## CHAPTER 1

## SYNOPSIS

## The main objective of the Doctor Appointment System is to manage the details of Doctor, Patient, Test, Medicine, Booking Number. It manages all the information about Doctor, Appointment, Booking Number, Doctor. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Doctor, Patient, Appointment, Test. It tracks all the details about the Test, Medicine, Booking Number.

The development of this system is to increase the function of patient record management. Besides that, it also to increase the security of patient record, minimize time in record calculation, and availability of record. The module that included in this system is patient record management, borrowing record, and record calculation. So, this computerized system will solve the problem that faced in the current manual system.

1. **INTRODUCTION**

Main idea behind developing this application is to provide a solution for hospitals to manage patient’s records and details in single software. Doctor appointment system application can manage information of patient’s health details, appointment details, payment details, past visiting details. This application will be very useful to handling patient details. We can easily identify that one particular patient details.

Ward wise view is very helpful for We can easily identified where patient to be admitted. In this project have one remembered screen which is very helpful to give a medicine to patient on that time. This system focused on the patient record management for surgical department which include the patient record and patient admission record. The development of this system is to increase the function of patient record management. Besides that, it also to increase the security of patient record, minimize time in record calculation, and availability of record. The module that included in this system is patient record management, borrowing record, and record calculation. So, this computerized system will solve the problem that faced in the current manual system.

* 1. **ORGANIZATION PROFILE**

Radnor Property Group is one of Pennsylvania’s leading real estate development companies with a specialty in forming public-private partnerships (PPPs) with non-profit institutions. In addition to forming PPPs, Radnor oftentimes develops and invests for its own portfolio. In PPPs Radnor partners with colleges, universities, religious institutions and other non-profits, assisting them in strategic real estate decisions and providing critical thinking and advice to implement financially successful real estate developments. As a partner RPG acts as a fiduciary in first meeting its public partners’ needs while focusing on profitable real estate development outcomes. It assists its partners’ by providing entrepreneurial expertise to maximize profits while adhering to the ethos of the organization.

Radnor Property Group was founded by David Yeager in 1999. Since then it has worked on over 3.8 million square feet of development projects and $330 million of improvements including over 25 nonprofit institutions in higher education and religious organizations. During that time Radnor has had ownership interest in approximately 900,000 square feet valued at approximately $210 million. The company specializes in complex urban projects involving ground lease arrangements, tax credits, public subsidy programs and Brownfield projects. Radnor’s specialty in PPPs has resulted in a unique niche for the company resulting in numerous development opportunities in off-campus student housing, commercial office, childcare facilities, specialty retail and recreation centers. Radnor’s proficiency in federal and state programs has resulted in numerous urban infill development projects utilizing New Markets Tax Credits, Historic Tax Credits, state grant and low-interest loan programs. Radnor’s senior management and staff have extensive experience in real estate development, finance, economic development, construction management, leasing, asset management, and property management. The company has in-depth knowledge of local markets, contacts, and state/local economic development programs.

* 1. **SYSTEM SPECIFICATION**

### JAVA

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications. The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types. Unlike Windows executables (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE).

**ORACLE**

Oracle Database (Oracle DB) is a relational database management system (RDBMS) from Oracle Corporation. Originally developed in 1977 by Lawrence Ellison and other developers, Oracle DB is one of the most trusted and widely used relational database engines for storing, organizing and retrieving data by type while still maintaining relationships between the various types.

The system is built around a relational database framework in which data objects may be directly accessed by users (or an application front end) through structured query language (SQL). Oracle is a fully scalable relational database architecture and is often used by global enterprises which manage and process data across wide and local area networks. The Oracle database has its own network component to allow communications across networks.

Oracle DB is also known as Oracle RDBMS and, sometimes, simply as Oracle.

Databases are used to provide structure and organization to data stored electronically in a computer system. Before they were adopted, early computers stored data in flat file structures where information in each file was separated by commas (CSV files). However, as the number of fields and rows that defined the characteristics and structure of each piece of data continued increasing, it was only a matter of time before this approach would become unmanageable.

Relational models for database management represented the ideal solution to this issue by organizing data in entities and attributes that further describe them. Today, Oracle Database represents the RDBMS with the largest market share. Oracle DB rivals Microsoft’s SQL Server in the enterprise database market. There are other database offerings, but most of this command tiny market share compared to Oracle DB and SQL Server. Fortunately, the structures of Oracle DB and SQL Server are quite similar, which is a benefit when learning database administration

* + 1. HARDWARE CONFIGURATION

Processor : P4 700 GHz

RAM Capacity : 4 GB

Hard Disk : 180 GB

1.2.2 SOFTWARE SPECIFICATION

Operating System : Windows 8

Front End : JAVA

Back End : Oracle

## CHAPTER 2

**SYSTEM STUDY**

**2.1 EXISTINGSYSTEM**

The existing system is bus pass management is very hard to manage manually. In this system passenger should take the pass anywhere we go. It may sometime chance to miss the pass. Passenger only responsible for the bus pass. This may hard to use this system which is have some drawbacks

**2.1.1 DRAWBACKS**

The existing system has the following drawbacks.

* Waiting a long time in queue.
* Getting some irritation from the nurse response.
* We can’t track the patient manually.

## 2.2 PROPOSEDSYSTEM

This tracking system will help to tracking the patient as room wise. Attender directly find the patient room and patient details. Doctors also use this application which will be very useful in the hospital. Everyone can be known about the patient details.

**2.2.1 FEATURES**

* No need to waiting in queue
* Manpower reduces
* Time saving

## FEASIBILITYSTUDY

The feasibility of the system is analyzed in this phase and business proposal is put forth with general plan for the project and cost estimates. During the system analysis of the project, the feasibility study of proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in feasibility analysis are

* + - Technical Feasibility
    - Economic Feasibility
    - Operational Feasibility

## Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, whether the certain current resources and technology can be upgraded or added in the software to accomplish specified user requirements.

The technical requirements of the application are simple and basic. Python is used for the developers of the application and the framework is largely used by many, thus there will be enough support for future enhancements. The framework is stable and the support from the developers is constantly updated. The devices which have internet connectivity are enough for the application.

## Economic Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on.

The cost of application development is very less and the cost of implementation is also less. It can be developed with the system with minimum requirements and can also be operated with the system with some basic requirements that are available the existing systems. For this, it is essential to consider expenses made on purchases and activities required to carry out software development.

## Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve user requirements. This feasibility is dependent on developer and involves visualizing whether the software will operate after it has been developed and be operative.

The application is developed based on the user requirements and is developed on the priority of the user requirements such as an integrated service and reviewing platform.

## PROBLEMANALYSIS

The orders from public are submitted through email, letter or form to the corresponding authority. The registered orders are forwarded to corresponding authority and report about the orders is submitted. The status of orders addressed is not recorded and maintained properly. The maps are not implemented. So, the customer does not know live location of the package. There is need for the software which receives the orders through online, forwarded to respective peoples for rectification and post the status of orders.

## CHAPTER 3

**SYSTEM DESIGN AND DEVELOPMENT**

**3.1 FILE DESIGN**

A file system is the data structure designed to support the abstraction of the data blocks as an archive and collection of files. In other words, a file system organizes the data blocks into files, directories, and file information. This data structure is unique because it is stored on secondary storage (usually the disk), which is a very slow device.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Directory | File 1 | File 2 | File 3 | Unsued Space |

sequence of sectors: 0 → n

The directory is a fixed size of 10 sectors.

The **directory entries** are at the beginning of the file. They contain information about the files in the directory. These entries have three fields:

* A null terminated string for the file name
* the starting block numbers
* number of blocks allocated.

**3.2 INPUT DESIGN**

Input Design is the process of converting a user-oriented description of the input into a computer-based system. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. Input Design is the first phase in the system design. Input designing is to converting the user-oriented information to the computer- oriented form. The input data items are grouped and analyzed to find out whether the proposed system can be developed from the user input. The system is developed using various processes screens formats.

The main objective of input design is to

* + - It should serve specific purpose effectively such as storing, recording, and retrieving the information.
    - It ensures proper completion with accuracy.
    - It should be easy to fill and straightforward.
    - It should focus on user’s attention, consistency and simplicity.

**3.3 OUTPUT DESIGN**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### Objectives of Output Design

The objectives of input design are −

* To develop output design that serves the intended purpose and eliminates the production of unwanted output.
* To develop the output design that meets the end users’ requirements.
* To deliver the appropriate quantity of output.
* To form the output in appropriate format and direct it to the right person.
* To make the output available on time for making good decisions.

Let us now go through various types of outputs −

### External Outputs

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

### 3.4 DATABASE DESIGN

The most important consideration in designing the database is how the information will be used. The main objective of designing a database is Data Integration, Data Integrity and Data Independence.

### Data Integration

In a database, information from several files is coordinated, accessed and operated upon as through it is in a single file. Logically, the information is centralized, physically, the data may be located on different devices, connected through data communication facilities.

### Data Integrity

Data integrity means storing all data in one place only and how each application access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy, that is data items need not be duplicated. A reduction in the direct access storage requirement.

### Data Independence

Dataindependenceistheinsulationofapplicationprogramsfromchangingaspects of physical data organization. This objective seeks to allow changes in the content and organizationofphysicaldatawithoutreprogrammingofapplicationandallowmodifications to application programs without reorganizing the physical data.

**3.5 SYSTEM DEVELOPMENT**

**3.5.1 DESCRIPTION OF MODULE**

The main module in this project is listed below

* Admin Login
* Doctor Registration
* Case Registration
* City wise & district wise count
* Accident history

1. **Admin Login**

Here admin is an only one login to follow this software, if there’s a lot of admins are there, they can also normally use it. Admin the main role is a do after the login.

1. **Accident details entry**

If anywhere the accident has been happened the admin collect all the information and store this module. These values are stored into the database table. Once the details have been captured it will automatically close.

1. **Case Registration**

This module will be handling the process of register the case to the respective members. If once case has been scheduled police department will take care for the investigation.

1. **City wise & district wise count**

We can search and count details as date wise, city and district wise which given cumulative frequency result, which may avoid the solution for the frequent accident.

1. **Accident history**

We can track the accident history by the accident id, collect the all the information about the accident and display in front of the screen,

## CHAPTER 4

## SYSTEM TESTING

Testingisanintegralpartofanysystemdevelopmentlifecycle.Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software aretobetestedforthebestqualityassurancethatthesystemmeetsthespecificationand requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

* + - * Unit Testing
      * Integration Testing
      * Validation Testing

### UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

### INTEGRATION TESTING

Integration Testing is the process of this activity can be considered as testing the design and hence module interaction. The primary objective of integration testing is to discover errors in the interfaces between the components. Login form and registration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submitting complaints.

### VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested independently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, email for further process.

## CHAPTER 5

* 1. **CONCLUSION**

This project can be mainly used to managing the accident records, who doing regular accidents can be monitor by this system. At spot the respective police can easily identify the criminal’s derails as well. Each and every record should be register by the application. These data’s will be using for searching the person history

**BIBLIOGRAPHY**

* **AG98**  
  Ken Arnold and James Gosling, The Java Programming Language, second ed., Addison-Wesley, 1998.
* **Chan98**  
  Patrick Chan, The Java Developers Almanac, Addison-Wesley, 1998.
* **CM96**  
  Peter Coad and Mark Mayfield, Java Design: Building Better Apps and Applets, Yourdon Press, 1996.
* **CH97**  
  Gary Cornell and Cay S. Horstmann, Core Java, second ed., SunSoft Press, 1997
* **ELW98**  
  Robert Eckstein and Marc Loy and Dave Wood, Java Swing, O'Reilly, 1998.
* **Englander97**  
  Robert Englander, Developing Java Beans, O'Reilly, 1997.

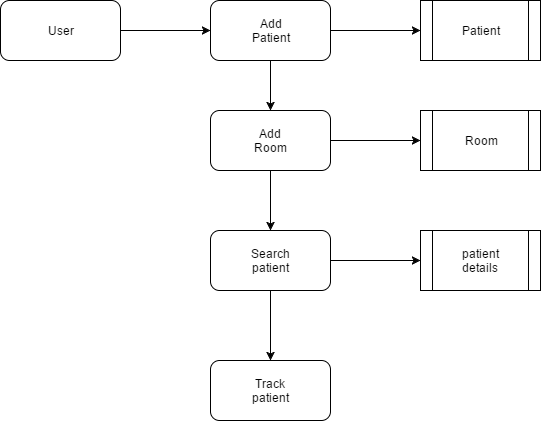
**APPENDICES**

1. **DATA FLOW DIAGRAM**

**LEVEL 0:**

****

**LEVEL 1:**

****

**B. TABLE STRUCTURE**

**TABLE NAME: ADMIN**

**PRIMARY\_KEY: ID**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| Id | Int |  | Admin id |
| Username | Varchar | 30 | Admin username |
| password | Varchar | 30 | Admin password |

**TABLE NAME: PATIENT**

**PRIMARY KEY: ID**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| Id | Int | 10 | Patient id |
| Name | Varchar | 30 | Application name |
| Address | Varchar | 30 | Address |
| Contactno | Varchar | 10 | Contact number |
| Emergency | Varchar | 10 | Emergency num |
| Age | Int | 10 | age |

**TABLE NAME: ADMISSION**

**PRIMARY KEY: ADMISSIONID**

**FOREIGN KEY: UID**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **DESCRIPTION** |
| Admissionid | Int | 10 | Admission id |
| Uid | Int | 10 | User id |
| Floor | Varchar | 30 | Floor name |
| Room no | Varchar | 30 | Room number |
| Doctor | Varchar | 10 | Doctor |
| Disease | Varchar | 10 | Disease |

1. **SAMPLE CODING**

<html>

<head>

<link rel="stylesheet" type="text/css" href="/stylesheets/bootstrap.css">

<link rel="stylesheet" type="text/css" href="/stylesheets/login.css">

<link rel="stylesheet" type="text/css" href="/stylesheets/sweetalert.css">

<script src="/javascripts/jquery.js"></script>

<script src="/javascripts/bootstrap.js"></script>

<script src="/javascripts/sweetalert.js"></script>

<script src="/javascripts/src/login.js"></script>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-sm-6 col-md-4 col-md-offset-4">

<h1 class="text-center login-title">Sign in to continue Pooling Vote</h1>

<div class="account-wall">

<img class="profile-img" src="https://lh5.googleusercontent.com/-b0-k99FZlyE/AAAAAAAAAAI/AAAAAAAAAAA/eu7opA4byxI/photo.jpg?sz=120"

alt="">

<input type="text" name="username" class="form-control" placeholder="Username" required autofocus>

<input type="password" name="password" class="form-control" placeholder="Password" required>

<button class="btn btn-lg btn-primary btn-block" id="login()" onclick="login()">

Sign in</button>

</div>

</div>

</div>

</div>

</body>

</html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="/stylesheets/bootstrap.css">

<link rel="stylesheet" type="text/css" href="/stylesheets/admin.css">

<link rel="stylesheet" type="text/css" href="/stylesheets/sweetalert.css">

<script src="/javascripts/jquery.js"></script>

<script src="/javascripts/bootstrap.js"></script>

<script src="/javascripts/sweetalert.js"></script>

<script src="/javascripts/chart.js"></script>

<!--<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>-->

<script src="/javascripts/src/admin.js"></script>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-md-12">

<!-- Nav tabs -->

<div class="card">

<ul class="nav nav-tabs" role="tablist">

<li role="presentation" class="active"><a href="#profile" aria-controls="profile" role="tab" data-toggle="tab">Profile</a></li>

<li role="presentation"><a href="#candidate" aria-controls="candidate" role="tab" data-toggle="tab">Candidate</a></li>

<li role="presentation"><a href="#voter" aria-controls="voter" role="tab" data-toggle="tab">Voter</a></li>

<li role="presentation"><a href="#report" aria-controls="report" role="tab" data-toggle="tab">Report</a></li>

<li role="presentation"><a onclick="window.location.href='/'" style="cursor:pointer" aria-controls="report" role="tab" data-toggle="tab">Logout</a></li>

</ul>

<div class="tab-content">

<th>Count</th>

</tr>

</thead>

<tbody id="collegesec ">

</tbody>

</table>

</div>

<div class ="col-sm-6 ">

<h1 style="color:BLUE ">Finance Secretary</h1>

<table class="table table-bordered ">

<thead>

<tr>

<th>Rank</th>

<th>Name</th>

<th>Count</th>

</tr>

</thead>

<tbody id="financesec ">

</tbody>

</table>

</div>

<div class ="col-sm-6 ">

<h1 style="color:BLUE ">Treasurer</h1>

<table class="table table-bordered ">

<thead>

<tr>

<th>Rank</th>

<th>Name</th>

<th>Count</th>

</tr>

</thead>

<tbody id="treasurer ">

</tbody>

</table>

</div>

<div class ="col-sm-6 ">

<h1 style="color:BLUE ">Sports Secretary</h1>

<table class="table table-bordered ">

<thead>

<tr>

<th>Rank</th>

<th>Name</th>

<th>Count</th>

</tr>

</thead>

<tbody id="sportssec ">

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Modal -->

<div id="myModal " class="modal fade " role="dialog ">

<div class="modal-dialog ">

<!-- Modal content-->

<div class="modal-content ">

<div class="modal-header ">

<button type="button " class="close " data-dismiss="modal ">&times;</button>

<h4 class="modal-title "> Deatail</h4>

</div>

<div class="modal-body ">

<div class="row ">

<div class="col-xs-12 ">

<div class="well well-sm ">

<div class="row ">

<div class="col-sm-6 col-md-4 ">

<img id="modalimg " src="http://placehold.it/380x500 " alt=" " class="img-rounded img-responsive " />

</div>

<div class="col-sm-6 col-md-8 ">

<h4 id="modname "></h4>

<small><cite id="modmobile " title="San Francisco, USA "></cite></small>

<p>

<i class="glyphicon glyphicon-globe "></i><a id="modpost "></a>

<br />

<i class="glyphicon glyphicon-envelope "></i><a id="modemail "></a>

<br />

<i class="glyphicon glyphicon-gift "></i><a id="modaddress "></a>

</p>

<!-- Split button -->

</div>

</div>

</div>

</div>

</div>

</div>

<div class="modal-footer ">

<button type="button " class="btn btn-default " data-dismiss="modal ">Close</button>

</div>

</div>

</div>

</div>

</body>

</html>

package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

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.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.response.GetAppointmentDetailsResponse;

import com.example.demo.response.GetBillingResponse;

import com.example.demo.response.GetCustomerResponse;

import com.example.demo.response.GetDoctorListResponse;

import com.example.demo.response.GetEmployeeResponse;

import com.example.demo.response.GetPatientResponse;

import com.example.demo.response.GetProductResponse;

import com.example.demo.response.GetStockResponse;

import com.example.demo.service.ApiService;

@RestController

@RequestMapping(value = { "/api" })

public class ApiController {

@Autowired

ApiService service;

@GetMapping("/login/{username}/{password}")

public Integer login(@PathVariable String username,@PathVariable String password) {

List<Object[]> res = service.login(username,password);

if(res.size()==0) {

return 0;

}else {

return (Integer)res.get(0)[0];

}

}

@PostMapping("/add\_patient/{firstname}/{lastname}/{mobile}/{email}/{attendar}/{emergency}")

public String add\_patient(@PathVariable String firstname,

@PathVariable String lastname,

@PathVariable String mobile,

@PathVariable String email,

@PathVariable String attendar,

@PathVariable String emergency) {

service.add\_patient(firstname,lastname,mobile,email,attendar,emergency);

return "Patient Saved Sucessfully";

}

@PostMapping("/adddoctor/{doctorname}/{qualification}/{speciality}/{username}/{password}")

public String adddoctor(@PathVariable String doctorname,

@PathVariable String qualification,

@PathVariable String speciality,

@PathVariable String username,

@PathVariable String password) {

service.adddoctor(doctorname,qualification,speciality,username,password);

return "Doctor Saved Sucessfully";

}

@GetMapping("/get\_patient/{mobile}")

public ResponseEntity<GetPatientResponse> get\_mobile(@PathVariable String mobile) {

return ResponseEntity.ok().body(service.get\_patient(mobile));

}

@PostMapping("/book\_appointment/{doctor\_id}/{patient\_id}/{amount}/{appointment\_date}")

public String book\_appointment(

@PathVariable Integer doctor\_id,

@PathVariable Integer patient\_id,

@PathVariable Integer amount,

@PathVariable String appointment\_date) {

service.book\_appointment(doctor\_id,patient\_id,amount,appointment\_date);

return "Appointment Booked Sucessfully";

}

@GetMapping("/get\_doctorlist")

public ResponseEntity<List<GetDoctorListResponse>> get\_doctorlist() {

return ResponseEntity.ok().body(service.get\_doctorlist());

}

@GetMapping("/get\_appointment\_details\_doctor/{id}/{date}")

public ResponseEntity<List<GetAppointmentDetailsResponse>> get\_appointment\_details\_doctor(@PathVariable Integer id,@PathVariable String date) {

return ResponseEntity.ok().body(service.get\_appointment\_details\_doctor(id,date));

}

}

package com.example.demo.dao;

import java.math.BigInteger;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.NativeQuery;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

@Repository

public class ApiDao {

@Autowired

SessionFactory sf;

public void add\_employee(String name, String mobile, String address, String gender, Integer salary, Integer age) {

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `employee` (`id`, `name`, `mobile`, `address`, `gender`, `salary`, `age`) VALUES (NULL, '"

+ name + "', '" + mobile + "', '" + address + "', '" + gender + "', '" + salary + "', '" + age + "');";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_customer(String name, String mobile, String address, String gender, String email) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `customer` (`id`, `name`, `mobile`, `address`, `gender`, `email`) VALUES (NULL, '"

+ name + "', '" + mobile + "', '" + address + "', '" + gender + "', '" + email + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_employee() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select \* from employee";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> get\_customer() {

Session session = sf.getCurrentSession();

String sql = "Select \* from customer";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_product(String company, String model, Integer price) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `product` (`id`, `company`, `model`, `price`) VALUES (NULL, '" + company + "', '"

+ model + "', '" + price + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_product() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select \* from product";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_purchase(Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `purchase` (`id`, `product\_id`, `quantity`, `date`) VALUES (NULL, '" + product\_id

+ "', '" + quantity + "', current\_timestamp());";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_sales(Integer customer\_id, Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `sales` (`id`, `customer\_id`, `product\_id`, `quantity`) VALUES (NULL, '" + customer\_id

+ "', '" + product\_id + "', '" + quantity + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_stock() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select p.company,p.model,COALESCE(sum(pqty),0) - COALESCE(sum(sqty),0) qty from product p \r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) pqty from purchase GROUP by product\_id) as a on a.product\_id = p.id\r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) sqty from sales GROUP by product\_id) as b on b.product\_id = p.id\r\n"

+ "GROUP BY p.company,p.model";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public Integer get\_mobile(String mobile) {

Session session = sf.getCurrentSession();

String sql = "Select id,name from customer where mobile='" + mobile + "'";

NativeQuery nq = session.createNativeQuery(sql);

List<Object[]> list = nq.getResultList();

if (list.size() != 0) {

return (Integer) list.get(0)[0];

} else {

return null;

}

}

public List<Object[]> get\_billing() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select customer.name,customer.mobile,product.company,product.model,sales.quantity,product.price from sales LEFT JOIN customer on(customer.id=sales.customer\_id) LEFT JOIN product on(product.id=sales.product\_id) ";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_patient(String firstname, String lastname, String mobile, String email, String attendar,

String emergency) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `patient` (`id`, `firstname`, `lastname`, `mobile`, `email`, `attendar`, `emergency`) "

+ "VALUES (NULL, '"+firstname+"', '"+lastname+"', '"+mobile+"', '"+email+"', '"+attendar+"', '"+emergency+"');";

session.createSQLQuery(sql).executeUpdate();

}**package** com.example.demo.response;

**public** **class** GetProductResponse {

**private** Integer id;

**private** String company;

**private** String model;

**private** Integer price;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getCompany() {

**return** company;

}

**public** **void** setCompany(String company) {

**this**.company = company;

}

**public** String getModel() {

**return** model;

}

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** Integer getPrice() {

**return** price;

}

**public** **void** setPrice(Integer price) {

**this**.price = price;

}

}

package com.example.demo.dao;

import java.math.BigInteger;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.NativeQuery;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

@Repository

public class ApiDao {

@Autowired

SessionFactory sf;

public void add\_employee(String name, String mobile, String address, String gender, Integer salary, Integer age) {

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `employee` (`id`, `name`, `mobile`, `address`, `gender`, `salary`, `age`) VALUES (NULL, '"

+ name + "', '" + mobile + "', '" + address + "', '" + gender + "', '" + salary + "', '" + age + "');";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_customer(String name, String mobile, String address, String gender, String email) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `customer` (`id`, `name`, `mobile`, `address`, `gender`, `email`) VALUES (NULL, '"

+ name + "', '" + mobile + "', '" + address + "', '" + gender + "', '" + email + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_employee() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select \* from employee";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> get\_customer() {

Session session = sf.getCurrentSession();

String sql = "Select \* from customer";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_product(String company, String model, Integer price) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `product` (`id`, `company`, `model`, `price`) VALUES (NULL, '" + company + "', '"

+ model + "', '" + price + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_product() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select \* from product";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_purchase(Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `purchase` (`id`, `product\_id`, `quantity`, `date`) VALUES (NULL, '" + product\_id

+ "', '" + quantity + "', current\_timestamp());";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_sales(Integer customer\_id, Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `sales` (`id`, `customer\_id`, `product\_id`, `quantity`) VALUES (NULL, '" + customer\_id

+ "', '" + product\_id + "', '" + quantity + "');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_stock() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select p.company,p.model,COALESCE(sum(pqty),0) - COALESCE(sum(sqty),0) qty from product p \r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) pqty from purchase GROUP by product\_id) as a on a.product\_id = p.id\r\n"

+ "LEFT JOIN (select product\_id,COALESCE(SUM(quantity),0) sqty from sales GROUP by product\_id) as b on b.product\_id = p.id\r\n"

+ "GROUP BY p.company,p.model";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public Integer get\_mobile(String mobile) {

Session session = sf.getCurrentSession();

String sql = "Select id,name from customer where mobile='" + mobile + "'";

NativeQuery nq = session.createNativeQuery(sql);

List<Object[]> list = nq.getResultList();

if (list.size() != 0) {

return (Integer) list.get(0)[0];

} else {

return null;

}

}

public List<Object[]> get\_billing() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select customer.name,customer.mobile,product.company,product.model,sales.quantity,product.price from sales LEFT JOIN customer on(customer.id=sales.customer\_id) LEFT JOIN product on(product.id=sales.product\_id) ";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public void add\_patient(String firstname, String lastname, String mobile, String email, String attendar,

String emergency) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `patient` (`id`, `firstname`, `lastname`, `mobile`, `email`, `attendar`, `emergency`) "

+ "VALUES (NULL, '"+firstname+"', '"+lastname+"', '"+mobile+"', '"+email+"', '"+attendar+"', '"+emergency+"');";

session.createSQLQuery(sql).executeUpdate();

}

public Object[] get\_patient(String mobile) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select id,firstname,lastname from patient where mobile='" + mobile + "'";

NativeQuery nq = session.createNativeQuery(sql);

List<Object[]> list = nq.getResultList();

if (list.size() != 0) {

return list.get(0);

} else {

return null;

}

}

public void book\_appointment(Integer doctor\_id, Integer patient\_id, Integer amount, String appointment\_date) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String q1 = "SELECT \* FROM `appointment` WHERE `doctor\_id`="+doctor\_id+" and `appointment\_date`='"+appointment\_date+"' ";

NativeQuery nqa = session.createNativeQuery(q1);

Integer token = nqa.list().size()+1;

String sql = "INSERT INTO `appointment` (`id`, `doctor\_id`, `patient\_id`, `amount`, `appointment\_date`,`token`) VALUES (NULL, '"+doctor\_id+"', '"+patient\_id+"', '"+amount+"', '"+appointment\_date+"','"+token+"');";

session.createSQLQuery(sql).executeUpdate();

}

public void adddoctor(String doctorname, String qualification, String speciality, String username,

String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `doctor` (`id`, `doctorname`, `speciality`, `qualification`, `username`, `password`) "

+ "VALUES (NULL, '"+doctorname+"', '"+qualification+"', '"+speciality+"', '"+username+"', '"+password+"');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_doctorlist() {

Session session = sf.getCurrentSession();

String sql = "select id,doctorname,speciality from doctor";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> login(String username, String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select \* from doctor where username='"+username+"' and password='"+password+"'";;

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> get\_appointment\_details\_doctor(Integer id, String date) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select p.firstname,p.lastname,p.mobile,a.appointment\_date,a.token from patient p right join appointment a ON(p.id=a.patient\_id)\r\n" +

" where a.doctor\_id="+id+" and a.appointment\_date='"+date+" ' order by a.token";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

}

**package** com.example.demo.response;

**public** **class** GetProductResponse {

**private** Integer id;

**private** String company;

**private** String model;

**private** Integer price;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getCompany() {

**return** company;

}

**public** **void** setCompany(String company) {

**this**.company = company;

}

**public** String getModel() {

**return** model;

}

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** Integer getPrice() {

**return** price;

}

**public** **void** setPrice(Integer price) {

**this**.price = price;

}

}

public Object[] get\_patient(String mobile) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "Select id,firstname,lastname from patient where mobile='" + mobile + "'";

NativeQuery nq = session.createNativeQuery(sql);

List<Object[]> list = nq.getResultList();

if (list.size() != 0) {

return list.get(0);

} else {

return null;

}

}

public void book\_appointment(Integer doctor\_id, Integer patient\_id, Integer amount, String appointment\_date) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String q1 = "SELECT \* FROM `appointment` WHERE `doctor\_id`="+doctor\_id+" and `appointment\_date`='"+appointment\_date+"' ";

NativeQuery nqa = session.createNativeQuery(q1);

Integer token = nqa.list().size()+1;

**package** com.example.demo.response;

**public** **class** GetAppointmentDetailsResponse {

**private** String firstname;

**private** String lastname;

**private** String mobile;

**private** String appointmentdate;

**private** Integer tokenno;

**public** String getFirstname() {

**return** firstname;

}

**public** **void** setFirstname(String firstname) {

**this**.firstname = firstname;

}

**public** String getLastname() {

**return** lastname;

}

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

**public** String getAppointmentdate() {

**return** appointmentdate;

}

**public** **void** setAppointmentdate(String appointmentdate) {

**this**.appointmentdate = appointmentdate;

}

**public** Integer getTokenno() {

**return** tokenno;

}

**public** **void** setTokenno(Integer tokenno) {

**this**.tokenno = tokenno;

}

}

String sql = "INSERT INTO `appointment` (`id`, `doctor\_id`, `patient\_id`, `amount`, `appointment\_date`,`token`) VALUES (NULL, '"+doctor\_id+"', '"+patient\_id+"', '"+amount+"', '"+appointment\_date+"','"+token+"');";

session.createSQLQuery(sql).executeUpdate();

}

public void adddoctor(String doctorname, String qualification, String speciality, String username,

String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `doctor` (`id`, `doctorname`, `speciality`, `qualification`, `username`, `password`) "

+ "VALUES (NULL, '"+doctorname+"', '"+qualification+"', '"+speciality+"', '"+username+"', '"+password+"');";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_doctorlist() {

Session session = sf.getCurrentSession();

String sql = "select id,doctorname,speciality from doctor";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> login(String username, String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select \* from doctor where username='"+username+"' and password='"+password+"'";;

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> get\_appointment\_details\_doctor(Integer id, String date) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select p.firstname,p.lastname,p.mobile,a.appointment\_date,a.token from patient p right join appointment a ON(p.id=a.patient\_id)\r\n" +

" where a.doctor\_id="+id+" and a.appointment\_date='"+date+" ' order by a.token";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

}

package com.example.demo.service;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.List;

import javax.transaction.Transactional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetAppointmentDetailsResponse;

import com.example.demo.response.GetBillingResponse;

import com.example.demo.response.GetCustomerResponse;

import com.example.demo.response.GetDoctorListResponse;

import com.example.demo.response.GetEmployeeResponse;

import com.example.demo.response.GetPatientResponse;

import com.example.demo.response.GetProductResponse;

import com.example.demo.response.GetStockResponse;

@Service

@Transactional

public class ApiService {

@Autowired

ApiDao dao;

public void add\_employee(String name, String mobile, String address, String gender, Integer salary, Integer age) {

dao.add\_employee(name,mobile,address,gender,salary,age);

}

**package** com.example.demo.response;

**public** **class** GetAppointmentDetailsResponse {

**private** String firstname;

**private** String lastname;

**private** String mobile;

**private** String appointmentdate;

**private** Integer tokenno;

**public** String getFirstname() {

**return** firstname;

}

**public** **void** setFirstname(String firstname) {

**this**.firstname = firstname;

}

**public** String getLastname() {

**return** lastname;

}

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

**public** String getAppointmentdate() {

**return** appointmentdate;

}

**public** **void** setAppointmentdate(String appointmentdate) {

**this**.appointmentdate = appointmentdate;

}

**public** Integer getTokenno() {

**return** tokenno;

}

**public** **void** setTokenno(Integer tokenno) {

**this**.tokenno = tokenno;

}

}

public void add\_customer(String name, String mobile, String address, String gender, String email) {

// TODO Auto-generated method stub

dao.add\_customer(name,mobile,address,gender,email);

}

public List<GetEmployeeResponse> get\_employee() {

// TODO Auto-generated method stub

List<Object[]> result =dao.get\_employee();

List<GetEmployeeResponse> response = new ArrayList<GetEmployeeResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetEmployeeResponse obj = new GetEmployeeResponse();

obj.setId((Integer)row[0]);

obj.setName((String)row[1]);

obj.setMobile((String)row[2]);

obj.setAddress((String)row[3]);

obj.setGender((String)row[4]);

obj.setSalary((Integer)row[5]);

obj.setAge((Integer)row[6]);

response.add(obj);

}

return response;

}

public List<GetCustomerResponse> get\_customer() {

// TODO Auto-generated method stub

List<Object[]> result =dao.get\_customer();

List<GetCustomerResponse> response = new ArrayList<GetCustomerResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetCustomerResponse obj = new GetCustomerResponse();

obj.setId((Integer)row[0]);

obj.setName((String)row[1]);

obj.setMobile((String)row[2]);

obj.setAddress((String)row[3]);

obj.setGender((String)row[4]);

obj.setEmail((String)row[5]);

response.add(obj);

}

return response;

}

public void add\_product(String company, String model, Integer price) {

// TODO Auto-generated method stub

dao.add\_product(company,model,price);

}

public List<GetProductResponse> get\_product() {

// TODO Auto-generated method stub

List<Object[]> result =dao.get\_product();

List<GetProductResponse> response = new ArrayList<GetProductResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetProductResponse obj = new GetProductResponse();

obj.setId((Integer)row[0]);

obj.setCompany((String)row[1]);

obj.setModel((String)row[2]);

obj.setPrice((Integer)row[3]);

response.add(obj);

}

return response;

}

public void add\_purchase(Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

dao.add\_purchase(product\_id,quantity);

}

public void add\_sales(Integer customer\_id, Integer product\_id, Integer quantity) {

// TODO Auto-generated method stub

dao.add\_sales(customer\_id,product\_id,quantity);

}

public List<GetStockResponse> get\_stock() {

List<Object[]> result =dao.get\_stock();

List<GetStockResponse> response = new ArrayList<GetStockResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetStockResponse obj = new GetStockResponse();

obj.setCompany\_name((String)row[0]+"-"+(String)row[1]);

obj.setQuantity((BigDecimal)row[2]);

response.add(obj);

}

return response;

}

public Integer get\_mobile(String mobile) {

// TODO Auto-generated method stub

return dao.get\_mobile(mobile);

}

public List<GetBillingResponse> get\_billing() {

List<Object[]> result =dao.get\_billing();

List<GetBillingResponse> response = new ArrayList<GetBillingResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetBillingResponse obj = new GetBillingResponse();

obj.setCustomer\_name((String)row[0]);

obj.setMobile((String)row[1]);

obj.setCompany((String)row[2]);

obj.setModel((String)row[3]);

obj.setQuantity((Integer)row[4]);

obj.setPrice((Integer)row[5]);

response.add(obj);

}

return response;

}

public List<Object[]> login(String username, String password) {

return dao.login(username,password);

}

public void add\_patient(String firstname, String lastname, String mobile, String email, String attendar,

String emergency) {

dao.add\_patient(firstname,lastname,mobile,email,attendar,emergency);

}**package** com.example.demo.response;

**public** **class** GetAppointmentDetailsResponse {

**private** String firstname;

**private** String lastname;

**private** String mobile;

**private** String appointmentdate;

**private** Integer tokenno;

**public** String getFirstname() {

**return** firstname;

}

**public** **void** setFirstname(String firstname) {

**this**.firstname = firstname;

}

**public** String getLastname() {

**return** lastname;

}

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;

}

**public** String getMobile() {

**return** mobile;

}

**public** **void** setMobile(String mobile) {

**this**.mobile = mobile;

}

**public** String getAppointmentdate() {

**return** appointmentdate;

}

**public** **void** setAppointmentdate(String appointmentdate) {

**this**.appointmentdate = appointmentdate;

}

**public** Integer getTokenno() {

**return** tokenno;

}

**public** **void** setTokenno(Integer tokenno) {

**this**.tokenno = tokenno;

}

}

public GetPatientResponse get\_patient(String mobile) {

// TODO Auto-generated method stub

Object[] obj= dao.get\_patient(mobile);

GetPatientResponse res = new GetPatientResponse();

res.setFirstname((String)obj[1]);

res.setId((Integer)obj[0]);

res.setLastname((String)obj[2]);

return res;

}

public void book\_appointment(Integer doctor\_id, Integer patient\_id, Integer amount, String appointment\_date) {

dao.book\_appointment(doctor\_id,patient\_id,amount,appointment\_date);

}

public void adddoctor(String doctorname, String qualification, String speciality, String username,

String password) {

// TODO Auto-generated method stub

dao.adddoctor(doctorname,qualification,speciality,username,password);

}

public List<GetDoctorListResponse> get\_doctorlist() {

List<Object[]> result =dao.get\_doctorlist();

List<GetDoctorListResponse> response = new ArrayList<GetDoctorListResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetDoctorListResponse obj = new GetDoctorListResponse();

obj.setId((Integer)row[0]);

obj.setName((String)row[1]);

obj.setSpeciality((String)row[2]);

response.add(obj);

}

return response;

}

public List<GetAppointmentDetailsResponse> get\_appointment\_details\_doctor(Integer id, String date) {

// TODO Auto-generated method stub

List<Object[]> result =dao.get\_appointment\_details\_doctor(id,date);

List<GetAppointmentDetailsResponse> response = new ArrayList<GetAppointmentDetailsResponse>();

for(int i=0;i<result.size();i++) {

Object[] row = result.get(i);

GetAppointmentDetailsResponse obj = new GetAppointmentDetailsResponse();

obj.setFirstname((String)row[0]);

obj.setLastname((String)row[1]);

obj.setMobile((String)row[2]);

obj.setTokenno((Integer)row[4]);

obj.setAppointmentdate((String)row[3]);

response.add(obj);

}

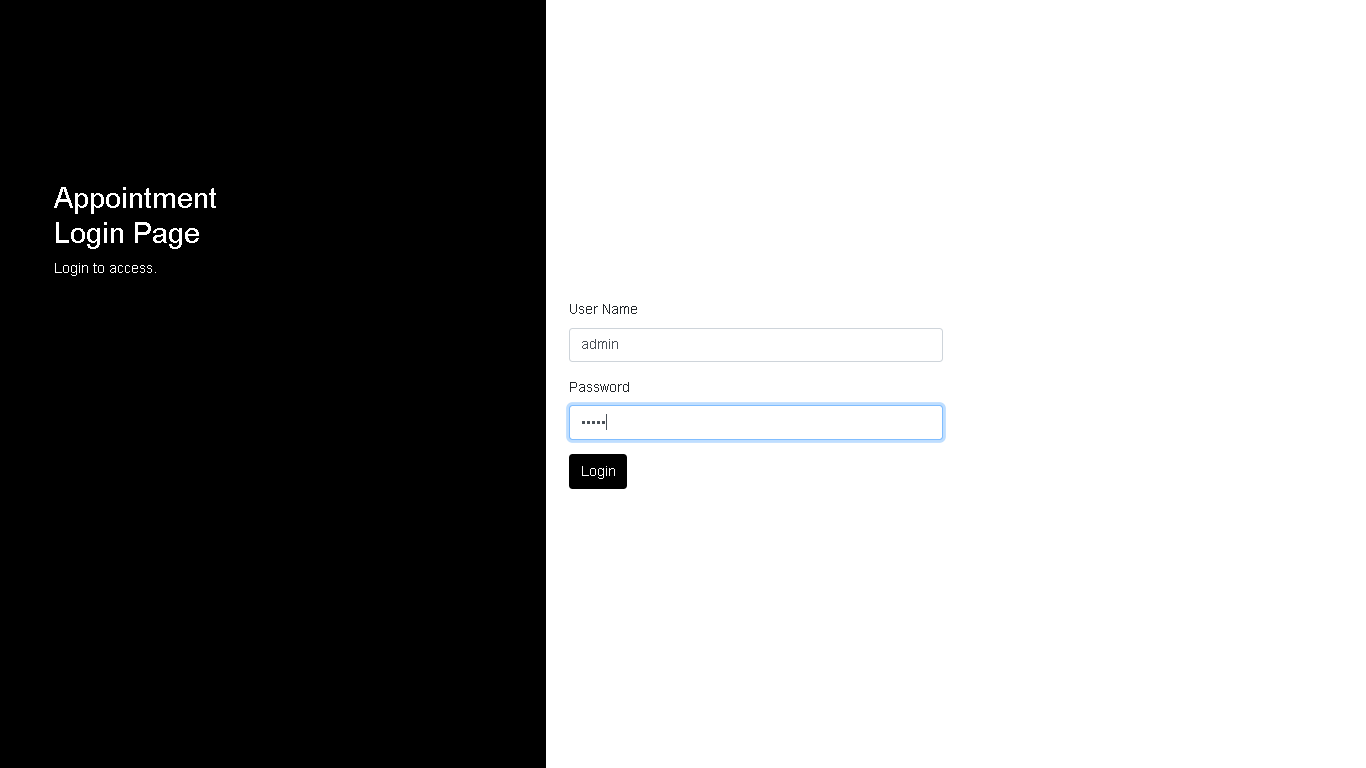
return response;

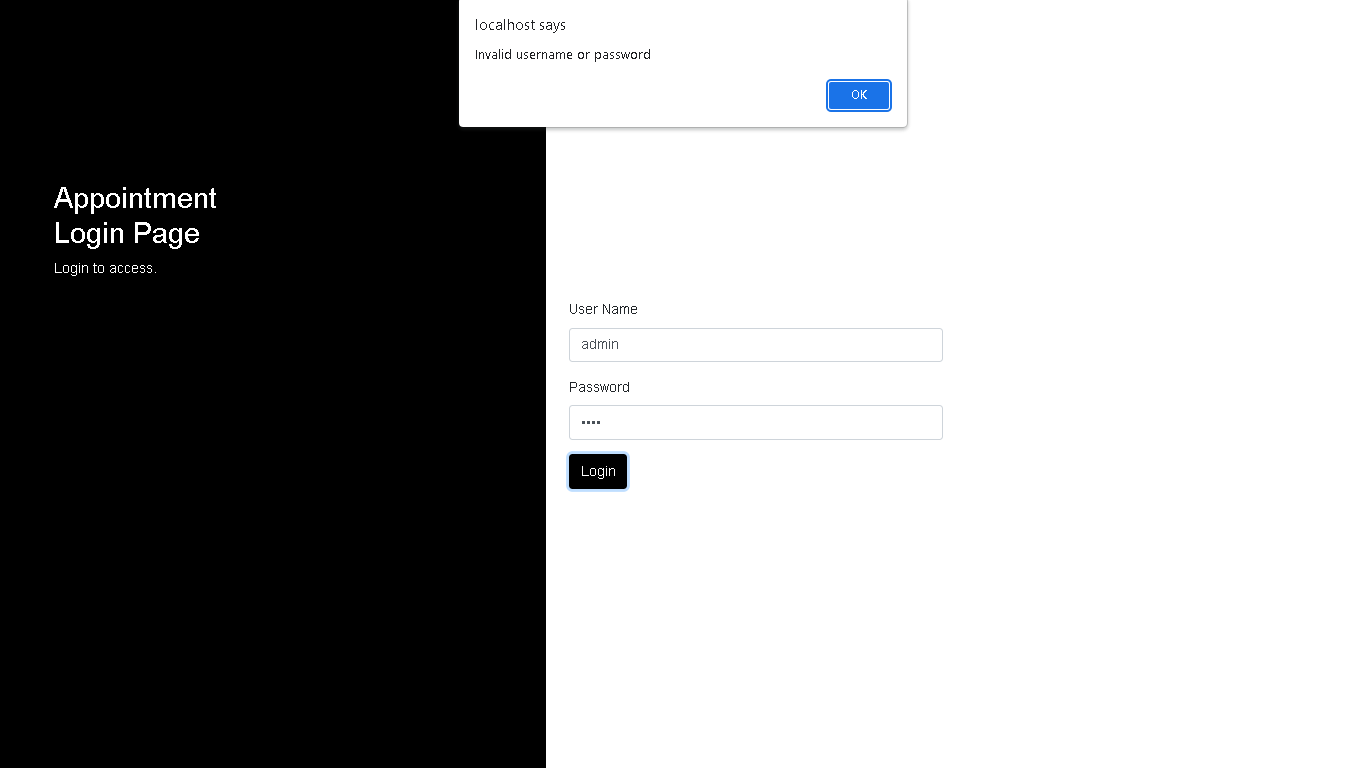
}

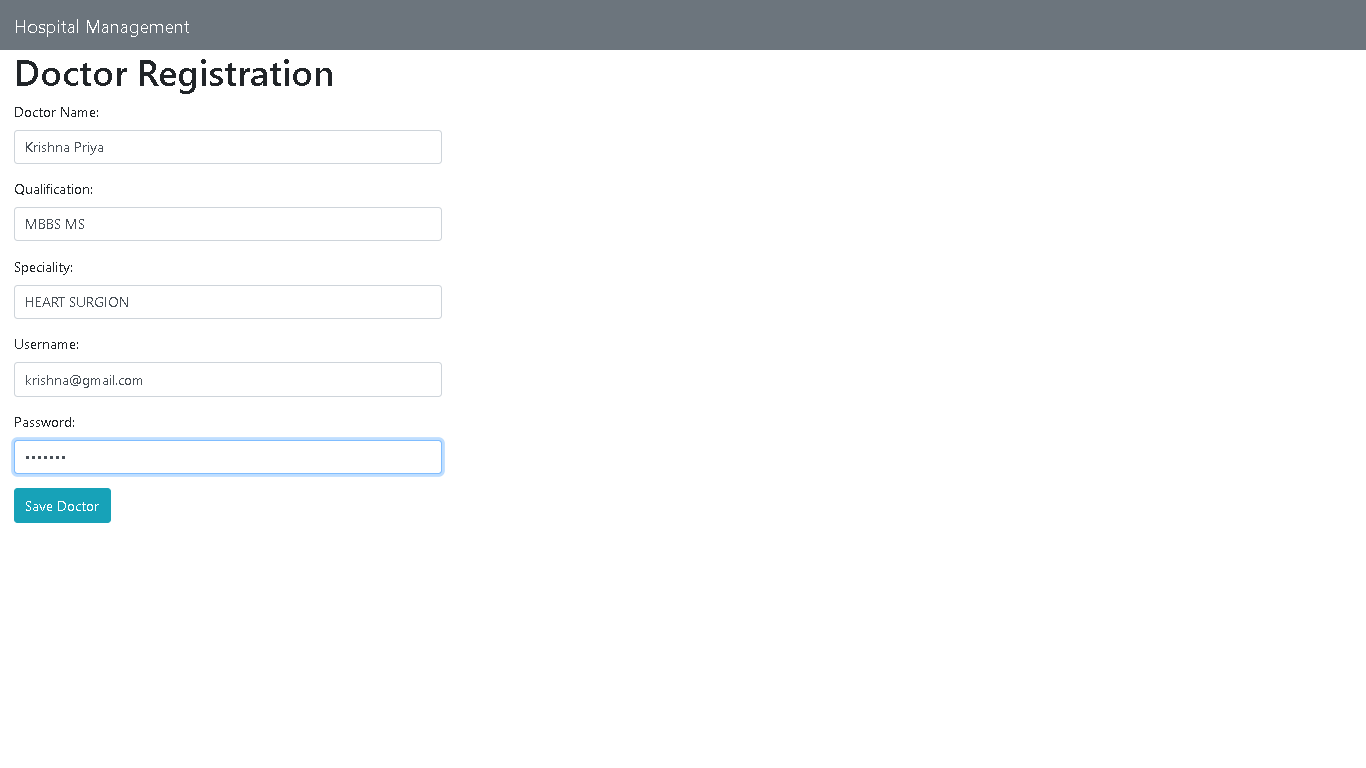
}

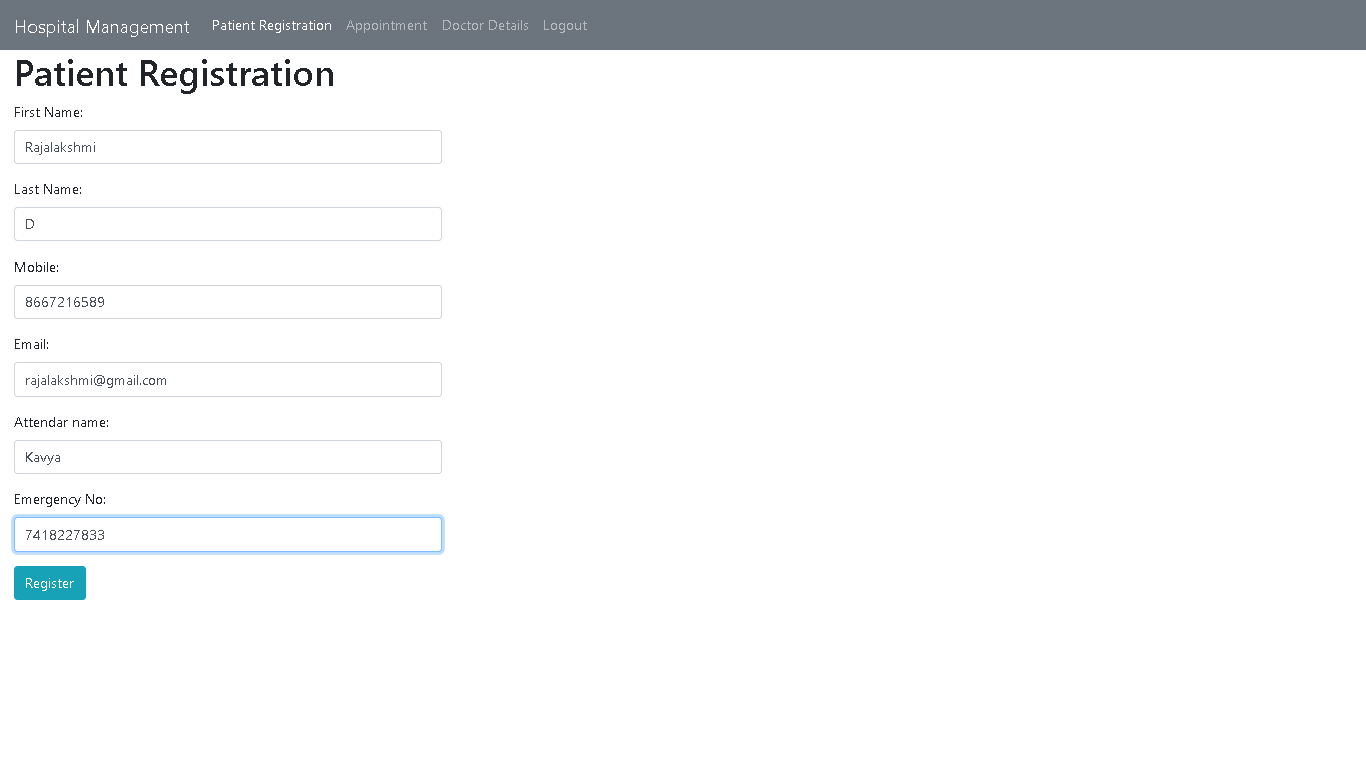
## APPENDIX II

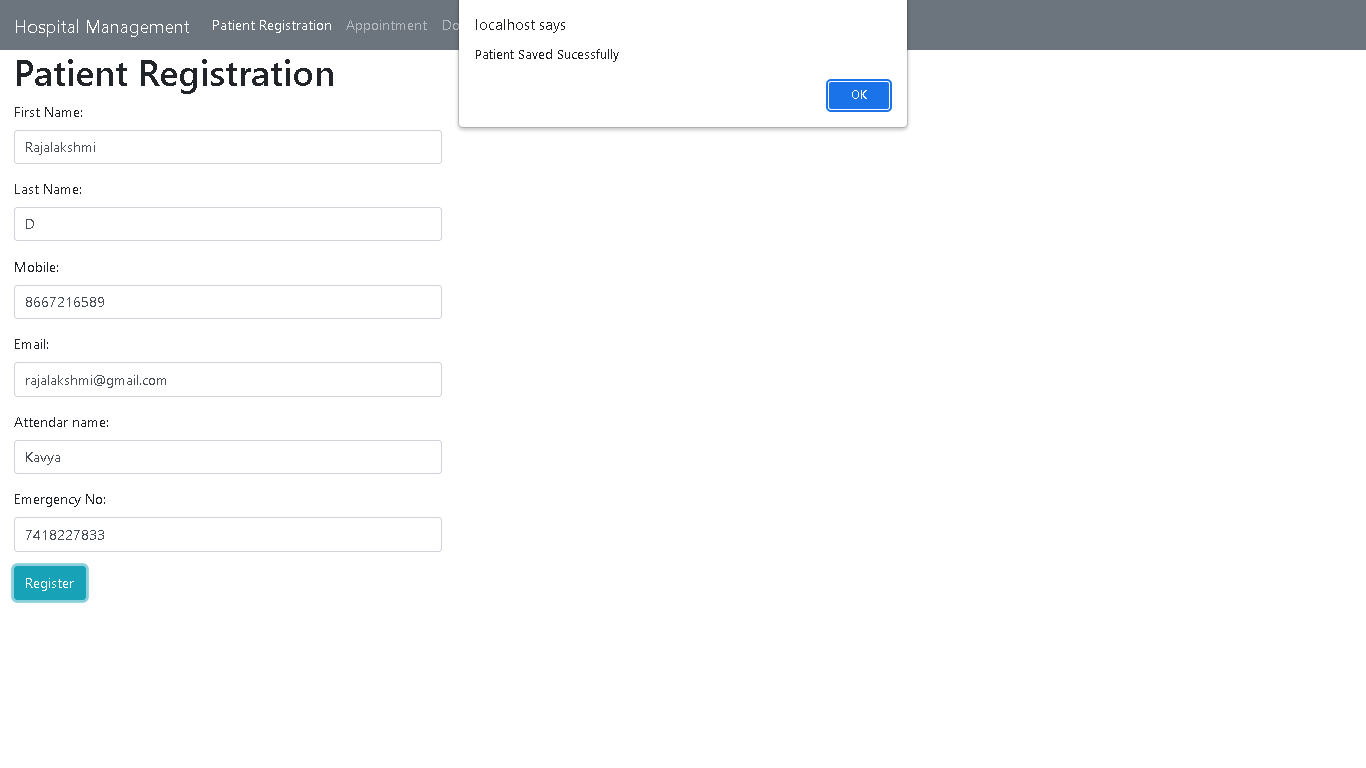
**INPUT/OUTPUT DESIGN**

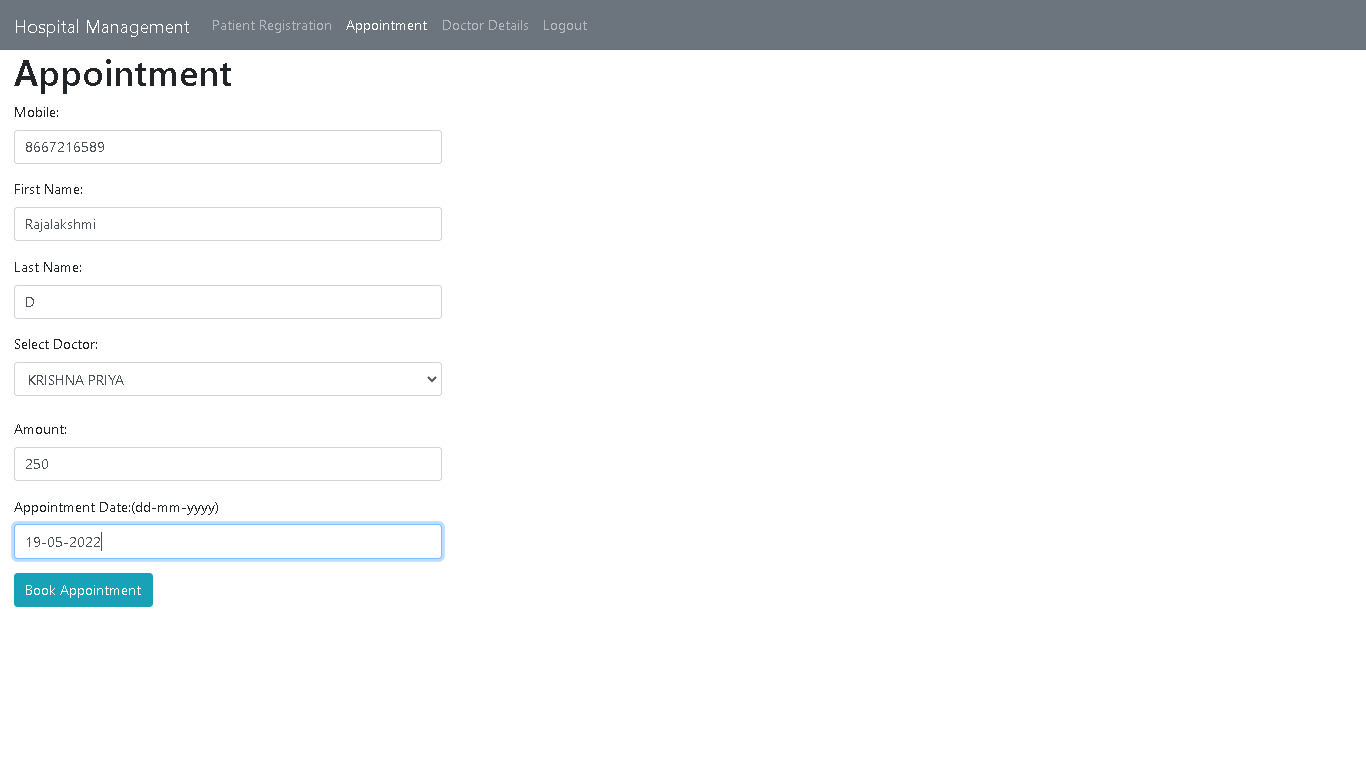


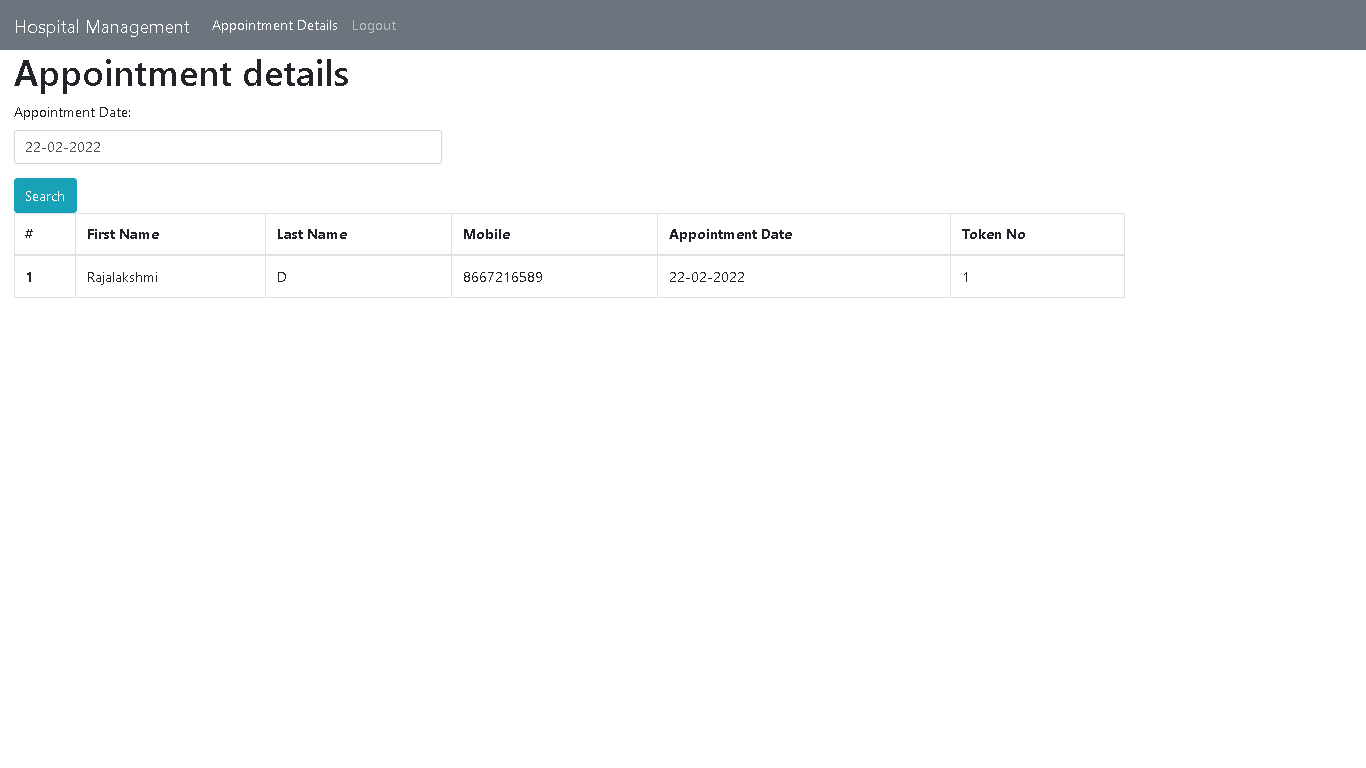


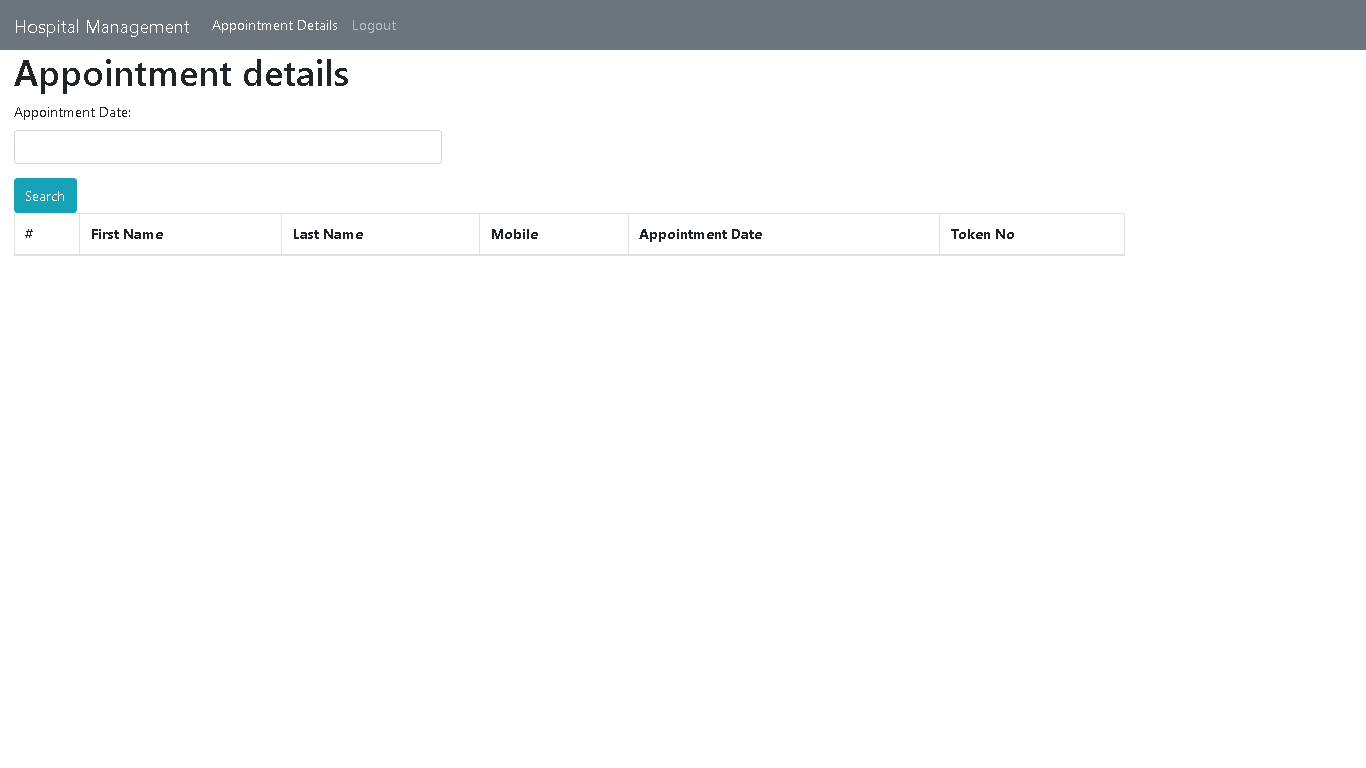












**REFERENCES**

**BOOK REFERENCES**

1. Jeff Forcier, Paul Bissex, Wesley J. Chun, (2018) “Python Web development with Django (Developer’s Library)”, WesleyProfessional.
2. Roger S. Pressman, (2017) “Software Engineering”, Tata McGraw-Hill –Europe.

## WEB REFERENCES

1. https://[www.javaprogramming.net](http://www.javaprogramming.net)
2. https://[www.w3schools.com](http://www.w3schools.com/)
3. https://[www.tutorialspoint.com](http://www.tutorialspoint.com/)