<u>Title:</u>

Hospital Management System

<u>Abstract:</u>

Hospitals currently use a manual system for the management and maintenance of critical information. The current system requires numerous paper forms, with data stores spread throughout the hospital management infrastructure. Often information (on forms) is incomplete, or does not follow management standards. Forms are often lost in transit between departments requiring a comprehensive auditing process to ensure that no vital information is lost. Multiple copies of the same information exist in the hospital and may lead to inconsistencies in data in various data stores.
A significant part of the operation of any hospital involves the acquisition, management and timely retrieval of great volumes of information. This information typically involves; patient personal information and medical history, staff information, room and ward scheduling, staff scheduling, operating theater scheduling and various facilities waiting lists. All of this information must be managed in an efficient and cost wise fashion so that an institution's resources may be effectively utilized HMS will automate the management of the hospital making it more efficient and error free. It aims at standardizing data, consolidating data ensuring data integrity and reducing inconsistencies.

<u>Introduction:</u>

Purpose

• The Software is for the automation of Hospital Management.
• It maintains two levels of users :
1. Administrator Level
2. Doctor Level
3. Receptionist Level

• The Software includes:-
1. Maintaining Patient details.
2. Maintaining Patient Appointments Records
3. Maintaining Doctor Details.
4. Maintaining Doctor Appointments.

Scope

It can be used in any Hospital, Clinic, Dispensary or Pathology labs for maintaining patient details.

<u>Problem Definition</u>

To implement a system to store and manage the Details of patients, doctors, admins and receptionists of a Particular Hospital.

Learning Objectives

This project is being undertaken by the students for them to:
- Understand Database connectivity to the frontend of an application
- Understand Database concepts an actual application
- Understand how backend and frontend work in tandem to make an application function

Learning Outcomes

After completion of this project, the students will be able to:
- Connect Database to the frontend of an application
- Implement Database concepts an actual application

Theory

JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver conver
method
calls into the ODBC function calls.

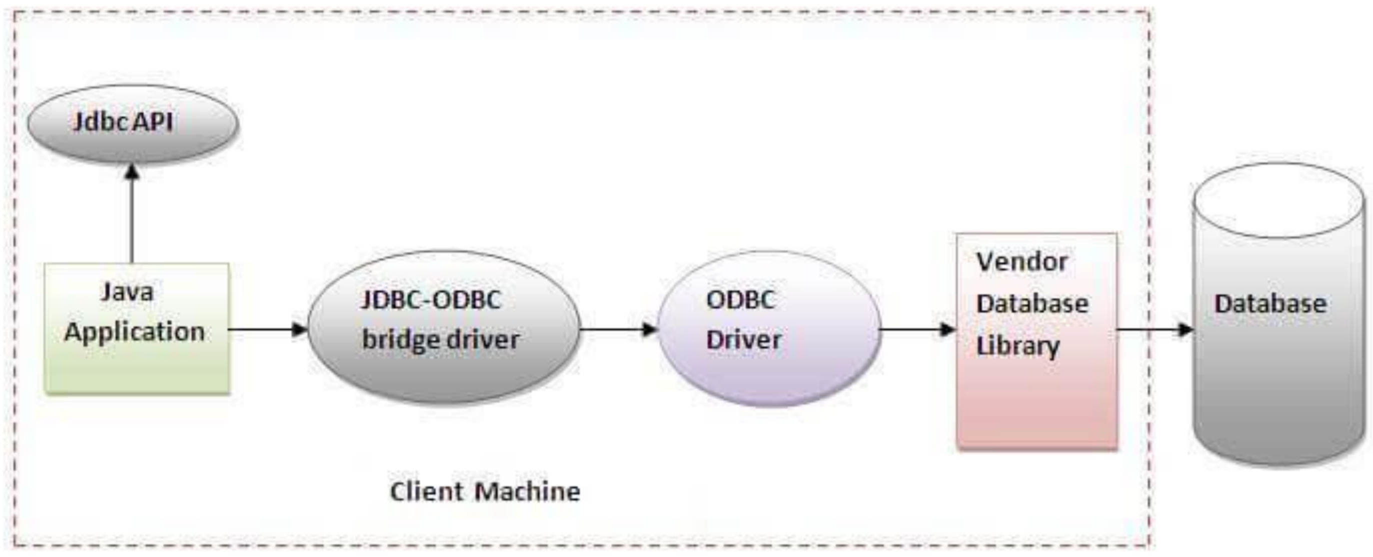


Figure- JDBC-ODBC Bridge Driver

Advantages:

- easy to use.
- can be easily connected to any database.

Disadvantages:

- Performance degraded because JDBC method call is converted into the ODBC function calls.
- The ODBC driver needs to be installed on the client machine.

The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface

- DatabaseMetaData interface
- RowSet interface

To connect Java application with the MySQL database, we need to follow following steps.

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/db_hospital** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sono is the database name. We may use any database, in such case, we need to replace the sono with our database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use **root** as the password.

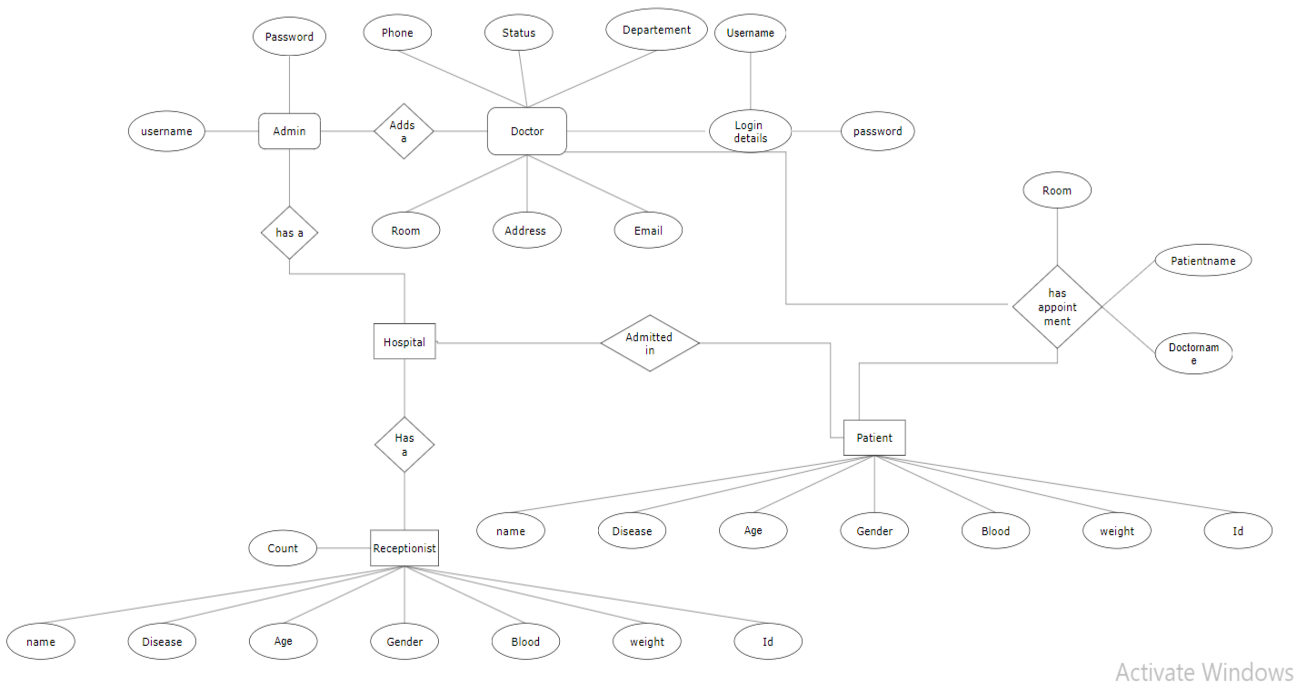
   Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.Unlike AWT, Java Swing provides platform-independent and lightweight components.The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

   The class **JFrame** is an extended version of **java.awt.Frame** that adds support for the JFC/Swing component architecture.

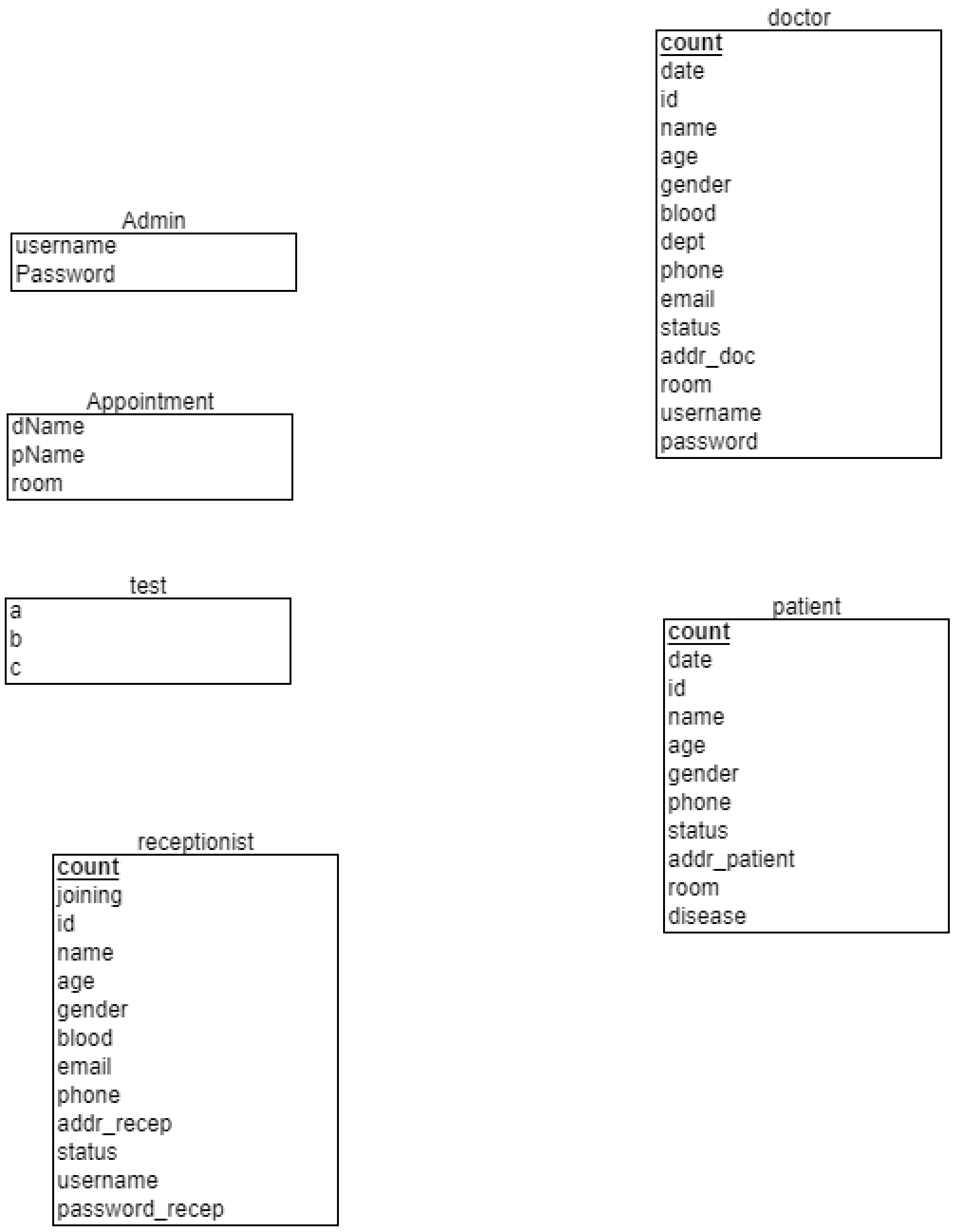
   JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

   Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

Class Diagram/ER Diagram



Relational Schema:



Output (SNAPSHOTS):

Fig.1) Main Portal

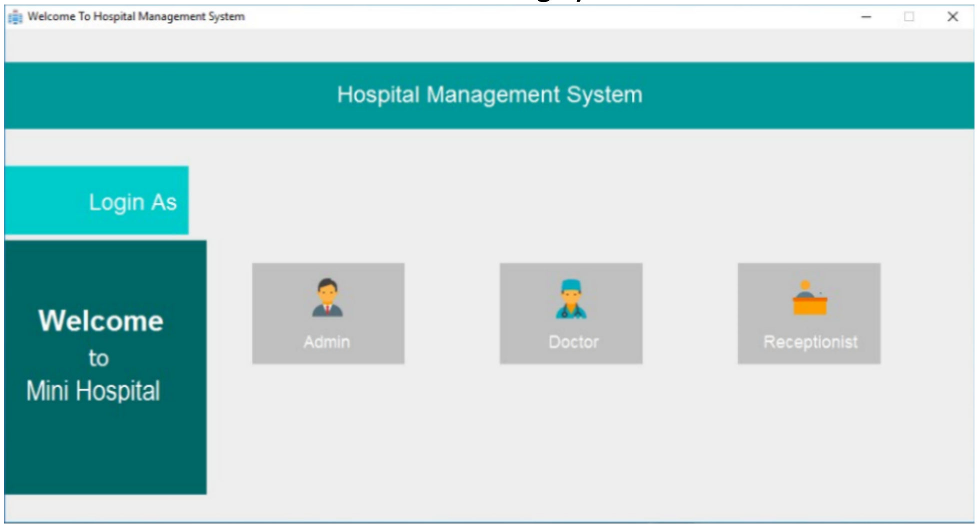
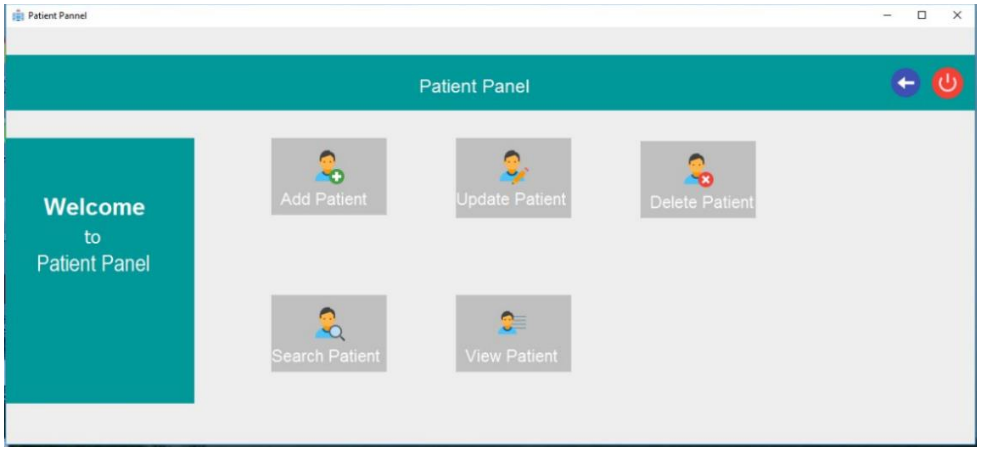
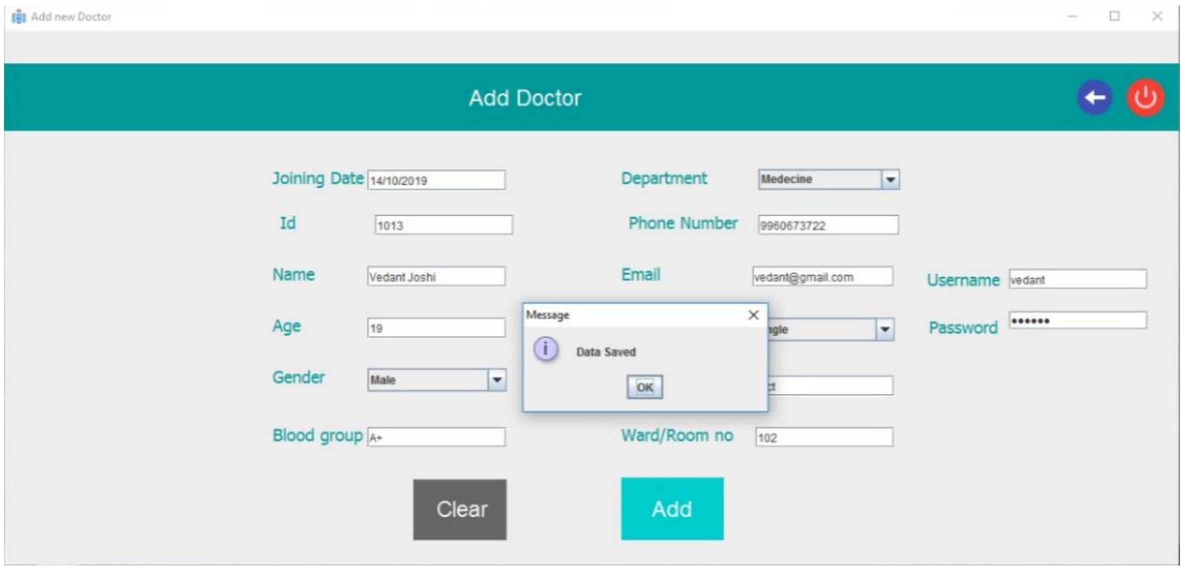

Welcome To Hospital Management System
Hospital Management System
Login As
Welcome
to
Mini Hospital
Admin
Doctor
Receptionist

Fig.2) Admin Login





Add new Doctor

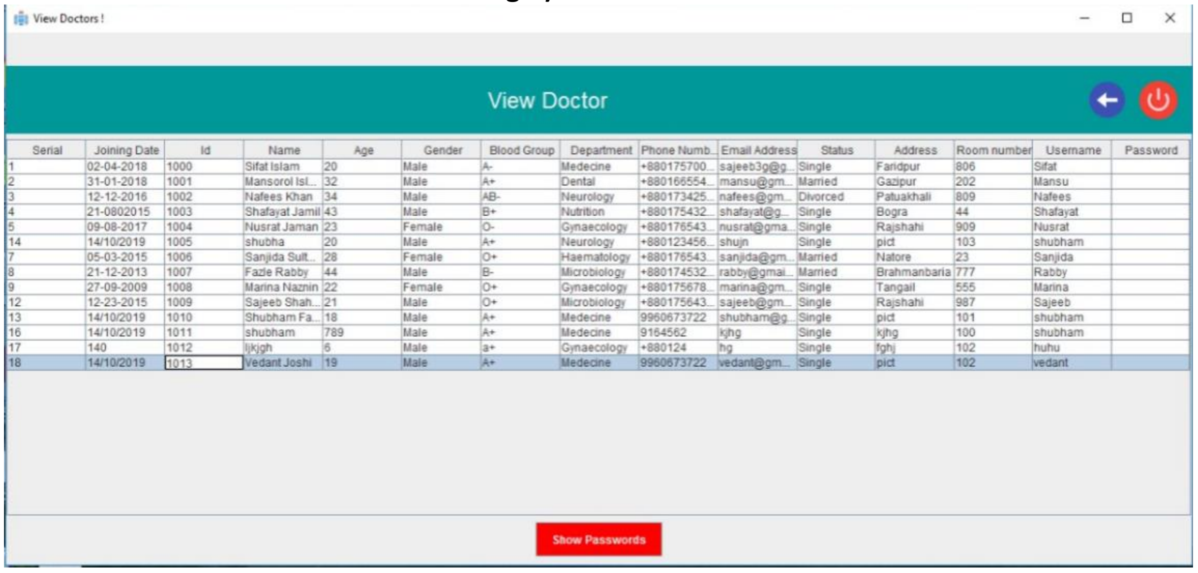
Add Doctor

Joining Date	14/10/2019	Department	Medecine		
Id	1013	Phone Number	9960673722		
Name	Vedant Joshi	Email	vedant@gmail.com	Username	vedant
Age	19		gle	Password	••••••
Gender	Male		ct		
Blood group	A+	Ward/Room no	102		

Message

Data Saved

Fig.3) Add Doctor

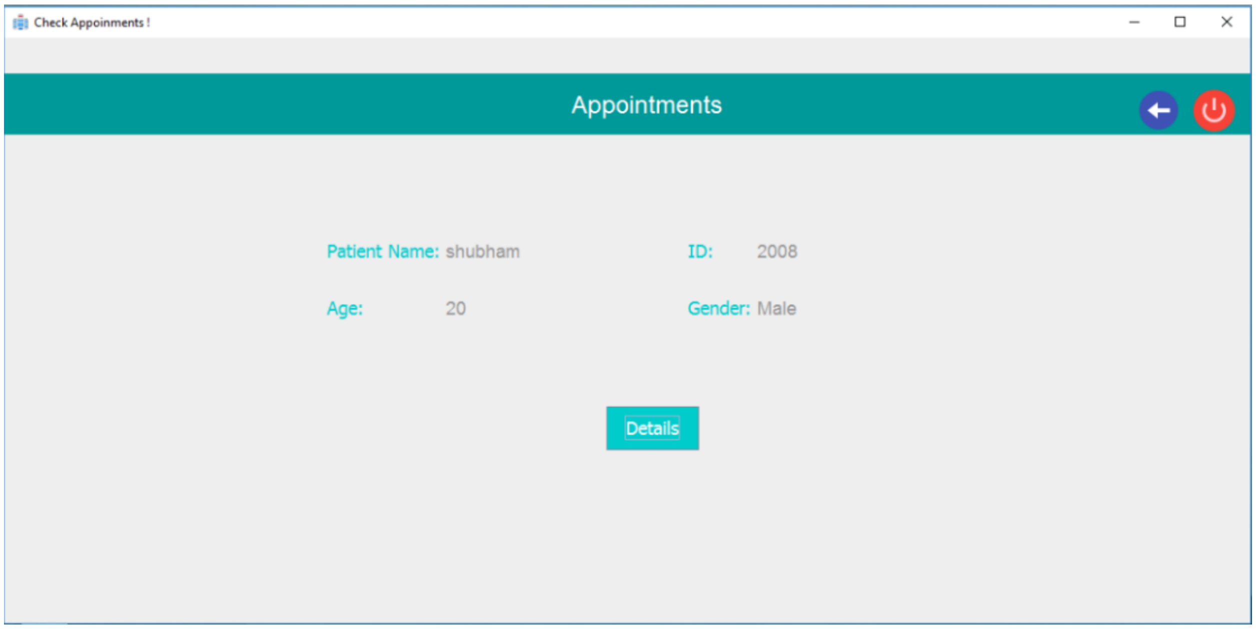


View Doctors !

View Doctor

Serial	Joining Date	Id	Name	Age	Gender	Blood Group	Department	Phone Numb...	Email Address	Status	Address	Room number	Username	Password
1	02-04-2018	1000	Sifat Islam	20	Male	A-	Medecine	+880175700...	sajeeb3g@g...	Single	Faridpur	806	Sifat	
2	31-01-2018	1001	Mansorol Isl...	32	Male	A+	Dental	+880166554...	mansu@gm...	Married	Gazipur	202	Mansu	
3	12-12-2016	1002	Nafees Khan	34	Male	AB-	Neurology	+880173425...	nafees@gm...	Divorced	Patuakhali	809	Nafees	
4	21-0802015	1003	Shafayat Jamil	43	Male	B+	Nutrition	+880175432...	shafayat@g...	Single	Bogra	44	Shafayat	
5	09-08-2017	1004	Nusrat Jaman	23	Female	O-	Gynaecology	+880176543...	nusrat@gma...	Single	Rajshahi	909	Nusrat	
14	14/10/2019	1005	shubha	20	Male	A+	Neurology	+880123456...	shujn	Single	pict	103	shubham	
7	05-03-2015	1006	Sanjida Sult...	28	Female	O+	Haematology	+880176543...	sanjida@gm...	Married	Natore	23	Sanjida	
8	21-12-2013	1007	Fazle Rabby	44	Male	B-	Microbiology	+880174532...	rabby@gmai...	Married	Brahmanbaria	777	Rabby	
9	27-09-2009	1008	Marina Naznin	22	Female	O+	Gynaecology	+880175678...	marina@gm...	Single	Tangail	555	Marina	
12	12-23-2015	1009	Sajeeb Shah...	21	Male	O+	Microbiology	+880175643...	sajeeb@gm...	Single	Rajshahi	987	Sajeeb	
13	14/10/2019	1010	Shubham Fa...	18	Male	A+	Medecine	9960673722	shubham@g...	Single	pict	101	shubham	
16	14/10/2019	1011	shubham	789	Male	A+	Medecine	9164562	kjhg	Single	kjhg	100	shubham	
17	140	1012	ljkjgh	6	Male	a+	Gynaecology	+880124	hg	Single	fghj	102	huhu	
18	14/10/2019	1013	Vedant Joshi	19	Male	A+	Medecine	9960673722	vedant@gm...	Single	pict	102	vedant	

Fig.4) View Doctors



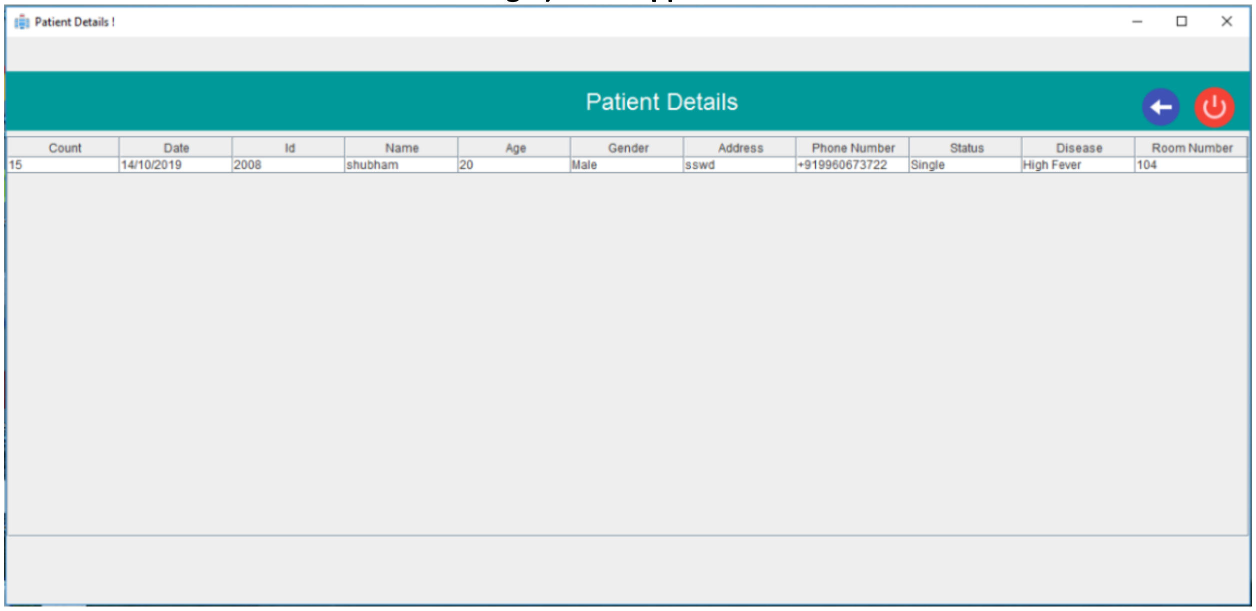
Patient Name: shubham

ID: 2008

Age: 20

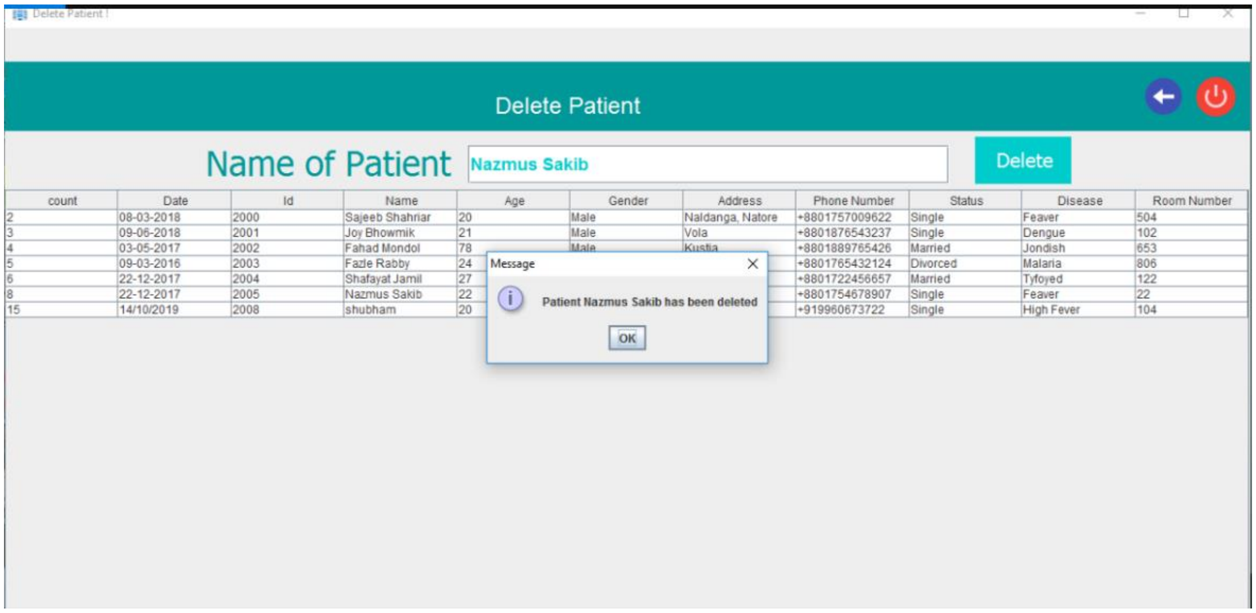
Gender: Male

Fig.5) View Appointments



Count	Date	Id	Name	Age	Gender	Address	Phone Number	Status	Disease	Room Number
15	14/10/2019	2008	shubham	20	Male	sswd	+919960673722	Single	High Fever	104

Fig.6) View Patient



count	Date	Id	Name	Age	Gender	Address	Phone Number	Status	Disease	Room Number
2	08-03-2018	2000	Sajeeb Shahriar	20	Male	Naldanga, Natore	+8801757009622	Single	Feaver	504
3	09-06-2018	2001	Joy Bhowmik	21	Male	Vola	+8801876543237	Single	Dengue	102
4	03-05-2017	2002	Fahad Mondol	78	Male	Kustia	+8801889765426	Married	Jondish	653
5	09-03-2016	2003	Fazle Rabby	24			+8801765432124	Divorced	Malaria	806
6	22-12-2017	2004	Shafayat Jamil	27			+8801722456657	Married	Tyfoyed	122
8	22-12-2017	2005	Nazmus Sakib	22			+8801754678907	Single	Feaver	22
15	14/10/2019	2008	shubham	20			+919960673722	Single	High Fever	104

Patient Nazmus Sakib has been deleted

Fig.7) Delete Patient

Source Codes:

1. JDBC Connectivity

```
public class Connector {

    public static void main(String[] args) {
        ConnectDb();
    }

    public static Connection ConnectDb() {
        Connection connection = null;
        try {

Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_hospital", "root",
"root")
            return connection;
        } catch (ClassNotFoundException | SQLException ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
        return connection;
    }
}
```

2.Login Validation

```
private void dLoginBtnActionPerformed(java.awt.event.ActionEvent evt) {
    connection = Connector.ConnectDb();
    String user = dUserField.getText();
    String pass = String.valueOf(dPassField.getPassword());

    try {
        String sql = "select username, password from doctor where username='" + user + "'";
        prp = connection.prepareStatement(sql);
```

```
            result = prp.executeQuery();
            result.first();
            userName = user;
            if (pass.equals(result.getString("password"))) {
                doctorActivity dActivity = new doctorActivity(userName);
                dActivity.setVisible(true);
                JOptionPane.showMessageDialog(null, "Login Succesful", "Welcome " + user,
JOptionPane.INFORMATION_MESSAGE);
                dispose();
                connection.close();
            } else {
                JOptionPane.showMessageDialog(null, "Login Failed", "Error", JOptionPane.WARNING_MESSAGE);
            }
        } catch (HeadlessException | SQLException e) {
            JOptionPane.showMessageDialog(null, "User or Password wrong."); }
```

3.Add Doctor

```
connection = Connector.ConnectDb();
        if (connection != null) {
            String date = rDateField.getText();
            String id = rIdField.getText();
            String name = rNameField.getText();
            int age = Integer.parseInt(rAgeField.getText());
            String gender = (String) rGenderField.getSelectedItem();
            String rblood = rBloodField.getText();
            String email = rEmailField.getText();
            String phone = rPhoneField.getText();
            String address = rAddressField.getText();
            String status = (String) rStatusField.getSelectedItem();
            String user = rUserField.getText();
            String pass = String.valueOf(rPassField.getPassword());
            String sql = "insert into
receptionist(joining,id,name,age,gender,blood,email,phone,address,status,username,password) values
(?,?,?,?,?,?,?,?,?,?,?,?)";
            try {
                prp = connection.prepareStatement(sql);
                prp.setString(1, date);
                prp.setString(2, id);
                prp.setString(3, name);
                prp.setInt(4, age);
                prp.setString(5, gender);
                prp.setString(6, rblood);
                prp.setString(7, email);
                prp.setString(8, phone);
                prp.setString(9, address);
                prp.setString(10, status);
                prp.setString(11, user);
                prp.setString(12, pass);
                prp.execute();
                JOptionPane.showMessageDialog(null, "Data Saved");
                  ReceptionistManagement rm = new ReceptionistManagement();
        rm.setVisible(true);
        dispose();
            } catch (SQLException e) {
            }
```
4.Delete Receptionist
```
connection = Connector.ConnectDb();
        String search = rDeleteField.getText();
        String sql = "Delete from receptionist where name ='" + search + "'";
        try {
            ps = connection.prepareStatement(sql);
            ps.execute();
            JOptionPane.showMessageDialog(null, "Patient " + search + " has been deleted");
            defaultTableModel.getDataVector().removeAllElements();
            defaultTableModel.fireTableDataChanged();
            loadData();
            connection.close();
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "patient named " + search + " not found");
        }
```
5.Update Patient
```
    connection = Connector.ConnectDb();
        String sql = "Update patient set date = '" + date + "', id = '" + id + "', name = '" + name + "', age = '" + age + "',
gender = '" + gender + "', address = '" + address + "', phone = '" + phone + "', status = '" + status + "', disease = '" +
disease + "', room = '" + room + "' where count = '" + count + "'";
        try {
            ps = connection.prepareStatement(sql);
            ps.execute();
            defaultTableModel.getDataVector().removeAllElements();
            defaultTableModel.fireTableDataChanged();
            loadData();
            JOptionPane.showMessageDialog(null, "Data Updated");
        } catch (HeadlessException | SQLException e) {
            JOptionPane.showMessageDialog(null, e);
        }
```
6.Set Appointment
```
     int row = pSTable.getSelectedRow();
        String pname = (pSTable.getValueAt(row, 2).toString());
        connection = Connector.ConnectDb();
        String sql = "insert into appointment(dName,pName,room) values (?,?,?)";
        try {
            ps = connection.prepareStatement(sql);
            ps.setString(1, dName);
            ps.setString(2, pname);
            ps.setInt(3, droom);
            ps.execute();
            JOptionPane.showMessageDialog(null, "PATIENT " + pname + " Appointment set with DOCTOR " + dName +
" at room no " + droom);
        } catch (HeadlessException | SQLException e) {
        }
```

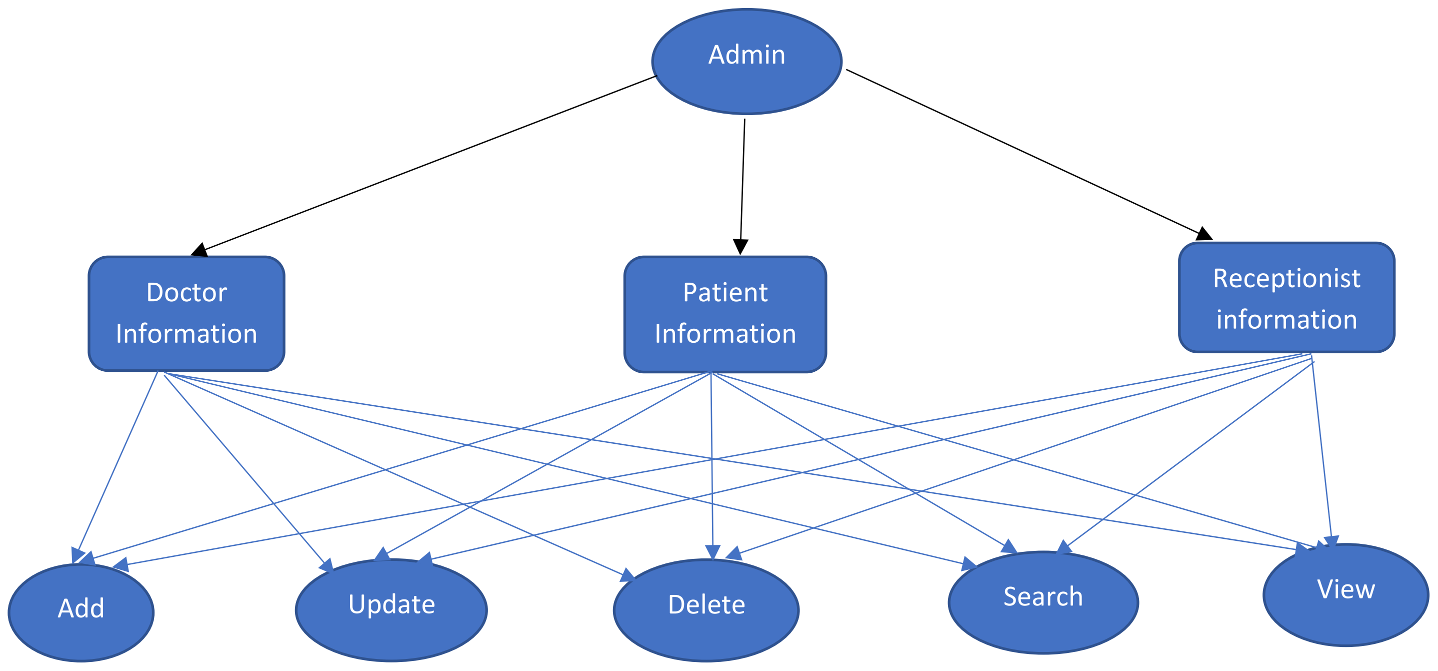
Test cases

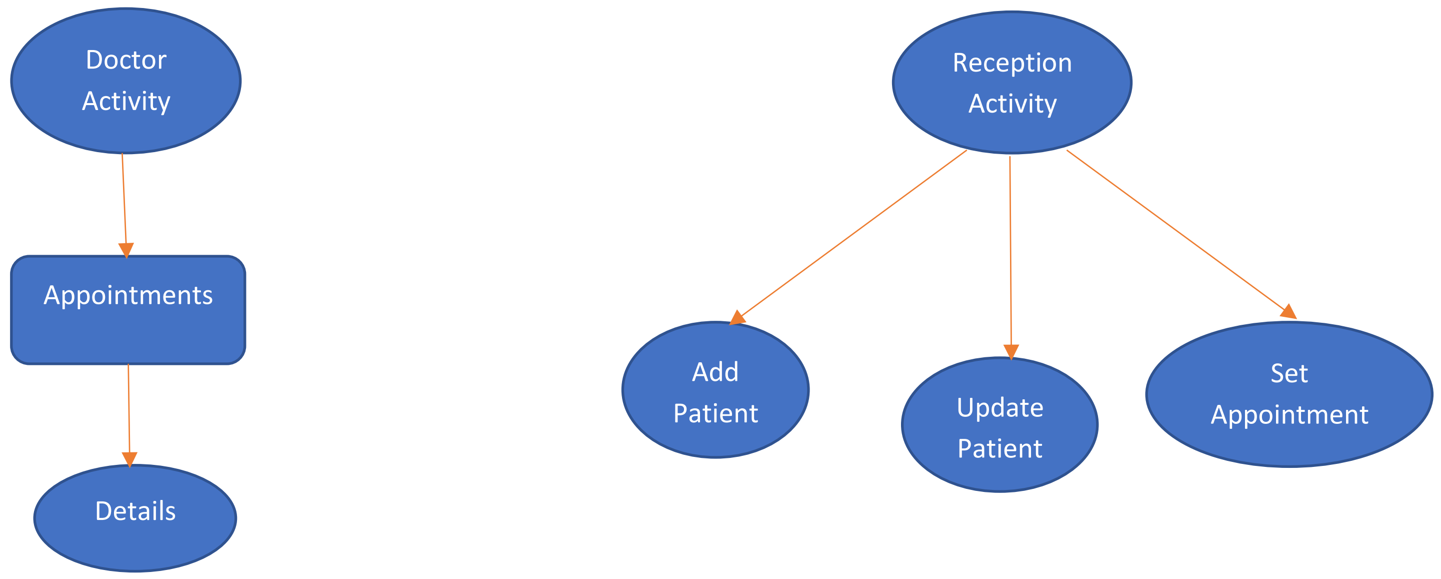
INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
Add Patient Name : Shubham Disease : Cancer Age: 20 Date: 14/10/2019 Gender: Male Ward No: 102 Phone:9977442255	Patient Added	Patient Added	**Pass**
Login Patient Name : Shubham Password : 12345	Login Unsuccessful	Login Unsuccessful	**Pass**
Login Admin Name : admin Password : admin	Login Successful	Login Successful	**Pass**
Set Appointment Department: Nutrition Patient Name: Shubham	Patient Appointment Set with doctor	Patient Appointment Set with doctor	**Pass**

Features:

1. **Improved Processes**: It helps to optimize the user experience. Medical specialists, patients, and hospital authorities can interact online, make the appointments and exchange information.
2. **Digital medical records:** The hospital database includes all the necessary patient data. The disease history, test results, prescribed treatment can be accessed by doctors without much delay in order to make an accurate diagnosis and monitor the patient's health. It enables lower risks of mistakes**.**
3. **Facility management**: Hospitals authorities are able to manage their available resources, analyze staff work, reduce the equipment downtime, optimize the supply chain, etc. Another fact to mention is that hospital staff deal with the digital data instead of endless paperwork.
4. **Better customer experience:** Since the clinic management system is patient-oriented, the treatment process can be less stressful. Doctors have more time for the examination and interaction with patients. In addition, all the requested information can be received online.

Data Modelling:





Conclusion:

The project Hospital Management System is for computerizing the working in a hospital. It is a great improvement over the manual system. This system is thoroughly checked and tested with dummy data and thus is found very reliable. Thus, we have successfully implemented Hospital Management System using SQL queries and Java Swing for the Frontend.