HOSPITAL APPOINTMENT SYSTEM

PROJECT CLOSURE REPORT

SUBMITTED BY

Chaitanya Rathod	**PRN ID 220970920007**
Disha Singh	**PRN ID 220970920009**
K. Nehru	**PRN ID 220970920013**
Krishna Mandhane	**PRN ID 220970920015**
Somnath Telang	**PRN ID 220970920041**

Under the guidance of
Dr. Ritu Khosla

In partial fulfilment of the requirements for the award of
POST GRADUATE DIPLOMA IN ADVANCE COMPUTING

C-DAC, IIPC Building, NIT Silchar, Silchar, Assam.

CERTIFICATE

This is to certify that the project entitled

"Hospital Appointment System"

Has been satisfactorily completed by:

Mr. Chaitanya Rathod

Miss. Disha Singh

Mr. K. Nehru

Mr. Krishna Mandhane

Mr. Somnath Telang

Under supervision and guidance for partial fulfilment of the

PG Diploma in Advanced Computing (DAC) Course

Of

Centre for Development of Advanced Computing (C-DAC), Pune. At

C-DAC, IIPC Building, NIT Silchar, Silchar, Assam.

Mr. Alok Dey

Course Co-ordinator Director

Batch: DAC September 2022 Batch

Date: 09th March,2023

ACKNOWLEDGMENTS

We have taken efforts in this project. However, it would not have been possible without the kind Support and help of many individuals. We would like to extend our sincere thanks to all of them. We are highly indebted to our guide **Dr. Ritu Khosla** for her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completing the project.

We express our thanks to our Director and our Course coordinator **Mr. Alok Dey** for extending their support. We would also thank our Institution and the faculty members without whom this project would have been a distant reality. Our thanks and appreciations also go to all people who have willingly helped us out with their abilities.

Submitted by:

Mr. Chaitanya Rathod

Miss. Disha Singh

Mr. K. Nehru

Mr. Krishna Mandhane

Mr. Somnath Telang

Centre for Development of Advanced Computing (C-DAC), Pune. At

C-DAC, IIPC Building, NIT Silchar, Silchar, Assam.

Date: 09th March,2023

ABSTRACT

The purpose of The Hospital Appointment System is to automate the existing manual system by the help of computerized equipment's and full-fledged computer software. Fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

The Hospital Appointment System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

Contents

Sr. No.	Topic	Page No.
1	Introduction	1
2	Background Theory and Literature Review	2
3	Problem statement	4
4	Specific requirements stage	5
5	Methodology	6
6	System flowcharts & DFDs	12
7	ER diagram	13
8	Table structure	14
9	Project Relevancy, Feasibility	15
10	Code	17
11	Result	36
12	Conclusion & recommendation	46
13	References	47

List of Figures and tables

Sr. No.	Figure caption	Page No.
1	System flow chart	12
2	ER diagram	13
3	Data Base Table Structure	14
3	Waterfall model	16
4	Home page	36
5	Patient Registration & Login	37
6	Appointment Booking by Patient	38
7	Admin Registering Doctor Details	39
8	Admin Registering Receptionist Details	39
9	Appointment Booking by Receptionist	40
10	Doctor Approving the Appointments	40
11	Patient can check Appointment status	41
12	Admin can Update/Delete Doctor, Receptionist profile	41
13	Database	42
14	Email Confirmation	43

Chapter 1
INTRODUCTION

The Hospital Appointment System has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly. Hospital Appointment System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and managing the information of Doctor, Booking, Doctor Schedule, Patient. Every Hospital Appointment System has different Hospital needs, therefore we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy user who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times.

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 make hospital appointment booking managing procedure digitalized.

1.2 Scope

The intention of the system is to increase the number of patients that can be treated. If the hospital appointment system is file based, management of the hospital has to 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.

Chapter 2 BACKGROUND THEORY

2.1 Background and Motivation

Hospital appointment scheduling, as the starting point of most non-urgent health care services, is undergoing major developments to support active involvement of patients. By using the Internet as a medium, patients are given more freedom in decision making about their preferences for the appointments and have improved access frequently.
Hospital appointments have been made with schedulers over the telephone or in person. These methods are based on verbal communications with Receptionist at hospital and allow for maximum flexibility in Scheduling the Appointments . However, because these traditional methods require the intervention of staffs, the ability to get a timely appointment is not only limited by the availability of appointment slots, but also by the busy staffs . Patients' satisfaction with appointment booking is influenced by their ability to book at the right time with the right specialists .

The Internet has recently emerged as another means to make appointments. Web-based appointment scheduling has been a popular research topic. Several studies conducted satisfaction surveys and found that Web-based appointment scheduling is an extremely important feature, and most patients would use the service again.

2.2 Literature Review

The development of ambulatory surgery and same-day admit surgery has prompted the development of paranesthesia evaluation clinics. Outpatient clinic scheduling was studied extensively between 1950 and 1980 (1–6). This research is the foundation for routine practices in clinic scheduling. Some hospitals may have lost awareness as to why clinic scheduling is performed in the manner that it is (7). In this article, I review the science of clinic scheduling appropriate to appointment systems for anesthesia clinics.

Appointment scheduling systems are used by Hospitals & Clinics to manage access to service providers. Many factors affect the performance of appointment systems including arrival and service time variability, patient and provider preferences, available information technology and the experience level of the scheduling staff. The most common types of health care delivery systems are described in this article with particular attention on the factors that make appointment scheduling challenging. For each environment relevant decisions ranging from a set of rules that guide schedulers to real-time responses to deviations from plans are described. A road map of the art in the design of appointment management systems is provided and future opportunities for novel applications are identified.

Outpatient appointment scheduling problems have recently gained increasing attention. This paper provides a comprehensive review of recent analytical and numerical optimization studies that present decision-support tools for designing and planning outpatient appointment systems (OASs). A structure for organizing the recent literature according to various criteria is provided, including a framework that classifies decisions at the strategic, tactical, and operational levels.

The OAS literature is evaluated from four perspectives: problem settings, environmental factors, modeling approaches, and solution methods. In addition, research gaps and areas of opportunity for future research are discussed.

This paper provides a comprehensive review of Appointment Scheduling (AS) in healthcare service while we propose appointment scheduling problems and various applications and solution approaches in healthcare systems. For this purpose, more than 150 scientific papers are critically reviewed. The literature and the articles are categorized based on several problem specifications, i.e., the flow of patients, patient preferences, and random arrival time and service. Several methods have been proposed to shorten the patient waiting time resulting in the shortest idle times in healthcare centers. Among existing modeling such as simulation models, mathematical optimization techniques, Markov chain, and artificial intelligence are the most practical approaches to optimizing or improving patient satisfaction in healthcare centers. In this study, various criteria are selected for structuring the recent literature dealing with outpatient scheduling problems at the strategic, tactical, or operational levels. Based on the review papers, some new overviews, problem settings, and hybrid modeling approaches are highlighted.

Chapter 3 PROBLEM STATEMENT

3.1 Problem Statement

In this hectic world, we don’t have time to wait in infamously long hospital queues. The problem is, queuing at hospital is often managed manually by administrative staff, we take a token, wait for our turn, then we are directed to meet doctor. What’s more exasperate is that we travelled a long distance only to learn the doctor isn’t available.

Chapter 4 Specific Requirement Stages

Requirement gathering stage

During this phase, detailed requirements of the software system to be developed are gathered fromclient.

4.1 Design stage

Plan the programming language, for Example Java, JSP. Or other high-level technical details of the project.

4.2 Built stage

After design stage, it is built stage that is nothing but coding the software.

4.3 Test stage

In this phase, you test the software to verify that it is built as per the specifications given by the client.

4.4 Deployment stage

Deploy the application in the respective environment

4.5 Maintenance stage

Once your system is ready to use, you may later require change the code as per customer request.

4.6 Tools

The tools required to develop the system are: Eclipse IDE, Apache tomcat server & MySQL Workbench ,VS Code, NodeJS .

Chapter 5 Methodology

Main methodology activities held during the research is acquiring information and knowledge about The Hospital Appointment System through reading books, and researches that were previously done in related area. All the research materials were obtained over the internet, Wikipedia and other websites. Next step taken is reading, comprehending and analyzing literature review and matching information obtained. This research emphasize The Hospital Appointment System, which include usability, user-friendly interface, reliability, costing and meeting needs of target users.

5.1 Tools and API

5.1.1 J2EE

The Enterprise Java Blue Prints for the J2EE platform describe the J2EE application model and best practices for using the J2EE platform. Building on the J2SE platform, the J2EE application model provides a simplified approach to developing highly scalable and highly available internet or intranet based applications.

Another advantage of the J2EE platform is that the application model encapsulates the layers of functionality in specific types of components. Business logic is encapsulated in Enterprise JavaBeans (EJB) components. Client interaction can be presented through plain HTML web pages, through web pages powered by applets, Java Servlets, or Java Server Pages technology, or through stand-alone Java applications. Components communicate transparently using various standards: HTML, XML, HTTP, SSL, RMI, IIOP, and others.

5.1.2 JPA

The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems. As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. So, ORM tools like Hibernate, TopLink and iBatis implements JPA specifications for data persistence.

5.1.3 SpringBoot

Spring Boot is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework.Our Spring Boot Tutorial includes all topics of Spring Boot such, as features, project, maven project, starter project wizard, Spring Initializr, CLI, applications, annotations, dependency management, properties, starters,
Actuator, JPA, JDBC, etc.
Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.

It is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring Framework.

It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.

5.1.4 Maven

Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. The tool provides allows developers to build and document the lifecycle framework.
Maven is written in Java and is used to build projects written in C#, Scala, Ruby, etc. Based on the Project Object Model (POM), this tool has made the lives of Java developers easier while developing reports, checks build and testing automation setups.

5.1.5 Postman API

The Postman API client is the foundational tool of Postman, and it enables you to easily explore, debug, and test your APIs while also enabling you to define complex API requests for HTTP, REST, SOAP, Graph QL, and Web Sockets.

5.1.6 Eclipse IDE

In the context of computing, Eclipse is an integrated development environment (IDE) for developingapplications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc. The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plug-ins and is designed to be extensible using additional plug-ins.
Developed using Java, the Eclipse platform can be used to develop rich client applications, integrateddevelopment environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available. The Java Development Tools (JDT) project provides a plug-in that allows Eclipse to be used as a Java IDE, C/C++ Development Tools (CDT) is a plug-i.

5.1.7 MYSQL

MySQL Workbench is a graphical tool for working with MySQL servers and databases. MySQL Workbench fully supports MySQL server version 5.7 and higher. Deprecated versions of MySQL Server (prior to version 5.7) are incompatible with MySQL Workbench and should be upgraded before you attempt to make a connection. MySQL is an open source RDBMS that relies on SQL for processing the data in the database. MySQL provides APIs for the languages C, C++, Eiffel, Java, Perl, PHP and Python. In addition, OLE DB and ODBC providers exist for MySQL data connection in the Microsoft environment. A MySQL .NET Native Provider is also available, which allows native MySQL to .NET access without the need for OLE DB. MySQL is most commonly used for Web applications and for embedded applications and has become a popular alternative to proprietary database systems because of its speed and reliability. MySQL can run on UNIX, Windows and Mac OS. MySQL is developed, supported and marketed by MySQL AB. The database is available for free under the terms of the GNU General Public License (GPL) or for a fee to those who do not wish to be bound by the terms of the GPL.

5.1.8 Node.js Server

Node. js (Node) is an open source, cross-platform runtime environment for executing JavaScript code. Node is used extensively for server-side programming, making it possible for developers to use JavaScript for client-side and server-side code without needing to learn an additional language.

5.1.9 Tomcat Server

It is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages and last but not least, the Java Servlet. The complete name of Tomcat is "Apache Tomcat" it was developed in an open, participatory environment and released in 1998 for the very first time. It began as the reference implementation for the very first Java-Server Pages and the Java Servlet API. However, it no longer works as the reference implementation for both of these technologies, but it is considered as the first choice among the users even after that. It is still one of the most widely used java-sever due to several capabilities such as good extensibility, proven core engine, and well-test and durable. Here we used the term "servlet" many times, so what is java servlet; it is a kind of software that enables the webserver to handle the dynamic(java-based) content using the Http protocols.

5.1.10 React.js

ReactJS tutorial provides basic and advanced concepts of ReactJS. Currently, ReactJS onof the most popular JavaScript front-end libraries which has a strong foundation and a large community.ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

5.1.11 VS Code IDE

VS code is short for Visual Studio Code. It is a strong code editor that is open-source developed via Microsoft. Visual Studio Code contains built-in support for debugging, snippets, code refactoring, integrated terminal, code completion, syntax highlighting, and embedded Git control. Vscode is cross-platform and available on macOS, Linux, and Windows.

5.2.1 System Requirements

The various hardware and software configurations and requirements have been mentioned in detail in this section to provide an effective insight into the development for this project.

5.3.1 External interfaces

System interfaces

The application runs in the latest version of Chrome or Firefox browser on Windows, Linux and Mac.

User interfaces

The web page shall provide a very intuitive and simple interface to the patients and other users, so that one can easily navigate through pages, user's registration, login process whereas client can easily manage and revoke other user's permissions.

Graphical Interface

That enables a person to communicate with a computer through the use of symbols, visual metaphors, and pointing devices.

Menu-driven Interface

Our web application provides with a range of options in the form of a list or menu dis- played in Fullscreen, pop-up, pull-down, or drop-down.

Form-based Interface

Used to get data from users and save into the database .

5.3.2 Hardware interfaces

Server side

The web application will be hosted on a web server which is listening on the web stan- dard port ****.

Client side

Monitor screen – the website shall display information to the user via the monitor screen Mouse – the website shall interact with the movement of the mouse and the mouse but- tons. The mouse shall activate areas for data input, command buttons and select options from menus. Keyboard – the website shall interact with the keystrokes of the keyboard. The keyboard will input data into the active area of the database.

5.3.3 Software interfaces

Server side

An Apache web server will accept all requests from the client and forward it accordingly. A database will be hosted centrally using MySQL.

Client side

An OS which is capable of running a modern web browser which supports JavaScript and HTML5. The Web page is downloaded from the Web server and the user can interact with this content in a Web browser, which acts as a client.

Web- Based Interface

Web browser:

Our website works on most popular web browsers i.e. Google Chrome, Microsoft Edge (formerly Internet Explorer), Mozilla Firefox, and Apple's Safari and works ok with all versions or just on a new one.

Communication standards and Network server communications protocols used:

The HTPP or HTTPS protocol(s) will be used to facilitate communication between the client and server.

Electronic forms:

HTML Forms for user registration and also to get data from patient requesting Appoint- ment booking for different types of Hospital.

Hardware Requirements

Hardware requirements is as follows:

Ram : 8GB

Hard Drive : 1TB

Processor : i3 above

Software Requirements

Software requirements is as follows :

Operating Systems : Windows 7 or Windows 10 or Ubuntu 16.

Technology : J2EE , React.js

IDE : Eclipse/VS code

Chapter 6 System Flowchart

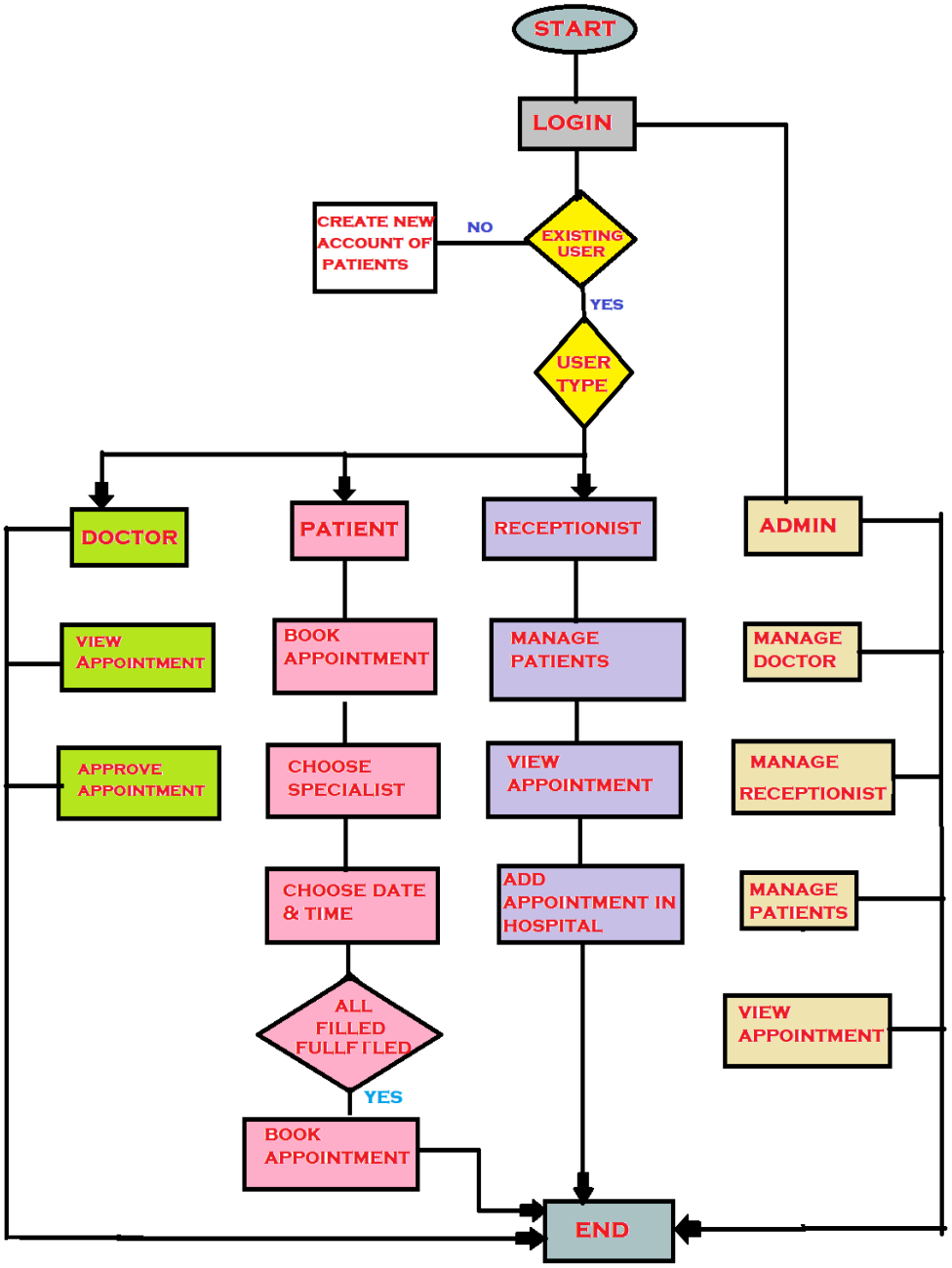

Figure 6.1: Control Flow

Chapter 7 Entity Relationship Diagram (ER-Diagram)

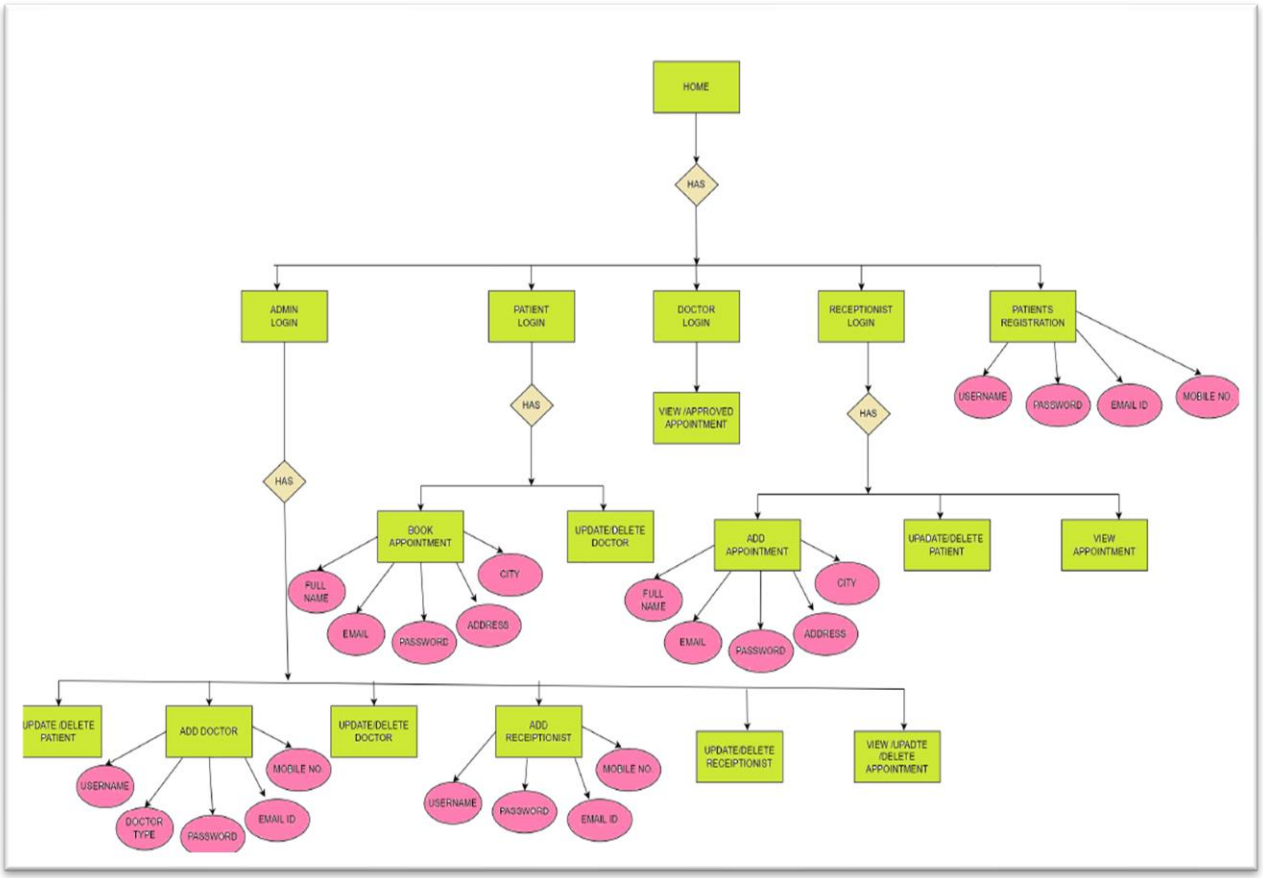

Figure 7.1: ER-Diagram

Chapter 8 Table Structure

The database is used for the purpose of handling information as an integrated whole. It is defined asa collection of interrelated data stored with less or no redundancy to serve many users quickly and effectively. We should design a database to see how data should be organized around user requirements. The objective of the database is to make information access, easy quick, inexpensive and flexible for other users. During database design the following objectives are concerned:

The tables used in the database are as follow:

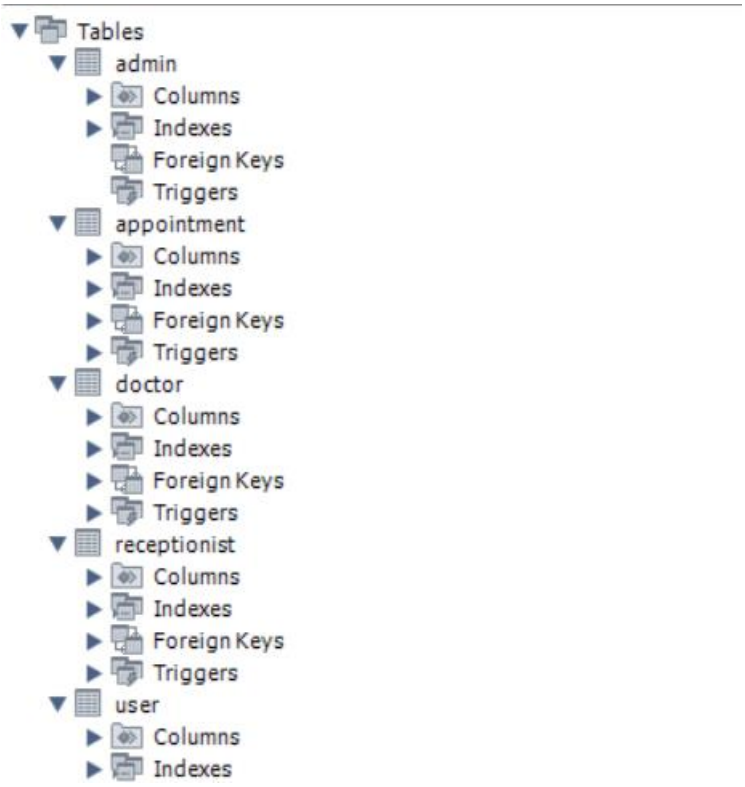

Figure 8.1: Table

Chapter 9 Project Relevancy, Feasibility

Technical Feasibility

Building this system is technically feasible. The hardware and software needed are all available, it not difficult to get them. Brief I can say the necessary resources needed for the development and maintenance of the system are available. I am using Java programming languages and mysql database.

Operationally Feasibility

The project I am developed is operationally feasible as there is no need for users to have good knowledge in computer before using it. The user can learn and use the system with easiness; he justneeds to read the manual or tutorial from the developers.

Economic Feasibility

Besides being technically feasible, developing this system is economically feasible as well. The development of the system does not require the developers to spend a lot of money. The tools I willbe using to develop the system are not expensive and the software's are open source. All I need is time. Even the maintenance of the system will not be expensive. The system is indeed economically feasible.

PROJECT ACTIVITIES

In order to give solution to problems in an industry, software developer or a team of developers must incorporate a development strategy that encompasses the process, methods and tools layers and generic phases. This strategy is often referred to as process model or a software developing paradigm. A process model for software developing is chosen based on the nature of project and application, the methods and tools to be used, and the controls and deliverables that are required. All software development can be characterized as a problem solving loop in which distinct stages are encountered. Regardless of the process model that is chosen for a software project, all of the stages coexist simultaneously at some level of detail. The methodology chosen to develop this system is waterfall model approach. I opted for this method because I found that it is the best for my project where the stages involved can assist my level of progress. Many developers prefer waterfall model and widely use it as a development strategy. 11 Waterfall model approach is chosen because the approach allows the development of the system to be revised after the stages is finished. Once the stages are not satisfied, then going back to the previous stages can be considered necessary to add or modify any features. The different stages for this model:

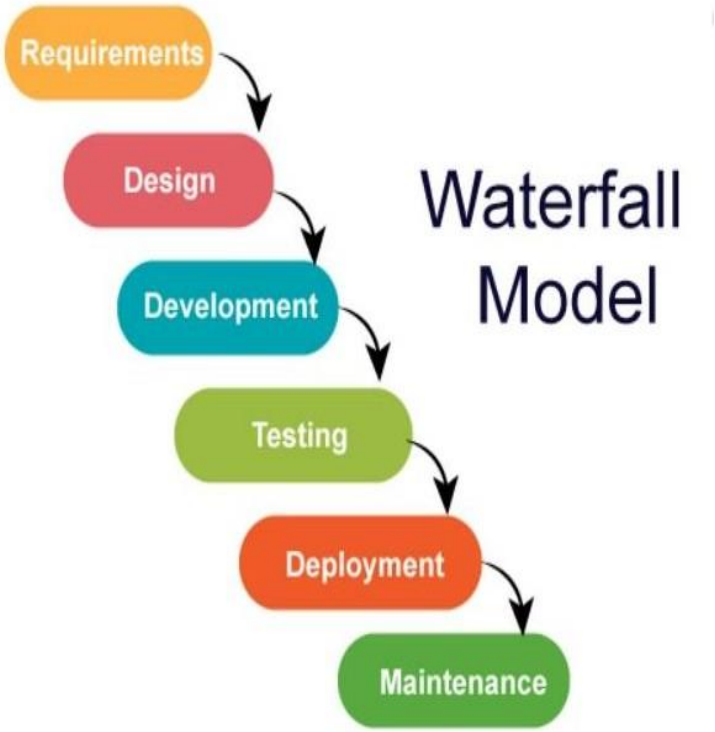

Figure 9.1 Waterfall Model

Chapter 10 CODE

Github repository link :

https://github.com/nehru1919/Hospital_Appointment_System

Frontend Code

Navbar.js

```
import React from 'react'
import { Link } from 'react-router-dom'
import logo from '../images/logo.png'

export default function Navbar() {
    return (
        <div style={{ marginBottom: "70px" }}>
            <nav className="navbar navbar-expand-lg shadow-lg p-3 mb-5 bg-white rounded">
            <a className="navbar-brand" href="#">
                <img src={logo} width="35" height="35" className="d-inline-block align-top"
alt="Logo"/>
                Medicare
            </a>
                <button className="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                    <span className="navbar-toggler-icon"></span>
                </button>
                <div className="collapse navbar-collapse" id="navbarNav">
                    <ul className="navbar-nav ml-auto">
                        <li className="nav-item">
                            <Link className="nav-link" to="/">Home</Link>
                        </li>
                        <li className="nav-item">
                            <Link className="nav-link" to="/login">Login</Link>
                        </li>
                        <li className="nav-item">
                            <Link className="nav-link" to="/register">Register</Link>
                        </li>

                    </ul>
                </div>
            </nav>
        </div>
    )
}
```

Registration.js

```
import React, { Component } from 'react'
import axios from "axios";

import { Link } from 'react-router-dom'
import Dropdown from 'react-bootstrap/Dropdown';
import medicare from '../images/medicare.jpg'

export class Register extends Component {

  constructor(props) {
    super(props);
    this.state = {
      users: [],
      username: '',
      email: '',
      mobile: '',
      password: ''
    }
  }
  componentDidMount() {
    axios.get("http://localhost:8102/api/")
      .then((res) => {
        this.setState({
          users: res.data,
          username: '',
          email: '',
          mobile: '',
          password: ''
        })
      })
  }
  submit(event, name) {
    event.preventDefault();
    axios.post("http://localhost:8102/api/", {
      username: this.state.username,
      email: this.state.email,
      mobile: this.state.mobile,
      password: this.state.password
    })
      .then((res) => {
        console.log(res);
        console.log(res.data);
```

```
        if (res.data == 0) {

          alert("Successufully register")
          window.location = '/login';



// this.props.history.push("/main")
        }
        else {
          alert("Credential Invalid")
        }

        this.componentDidMount();
      })

  }
render() {
    return (
      <div style={{ marginBottom: "70px" }}>

        <nav className="navbar navbar-expand-lg shadow-lg p-3 mb-5 bg-white rounded">
          <a className="navbar-brand" href="#">
            <img src={medicare} width="35" height="35" className="d-inline-block align-top"
alt="medicare" />
            Medicare
          </a>
          <button className="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span className="navbar-toggler-icon"></span>
          </button>
          <div className="collapse navbar-collapse" id="navbarNav">
            <ul className="navbar-nav ml-auto">
              <li className="nav-item">
                <Link className="nav-link" to="/">Home</Link>
              </li>
              <li className="nav-item">
                <Dropdown className="nav-item">
                  <Dropdown.Toggle id="dropdown-basic">
                    Login as
                  </Dropdown.Toggle>

                  <Dropdown.Menu>
                    <Link className="nav-link" to="/login">Patient</Link>
                    <Link className="nav-link" to="/adminlogin">Admin</Link>
                    <Link className="nav-link" to="/doctorlogin">Doctor</Link>
                    <Link className="nav-link" to="/receptionistlogin">Receptionist</Link>
                  </Dropdown.Menu>
                </Dropdown>
```

```
            </li>
            <li className="nav-item">
              <Dropdown className="nav-item">
                <Dropdown.Toggle id="dropdown-basic">
                  Register as
                </Dropdown.Toggle>

                <Dropdown.Menu>
                  <Link className="nav-link" to="/register">Patient</Link>


</Dropdown.Menu>
              </Dropdown>
            </li>

          </ul>
        </div>
      </nav>

      <h1 class="text-center">Patient Registration</h1>

      <form onSubmit={(e) => this.submit(e, this.state.username)} method="POST">
        <div class="form-group mt-3">
          <label for="exampleInputEmail1">Username</label>
          <input onChange={(e) => this.setState({ username: e.target.value })}
value={this.state.username} type="text" class="form-control mt-2" id="exampleInputEmail1"
aria-describedby="emailHelp" placeholder="Enter Username" minLength="2" required />
        </div>
        <div class="form-group mt-3">
          <label for="exampleInputEmail1">Email address</label>
          <input onChange={(e) => this.setState({ email: e.target.value })}
value={this.state.email} type="email" class="form-control mt-2" id="exampleInputEmail1"
aria-describedby="emailHelp" placeholder="Enter email" required />
        </div>
        <div class="form-group mt-3">
          <label for="exampleInputEmail1">Mobile number</label>
          <input onChange={(e) => this.setState({ mobile: e.target.value })}
value={this.state.mobile} type="text" class="form-control mt-2" id="exampleInputEmail1" aria-
describedby="emailHelp" placeholder="Enter Mobile number" maxLength="10"
minLength="10" required />
        </div>
        <div class="form-group mt-3">
          <label for="exampleInputPassword1">Password</label>
          <input onChange={(e) => this.setState({ password: e.target.value })}
value={this.state.password} type="password" class="form-control mt-2"
id="exampleInputPassword1" placeholder="Password"  minLength="5" required />
        </div>
        <div class="form-group mt-3">
          <button type="submit" name="action" class="btn btn-primary">Submit</button>
        </div>
```

```
        </form>
      </div>
    );
  }
}

export default Register
```

Login.js

```
import React, { Component } from 'react'
import axios from "axios";
import { Link } from 'react-router-dom'
import medicare from '../images/medicare.jpg'
import Dropdown from 'react-bootstrap/Dropdown';

export class AdminLogin extends Component {

  constructor(props) {
    super(props);
    this.state = {

users: [],
      username: '',
      password: ''
    }
  }
  componentDidMount() {
    axios.get("http://localhost:8102/adminlogin/")
      .then((res) => {
        this.setState({
          users: res.data,
          username: '',
          password: ''
        })
      })
  }
  submit(event, name) {
    event.preventDefault();
    axios.post("http://localhost:8102/adminlogin/", {
      username: this.state.username,
      password: this.state.password
    })
      .then((res) => {

        console.log(res);
        console.log(res.data);
        if (res.data == 0) {
```

```
        this.props.history.push("/adminmain1")
       }
       else {
        alert("Credential Invalid")
       }
      })
  }



render() {
    return (

      <div style={{ marginBottom: "70px" }}>
       <nav className="navbar navbar-expand-lg shadow-lg p-3 mb-5 bg-white rounded">

<a className="navbar-brand" href="#">
          <img src={medicare} width="35" height="35" className="d-inline-block align-top"
alt="Logo" />
          Medicare
         </a>
         <button className="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav"
          aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
          <span className="navbar-toggler-icon"></span>
         </button>
         <div className="collapse navbar-collapse" id="navbarNav">

<ul className="navbar-nav ml-auto">
            <li className="nav-item">
              <Link className="nav-link" to="/">Home</Link>
            </li>
            <li className="nav-item">
              <Dropdown className="nav-item">
                <Dropdown.Toggle id="dropdown-basic">
                  Login as
                </Dropdown.Toggle>

                <Dropdown.Menu>
                  <Link className="nav-link" to="/login">Patient</Link>
                  <Link className="nav-link" to="/adminlogin">Admin</Link>
                  <Link className="nav-link" to="/doctorlogin">Doctor</Link>
                  <Link className="nav-link" to="/receptionistlogin">Receptionist</Link>
                </Dropdown.Menu>
              </Dropdown>
            </li>
            <li className="nav-item">
              <Dropdown className="nav-item">
                <Dropdown.Toggle id="dropdown-basic">
                  Register as
```

```
              </Dropdown.Toggle>

              <Dropdown.Menu>
                <Link className="nav-link" to="/register">Patient</Link>
              </Dropdown.Menu>
            </Dropdown>
          </li>

        </ul>
      </div>
    </nav>
    <h1 class="text-center">Admin Login</h1>

    <form onSubmit={(e) => this.submit(e, this.state.username)} method="POST">

<div class="form-group mt-3">
        <label for="exampleInputEmail1">Username</label>

<input onChange={(e) => this.setState({ username: e.target.value })}

value={this.state.username} type="text" class="form-control mt-2" id="exampleInputEmail1"
aria-describedby="emailHelp" placeholder="Enter User Name" required />
      </div>
      <div class="form-group mt-3">
        <label for="exampleInputPassword1">Password</label>
        <input onChange={(e) => this.setState({ password: e.target.value })}
value={this.state.password} type="password" class="form-control mt-2"
id="exampleInputPassword1" placeholder="Password" minLength="5" required />
      </div>
      <div class="form-group mt-3">

 <button type="submit" name="action" class="btn btn-primary">Submit</button>
      </div>
    </form>
  </div>
  )
 }
}

export default AdminLogin
```

Book Appointment

```
import React, { Component } from 'react'
import { Link } from 'react-router-dom'
import logo from '../images/logo.png'
import axios from "axios";
import medicare from '../images/medicare.jpg'

export class BookAppoint extends Component {

   constructor(props) {
      super(props);
      this.state = {
         users: [],
         username: '',
         fname: '',
         email: '',
         address: '',
         city: '',
         type: '',
         timeofapp: '',
         dateofapp: ''
      }
   }

   componentDidMount() {
      axios.get("http://localhost:8102/appointment/")
        .then((res) => {
          this.setState({

 users: res.data,
            username: '',

 fname: '',
            email: '',
            address: '',
            city: '',
```

```
            type: '',
            timeofapp: '',
            dateofapp: ''
          })
        })
    }

  submit(event, name) {
    event.preventDefault();
    axios.post("http://localhost:8102/appointment/", {
      username: localStorage.getItem('LoginUsername'),
      fname: this.state.fname,
      email: this.state.email,
      address: this.state.address,
      city: this.state.city,
      type: this.state.type,
      timeofapp: this.state.timeofapp,

      doctorname: 'None',
      status: 'Pending',
      dateofapp: this.state.dateofapp
    })
      .then((res) => {
        console.log(res);
        console.log(res.data);
        if (res.data == 0) {

          alert("Appointment booked")
          window.location="/edit"
        }
        else {
          alert("Credential Invalid")
        }
      })

  }

  render() {
    return (
      <div style={{ marginBottom: "70px" }}>
        <nav className="navbar navbar-expand-lg shadow-lg p-3 mb-5 bg-white rounded">
          <a className="navbar-brand" href="#">
            <img src={medicare} width="35" height="35" className="d-inline-block align-
top" alt="Logo" />
            Medicare
          </a>
          <button className="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-

label="Toggle navigation">
```

```
<span className="navbar-toggler-icon"></span>
                </button>
                <div className="collapse navbar-collapse" id="navbarNav">
                  <ul className="navbar-nav ml-auto">
                    <li className="nav-item">
                      <Link className="nav-link" to="/appointment">Book
Appointment</Link>
                    </li>
                    <li className="nav-item">
                      <Link className="nav-link" to="/edit">View Appointment</Link>

</li>
                    <li className="nav-item">
                      <Link className="nav-link" to="/">Logout</Link>
                    </li>

                  </ul>
                </div>
              </nav>



              <h1 class="text-center">Book Appointment</h1>
              <form onSubmit={(e) => this.submit(e, this.state.username)} method="POST">

                <div class="form-group mt-3">
                  <div>
                    <label for="inputEmail4">Full name</label>
                    <input onChange={(e) => this.setState({ fname: e.target.value })}
value={this.state.fname} type="text" class="form-control" id="inputEmail4" required />
                  </div>
                </div>
                <div class="form-group mt-3">
                  <div>
                    <label for="inputPassword4">Email address</label>
                    <input onChange={(e) => this.setState({ email: e.target.value })}
value={this.state.email} type="email" class="form-control" id="inputPassword4" required />
                  </div>
                </div>
                <div class="form-group mt-3">
                  <label for="inputAddress">Address</label>
                  <input onChange={(e) => this.setState({ address: e.target.value })}
value={this.state.address} type="text" class="form-control" id="inputAddress" required />
                </div>
                <div class="form-group mt-3">
                  <div>
                    <label for="inputCity">City</label>
                    <select onChange={(e) => this.setState({ city: e.target.value })}
value={this.state.city} id="inputState" class="form-control" required>
```

```
                        <option selected>Choose...</option>
                        <option>Mumbai</option>
                        <option>Mumbai suburbon</option>
                        <option>Navi Mumbai</option>


<option>Pune</option>
                        <option>Ahemadnagar</option>
                        <option>Ratnagiri</option>
                    </select>
                  </div>
                </div>
                <div class="form-group mt-3">
                  <div class="form-group">
                    <label for="inputState">Doctor type</label>

<select onChange={(e) => this.setState({ type: e.target.value })} value={this.state.type}
id="inputState" class="form-control" required>
                        <option selected>Choose...</option>
                        <option>Cardiologist</option>
                        <option>Gynecologist </option>
                        <option>Neurologist</option>
                        <option>Skin specialist</option>
                        <option>Dentist</option>
                    </select>
                  </div>


                  {/* <div class="form-group mt-3">
                    <label for="inputState">Select time</label>
                    <input onChange={(e) => this.setState({ timeofapp: e.target.value })}
value={this.state.timeofapp} type="text" id="inputMDEx1" class="form-control" />
                  </div> */}


                  <div class="form-group mt-3">
                    <label for="inputMDEx1">Select time</label>
                    <input onChange={(e) => this.setState({ timeofapp: e.target.value })}
value={this.state.timeofapp} type="time" id="inputMDEx1" class="form-control" required/>
                  </div>

                  <div class="form-group mt-3">
                    <label for="inputState">Enter date</label>
                    <input onChange={(e) => this.setState({ dateofapp: e.target.value })}
value={this.state.dateofapp} type="date" class="form-control" id="inputAddress" required/>
                  </div>
                </div>
                <div class="form-group mt-3">
                  <button type="submit" name="action" class="btn btn-primary">Book
Appointment</button>
```

```
                </div>
              </form>
            </div>
        )
    }
}

export default BookAppoint
```

Style.css

```
@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap');
body{
    font-family: 'Roboto', sans-serif !important;
}
.nav-link,
.navbar-brand{
  color: #000 !important;
}
ul{
  margin-left: auto !important;
}
.navbar-nav > li{
  padding-left:7px;
  padding-right:7px;
}
.img1{
  float:right;
  width:400;
  height:330;


  margin-left: 50px;
}
.img2{
  float:left;
  width:400;
  height:330;
  margin-right: 50px;
}
.card-title{
  font-size: 21px;
  font-weight:bold ;
```

```
}
.img-fluid {
  border: 15px solid ;
  border-radius: 5px;
  border-color : white ;
  border-image: 0px round;
  padding: 0px;
  height:750;
}
```

Backend Code

Entity Class:

```
package com.silchar.medicare.Entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name="user")
public class User {
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)

    int id;
    private String username;
    private String email;
    private String mobile;
    private String password;

          public int getId() {
                return id;
          }
         public void setId(int id) {
                this.id = id;
          }
         public String getUsername() {
                return username;
          }
```

```
	public void setUsername(String username) {
		this.username = username;
	 }
	public String getEmail() {
		return email;
	 }
	public void setEmail(String email) {
		this.email = email;
	 }
	public String getMobile() {
		return mobile;
	 }
	public void setMobile(String mobile) {
		this.mobile = mobile;
	 }
	public String getPassword() {
		return password;

}
	public void setPassword(String password) {
		this.password = password
}
}
```

Repository:

```
package com.silchar.medicare.Repository;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.websocket.Session;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import  org.springframework.data.repository.query.Param;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import com.google.common.base.Optional;
import com.silchar.medicare.Entity.User;
@Repository
public interface UserRepository extends JpaRepository<User,Integer> {
	@Override
	java.util.List<User> findAll();
}
```

Controller:

```
package com.silchar.medicare.Controller;
import java.sql.SQLException;
import java.util.List;
import javax.persistence.EntityManager;
import javax.websocket.Session;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import com.silchar.medicare.Entity.User;
import com.silchar.medicare.Repository.UserRepository;
import com.silchar.medicare.validate.UserValidate;



@ComponentScan(basePackages = {"com.silchar.medicare.Controller"})
@RestController
@RequestMapping("/api")

@CrossOrigin
public class UserController {
   @Autowired
   private UserRepository userRepository;

         @GetMapping(path="/loadpatient")
   public List<User> GetUsers () {
                  return userRepository.findAll();
   }

   @GetMapping("/updatebyiduser/{id}")
   public User GetUser(@PathVariable Integer id) {
      return userRepository.findById(id).orElse(null);
   }

   @PostMapping("/")
   public int PostUser(@RequestBody User user) {

         String username = user.getUsername();
         String email = user.getEmail();
```

```
String mobile = user.getMobile();
        String pass = user.getPassword();

//                  System.out.print(id+ "\n");

       User n = new User();
       n.setUsername(username);
       n.setEmail(email);
       n.setMobile(mobile);
       n.setPassword(pass);
       User result = userRepository.save(n);
       if(result != null) {
                  System.out.print("Record inserted");
                  return 0;
                        }
      return 1;
        }

   @PostMapping("/login")
   public int PostUser1(@RequestBody User user) {

            UserValidate udao=new UserValidate();
                  String username = user.getUsername();
                  String pass = user.getPassword();
                  boolean checkingLogin=udao.validate(username, pass);
                  System.out.print(username+ "\n");
                  if(checkingLogin) {
                     return 0;
                  }
                  return 1;

 }

   @PutMapping("/editpatient")
   public int update(@RequestBody User user){

         User result = userRepository.save(user);
                  if(result != null) {
                  System.out.print("Record updated");
                  return 0;
                  }
                  return 1;
   }

@DeleteMapping("/deluser/{id}")
   public int DeleteUser(@PathVariable int id) {
```

```
                userRepository.deleteById(id);
        return 0;
    }
}
```

Java Based Hibernate Mapping Class:

```
package com.silchar.medicare.Controller;

import java.util.Properties;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.cfg.Environment;
import org.hibernate.service.ServiceRegistry;
import com.silchar.medicare.Entity.Doctor;
import com.silchar.medicare.Entity.Receptionist;
import com.silchar.medicare.Entity.User;

public class HibernateUtil {
    private static SessionFactory sessionFactory;
     public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            try {
                Configuration configuration = new Configuration();

                // Hibernate settings equivalent to hibernate.cfg.xml's properties
                Properties settings = new Properties();
                settings.put(Environment.DRIVER, "com.mysql.cj.jdbc.Driver");
                settings.put(Environment.URL,
"jdbc:mysql://localhost:3306/hospital?allowPublicKeyRetrieval=true&useSSL=false");
                settings.put(Environment.USER, "root");
                settings.put(Environment.PASS, "nehru1234");
settings.put(Environment.SHOW_SQL, "true")
settings.put(Environment.HBM2DDL_AUTO, "update");
                settings.put(Environment.DIALECT, "org.hibernate.dialect.MySQL5InnoDBDialect");
                configuration.setProperties(settings);
                configuration.addAnnotatedClass(User.class);


                configuration.addAnnotatedClass(Doctor.class);
                configuration.addAnnotatedClass(Receptionist.class);

                ServiceRegistry serviceRegistry = new StandardServiceRegistryBuilder()
                    .applySettings(configuration.getProperties()).build();
                System.out.println("Hibernate Java Config serviceRegistry created");
                sessionFactory = configuration.buildSessionFactory(serviceRegistry);
                return sessionFactory;
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return sessionFactory;
    }
}
```

User Validate Class:

```
package com.silchar.medicare.validate;
import org.hibernate.Session;
import org.hibernate.Transaction;
import com.silchar.medicare.Controller.HibernateUtil;

import com.silchar.medicare.Entity.User;
public class UserValidate {
public boolean validate(String userName, String password) {

        Transaction transaction = null;
            User userlogin = null;
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {
            // start a transaction
            transaction = session.beginTransaction();
            // get an user object
            userlogin=(User)session.createQuery("FROM User U WHERE U.username =
:userName").setParameter("userName", userName)
            .uniqueResult();
if (userlogin != null && userlogin.getPassword().equals(password)) {
                return true;
            }
            // commit transaction
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                //transaction.rollback();
            }
            e.printStackTrace();
        }
        return false;
    }
}
```

Configuration Class:

```
package com.silchar.medicare;
import org.springframework.boot.SpringApplication;
```

```
import org.springframework.boot.autoconfigure.SpringBootApplication;
import springfox.documentation.swagger2.annotations.EnableSwagger2;
@SpringBootApplication
@EnableSwagger2


public class DemoApplication {
public static void main(String[] args) {
SpringApplication.run(DemoApplication.class, args);
                                    }
}
```

Chapter 11 Result

Fig 11.1 Home Page

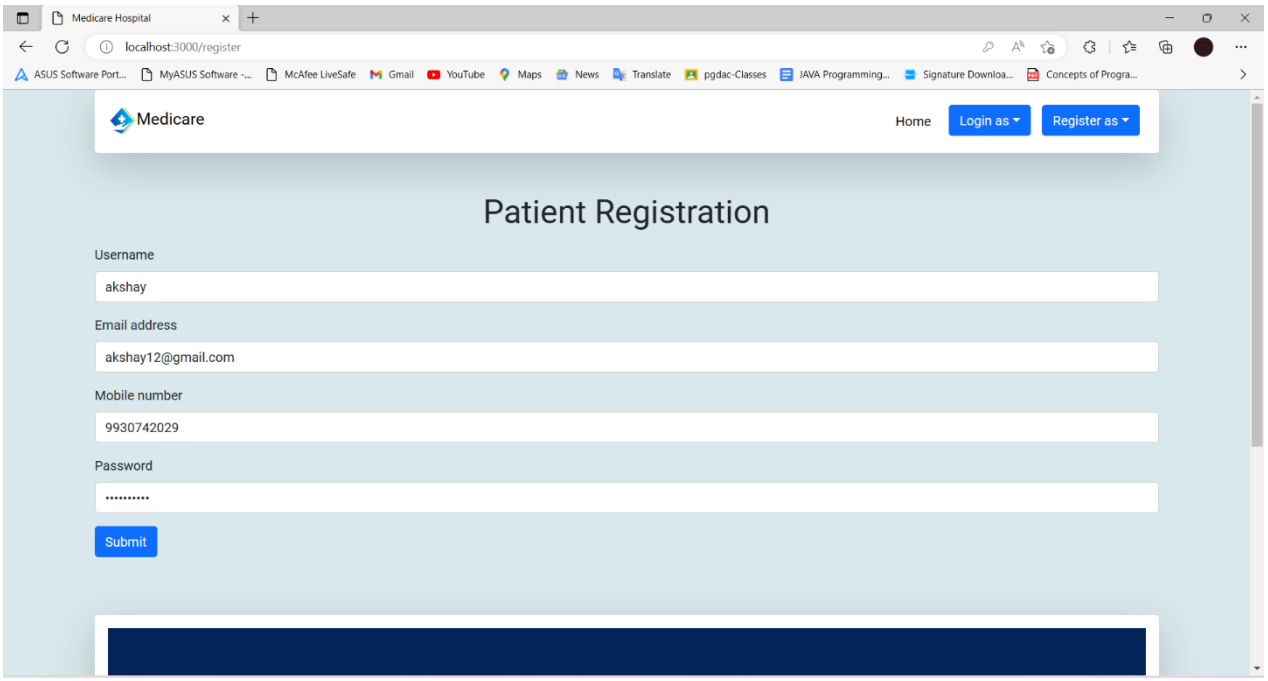

Fig 11.2: Patient Registration

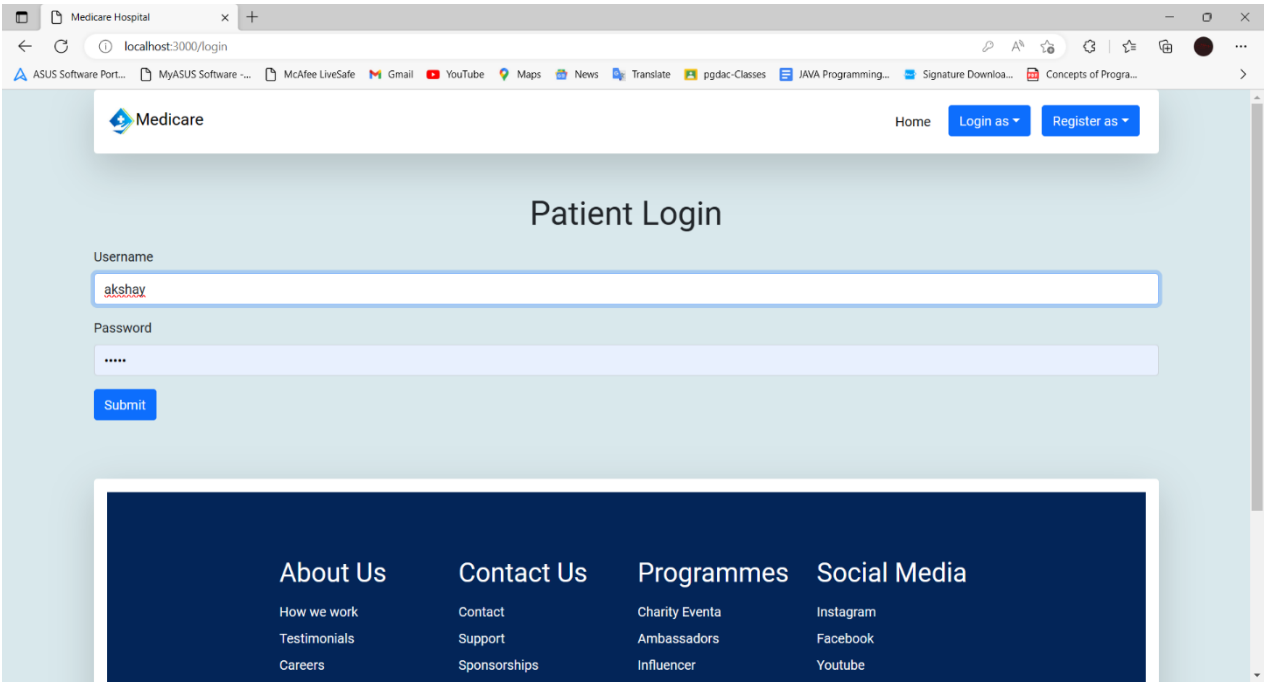

Fig 11.3: Patient Login

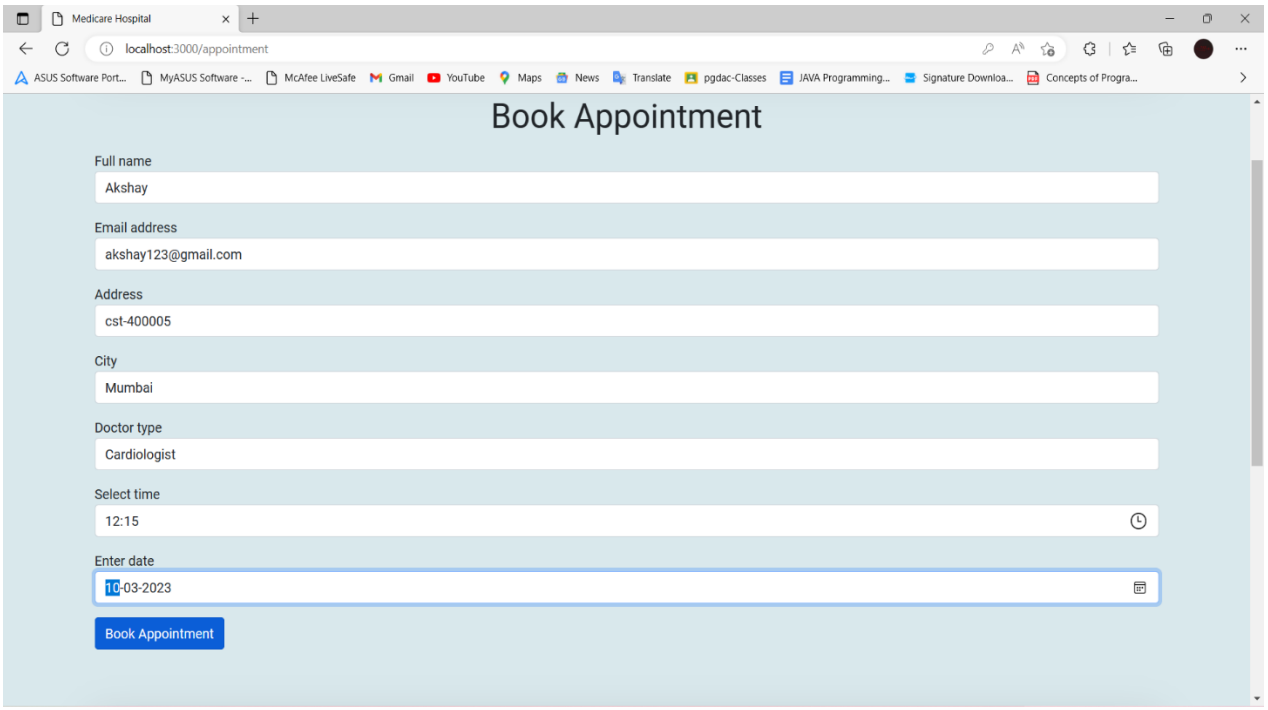

Fig 11.4: Appointment Booking by Patient

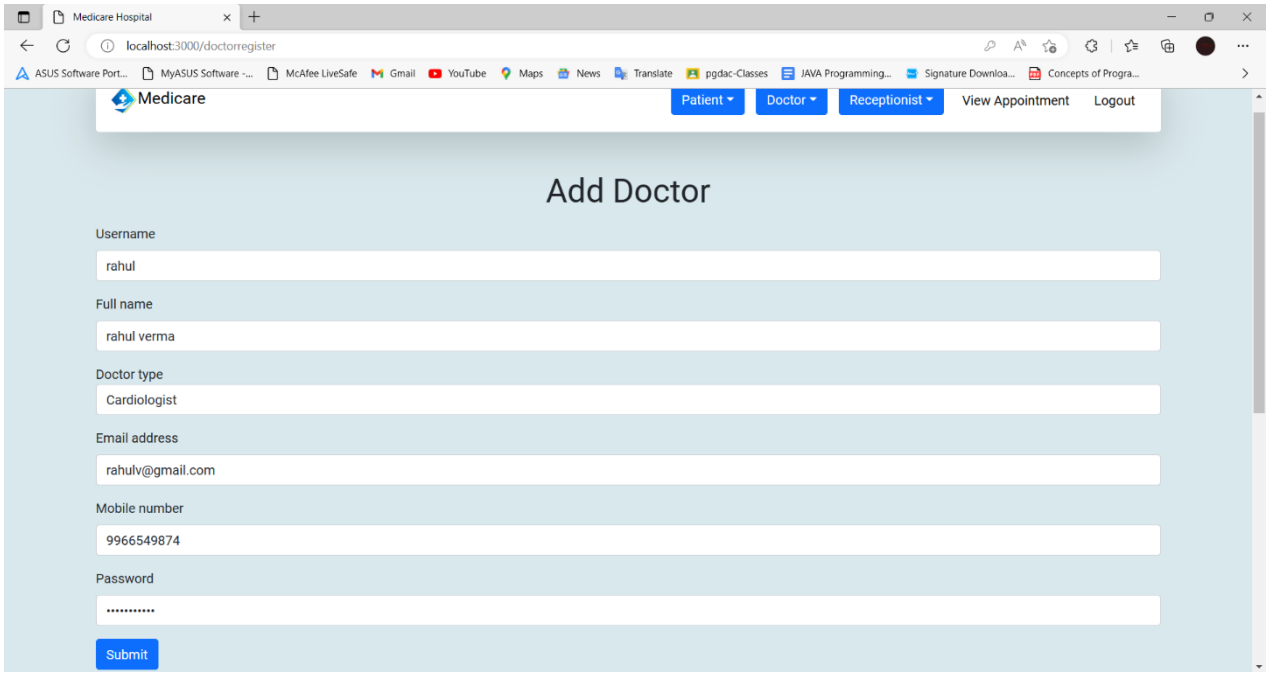

Fig 11.5: Admin Registering Doctor Details

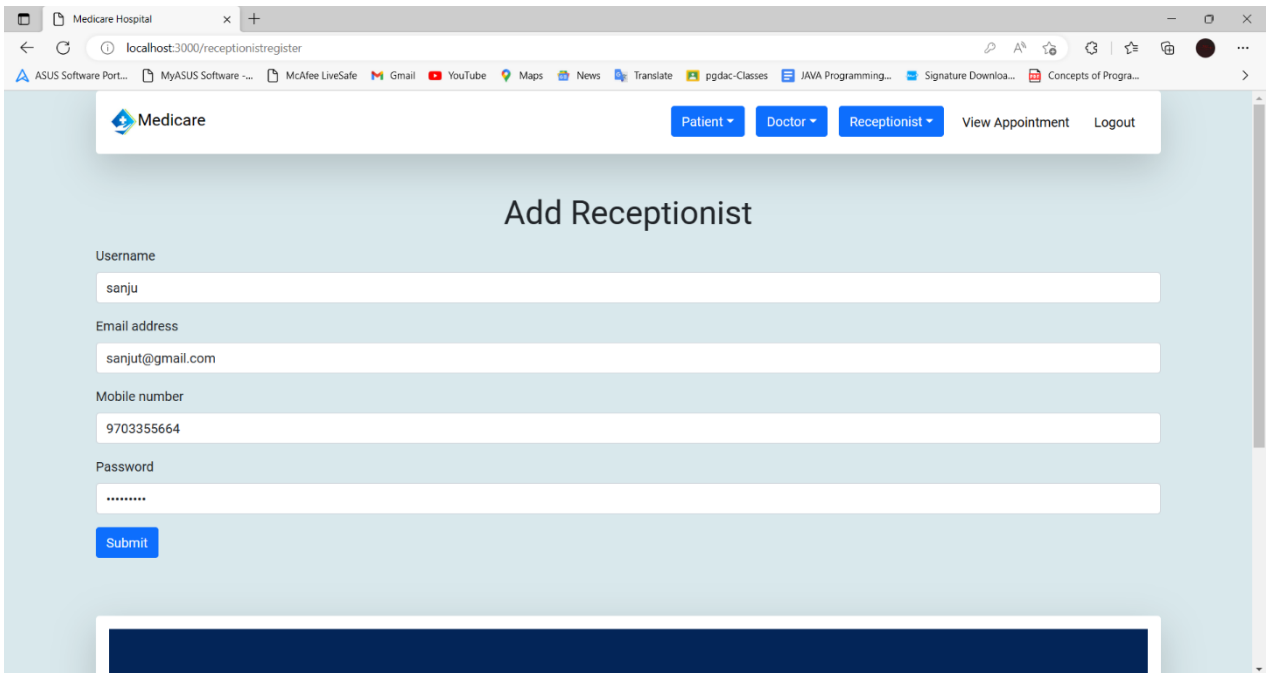

Fig 11.6: Admin Registering Receptionist Details

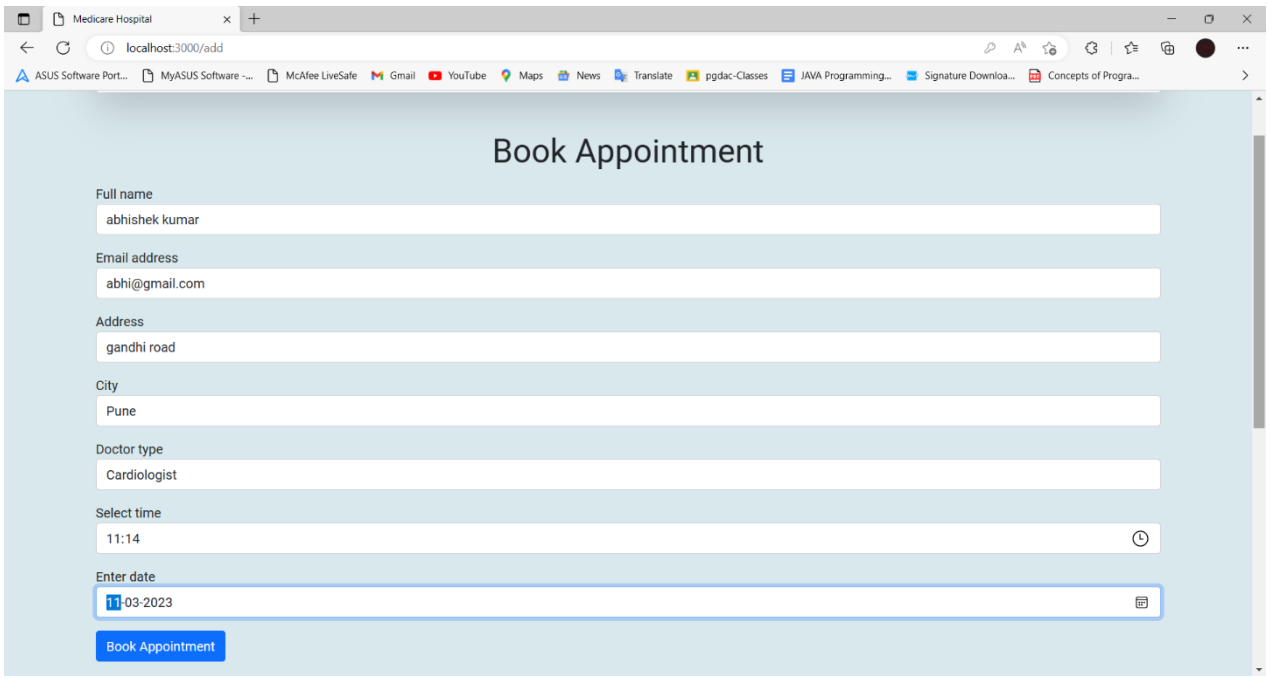

Fig 11.7: Appointment Booking by Receptionist

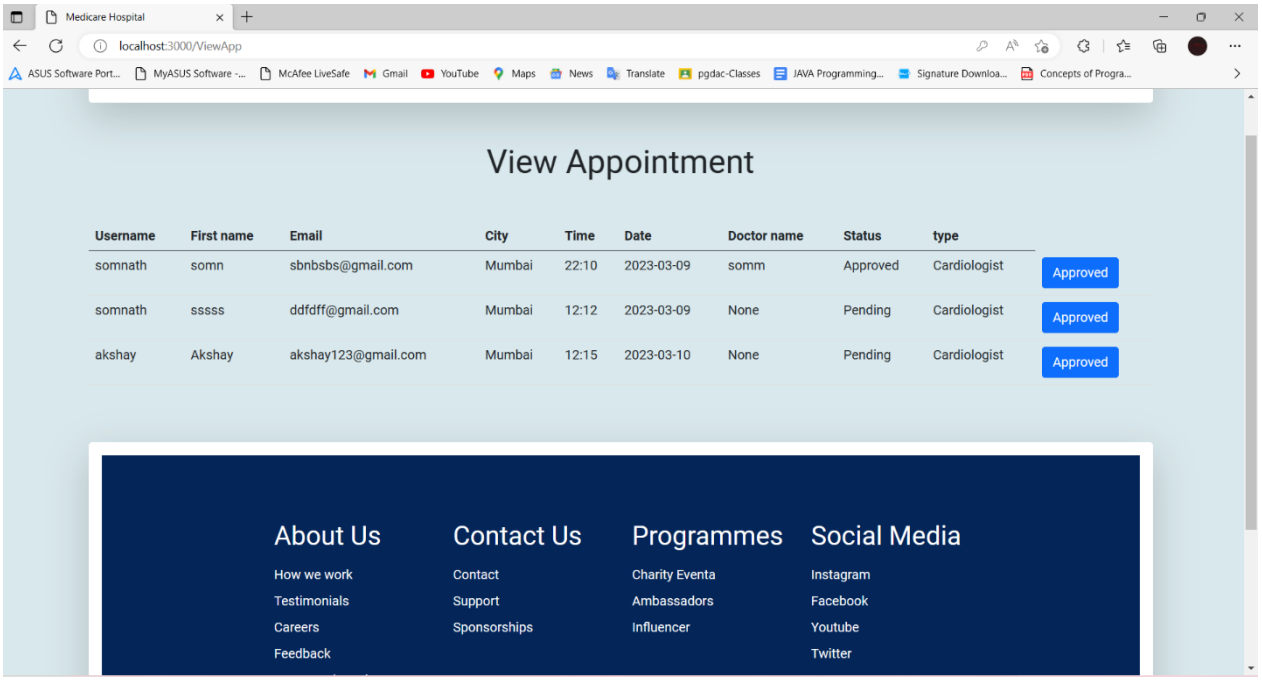

Fig 11.8: Doctor Approving the Appointments

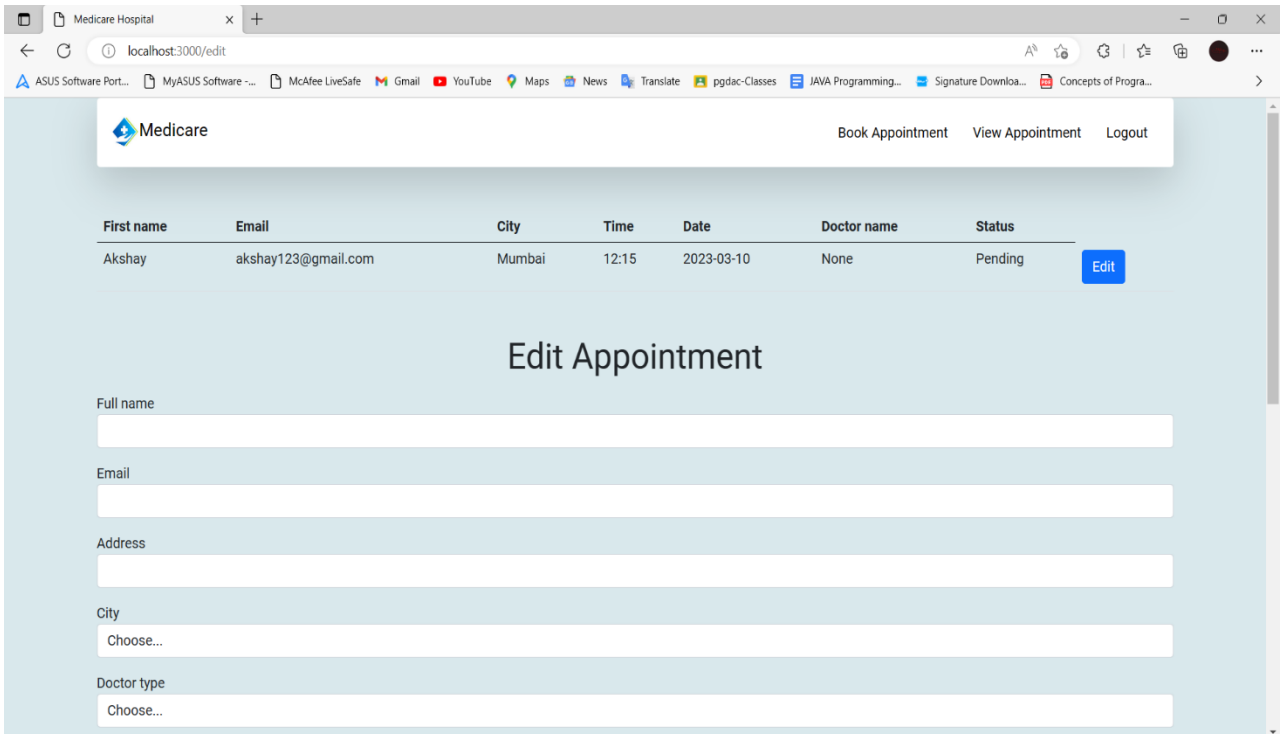

Fig 11.9 Patient can check Appointment status

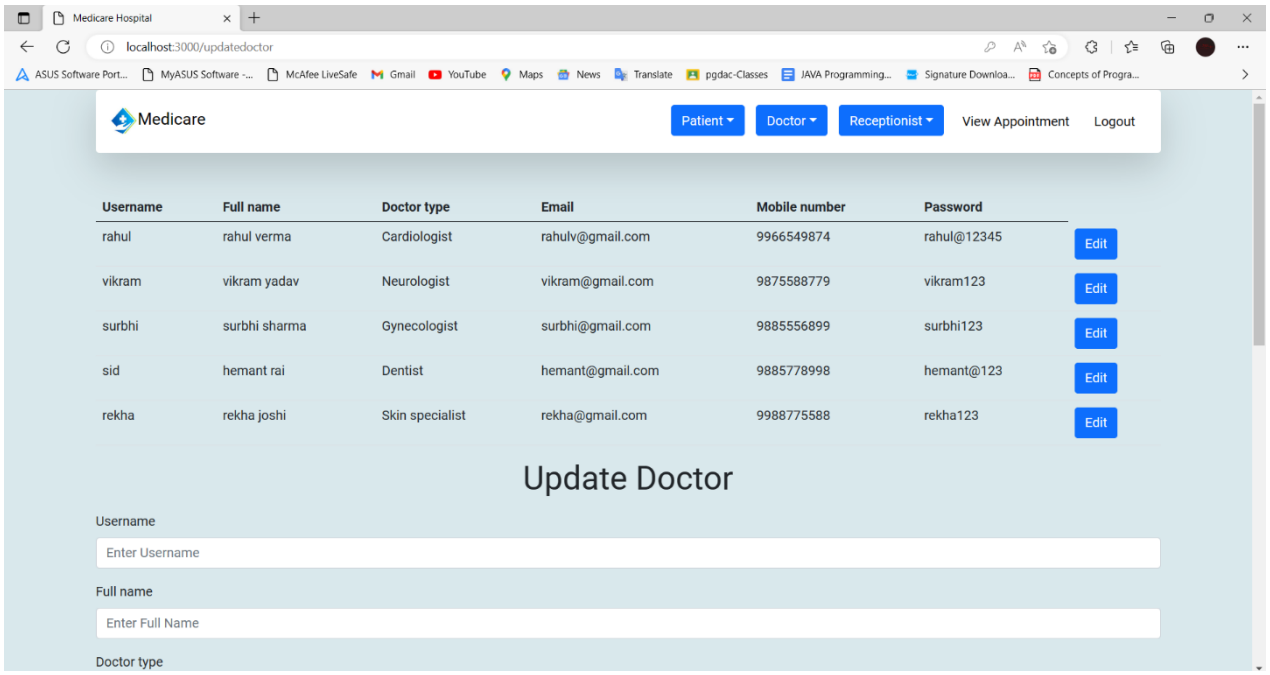

Fig 11.10 Admin can Update/Delete Doctor profile

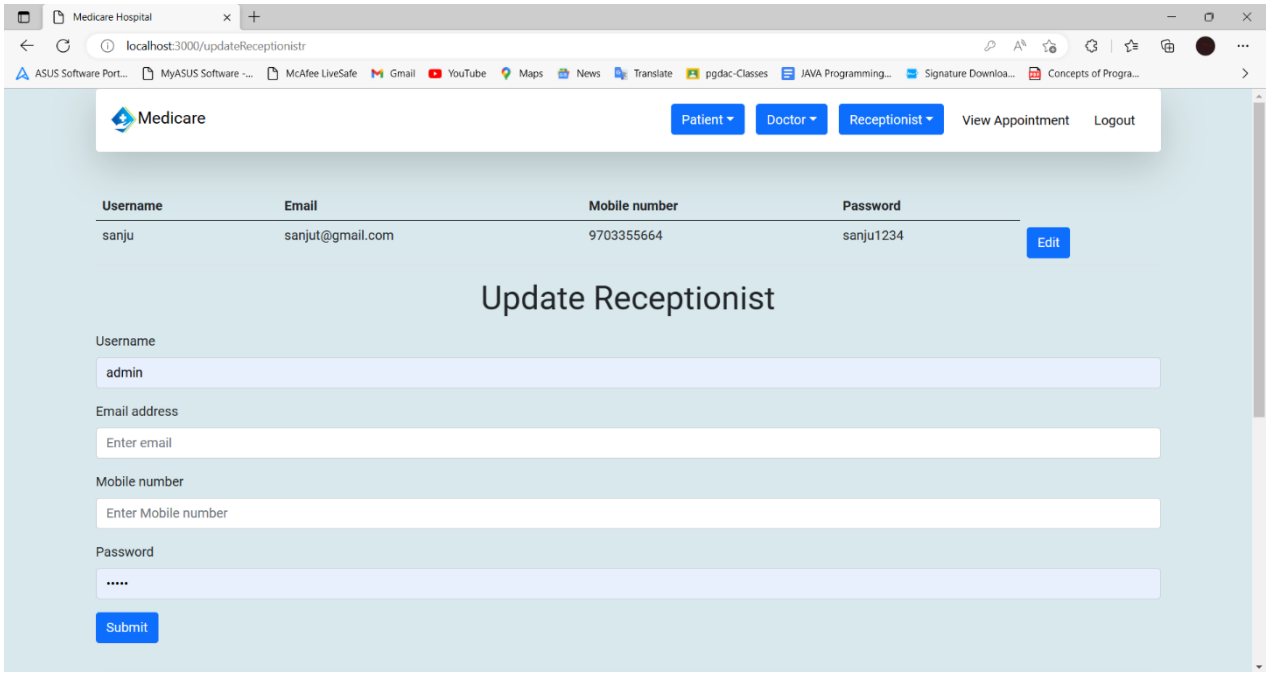

Fig 11.11 Admin can Update/Delete Receptionist profile

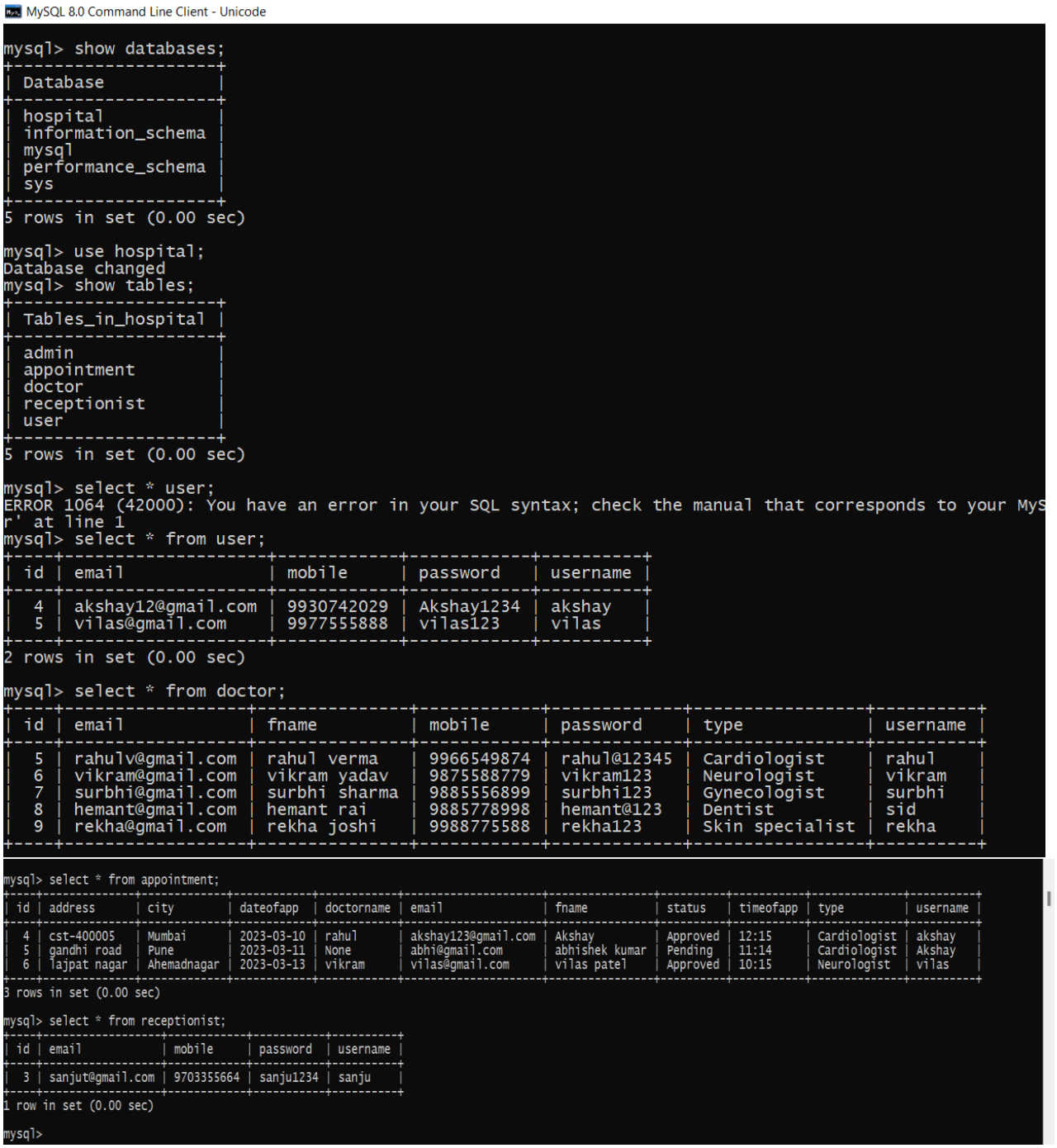

Fig 11.12 Database

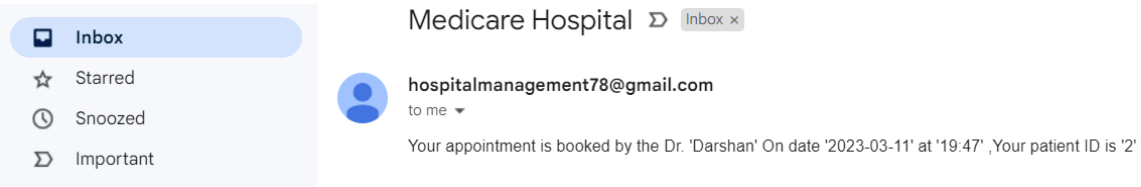

Fig 11.13 Email Confirmation

Chapter 12 Conclusion and Declaration

Our project is only a humble venture to satisfy the needs to manage their project work. Several user friendly coding have also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the school. The objective of software planning is to provide a frame work that enables the manger to make reasonable estimates made within a limited time frame at the beginning of the software project and should be updated regularly as the project progresses.

Recommendation

High security measures should be upheld in order to avoid cyber crime in this project. The key concept is to minimize the amount of paper and convert all forms of documentation to digital form. Itcan observe that the information required can be obtained with ease and accuracy in the computerized system. The user with minimum knowledge about computer can be able operate the system easily. The system also produces brief result required by the management.

Chapter 13 References

- Appointment scheduling in health care: Challenges and opportunities Diwakar Gupta & Brian Denton Graduate Program in Industrial & Systems Engineering, Department of Mechanical Engineering, University of Minnesota , 111 Church Street S.E., Minneapolis, MN, 55455, USA

- Outpatient appointment systems in healthcare: A review of optimization studies
  Author K.ling Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran,Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran,Goodman School of Business, Brock University, St. Catharines, Canada

- Appointment Scheduling Problem in Complexity Systems of the Healthcare Services: A Comprehensive Review
  Ali Ala and Feng Chen
  Department of Industrial Engineering & Management, Shanghai Jiao Tong University, Shanghai, China
  Sino-US Global Logistics Institute, Shanghai Jiao Tong University, Shanghai 200240, China

- Design of Appointment Systems for Preanesthesia Evaluation Clinics to Minimize Patient Waiting Times: A Review of Computer Simulation and Patient Survey Studies
  Dexter, Franklin MD, PhD
  Author Information
  Anesthesia & Analgesia 89(4):p 925, October 1999. | DOI: 10.1213/00000539-

- http://www.javaworld.com/javaworld/jw-01-1998/jw-01-Credentialreview.html

- Database Programming with JDBC and Java by O'Reilly

- Head First Java 2nd Edition

- http://www.jdbc-tutorial.com/

- Java and Software Design Concepts by Apress
- https://www.tutorialspoint.com/java/

- https://docs.oracle.com/javase/tutorial/

- http://www. geeksforgeeks/

- http://www. w3school/