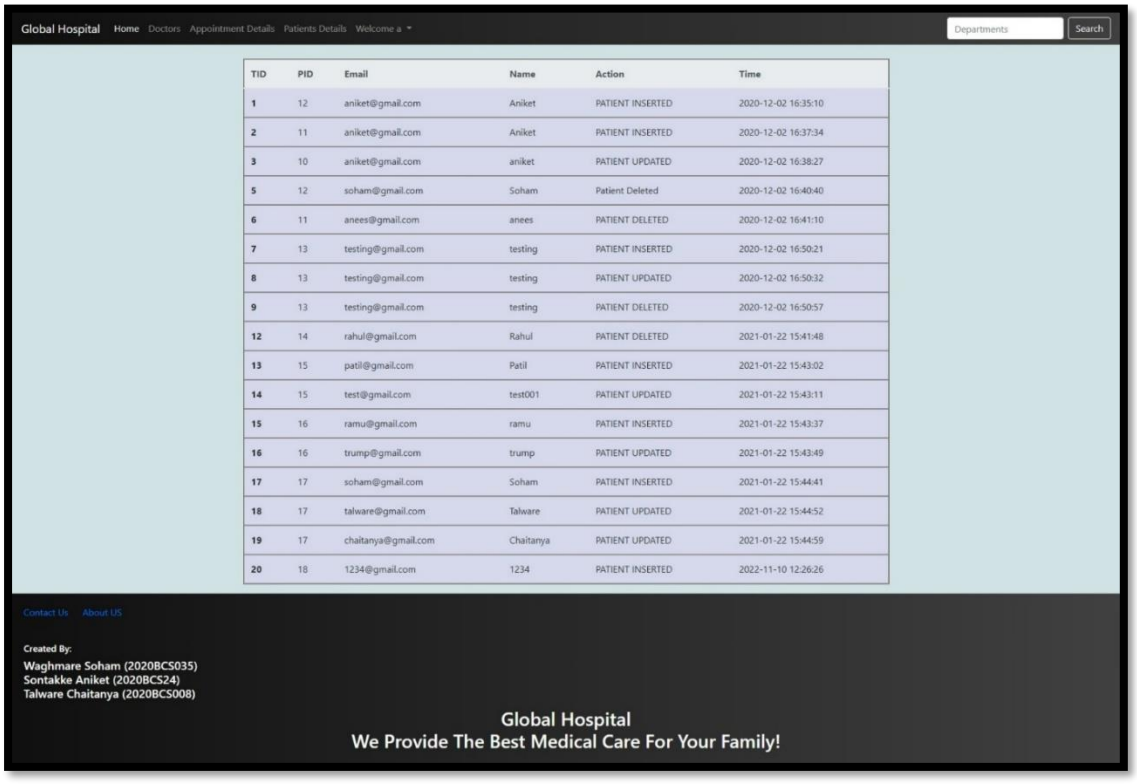


Patients can book an appointment here.

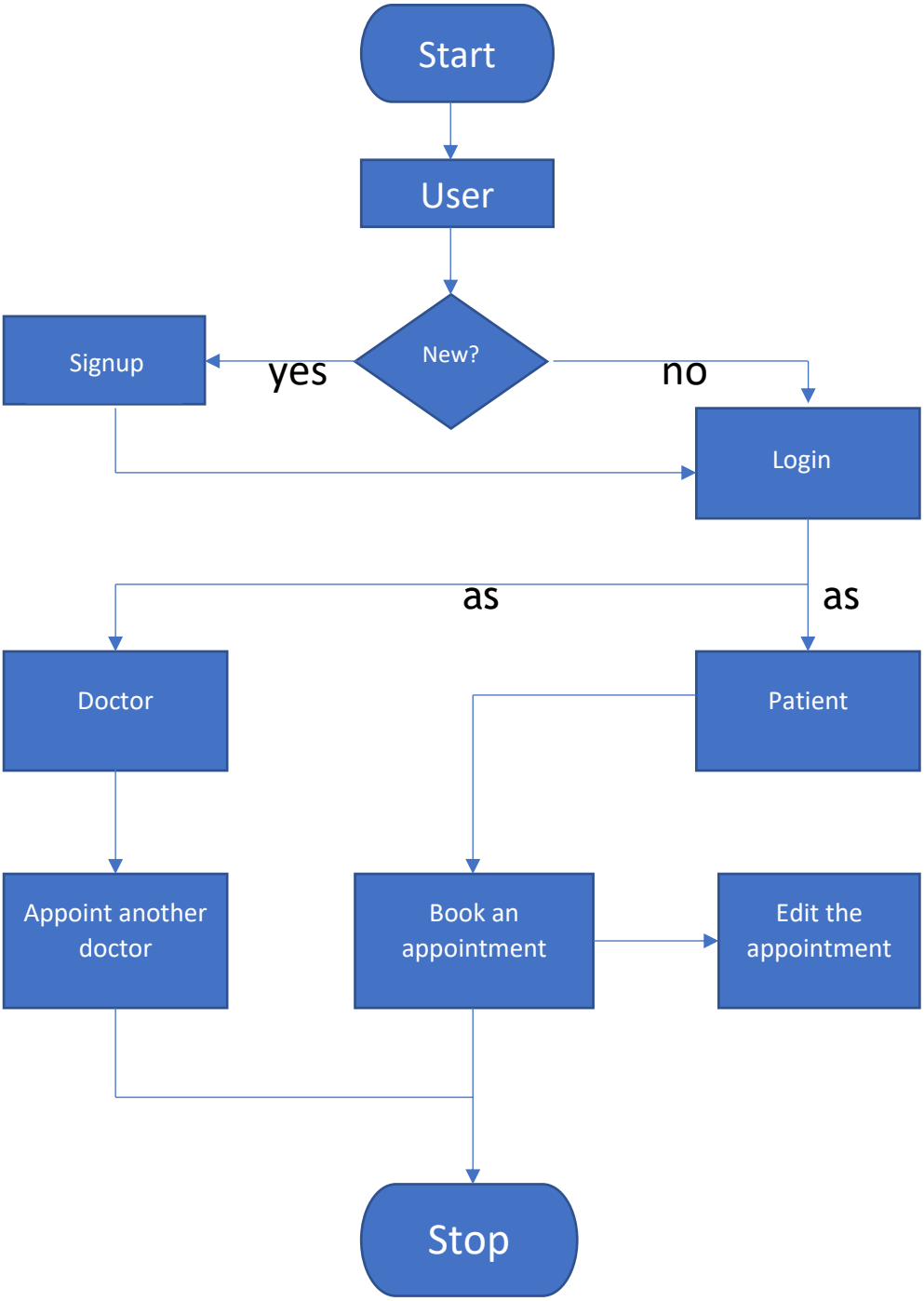
Page showing appointments

Details of all the patient's appointments is showcased here.

Page showing all records of patient's activities



4.2 Flowchart:



Chapter 5.

Conclusion

Working on the project was an excellent experience. It helped us to understand the importance of planning, designing and implementation so far, we have learnt in our theory books. It helped us unleashing our creativity while working in a team. It also realized the importance of team working, communication as a part of this project.

The project was successfully completed after a lot of efforts and work hours. This project underwent number of compiling, debugging, removing errors, making it bug free, adding more facilities in Hospital Management System and interactivity making it more reliable and useful.

This project focused that scheduling a project and adhering to that schedule creates a hard sense of time- management. It has also let us know that co-operative teamwork always produces effective results.

The entire project has been developed and deployed as per the requirements stated by the user. It is found to be bug free as per the testing standards that are implemented.

Chapter 6.

Future Scope

Hospitals and healthcare centres have undergone a change for its betterment. The administrations of healthcare sector are opting IT solutions for the better management and patient care in their hospital campus. Have a look at some salient features of hospital management software.

Daily functions like patient registration, monitoring blood bank, managing admission and overall management of various departments can be easily performed with higher accuracy after the installation of hospital software.

The modules of hospital management software are user-friendly and easy to access. It has a common user-friendly interface having several modules. The officials can utilize these modules in their processes without any hassle and make the best possible use of hospital management system.

Since, every hospital has some or the other points of worth those vary in comparison with to its competitors. Hence, most of the IT companies give on-demand solutions or feature of customization. It further implicates that hospital information management software can be customized by specifying personal requirements of the campus.

The automated functions of online hospital software make productivity effective. This web-based IT solution has automated operations and permit officials to continue with their work in a swift manner. It further implicates that complete automation of the hospital software makes productivity easily obtainable. All in all, this enhances the infrastructure of hospital administration.

This tool is a comprehensive solution that integrates all the departments by creating a common platform. In brief, hospital management system has all the modules that serve purpose of all the departments of healthcare institute. In fact, these modules have been competitively designed to make all the operations simplified.

Chapter 7.

References

https://www.karexpert.com/blogs/what-is-hospital-management-system/

https://mocdoc.in/blog/a-detailed-view-of-hospital-management-system-hms

https://1000projects.org/hospital-database-management-system-project-using-php-mysql.html

GitHub link to the project:

https://github.com/2020BCS008/Hospital-Management-System