Sri Lanka Institute of Information Technology

2020

3rd Year,1st Semester

IT3030 Programing Application and Frameworks

Project Title-HealthCare System

Batch:

Team Number-

Public VCS Repo- https://github.com/Chamika-mac/HealthCare_app/tree/hashara

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

2020/04/15

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: IT17021326

Name: Perera L.P.H

Service: Hospital Management Service

Description:

HealthCare is a hospital management system where the registered users can add a hospital to the healthcare system.

Workload:

I created a hospital management web service.

Hospital registration is done by Admin/user of the system. This process is consisting of adding a new hospital, removing a hospital and updating a hospital's details.

Table of Contents

Introduction

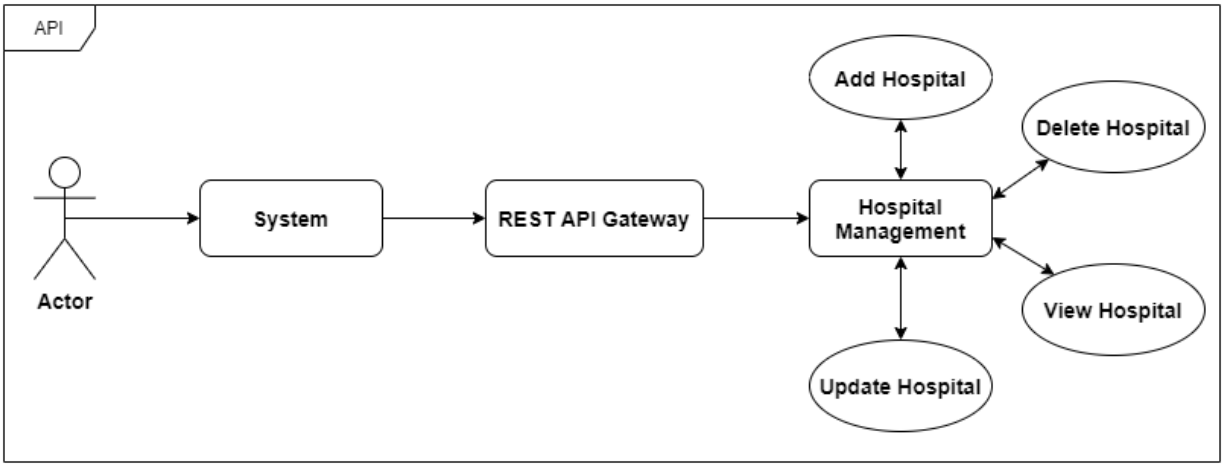
HealthCare is a hospital management system where the registered users can add a hospital to the healthcare system.

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 users. The user should sign up to the use the service and successful signup the user is able to log into the system. And he/she can add a hospital by giving user details. Furthermore, user/admin can delete hospital in any time and update the hospital details.

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

1. API- Hospital Management



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

Appendix

- HospitalResource.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.HospitalModel;

import controller.HospitalController;

@Path("hospitalResources")
public class HospitalResource {

	@GET
	@Path("hospitals")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
	public List<HospitalModel> getAllHospital() throws Exception {
		return HospitalController.getInstance().searchAll();
	}

	@GET
	@Path("hospital/{hospitalId}")
	@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
	public HospitalModel getHospital(@PathParam("hospitalId") int hospitalId)
throws Exception {
		return HospitalController.getInstance().search(hospitalId);
	}

	@POST
```

```
	@Path("hospital")
	public String saveHospital(HospitalModel obj) throws Exception {
		HospitalController.getInstance().save(obj);
		return "Hospital Saved";
	}

	@PUT
	@Path("hospital")
	public String updateHospital(HospitalModel obj) throws Exception {
		HospitalController.getInstance().update(obj);
		return "Hospital Updated";
	}

	@DELETE
	@Path("hospital/{hospitalId}")
	public String deleteHospital(@PathParam("hospitalId") int hospitalId) throws
Exception {
		HospitalModel obj = new HospitalModel();
		obj.setHospitalId(hospitalId);
		HospitalController.getInstance().delete(obj);
		return "Hospital Deleted";
	}

}
```

- HospitalModel.java

```
package com.rest.model;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class HospitalModel {

	private int hospitalId;
	private String hospitalName;
	private String hospitalRegId;
	private String hospitalAddress;
	private String email;
```

```
private String contactNum;
private String userName;
private String password;

public int getHospitalId() {
	return hospitalId;
}
public void setHospitalId(int hospitalId) {
	this.hospitalId = hospitalId;
}
public String getHospitalName() {
	return hospitalName;
}
public void setHospitalName(String name) {
	this.hospitalName = name;
}
public String getHospitalRegId() {
	return hospitalRegId;
}
public void setHospitalRegId(String regId) {
	this.hospitalRegId = regId;
}
public String getHospitalAddress() {
	return hospitalAddress;
}
public void setHospitalAddress(String address) {
	this.hospitalAddress = address;
}
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 getUserName() {
```

```
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}


}
```

- HospitalController.java

```
package controller;

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

import com.rest.model.HospitalModel;

import DBConnector.Connector;

public class HospitalController {

	Connector con = Connector.getInstance();

	private HospitalController() {

	}

	private static final HospitalController ac = new HospitalController();

	public static HospitalController getInstance() {
```

```
		return ac;
	}

	public void save(HospitalModel obj) throws Exception {
		con.getConnection();
		con.aud("INSERT INTO
hospital(hospitalName,hospitalRegId,hospitalAddress,email,contactNum,userName,pass
word) VALUES ('" + obj.getHospitalName()
				+ "', '" + obj.getHospitalRegId() + "', '" +
obj.getHospitalAddress() + "', '" + obj.getEmail() + "', '"
				+ obj.getContactNum() + "', " + "'" + obj.getUserName() +
"', '" + obj.getPassword() + "')");
	}


	public void update(HospitalModel obj) throws Exception {
		con.getConnection();
		con.aud("UPDATE hospital SET hospitalName = '" +
obj.getHospitalName() + "', hospitalRegId = '" + obj.getHospitalRegId() + "',
hospitalAddress = '"
				+ obj.getHospitalAddress() + "'," + "email = '" +
obj.getEmail() + "', contactNum ='" + obj.getContactNum()
				+ "', userName ='" + obj.getUserName() + "', password ='"
+ obj.getPassword() + "' " + "WHERE hospitalId='" + obj.getHospitalId()
				+ "'");
	}

	public void delete(HospitalModel obj) throws Exception {
		con.getConnection();
		con.aud("DELETE FROM hospital WHERE hospitalId='" +
obj.getHospitalId() + "'");
	}

	public List<HospitalModel> searchAll() throws Exception {
		List<HospitalModel> list = new ArrayList<HospitalModel>();
		con.getConnection();
		ResultSet rset = con.srh("SELECT * FROM hospital");
		while (rset.next()) {
			HospitalModel obj = new HospitalModel();
			obj.setHospitalId(rset.getInt(1));
			obj.setHospitalName(rset.getString(2));
```

```
                obj.setHospitalRegId(rset.getString(3));
                obj.setHospitalAddress(rset.getString(4));
                obj.setEmail(rset.getString(5));
                obj.setContactNum(rset.getString(6));
                obj.setUserName(rset.getString(7));
                obj.setPassword(rset.getString(8));

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



        public HospitalModel search(int hospitalId) throws Exception {
            con.getConnection();
            HospitalModel obj = null;
            ResultSet rset = con.srh("SELECT * FROM hospital WHERE hospitalId='" +
hospitalId + "'");
            while (rset.next()) {
                obj = new HospitalModel();
                obj.setHospitalId(rset.getInt(1));
                obj.setHospitalName(rset.getString(2));
                obj.setHospitalRegId(rset.getString(3));
                obj.setHospitalAddress(rset.getString(4));
                obj.setEmail(rset.getString(5));
                obj.setContactNum(rset.getString(6));
                obj.setUserName(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;
	}
}
```

- index.jsp

```
<html>
<body>
<h2>Jersey RESTful Web Application!</h2>
<p><a href="webresources/myresource">Jersey resource</a>
<p>Visit the <a href="http://jersey.java.net">Project Jersey website</a>
for more information on Jersey!
</body>
</html>
```

2. Service Design

2.1. Internal Logic

2.1.1. Class Diagram

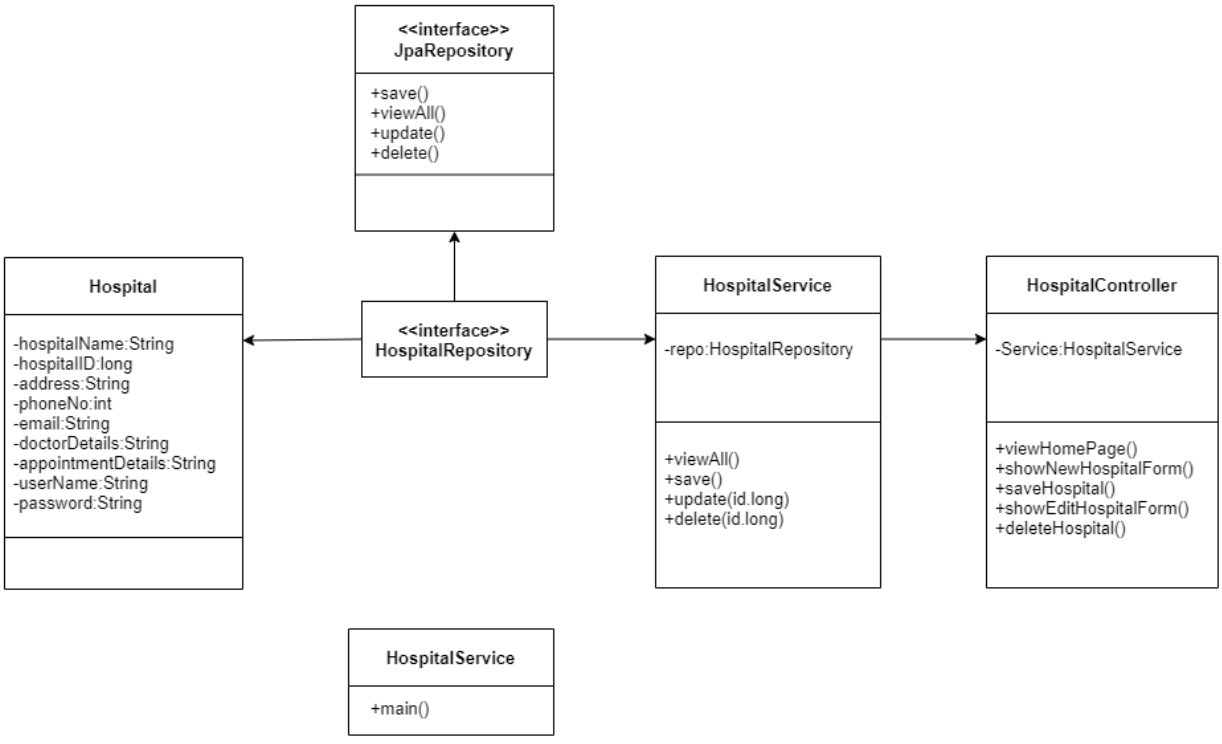


Figure2.1.1: Class Diagram

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

- Activity Diagram

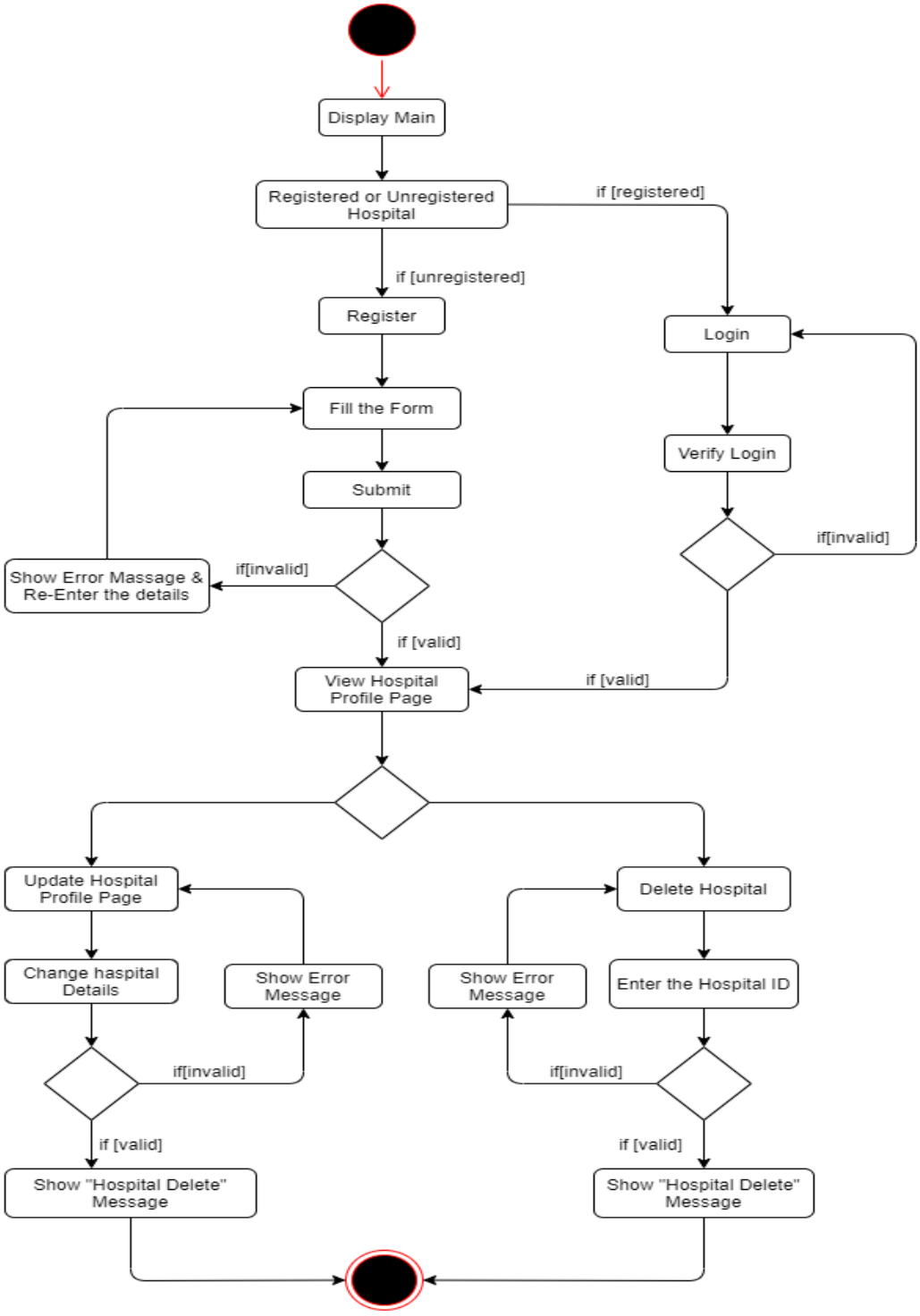


Figure 2.1.2: Activity Diagram

Admin/User need to provide valid login details or they need to register to the hospital management system. They can view the hospital profile page and also the main CURD functions related to edit the hospital details and delete the hospital in the system.

- User case diagram

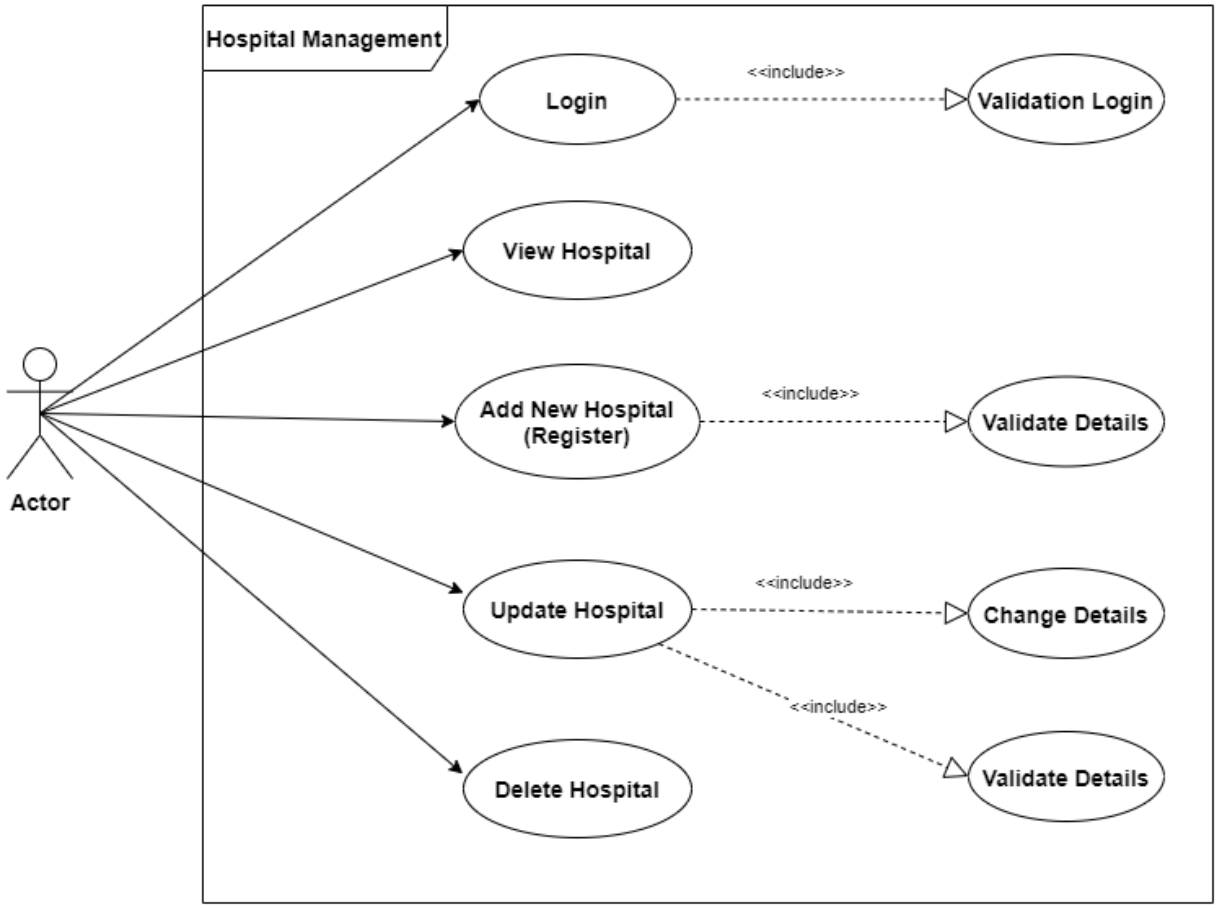


Figure 2.2.3: Use Case Diagram

This shows the functionality of the system with actors and use cases. The actor of this system is admin or user, as he/she can update, delete or view hospital details from the hospital management system.

- Flow Chart

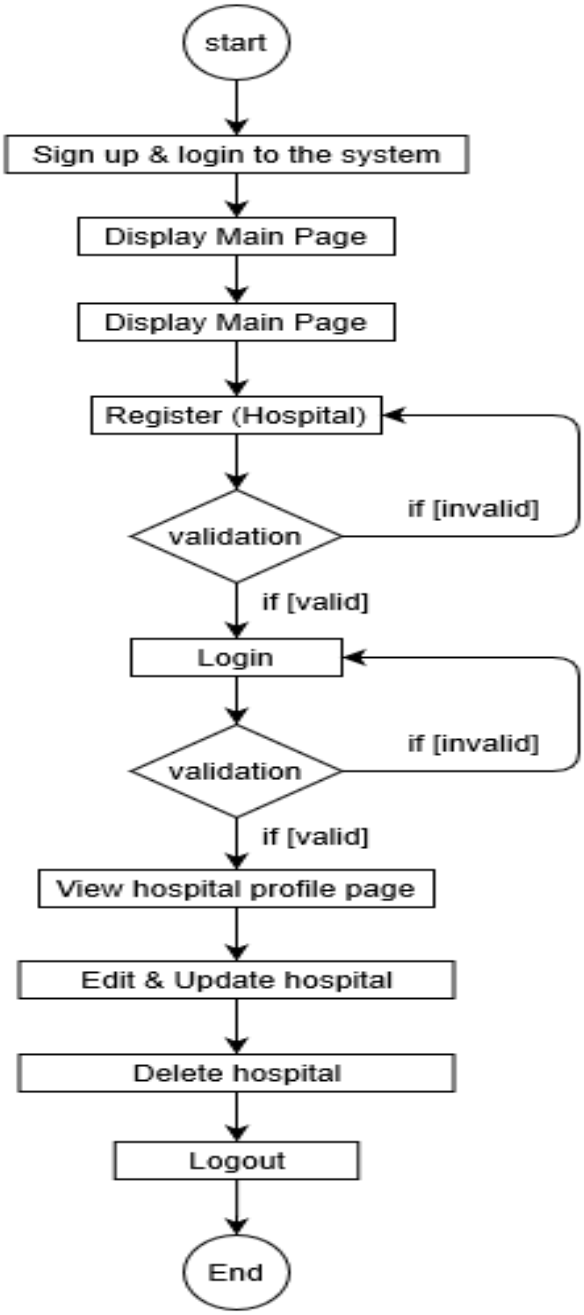


Figure 2.2.4: System flow chart

Overall Hospital Management system flow chart. This contain the overall project management web service workflow.

3. Database

3.1: ER diagram

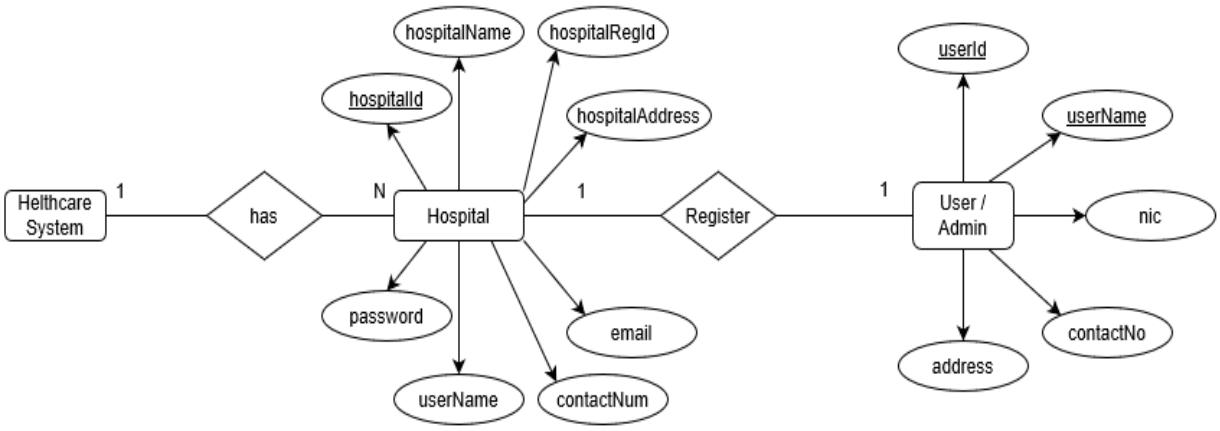


Figure 3.2: ER diagram

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

4. Service Development and testing.

4.1.Tools Used

	Tool used	Reason for selection
Back-end	Java - JAX-RS (Jersy)	Development part is easy.
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

4.2:Test cases

Test ID	Test Description	Test inputs(s)	Expected output(s)	Actual Output(s)	Result (pass/fail)
01	Add new hospital	hospitalId: hospitalName: Nawaloka hospitalRegId: abc1979 hospitalAddress: 34 Colombo contactNum: 0112444037 email: nawaloka@gmail.com userName: nawaloka79 password: nawaloka1979	New hospital is added to system.	New hospital is added to system.	pass
02	Update hospital	hospitalId: 1 hospitalName: Nawaloka hospitalRegId: abc1979 hospitalAddress: 34 Colombo contactNum: 0112444037 email: nawaloka@gmail.com userName: nawaloka79 password: abc111111111	Update hospital details using hospital id.	Update hospital details using hospital id.	pass
03	Delete hospital	hospitalId: 1 hospitalName: Nawaloka hospitalRegId: abc1979 hospitalAddress: 34 Colombo contactNum: 0112444037 email: nawaloka@gmail.com userName: nawaloka79 password: abc111111111	Delete hospital by using hospital id.	Delete hospital by using hospital id.	pass
04	View hospital list		View hospital list	View hospital list	pass

5. References

https://www.youtube.com

https://www.javatpoint.com/phpmyadmin

https://howtodoinjava.com/jersey/jersey-restful-client-examples/

https://www.tutorialspoint.com/maven/