



## **ITE222 Final Project Report**

**Project Title: Stamford Hospital application development  
with the database system.**

**Course: ITE222 - Programming II**

**Section 1**

**Term 1-2021**

**Course Teacher: Atikom Srivallop**

**Group Members:**

**Kongpob Wisitsakwasin 2011090005**

**Francis Punyawit Bergmans 1901160018**

**Phachathakorn Prathumma 2010070003**

**Fardina Kabir 1910080005**

**Sukhrit Piyasachadet 2010220008**

## Table of Contents

<b>Introduction .....</b>	<b>2</b>
<b>ER Diagram .....</b>	<b>3</b>
<b>Account .....</b>	<b>4</b>
<b>Cancel .....</b>	<b>6</b>
<b>Create .....</b>	<b>12</b>
<b>Home.....</b>	<b>21</b>
<b>Sign-In.....</b>	<b>24</b>
<b>Sign-Up .....</b>	<b>26</b>
<b>View .....</b>	<b>29</b>
<b>Background creation .....</b>	<b>34</b>
<b>Initial Java codes .....</b>	<b>37</b>
<b>Code connecting database.....</b>	<b>40</b>
<b>Database codes.....</b>	<b>42</b>
<b>Conclusion .....</b>	<b>43</b>

# Introduction

For a hospital or healthcare facility, the Stamford Hospital appointment application was built and designed. Because these apps are frequently used by individuals in distress, we needed to make sure that ours was user-friendly. Users must be able to complete tasks quickly. In general, the app contains pages that users frequently need to view. We'll provide additional information and specifics regarding the content created for each page, as well as website codes and screenshots. We tried to make the application appealing to the users while still being responsive, authoritative, user-friendly, and informative.

To create the application, we used the JAVA language and the JAVA Swing window builder. Furthermore, our group focuses on the program or application, but we will most likely create additional classes and utilize more object-oriented programming in the future. We incorporate the notion of what we will do in week three. In week 4, we decide what we will do about the hospital appointment, then we build our database and begin programming in SQL. Week 6 is when we design and construct the GUI for our software once we finish the database.

Week 7 is when we begin learning about window builders. Week 8 is when we begin coding the account and sign-in pages. In week 9, we begin coding class for database connections and consider interface design. We begin developing the home page and creating pages in week ten. We can see how to have the user's name displayed behind welcome on the home page. We learned how to make a date and time placeholder on the create page. Considering how we utilized the array on this page as well.

As a consequence, we've decided to utilize an array to hold all of the data that the user enters in the Create Appointment form, and this data will be saved in the database after users click the Create button. We cancel and view appointments in week 11. On the cancel page, we plan to create software that checks if the user has entered an appointment id or not. On the view page, we can see how to display a table that lists all of the user's appointments.

In the upcoming section, we will be going over every step of the application that users go through while using the app with screen captures of the app display of every step and java coding used for it, and at the same time give explanations about the ER-diagram based on which the app was made and MySQL Database code used for it.

## ER Diagram

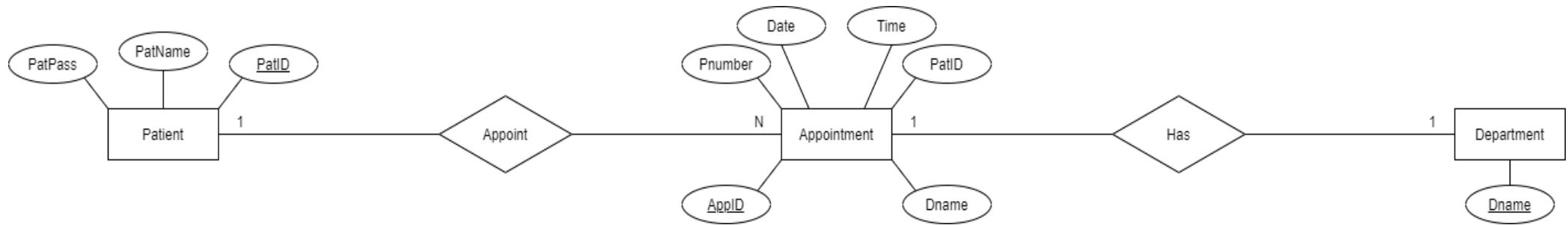


Figure 1: ER Diagram

The ER Diagram above shows three entities that are Patient, Appointment, and Department. The relation between patient and appointment is Appoint and the relation between Appointment and Department is Has. The attributes for the Patient are PatPass, PatName, and PatID. The attributes for the appointment AppID, Pnumber, Date, Time, PatID, and Dname. Finally, the department has the attribute Dname. Therefore, the above figure is a proper representation of the ER Diagram, based on which Java and MySQL codes were utilized for the development of the app and made the app start successfully with complete functions as per order and planned.

## Account

The account page has two options and they are sign-in and sign-up. New users need to choose sign-up since they have not created an account of theirs before. Current users who already have an account, do not require to sign-up and hence they can sign in. The following figure shows the account page and java code of the application.

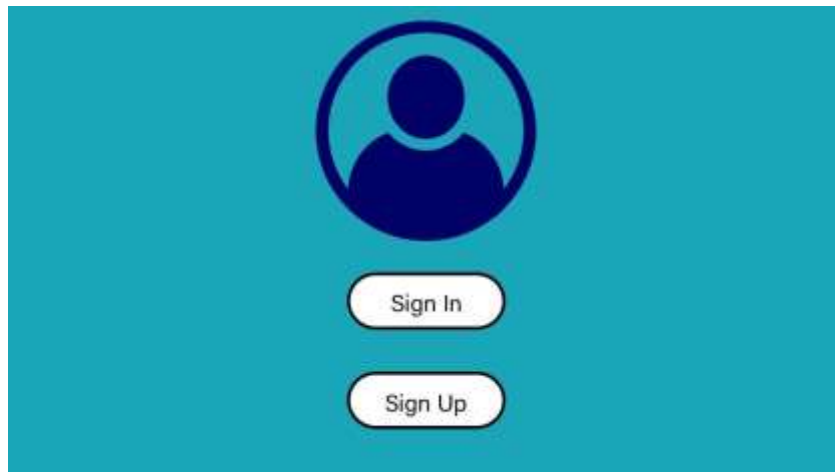


Figure 2: Account page of the app

Figure 3 shows the java coding for the appearance of a sign-up button on the account page.

```
//-----Account Page-----  
  
//create sign up button  
JButton sup_acc = new JButton("");  
sup_acc.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        //close account page  
        account.setVisible(false);  
        //open sign up page  
        signUp.setVisible(true);  
    }  
});  
sup_acc.setCursor(new Cursor(Cursor.HAND_CURSOR));  
sup_acc.setOpaque(false);  
sup_acc.setContentAreaFilled(false);  
sup_acc.setBorderPainted(false);  
sup_acc.setBorder(null);  
sup_acc.setBounds(510, 564, 245, 88);  
account.add(sup_acc);
```

Figure 3: Java code for Sign-up button on the Account page

```
//create sign in button
JButton sin_acc = new JButton("");
sin_acc.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close account page
        account.setVisible(false);
        //open sign in page
        signIn.setVisible(true);
    }
});
sin_acc.setCursor(new Cursor(Cursor.HAND_CURSOR));
sin_acc.setOpaque(false);
sin_acc.setContentAreaFilled(false);
sin_acc.setBorderPainted(false);
sin_acc.setBorder(null);
sin_acc.setBounds(510, 414, 245, 88);
account.add(sin_acc);
```

Figure 4: Java code for Sign in button on the Account page

Figure 4 shows the java coding for the appearance of a sign-in button on the account page.

The above codes were successful in contemplating the function of the Sign-up and Sign-in buttons on the Account page of the app. The creation of the Account page of the app includes creating a sign-up button that codes for close account and open sign-up page. The development of the Account page of the app includes creating a sign-in button as well that codes for close account and open sign-up page.

Therefore, sign-in and sign-up are the two choices on the account page. Because they have never established an account before, new users must select sign-up. Current users who already have an account do not need to register and may login in. The account page and java code for the program are shown in the diagram below.

## Cancel

The page includes Create Appointment, Cancel Appointment, and View Appointment. The application requires user input information for the App ID and Password. The cancel button is made fully functional to cancel any appointments the users may have made before.

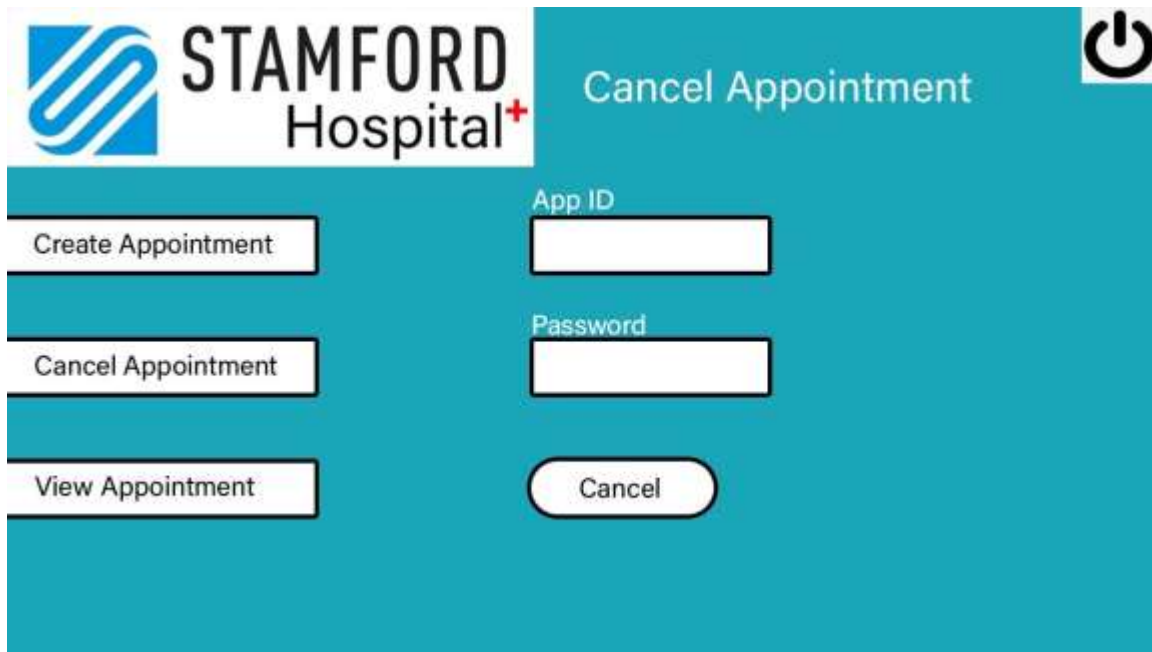


Figure 5: The step for cancelling the appointment with a screen capture of the Cancel Appointment page of the app

The following figures show the Java coding that has been used for the creation of the cancel page. At first, it requires creating a variable stored appointment id, creates a variable stored password, creates a log out button, and creates the confirmed block.

For ‘//yes = 0, no = 1, cancel = 2’, if loop has been used to write close cancel page, open account page, set appointment id text to null, set appointment id text to null, set user name text to null and finally set password text to null.

For the completion of the development of the page, we created a sidebar create button that is also required for writing codes for the close cancel page, set appointment id text to null, set password text to null, delete all rows in JTable, and create variable stored select statements.

```
//-----Cancel Page-----
```

```
//create variable stored appointment id
```

```
TextField appID_can = new TextField();  
appID_can.setBorder(null);  
appID_can.setFont(new Font("Arial", Font.PLAIN, 28));  
appID_can.setBounds(584, 254, 250, 31);  
cancel.add(appID_can);  
appID_can.setColumns(10);
```

```
//create variable stored password
```

```
PasswordField password_can = new PasswordField();  
password_can.setBorder(null);  
password_can.setFont(new Font("Arial", Font.PLAIN, 28));  
password_can.setBounds(584, 390, 250, 31);  
cancel.add(password_can);
```

```
//create log out button
```

```
 JButton logOut_can = new JButton("");  
logOut_can.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        //create confirmed block  
        int confirmLogout = JOptionPane.showConfirmDialog(null, "Do you confirm to log out?", "Select an Option...", JOptionPane.YES_NO_OPTION);  
  
        //yes = 0, no = 1, cancel = 2  
        if (confirmLogout == 0) {  
            //close cancel page  
            cancel.setVisible(false);  
            //open account page  
            account.setVisible(true);  
  
            //set appointment id text to null  
            appID_can.setText("");  
            //set password text to null  
            password_can.setText("");  
  
            //set user name text to null  
            username_sin.setText("");  
            //set password text to null  
            password_sin.setText("");  
        }  
    }  
});
```

Activate Windows  
Go to Settings to activate Windows.

```
logOut_can.setCursor(new Cursor(Cursor.HAND_CURSOR));  
logOut_can.setOpaque(false);  
logOut_can.setContentAreaFilled(false);  
logOut_can.setBorderPainted(false);  
logOut_can.setBorder(null);  
logOut_can.setBounds(1188, 5, 85, 85);  
cancel.add(logOut_can);
```



```

//create side bar create button
JButton cre_sb_can = new JButton("");
cre_sb_can.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close cancel page
        cancel.setVisible(false);
        //open create page
        create.setVisible(true);

        //set appointment id text to null
        appID_can.setText("");
        //set password text to null
        password_can.setText("");
    }
});

```

```

cre_sb_can.setCursor(new Cursor(Cursor.HAND_CURSOR));
cre_sb_can.setOpaque(false);
cre_sb_can.setContentAreaFilled(false);
cre_sb_can.setBorderPainted(false);
cre_sb_can.setBorder(null);
cre_sb_can.setBounds(-7, 240, 344, 58);
cancel.add(cre_sb_can);

```

```

//create side bar view button
JButton view_sb_can = new JButton("");
view_sb_can.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close cancel page
        cancel.setVisible(false);
        //open view page
        view.setVisible(true);

        //set appointment id text to null
        appID_can.setText("");
        //set password text to null
        password_can.setText("");

        DefaultTableModel tbModel = (DefaultTableModel) table.getModel();

        //delete all rows in JTable
        tbModel.setRowCount(0);

        //create variable stored select statement
        String sqlSelect = "SELECT * FROM Appointment WHERE PatID = " + "" + username_sin.getText() + "";
    }
});

```

```

try {
    //run select statement
    ResultSet rs = con.createStatement().executeQuery(sqlSelect);

    while (rs.next()) {

        //these variables store data of each attribute in a row
        String appID = String.valueOf(rs.getInt("AppID"));
        String date = rs.getString("Date");
        String time = rs.getString("Time");
        String name = rs.getString("PatID");
        String phone = rs.getString("Pnumber");
        String dept = rs.getString("Dname");

        //create array stored those variables
        String[] tbData = {appID, date, time, name, phone, dept};

        //add all data in array to JTable
        tbModel.addRow(tbData);
    }
}

```

```

    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}
});
view_sb_can.setCursor(new Cursor(Cursor.HAND_CURSOR));
view_sb_can.setOpaque(false);
view_sb_can.setContentAreaFilled(false);
view_sb_can.setBorderPainted(false);
view_sb_can.setBorder(null);
view_sb_can.setBounds(-5, 509, 344, 58);
cancel.add(view_sb_can);

```

```

//create cancel button
JButton can_cancel = new JButton("");
can_cancel.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create variable stored password inputed and converted to String
        String pass_cancel_input = new String(password_cancel.getPassword());

        //create variable stored select statement
        String sqlSelect = "SELECT AppID, Appointment.PatID, PatPass FROM Patient INNER JOIN Appointment ON Patient.PatID = Appointment.PatID WHERE";

        try {
            //run select statement
            ResultSet rs = con.createStatement().executeQuery(sqlSelect);

            if (rs.next()) {

                /**Correct**

                Statement stat = con.createStatement();

```

```

//create side bar cancel button
JButton can_sb_can = new JButton("");
can_sb_can.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //open cancel page
        cancel.setVisible(true);

        //set appointment id text to null
        appID_can.setText("");
        //set password text to null
        password_can.setText("");
    }
});
can_sb_can.setCursor(new Cursor(Cursor.HAND_CURSOR));
can_sb_can.setOpaque(false);
can_sb_can.setContentAreaFilled(false);
can_sb_can.setBorderPainted(false);
can_sb_can.setBorder(null);
can_sb_can.setBounds(-7, 376, 344, 58);
cancel.add(can_sb_can);

```

```

WHERE Appointment.AppID = " + appID_can.getText() + " AND Patient.PatID = " + "" + username_sin.getText() + "" AND Patient.PetPass = " + "

```

```

try {
    //run select statement
    ResultSet rs = con.createStatement().executeQuery(sqlSelect);

    if (rs.next()) {

        /**Correct**

        Statement stmt = con.createStatement();

```

```

/**Correct**

Statement stmt = con.createStatement();

//create variable stored delete statement
String sqlDelete = "DELETE FROM Appointment WHERE AppID = " + appID_can.getText() + " AND PatID = " + "" + username_sin.getText() + "";

//run delete statement
stmt.execute(sqlDelete);

JOptionPane.showMessageDialog(null, "Cancel Appointment Successfully");

//close cancel page
cancel.setVisible(false);

```

```

/**Correct**
Statement stmt = con.createStatement();
//create variable stored delete statement
String sqlDelete = "DELETE FROM Appointment WHERE AppID = " + appId_con.getText() + " AND PatID = " + "" + username_sin.getText() + """;

//run delete statement
stmt.execute(sqlDelete);

JOptionPane.showMessageDialog(null, "Cancel Appointment Successfully");

//close cancel page
cancel.setVisible(false);
//open home page
home.setVisible(true);

```

Figure 6: The screen captures above are the Java codes for cancelling the page

Creating a variable stored select statement involves a try loop to create a run select statement and the try loop includes a while loop that includes variables storing data of each attribute in a row, creating an array storing those variables, and adding all the data in the array to the JTable.

The above figures have also been used to create a sidebar cancel button, that required writing Java codes for open cancel pages, set appointment id text to null, and set password text to null.

For the creation of the cancel button, we created variable stored password inputted and converted to String and created variable stored select statement. Try loop has been used to run select statements that also requires using an if loop for checking correct or not.

Thus, variable stored delete statement, run delete statement, close cancel page and open home page has been created; appointment id text to null and password text to null has been set, Else loop has been written following if loop, that works when incorrect. The else loop includes setting appointment id text to null and password text to null.

## Create

The Create page is used for creating appointments at the hospital and it has three options on the page, such as the Create Appointment, Cancel Appointment, and finally view appointment. The Create page asks for user input information such as the Date, Time, Phone Number, and Department. The create button has been created for making sure the appointment has been created successfully.

The screenshot shows the 'Create Appointment' page of the Stamford Hospital app. The page features a teal background. In the top left corner, there is the Stamford Hospital logo. In the top right corner, there is a power icon. On the left side, there is a vertical sidebar with three buttons: 'Create Appointment', 'Cancel Appointment', and 'View Appointment'. The main area contains four input fields: 'Date', 'Time', 'Phone Number', and 'Department'. Below these fields is a 'Create' button.

Figure 7: The create page of the app

The making of the create page involved writing a series of java codes such as create a variable stored date that includes making placeholder and create variable stored time that also includes making placeholder.

The code involves creating variable stored phone number, creating variable stored department, creating logout button that involves creating confirmed block and uses if loop for //yes = 0, no = 1. i.e., if (confirmLogOut == 0) that involves close create page, open account page, set phone number text to null, set default text in JComboBox, set user name text to nul, and set password text to null.

The create page also involves creating a sidebar create button that consists of an open create page, including setting phone number text to null and setting default text in JComboBox. Writing codes for creating the page involves creating side bar cancel button that includes close create page, open cancel page, setting phone number text

to null and setting default text in JComboBox. The page has coding for creating a sidebar view button that includes close create page, open view page, set phone number text to null, set default text in JComboBox, delete all rows in JTable and finally create variable stored select statement.

There is a try loop coding for run select statement in the Java coding for creating the create page; while loop is then used followed by the try loop that involves forming variables to store data of each attribute in a row e.g. `String appID = String.valueOf(rs.getInt("AppID"));` `String date = rs.getString("Date");` `String time = rs.getString("Time");` `String name = rs.getString("PatID");` `String phone = rs.getString("Pnumber");` `String dept = rs.getString("Dname");`. And then create an array for storing those variables e.g. `String[] tbData = {appID, date, time, name, phone, dept};`.

Then add all data in the array to JTable. For developing the create page, the Java coding also involves creating create button that needed creating variable stored date input and converted to String, create variable stored time input and converted to String, create variable stored phone number input and converted to String, create variable stored department selected and converted to String, create array to store all information about appointment, add all information about appointment to array, create variable stored appointment id, read data from file and then write data to file finally.



```
//-----Create Page-----

//create variable stored date
JTextField date_cre = new JTextField("YYYY-MM-DD");
date_cre.setForeground(Color.LIGHT_GRAY);
date_cre.addFocusListener(new FocusAdapter() {

    //make placeholder
    @Override
    public void focusGained(FocusEvent e) {
        if (date_cre.getText().equals("YYYY-MM-DD")) {
            date_cre.setText("");
            date_cre.setForeground(Color.BLACK);
            date_cre.setFont(new Font("Arial", Font.PLAIN, 28));
        }
    }

    @Override
    public void focusLost(FocusEvent e) {
        if (date_cre.getText().equals("")) {
            date_cre.setForeground(Color.LIGHT_GRAY);
            date_cre.setFont(new Font("Arial", Font.ITALIC, 28));
            date_cre.setText("YYYY-MM-DD");
        }
    }
});

date_cre.setBorder(null);
date_cre.setFont(new Font("Arial", Font.ITALIC, 28));
date_cre.setBounds(583, 257, 250, 31);
create.add(date_cre);
date_cre.setColumns(10);
```

```
//create variable stored time
JTextField time_cre = new JTextField("HH:MM:SS");
time_cre.setForeground(Color.LIGHT_GRAY);
time_cre.addFocusListener(new FocusAdapter() {
```

```

//make placeholder
@Override
public void focusGained(FocusEvent e) {
    if (time_cre.getText().equals("HH:MM:SS")) {
        time_cre.setText("");
        time_cre.setForeground(Color.BLACK);
        time_cre.setFont(new Font("Arial", Font.PLAIN, 28));
    }
}

@Override
public void focusLost(FocusEvent e) {
    if (time_cre.getText().equals("")) {
        time_cre.setForeground(Color.LIGHT_GRAY);
        time_cre.setFont(new Font("Arial", Font.ITALIC, 28));
        time_cre.setText("HH:MM:SS");
    }
}
});
time_cre.setBorder(null);
time_cre.setFont(new Font("Arial", Font.ITALIC, 28));
time_cre.setBounds(907, 257, 250, 31);
create.add(time_cre);
time_cre.setColumns(10);

```

```

//create variable stored phone number
JTextField phone_cre = new JTextField();
phone_cre.setBorder(null);
phone_cre.setFont(new Font("Arial", Font.PLAIN, 28));
phone_cre.setBounds(583, 393, 250, 31);
create.add(phone_cre);
phone_cre.setColumns(10);

```

```

//create variable stored department
JComboBox<String> dept_cre = new JComboBox<String>();

try {

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM Department");

    if (true) {
        dept_cre.addItem("Choose...");
    }

    while (rs.next()) {
        //create variable stored result in a record
        String specName = rs.getString("Dname");

        //add the result in a record to JComboBox
        dept_cre.addItem(specName);
    }
}

```



```

    } catch (SQLException e1) {
        e1.printStackTrace();
    }

    dept_cre.setBorder(null);
    dept_cre.setFont(new Font("Arial", Font.PLAIN, 22));
    dept_cre.setBounds(907, 393, 250, 31);
    create.add(dept_cre);

```

```

//create log out button
JButton logOut_cre = new JButton("");
logOut_cre.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create confirmed block
        int confirmLogout = JOptionPane.showConfirmDialog(null, "Do you confirm to log out?", "Select an Option...", JOptionPane.YES_NO_OPTION);

        //yes = 0, no = 1
        if (confirmLogout == 0) {
            //close create page
            create.setVisible(false);
            //open account page
            account.setVisible(true);

            date_cre.setText("YYYY-MM-DD");
            date_cre.setForeground(Color.LIGHT_GRAY);
            date_cre.setFont(new Font("Arial", Font.ITALIC, 29));

            time_cre.setText("HH:MM:SS");
            time_cre.setForeground(Color.LIGHT_GRAY);
            time_cre.setFont(new Font("Arial", Font.ITALIC, 28));

            //set phone number text to null
            phone_cre.setText("");

```

```

            //set default text in JComboBox
            dept_cre.setSelectedItem("Choose...");

            //set user name text to null
            username_sin.setText("");
            //set password text to null
            password_sin.setText("");
        }
    }
});

logOut_cre.setCursor(new Cursor(Cursor.HAND_CURSOR));
logOut_cre.setOpaque(false);
logOut_cre.setContentAreaFilled(false);
logOut_cre.setBorderPainted(false);
logOut_cre.setBorder(null);
logOut_cre.setBounds(1188, 5, 85, 85);
create.add(logOut_cre);

```

```

can_sb_cre.setCursor(new Cursor(Cursor.HAND_CURSOR));
can_sb_cre.setOpaque(false);
can_sb_cre.setContentAreaFilled(false);
can_sb_cre.setBorderPainted(false);
can_sb_cre.setBorder(null);
can_sb_cre.setBounds(-7, 373, 344, 58);
create.add(can_sb_cre);

```

```

//create side bar view button
JButton view_sb_cre = new JButton("");
view_sb_cre.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close create page
        create.setVisible(false);
        //open view page
        view.setVisible(true);

        date_cre.setText("YYYY-MM-DD");
        date_cre.setForeground(Color.LIGHT_GRAY);
        date_cre.setFont(new Font("Arial", Font.ITALIC, 28));

        time_cre.setText("HH:MM:SS");
        time_cre.setForeground(Color.LIGHT_GRAY);
        time_cre.setFont(new Font("Arial", Font.ITALIC, 28));

        //set phone number text to null
        phone_cre.setText("");
        //set default text in JComboBox
        dept_cre.setSelectedItem("Choose...");

        DefaultTableModel tbModel = (DefaultTableModel) table.getModel();

```

```

//delete all rows in JTable
tbModel.setRowCount(0);

//create variable stored select statement
String sqlSelect = "SELECT * FROM Appointment WHERE PatID = " + "" + username_sin.getText() + "";

try {
    //run select statement
    ResultSet rs = con.createStatement().executeQuery(sqlSelect);

    while (rs.next()) {
        //these variables store data of each attribute in a row
        String appID = String.valueOf(rs.getInt("AppID"));
        String date = rs.getString("Date");
        String time = rs.getString("Time");
        String name = rs.getString("PatID");
        String phone = rs.getString("Pnumber");
        String dept = rs.getString("Dname");

        //create array stored those variables
        String[] tbData = {appID, date, time, name, phone, dept};

        //add all data in array to JTable
        tbModel.addRow(tbData);
    }
}

```

```

        //add all data in array to jTable
        tbModel.addRow(tbData);
    }

    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}

});
view_sb_cre.setCursor(new Cursor(Cursor.HAND_CURSOR));
view_sb_cre.setOpaque(false);
view_sb_cre.setContentAreaFilled(false);
view_sb_cre.setBorderPainted(false);
view_sb_cre.setBorder(null);
view_sb_cre.setBounds(-7, 507, 344, 58);
create.add(view_sb_cre);

```

```

//create create button
JButton cre_cre = new JButton("");
cre_cre.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create variable stored date inputed and converted to String
        String date_cre_input = new String(date_cre.getText());

        //create variable stored time inputed and converted to String
        String time_cre_input = new String(time_cre.getText());

        //create variable stored phone number inputed and converted to String
        String phone_cre_input = new String(phone_cre.getText());

        //create variable stored department selected and converted to String
        String dept_cre_input = dept_cre.getSelectedItem().toString();

        //create array to store all information about appointment
        ArrayList<String> info_cre_appoint = new ArrayList<String>();
    }
});

```

```

//add all information about appointment to array
info_cre_appoint.add(date_cre_input);
info_cre_appoint.add(time_cre_input);
info_cre_appoint.add(phone_cre_input);
info_cre_appoint.add(dept_cre_input);

//create variable stored appointment id
Integer appID = 10000;

//read data from file
try {
    Scanner read = new Scanner(new File("APP_ID.txt"));

    if (read.hasNext()) {
        appID = read.nextInt();
        appID += 1;
    }
} catch (FileNotFoundException e2) {
    e2.printStackTrace();
}

```

```

//write data to file
try {

    FileWriter write = new FileWriter("APP_ID.txt");

    write.write(appID.toString());
    write.flush();

} catch (IOException e2) {
    e2.printStackTrace();
}

try {

    Statement stmt = con.createStatement();

    //create variable stored insert statement
    String sqlInsert = "INSERT INTO Appointment VALUES (" + appID.toString() + ", " + info_cre_appoint.get(0) + ", " + info_cre_appoint.get(
    //run the insert statement
    stmt.execute(sqlInsert);

    JOptionPane.showMessageDialog(null, "Create Appointment Successfully");

```

```

//run the insert statement
stmt.execute(sqlInsert);

JOptionPane.showMessageDialog(null, "Create Appointment Successfully");

```

```

t.get(0) + ", " + info_cre_appoint.get(1) + ", " + username_sin.getText() + ", " + info_cre_appoint.get(2) + ", " + info_cre_appoint.get(3) + "));

```

```

//read data from file
try {

    Scanner read = new Scanner(new File("APP_ID.txt"));

    if (read.hasNext()) {
        appID = read.nextInt();
        appID -= 1;
    }

} catch (FileNotFoundException e2) {
    e2.printStackTrace();
}

//write data to file
try {

    FileWriter write = new FileWriter("APP_ID.txt");

    write.write(appID.toString());
    write.flush();

```

```

//close create page
create.setVisible(false);
//open home page
home.setVisible(true);

date_cre.setText("YYYY-MM-DD");
date_cre.setForeground(Color.LIGHT_GRAY);
date_cre.setFont(new Font("Arial", Font.ITALIC, 28));

time_cre.setText("HH:MM:SS");
time_cre.setForeground(Color.LIGHT_GRAY);
time_cre.setFont(new Font("Arial", Font.ITALIC, 28));

//set phone number text to null
phone_cre.setText("");
//set default text in JComboBox
dept_cre.setSelectedItem("Choose...");

} catch (SQLException e1) {

    if (info_cre_appoint.get(3).equals("Choose...")) {
        JOptionPane.showMessageDialog(null, "Please choose the department");
    }
}

```

```

//write data to file
try {

    FileWriter write = new FileWriter("APP_ID.txt");

    write.write(appID.toString());
    write.flush();

} catch (IOException e2) {
    e2.printStackTrace();
}

}

});
cre_cre.setCursor(new Cursor(Cursor.HAND_CURSOR));
cre_cre.setOpaque(false);
cre_cre.setContentAreaFilled(false);
cre_cre.setBorderPainted(false);
cre_cre.setBorder(null);
cre_cre.setBounds(568, 507, 218, 67);
create.add(cre_cre);

```

Figure 8: Java codes for the development of the Create page of the app



# Home

The homepage includes the create appointment, cancel appointment, and view appointment as shown in figure 9. The page includes a graphical picture of a calendar with a timer to make it aesthetically pleasing to users and making the interface easy to use and friendly at the same time. In the following sections of the home page, we will be giving a few explanations about the java coding that we did for creating the home page.

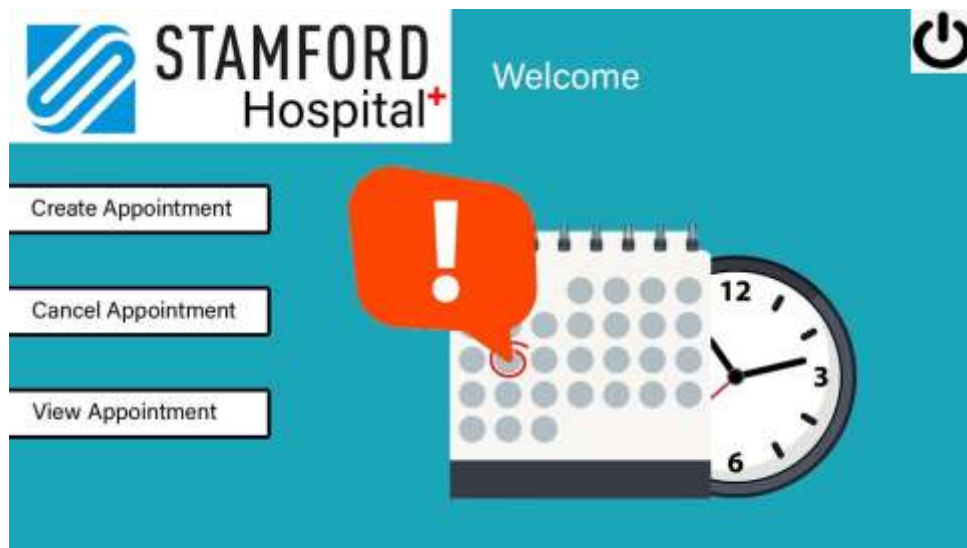


Figure 9: The homepage of the App

```
//-----Home Page-----  
  
//create table for showing the appointment of a user  
JTable table = new JTable();  
table.setFont(new Font("Arial", Font.PLAIN, 14));  
table.setModel(new DefaultTableModel(  
    new Object[][] {  
    },  
    new String[] {  
        "AppID", "Date", "Time", "Username", "Phone", "Department"  
    }  
));  
scrollPane.setViewportView(table);  
  
logOut_home.setCursor(new Cursor(Cursor.HAND_CURSOR));  
logOut_home.setOpaque(false);  
logOut_home.setContentAreaFilled(false);  
logOut_home.setBorderPainted(false);  
logOut_home.setBorder(null);  
logOut_home.setBounds(1188, 5, 85, 86);  
home.add(logOut_home);
```

```

//create create side bar button
JButton cre_sb_home = new JButton("");
cre_sb_home.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close home page
        home.setVisible(false);
        //open create page
        create.setVisible(true);
    }
});
cre_sb_home.setCursor(new Cursor(Cursor.HAND_CURSOR));
cre_sb_home.setOpaque(false);
cre_sb_home.setContentAreaFilled(false);
cre_sb_home.setBorderPainted(false);
cre_sb_home.setBorder(null);
cre_sb_home.setBounds(-7, 240, 344, 58);
home.add(cre_sb_home);

```

```

//create variable stored select statement
String sqlSelect = "SELECT * FROM Appointment WHERE PatID = " + "'" + username_sin.getText() + "'";

try {
    //run select statement
    ResultSet rs = con.createStatement().executeQuery(sqlSelect);

    while (rs.next()) {
        //these variables store data of each attribute in a row
        String appID = String.valueOf(rs.getInt("AppID"));
        String date = rs.getString("Date");
        String time = rs.getString("Time");
        String name = rs.getString("PatID");
        String phone = rs.getString("Pnumber");
        String dept = rs.getString("Dname");

        //create array stored those variables
        String[] tbData = {appID, date, time, name, phone, dept};

        //add all data in array to JTable
        tbModel.addRow(tbData);
    }
}

```

```

    } catch (SQLException e1) {
        e1.printStackTrace();
    }
}

});
view_sb_home.setCursor(new Cursor(Cursor.HAND_CURSOR));
view_sb_home.setOpaque(false);
view_sb_home.setContentAreaFilled(false);
view_sb_home.setBorderPainted(false);
view_sb_home.setBorder(null);
view_sb_home.setBounds(-5, 509, 344, 58);
home.add(view_sb_home);

```

```

//create side bar cancel button
JButton can_sb_home = new JButton("");
can_sb_home.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close home page
        home.setVisible(false);
        //open cancel page
        cancel.setVisible(true);
    }
});
can_sb_home.setCursor(new Cursor(Cursor.HAND_CURSOR));
can_sb_home.setOpaque(false);
can_sb_home.setContentAreaFilled(false);
can_sb_home.setBorderPainted(false);
can_sb_home.setBorder(null);
can_sb_home.setBounds(-7, 376, 344, 58);
home.add(can_sb_home);

```

Figure 10: Java code for the creation of the homepage of the app

The figure above shows the Java code for the creation of the homepage of the app. At first, it includes creating table for showing the appointment of a user, creating logout button, creating confirmed block and if loop for 'yes = 0, no = 1, cancel = 2' i.e., if (confirmLogout == 0); the if loop therefore includes close home page, open account page, setting user name text to null and finally setting password text to null. The Java code for the creation of the homepage of the app also includes creating create sidebar button that involves close home page and open create page.

The java code also creates a sidebar view button that involves coding for close home page, open view page, delete all rows in JTable and finally create a variable stored select statement. The try loop has been used for run select statements. There is a while loop used following the try loop where the while loop includes coding for variables to store data of each attribute in a row, for instance, String appID = String.valueOf(rs.getInt("AppID")); String date = rs.getString("Date"); String time = rs.getString("Time"); String name = rs.getString("PatID"); String phone = rs.getString("Pnumber"); and String dept = rs.getString("Dname");. In the while loop, create an array for storing those variables, e.g. String[] tbData = {appID, date, time, name, phone, dept};. Then finally add all data in the array to JTable. The Java code for the creation of the homepage of the app includes creating a sidebar cancel button that includes coding for close home page and open the cancel page.



# Sign-In

The sign-in page of the app enables users for information input i.e., both for the Username and password and also means that users need to log in with the username and password that they have already chosen during account creation that is while doing the sign-up step or when registering the account. Therefore, to illustrate this more clearly, we used a screen capture of the sign-in page of the app and the overall java codes used for creating the sign-in page of the app.



Figure 11: Screen capture of Sign In page of the app

```
//-----Sign In Page-----  
  
//create variable stored password  
JPasswordField password_sin = new JPasswordField();  
password_sin.setBorder(null);  
password_sin.setFont(new Font("Arial", Font.PLAIN, 28));  
password_sin.setBounds(690, 491, 254, 31);  
signIn.add(password_sin);  
  
//create variable stored user name  
JTextField username_sin = new JTextField();  
username_sin.setBorder(null);  
username_sin.setFont(new Font("Arial", Font.PLAIN, 28));  
username_sin.setBounds(336, 491, 254, 31);  
signIn.add(username_sin);  
username_sin.setColumns(10);  
  
//create variable stored user name showed behind welcome  
JLabel welcome = new JLabel("");  
welcome.setForeground(Color.WHITE);  
welcome.setFont(new Font("Arial", Font.PLAIN, 43));  
welcome.setBounds(841, 73, 321, 43);  
home.add(welcome);
```

```

//create sign in button
JButton sin_sin = new JButton("");
sin_sin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create variable stored password inputed and converted to String
        String pass_sin_input = new String(password_sin.getPassword());

        //create variable stored select statement
        String sqlSelect = "SELECT PatID, " +
"PatPass FROM Patient WHERE PatID = " + "'" + username_sin.getText() + "'" +
        " AND PatPass = " + "'" + pass_sin_input + "'";

        try {
            //run select statement
            ResultSet rs = con.createStatement().executeQuery(sqlSelect);

            if (rs.next()) {
                /**Correct**

                //create reminded block
                JOptionPane.showMessageDialog(null, "Sign In Successfully");

                //set user name behind welcome in home page
                welcome.setText(username_sin.getText());

                //close sign in page
                signIn.setVisible(false);
                //open home page
                home.setVisible(true);
            } else {
                //incorrect
                JOptionPane.showMessageDialog(null, "Incorrect, please try again");

                //set user name text to null
                username_sin.setText("");
                //set password text to null
                password_sin.setText("");
            }
        }
    }
});

```

```

        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
});
sin_sin.setCursor(new Cursor(Cursor.HAND_CURSOR));
sin_sin.setOpaque(false);
sin_sin.setContentAreaFilled(false);
sin_sin.setBorderPainted(false);
sin_sin.setBorder(null);
sin_sin.setBounds(526, 592, 214, 67);
signIn.add(sin_sin);

```

Figure 12: The screen captures above represent the java code for the development of the sign-in page of the app

The java coding for the development of the sign-in page of the app includes creating a sign-in button, creating variable stored password input info and converting it to String and creating variable stored select statement.

The java coding also includes an if loop for run select statements e.g. "if(rs.next());". The coding used when the "if loop" is correct, includes creating a reminder block, setting user name behind welcome in home page, close sign in page and open home page.

The else code following the "if loop" includes codes when incorrect, and they are setting user name text to null and password text to null. The java coding for the development of the sign-in page of the app includes creating a back button that includes close sign in page, open account page, set user name text to null and finally set password text to null.

## Sign-Up

The sign-up page of the app enables users for information input i.e. For the Username, password and name, and also means that users need to register with the username of their choice, password and their name. Therefore, to illustrate this more clearly, we used a screen capture of the sign-up page of the app and the overall java codes used for creating the sign-up page of the app.



Figure 13: Screen capture of Sign-Up page of the app

```
//create back button
JButton back_sin = new JButton("");
back_sin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close sign in page
        signIn.setVisible(false);
        //open account page
        account.setVisible(true);

        //set user name text to null
        username_sin.setText("");
        //set password text to null
        password_sin.setText("");
    }
});
back_sin.setCursor(new Cursor(Cursor.HAND_CURSOR));
back_sin.setOpaque(false);
back_sin.setContentAreaFilled(false);
back_sin.setBorderPainted(false);
back_sin.setBorder(null);
back_sin.setBounds(-7, 5, 92, 91);
signIn.add(back_sin);
```

```
//create sign up button
JButton sup_sup = new JButton("");
sup_sup.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create variable stored password inputed and converted to String
        String pass_sup_input = new String(password_sup.getPassword());

        try {
            Statement stmt = con.createStatement();
```

The java codes used for creating the sign-up page of the app includes create variable stored user name, create variable stored first name, create variable stored password and create back button.

```

Statement stmt = con.createStatement();

//create variable stored insert statement
String sqlInsert = "INSERT INTO `patient` (`PatID`, `PatPass`, `PatName`) VALUES ('" + username_sup.getText() + "', '" + pass_sup_input + "', '" +
`PatName`) VALUES ('" + username_sup.getText() + "', '" + pass_sup_input + "', '" + name_sup.getText() + "')";

//run insert statement
stmt.execute(sqlInsert);

JOptionPane.showMessageDialog(null, "Sign Up Successfully");

//close sign up page
signUp.setVisible(false);
//open account page
account.setVisible(true);

```

Figure 14: Java codes for the development of the Sign-in page

The coding for the create back button includes close sign-up page, open account page, set user name text to null, set name text to null and set password text to null. The java code used for creating the sign-up page of the app includes a create sign up button as well that includes creating a variable stored password inputted and converted to String. There is a try loop for creating variable stored insert statements that needs coding for, run insert statements, close sign-up page, open account page, set user name text to null, set name text to null and finally set password text to null.

## View

The view page of the app enables users to show the appointments that they have created. The view page includes the create appointment, cancel appointment, and view appointment buttons as well. Therefore, to illustrate this more clearly, we used a screen capture of the view page of the app and the overall java codes used for creating the view up page.



Figure 15: View page of the app

```
//-----View Page-----

//create log out button
JButton logOut_view = new JButton("");
logOut_view.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //create confirmed block
        int confirmLogOut = JOptionPane.showConfirmDialog(null, "Do you confirm to log out?", "Select an Option...", JOptionPane.YES_NO_OPTION);

        //yes = 0, no = 1
        if (confirmLogOut == 0) {
            //close view page
            view.setVisible(false);
            //open account page
            account.setVisible(true);

            //set user name text to null
            username_sin.setText("");
            //set password text to null
            password_sin.setText("");
        }
    }
});
```

```
logOut_view.setCursor(new Cursor(Cursor.HAND_CURSOR));
logOut_view.setOpaque(false);
logOut_view.setContentAreaFilled(false);
logOut_view.setBorderPainted(false);
logOut_view.setBorder(null);
logOut_view.setBounds(1188, 5, 85, 85);
view.add(logOut_view);
```

```
defined
logOut_view.setBounds(1188, 5, 85, 85);
view.add(logOut_view);

//create side bar create button
JButton cre_sb_view = new JButton("");
cre_sb_view.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close view page
        view.setVisible(false);
        //open create page
        create.setVisible(true);
    }
});
cre_sb_view.setCursor(new Cursor(Cursor.HAND_CURSOR));
cre_sb_view.setOpaque(false);
cre_sb_view.setContentAreaFilled(false);
cre_sb_view.setBorderPainted(false);
cre_sb_view.setBorder(null);
cre_sb_view.setBounds(-7, 237, 344, 58);
view.add(cre_sb_view);
```



```

//create side bar cancel button
JButton can_sb_view = new JButton("");
can_sb_view.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //close view page
        view.setVisible(false);
        //open cancel page
        cancel.setVisible(true);
    }
});
can_sb_view.setCursor(new Cursor(Cursor.HAND_CURSOR));
can_sb_view.setOpaque(false);
can_sb_view.setContentAreaFilled(false);
can_sb_view.setBorderPainted(false);
can_sb_view.setBorder(null);
can_sb_view.setBounds(-7, 373, 344, 58);
view.add(can_sb_view);

```

```

//create side bar view button
JButton view_sb_view = new JButton("");
view_sb_view.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //open view page
        view.setVisible(true);

        DefaultTableModel tbModel = (DefaultTableModel) table.getModel();

        //delete all rows in JTable
        tbModel.setRowCount(0);

        //create variable stored select statement
        String sqlSelect = "SELECT * FROM Appointment WHERE PatID = " + "'" + username_sin.getText() + "'";

        try {
            //run select statement
            ResultSet rs = con.createStatement().executeQuery(sqlSelect);

            while (rs.next()) {
                //these variables store data of each attribute in a row
                String appID = String.valueOf(rs.getInt("AppID"));
            }
        }
    }
});

```



```

        while (rs.next()) {

            //these variables store data of each attribute in a row
            String appID = String.valueOf(rs.getInt("AppID"));
            String date = rs.getString("Date");
            String time = rs.getString("Time");
            String name = rs.getString("PatID");
            String phone = rs.getString("Pnumber");
            String dept = rs.getString("Dname");

            //create array stored those variables
            String[] tbData = {appID, date, time, name, phone, dept};

            //add all data in array to JTable
            tbModel.addRow(tbData);
        }

    } catch (SQLException e1) {
        e1.printStackTrace();
    }

}

});

```

```

view_sb_view.setCursor(new Cursor(Cursor.HAND_CURSOR));
view_sb_view.setOpaque(false);
view_sb_view.setContentAreaFilled(false);
view_sb_view.setBorderPainted(false);
view_sb_view.setBorder(null);
view_sb_view.setBounds(-7, 507, 344, 58);
view.add(view_sb_view);

```

```

//-----Create BS-----
JLabel imgAcc = new JLabel("");
imgAcc.setBounds(-7, 5, 1280, 720);
imgAcc.setIcon(new ImageIcon("C:\\Users\\HMD\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Account.jpg"));
account.add(imgAcc);
signIn.setLayout(null);

JLabel imgSin = new JLabel("");
imgSin.setBounds(-7, 5, 1280, 720);
imgSin.setIcon(new ImageIcon("C:\\Users\\HMD\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Sign In.jpg"));
signIn.add(imgSin);
signUp.setLayout(null);

JLabel imgSup = new JLabel("");
imgSup.setBounds(-7, 5, 1280, 720);
imgSup.setIcon(new ImageIcon("C:\\Users\\HMD\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Sign Up.jpg"));
signUp.add(imgSup);
home.setLayout(null);

JLabel imgHome = new JLabel("");
imgHome.setBounds(-7, 5, 1280, 720);
imgHome.setIcon(new ImageIcon("C:\\Users\\HMD\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Home.jpg"));
home.add(imgHome);
cancel.setLayout(null);

```

Figure 16: Java codes for the development of the view page

The Java codes for the development of the view page includes creating logout button that needs coding for create confirmed block and if loop for "yes = 0, no = 1" e.g., "if (confirmLogout == 0)" that needs coding for close view page, open account page, set user name text to null and set password text to null finally. The Java code for the development of the view page also includes a create sidebar create button that needs coding for close view page and open create page. The Java code involves creating sidebar cancel button that needs coding for close view page and open cancel page. The Java code for the development of the view page includes creating a sidebar view button that needs coding for the open view page, delete all rows in JTable, and create variable stored select statements. There is a try loop for run select statement and the while loop e.g., "while (rs.next())" following the try loop that codes for creating variables that will store data of each attribute in a row, create array for those stored variables e.g. String[] tbData = {appID, date, time, name, phone, dept}; and then add all data in array to JTable finally.

## Background creation

A Graphical Interface (GUI) is a visual representation of communication that allows the user to engage with the machine more easily. The acronym GUI stands for Graphical User Interface. It is a popular user interface that incorporates graphical representations such as buttons and icons, and communication may be accomplished by interacting with these icons rather than the more traditional text-based or command-based methods. There have been screen captures of the app that had GUI created in most of the figures presented before. The GUI of our app had the background color set to a form of blue theme-colored, along with the logo of Stamford Hospital and a few graphical pictures used as icon or symbol for some pages. Therefore, we will be presenting about the java codes for the UI of the Stamford Hospital application clearly in the upcoming figure.

```
//-----Create BG-----
JLabel imgAcc = new JLabel("");
imgAcc.setBounds(-7, 5, 1280, 720);
imgAcc.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Account.jpg"));
account.add(imgAcc);
signIn.setLayout(null);

JLabel imgSin = new JLabel("");
imgSin.setBounds(-7, 5, 1280, 720);
imgSin.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Sign In.jpg"));
signIn.add(imgSin);
signUp.setLayout(null);

JLabel imgSup = new JLabel("");
imgSup.setBounds(-7, 5, 1280, 720);
imgSup.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Sign Up.jpg"));
signUp.add(imgSup);
home.setLayout(null);

JLabel imgHome = new JLabel("");
imgHome.setBounds(-7, 5, 1280, 720);
imgHome.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Home.jpg"));
home.add(imgHome);
cancel.setLayout(null);
```

Activate Windows

```
JLabel imgCan = new JLabel("");
imgCan.setBounds(-7, 5, 1280, 720);
imgCan.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Cancel.jpg"));
cancel.add(imgCan);
create.setLayout(null);

JLabel imgCre = new JLabel("");
imgCre.setBounds(-7, 5, 1280, 720);
imgCre.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\Create.jpg"));
create.add(imgCre);
view.setLayout(null);

JLabel imgView = new JLabel("");
imgView.setBounds(-7, 5, 1280, 720);
imgView.setIcon(new ImageIcon("C:\\Users\\KONG\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital Appointment\\GUI\\View.jpg"));
view.add(imgView);
}
}
```

Figure 17: Java codes for background creation

## Initial Java codes

```
import java.awt.EventQueue;

import java.awt.Cursor;
import javax.swing.JFrame;
import java.awt.CardLayout;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.ImageIcon;
import java.awt.Toolkit;
import javax.swing.JButton;
import javax.swing.JTextField;

import java.sql.Statement;
import java.util.ArrayList;
import java.util.Scanner;
import java.awt.Font;
import javax.swing.JPasswordField;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
import java.awt.Color;
import java.awt.event.FocusAdapter;
```

```
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;

import javax.swing.JComboBox;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
```

```

public class MyApp {

    private JFrame frmHospitalAppointment;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MyApp window = new MyApp();
                    window.frmHospitalAppointment.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
}

```

```

/**
 * Create the application.
 */
public MyApp() { initialize(); }

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    //about a database connection
    ConnDB connectDB = new ConnDB();
    Connection con = connectDB.conAndCheckDB();

    //-----Create Frame-----
    frmHospitalAppointment = new JFrame();
    frmHospitalAppointment.setIconImage(Taukit.getDefaultTaukit().getImage("C:\\Users\\HNM\\Desktop\\IT\\Programming\\Program\\Java\\Project\\Hospital\\tauikit.png"));
    frmHospitalAppointment.setTitle("Hospital Appointment");
    frmHospitalAppointment.setBounds(100, 100, 1300, 720);
    frmHospitalAppointment.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frmHospitalAppointment.getContentPane().setLayout(new CardLayout(0, 0));
}

```



```
//-----Create Panel-----
JPanel account = new JPanel();
frmHospitalAppointment.getContentPane().add(account, "name_52639400151400");
account.setLayout(null);

JPanel signIn = new JPanel();
frmHospitalAppointment.getContentPane().add(signIn, "name_53471734861800");

JPanel signUp = new JPanel();
frmHospitalAppointment.getContentPane().add(signUp, "name_54570855376800");

JPanel home = new JPanel();
frmHospitalAppointment.getContentPane().add(home, "name_31189105718700");

JPanel cancel = new JPanel();
frmHospitalAppointment.getContentPane().add(cancel, "name_81266251937900");

JPanel create = new JPanel();
frmHospitalAppointment.getContentPane().add(create, "name_1002291246100");

JPanel view = new JPanel();
frmHospitalAppointment.getContentPane().add(view, "name_60931060230200");

//-----Create Scroll Pane-----

JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(422, 237, 758, 339);
view.add(scrollPane);
```

Figure 18: Initial Java codes for starting the creation of the app

The initial codes for starting the coding for the development of the app needed a lot of imports that have been illustrated by the screen captures presented. The coding above represents the code for launching and creating the application, then initializing the contents of the frame that includes java coding for creating frame, creating panel and finally creating scroll pane.



## **Code connecting database**

The interface file `AbtDB.java` is utilized, and the `ConDB.java` file contains the codes that result in the creation of a connection between Java and the database. The `AbtDB.java` and `ConDB.java` files, containing code for establishing a connection between the Java coding programming language and the MySQL Database, are shown in the following diagram.

```
ConDB.java
import java.sql.Connection;
import java.sql.DriverManager;

import javax.swing.JOptionPane;

public class ConDB implements AbtDB {

    public static String dbName = "Hospital_Appointment";
    public static String user = "root";
    public static String pass = "";
    public static String hostName = "localhost";
    public static String dbDriver = "com.mysql.jdbc.Driver";

    @Override
    public Connection conAndCheckDB() {
        try {
            Class.forName(dbDriver);
            String url = "jdbc:mysql://" + hostName + "/" + dbName;
            Connection connect = DriverManager.getConnection(url, user, pass);

            if (connect != null) {
                JOptionPane.showMessageDialog(null, "Connected Database");
            }
            return connect;
        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Not Connected Database");
        }
        return null;
    }
}

AbtDB.java
import java.sql.Connection;

public interface AbtDB {

    Connection conAndCheckDB();
}
```

Figure 19: AbtDB.java is used for an interface and ConDB.java file consists of codes that result in the formation of connection between Java and the database

## Database codes

The screenshots display the MySQL Database, which was used to construct the Appointment, Department, and Patient tables. Appoint is the relationship between Patient and Appointment, while Has been the relationship between Appointment and Department. PatPass, PatName, and PatID are the Patient's characteristics. AppID, Pnumber, Date, Time, PatID, and Dname are the characteristics for the appointment. Finally, Dname is an attribute for the department. As a result, the following codes are a good depiction of the ER Diagram, which was used to create the app using Java and MySQL scripts, resulting in the program starting properly with all functions as ordered and intended.

```
DROP DATABASE IF EXISTS `Hospital_Appointment`;
CREATE DATABASE `Hospital_Appointment`;

USE `Hospital_Appointment`;

DROP TABLE IF EXISTS `Appointment`;
CREATE TABLE `Appointment` (
  `AppID` INT(5) NOT NULL,
  `Date` DATE NOT NULL,
  `Time` TIME NOT NULL,
  `PatID` VARCHAR(15),
  `Pnumber` CHAR(10) NOT NULL,
  `Dname` VARCHAR(45),
  PRIMARY KEY (`AppID`),
  FOREIGN KEY (`PatID`) REFERENCES `Patient` (`PatID`), FOREIGN KEY (`Dname`) REFERENCES `Department` (`Dname`)
);

DROP TABLE IF EXISTS `Department`;
CREATE TABLE `Department` (
  `Dname` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Dname`)
);

INSERT INTO `Department` VALUES ('Medicine'), ('Heart'), ('Respiratory'), ('Brain'), ('Bone'), ('Obstetric'), ('Skin'), ('Digestive'),

DROP TABLE IF EXISTS `Patient`;
CREATE TABLE `Patient` (
  `PatID` VARCHAR(15) NOT NULL,
  `PatPass` VARCHAR(15) NOT NULL,
  `PatName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`PatID`)
);

INSERT INTO `Patient` VALUES ('Admin', 'Admin12345', 'Admin');
```

Figure 20: The screen capture shows MySQL Database that has been used to create tables for Appointment, Department and Patient

## Conclusion

Overall, we have been through every step of the application and they are, the development of the ER Diagram, creation of the pages such as Account, Cancel, Create, Home, Sign-In, Sign-Up and View pages, Java files named AbtDB.java and ConDB.java containing code connecting database and Java application, Initial codes of Java, Database Table Codes and Background creation or GUI that users go through while using the app. The screen captures of the app displays every step and java coding used for it previously, and at the same time give explanations about the ER-diagram based on which the app was made and MySQL Database code used for it.

Our program uses JAVA language and window builder JAVA Swing to make the program. In addition, our group focuses on the program or application, but in the future, we will add more classes and use more object-oriented programming, certainly in our program. In week 3, we integrate the idea of what we will do. In week 4, we conclude that we will do hospital appointments after that we design our database and start coding SQL language. When we finish the database, we design and create GUI for our program in week 6. In week 7, we start studying window builder.

In week 8, we start coding the account and sign in. In week 9, we start coding class for database connection and also think about interfaces. In week 10, we start coding the home page and creating a page. In the home page, we find how to make the name of the user appear behind welcome. For the create page, we find how to make placeholders in date and time. Also, think about how to use arrays on this page. As a result, we decided to use an array to store all information that users fill in to create appointments, and this information will be stored in the database after you click the create button.

In week 11, we cancel and view appointments. In the cancel page, we think that we will make a program to check that the user has an appointment id that is inputted or not. In the view page, we find how to show table that shows all appointments of the user. Object Oriented Programming makes us put the information easily in the parameter, such as text. If our project doesn't have Object Oriented Programming, we have to write code a lot. It is like if we don't have it, we have to write the structure all the time.

Every industry has its own set of design rules for applications. For the most part, they are very similar. Hospitals must bear in mind, however, that the great majority

of patients who visit their application are searching for urgent help or information prior to making an appointment. As a result, hospitals should aim to follow a set of basic principles that we discussed before. The following characteristics must be provided on hospital websites: It should be straightforward to use. Display the most commonly accessed information and files from a central location. Allow anybody to be completely available on a mobile device. We must keep these requirements in mind while designing apps for hospitals.