Sri Lanka Institute of Information Technology

2020

3rd Year,1st Semester

IT3030 Programing Application and Frameworks

Project Title-HealthCare System

Public VCS Repo- https://github.com/Chamika-mac/HealthCare_app/tree/Sudaraka(IT17022620)

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of science Special Honors Degree in Information Technology

2020/04/19

Declaration

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Student Details

Registration Number: IT17022620

Name: Ampemohotti A.M.D.S.R.

Service: Doctor registration and Making Appointments

Description:

HealthCare is a hospital management system where the registered users can make appointments with the registered doctors who visit the registered hospitals. The users can even make the payments for the appointments online.

Workload:

I created two web services doctor registration and appointment management.

Doctor registration is done by Admin of the system. This process is consisting of adding a new doctor, removing a doctor and updating a doctor's details.

Appointment is done by user of the system. This process is consisting of adding a new appointment, removing an appointment and updating an appointment's details

Table of Contents

Introduction

HealthCare is a hospital management system where the registered users can make appointments with the registered doctors who visit the registered hospitals. The users can even make the payments for the appointments online.

The web application is implemented using the technologies such as Java - JAX-RS (Jersy) on Tomcat and MYSQL database.

User registration process consists of creating new accounts for patients. The patient should sign up to the use the service and successful signup the patient is able to log into the system. And he/she can make an appointment by giving patient details, available appointments of specific doctor, available times and dates for appointments and fees that are provided by the hospital and doctor. Furthermore, patient can cancel appointments in any time and update the details.

Doctor registration done by Admin of the system. This process is consisting of adding a new doctor, removing a doctor and updating a doctor's details. Hospital registration is also done by the Admin of the system. This process consists of adding a new hospital, removing a hospital and updating details of a hospital. Online payments function can pay fees of channelings. The system contains of 5 major web services. Patient Management, Hospital Management, Doctor Management, Appointment Management and Online Payment Management.

Appendix

API-Doctor Management

DoctorResource.java

```
package com.rest.api;

import java.util.List;

import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;


import com.rest.model.DoctorModel;
import controller.DoctorController;

@Path("doctorResources")
public class DoctorResource {

	@GET
	@Path("doctors")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JS
ON})
	public List<DoctorModel> getAlldoctor() throws Exception {
		return DoctorController.getInstance().searchAll();
	}

	@GET
	@Path("doctor/{doctorId}")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JS
ON})
```

```
	public DoctorModel getDoctor(@PathParam("doctorId") int doctorId) throws
Exception {
		return DoctorController.getInstance().search(doctorId);
	}

	@POST
	@Path("doctor")
	public String saveDoctor(DoctorModel obj) throws Exception {
		DoctorController.getInstance().save(obj);
		return "Doctor Saved";
	}

	@PUT
	@Path("doctor")
	public String updateDoctor(DoctorModel obj) throws Exception {
		DoctorController.getInstance().update(obj);
		return "Doctor Updated";
	}

	@DELETE
	@Path("doctor/{doctorId}")
	public String deleteAppintment(@PathParam("doctorId") int doctorId) throws
Exception {
		DoctorModel obj = new DoctorModel();
		obj.setdoctorId(doctorId);
		DoctorController.getInstance().delete(obj);
		return "Doctor Deleted";
	}

}
```

DoctorModel.java

```
package com.rest.model;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class DoctorModel {

	private int doctorId;
	private String Name;
	private String UserName;
```

```
private String ContactNum;
private String Email;
private String Sex;
private String Address;
private String Password;

public int getdoctorId() {
	return doctorId;
}
public void setdoctorId(int doctorId) {
	this.doctorId = doctorId;
}
public String getName() {
	return Name;
}
public void setName(String Name) {
	this.Name = Name;
}
public String getUserName() {
	return UserName;
}
public void setUserName(String UserName) {
	this.UserName = UserName;
}
public String getContactNum() {
	return ContactNum;
}
public void setContactNum(String ContactNum) {
	this.ContactNum = ContactNum;
}
public String getEmail() {
	return Email;
}
public void setEmail(String Email) {
	this.Email = Email;
}
public String getSex() {
	return Sex;
}
public void setSex(String Sex) {
	this.Sex = Sex;
}
public String getAddress() {
	return Address;
}
public void setAddress(String Address) {
	this.Address = Address;
}
public String getPassword() {
	return Password;
}
public void setPassword(String Password) {
	this.Password = Password;
}
```

```
}
```

DoctorController.java

```
package controller;

import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import com.rest.model.DoctorModel;

import DBConnector.Connector;

public class DoctorController {

	Connector con = Connector.getInstance();

	private DoctorController() {

	}

	private static final DoctorController ac = new DoctorController();

	public static DoctorController getInstance() {
		return ac;
	}

	public void save(DoctorModel obj) throws Exception {
		con.getConnection();
		con.aud("INSERT INTO
doctor(Name,UserName,ContactNum,Email,Sex,Address,Password) VALUES ('" +
obj.getName()
				+ "', '" + obj.getUserName() + "', '" + obj.getContactNum()
+ "', '" + obj.getEmail() + "', '"
				+ obj.getSex() + "', " + """ + obj.getAddress() + "', '" +
obj.getPassword() + "')");
	}

	public void update(DoctorModel obj) throws Exception {
		con.getConnection();
		con.aud("UPDATE doctor SET Name = '" + obj.getName() + "',
UserName = '" + obj.getUserName()
```

```
                                + "', ContactNum = '" + obj.getContactNum() + "'," +
"Email = '" + obj.getEmail() + "', Sex='"
                                + obj.getSex() + "', Address='" + obj.getAddress() + "',
Password='" + obj.getPassword() + "' "
                                + "WHERE doctorId='" + obj.getdoctorId() + "'");
        }

        public void delete(DoctorModel obj) throws Exception {
                con.getConnection();
                con.aud("DELETE FROM doctor WHERE doctorId='" +
obj.getdoctorId() + "'");
        }

        public List<DoctorModel> searchAll() throws Exception {
                List<DoctorModel> list = new ArrayList<DoctorModel>();
                con.getConnection();
                ResultSet rset = con.srh("SELECT * FROM doctor");
                while (rset.next()) {
                        DoctorModel obj = new DoctorModel();
                        obj.setdoctorId(rset.getInt(1));
                        obj.setName(rset.getString(2));
                        obj.setUserName(rset.getString(3));
                        obj.setContactNum(rset.getString(4));
                        obj.setEmail(rset.getString(5));
                        obj.setSex(rset.getString(6));
                        obj.setAddress(rset.getString(7));
                        obj.setPassword(rset.getString(8));

                        list.add(obj);
                }
                return list;
        }

        public DoctorModel search(int doctorId) throws Exception {
                con.getConnection();
                DoctorModel obj = null;
                ResultSet rset = con.srh("SELECT * FROM doctor WHERE doctorId='" +
doctorId + "'");
                while (rset.next()) {
                        obj = new DoctorModel();
                        obj.setdoctorId(rset.getInt(1));
                        obj.setName(rset.getString(2));
                        obj.setUserName(rset.getString(3));
```

```
                obj.setContactNum(rset.getString(4));
                obj.setEmail(rset.getString(5));
                obj.setSex(rset.getString(6));
                obj.setAddress(rset.getString(7));
                obj.setPassword(rset.getString(8));

            }
            return obj;
        }

    }
```

Connector.java

```
    package DBConnector;

    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.Statement;

    public class Connector {

        private Connector() {

        }

        private static final Connector obj = new Connector();

        public static Connector getInstance() {
            return obj;
        }

        private static Connection con;
        ResultSet rs;

        public Connection getConnection() throws Exception {
            if (con == null) {
                Class.forName("com.mysql.jdbc.Driver");
                con =
    DriverManager.getConnection("jdbc:mysql://localhost:3306/rest_api", "root", "");
```

```
        }
        return con;
}

public int aud(String sql) throws Exception {
        getConnection();
        Statement st = con.createStatement();
        int i = st.executeUpdate(sql);
        return i;
}

public int audr(String sql) throws Exception {
        getConnection();
        Statement st = con.createStatement();
        int i = st.executeUpdate(sql);
        ResultSet rs = st.executeQuery("SELECT LAST_INSERT_ID()");
        while (rs.next()) {
                i = rs.getInt("LAST_INSERT_ID()");
        }
        return i;
}

public ResultSet srh(String sql) throws Exception {
        getConnection();
        Statement st = con.createStatement();
        rs = st.executeQuery(sql);
        return rs;
}

public int checkavailable(String sql, String column) throws Exception {
        int i = 0;
        rs = srh(sql);
        while (rs.next()) {
                String s = rs.getString(column);
                if (s.equals(null)) {
                        i = 0;
                } else {
                        i = 1;
                }
        }
        return i;
}
```

```
	public int nextnum(String sql, String column) throws Exception {
		int id = 0;
		rs = srh(sql);
		while (rs.next()) {
			id = rs.getInt(column) + 1;
		}
		return id;
	}
}
```

API-Appointment Management

AppointmentResource.java

```
package com.rest.api;

import javax.ws.rs.Path;
import javax.ws.rs.PathParam;

import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.DELETE;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import com.rest.model.AppointmentModel;

import controller.AppointmentController;

@Path("appointmentResources")
public class AppointmentResource {

	@GET
	@Path("appointments")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JS
ON})
```

```
	public List<AppointmentModel> getAllAppointment() throws Exception {

		return AppointmentController.getInstance().searchAll();
	}

	@GET
	@Path("appointment/{appointmentId}")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
	public AppointmentModel getAppointment(@PathParam("appointmentId") int appointmentId) throws Exception {
		return AppointmentController.getInstance().search(appointmentId);
	}

	@POST
	@Path("appointment")
	public String saveAppintment(AppointmentModel obj) throws Exception {
		AppointmentController.getInstance().save(obj);
		return "Appointement Saved";
	}

	@PUT
	@Path("appointment")
	public String updateAppintment(AppointmentModel obj) throws Exception {
		AppointmentController.getInstance().update(obj);
		return "Appointement Updated";
	}

	@DELETE
	@Path("appointment/{appointmentId}")
	public String deleteAppintment(@PathParam("appointmentId") int appointmentId) throws Exception {
		AppointmentModel obj = new AppointmentModel();
		obj.setAppointementId(appointmentId);
		AppointmentController.getInstance().delete(obj);
		return "Appointement Deleted";
	}

}
```

AppointmentModel.java

```java
package com.rest.model;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class AppointmentModel {

	private int appointementId;
	private String Name;
	private String date;
	private String time;
	private String doctor_name;
	private String email;
	private String contactNum;
	private String Hospital_Name;

	public int getAppointementId() {
		return appointementId;
	}
	public void setAppointementId(int appointementId) {
		this.appointementId = appointementId;
	}
	public String getName() {
		return Name;
	}
	public void setName(String name) {
		Name = name;
	}
	public String getDate() {
		return date;
	}
	public void setDate(String date) {
		this.date = date;
	}
	public String getTime() {
		return time;
	}
	public void setTime(String time) {
		this.time = time;
	}
	public String getDoctor_name() {
		return doctor_name;
	}
	public void setDoctor_name(String doctor_name) {
		this.doctor_name = doctor_name;
	}
	public String getEmail() {
		return email;
	}
	}
```

```
	public void setEmail(String email) {
		this.email = email;
	}
	public String getContactNum() {
		return contactNum;
	}
	public void setContactNum(String contactNum) {
		this.contactNum = contactNum;
	}
	public String getHospitalName() {
		return Hospital_Name;
	}
	public void setHospitalName(String Hospital_Name) {
		this.Hospital_Name = Hospital_Name;
	}

}
```

AppointmentController.java

```
package controller;


import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;


import com.rest.model.AppointmentModel;


import DBConnector.Connector;


public class AppointmentController {


	Connector con = Connector.getInstance();
```

```
	private AppointmentController() {


	}


	private static final AppointmentController ac = new AppointmentController();


	public static AppointmentController getInstance() {
		return ac;
	}


	public void save(AppointmentModel obj) throws Exception {
		con.getConnection();
		con.aud("INSERT INTO
appointment(Name,date,time,doctor_name,email,contactNum,Hospital_Name) VALUES ('" +
obj.getName() + "', "
				+ "'" + obj.getDate() + "', '" + obj.getTime() + "', '" +
obj.getDoctor_name() + "', '" + obj.getEmail()
				+ "', " + "'" + obj.getContactNum() + "', '" + obj.getHospitalName()
+ "')");
	}


	public void update(AppointmentModel obj) throws Exception {
		con.getConnection();
		con.aud("UPDATE appointment SET Name = '" + obj.getName() + "', date = '" +
obj.getDate() + "', time = '"
				+ obj.getTime() + "'," + "doctor_name = '" + obj.getDoctor_name()
+ "', email='" + obj.getEmail()
```

```
                    + "', contactNum='" + obj.getContactNum() + "',
Hospital_Name='" + obj.getHospitalName() + "' " + "WHERE appointmentId='" +
obj.getAppointementId()
                    + "'");
    }

    public void delete(AppointmentModel obj) throws Exception {
        con.getConnection();
        con.aud("DELETE FROM appointment WHERE appointmentId='" +
obj.getAppointementId() + "'");
    }

    public List<AppointmentModel> searchAll() throws Exception {
        List<AppointmentModel> list = new ArrayList<AppointmentModel>();
        con.getConnection();
        ResultSet rset = con.srh("SELECT * FROM appointment");
        while (rset.next()) {
            AppointmentModel obj = new AppointmentModel();
            obj.setAppointementId(rset.getInt(1));
            obj.setName(rset.getString(2));
            obj.setDate(rset.getString(3));
            obj.setTime(rset.getString(4));
            obj.setDoctor_name(rset.getString(5));
            obj.setEmail(rset.getString(6));
            obj.setContactNum(rset.getString(7));
```

```
                obj.setHospitalName(rset.getString(8));


                list.add(obj);
            }
            return list;
        }

        public AppointmentModel search(int appointmentId) throws Exception {
            con.getConnection();
            AppointmentModel obj = null;
            ResultSet rset = con.srh("SELECT * FROM appointment WHERE
appointmentId='" + appointmentId + "'");
            while (rset.next()) {
                obj = new AppointmentModel();
                obj.setAppointementId(rset.getInt(1));
                obj.setName(rset.getString(2));
                obj.setDate(rset.getString(3));
                obj.setTime(rset.getString(4));
                obj.setDoctor_name(rset.getString(5));
                obj.setEmail(rset.getString(6));
                obj.setContactNum(rset.getString(7));
                obj.setHospitalName(rset.getString(8));


            }
            return obj;
```

```
	}

}
```

Connector.java

```
package DBConnector;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Connector {

	private Connector() {

	}

	private static final Connector obj = new Connector();

	public static Connector getInstance() {
		return obj;
	}

	private static Connection con;
```

```
	ResultSet rs;

	public Connection getConnection() throws Exception {
		if (con == null) {
			Class.forName("com.mysql.jdbc.Driver");
			con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/rest_api", "root", "");

		}
		return con;
	}

	public int aud(String sql) throws Exception {
		getConnection();
		Statement st = con.createStatement();
		int i = st.executeUpdate(sql);
		return i;
	}

	public int audr(String sql) throws Exception {
		getConnection();
		Statement st = con.createStatement();
		int i = st.executeUpdate(sql);
		ResultSet rs = st.executeQuery("SELECT LAST_INSERT_ID()");
		while (rs.next()) {
```

```
                i = rs.getInt("LAST_INSERT_ID()");
        }
        return i;
}

public ResultSet srh(String sql) throws Exception {
        getConnection();
        Statement st = con.createStatement();
        rs = st.executeQuery(sql);
        return rs;
}

public int checkavailable(String sql, String column) throws Exception {
        int i = 0;
        rs = srh(sql);
        while (rs.next()) {
                String s = rs.getString(column);
                if (s.equals(null)) {
                        i = 0;
                } else {
                        i = 1;
                }
        }
        return i;
}
```

```
    public int nextnum(String sql, String column) throws Exception {
        int id = 0;
        rs = srh(sql);
        while (rs.next()) {
            id = rs.getInt(column) + 1;
        }
        return id;
    }
}
```

API – DOCTOR MANAGMNET

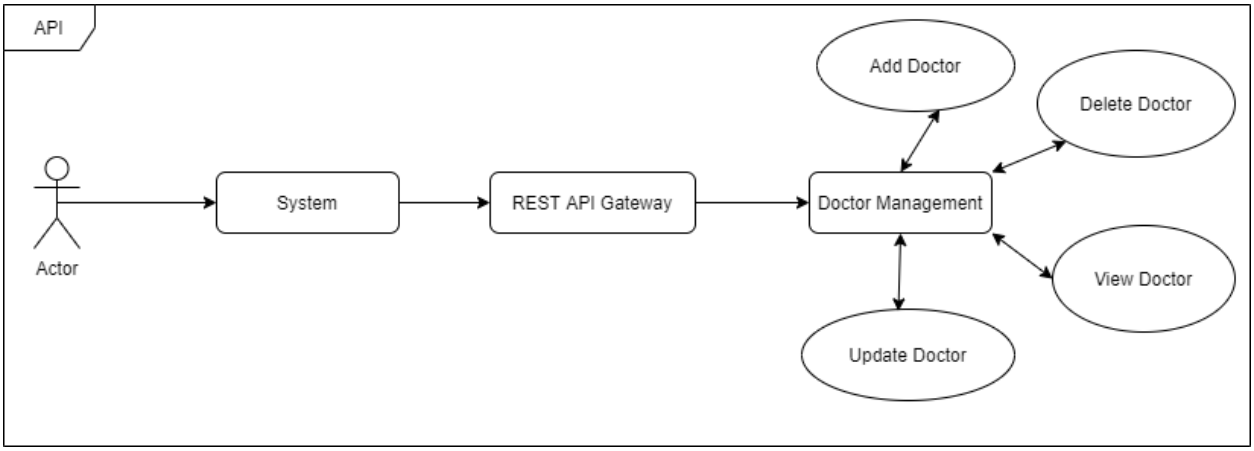


Figure 1.1:API diagram of doctor management

Doctor registration done by Admin of the system. This process is consisting of adding a new doctor, removing a doctor and updating a doctor's details.

The API diagram shows how doctor management web service is exposed to users via a RESTful API.

Service Design

1. Internal Logic

1.1. Class Diagram

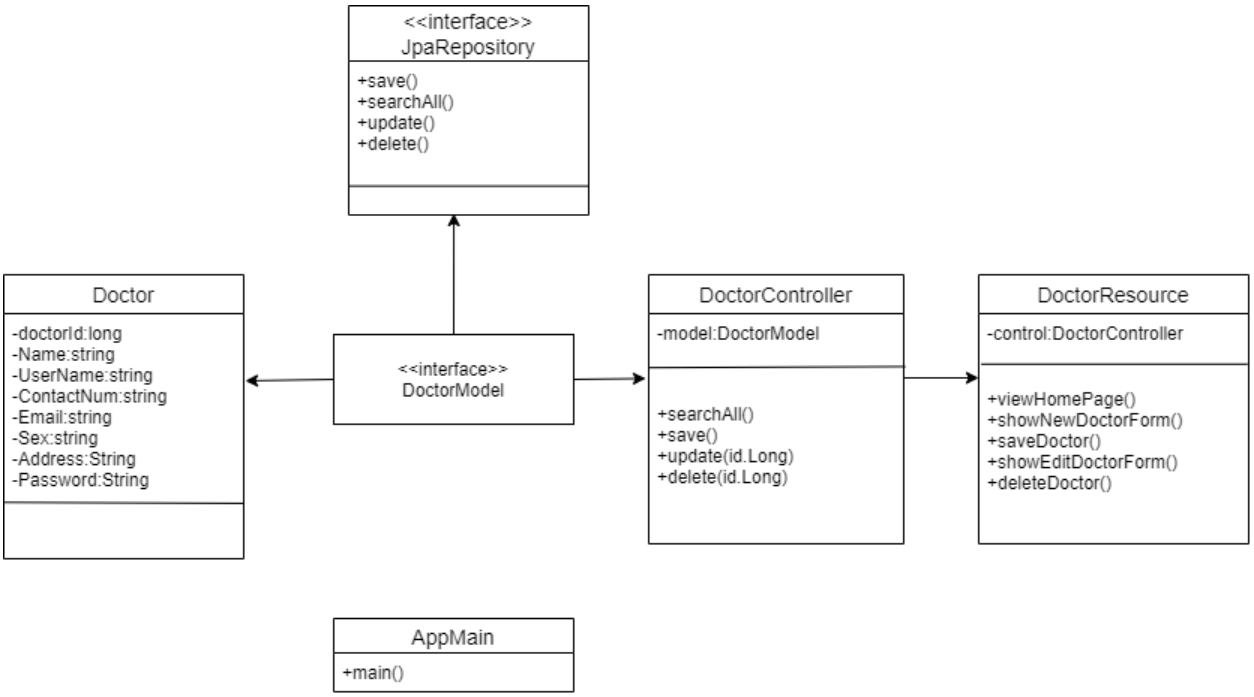


Figure1.2: Class Diagram of doctor management

This class diagram shows the whole classes, attributes, operations and relationships between objects related to the doctor management service is modelled here. Here we use styles and pattern like MVC.

1.2. Activity Diagram

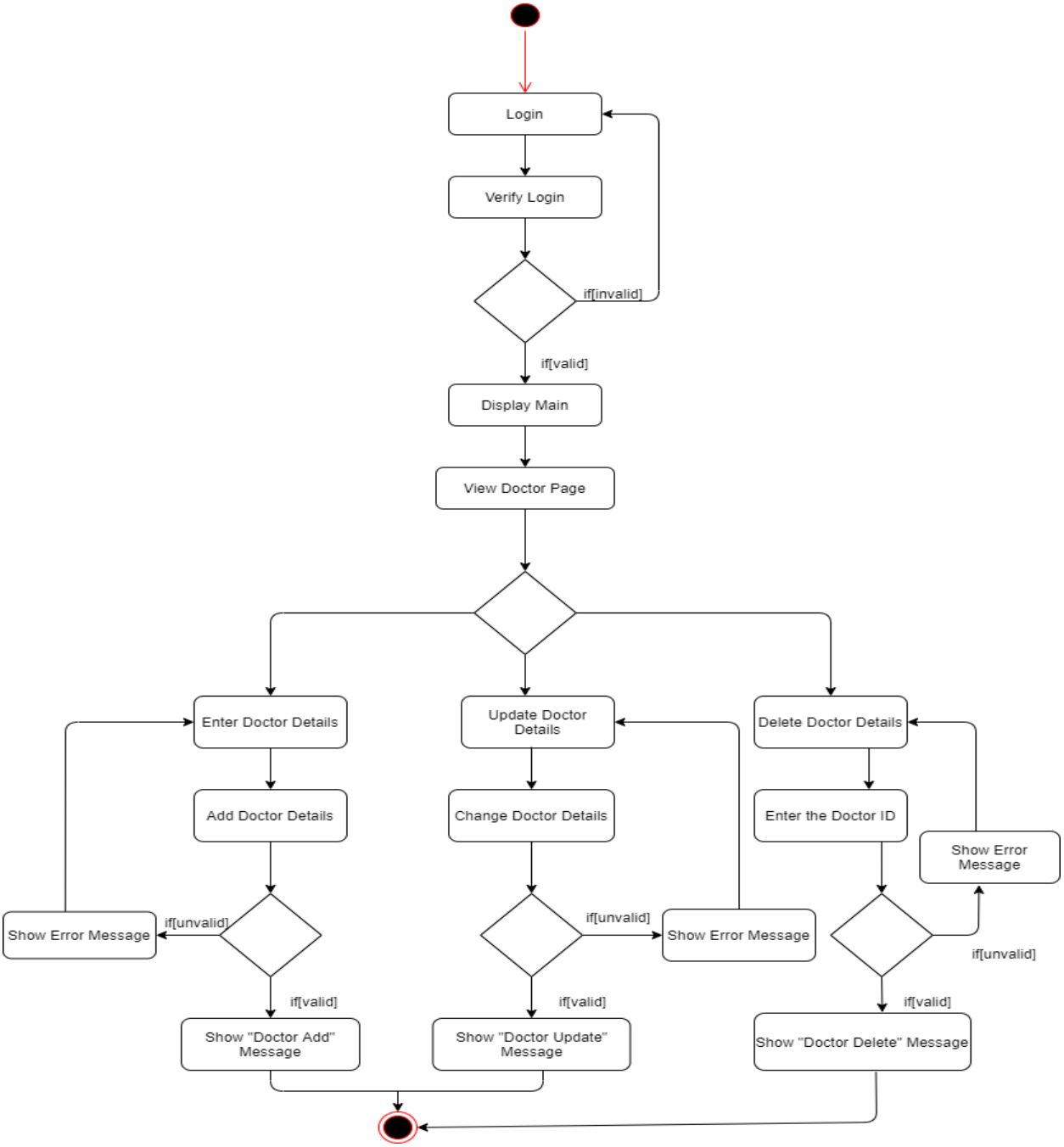
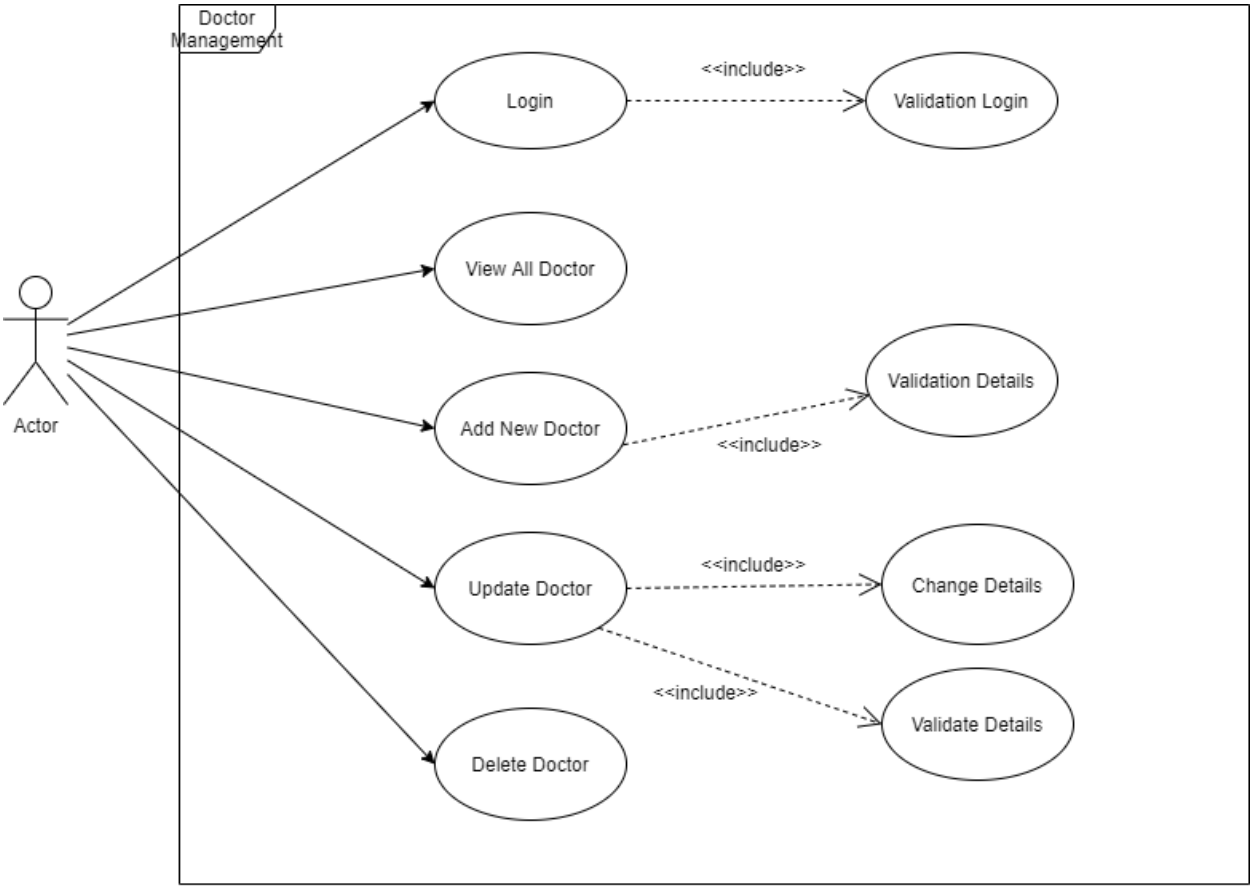


Figure 1.3:Activity diagram of doctor management

Admin has provided valid login details and log into system. And also the main CURD functions related to adding a new doctor, removing a doctor and updating a doctor's details in the system.

1.3. User case diagram



Usese

Figure 1.4:Use Case Diagram of doctor management

This shows the functionality of the system with actors and use cases. The actor of this system is admin, as he/she can manage adding a new doctor, removing a doctor and updating a doctor's details.

1.4. Flow Chart

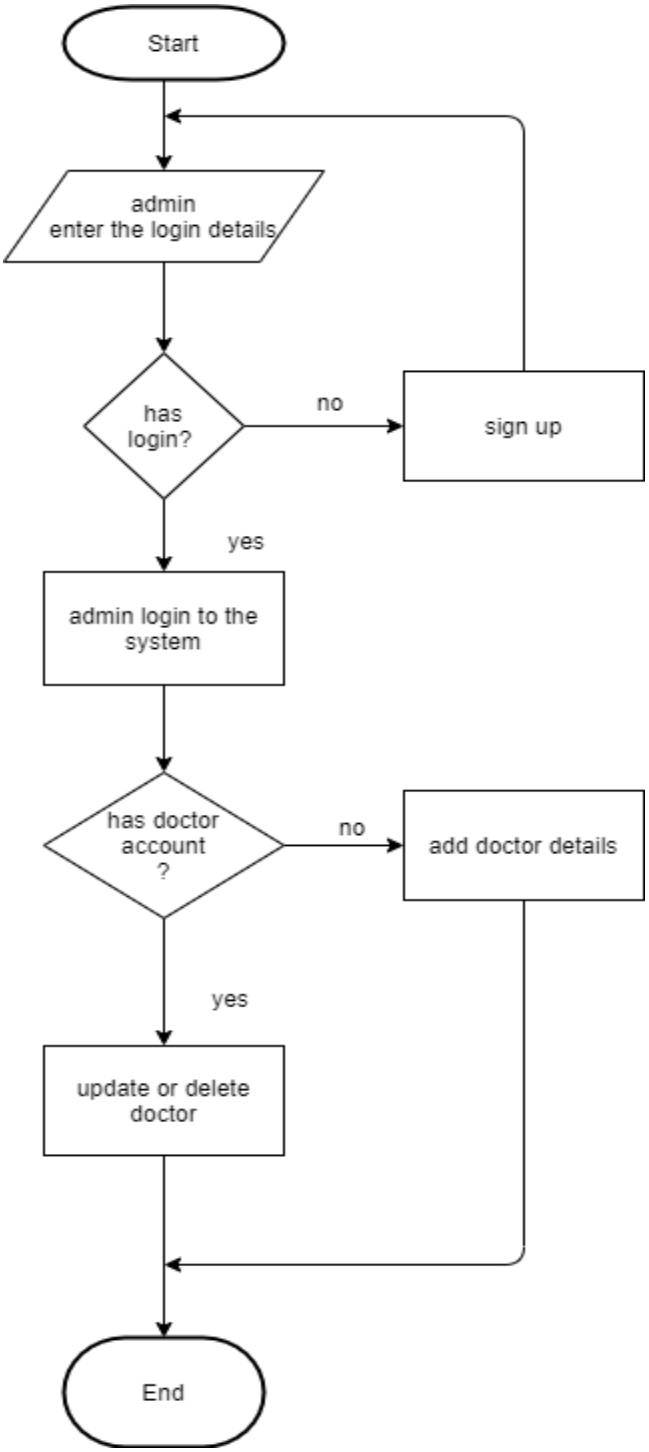


Figure 1.5:System flow chart

Overall doctor management system flow chart. This contain the overall doctor management web service workflow.

Database

1.5. ER diagram

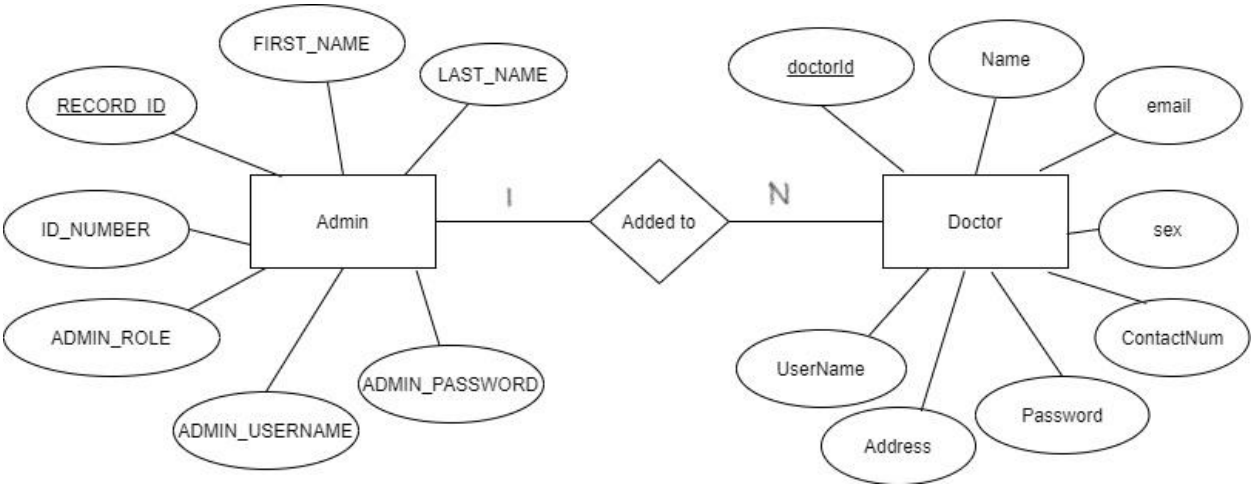


Figure 1.6:ER diagram of doctor management

Main entity types and the relationship between the entities related to identified service by ER diagram. Here mainly two tables are created.

API- Appointment Management

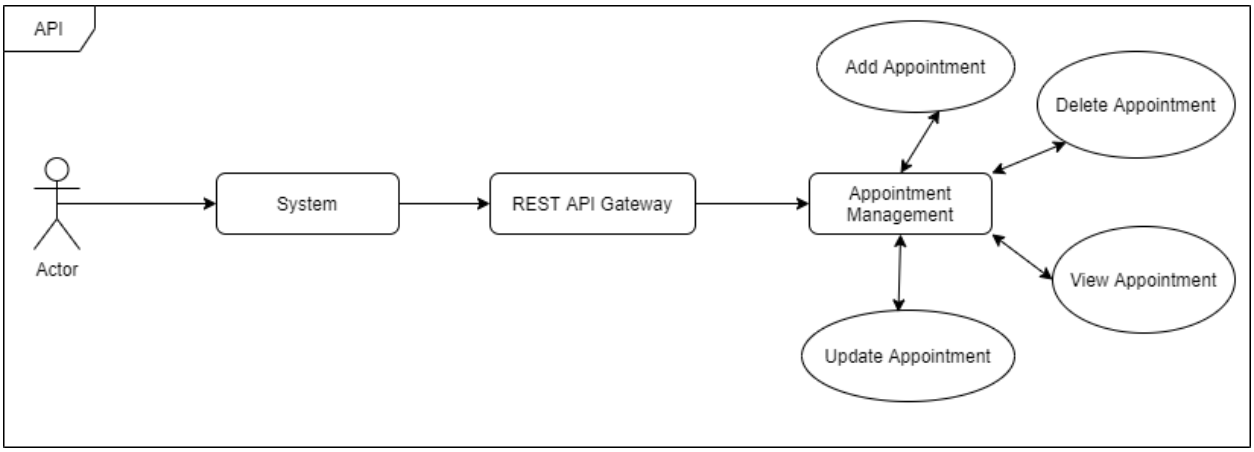


Figure 2.1:API diagram of appointment management

Making Appointments is done by the patient. It involves patient details, available appointments of a specific doctor, available times and dates for appointments and fees that are provided by the hospital and doctor. patient can cancel and update appointments in any time.

The API diagram shows how appointment management web service is exposed to users via a RESTful API.

Service Design

2. Internal Logic

2.1. Class Diagram

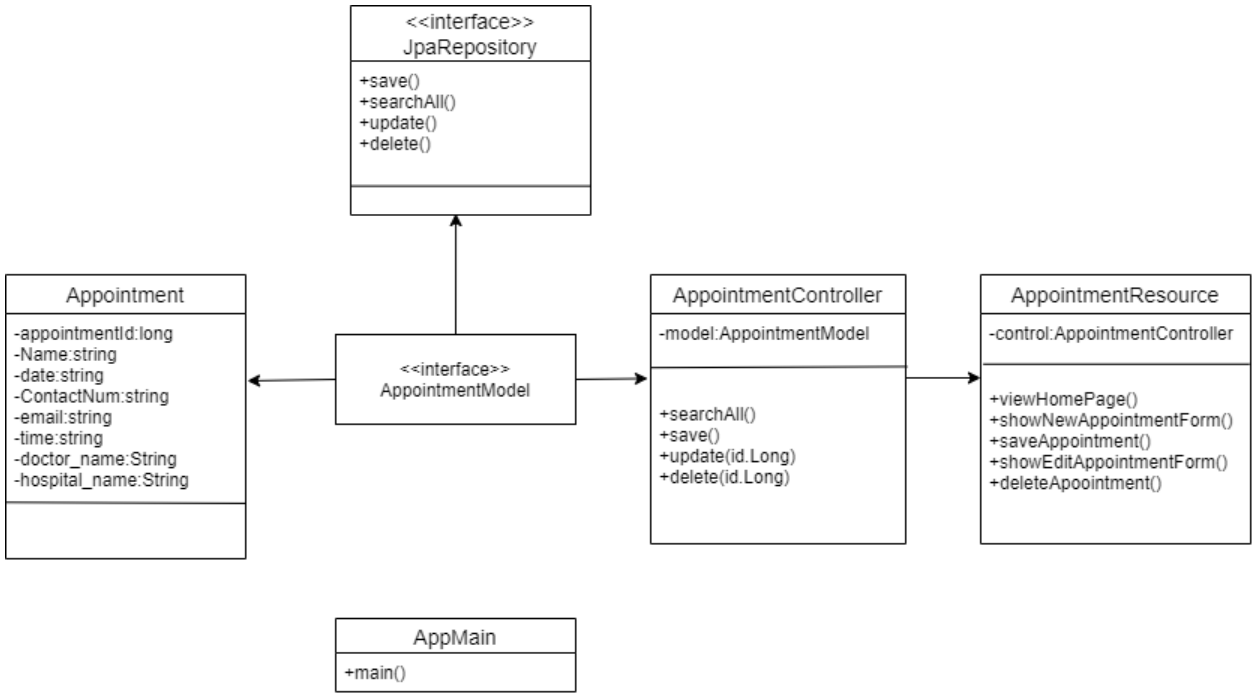


Figure2.1.: Class Diagram of appointment management

This class diagram shows the whole classes, attributes, operations and relationships between objects related to the appointment management service is modelled here. Here we use styles and pattern like MVC.

2.2. Activity Diagram

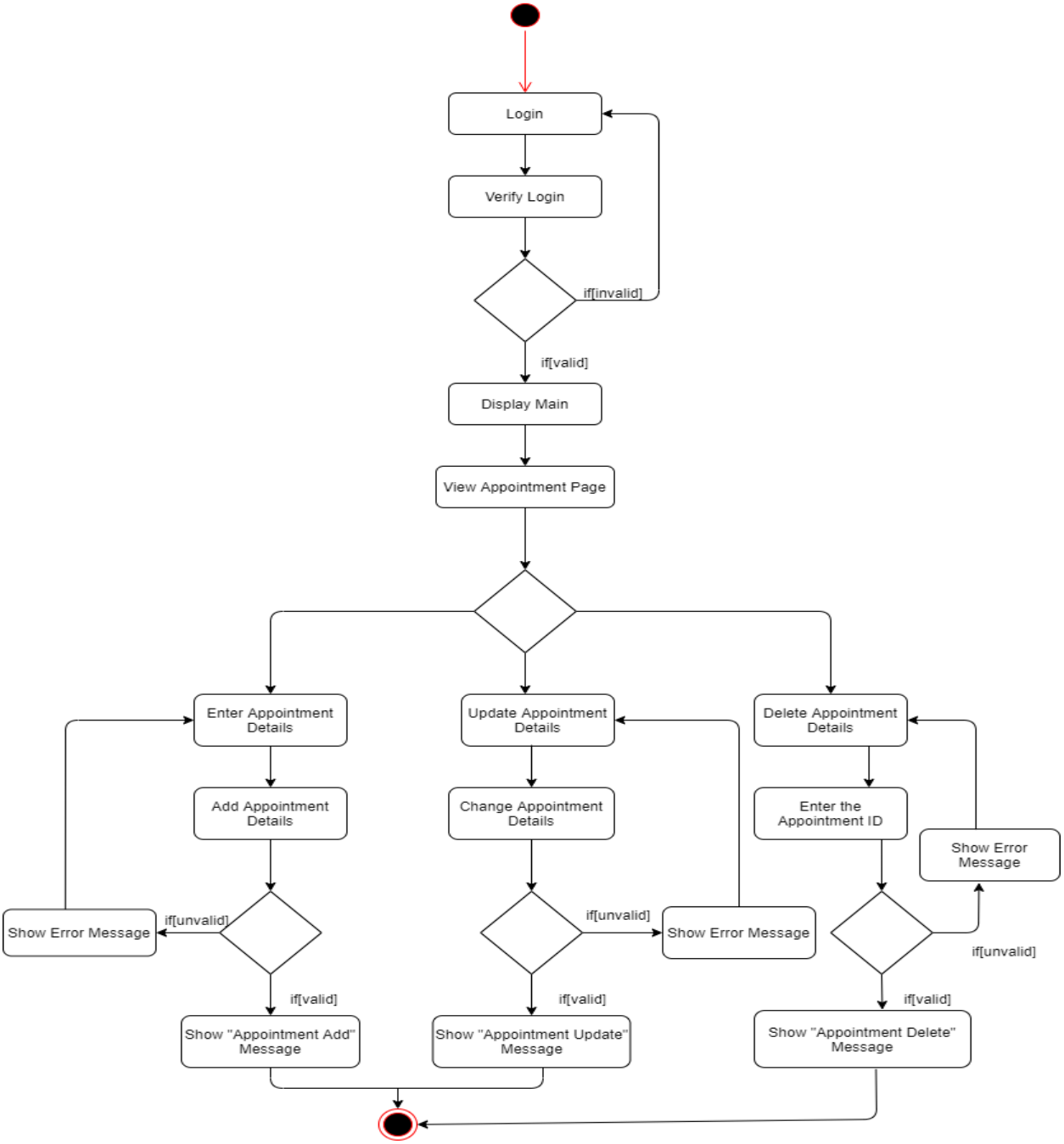
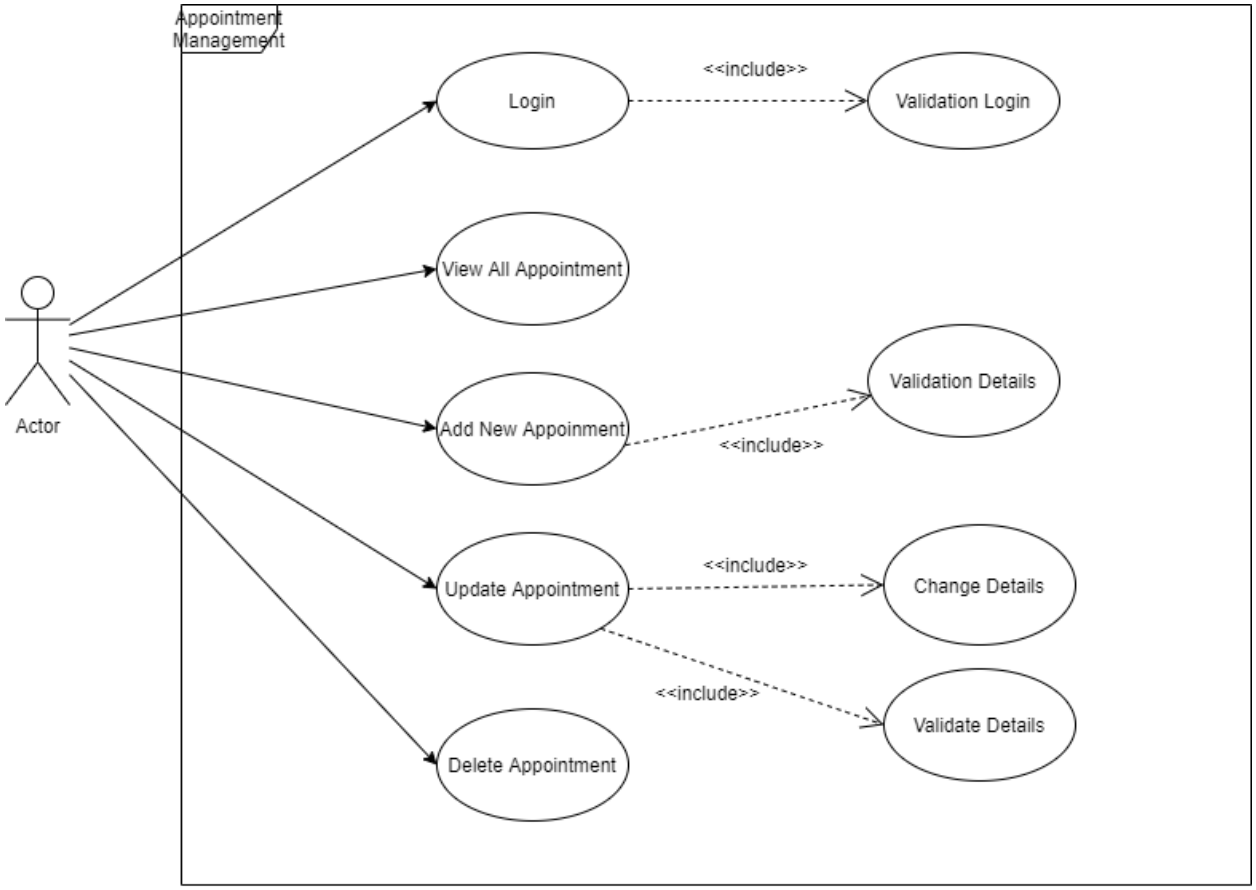


Figure 2.2:Activity diagram of appointment management

Patient has provided valid login details and log into system. And also the main CURD functions related to adding a new appointment, removing an appointment and updating an appointment details in the system.

2.3. User case diagram



Usese

Figure 2.3:UserCase Diagram of appointment management

This shows the functionality of the system with actors and use cases. The actor of this system is patient, as he/she can manage adding a new appointment, removing an appointment and updating an appointment details.

2.4. Flow Chart

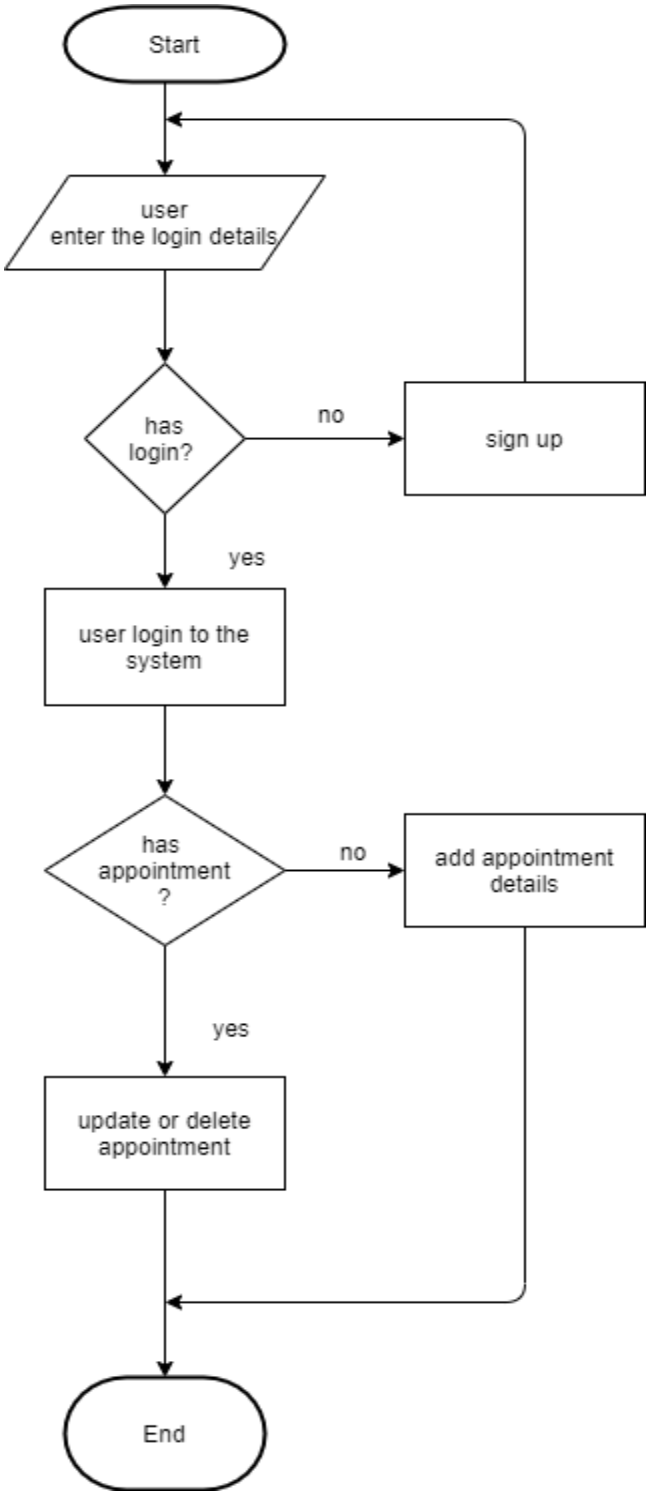


Figure 2.4:System flow chart

Overall appointment management system flow chart. This contain the overall appointment management web service workflow.

Database

2.5. ER diagram

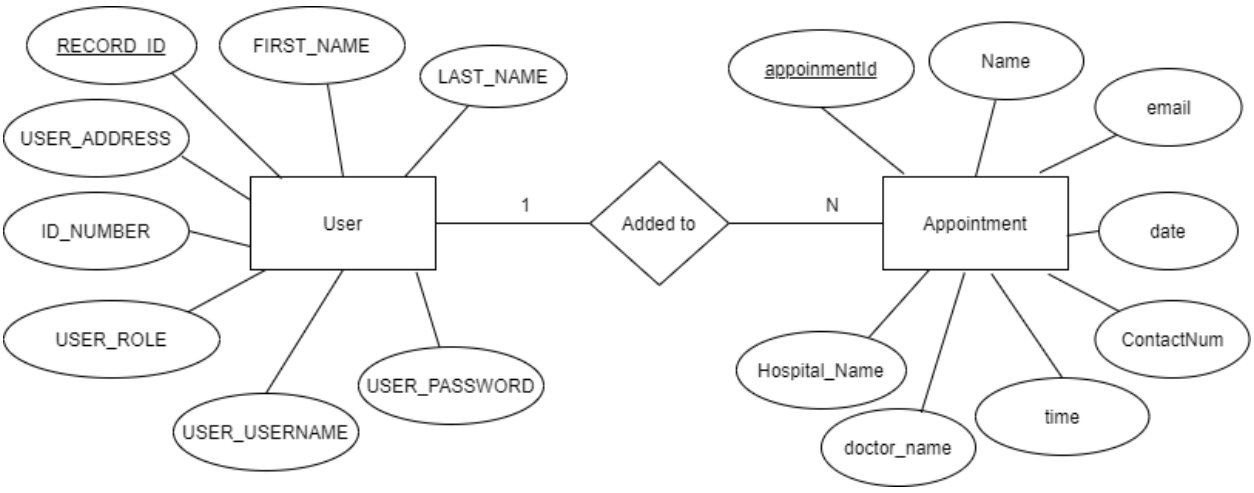


Figure 2.5:ER diagram of appointment management

Main entity types and the relationship between the entities related to identified service by ER diagram. Here mainly two tables are created.

Service Development and testing.

Tools Used

	Tool used	Reason for selection
Back-end	Java - JAX-RS (Jersy)	Jersy provide easy marshalling unmarshalling of XML/JSON data, helping the developers
Server	Tomcat server	Easy configuration
database	MYSQL Phpmyadmin	Can creating database easily
Build tool	Maven	Want to know about various build tools and easy to learn.
IDE	Eclipse IDE	Good development IDE
Testing	Postman	Testing tool

Test cases

Doctor Management

Test ID	Test Description	Test inputs(s)	Expected output(s)	Actual Output(s)	Result(pass /fail)
01	Add new doctor	doctorId: name:Dr Rohan Perera userName:rohan88 address:11,church road,Colombo contactNum:0712712984 email:rohan@gmail.com sex:male password:rohan123	New doctor details is added to system.	New doctor details is added to system. “Doctor Saved”	pass
02	Update doctor details.	doctorId: 11 name:Dr Rohan Perera userName:rohan88 address:11,church road,Colombo contactNum:0712712984 email:perera@gmail.com sex:male password:rohan000	Update doctor details using doctor id.	Update doctor details using doctor id “Doctor Updated”	pass
03	Delete doctor details	doctorId: 8	Delete doctor by using doctor id.	Delete doctor by using doctor id. “Doctor Deleted”	pass
04	View all doctor details		View doctor list	View doctor list	pass
05	View doctor details	doctorId:11	View doctor details using doctor id.	View doctor details	pass

				using doctor id.	

Appointment Management

Test ID	Test Description	Test inputs(s)	Expected output(s)	Actual Output(s)	Result(pass/fail)
01	Making a new appointment.	appointmentId: name:Sudaraka Ampemohotti date:2020-04-16 time:0730 doctor_name:Dr Gamage email:sudaraka@gmail.com contactNum:0768779530 hospitalName:Browns Hospital	New appointment is added to system.	New appointment is added to system. “Appointment Saved”	pass
02	Update new appointment	appointmentId: 9 name:Sudaraka Ampemohotti date:2020-04-16 time:0730 doctor_name:Dr Namal email:sudaraka@gmail.com contactNum:0768779530 hospitalName:Browns Hospital	Update appointment details using appointment Id	Update appointment details using appointment Id “Appointment Updated”	pass
03	Delete appointment	appointmentId: 8	Delete appointment by using appointment id.	Delete appointment by using appointment id. “Appointment Deleted”	pass
04	View all appointments details		View appointments list	View appointments list	pass
05	View appointment details	appointmentId:9	View appointment details using appointmentId	View appointment details using appointmentId	pass

References

"Developing RESTful APIs with JAX-RS," YouTube. [Online]. Available: https://www.youtube.com/playlist?list=PLqq-6Pq4lTTZh5U8RbdXq0WaYvZBz2rbn. [Accessed: 28-Mar-2020].

A. Indunil, "Developing Restful API's," YouTube. [Online]. Available: https://www.youtube.com/channel/UC5rDPzKkbYDu5jqnz8Tmmsw. [Accessed: 03-Apr-2020].

"Java Made Easy," YouTube. [Online]. Available: https://www.youtube.com/channel/UCuTAMEhSO2E86W0XOWZa11g/videos. [Accessed: 05-Apr-2020].