Smart Internz

HOSPITAL MANAGEMENT SYSTEM USING JAVA SPRING BOOT

TEAM NUMBER : 545

TEAM MEMBERS :
PATRICK ABESHEK A 20BCE1721
ADHAV M 20BCE1743
ASHWIN SIVASANKAR 20BCE1804

1.INTRODUCTION

1.1 Overview

In this Project we aim to solve the traditional issues of hospital management. The existing system provides a paper based solution for keeping OPD records of patients and hospital staff, but it gives overload to Doctor, Receptionist and Administrator. The main issues were inappropriate data keeping, time wastage in storage, retrieval also patients were unable to understand the prescription etc. These issues are solved by providing a separate user account for doctors and other staff. Keeping each patient's data separate and track previous visits in a single click.

This project uses MYSQL as backend and is developed in Java so it provides features such as platform independence, high performance and security. It is a web application which mainly uses SpringMVC and Hibernate frameworks.

It provides some enhanced features such as: an easy interface to add, remove employees as well as it provides PDF of prescription. Thus, reducing the need to manually write and sign by doctor.

1.2 Purpose

Hospitals are the essential part of our lives, providing best medical facilities to people suffering from various ailments, which may be due to change in climatic conditions, increased work-load, emotional trauma, stress etc. It is necessary for the hospitals to

keep track of its day-to-day activities &. records of its patients, doctors that keep the hospital running smoothly & successfully. Our objective is to digitize all the versions of the manual system , and we named it "Hospital Management System".

2 LITERATURE SURVEY

2.1 Existing problem

The existing system relied on paper-based methods, which involved keeping track of all activities such as patient records, doctor information, and other staff personnel records. However, this approach proved to be cumbersome, error-prone, and inefficient. With the continuous increase in population and the number of people visiting the hospital, maintaining these records on paper became highly unreliable and time-consuming. Additionally, the paper-based system was too slow to provide updated lists of required items within a reasonable timeframe. Moreover, it was not economically or technically feasible to maintain these records on paper.

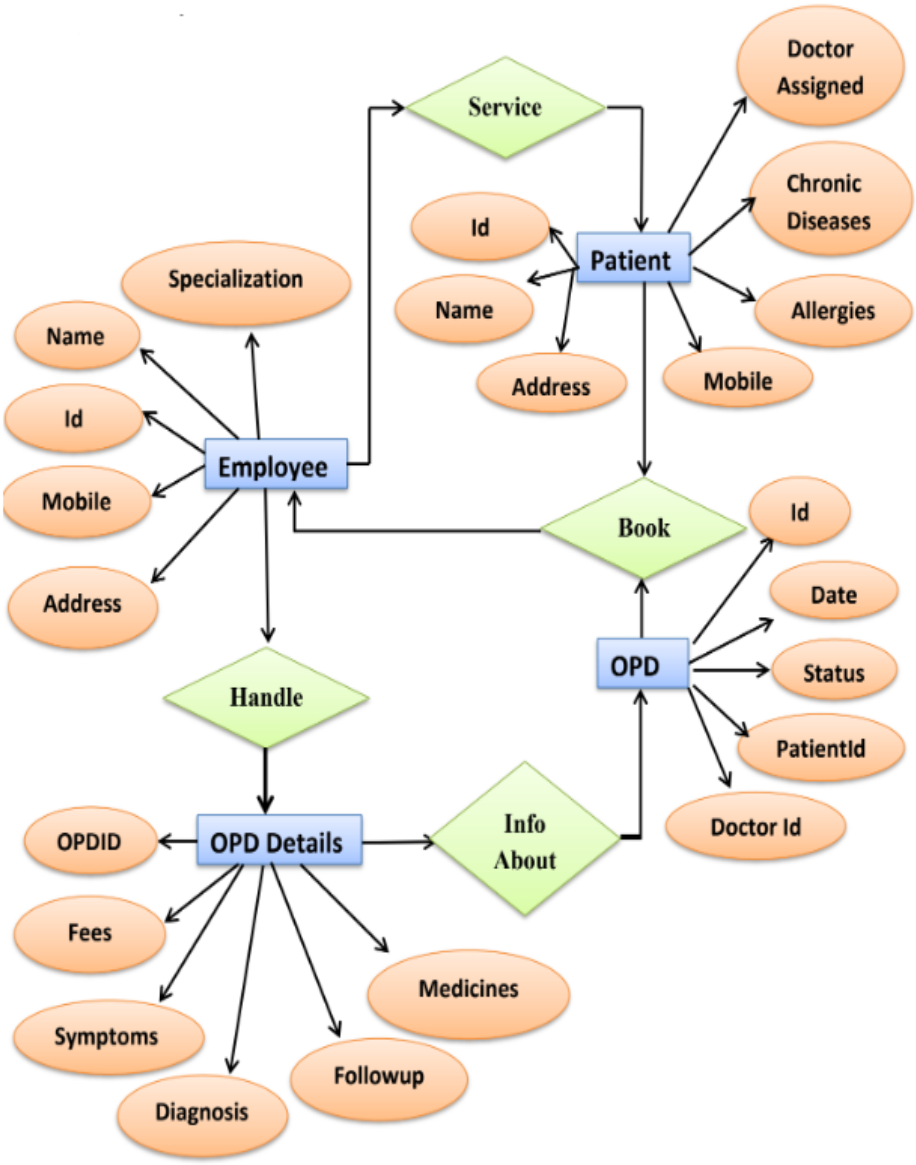
2.2 Proposed solution

Our project aims to create a paperless hospital environment, achieving up to 90% digitalization. We provide a low-cost and reliable Hospital Management System (HMS) that ensures data security, robust storage, and simplified day-to-day operations. The software is user-friendly, allowing doctors to easily manage patient registration, doctor assignments, staff additions, and billing. It is designed for hospitals and clinics, particularly those of small to medium scale, to improve efficiency and accuracy while reducing costs.

3 THEORETICAL ANALYSIS

3.1 Block diagram

Diagrammatic overview of the project.


Service
Doctor Assigned
Chronic Diseases
Id
Patient
Name
Allergies
Address
Mobile
Specialization
Name
Id
Employee
Mobile
Address
Book
Id
Date
OPD
Status
PatientId
Doctor Id
Handle
OPDID
OPD Details
Info About
Fees
Medicines
Symptoms
Followup
Diagnosis

3.2 Hardware / Software designing

Software requirements (Minimum)

- Windows 7
- JRE 1.8
- MySQL server

Hardware Requirements (Minimum)

- Core i5 processor
- 4GB Ram
- 10GB of hard disk space

Software Interfaces

- Operating System: Windows 7 Ultimate or above
- Back End: MySQL, JRE 1.8 or above
- Front End: Bootstrap, jQuery

Communications Interfaces

- Web Browser: Microsoft Edge, Chrome or Firefox

4 EXPERIMENTAL INVESTIGATIONS

Unit Testing

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

Integration Testing

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

System Testing

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

Acceptance Testing

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

Security Testing

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

6.2.6 Functional Testing

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

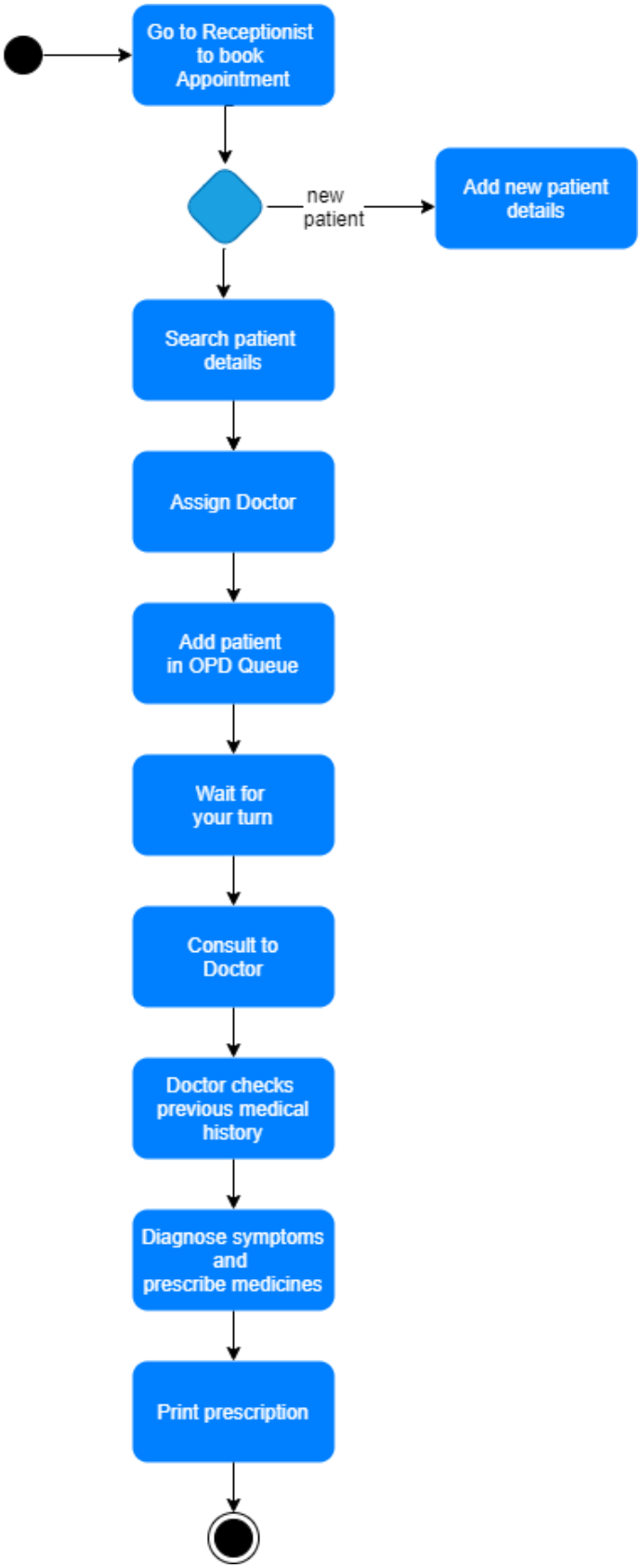
6.2.7 Recovery Testing

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

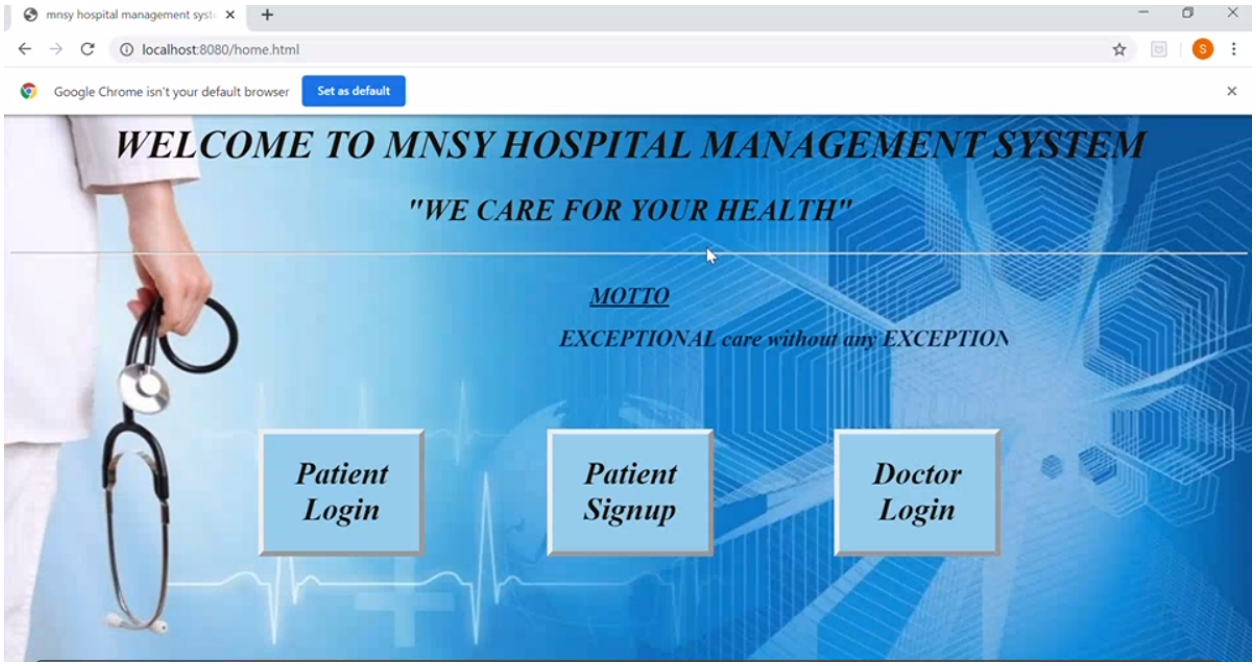
6.2.8 Hardware/Software Testing

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

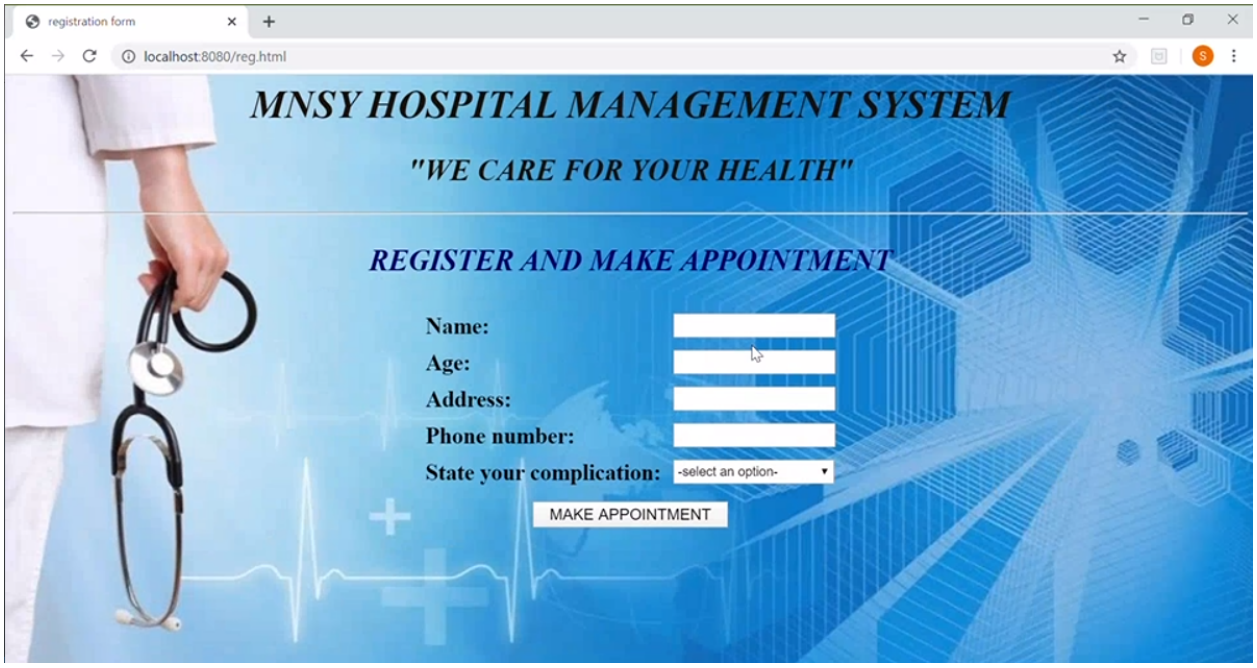
5 FLOWCHART


Go to Receptionist to book Appointment
new patient
Add new patient details
Search patient details
Assign Doctor
Add patient in OPD Queue
Wait for your turn
Consult to Doctor
Doctor checks previous medical history
Diagnose symptoms and prescribe medicines
Print prescription

6 RESULT
HOME PAGE:

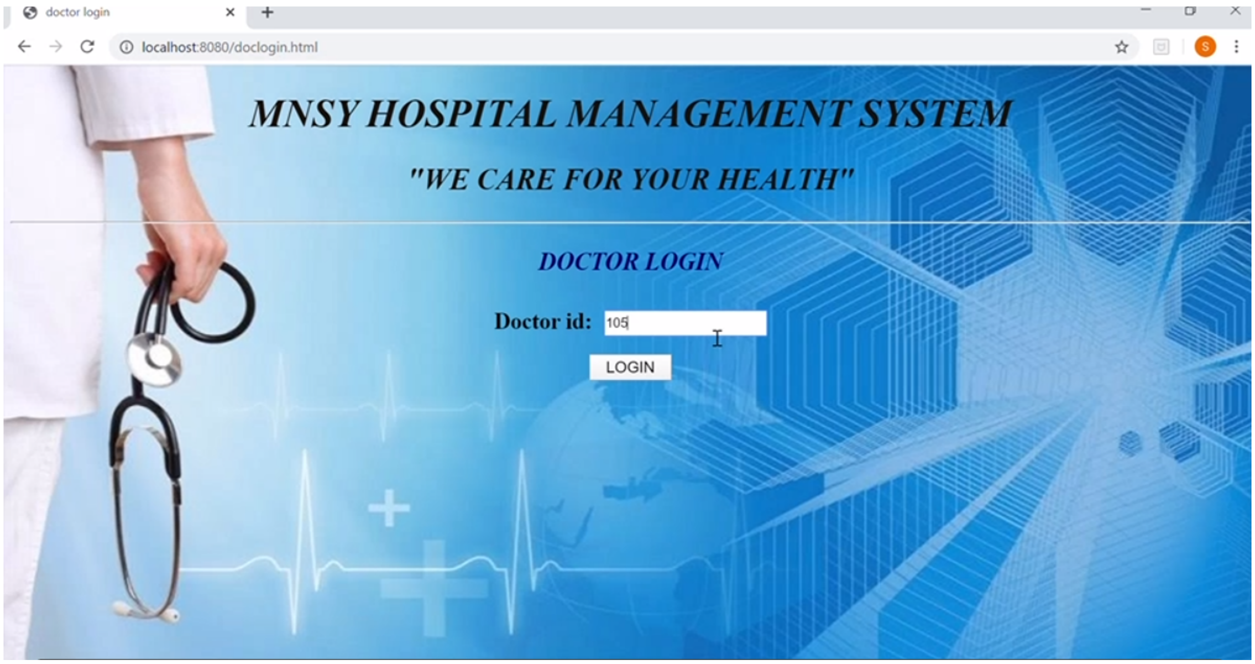


APPOINTMENT PAGE:



PATIENT SCHEDULE PAGE:

DOCTOR LOGIN PAGE:



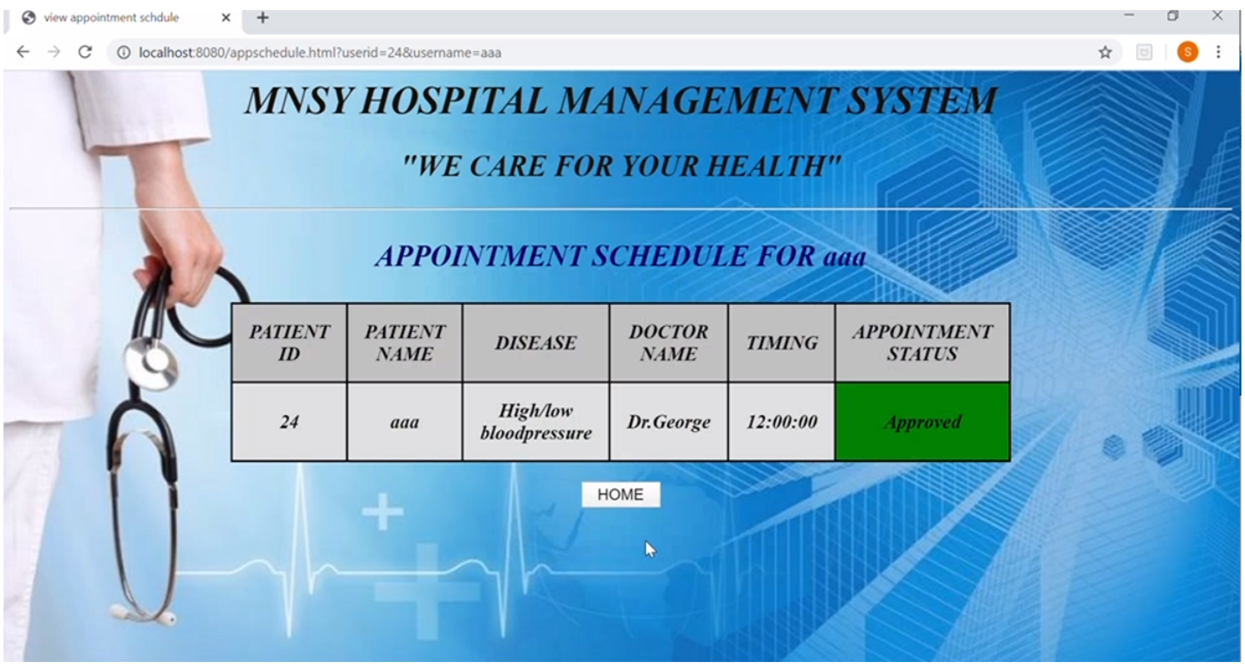
DOCTOR DETAILS PAGE:

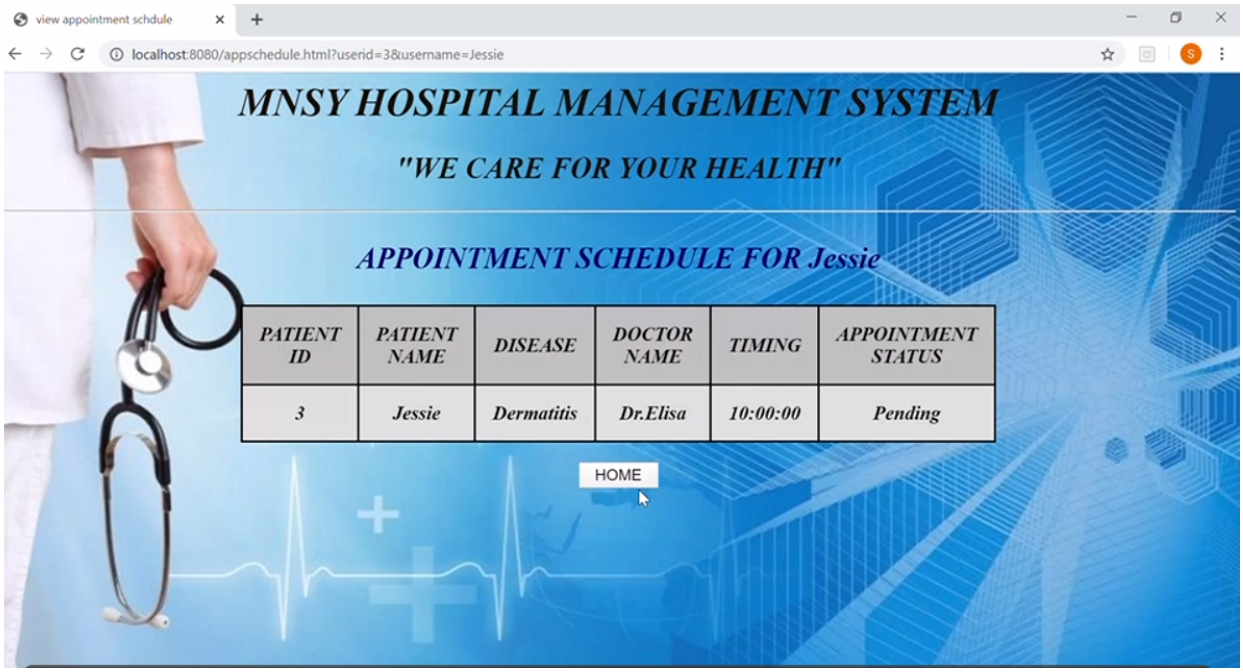
APPOINTMENT PAGE:

PATIENT LOGIN PAGE:

PATIENT DETAILS PAGE:

PATIENT APPOINTMENT SCHEDULE PAGE:





7 ADVANTAGES & DISADVANTAGES

Advantages of Incremental Model:-

1. The software will be generated quickly during the software life cycle.
2. It is flexible and less expensive to change requirements and scope.
3. Throughout the development stages changes can be done.
4. This Model is less costly compared to others.
5. A customer can respond to each building.
6. Errors are easy to identify.

Disadvantages of Incremental Model:-

1. It requires a good planning designing.
2. Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle.
3. Each iteration phase is rigid and does not overlap each other.
4. Rectifying a problem in one unit requires correction in all the units and consumes a lot of time.

8 APPLICATIONS

This project is a comprehensive software solution designed to streamline and automate various processes within a healthcare facility. Using Java Spring Boot, you can develop robust applications for Hospital Management Systems. Here are some key applications that can be built using Java Spring Boot:

- Patient Management: Create modules to manage patient information, including registration, appointment scheduling, medical history, billing, and discharge.

- Doctor and Staff Management: Develop features to manage doctor and staff profiles, including their schedules, specialties, and availability.

- Appointment Scheduling: Design a module that allows patients to schedule appointments online, check the availability of doctors, and receive confirmation notifications.

When developing a Hospital Management System using Java Spring Boot, you can leverage the framework's features like dependency injection, MVC architecture, and Spring Data JPA for database interactions. Additionally, you can use Spring Security for authentication and authorization purposes.

9 CONCLUSION

This project helps in making paperless activities. It reduces the workload from Doctor and Receptionist. It provides more ease and flexibility to Doctor, Administrator and Receptionist. This digitalization has reduced hospital costs. This work has created a little awareness and promotes the idea that the concept of paperless office is reality.

10 FUTURE SCOPE

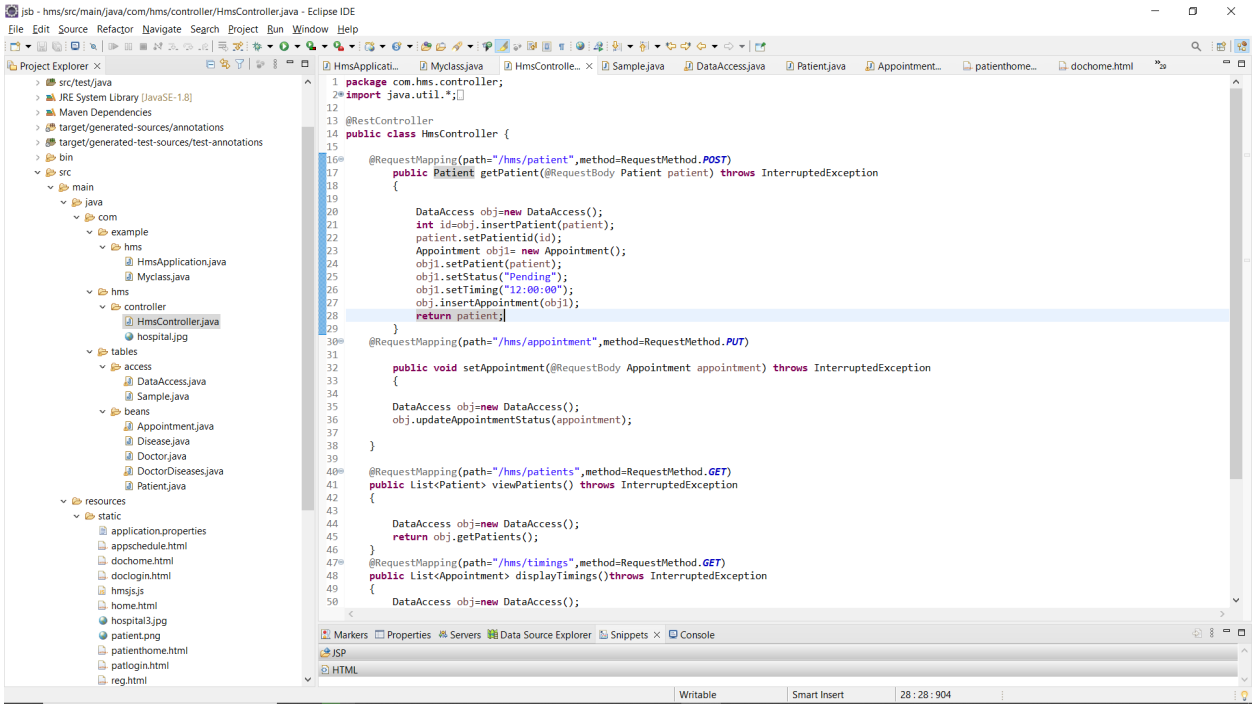
1. To include the Patient module.
2. To manage the IPD section. .

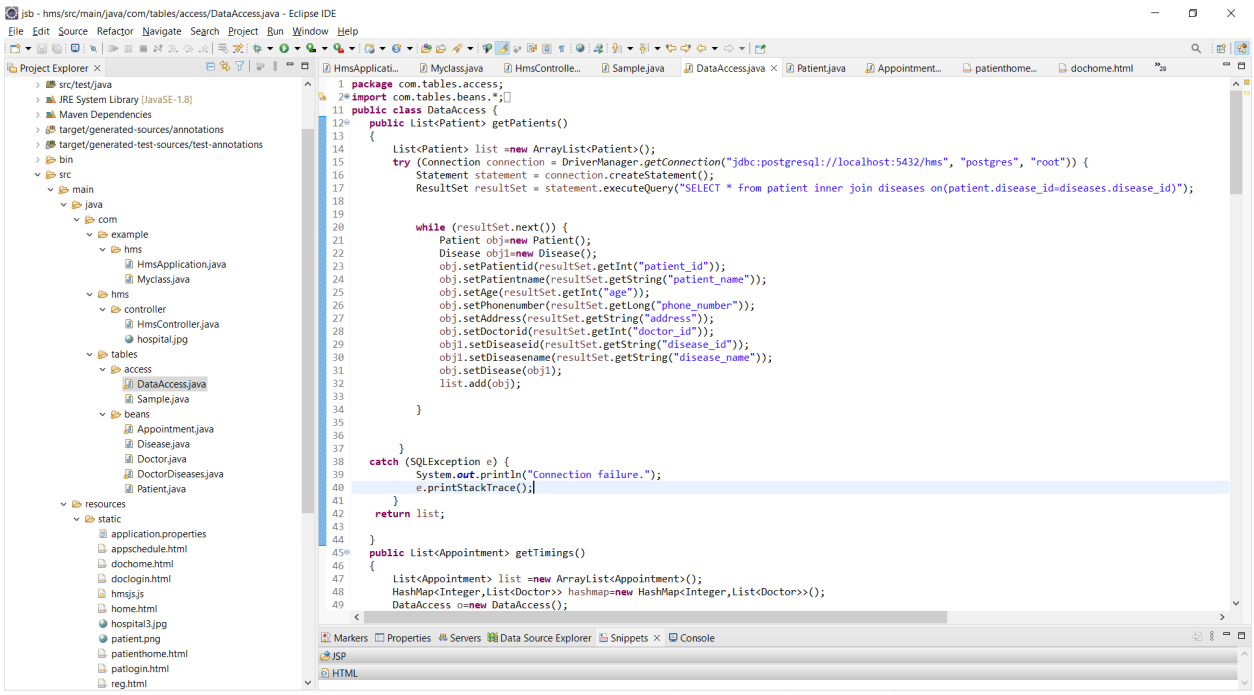
11 BIBLIOGRAPHY

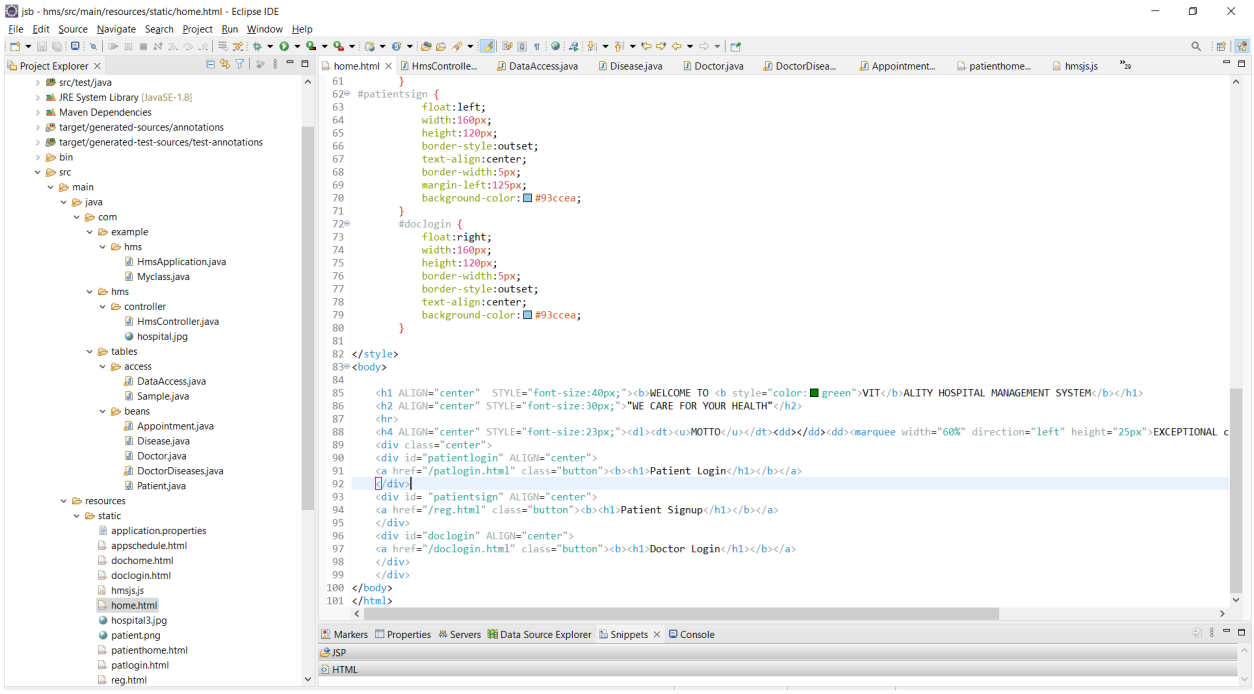
References :-
https://www.youtube.com/watch?v=BkRZfxznaOo
https://www.youtube.com/watch?v=JR7-EdxDSf0
https://www.youtube.com/watch?v=tylOF-uxASg https://youtu be/ikQOR9I5640w
https://www.devglan.com/spring-mve/storing-hashed-password-database-java
https://www.freestudentprojects.com/studentprojectreport/projectreport/hospitalmanagement-system-project-report/.
https://sites.google.com/site/ignoubcafinalyearprojects/project-report/hospitalmanagement-system-project-rep

APPENDIX

Source Code


jsb - hms/src/main/java/com/hms/controller/HmsController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
bin
src
main
java
com
example
hms
HmsApplication.java
Myclass.java
hms
controller
HmsController.java
hospital.jpg
tables
access
DataAccess.java
Sample.java
beans
Appointment.java
Disease.java
Doctor.java
DoctorDiseases.java
Patient.java
resources
static
application.properties
appschedule.html
dochome.html
doclogin.html
hmsjs.js
home.html
hospital3.jpg
patient.png
patienthome.html
patlogin.html
reg.html
HmsApplicati...
Myclass.java
HmsControlle...
Sample.java
DataAccess.java
Patient.java
Appointment...
patienthome...
dochome.html
1 package com.hms.controller;
2 import java.util.*;
13 @RestController
14 public class HmsController {
16 @RequestMapping(path="/hms/patient",method=RequestMethod.POST)
17 public Patient getPatient(@RequestBody Patient patient) throws InterruptedException
18 {
20 DataAccess obj=new DataAccess();
21 int id=obj.insertPatient(patient);
22 patient.setPatientid(id);
23 Appointment obj1= new Appointment();
24 obj1.setPatient(patient);
25 obj1.setStatus("Pending");
26 obj1.setTiming("12:00:00");
27 obj.insertAppointment(obj1);
28 return patient;
29 }
30 @RequestMapping(path="/hms/appointment",method=RequestMethod.PUT)
32 public void setAppointment(@RequestBody Appointment appointment) throws InterruptedException
33 {
35 DataAccess obj=new DataAccess();
36 obj.updateAppointmentStatus(appointment);
38 }
40 @RequestMapping(path="/hms/patients",method=RequestMethod.GET)
41 public List<Patient> viewPatients() throws InterruptedException
42 {
44 DataAccess obj=new DataAccess();
45 return obj.getPatients();
46 }
47 @RequestMapping(path="/hms/timings",method=RequestMethod.GET)
48 public List<Appointment> displayTimings()throws InterruptedException
49 {
50 DataAccess obj=new DataAccess();
Markers Properties Servers Data Source Explorer Snippets Console
JSP
HTML
Writable
Smart Insert
28 : 28 : 904


jsb - hms/src/main/java/com/tables/access/DataAccess.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
bin
src
main
java
com
example
hms
HmsApplication.java
Myclass.java
hms
controller
HmsController.java
hospital.jpg
tables
access
DataAccess.java
Sample.java
beans
Appointment.java
Disease.java
Doctor.java
DoctorDiseases.java
Patient.java
resources
static
application.properties
appschedule.html
dochome.html
doclogin.html
hmsjs.js
home.html
hospital3.jpg
patient.png
patienthome.html
patlogin.html
reg.html
HmsApplicati...
Myclass.java
HmsControlle...
Sample.java
DataAccess.java
Patient.java
Appointment...
patienthome...
dochome.html
1 package com.tables.access;
2 import com.tables.beans.*;
11 public class DataAccess {
12 public List<Patient> getPatients()
13 {
14 List<Patient> list =new ArrayList<Patient>();
15 try (Connection connection = DriverManager.getConnection("jdbc:postgresql://localhost:5432/hms", "postgres", "root")) {
16 Statement statement = connection.createStatement();
17 ResultSet resultSet = statement.executeQuery("SELECT * from patient inner join diseases on(patient.disease_id=diseases.disease_id)");
20 while (resultSet.next()) {
21 Patient obj=new Patient();
22 Disease obj1=new Disease();
23 obj.setPatientid(resultSet.getInt("patient_id"));
24 obj.setPatientname(resultSet.getString("patient_name"));
25 obj.setAge(resultSet.getInt("age"));
26 obj.setPhonenumber(resultSet.getLong("phone_number"));
27 obj.setAddress(resultSet.getString("address"));
28 obj.setDoctorid(resultSet.getInt("doctor_id"));
29 obj1.setDiseaseid(resultSet.getString("disease_id"));
30 obj1.setDiseasename(resultSet.getString("disease_name"));
31 obj.setDisease(obj1);
32 list.add(obj);
37 }
38 catch (SQLException e) {
39 System.out.println("Connection failure.");
40 e.printStackTrace();
41 }
42 return list;
44 }
45 public List<Appointment> getTimings()
46 {
47 List<Appointment> list =new ArrayList<Appointment>();
48 HashMap<Integer,List<Doctor>> hashmap=new HashMap<Integer,List<Doctor>>();
49 DataAccess o=new DataAccess();
Markers Properties Servers Data Source Explorer Snippets Console
JSP
HTML


jsb - hms/src/main/resources/static/home.html - Eclipse IDE
File Edit Source Navigate Search Project Run Window Help
Project Explorer
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
bin
src
main
java
com
example
hms
HmsApplication.java
Myclass.java
hms
controller
HmsController.java
hospital.jpg
tables
access
DataAccess.java
Sample.java
beans
Appointment.java
Disease.java
Doctor.java
DoctorDiseases.java
Patient.java
resources
static
application.properties
appschedule.html
dochome.html
doclogin.html
hmsjs.js
home.html
hospital3.jpg
patient.png
patienthome.html
patlogin.html
reg.html
home.html
HmsControlle...
DataAccess.java
Disease.java
Doctor.java
DoctorDisea...
Appointment...
patienthome...
hmsjs.js
61 }
62 #patientsign {
63 float:left;
64 width:160px;
65 height:120px;
66 border-style:outset;
67 text-align:center;
68 border-width:5px;
69 margin-left:125px;
70 background-color: #93ccea;
71 }
72 #doclogin {
73 float:right;
74 width:160px;
75 height:120px;
76 border-width:5px;
77 border-style:outset;
78 text-align:center;
79 background-color: #93ccea;
80 }
81
82 </style>
83 <body>
84
85 <h1 ALIGN="center" STYLE="font-size:40px;">WELCOME TO VITALITY HOSPITAL MANAGEMENT SYSTEM</h1>
86 <h2 ALIGN="center" STYLE="font-size:30px;">"WE CARE FOR YOUR HEALTH"</h2>
87 <hr>
88 <h4 ALIGN="center" STYLE="font-size:23px;"><dl><dt><u>MOTTO</u></dt><dd></dd><dd><marquee width="60%" direction="left" height="25px">EXCEPTIONAL c
89
90
91 <h1>Patient Login</h1>
92
93
94 <h1>Patient Signup</h1>
95
96
97 <h1>Doctor Login</h1>
98
99
100 </body>
101 </html>
Markers Properties Servers Data Source Explorer Snippets Console
JSP
HTML