

INTRODUCTION

The project Hospital Management System includes registration of patients, storing their details into the system. The software has the facility to give a unique ID for every patient and stores the details of every patient and the staff automatically. Receptionist can search availability of a doctor and the details of a patients using the ID. The Hospital Management System can be entered using a username and password. It is accessible either by an administrator or receptionist. Only they can add data into the databases. The data can be retrieved easily.

The interface is very user-friendly. The data are well protected for personal use and makes the data processing very fast. Hospital Management System is a powerful, flexible, and easy to use and is designed for multi-specialty hospitals, to cover a wide range of hospital administration and management processes. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration in a seamless flow.

Hospital Management System is a software product suite designed to improve the quality and management of hospital management in the areas of clinical process analysis and activity-based costing. Hospital Management System enables you to develop your organization and improve its effectiveness and quality of work. Managing the key processes efficiently is critical to the success of the hospital helps you manage your processes.

1.1.RELATED WORK

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous papers forms, with data Stores spread throughout the hospital management infrastructure. Often information (On forms) is incomplete or does not follow management system.

Forms are often lost in transit between departments requiring a compressive Auditing process to ensure that no vital information is lost. Multiple copies of the same

Information exists in the hospital and may lead to inconsistency in data in various data stores.

1.2. PROPOSED WORK

The Hospital Management System is designed for Any Hospital to replace their Existing manual, paper-based system. The new system is to control the following information: Patient information, User information, Doctors information, Patients Prescription. These Services are to be provided in an efficient, cost effective manner, with the goal of reducing the time and resources currently required for such tasks.

1.3.OBJECTIVES OF THE WORK

- 1) More efficient, easy for retrieval data and error free.
- 2) Recording information about the patients.
- 3) Recording information about the doctors.
- 4) Recording information related to diagnosis given to Patients.
- 5) Keeping information about various diseases and medicines available to cure them.

These are the various jobs that need to be done in a Hospital by the operational staff and Doctors. All these works are done on numerous paper forms.

1.4. KEY FEATURES

- 1) Information about Patients is done by just writing the Patients name, age and gender. Whenever the patient comes up his information is stored freshly.
- 2) Diagnosis information to patients is generally recorded on the document, which contains Patient information. It is destroyed after some time period to decrease the paper load in the office.
- 3) Information about various diseases is not kept as any document. Doctors themselves do this job by remembering various medicines.

All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and take care of Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them at that time.

1.5. ORGANIZATION OF THE PROJECT

This project report has been broadly divided into six chapters:

- CHAPTER1 gives purpose of system, project motivation, problem statement, methodology used.
- CHAPTER2 requirement specification of the project.
- CHAPTER3 gives the system design of the project.
- CHAPTER4 gives the implementation details of the project.
- CHAPTER5 gives the results of the project.

REQUIREMENT SPECIFICATION

2.1. System Analysis

System analysis will be performed to determine if it is feasible to design information based on policies and plans of the organization and on user requirements and to eliminate the weakness of the present system.

- The new system should be cost effective.
- To expand management, improve productivity and services.
- To enhance user / system interface.
- To improve equality and usability.
- To upgrade system reliability, availability, flexibility and growth potential.

2.2. Functional Requirement

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are Expected to appointments that day.
- Accepts submission in forms of raw patients, doctors, prescription at submit Point. Performs

analysis of drug inventory, patients and diagnosis treatment.

- Each user using the system shall be uniquely identified by his or her Identification number.

2.3. Non-Functional Requirements

The following are the system requirements for online feedback collection systems.

Hardware Specifications:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware

requirements.

Processor	:	Intel Core & AMD
Processor Speed	:	2.5 GHz to 2.8 GHz
RAM	:	8 GB RAM
Hard Disk	:	1 TB

Software

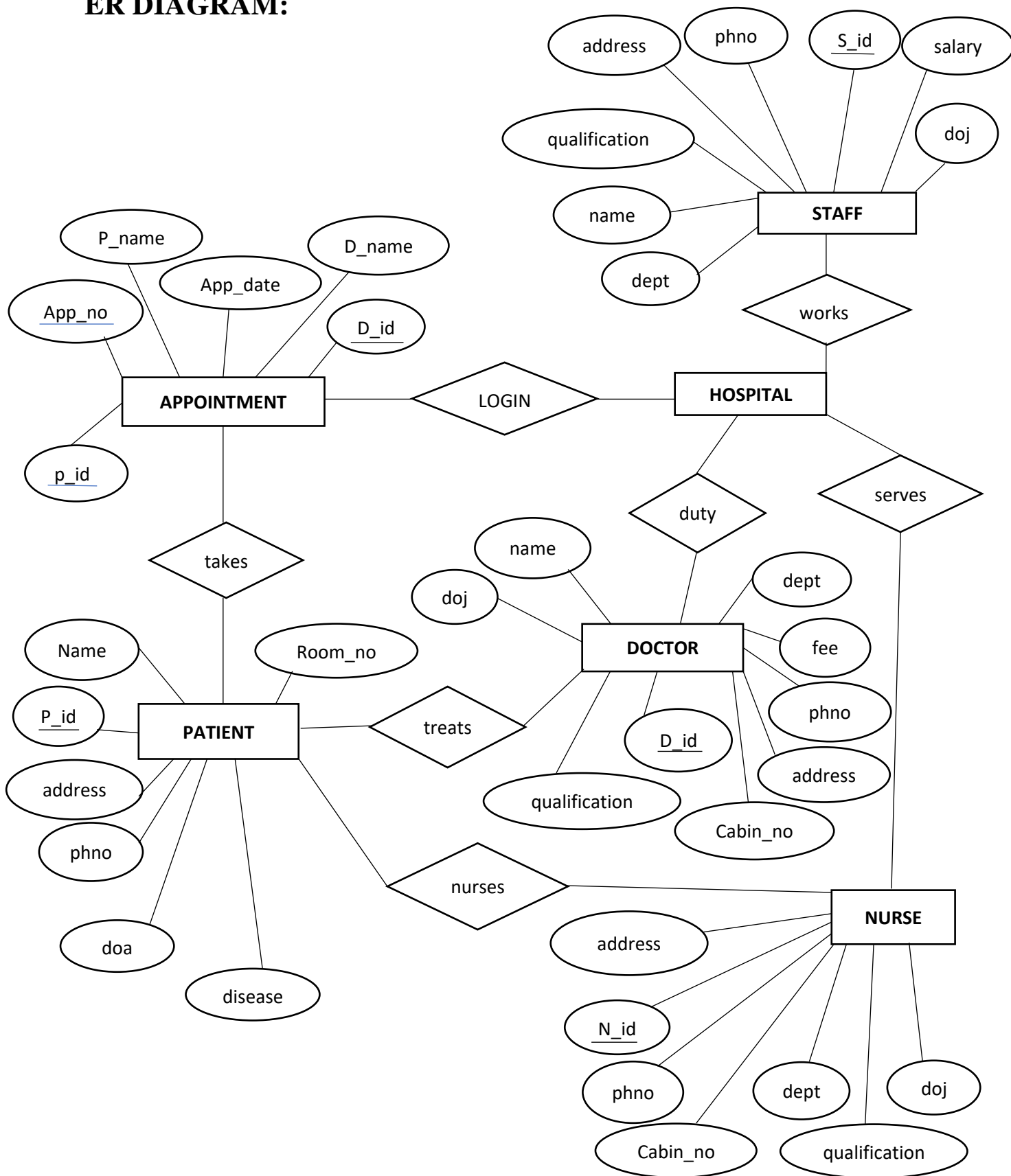
Specifications:

Software requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installed package and need to be installed separately before the software is installed.

Operating System	:	Windows 10
Database	:	MySQL
Front End	:	IntelliJ (Java Swing), WindowBuilder Pro
Server Side	:	JDBC

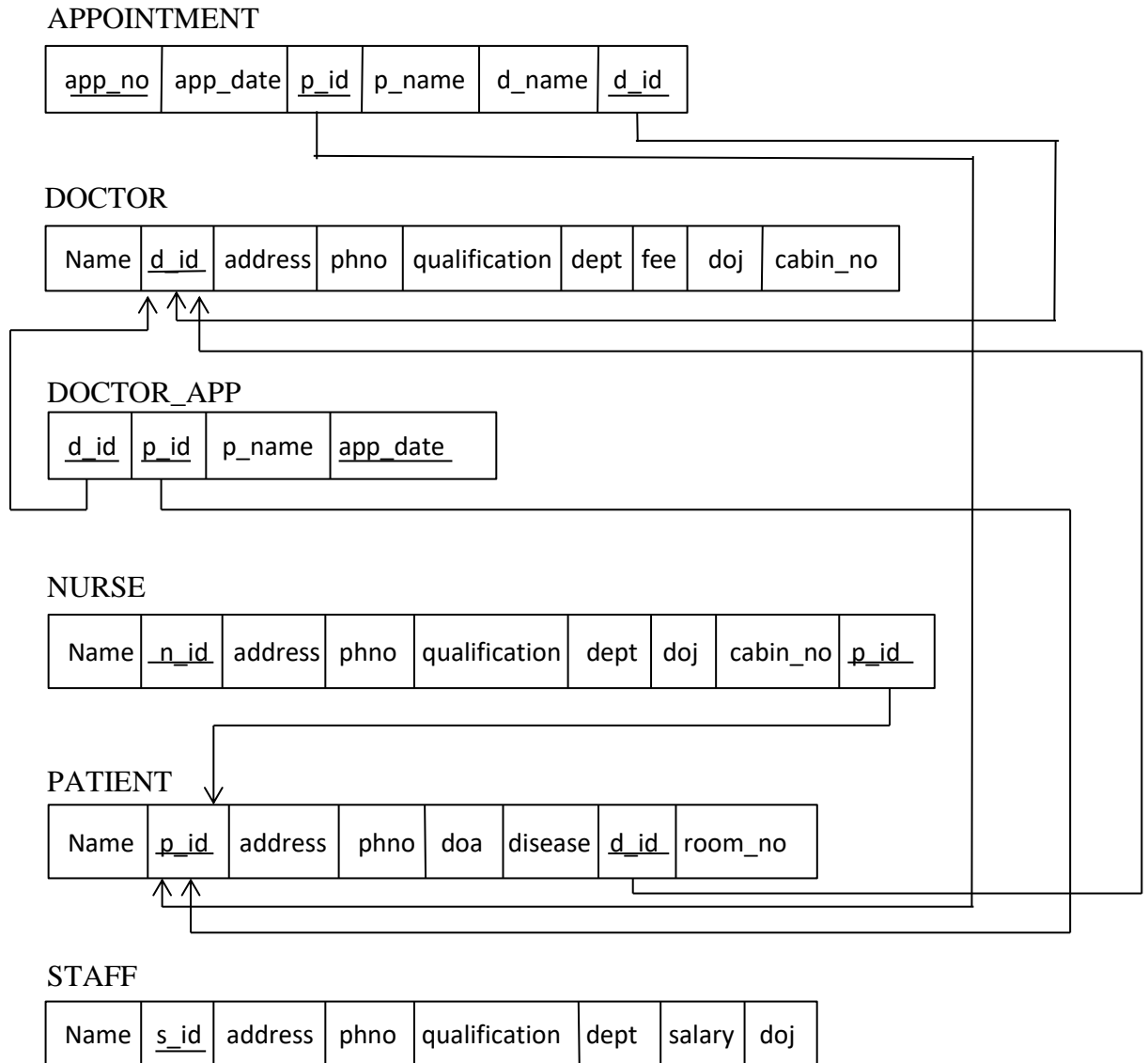
CHAPTER 3

ER DIAGRAM:



This project has been developed using MySQL software which is queries oriented. Changes at the queries and the way in which it uses a system states may causes anticipated changes in the behavior of other result.

- **Schema Diagram**



3.3 Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly and deletion anomaly.

1. **Update anomalies** – If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old several places, a few instances leave the database in an inconsistent state.
2. **Deletion anomalies** – When tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
3. **Insert anomalies** – when tried to insert data in a record that does not exist at all.

Normalization forms:

1. First Normal Form

First Normal form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an domain are invisible units.

Let's take table Admin(Receptionist) and User(Nurse, Doctor).

APPOINTMENT

app_no	app_date	p_id	p_name	d_name	d_id
--------	----------	------	--------	--------	------

- There is no multivalued attribute in any of this tables so it is 1NF

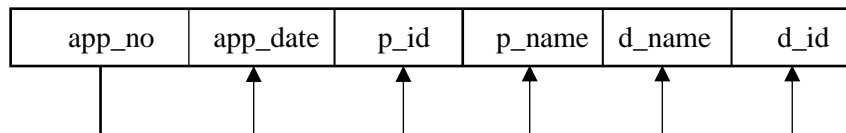
2. Second Normal Form

Before learning about the second normal form, need to understand the following

1. **Prime attribute** – An attribute, which is a part of the candidate- key, is known as a prime attribute.
2. **Non-prime attribute** – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If followed second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X , for which $Y \rightarrow A$ also holds true. **Partial dependency** is not allowed in 2NF.

APPOINTMENT



Let's take tables Admin and User
 Since there are no partial dependency attributes in the above tables, they are said to be in 2NF.

3. Third Normal Form

For a relation to be in 3NF, it must be in 2NF and the following must satisfy-

1. No non-prime attribute is transitively dependent on prime key attribute.
2. For any non-trivial functional dependency, $X \rightarrow A$, then either-
 - X is a super key or,
 - A is a prime attribute, so there not exist a **transitive dependency**.

For Admin and user tables:

From 2NF, it's clear that there are no partial dependency attributes in the above tables.

As per rules of 3NF No transitive keys so they are said to be in 3NF.

IMPLEMENTATIONS

1. Login.java

```
String username = textField.getText();
String password = new String(passwordField.getPassword());
if (username.equals("admin") && password.equals("admin@123")) {
    JOptionPane.showMessageDialog(null, "Login Successful." + "\n" + "\n" +
        "Welcome to Hospital Management System." + "\n" + "Please click OK to
        proceed.", "Login Successful", JOptionPane.NO_OPTION);
    textField.setText(null);
    passwordField.setText(null);
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Activities().setVisible(true);
        }
    });
} else if (username.equals("doctor") && password.equals("doctor@123")) {
    JOptionPane.showMessageDialog(null, "Login Successful." + "\n" + "\n" +
        "Welcome Doctor, Have a Nice Day" + "\n" + "Please click OK to proceed.",
        "Login Successful", JOptionPane.NO_OPTION);
    textField.setText(null);
    passwordField.setText(null);
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            new DoctorLogin().setVisible(true);
        }
    });
} else {
    JOptionPane.showMessageDialog(null, " Either Username Or Password Is
    Incorrect! " + " Please Try Again. ", " Login Failed ",
    JOptionPane.NO_OPTION);
    textField.setText(null);
    passwordField.setText(null);
}
```

2. Activities.java

```
JButton jButton1 = new JButton();
jButton1.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\ADD PATIENT.jpg"));
jButton1.setBorder(null);
jButton1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
```

```

        public void run() { new AddPatient().setVisible(true); }
    });
}
});

JButton jButton2 = new JButton();
jButton2.setBorder(null);
jButton2.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\SEARCH PATIENT.jpg"));
jButton2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new SearchPatient().setVisible(true); }
        });
    }
});

JButton jButton3 = new JButton();
jButton3.setBorder(null);
jButton3.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\ADD DOCTOR.jpg"));
jButton3.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new AddDoctor().setVisible(true); }
        });
    }
});

JButton jButton4 = new JButton();
jButton4.setBorder(null);
jButton4.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\SEARCH DOCTOR.jpg"));
jButton4.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new SearchDoctor().setVisible(true); }
        });
    }
});

JButton jButton5 = new JButton();
jButton5.setBorder(null);
jButton5.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\ADD NURSE.jpg"));
jButton5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new AddNurse().setVisible(true); }
        });
    }
});

```

```

        });
    }
});

JButton jButton6 = new JButton();
jButton6.setBorder(null);
jButton6.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\SEARCH NURSE.jpg"));
jButton6.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new SearchNurse().setVisible(true); }
        });
    }
});

JButton jButton7 = new JButton();
jButton7.setBorder(null);
jButton7.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\ADD STAFF.jpg"));
jButton7.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new AddStaff().setVisible(true); }
        });
    }
});

JButton jButton8 = new JButton();
jButton8.setBorder(null);
jButton8.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\SEARCH STAFF.jpg"));
jButton8.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {public void run() {new
SearchStaff().setVisible(true); }
        });
    }
});

JButton jButton9 = new JButton();
jButton9.setBorder(null);
jButton9.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\GET APPOINTMENT.jpg"));
jButton9.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new GetAppointment().setVisible(true); }
        });
    }
});

```

```

    }
});

JButton jButton10 = new JButton();
jButton10.setBorder(null);
jButton10.setIcon(new ImageIcon("E:\\IntelliJ\\Fazry's Hospital Management
System\\Images\\activities\\APPOINTMENT STATUS.jpg"));
jButton10.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        EventQueue.invokeLater(new Runnable() {
            public void run() { new
AppointmentStatus().setVisible(true); }
        });
    }
});

```

3. AddDoctor.java

```

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create statement
    statement = connection.createStatement();

    String S = "insert into doctors
(fname,lname,address,phone,qualification,department,fee,date,age) values('" +
textField1.getText() + "','" + textField2.getText() + "','" + textField3.getText() +
 "','" + textField4.getText() + "','" + textField5.getText() + "','" +
textField6.getText() + "','" + textField7.getText() + "','" + textField8.getText() +
 "','" + textField9.getText() + "')";

    statement.executeUpdate(S);

    JOptionPane.showMessageDialog(null, "Record Successfully Inserted");

    textField1.setText("");
    textField2.setText("");
    textField3.setText("");
    textField4.setText("");
    textField5.setText("");
    textField6.setText("");
    textField7.setText("");
    textField8.setText(f.format(date));
    textField9.setText("");

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, ex, "ERROR", 0);
}

```

```
}
```

4. AddNurse.java

```
try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create statement
    statement = connection.createStatement();

    String S = "insert into nurse
    (fname,lname,address,phone,qualification,department,date,ward) values('" +
    textField1.getText() + "','" + textField2.getText() + "','" + textField3.getText() +
    "','" + textField4.getText() + "','" + textField5.getText() + "','" +
    textField6.getText() + "','" + textField7.getText() + "','" + textField8.getText() +
    "')";

    statement.executeUpdate(S);

    JOptionPane.showMessageDialog(null, "Record Successfully Inserted");

    textField1.setText("");
    textField2.setText("");
    textField3.setText("");
    textField4.setText("");
    textField5.setText("");
    textField6.setText("");
    textField7.setText(f.format(date));
    textField8.setText("");

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, ex, "ERROR", 0);
}
```

5. AddPatient.java

```
try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // get connection to database
    statement = connection.createStatement();

    String S = "insert into patients
    (fname,lname,address,phone,date,disease,age,sex) values('"+
    textField1.getText() + "','" + textField2.getText() + "','" + textField3.getText() +
    "','" + textField4.getText() + "','" + textField5.getText() + "','" +
```

```

textField6.getText() + "','" + textField7.getText() + "','" + textField8.getText() +
    "');"

statement.executeUpdate(S);

JOptionPane.showMessageDialog(null, "Record Successfully Inserted");

textField1.setText("");
textField2.setText("");
textField3.setText("");
textField4.setText("");
textField5.setText(f.format(date));
textField6.setText("");
textField7.setText("");
textField8.setText("");

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, ex, "ERROR", 0);
}

```

6. AddStaff.java

```

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create statement
    statement = connection.createStatement();

    String S = "insert into nurse
(fname,lname,address,phone,qualification,department,date,ward) values('" +
    textField1.getText() + "','" + textField2.getText() + "','" + textField3.getText() +
    "','" + textField4.getText() + "','" + textField5.getText() + "','" +
    textField6.getText() + "','" + textField7.getText() + "','" + "');"

    statement.executeUpdate(S);

    JOptionPane.showMessageDialog(null, "Record Successfully Inserted");

    textField1.setText("");
    textField2.setText("");
    textField3.setText("");
    textField4.setText("");
    textField5.setText("");
    textField6.setText("");
    textField7.setText(f.format(date));

} catch (Exception ex) {

```

```

        JOptionPane.showMessageDialog(null, ex, "ERROR", 0);
    }

```

7. GetAppointment.java

```

protected void submitButtonActionPerformed() {
    String App_Date = textField3.getText();
    boolean isValidDate = compareDate(App_Date);
    if (isValidDate) {

        try {
            // get connection to database
            connection = DriverManager.getConnection(dbUrl, username,
password);

            // get statement
            statement = connection.createStatement();

            String S = "insert into appointment
(p_id,d_id,app_date,blood_pressure,blood_sugar,weight,temperature)
values(" + textField1.getText() + "," + textField2.getText() + "," +
textField3.getText() + "," + textField4.getText() + "," +
textField5.getText() + "," + textField6.getText() + "," +
textField7.getText() + ")";

            statement.executeUpdate(S);

            JOptionPane.showMessageDialog(null, "Record Successfully
Inserted");

            textField1.setText("1");
            textField2.setText("2");
            textField3.setText(curr);

        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex, "ERROR", 0);
        }
    } else {
        textField3.setText(curr);
    }
}

private boolean compareDate(String App_Date) {
    // check date format YYYY-MM-dd
    try {
        DateFormat df = new SimpleDateFormat(datePattern);
        df.setLenient(false);
        df.parse(App_Date);
    }
}

```



```

    } catch (ParseException ex) {
        JOptionPane.showMessageDialog(null, " Please enter the date in the
        form of YYYY-MM-DD ", " Invalid Date! ", .NO_OPTION);
        return false;
    }

    // check whether the input date coming on or after the current date
    try {
        Date currentDate = ft.parse(curr);
        Date inputDate = ft.parse(App_Date);
        if (inputDate.compareTo(currentDate) >= 0) {
            return true;
        } else {
            JOptionPane.showMessageDialog(null, " Please enter a date
            on or after " + curr, " Invalid Date! ",
            JOptionPane.NO_OPTION);
            return false;
        }
    }

    } catch (Exception e) { }
    return false;
}

```

8. AppointmentStatus.java

```

String P_ID = textField1.getText();
String DATE = textField2.getText();
String D_ID = null;

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create a statement
    statement = connection.createStatement();

    // execute sql query
    String query1 = "SELECT app_id,d_id FROM appoinment WHERE p_id=" +
    P_ID + " AND app_date=" + DATE + """;
    String query2 = "SELECT fname,lname,disease FROM patients WHERE id=" +
P_ID;

    resultSet = statement.executeQuery(query1);

    if (resultSet.next()) {
        jLabel1.setText(resultSet.getString("app_id"));
        D_ID = resultSet.getString("d_id");
    }
}

```

```

        } else {
            JOptionPane.showMessageDialog(null, "NO APPOINTMENT
FOUND!", "ERROR", 0);
        }

        resultSet = statement.executeQuery(query2);

        if (resultSet.next()) {
            jLabel3.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
            jLabel4.setText(resultSet.getString("disease"));
        }

        String query3 = "SELECT fname,lname FROM doctors WHERE id=" + D_ID;

        resultSet = statement.executeQuery(query3);

        if (resultSet.next()) {
            jLabel2.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
        }

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "DATABASE NOT CONNECTED!",
"ERROR", 0);
    }
}

```

9. SearchDoctor.java

```

String ID = textField1.getText();
try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create a statement
    statement = connection.createStatement();

    // execute query
    String query = "SELECT fname,lname,phone,qualification,department,fee
FROM doctors WHERE id=" + ID;

    resultSet = statement.executeQuery(query);

    if (resultSet.next()) {
        jLabel1.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
        jLabel2.setText(resultSet.getString("phone"));
        jLabel3.setText(resultSet.getString("qualification"));
    }
}

```

```

        jLabel4.setText(resultSet.getString("department"));
        jLabel5.setText(resultSet.getString("fee"));

    } else {
        JOptionPane.showMessageDialog(null, "INVALID ID!", "ERROR",
0);
    }

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, "DATABASE NOT CONNECTED!",
"ERROR", 0);
}

```

10. SearchNurse.java

```

String ID = textField1.getText();

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // get statement
    statement = connection.createStatement();

    // execute sql query
    String query = "SELECT fname,lname,phone,qualification,department,ward
FROM nurse WHERE id=" + ID;

    resultSet = statement.executeQuery(query);

    if (resultSet.next()) {
        jLabel1.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
        jLabel2.setText(resultSet.getString("phone"));
        jLabel3.setText(resultSet.getString("qualification"));
        jLabel4.setText(resultSet.getString("department"));
        jLabel5.setText(resultSet.getString("ward"));

    } else {
        JOptionPane.showMessageDialog(null, "INVALID ID!", "ERROR",
0);
    }

} catch (Exception ex) {

    JOptionPane.showMessageDialog(null, "DATABASE NOT CONNECTED!",
"ERROR", 0);
}

```

11. SearchPatient.java

```
String ID = textField1.getText();

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // get statement
    statement = connection.createStatement();

    // execute sql query
    String query = "SELECT fname,lname,age,disease,phone,address FROM
patients WHERE id=" + ID;

    resultSet = statement.executeQuery(query);

    if (resultSet.next()) {
        jLabel1.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
        jLabel2.setText(resultSet.getString("age"));
        jLabel3.setText(resultSet.getString("disease"));
        jLabel4.setText(resultSet.getString("phone"));
        jLabel5.setText(resultSet.getString("address"));
    } else {
        JOptionPane.showMessageDialog(null, "INVALID ID!", "ERROR",
0);
    }

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, "DATABASE NOT CONNECTED!",
"ERROR", 0);
}
```

12. SearchStaff.java

```
String ID = textField1.getText();

try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create statement
    statement = connection.createStatement();

    // execute sql query
```

```

String query = "SELECT fname,lname,phone,qualification,department FROM
staff WHERE id=" + ID;

resultSet = statement.executeQuery(query);

if (resultSet.next()) {
    jLabel1.setText(resultSet.getString("fname") + " " +
resultSet.getString("lname"));
    jLabel2.setText(resultSet.getString("phone"));
    jLabel3.setText(resultSet.getString("qualification"));
    jLabel4.setText(resultSet.getString("department"));

} else {
    JOptionPane.showMessageDialog(null, "INVALID ID!", "ERROR",
0);
}

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, "DATABASE NOT CONNECTED!",
"ERROR", 0);
}

```

13. DoctorLogin.java

```

String FNAME = null;
String LNAME = null;
String DISEASE = null;
String D_ID = id;

String[] columnNames = { "App_ID", "App_Date", "Patient Name", "Disease", "Blood
Pressure", "Blood Sugar", "Weight", "Temperature" };

DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);
try {
    // get connection to database
    connection = DriverManager.getConnection(dbUrl, username, password);

    // create statement
    String query1 = "SELECT
app_id,p_id,blood_pressure,blood_sugar,weight,temperature,app_date FROM
appointment WHERE d_id=?";
    statement = connection.prepareStatement(query1);
    statement.setString(1, D_ID);
    ResultSet resultSet1 = statement.executeQuery();

    try {
        while (resultSet1.next()) {
            int APP_ID = resultSet1.getInt("app_id");

```

```

        int P_ID = resultSet1.getInt("p_id");
        String PRESSURE = resultSet1.getString("blood_pressure");
        String SUGAR = resultSet1.getString("blood_sugar");
        String WEIGHT = resultSet1.getString("weight");
        String TEMPERATURE =
resultSet1.getString("temperature");
        String DATE = resultSet1.getString("app_date");
        try {
            DriverManager.getConnection(dbUrl, username,
password);
            String query2 = "SELECT fname,lname,disease
FROM patients          WHERE id=?";
            statement = connection.prepareStatement(query2);
            statement.setInt(1, P_ID);
            ResultSet resultSet2 = statement.executeQuery();
            if (resultSet2.next()) {
                FNAME = resultSet2.getString("fname");
                LNAME = resultSet2.getString("lname");
                DISEASE =
resultSet2.getString("disease");
            }

        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, this,
"Error", 0);
        }

        Object[] data = { APP_ID, DATE, FNAME + " " + LNAME,
DISEASE, PRESSURE, SUGAR, WEIGHT,
TEMPERATURE };
        tableModel.addRow(data);
        table.setModel(tableModel);
    }

    } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, this, "Error", 0);
    }

} catch (Exception ex) {
    JOptionPane.showMessageDialog(null, this, "Error", 0);
}
}

```

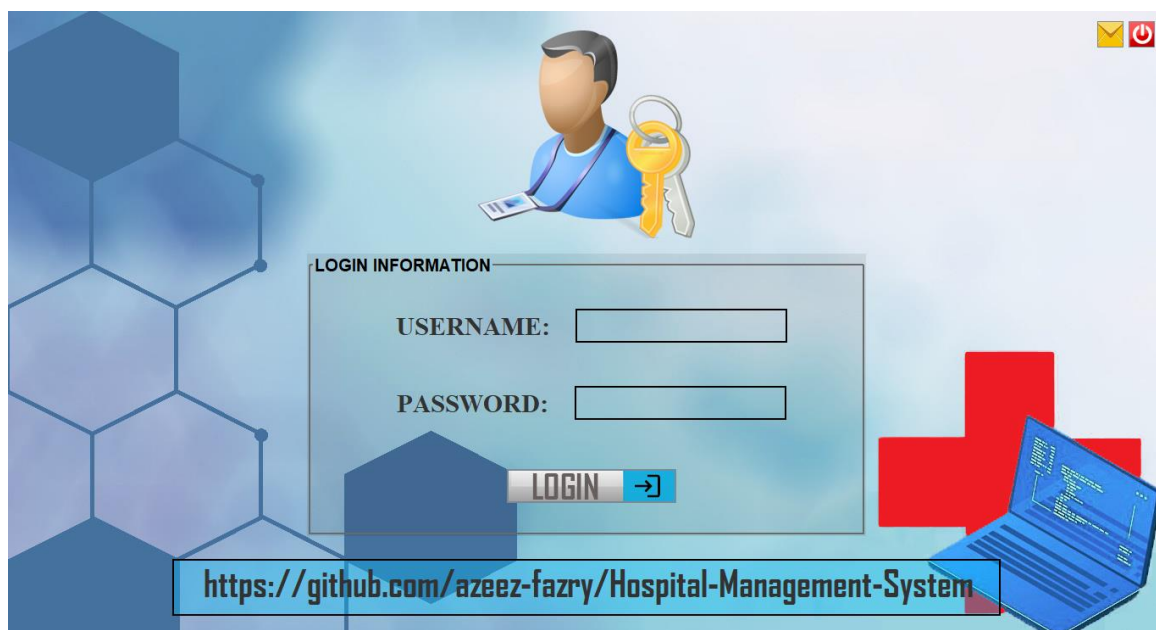
SNAPSHOTS

FRONT END DESIGN:

5.1 Home Page:



5.2 Login Page:



5.3 Admin Page:

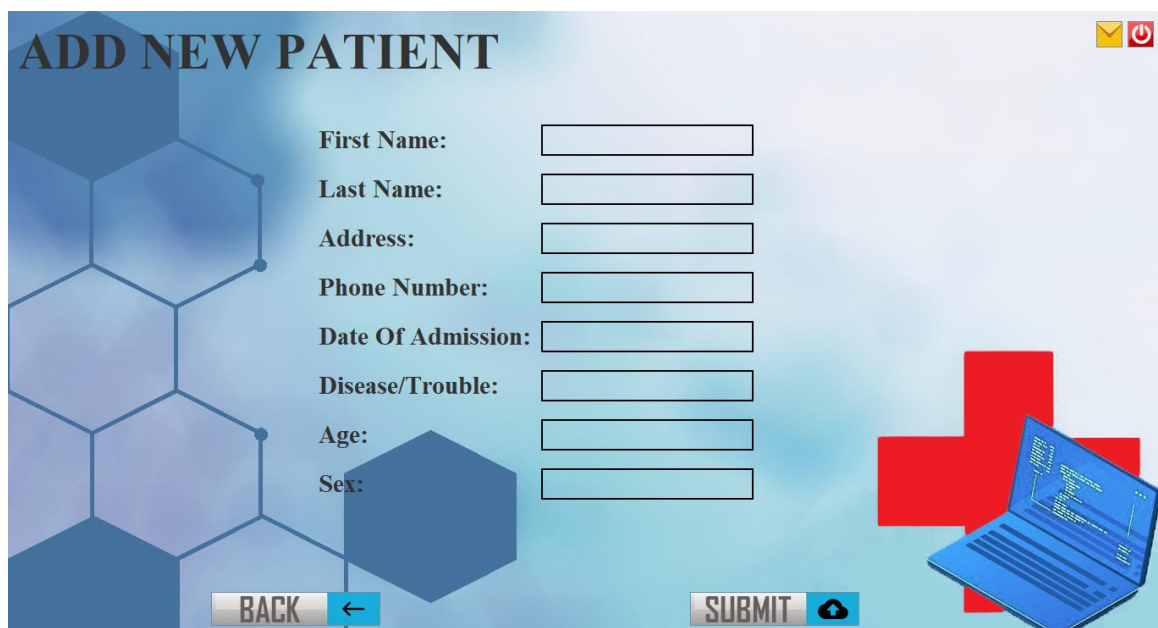


The screenshot displays the Admin Page of the Hospital Management System. On the left, there are five summary cards for different roles: PATIENT (Total: 12), DOCTOR (Total: 12), APPOINTMENT (Total: 10), NURSE (Total: 10), and STAFF (Total: 12). Each card includes an 'ADD' button in red and a 'VIEW' button in blue. The main area features a large title 'HOSPITAL MANAGEMENT SYSTEM' in red and blue, with a red cross graphic. A URL is displayed at the bottom: <https://github.com/azeez-fazry/Hospital-Management-System>. The background has a blue hexagonal pattern and a laptop icon.

Role	Total	ADD	VIEW
PATIENT	12	ADD	VIEW
DOCTOR	12	ADD	VIEW
APPOINTMENT	10	ADD	VIEW
NURSE	10	ADD	VIEW
STAFF	12	ADD	VIEW

<https://github.com/azeez-fazry/Hospital-Management-System>

5.4 Add Patient:



The screenshot shows the 'ADD NEW PATIENT' form. It includes input fields for First Name, Last Name, Address, Phone Number, Date Of Admission, Disease/Trouble, Age, and Sex. There are 'BACK' and 'SUBMIT' buttons at the bottom. The background features a blue hexagonal pattern, a red cross, and a laptop icon.

ADD NEW PATIENT

First Name:

Last Name:

Address:



Phone Number:

Date Of Admission:

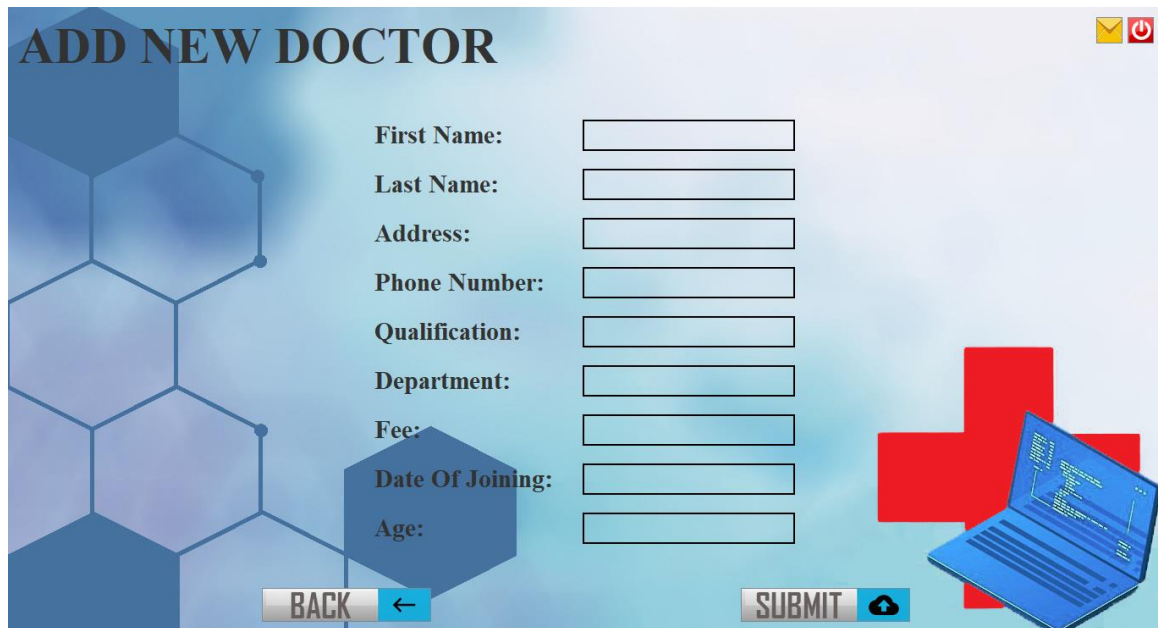
Disease/Trouble:

Age:

Sex:

BACK  **SUBMIT** 

5.5 Add Doctor:



ADD NEW DOCTOR

First Name:

Last Name:

Address:

Phone Number:


Qualification:


Department:

Fee:

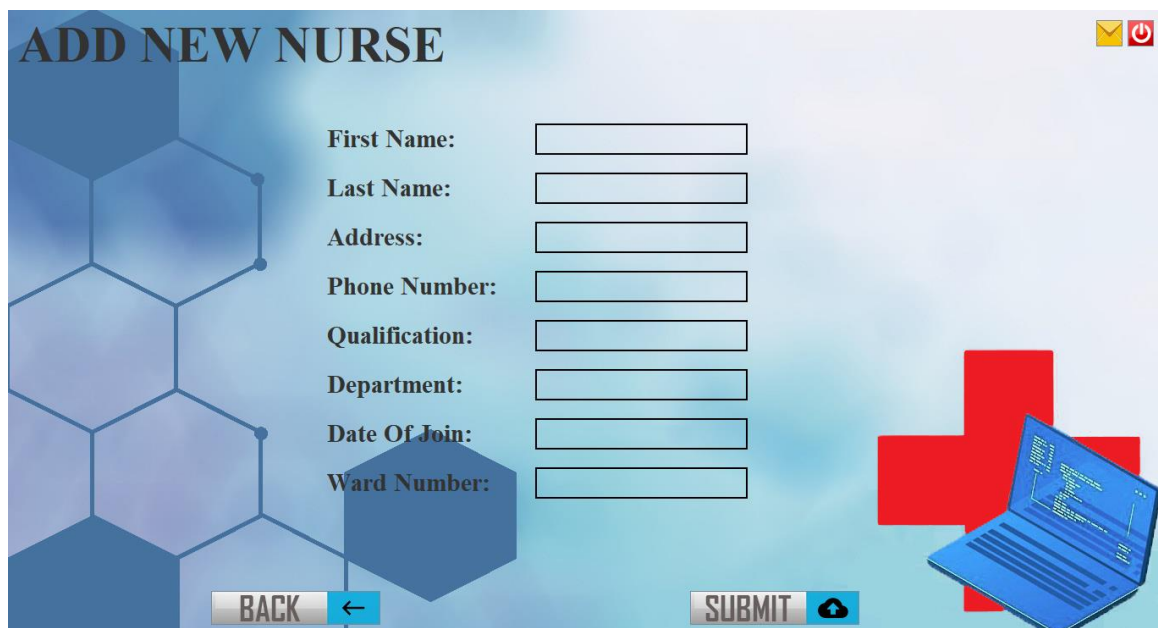
Date Of Joining:

Age:

BACK 

SUBMIT 

5.6 Add Nurse:



ADD NEW NURSE

First Name:

Last Name:

Address:


Phone Number:


Qualification:

Department:

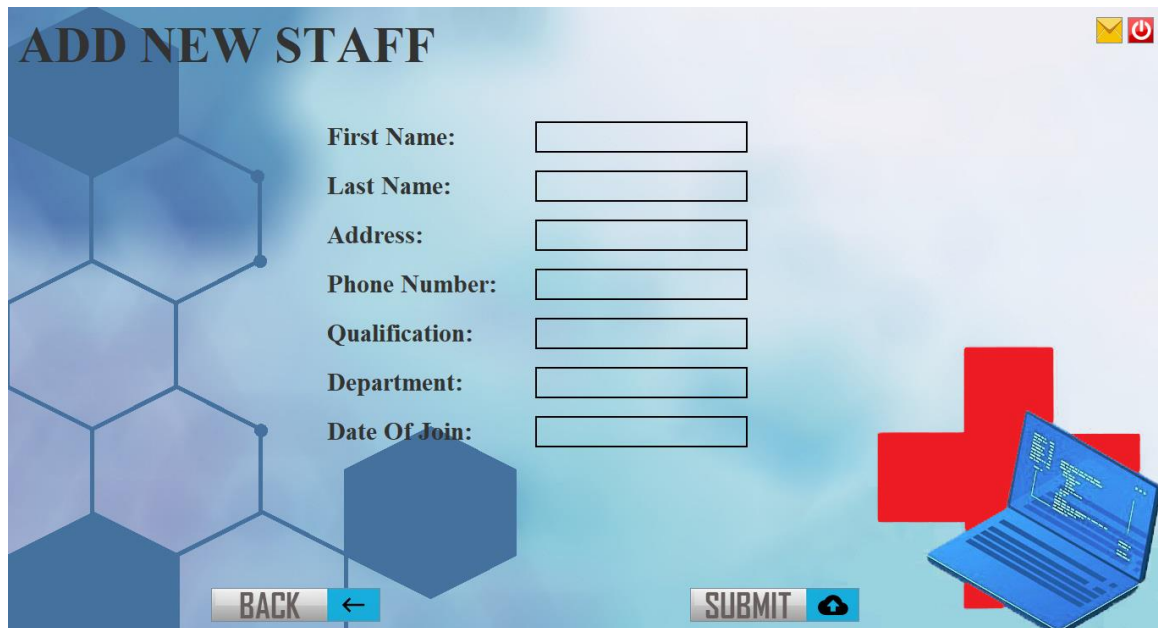
Date Of Join:

Ward Number:

BACK 

SUBMIT 

5.7 Add Staff:



ADD NEW STAFF

First Name:

Last Name:


Address:


Phone Number:

Qualification:

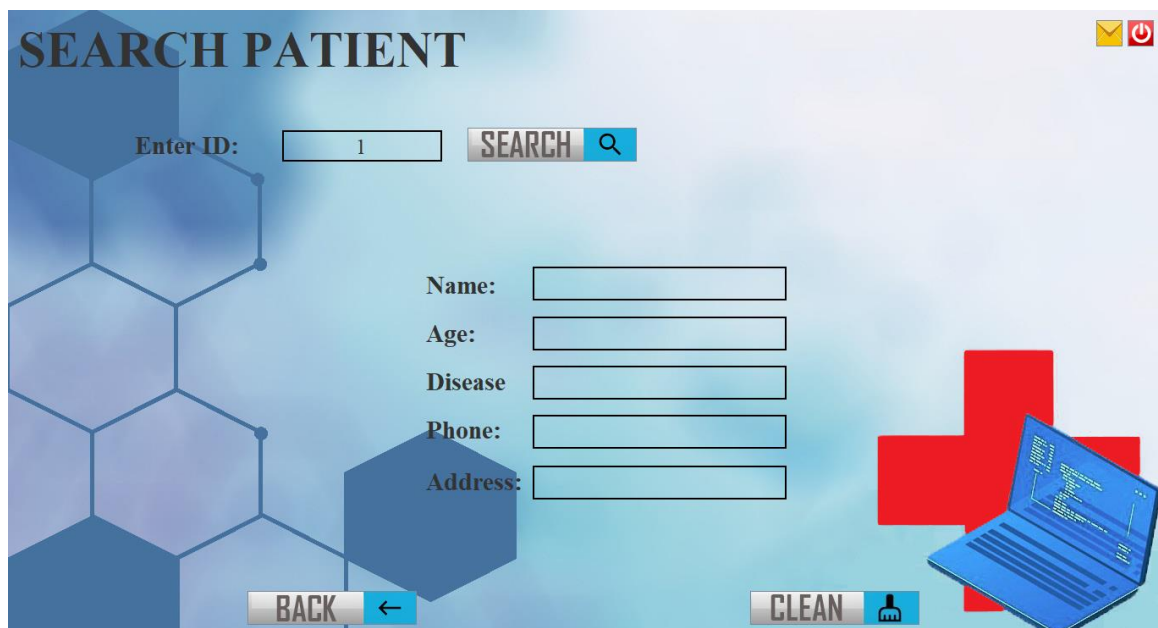
Department:

Date Of Join:


BACK 

SUBMIT 

5.8 Search Patient:



SEARCH PATIENT

Enter ID: **SEARCH** 


Name:


Age:

Disease:

Phone:

Address:

BACK 

CLEAN 

5.9 Search Doctor:

SEARCH DOCTOR

Enter ID: **SEARCH**

Name:

Phone:

Qualification:

Department:

Fee:

BACK **CLEAN**

5.10 Search Nurse:

SEARCH NURSE

Enter ID: **SEARCH**

Name:

Phone:

Qualification:

Department:

Ward Number:

BACK **CLEAN**

5.11 Search Staff:

SEARCH STAFF

Enter ID: **SEARCH** 🔍

Name:

Phone:

Qualification:

Department:

BACK ← **CLEAN** 🗑️

5.12 Get New Appointment:

GET NEW APPOINTMENT

PATIENT INFORMATION

Patient ID:

Blood Pressure:

Blood Sugar:

Weight:

Temperature:

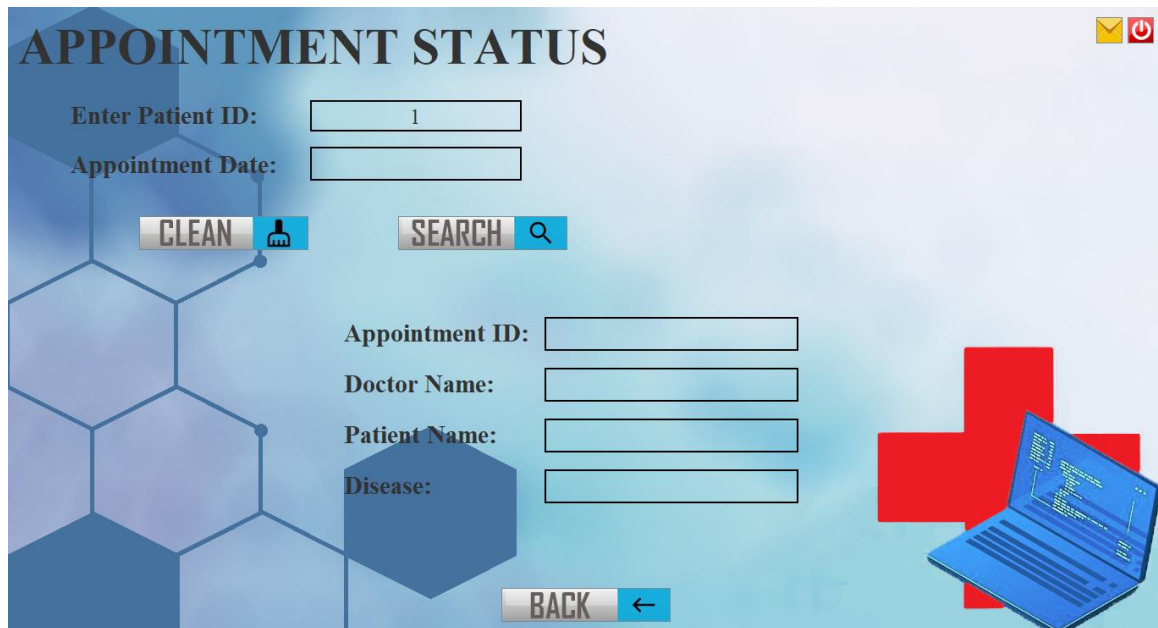
DOCTOR & DATE

Doctor ID:

Appointment Date:

BACK ← **SUBMIT** 📤



5.13 View Appointment Status:



APPOINTMENT STATUS

Enter Patient ID:

Appointment Date:


CLEAN  **SEARCH** 


Appointment ID:

Doctor Name:

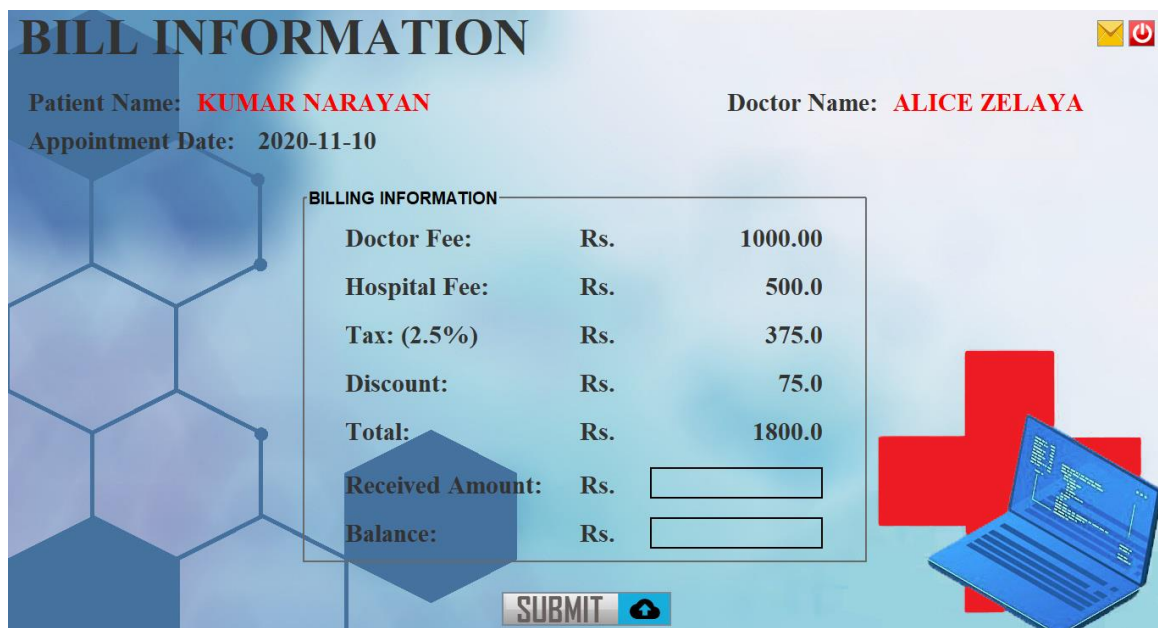
Patient Name:

Disease:

BACK 



5.14 Bill Info:





BILL INFORMATION

Patient Name: **KUMAR NARAYAN** Doctor Name: **ALICE ZELAYA**

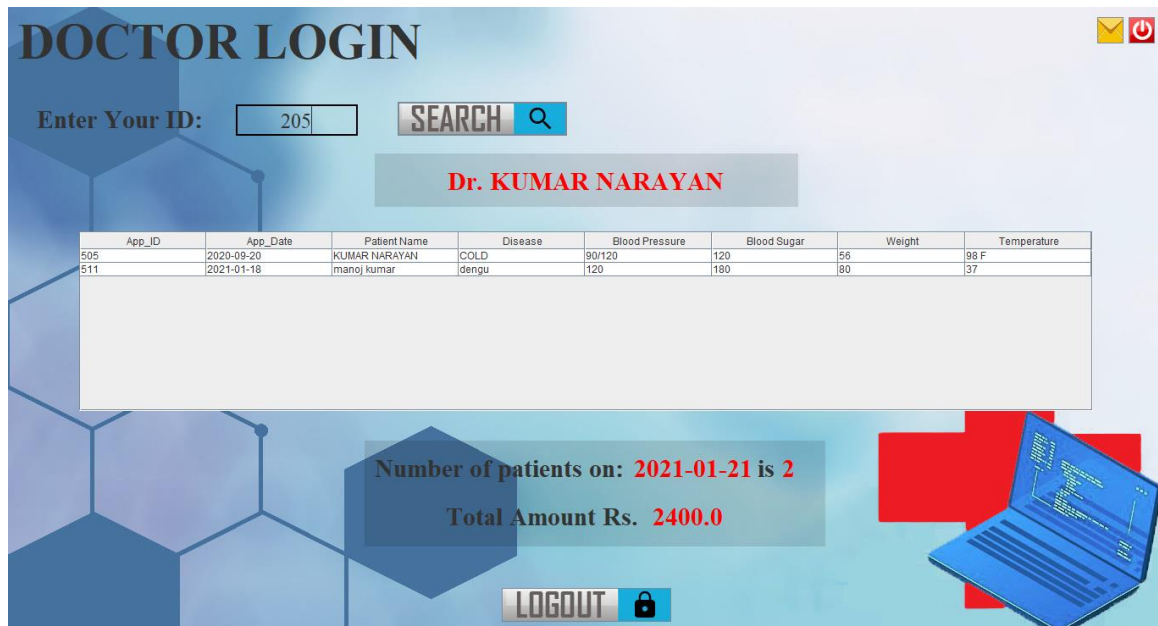
Appointment Date: 2020-11-10

BILLING INFORMATION		
Doctor Fee:	Rs.	1000.00
Hospital Fee:	Rs.	500.0
Tax: (2.5%)	Rs.	375.0
Discount:	Rs.	75.0
Total:	Rs.	1800.0
Received Amount:	Rs.	<input type="text"/>
Balance:	Rs.	<input type="text"/>


SUBMIT 



5.15 Doctor Login:




DOCTOR LOGIN

Enter Your ID: **SEARCH** 

Dr. KUMAR NARAYAN

App_ID	App_Date	Patient Name	Disease	Blood Pressure	Blood Sugar	Weight	Temperature
505	2020-09-20	KUMAR NARAYAN	COLD	90/120	120	56	98 F
511	2021-01-18	manoj kumar	dengu	120	180	80	37

Number of patients on: **2021-01-21** is **2**
Total Amount Rs. **2400.0**

LOGOUT 

5.16 About Us:



HOSPITAL MANAGEMENT SYSTEM

DEVELOPED FOR

CITY ENGINEERING COLLEGE
BANGALORE

DEVELOPED BY

AZEEZ MOHAMED FAZRY
-- 1CE18CS010 --
azeezfazry@gmail.com
<http://github.com/azeez-fazry>

Collaboration With

MAHESH R
-- 1CE18CS040 --
mahemahe04716@gmail.com

BACK 

5.17 DATABASE SNAPSHOTS:

5.17.1 DATABASE TABLES:

```
Command Prompt - mysql -u admin -p
mysql> SHOW TABLES;
+-----+
| Tables_in_hospital_management_system |
+-----+
| appointment
| billinfo
| doctor
| nurse
| patient
| staff
+-----+
6 rows in set (0.01 sec)

mysql> _
```

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

The project Hospital Management System is for computerizing the working in a hospital. It is a great improvement over the manual system. The computerization of the system has speed up the process. In the current system the front office managing is very slow. The hospital management system was thoroughly checked and tested with duplicate data and thus is found to be very reliable. The software takes care of all requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital. It generates prescription and also provides facility for searching the details of patient. The system also provides the facility of backup as per the requirement.

FUTURE ENHANCEMENT

The Proposed system in Hospital Management system is we can enhance this system by including more facilities like pharmacy system for the stock details of medicines in the pharmacy. Providing such features enables the users to include more comments into the system.

BIBILIOGRAPHY

1. "Database System Concepts" by Abraham Silberschatz and S Sudarshan.
2. "Fundamentals of database system" by R Elmasri & Navathe
3. "Database Management System" by Raghu Ramakrishnan

www.stackoverflow.com

www.tutorialspoint.com

www.material.io