Criteria C- Documentation

Table of Contents

By developing the application in Java NetBeans IDE and utilizing PhpMyAdmin (MAMP), all of the success criteria are met. All of the comments describing the code written are heavily annotated in **Appendix-5.**

Technique 1- Global Variable

Depending on the kind of user, the global variable "glv" is allocated a value (employee, admin). When utilizing the "menu" button, this makes it simpler to identify forms because two forms might link to the same page for different people, and "glv" can aid to track that back to the proper form.

Class login

```
public static int glv=0; //declaring global variable as 0
```

"glv" is declared as 0 on the login page

Class EmployeeDashboard

```
public EmployeeDashboard() {
    initComponents();
    login.glv=2;// glv=2 is for employees
```

Class AdminDashboard

```
public AdminDashboard() {
    initComponents();
    login.glv=1;// glv=1 is for admins
```

"glv" is assigned according to the user type

Class NewPatient

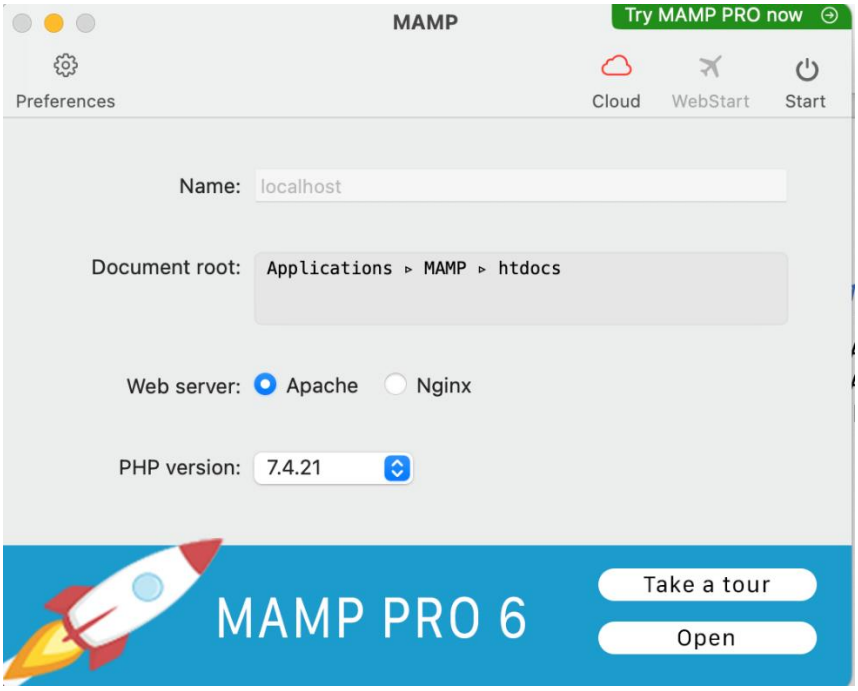
```
private void bMenuActionPerformed(java.awt.event.ActionEvent evt) {
    if (login.glv==1) //The if statement checks the "glv" value
            {
                this.dispose();
                new AdminDashboard().setVisible(true);
            }
            else
            {
                this.dispose();
                new EmployeeDashboard().setVisible(true);
            }
```

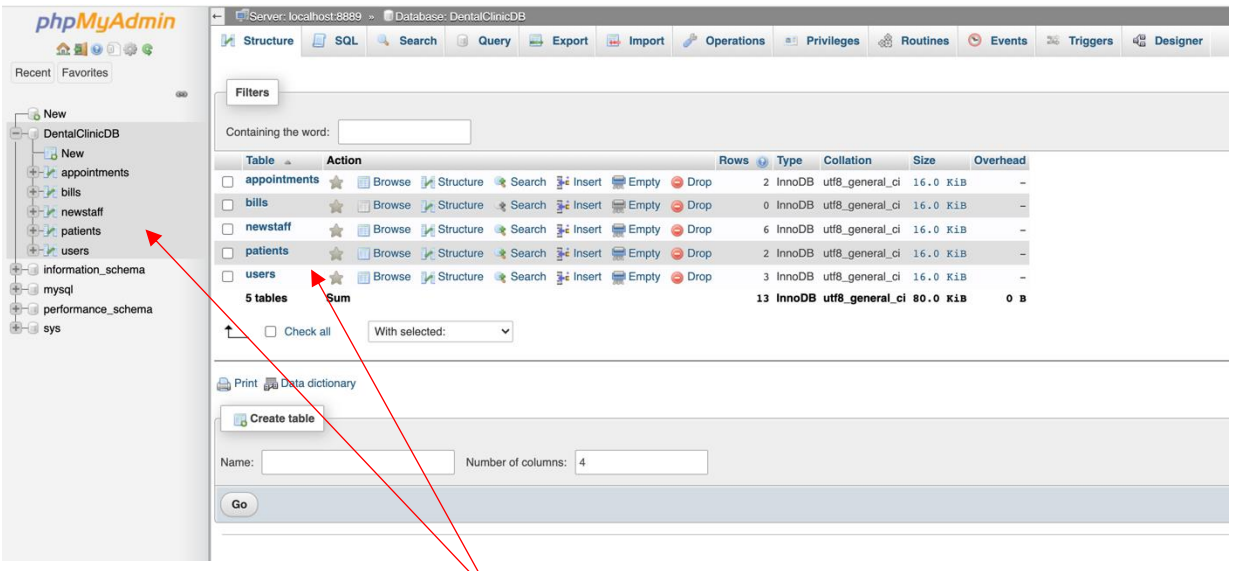
The if statement checks the "glv" value and navigates to the employee or admin dashboard depending on the value of "glv"

Link to Success Criteria 1

Technique 2- External Software

MAMP is a piece of third-party software that links the software to an online database using PhpMyAdmin. PhpMyAdmin is a piece of online software that saves a local online database and allows us to add, remove, and update data to and from it.

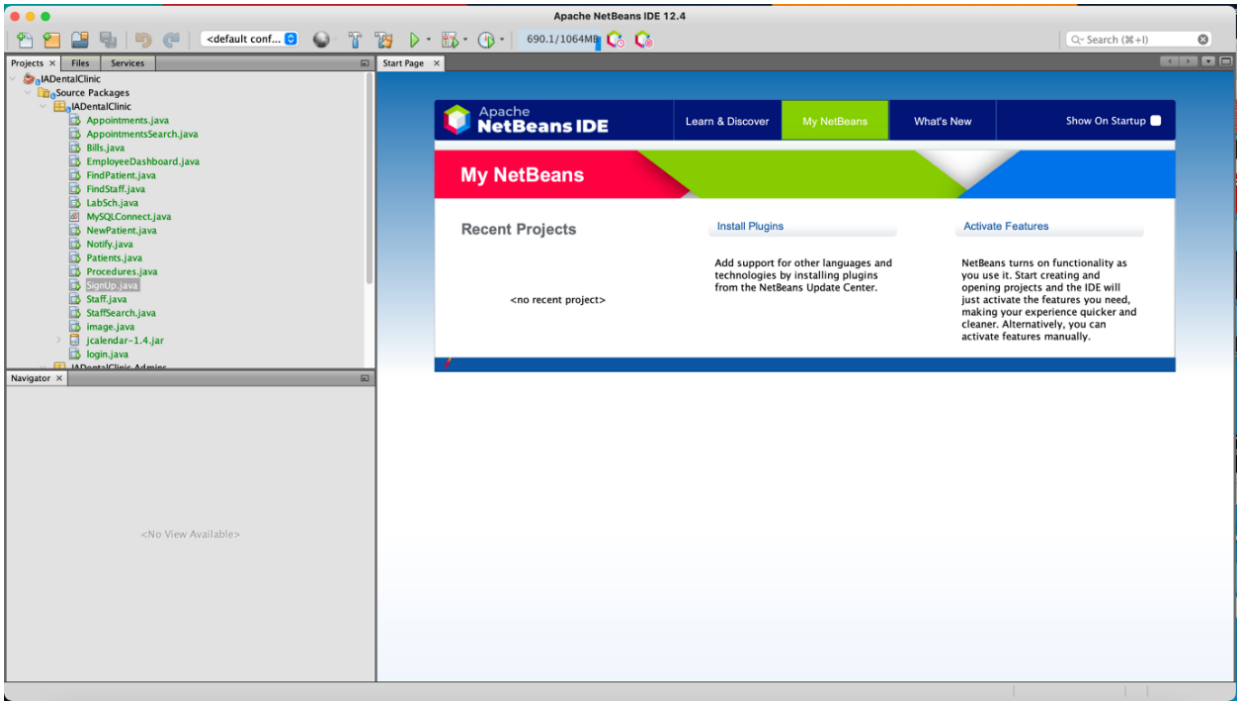




All the tables stored in the database of PhpMyAdmin

MAMP and PhpMyAdmin are connected which helps create a local database that can be used in the software.

Apache NetBeans is a software developer for java which allows integrated development. It also enables the development of applications from a set of modular software components known as modules.



Different classes used for the development of the whole system

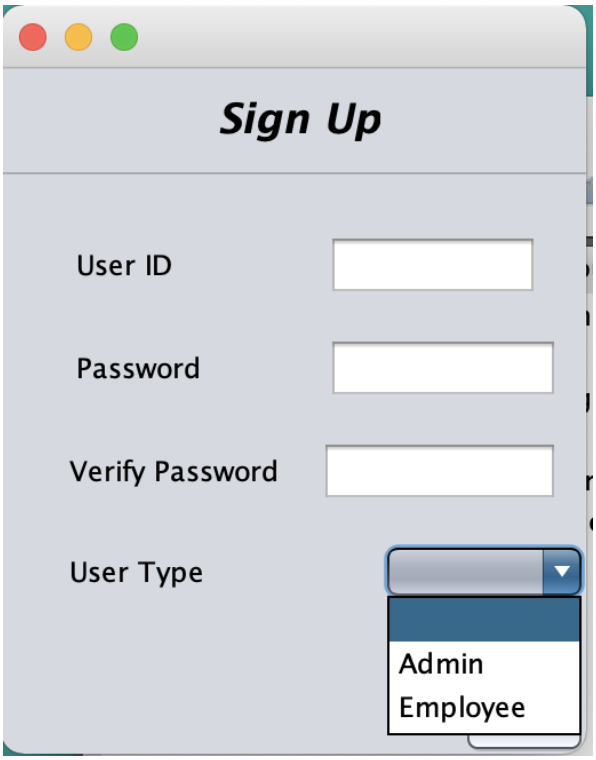
Technique 3- Java Database connectivity (JDBC)

I utilized JDBC to link NetBeans and MAMP, which also aids in the creation of an online database using PhpMyAdmin. This strategy will be present in practically every source code of this product to assist in meeting all of the success criteria specified.

```
package IADentalClinic;
import java.sql.*;
import javax.swing.*;
public class MySQLConnect {
    Connection conn=null;
    public static Connection ConnectDB(){ //allows the local software to connect to the database server
        try{
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:8889/DentalClinicDB","root","root");
            return conn;
        }
    catch(ClassNotFoundException | SQLException e){
    JOptionPane.showMessageDialog(null,e);
    return null;
    }
    }
}
```

Technique 4- Selection of Control System

This technique is used to decide between 2 or more options. When a staff member or the client is signing it used to distinguish between the admin and the employee.



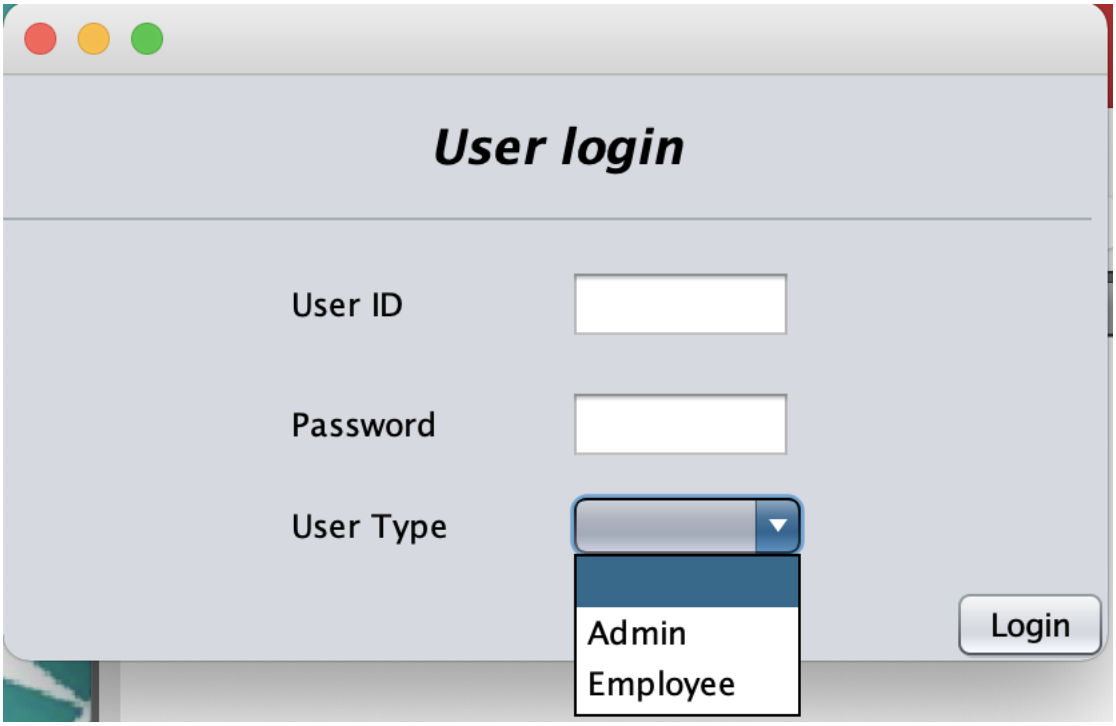
The user can select their user type to be as the admin or the employee. The user can also add more user types if needed later.

Link to Success Criteria 1

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="insert into users values (?,?,?)";
    if(this.tfPassword.getText().equals(this.tfVerifyPassword.getText())) {// verifies if password entered is matching
    try{

    pst = conn.prepareStatement(Sql);
    pst.setString(1,this.tfUserID.getText());
    pst.setString(2,this.tfPassword.getText());
    pst.setString(3, (String) this.cboUserType.getSelectedItem());
    int i = pst.executeUpdate(); //the values will get verified and then get stored in the database.
    if(i>0){
    JOptionPane.showMessageDialog(null, "User Successfully Added"); //A message will be displayed
        new login().setVisible(true);
        this.dispose();
```

Once the user saves their login details, the values will get verified and then get stored in the database.

A message will be displayed indicating that the user data has been successfully stored.



Via the login page, the user can then select if they wish to login as the admin or the employee.

```
private void bLoginActionPerformed(java.awt.event.ActionEvent evt) {
     connd=MySQLConnect.ConnectDB();
  int flag=0;
    String Sql="select * from users where username=? and password=? and usertype=?";
    try{// checks the values, stores it in a variable temporarily

        pstd=connd.prepareStatement(Sql);
        pstd.setString(1,tfUserID.getText());
        pstd.setString(2,tfPass.getText());
        pstd.setString(3, (String) this.cboUserType.getSelectedItem());
        rsd = pstd.executeQuery();
        while(rsd.next()){
            if(this.cboUserType.getSelectedItem().equals("Admin")) {
                flag=1;
                JOptionPane.showMessageDialog(null,"Successfully logged in as Admin!");
                this.dispose();
                new AdminDashboard().setVisible(true);
```

This validates the values, temporarily puts them in a variable, then compares them to the database.

```
        if(this.cboUserType.getSelectedItem().equals("Admin")) {
            flag=1;// Checking the user type using an if-else statement.
            JOptionPane.showMessageDialog(null,"Successfully logged in as Admin!");
            this.dispose();
            new AdminDashboard().setVisible(true);

        }
        else{
            flag=2;// the input given is compared with the database
            JOptionPane.showMessageDialog(null,"Successfully logged in as Employee!");
            this.dispose();
            new EmployeeDashboard().setVisible(true);
        }

        }
    if(flag==0)
    {
        JOptionPane.showMessageDialog(null,"User Name or Password Incorrect!");
        this.dispose();// if there is an error while logging in, a message will be displayed
        new login().setVisible(true);
    }

}catch(HeadlessException | SQLException e){
    JOptionPane.showMessageDialog(null, e);
}
```

Using an if-else expression to check the user type. The user's input is compared to the servers. If the input does not match, an error message is displayed.

Technique 5- Exception handling

If there is an issue or problem in the database while storing, removing, or updating data, or if the user does not fill in a needed field in a form, an error message will be displayed. For such a case, this code will be found in every class of software.

```
private void bDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="delete from patients where id=?";
    try{
        pst=conn.prepareStatement(Sql);
        pst.setInt(1,Integer.parseInt(this.tfID.getText()));
        int i = pst.executeUpdate();
        if(i>0){
            JOptionPane.showMessageDialog(null, "Patient Successfully Deleted");
            this.dispose();
            new FindPatient().setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(null,"Patient Data not Deleted!");
        }
        //The 'try-catch' phrases examine errors while connecting to database.
    }catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
```

The 'try-catch' phrases investigate issues encountered while connecting to a database.

Technique 6- Complex Query

This technique allows you to delete, update or select any data on different tables stores with some specific commands.

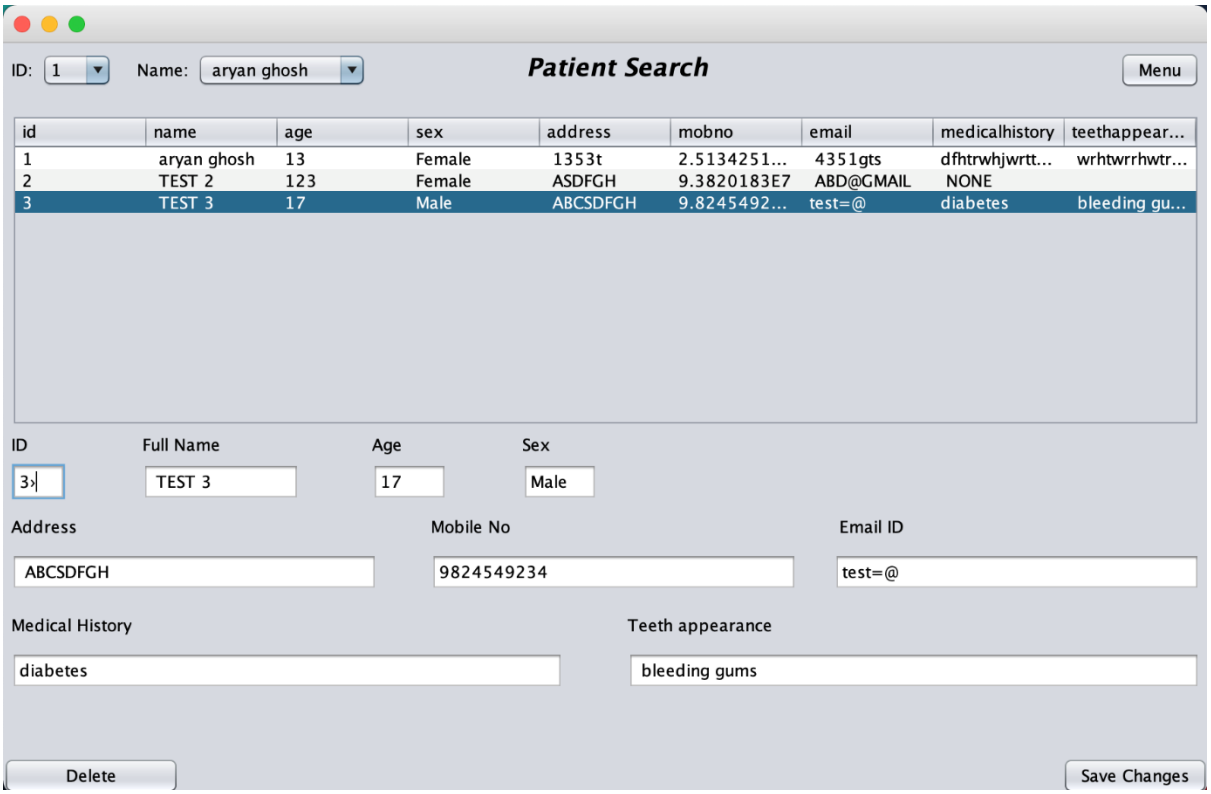
```
public void displayItem()
{
    String sql1 ="select * from patients";
    try {//Selects data from 'patients' database.
        pst1 = conn.prepareStatement(sql1);
         rs=pst1.executeQuery();
         tPatient.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (SQLException ex) {
        Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
public void displayAllItem(String id)
{
    try{
        String sql2 ="select * from patients where id=?";//Selects data with respect to id
            pst1=conn.prepareStatement(sql2);
            pst1.setString(1, id);
            rs1=pst1.executeQuery();
            tPatient.setModel(DbUtils.resultSetToTableModel(rs1));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVER
        }
       }
public void displayAllItemName(String name)
{
    try{
        String sql3 ="select * from patients where name=?";//Selects data with respect to name
            pst2=conn.prepareStatement(sql3);
            pst2.setString(1, name);
            rs2=pst2.executeQuery();
            tPatient.setModel(DbUtils.resultSetToTableModel(rs2));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
```

Selects data from 'patients' database.

Selects data from the 'patients' database based on id and name and compares it to the user input.

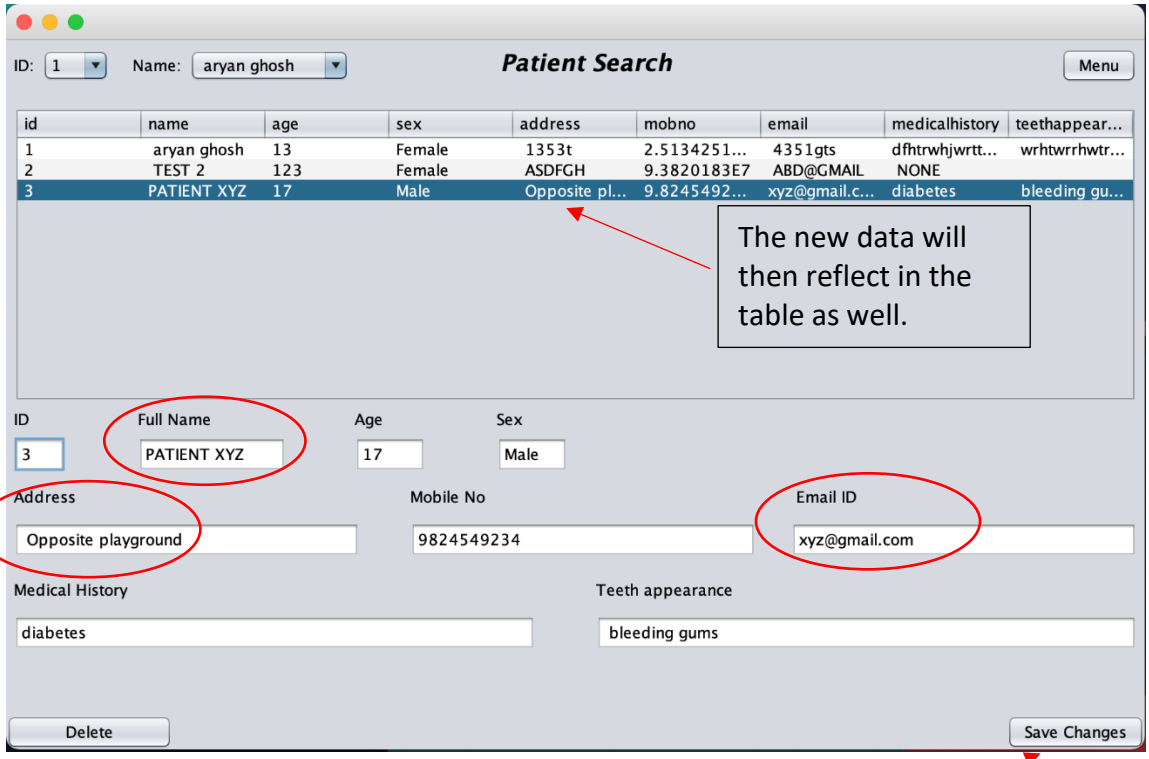
This particular form helps the user to update, delete or view any patient data that is already stores in the database. With the commands shown above the user can select and view the patient data.

Link to Success Criteria 2



By clicking on the list in the table displayed on this form, the selected patient data will be displayed in the text fields given below for the user to view them fully.

If the user wishes to make any changes to the data, they must make the changes in the given text fields and then click ‘save changes’.
To delete data, they must select the patient and then click ‘delete’



```
private void tPatientMouseClicked(java.awt.event.MouseEvent evt) {
    DefaultTableModel model = (DefaultTableModel)tPatient.getModel();
    int i = tPatient.getSelectedRow();//retrieving data
    this.tfID.setText(model.getValueAt(i, 0).toString());
    this.tfName.setText(model.getValueAt(i, 1).toString());
    this.tfAge.setText(model.getValueAt(i, 2).toString());
    this.tfSex.setText(model.getValueAt(i, 3).toString());
    this.tfAddress.setText(model.getValueAt(i, 4).toString());
    this.tfMob.setText(model.getValueAt(i, 5).toString());
    this.tfEmail.setText(model.getValueAt(i, 6).toString());
    this.tfMed.setText(model.getValueAt(i, 7).toString());
    this.tfTeeth.setText(model.getValueAt(i, 8).toString());// all data will be displayed in text boxes
```

Obtaining all the text fields to display them in the field boxes.

Retrieving data from the table stored in the database.

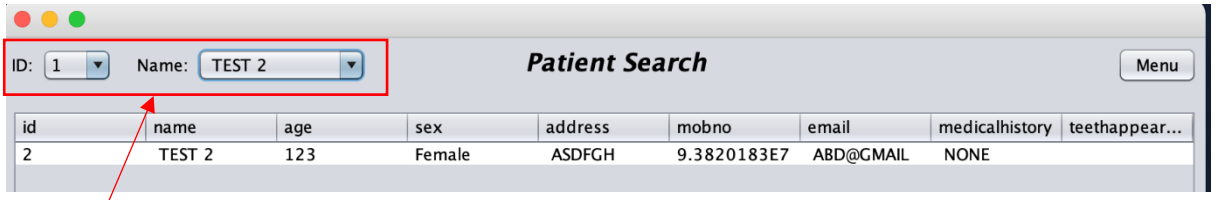
The command below, allows you to search for a specific data set based on either the id or their name stored in the database via a combo box

```
private void cboIDItemStateChanged(java.awt.event.ItemEvent evt) {
    displayAllItem((String) this.cboID.getSelectedItem());
}

private void cboNameItemStateChanged(java.awt.event.ItemEvent evt) {
    displayAllItemName((String) this.cboName.getSelectedItem());
}
```

Retrieves the data stored in the database.

This same technique can be used with a text field

```
private void bNameSearchActionPerformed(java.awt.event.ActionEvent evt) {
    String sql ="select * from patients where name=?";
try{// retrieval of data from the database
    pst = conn.prepareStatement(sql);// connectivity to database
    pst.setString(1,this.tfNameSearch.getText());// reading of data
    rs=pst.executeQuery();
  this.tPatient.setModel(DbUtils.resultSetToTableModel(rs));


    } catch (SQLException ex) {
        Logger.getLogger(FindStaff.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



By selecting the id number or name, you can search for a data set.

Updating data:

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="update patients set name=?, age=?, sex=?, address=?, mobno=?, email=?, medicalhistory=?, teethappearance=? where id=?";
    try{// the data is stored in temporary variables and then updates the database

        pst=conn.prepareStatement(Sql);

        pst.setString(1,this.tfName.getText());
        pst.setString(2,this.tfAge.getText());
        pst.setString(3,this.tfSex.getText());
        pst.setString(4,this.tfAddress.getText());
        pst.setString(5,this.tfMob.getText());
        pst.setString(6,this.tfEmail.getText());
        pst.setString(7,this.tfMed.getText());
        pst.setString(8,this.tfTeeth.getText());
        pst.setString(9,this.tfID.getText());
        // every field in the particular database will be updated with new data
        int i = pst.executeUpdate();
        if(i>0){
        //if all conditions are matched the new data will be stored
            JOptionPane.showMessageDialog(null, "Patient Data Updated!");
            this.dispose();
            new FindPatient().setVisible(true);
        }
        else{// an error message will be displayed if there is an issue wiht the new data
            JOptionPane.showMessageDialog(null,"Error! Data not Saved");
        }
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
```

Updates data in ‘patients' database

Updates data in every text field.

Deleting data:

```
private void bDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="delete from patients where id=?";
    try{// the data is obtained and then deleted from the database
        pst=conn.prepareStatement(Sql);
        pst.setInt(1,Integer.parseInt(this.tfID.getText()));
        int i = pst.executeUpdate();
        if(i>0){
            JOptionPane.showMessageDialog(null, "Patient Successfully Deleted");
            this.dispose();
            new FindPatient().setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(null,"Patient Data not Deleted!");
        }
        //The 'try-catch' phrases examine errors while connecting to database.
    }catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
```

Data is retrieved from text fields and deleted from 'patients' with respect to ID.

Link to Success Criteria 3, 4, 12

it adds the data inputted by the user in the form and stores it in the specific database to be used later in the future.

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="insert into users values (?,?,?)";
    if(this.tfPassword.getText().equals(this.tfVerifyPassword.getText()))
    {

        try{
        pst = conn.prepareStatement(Sql);
        pst.setString(1,this.tfUserID.getText());
        pst.setString(2,this.tfPassword.getText());
        pst.setString(3, (String) this.cboUserType.getSelectedItem());

        int i = pst.executeUpdate();
        if(i>0){

            JOptionPane.showMessageDialog(null, "User Successfully Added");
            new login().setVisible(true);
            this.dispose();
        }
        else{
             JOptionPane.showMessageDialog(null,"Error-Re-enter ");
        }
     }
     catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
    }
    else
    {
        JOptionPane.showMessageDialog(null,"Password do not match ");
    }
}
```

Data is extracted from text fields and saved in the 'users'

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="insert into patients values (?,?,?,?,?,?,?,?,?)";
     try{
        pst = conn.prepareStatement(Sql);
        pst.setString(1,this.tfID.getText());
        pst.setString(2,this.tfName.getText());
        pst.setString(3,this.tfAge.getText());
        pst.setString(4,Sex);
        pst.setString(5,this.tfAddress.getText());
        pst.setString(6,this.tfMob.getText());
        pst.setString(7,this.tfemail.getText());
        pst.setString(8,this.tfMedHis.getText());
        pst.setString(9,this.tfTandG.getText());

        int i = pst.executeUpdate();
        if(i>0){

            JOptionPane.showMessageDialog(null, "Patient Successfully Added");
            new EmployeeDashboard().setVisible(true);
            this.dispose();
        }
        else{
             JOptionPane.showMessageDialog(null,"Data is not saved");
        }
     }
     catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Data is extracted from text fields and saved in a database called 'patients.'

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
   String Sql="insert into newstaff values (?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
     try{
        pst = conn.prepareStatement(Sql);
        pst.setString(1,this.tfID.getText());
        pst.setString(2,this.tfFullName.getText());
        pst.setString(3,this.tfAge.getText());
        pst.setString(4,Sex);
        pst.setString(5,this.tfAddress.getText());
        pst.setString(6,this.tfMobNo.getText());
        pst.setString(7,this.tfEmail.getText());
        pst.setString(8, (String) this.CboPos.getSelectedItem());
        pst.setString(9,this.tfSalary.getText());
        pst.setString(10,this.tfEname.getText());
        pst.setString(11,this.tfEaddress.getText());
        pst.setString(12,this.tfEmob.getText());
        pst.setString(13,this.tfEemail.getText());
        pst.setString(14,this.tfRelation.getText());


        int i = pst.executeUpdate();
        if(i>0)
        {

            JOptionPane.showMessageDialog(null, "Staff Successfully Added!");
            if (login.glv==1)
            {
                this.dispose();
                new AdminDashboard().setVisible(true);
            }
            else
            {
                this.dispose();
                new EmployeeDashboard().setVisible(true);
            }

        }
        else{
             JOptionPane.showMessageDialog(null,"Error, Adding staff");
        }
     }
     catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Data is extracted from text fields and saved in the 'new staff' folder.

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {

    String Sql="insert into appointments values (?,?,?,?)";
     try{
        pst = conn.prepareStatement(Sql);
        pst.setString(1,this.tfID.getText());
        pst.setString(2,this.tfName.getText());
        pst.setString(3,((JTextField)this.dcDate.getDateEditor().getUiComponent()).getText());
        pst.setString(4,this.tfTime.getText());


        int i = pst.executeUpdate();
        if(i>0)
        {

            JOptionPane.showMessageDialog(null, "Appointment Scheduled");
            if (login.glv==1)
            {
                this.dispose();
                new AdminDashboard().setVisible(true);
            }
            else
            {
                this.dispose();
                new EmployeeDashboard().setVisible(true);
            }
        }
        else
        {
             JOptionPane.showMessageDialog(null,"Error in Scheduling");
        }
     }
     catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }

}
```

Data is extracted from text fields and saved in the 'appointments' folder.

```
private void bSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String Sql="insert into bills values (?,?,?,?,?,?)";
             try{
                pst = conn.prepareStatement(Sql);
                pst.setString(1,this.tfCost.getText());
                pst.setString(2, (String) this.cboProcedure.getSelectedItem());
                pst.setString(3,this.tfAdd.getText());
                pst.setString(4,this.tfTotal.getText());
                pst.setString(5,this.tfCash.getText());
                pst.setString(6,this.tfCard.getText());

                int i = pst.executeUpdate();
                if(i>0){

                    JOptionPane.showMessageDialog(null, "Patient Successfully Added");
                    new EmployeeDashboard().setVisible(true);
                    this.dispose();
                }
                else{
                     JOptionPane.showMessageDialog(null,"Data is not saved");
                }
             }
             catch(HeadlessException | SQLException e){
                JOptionPane.showMessageDialog(null, e);
             }
}
```

Data is extracted from text fields and saved in the 'bills' folder.

Link to Success Criteria 2, 3, 4, 11, 12

Technique 8- Constructor

The constructor runs a particular code that once you run it, the information in the tables will automatically get displayed once the form opens.

```
public class Appointments extends javax.swing.JFrame {
    Connection conn=null;
    PreparedStatement pst=null;
    int rst;// connecting to the database

    public Appointments() {
        initComponents();
        conn=MySQLConnect.ConnectDB();
        getID();//performing get ID when form is opened
    }
    public void getID()
{//get ID performed
    int tempid=0;
    try{
            String sql2 ="select * from appointments";
            pst=conn.prepareStatement(sql2);
            ResultSet rs=pst.executeQuery();
            while(rs.next())
            {
                tempid = rs.getInt(1);
            }
            this.tfID.setText(Integer.toString(tempid+1));
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
}
```

Connects the database

Performs 'getID' when the form is opened.

```
public class NewPatient extends javax.swing.JFrame {
    String Sex=" ";
    Connection conn=null;
    PreparedStatement pst=null;
    int rst;
    public NewPatient() {
        initComponents();
        conn=MySQLConnect.ConnectDB();
        getID();//performs get ID
    }
public void getID()
{// performs get ID
    int tempid=0;
    try{
            String sql2 ="select * from patients";
            pst=conn.prepareStatement(sql2);
            ResultSet rs=pst.executeQuery();
            while(rs.next())
            {
                tempid = rs.getInt(1);
            }
            this.tfID.setText(Integer.toString(tempid+1));
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
}
```

Performs 'getID' when the form is opened.

```
public AppointmentsSearch() {
    initComponents();
    conn=MySQLConnect.ConnectDB();//database connection
    cboID.removeAllItems();
    cboName.removeAllItems();//combobox to retrieve data
    String sql="select id from appointments";
    try{//selecting data from the appointments database
        pst = conn.prepareStatement(sql);
        ResultSet rs = pst.executeQuery();
        while(rs.next()){
            String name=rs.getString("id");
            cboID.addItem(name);//stores data in the combobox
        }
    }catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
    displayItem();//performs displayitem when form is opened
```

Performs 'displayItem' when the form is opened.

```
public void displayItem()
{//performs displayitem when form is opened
    String sql1 ="select * from appointments";
    try {//inserting data from appointments database
        pst1 = conn.prepareStatement(sql1);
         rs=pst1.executeQuery();
         tAppt.setModel(DbUtils.resultSetToTableModel(rs));
    }   //inserting data in the table 'tAppt'
    catch (SQLException ex) {
        Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
public void displayAllItem(String id)
{//performs displayallitem
    try{
        String sql2 ="select * from appointments where id=?";
            pst1=conn.prepareStatement(sql2);
            pst1.setString(1, id);
            rs1=pst1.executeQuery();
            tAppt.setModel(DbUtils.resultSetToTableModel(rs1));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
public void displayAllItemName(String name)
{//performs displayallitemname
    try{
        String sql3 ="select * from appointments where name=?";
            pst2=conn.prepareStatement(sql3);
            pst2.setString(1, name);
            rs2=pst2.executeQuery();
            tAppt.setModel(DbUtils.resultSetToTableModel(rs2));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
```

Performs 'displayItem'/ 'displayAllItem'/ 'displayAllItemName' when the form is opened.

```
public class FindStaff extends javax.swing.JFrame {
    Connection conn=null;
    PreparedStatement pst=null;
    ResultSet rs=null;
    /** database connection
     * Creates new form FindStaff
     */
    public FindStaff() {
        initComponents();
        conn=MySQLConnect.ConnectDB();//database connection
        displayItem();// performs displayitem
    }
```

Performs 'displayItem' when the form is opened.

```
public void displayItem()
{//performs displayitem when form is opened
    String sql ="select * from newstaff";
    try {//inserting data from staff database
        pst = conn.prepareStatement(sql);
         rs=pst.executeQuery();
         tStaff.setModel(DbUtils.resultSetToTableModel(rs));
    }   //inserting data in the table 'tStaff'
    catch (SQLException ex) {
        Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
public void displayAllItem(String id)
{//performs displayallitem
    try{
        String sql ="select * from newstaff where id=?";//selects with relation to ID
            pst=conn.prepareStatement(sql);
            pst.setString(1, id);
            rs=pst.executeQuery();
            tStaff.setModel(DbUtils.resultSetToTableModel(rs));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
public void displayAllItemName(String name)
{//performs displayallitemname
    try{
        String sql ="select * from newstaff where name=?";
            pst=conn.prepareStatement(sql);//selects with relation to name
            pst.setString(1, name);
            rs=pst.executeQuery();
            tStaff.setModel(DbUtils.resultSetToTableModel(rs));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
```

Performs 'displayItem'/ 'displayAllItem'/ 'displayAllItemName' when the form is opened.

```
public class NewStaff extends javax.swing.JFrame {
    String Sex="";//initialising variable
    Connection conn=null;
    PreparedStatement pst=null;//database connection
    int rst;
    public NewStaff() {
        initComponents();
        conn=MySQLConnect.ConnectDB();// database connection
        getID();//getID action performed
    }
public void getID()
{//performing getID
    int tempid=0;
    try{
            String sql2 ="select * from newstaff";// inserting from database
            pst=conn.prepareStatement(sql2);
            ResultSet rs=pst.executeQuery();
            while(rs.next())
            {
                tempid = rs.getInt(1);//setting getID as 1
            }
            this.tfID.setText(Integer.toString(tempid+1));//incrementing getID by 1
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
}
```

Performs 'getID' when the form is opened.

```
public void displayItem()
{//performs displayitem
    String sql1 ="select * from patients";
    try {//Selects data from 'patients' database.
        pst1 = conn.prepareStatement(sql1);
         rs=pst1.executeQuery();
         tPatient.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (SQLException ex) {
        Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
    }
    }
public void displayAllItem(String id)
{//performs displayallitem
    try{
        String sql2 ="select * from patients where id=?";//Selects data with respect to id
            pst1=conn.prepareStatement(sql2);
            pst1.setString(1, id);
            rs1=pst1.executeQuery();
            tPatient.setModel(DbUtils.resultSetToTableModel(rs1));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
public void displayAllItemName(String name)
{//performs displayallitemname
    try{
        String sql3 ="select * from patients where name=?";//Selects data with respect to name
            pst2=conn.prepareStatement(sql3);
            pst2.setString(1, name);
            rs2=pst2.executeQuery();
            tPatient.setModel(DbUtils.resultSetToTableModel(rs2));
        } catch (SQLException ex) {
            Logger.getLogger(NewStaff.class.getName()).log(Level.SEVERE, null, ex);
        }
       }
```

Performs 'displayItem' when the form is opened.

In everyone form there were similar actions performed in the constructor which were repeated across different forms (eg. Staff, patients appointments)

Technique 9- Auto Incrementation and Calculations

When a new set of data is introduced to the database, it automatically pre-enters and increments an integer. Calculations are performed in order to display the patient IDs and appointment IDs.

```
public void getID()
{// performs get ID
    int tempid=0;
    try{
            String sql2 ="select * from patients";// inserting from database
            pst=conn.prepareStatement(sql2);
            ResultSet rs=pst.executeQuery();
            while(rs.next())
            {
                tempid = rs.getInt(1);//setting getID as 1
            }
            this.tfID.setText(Integer.toString(tempid+1));//incrementing getID by 1
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex);
        }
}
```

Initialize id at 1

As it does throughout the while loop, tempid is continually incremented for the following data set.

Technique 10- PDF Generator

This technique allows the user to generate a pdf and store it on their device or they can also print a physical copy for their reference.

```
package IADentalClinic;

import java.awt.HeadlessException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.MessageFormat;// importing messageformat to allow print function
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import net.proteanit.sql.DbUtils;
```

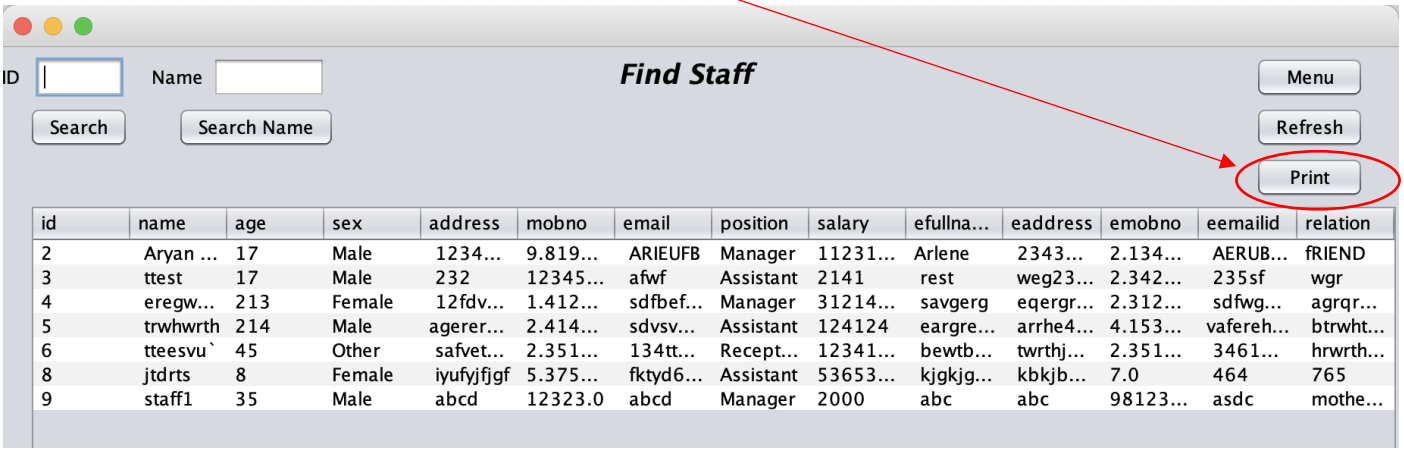
Importing the method for the print function.

Link to Success Criteria 9

```
private void bPrintActionPerformed(java.awt.event.ActionEvent evt) {
    MessageFormat header = new MessageFormat("Data Summary");// retrieving all data from the database
    MessageFormat footer = new MessageFormat(".");
   try{

      tStaff.print(JTable.PrintMode.NORMAL, header, footer);
       // printing the entire database table
   }
   catch (java.awt.print.PrinterException e){
       System.err.format("Cannot Print Bill", e.getMessage());// error message if unable to print

   }
}
```

Print all the data in the table.

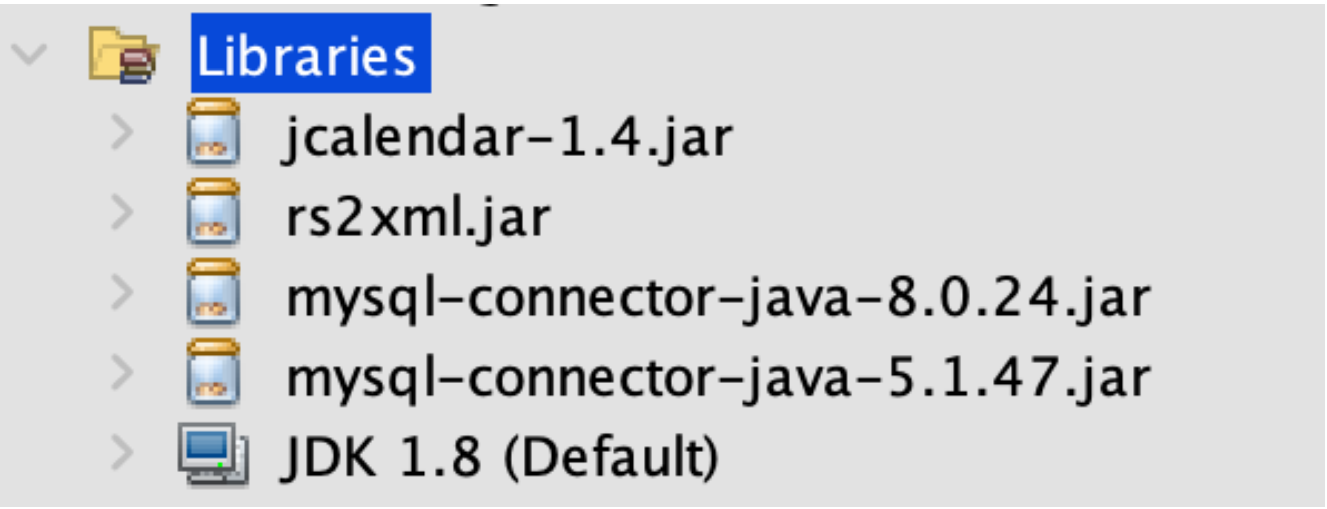
Obtains all the data from the table and prepares for print.



id	name	age	sex	address	mobno	email	position	salary	efullna...	eaddress	emobno	eemailid	relation
2	Aryan ...	17	Male	1234...	9.819...	ARIEUFB	Manager	11231...	Arlene	2343...	2.134...	AERUB...	fRIEND
3	ttest	17	Male	232	12345...	afwf	Assistant	2141	rest	weg23...	2.342...	235sf	wgr
4	eregw...	213	Female	12fdv...	1.412...	sdfbef...	Manager	31214...	savgerg	eqergr...	2.312...	sdfwg...	agrqr...
5	trwhwrth	214	Male	agerer...	2.414...	sdvsv...	Assistant	124124	eargre...	arrhe4...	4.153...	vafereh...	btrwht...
6	tteesvu`	45	Other	safvet...	2.351...	134tt...	Recept...	12341...	bewtb...	twrthj...	2.351...	3461...	hrwrth...
8	jtdrts	8	Female	iyufyjfjgf	5.375...	fktyd6...	Assistant	53653...	kjgkjg...	kbkjb...	7.0	464	765
9	staff1	35	Male	abcd	12323.0	abcd	Manager	2000	abc	abc	98123...	asdc	mothe...

Technique 11- Additional Libraries

Increasing access to data types and formats that are important in software development but are not currently accessible in NetBeans. Jcalendar is a library that was used to obtain the date chooser option while entering the date into the database.



Above displayed are all the additional libraries that were used in the development of the entire system.

Word Count: 993

Works Cited

1. “Downloading Apache NetBeans 13.” *Apache NetBeans*, 4 March 2022, https://netbeans.apache.org/download/nb13/nb13.html. Accessed 23 March 2022.

2. “Java Global Variable Tutorial - Learn Global Variables in Java.” *YouTube*, 19 September 2019, https://www.youtube.com/watch?v=retJEPRJeZA. Accessed 23 March 2022.

3. “JAVA Tutorial - How To Add A Row To JTable From JTextfields in Java NetBeans [With Source Code].” *YouTube*, 17 March 2016, https://www.youtube.com/watch?v=F0Zq2fAUpXg. Accessed 23 March 2022.