# INTRODUCTION TO PROJECT

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

computerize all details regarding patient details and hospital details.

The system will be used as the application that serves hospitals, clinic, dispensaries or

other health institutions. The intention of the system is to increase the number of

patients that can be treated and managed properly.

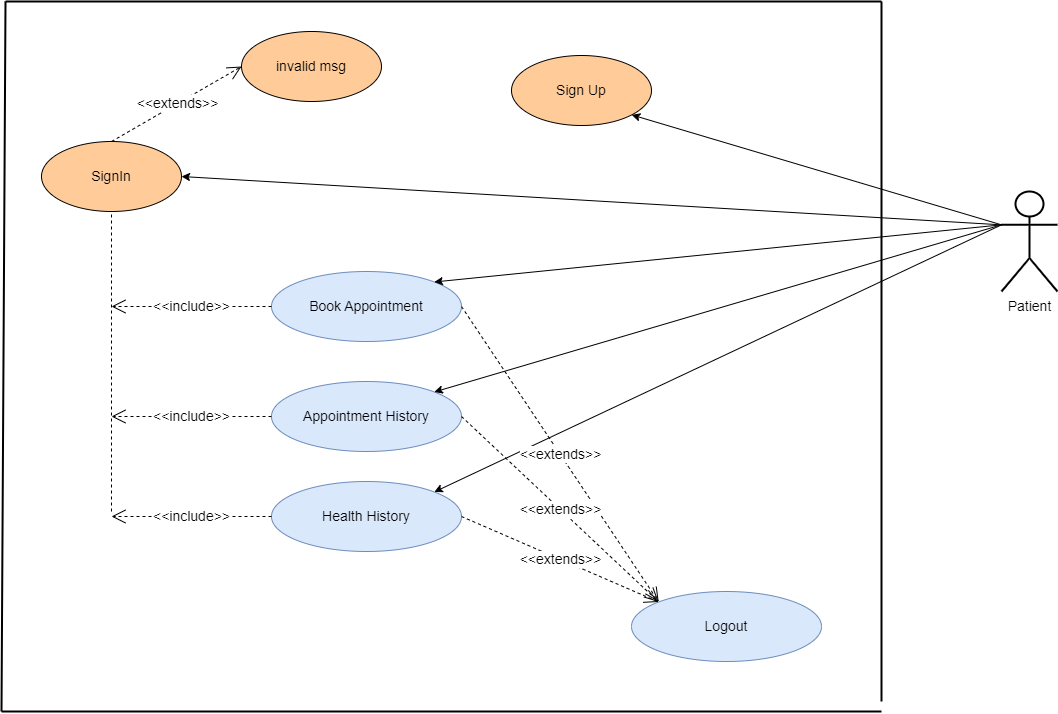
If the hospital management system is file based, management of the hospital has to put

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.

**2.REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**



**2.1.1 User Account**

Patients can choose the best preferred appointments from the options provided and can also change the appointment schedule or cancel it. After appt. is confirmed by the respective doctor they can pay their consultant fee online. Patients have access to only

their records.

Key functions:

• Make appointment.

• Cancel appointment.

• Update Details.

• Payment.

**2.1.2 Registration and creation of user profile**

DESCRIPTION - The new patient can register themselves and add their details like name, age, gender, blood group etc. The patient entry will be made in the hms database.

PRE -CONDITION – The patient must be a new patient, If necessary fields left by user then prompt user to fill the necessary fields.

MAIN FLOW OF EVENTS

1. Patient selects sign up in login module.

2. A registration form get displayed

3. Patient fills the required details. POST CONDITIONS - Patient record is added to hms database.

**2.1.3 Doctor**

DESCRIPTION- The doctor view patient record/ update his details and add description of the treatment given to patient.

PRE-CONDITION – The doctor must be a registered doctor, System does not allow the

doctor to modify the qualification, hospital managed details.

MAIN FLOW OF EVENTS

1.Doctor logs in to the system.

2. Doctor may select view patient.

2.1 Patient record is displayed with treatment history.

3. Doctor add description of patient treatment.

4. Doctor may select appointment details

4.1 Appointment Requests is displayed with schedule.

5. Doctor confirm or cancel appointment.

POST CONDITION – The patient and doctor ‘s database are updated.

**2.1.4. Admin**

DESCRIPTION - The admin add doctor, update docotr details and verify payment and

generate Bill/Reciept for the same.

MAIN FLOW OF EVENTS

1. Admin logs in the system.

2. Admin may add doctor new doctor.

2.1 admin fills the doctor’s details.

3. Admin view Doctor record.

3.1 Admin enters the doctor id in the system.

3.2 Doctor details are displayed, Admin can update details.

4. Admin Verify the payment submited by the Patient.

4.1 Generate Bill/Reciept and confirmation message for the same.

PRE –CONDITION - Admin must first log in with his/her credentials.

POST CONDITION - The hms database is updated.

28 | P a g e

**2.1.5. User characteristics**

**ADMIN**

Admin has the full access to the system which means he is able to manage any activity

with regard to the system. He is the highest privileged user who can access to the

system.

Key functions:

•Access patient record, doctor Record.

•Add new doctor entry in system database.

• Confirm Payment and Generate Bill.

• View Records.(Total no of patients treated, doctor added/remove, consultant fee).

**PATIENT**

Patients can choose the best preferred appointments from the options provided and

can also change the appointment schedule or cancel it. After appt. is confirmed by the

respective doctor they can pay their consultant fee online. Patients have access to only

their records.

Key functions:

• Make appointment.

• Cancel appointment.

• Update Details.

• Payment.

•View Payment History.

**DOCTOR**

Doctors can view the patient appointment list and provide the confirmation or make

changes in the appointment list if required. Doctors have access to only records of those

patients whom they are treating.

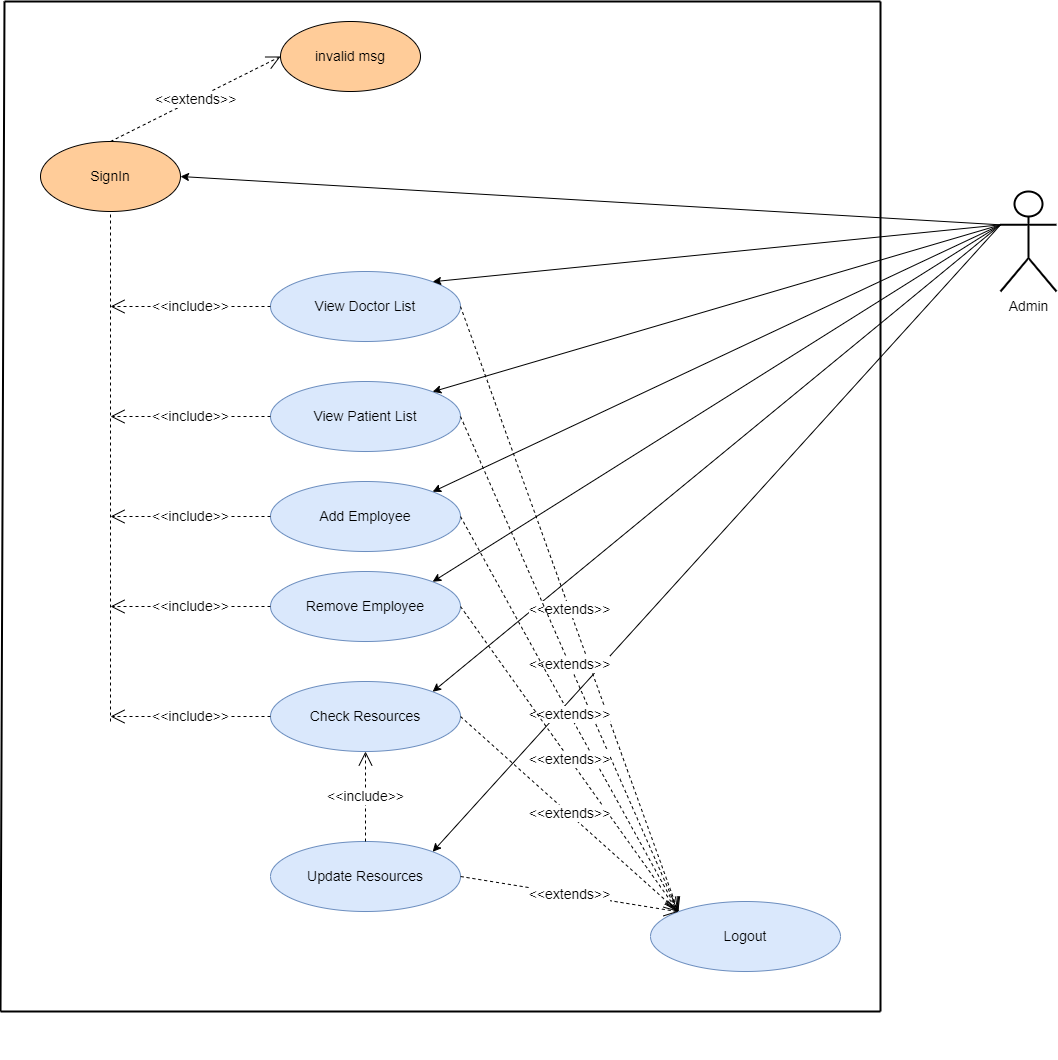
Key functions:

• Confirmation of appointment.

• Cancellation of appointment.

• Modification of appointment list.

• Add Prescription.



Admin should be able to login, add Employee information, remove Employee information, view doctor list, check resources, Update resources and see user Information according to user Id.

* 1. **NON-FUNCTIONAL REQUIREMENTS**

**2.2.1 Interface**

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and

communication interfaces and provides basic prototypes of the user interface.

▪ The protocol used shall be HTTP.

▪ The Port number used will be 8080.

▪ There shall be logical address of the system in IPv4 format.

**2.2.2 Performance**

* **Number of Concurrent Users:**

ARS shall be able to handle at least 1000 transactions/inquiries per

second

* **Registration of Patients:**

The system is susceptible to any temporary server failure since it uses the strong feature of Struts 2 and Hibernate. Hence the examination will be continued even if the sever gets disconnected in between the examination.

**2.2.3 Constraint**

ARS shall be able to handle at least 1000 transactions/inquiries persecond

* + 1. **Hardware Requirements:**
* Laptop/Desktop PC-Purpose of this is to give information when Patients ask information about doctors, medicine available lab tests etc. To perform such Action, it needs very efficient computer otherwise due to that reason patients have to wait for a long time to get what they ask for.
* Laser Printer (B/W) - This device is for printing patients’ info etc. Software Interfaces

**2.2.5 Software Requirements:**

**▪ JDK 1.8** - Java is fast, secure, and reliable. From laptops to data centers, game

consoles to scientific supercomputers, cell phones to the Internet.

**▪ Mysql server -** Database connectivity and management

**▪ OS Windows 7/8/8.1-** Very user friendly and common OS

**▪ JRE 1.8 -** JAVA Runtime Environment for run Java Application and System

**3. DESIGN**

**3.1 Database Design**

The following table structures depict the database design.

# Table1: Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key Type/ Constraint** | **Column Name** | **Data Type** | **Length** | **Allow Null (1=Yes;0=No)** |
| 3 | User\_id | INT | 4 | 0 |
| 0 | Blood\_group | Varchar2 | 10 | 1 |
| 0 | Dob | Date | 255 | 1 |
| 0 | email | Varchar2 | 45 | 1 |
| 0 | First\_name | Varchar2 | 45 | 1 |
| 0 | gender | Varchar2 | 20 | 1 |
| 0 | Last\_name | Varchar2 | 45 | 1 |
| 0 | Mob\_no | Varchar2 | 15 | 1 |
| 0 | Password | Varchar2 | 10 | 1 |
| 0 | Security\_ans | Varchar2 | 100 | 1 |
| 0 | Security\_ans | Varchar2 | 100 | 1 |

## **Table2: Employee**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Employee\_id | INT | 15 | 0 |
| 0 | Hire\_date | Date | 15 | 1 |
| 0 | Qualification | Varchar2 | 100 | 0 |
| 0 | Role | Varchar2 | 250 | 0 |
| 0 | Salary | Double | 15 | 1 |
| 0 | Status | Tiny\_int | 1 | 1 |
| 0 | User\_id | Int | 4 | 0 |

# Table3: Patient

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Patient\_id | Int | 5 | 0 |
| 0 | Action | Tiny\_int | 1 | 1 |
| 0 | Admit\_status | Tiny\_int | 1 | 1 |
| 0 | Allocated\_bed | Varchar2 | 45 | 1 |
| 0 | Current\_status | Tiny\_int | 1 | 1 |
| 0 | User\_id | Int | 1 | 0 |
| 0 | Ward\_id | Int | 1 | 0 |
| 0 | Doctor\_id | Int | 1 | 0 |

### Table4: Health History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Healthhistory\_id | Int | 5 | 0 |
| 3 | admit\_date | Date | 10 | 0 |
| 0 | appointment\_date | Date | 10 | 1 |
| 0 | appointment\_time | Time | 10 | 1 |
| 3 | discharge\_date | Date | 10 | 0 |
| 0 | diseases | Varchar | 300 | 1 |
| 0 | payment\_date | Date | 10 | 1 |
| 0 | prescription\_instructon | Varchar | 1000 | 1 |
| 0 | Symptoms | Varchar | 1000 | 1 |
| 0 | patient\_id | Int | 3 | 1 |
| 0 | allocated\_bed | Varchar | 45 | 1 |
| 0 | paid\_amount | Double | 10 | 1 |
| 0 | payment\_status | Tinyint | 1 | 1 |

**Table5: Doctor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Doctor\_id | Int | 5 | 0 |
| 0 | Doctor\_fee | Double | 10 | 1 |
| 3 | scheduled\_end\_time | Time | 15 | 1 |
| 0 | scheduled\_start\_time | Time | 4 | 1 |
| 0 | Employee\_id | int | 5 | 0 |

### Table6:Address

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Address\_id | Int | 5 | 0 |
| 0 | area\_name | Varchar | 45 | 1 |
| 0 | building\_name | Varchar | 45 | 1 |
| 0 | City | Varchar | 45 | 1 |
| 0 | Country | Varchar | 45 | 1 |
| 0 | pincode | Int | 8 | 1 |
| 0 | plot\_no | Varchar | 45 | 1 |
| 0 | state | Varchar | 45 | 1 |
| 0 | User\_id | Int | 5 | 0 |

**Table7: Medicines**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Medicines \_id | Int | 5 | 0 |
| 0 | Duration | Varchar | 255 | 1 |
| 3 | medicine\_charges | Double | 15 | 1 |
| 0 | quantity | Int | 10 | 1 |
| 0 | Health\_history\_id | int | 5 | 0 |

**Table8: Resources**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Resources \_id | Int | 5 | 0 |
| 0 | Occupy\_quantity | Int | 10 | 1 |
| 3 | Remaining\_quantity | Int | 10 | 1 |
| 0 | Resource\_name | Varchar | 45 | 1 |
| 0 | Total\_quantity | Int | 5 | 0 |

**Table9: Ward**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Ward \_id | Int | 5 | 0 |
| 0 | Ward\_charges | Double | 10 | 1 |
| 3 | Ward\_type | Varchar | 45 | 1 |

**Table9: Role**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Id | Int | 5 | 0 |
| 0 | Name | Varchar | 255 | 1 |

**Table10: User role**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | User | Int | 5 | 0 |
| 0 | Role | Int | 5 | 0 |

**E-R Diagram,Dataflow diagram and Class Diagram:**

Go to Appendix A

**4. CODING STANDARDS IMPLEMENTED**

### Naming and Capitalization

Below summarizes the naming recommendations for identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances.

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Case** | **Examples** | **Additional Notes** |
| Class | Pascal | Person, BankVault, SMSMessage, Dept | Class names should be based on "objects" or "real things" and should generally be **nouns**. No ‘\_’ signs allowed. Do not use type prefixes like ‘C’ for class. |
| Method | Camel | getDetails, updateStore | Methods should use **verbs** or verb phrases. |
| Parameter | Camel | personName, bankCode | Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios. |
| Interface | Pascal with "I" prefix | Disposable | Do not use the ‘\_’ sign |
| Property | Pascal | ForeColor, BackColor | Use a noun or noun phrase to name properties. |
| Associated private member variable | \_camelCase | \_foreColor, \_backColor | Use underscore camel casing for the private member variables |
| Exception Class | Pascal with "Exception" suffix | WebException, |  |

### Comments

* Comment each type, each non-public type member, and each region declaration.
* Use end-line comments only on variable declaration lines. End-line comments are comments that follow code on a single line.
* Separate comments from comment delimiters (apostrophe) or // with one space.
* Begin the comment text with an uppercase letter.
* End the comment with a period.
* Explain the code; do not repeat it.

**5. TEST REPORT**

**Another group called Linux did the testing and the report of the testing is given hereunder.**

**GENERAL TESTING:**

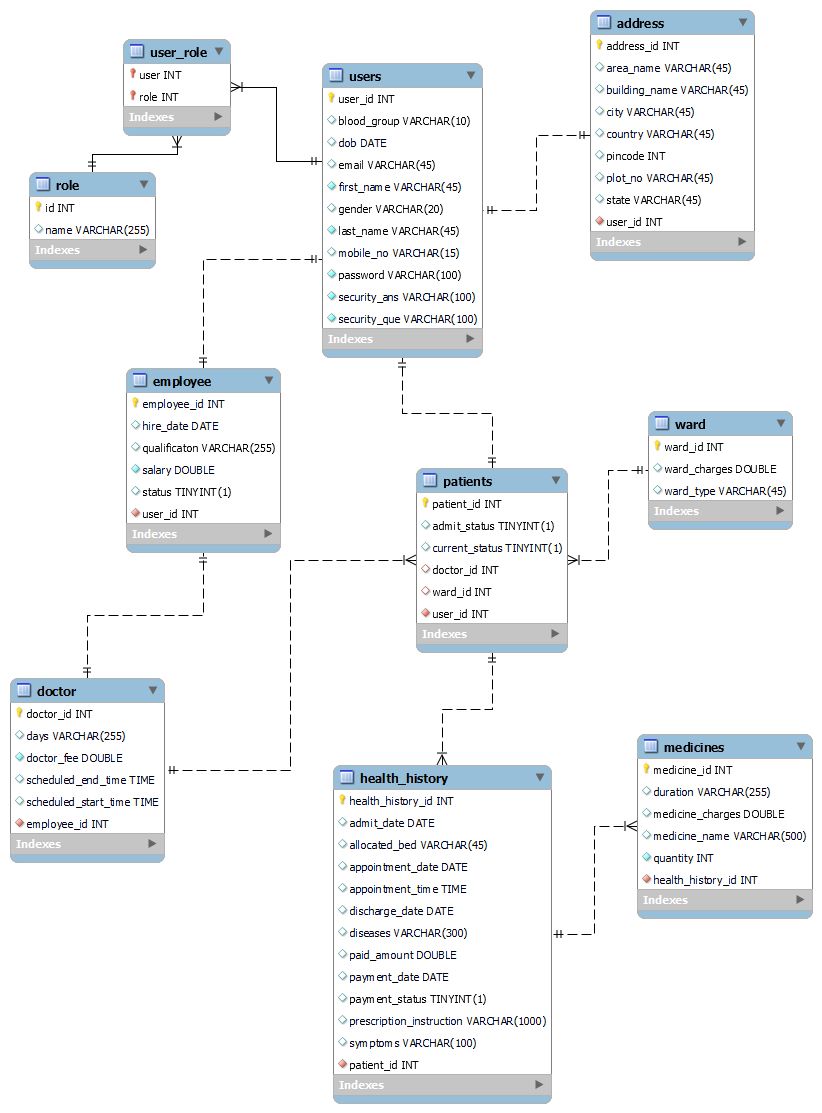
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR-NO** | **TEST CASE** | **EXPECTED RESULT** | **ACTUAL RESULT** | **ERROR MESSAGE** |
| 1 | Register Page | Redirected to Next page | OK | Nothing |
| 2 | Login Page | Pop-up will come | Ok | Please enter username and password again . |
| 3 | Reset login | Only users password will be reseted | Ok | Nothing |
| 4 | Quick search patient | Gives all flight details | Ok | Nothing |
| 5 | Booking Appointment | All the fields should be filled for submission | Ok | Nothing |
| 6 | Checking login or not | User is logged in or not | Ok | Nothing |
| 7 | Add person details for Appointment | Add informations according to no of seats allocated | Ok | Nothing |
| 8 | Go to health history | Set added information about person | Ok | Nothing |
| 9 | Logout | It will logout from user profile. | Ok | Nothing |
|  | **STATIC TESTING** |  |  |  |
| **SR-NO** | **Deviation** | **Program** |  |  |
| 1 | Commenting not followed | All Web Application |  |  |

**6. PROJECT MANAGEMENT RELATED STATISTICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **WORK PERFORMED** | ****SLC Phase**** | **Additional Notes** |
| OCT 15 2022 | Project Allotment and User Requirements Gathering | Feasibility Study | Our team met the client Mr. Nitinkudale (CEO, SIIT Pune) to know his requirements. |
| OCT 22 2022 | Initial SRS Document Validation  And Team Structure Decided | Requirement Analysis  (Elicitation) | The initial SRS was presented to the client to understand his requirements better |
| JAN 18,2015 | Designing the use-cases, Class Diagram, Collaboration Diagram, E-R Diagram and User Interfaces | Requirement Analysis &  Design Phase | Database Design completed |
| JAN 19,2015 | Business Logic Component design Started | Design Phase | ---------------------- |
| JAN 20,2015 | Coding Phase Started | Coding Phase | 70% of Class Library implemented. |
| JAN 21,2015 | Implementation of Web Application and Window Application Started | Coding Phase | Class Library Development going on. |
| JAN 22, 2015 | Off | Off | Off |
| JAN 23, 2015 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | Class Library Modified as per the need. |
| JAN 24, 2015 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | -- |
| JAN 25, 2015 | After Ensuring Proper Functioning the Required Validations were Implemented | Coding Phase and Unit Testing | Module Integration was done by the Project Manager |
| JAN 26, 2015 | The Project was Tested by the respective Team Leaders and the Project Manager | Testing Phase (Module Testing) | -- |
| JAN 27, 2015 | The Project was Submitted to Other Project Leader of Other Project Group For Testing | Testing Phase (Acceptance Testing) | The Project of Other Team was Taken up by the Team for Testing |
| JAN 28-29, 2015 | The Errors Found were Removed | Debugging | The Project was complete for submission |
| JAN 30, 2015 | Final Submission of Project |  |  |

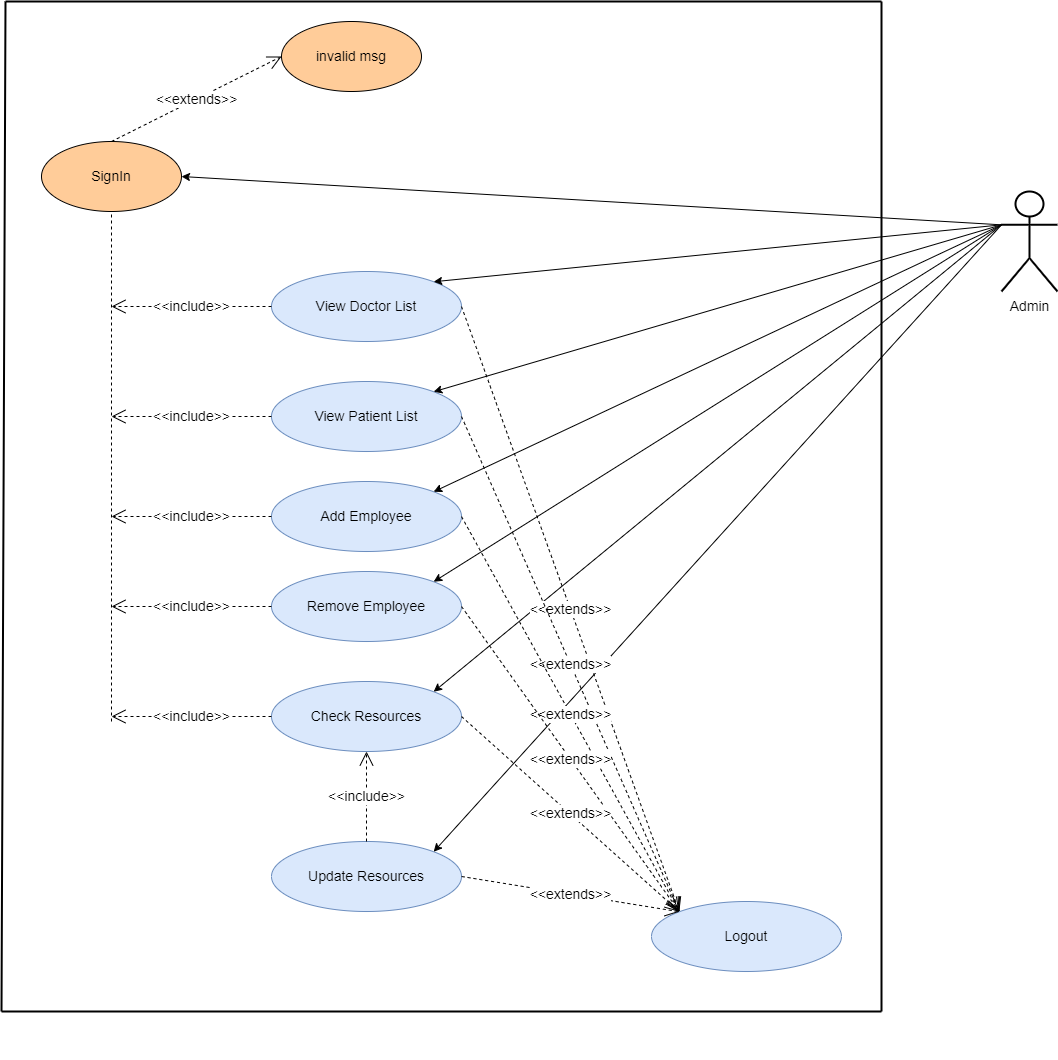
Appendix A

Entity Relationship Diagram

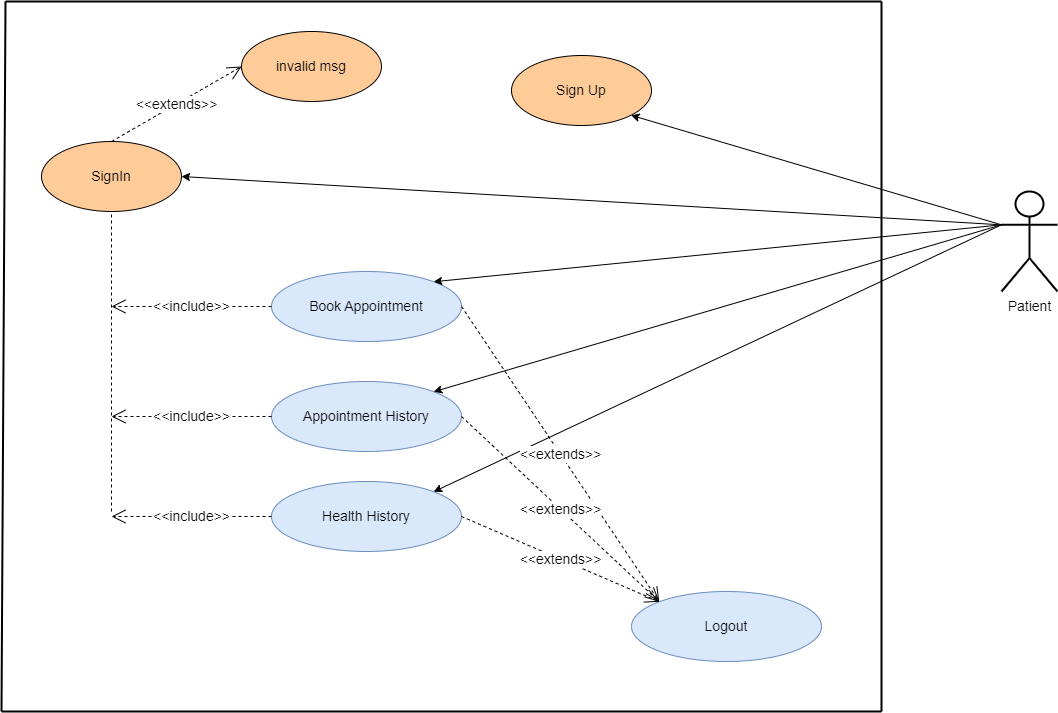


**Use Case Diagram**

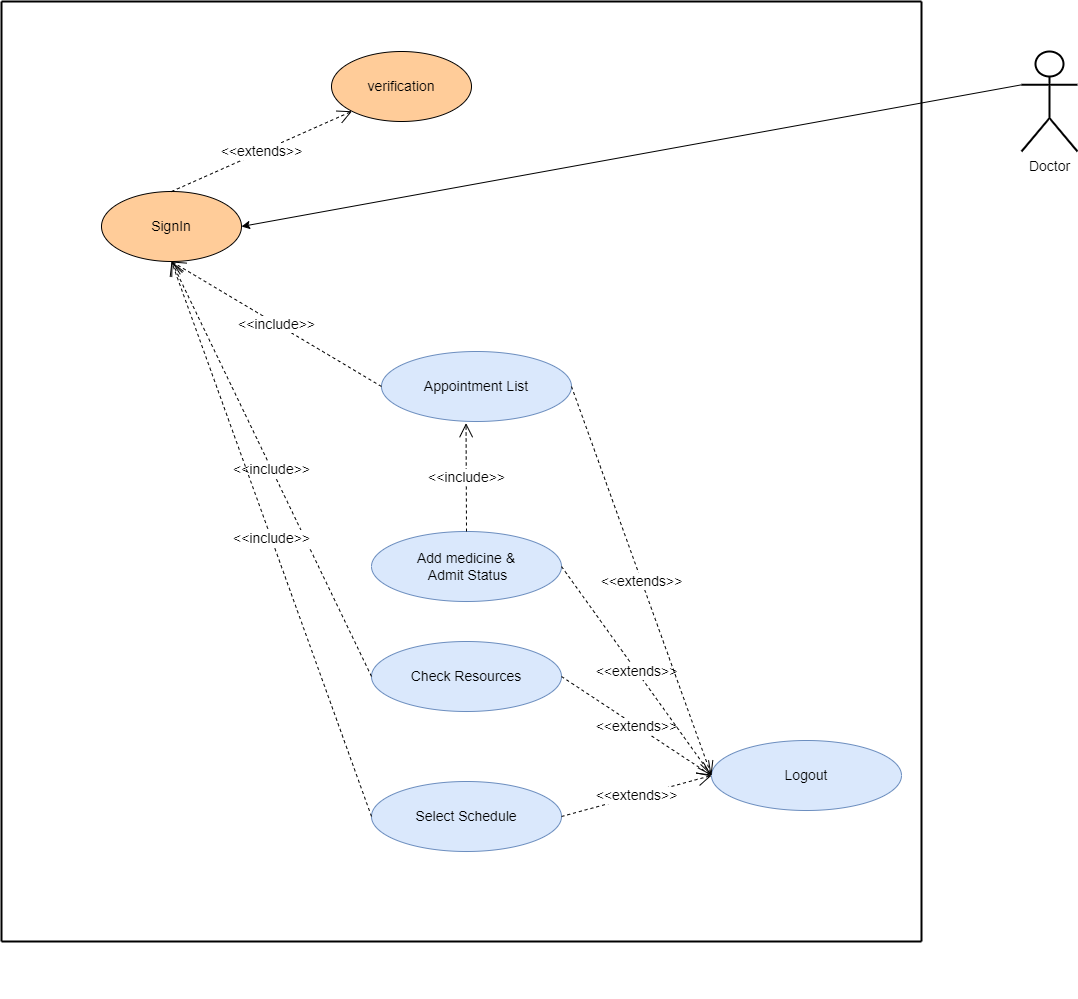
**Admin:**



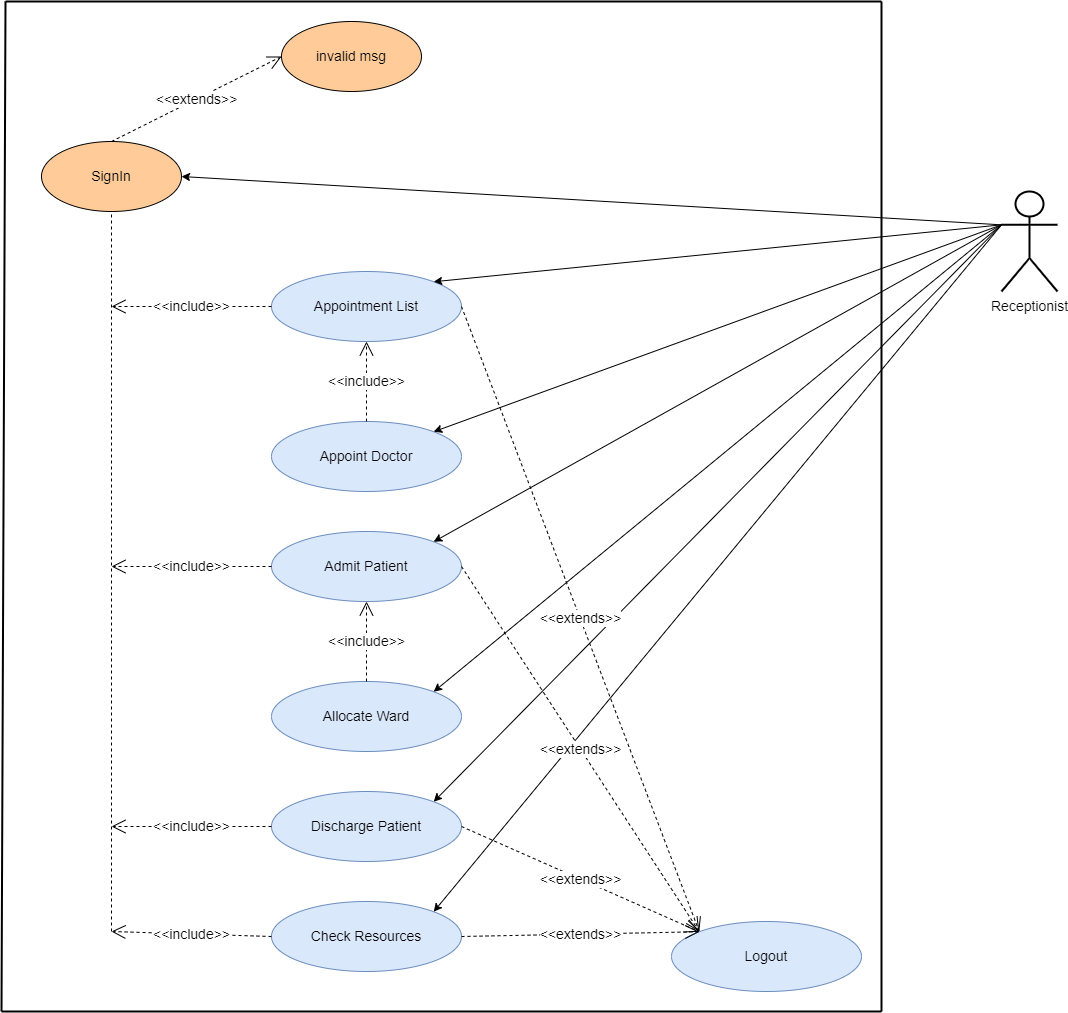
**Patient:**



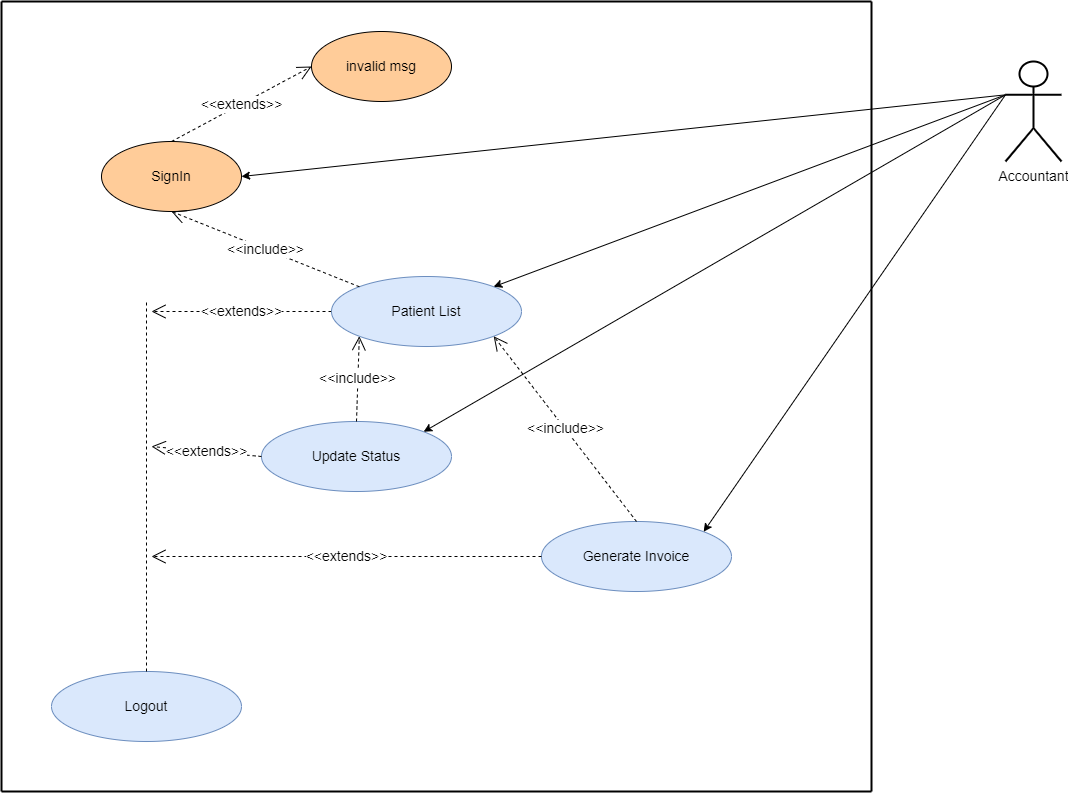
**Doctor:**



**Receptionist:**

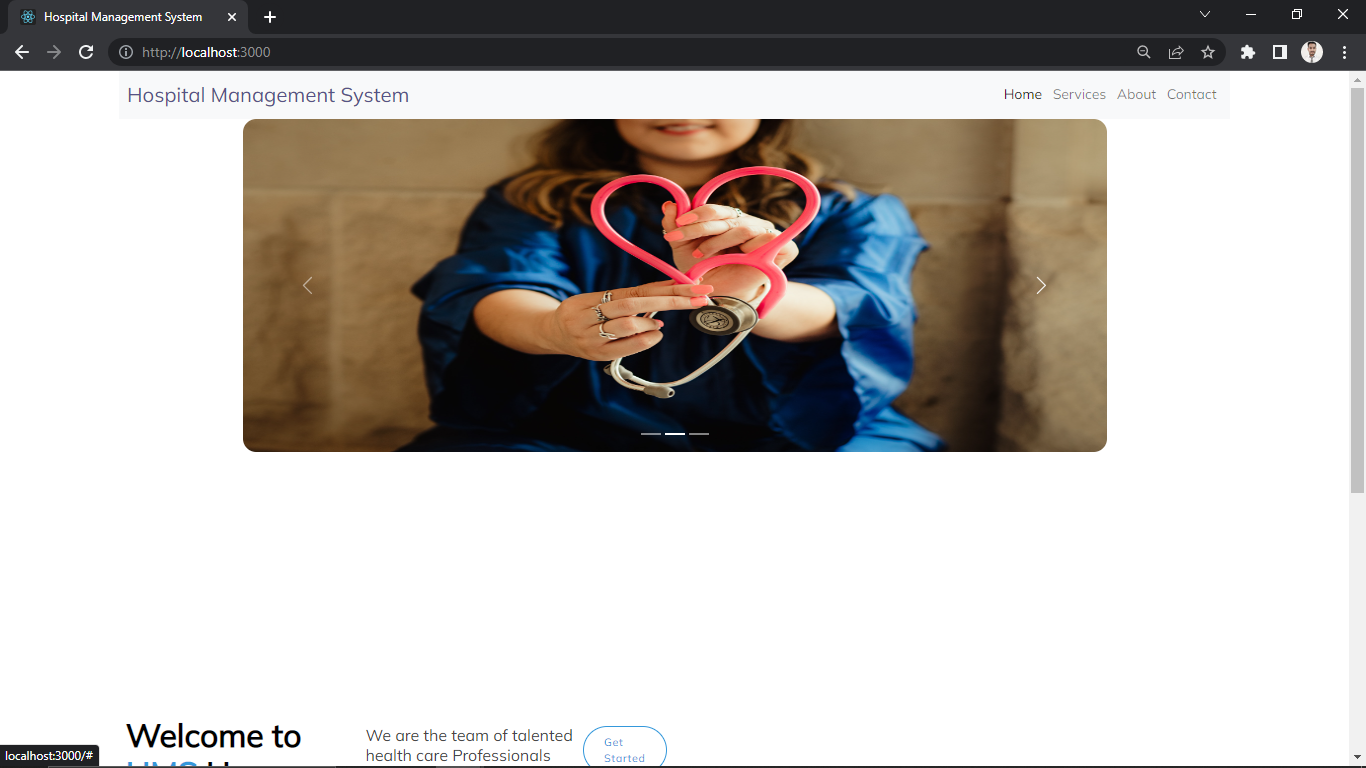


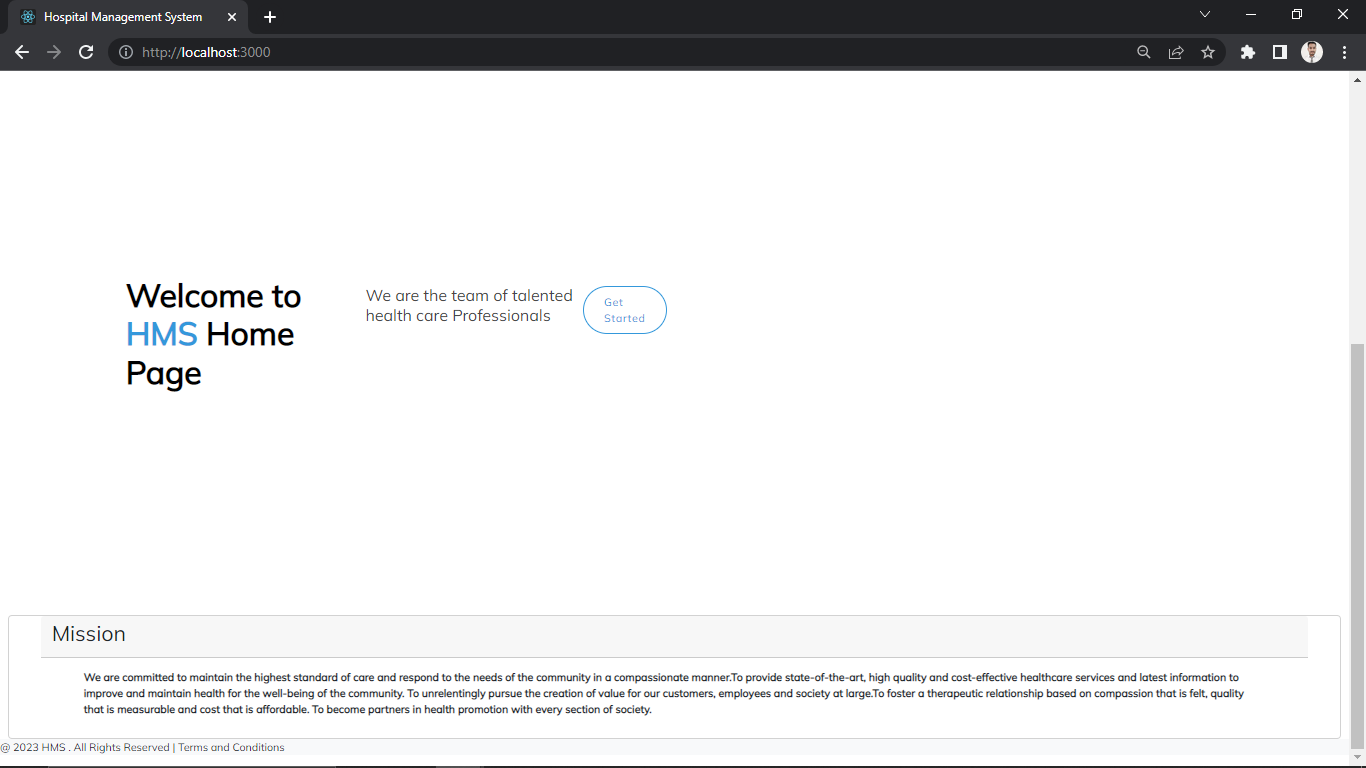
**Accountant:**



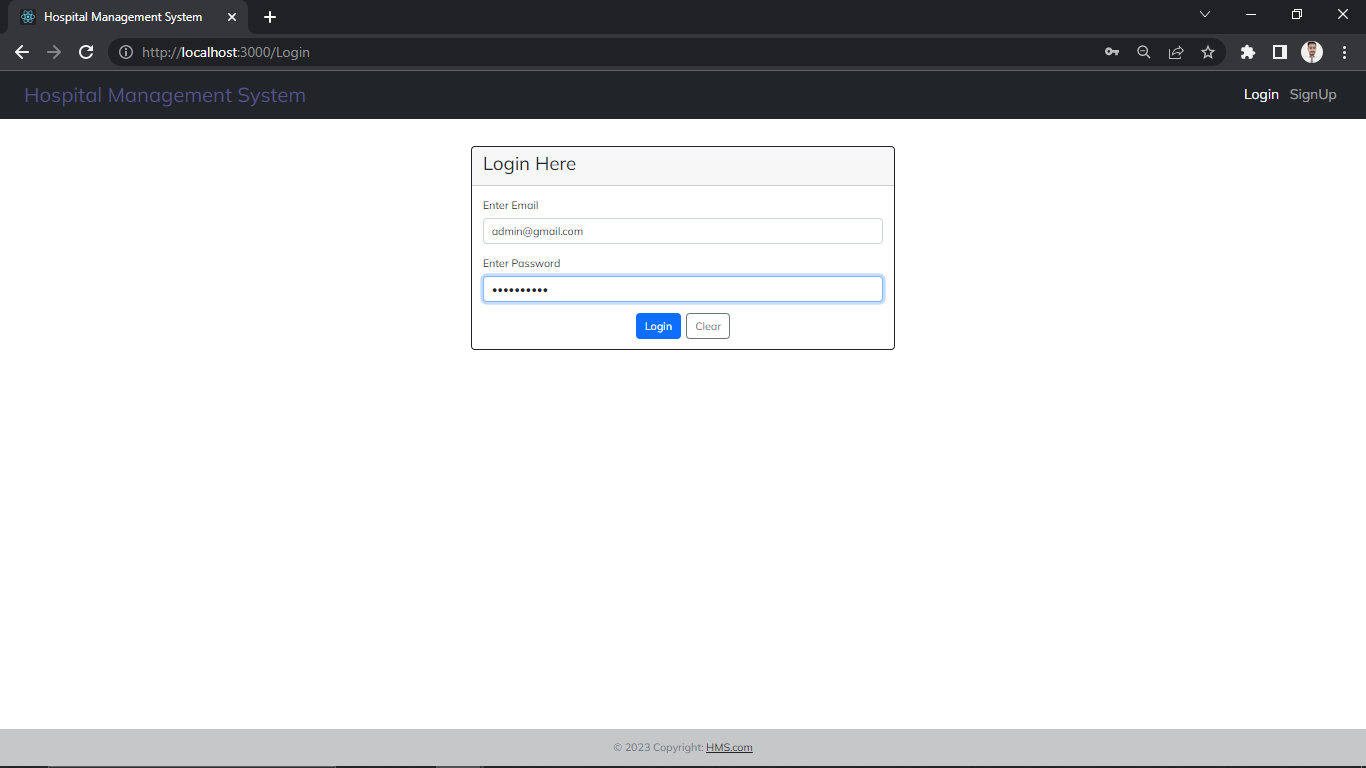
Appendix B

Homepage:

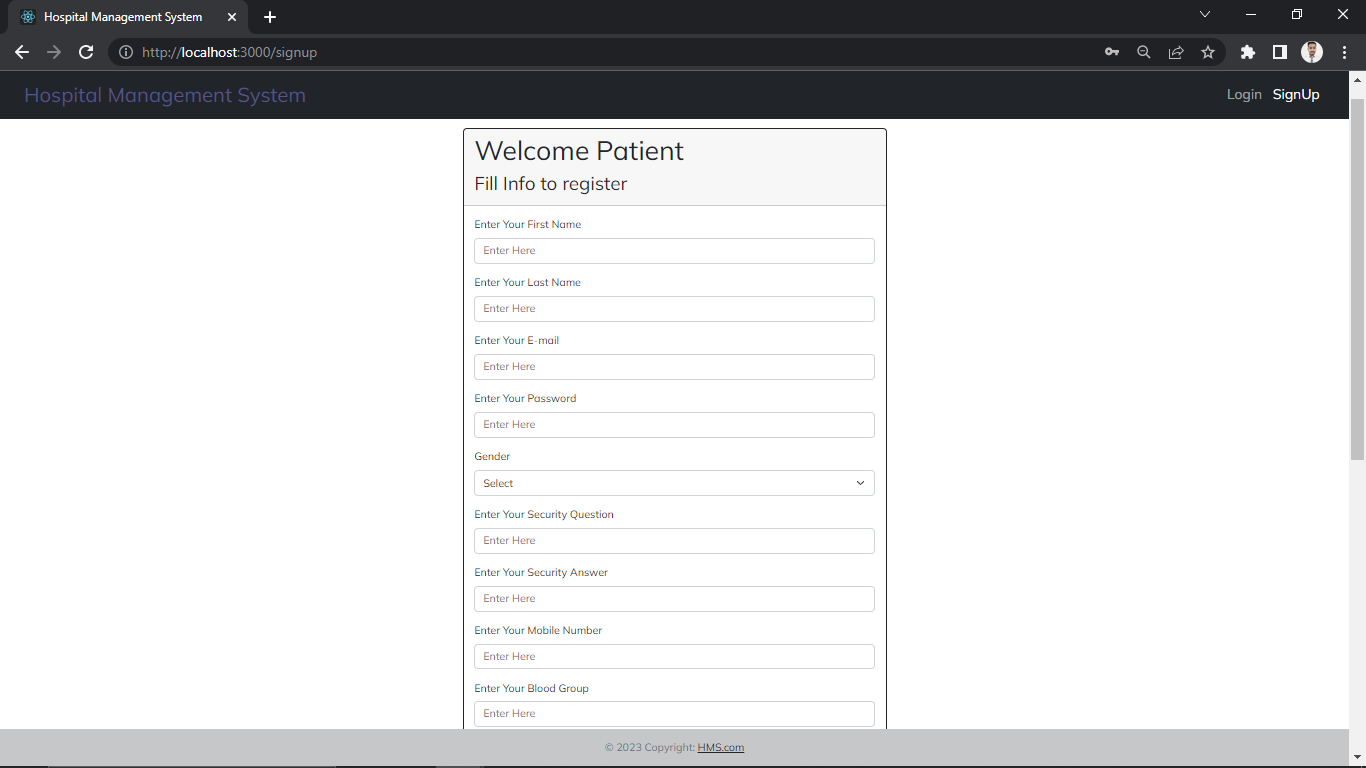




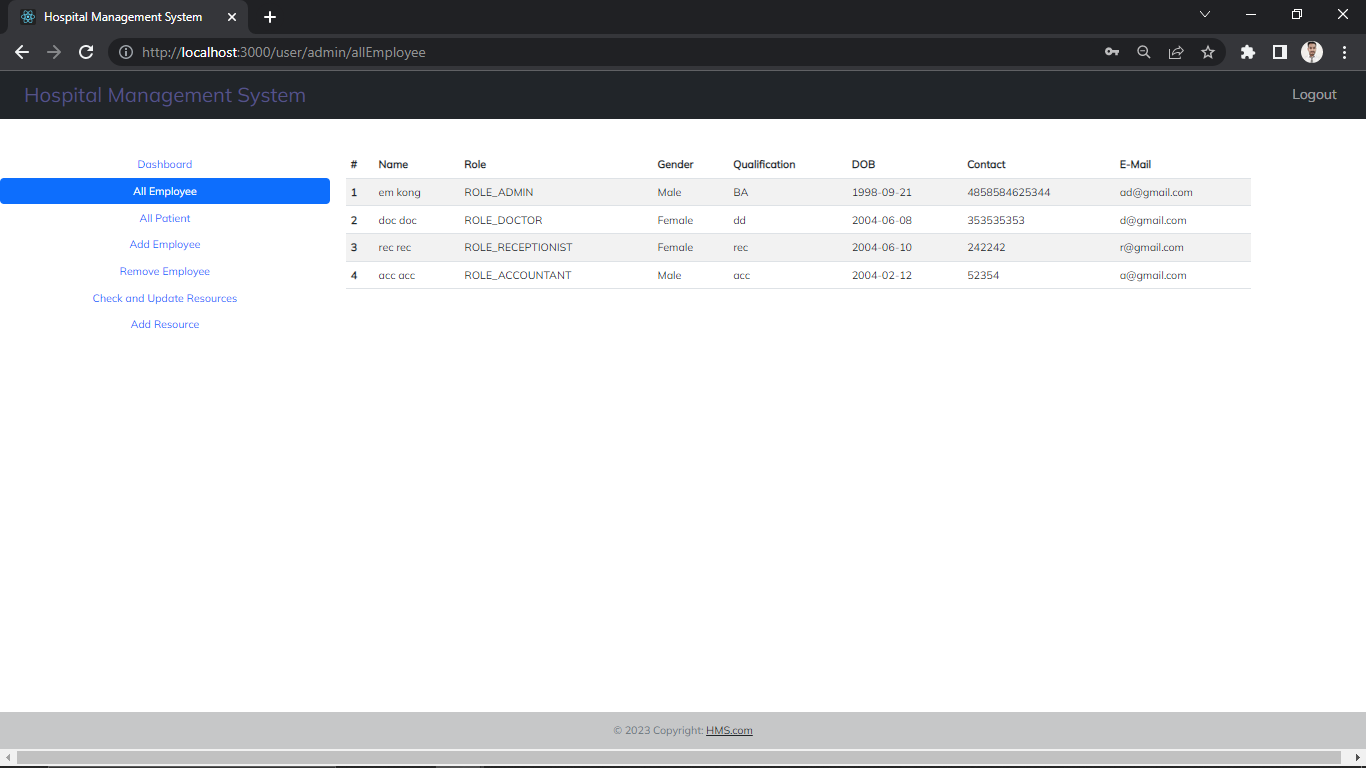
LoginPopup:



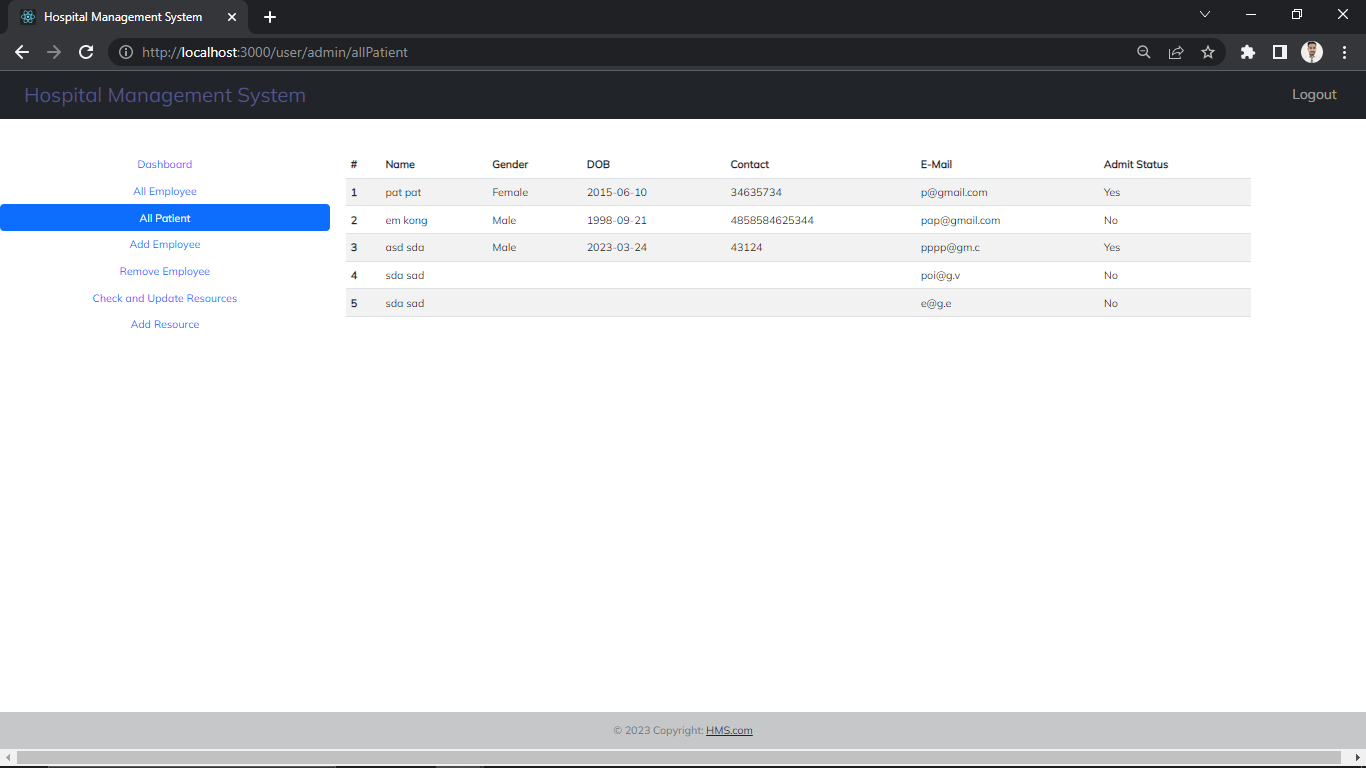
User Registration:



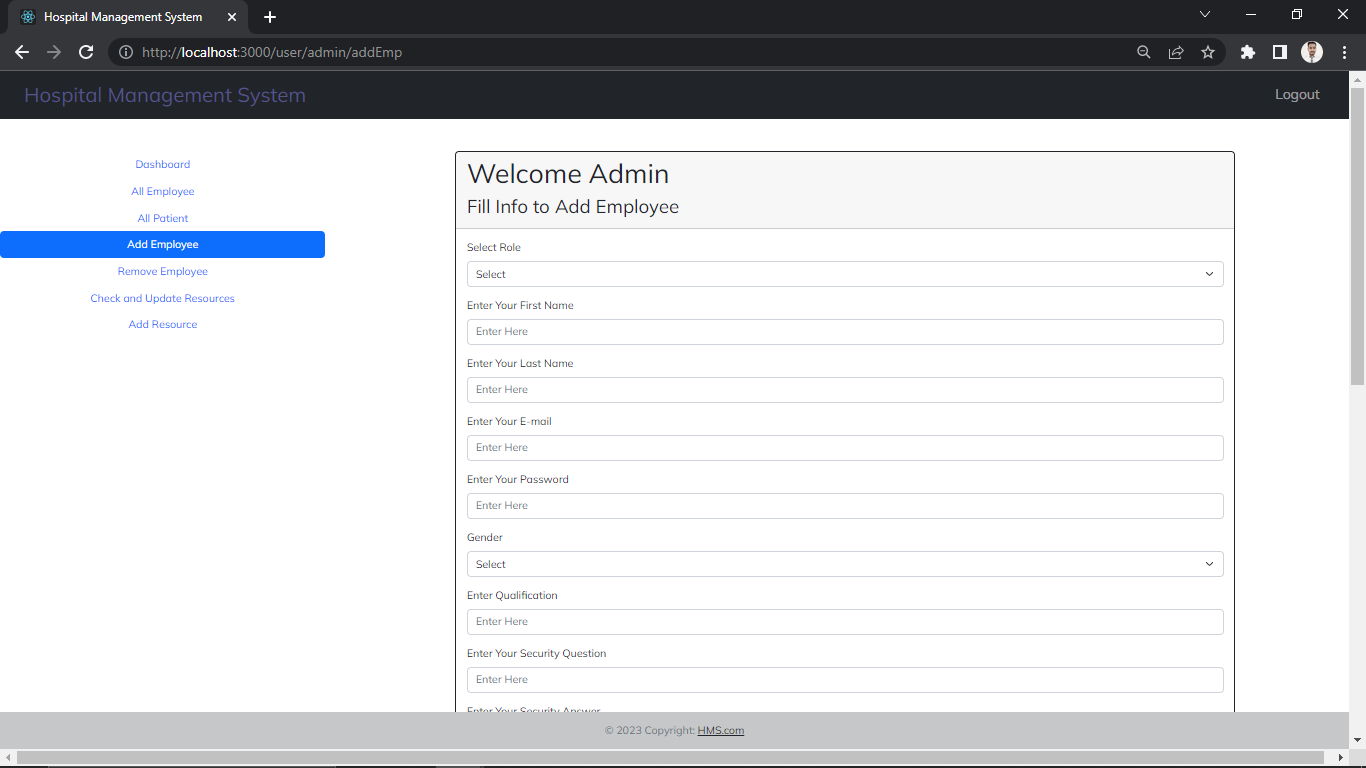
**All Employee**



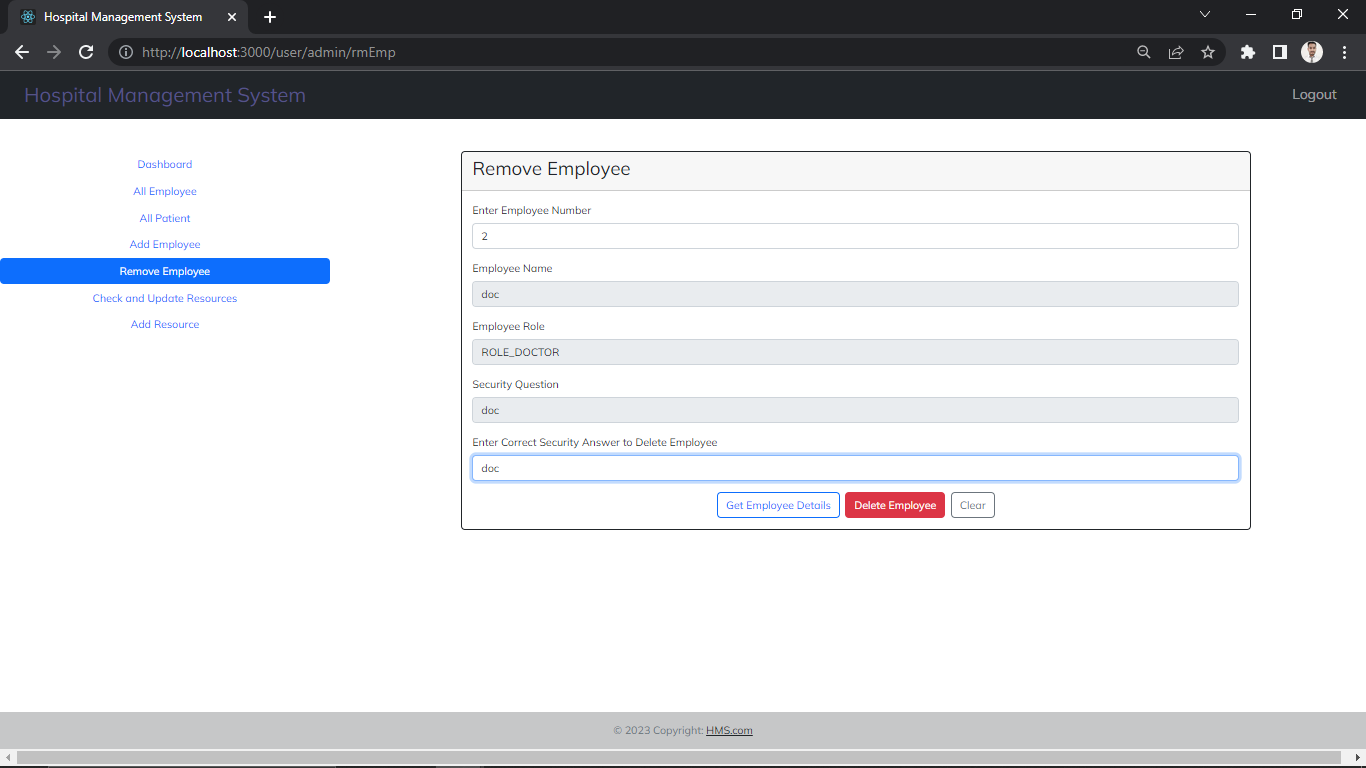
**All Patient**



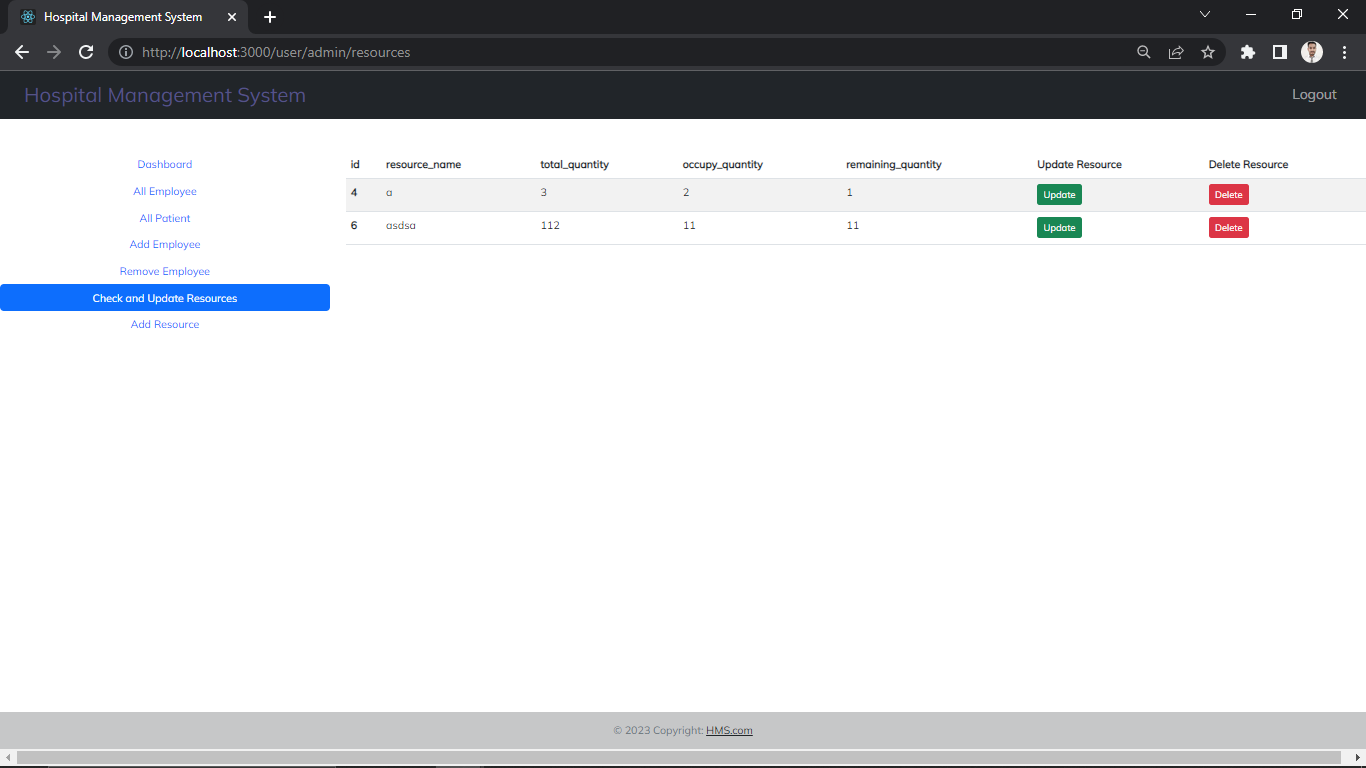
**Add Employee**



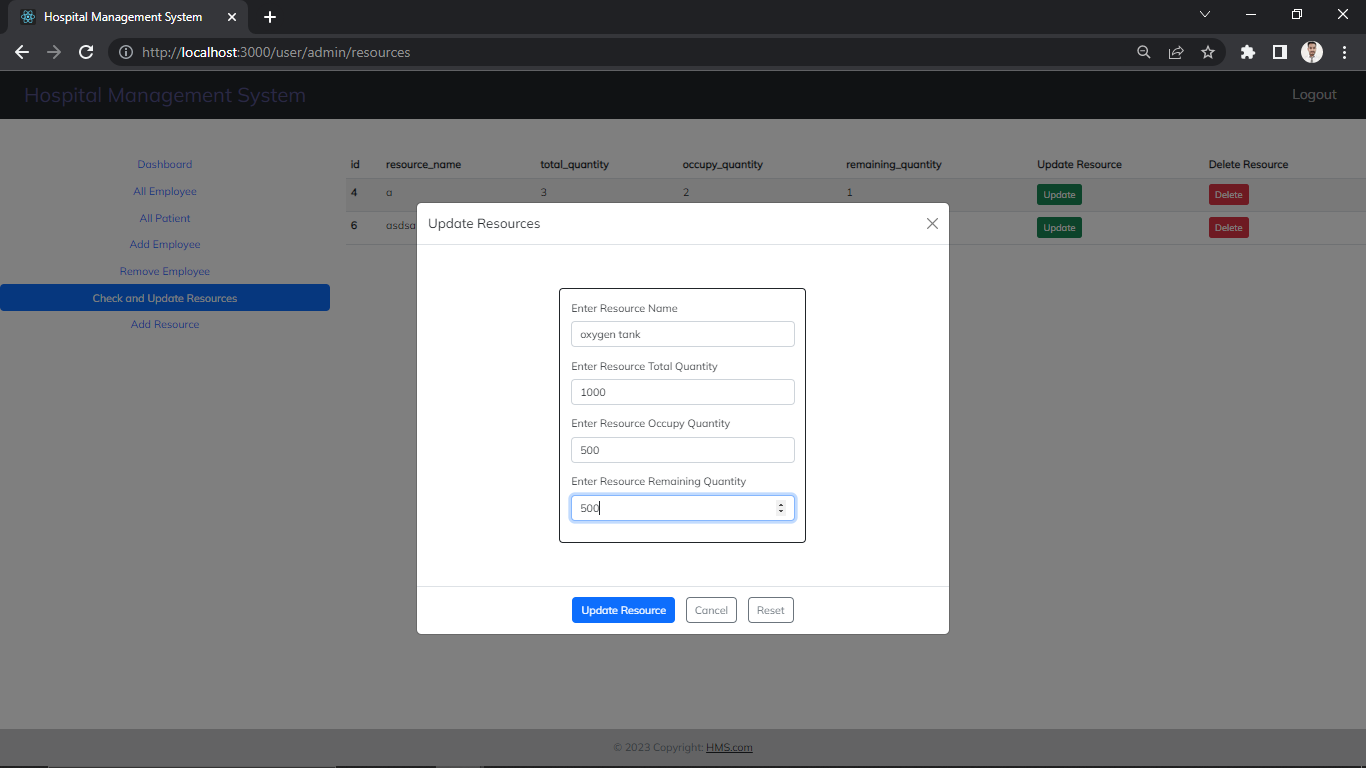
**Remove Employee**



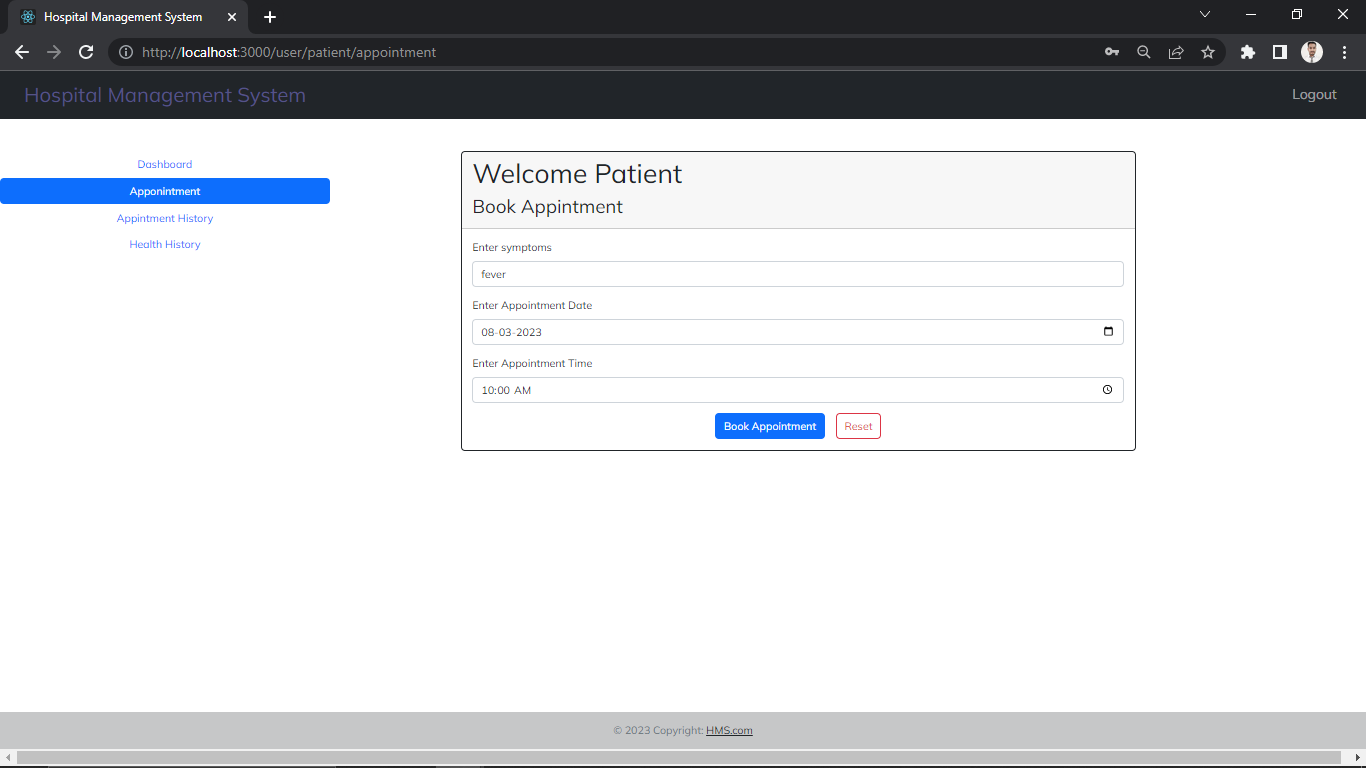
Resources



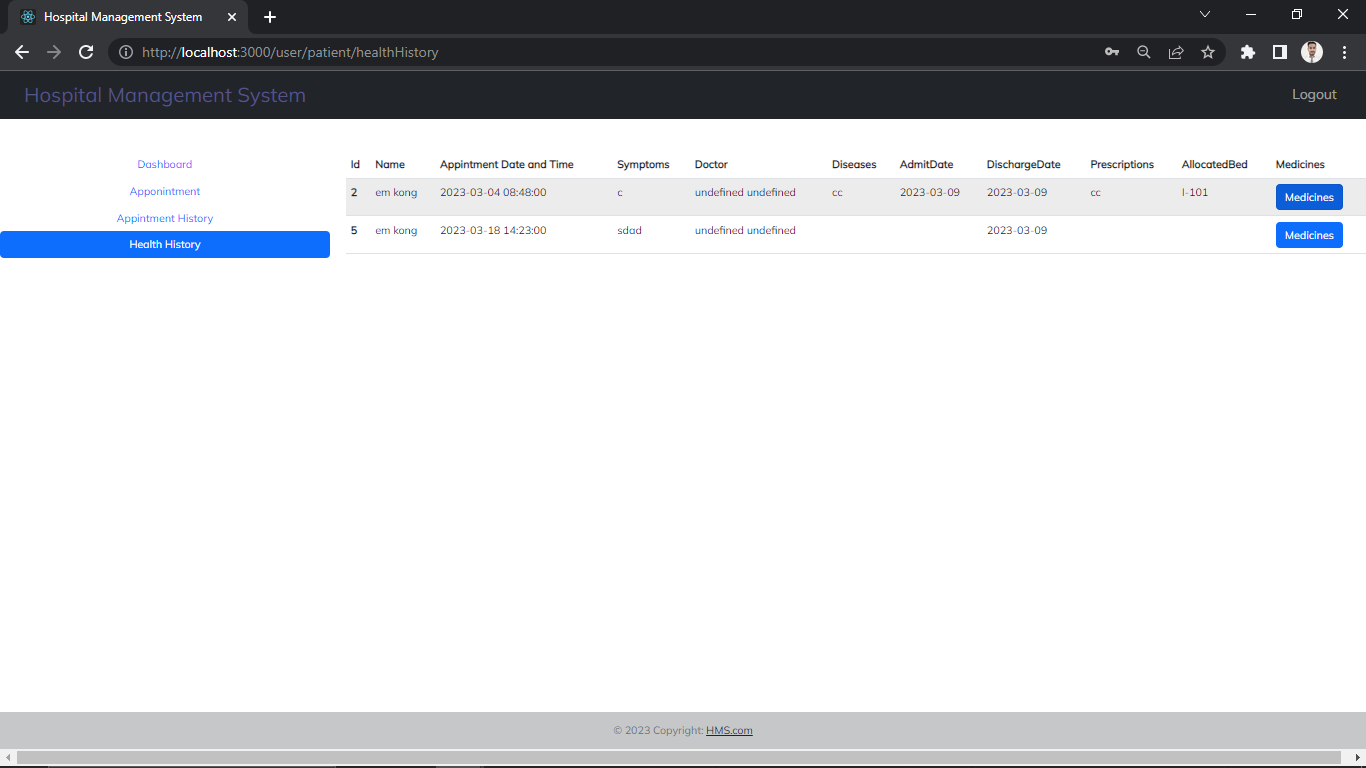
Update Resources



Appointments



Health History



**7.REFERENCES:**

<http://www.google.com>

http://w3school.com/

<http://www.w3.org>

http://www.reactstrap.github.io

http://www.javatpoint.com

http://adityabirlahospital.com/