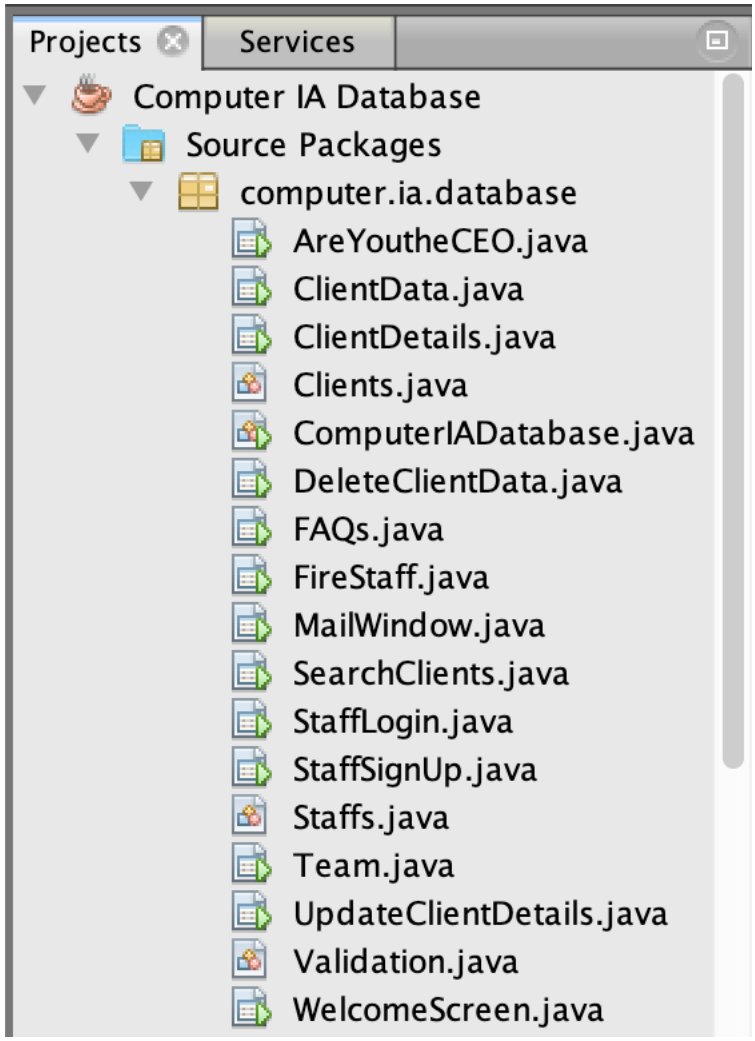
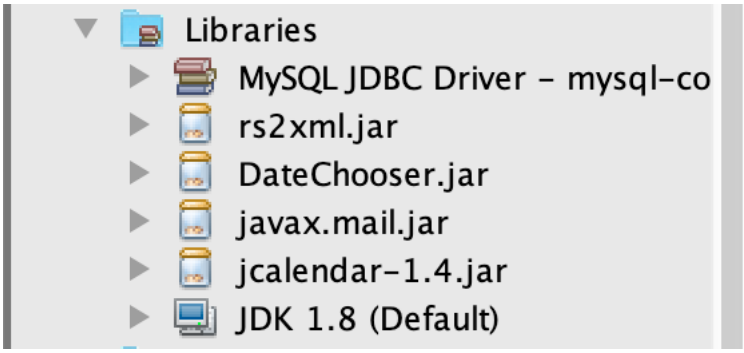
Criterion C: Development

All the Classes and Methods used:



All the Libraries used:



All the Imports used:

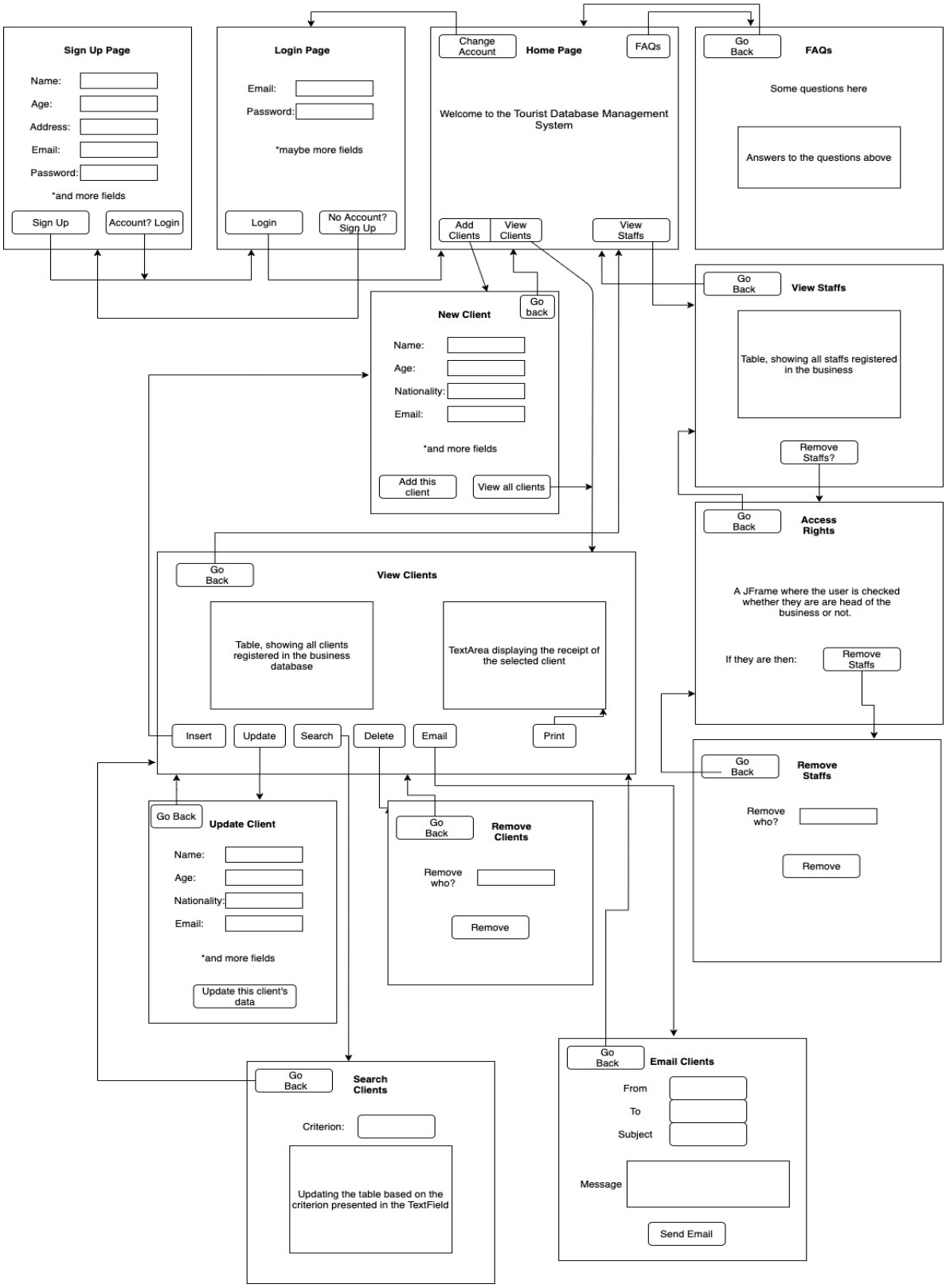
```
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import java.sql.Connection;
import java.util.ArrayList;
import javax.swing.table.DefaultTableModel;
import java.sql.PreparedStatement;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.swing.RowFilter;
import javax.swing.table.TableRowSorter;
import javax.swing.JFrame;
import javax.swing.table.TableModel;
```

Techniques used:

1. Database,
2. Data Verification,
3. Graphical User Interface (GUI),
4. Methods,
5. Exception Handling,
6. SQL Commands,
7. ArrayList,
8. Accessor Methods.

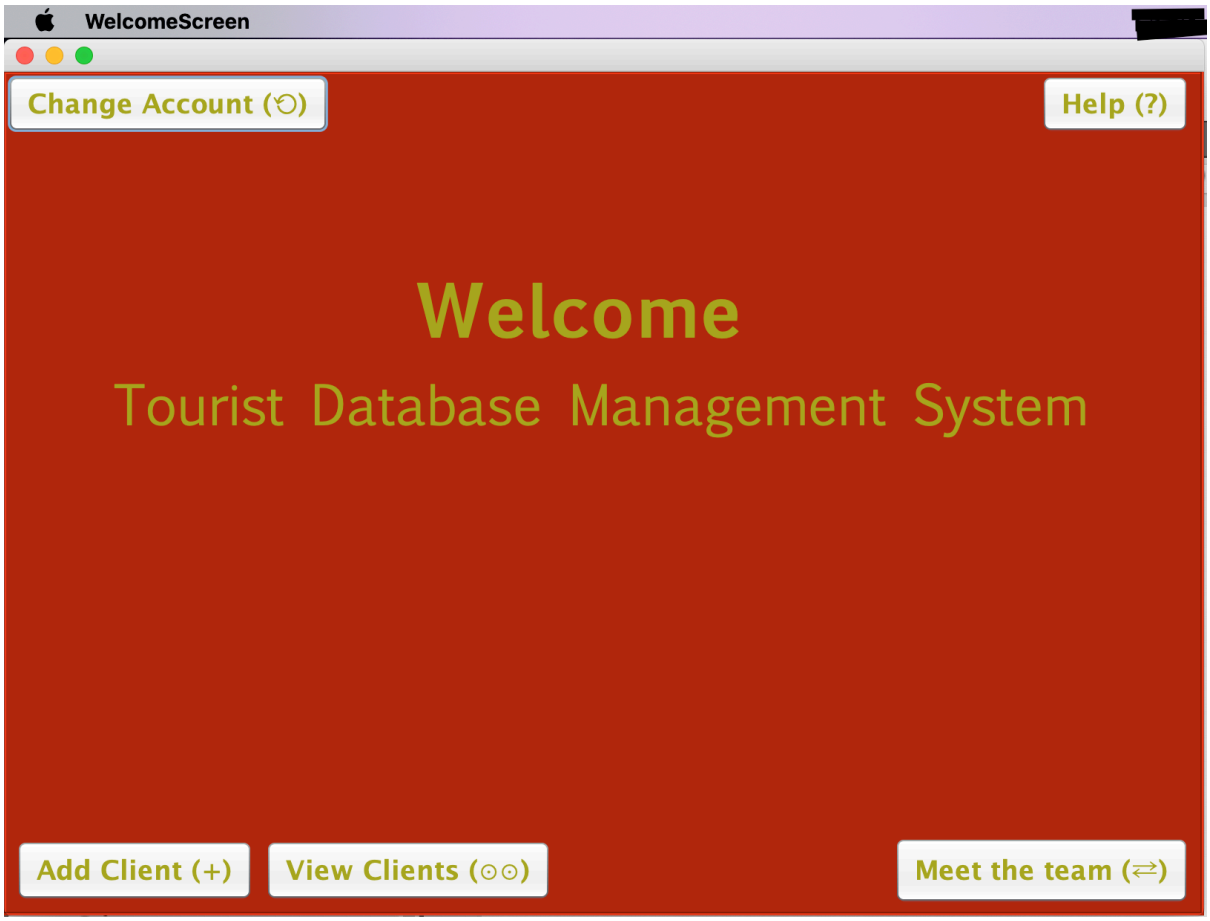
JFrames of the Software[1]

The following JFrames shows drawn format of the frames



[1] Technique no.3: Graphical User Interface (GUI)

Main Window (WelcomeScreen.java)

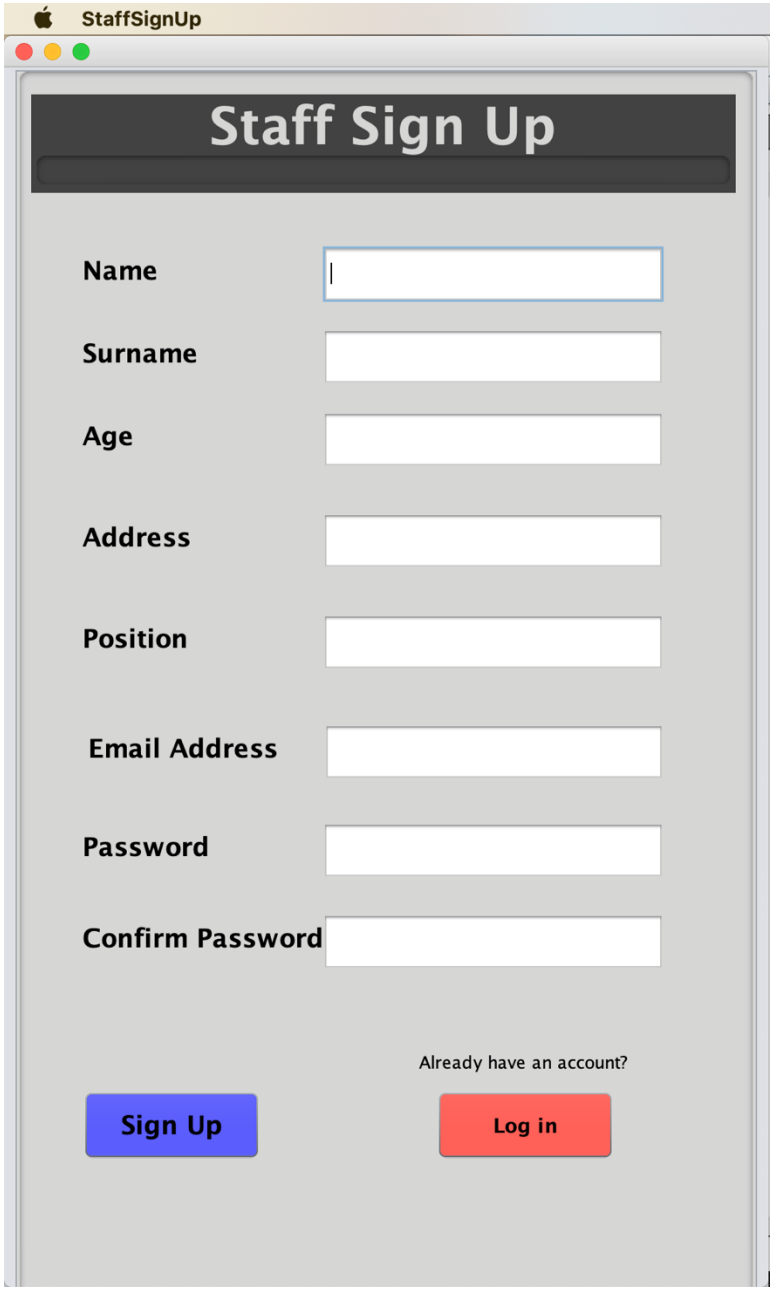


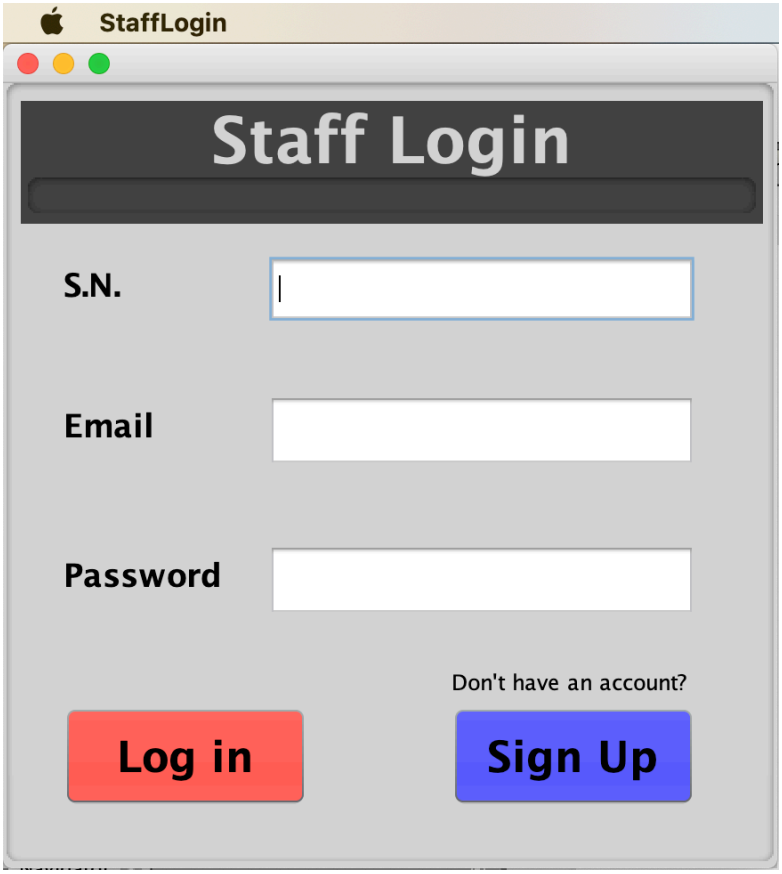
This JFrame is shown after logging into the database.

From this window, each of the 5 buttons have different functionalities

```
private void jBtnChangeActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    StaffLogin bus = new StaffLogin();
    bus.setVisible(true);
    dispose();

}

private void jbtnFAQsActionPerformed(java.awt.event.ActionEvent evt) {

    FAQs car = new FAQs();
    car.setVisible(true);
    dispose();

}

private void jbtnAddClientActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    ClientDetails bike = new ClientDetails();
    bike.setVisible(true);
    dispose();

}

private void jbtnViewClientsActionPerformed(java.awt.event.ActionEvent evt) {

    ClientData bus = new ClientData();
    bus.setVisible(true);
    dispose();

}

private void jbtnMeetTheTeamActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    Team car = new Team();
    car.setVisible(true);
    dispose();

}
```

1. “Change Account” takes the user back to the sign-up pages. They would have to sign-up and log-in to the database so that they can edit the client data.





I implemented the codes that will ensure no null values are entered.

```
if (jEmail.getText().trim().isEmpty() && jPass.getText().trim().isEmpty() && jConpass.getText().trim().isEmpty() &&
    lbl_email.setText("Email field is empty");
    lbl_pass.setText("Password field is empty");
    lbl_conpass.setText("Confirm Password field is empty");
    lbl_name.setText("Name field is empty");
    lbl_surname.setText("Surname field is empty");
    lbl_age.setText("Age field is empty");
    lbl_address.setText("Address field is empty");
    lbl_position.setText("Position field is empty");
}
```

Every field needs to have element in it to be processed. This feature is implemented throughout the program where needed.

Additionally, I have also used **regex** to check if the email is in standard format or not.

[2]

```
public class Validation {

    public static boolean email_Validation (String email){

        boolean status = false;

        String email_Pattern = "^[a-zA-Z0-9]+[@]{1}+[a-zA-Z0-9]+[.]{1}+[a-zA-Z0-9]+$";

        Pattern pattern  = Pattern.compile(email_Pattern);
        Matcher matcher = pattern.matcher(email);

        if (matcher.matches()){
            status = true;
        }
        else{
            status = false;
        }

        return status;
    }

}
```

I called this method in the JFrame with the following code.

```
boolean status = Validation.email_Validation(jEmail.getText());
```

Also, implemented a password checker inside the regex so that the passwords fields match

```
if (Password.equals(Conpassword)){
```

After everything is true, then the entered data will be transferred to a table in the database and the new user can be logged in after this window collapses and the login window pops up.

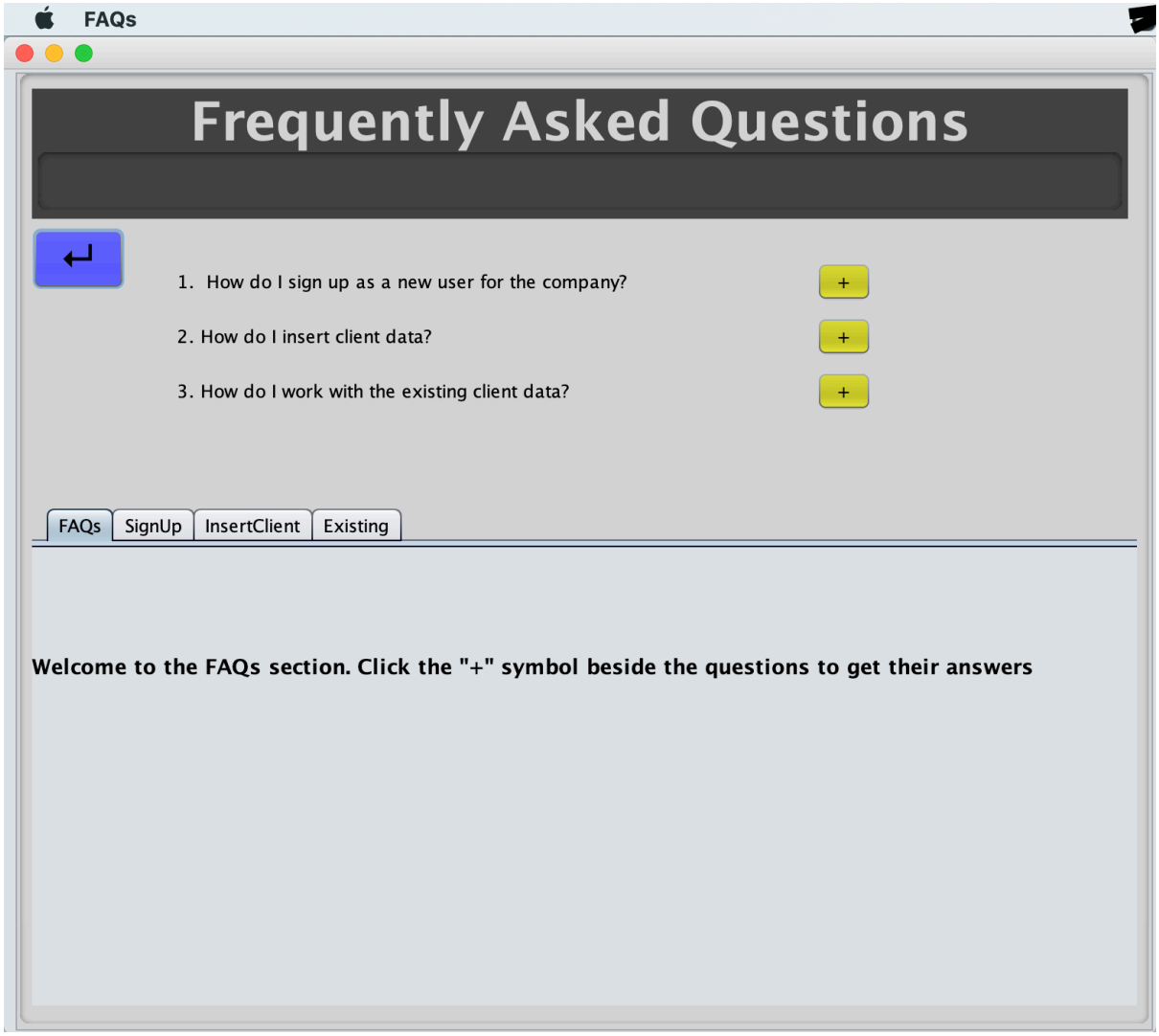
The StaffLogin.java window will only ask for the S.N., email address and password.

In the staff table, the S.N, email and password were in columns 1, 7 and 8 respectively. So keeping that in mind, I used the .getString for their respective columns from the table,

```
while (rs.next()){
    if (rs.getString(1).equals(sn) && rs.getString(7).equals(email) && rs.getString(8).equals(password)){
        log = 0;
        break;
    }
}
```

[2] Technique no.2: Data Verification

2. "Help (?)" takes the user to a Frequently Asked Questions (FAQs) section. There are some common questions that have their respective answers below in the jTabbedPane.

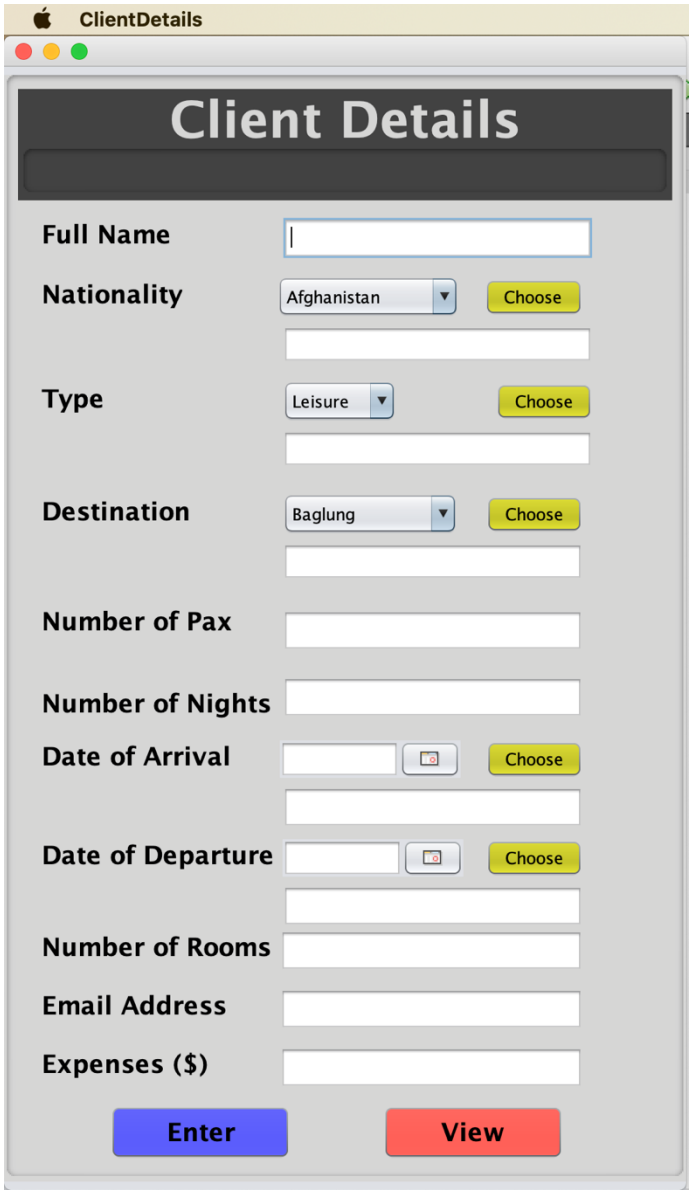


For answer to their question, they simply have to click the '+' button next to it and it will update the jTabbedPane below it.

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

    FAQsTabbed.setSelectedIndex(1);
}
```

I utilized the index number feature to display the relevant answers. So, index '1' would be the second tab.

3. "Add Client" takes the user to a JFrame that enables them to enter client data to the database.

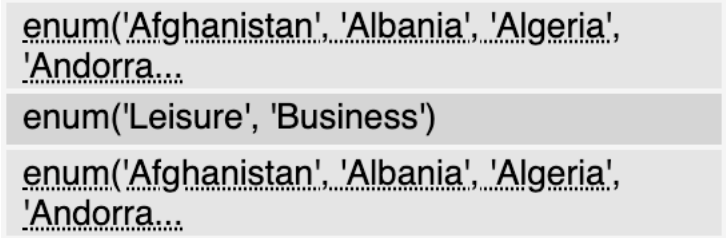


The staff/user enters data to the database. This data is stored in a client table in the database.

My database contains 3 tables: StaffInformation (previously shown), ClientInformation(this) and SecretPIN(upcoming).[3]

Besides the normal jTextFields, I have also used jComboBox and jDateChooser.

The combo boxes list all the relevant options for that field. Previously I opted for taking the options from the database by using 'enum' data type

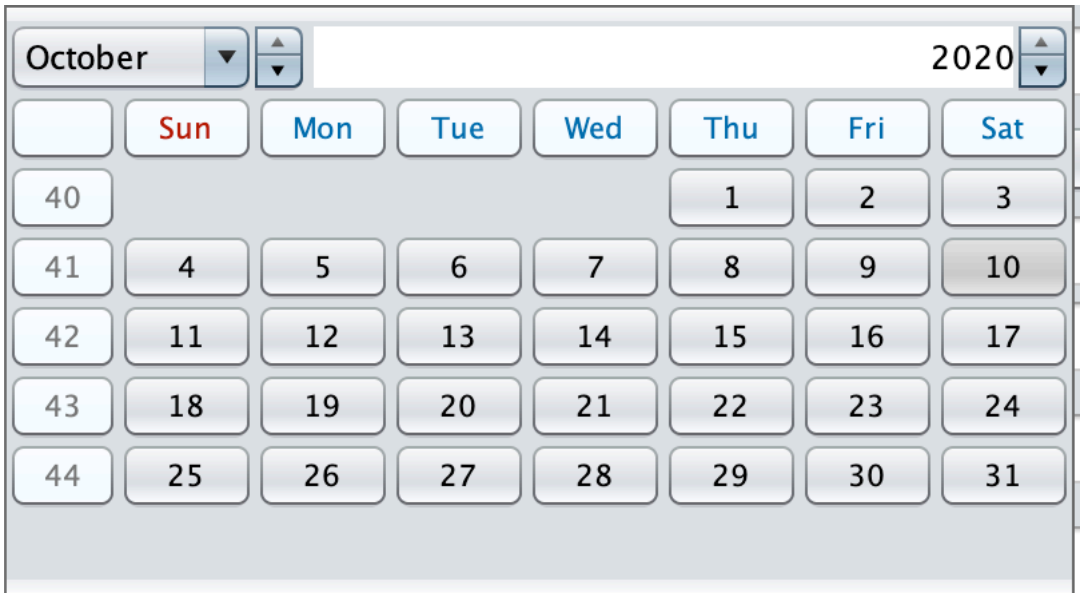


Instead, I kept the contents in the combo box itself.

The 'Choose' button next to it transfers the chosen option to the text field below.

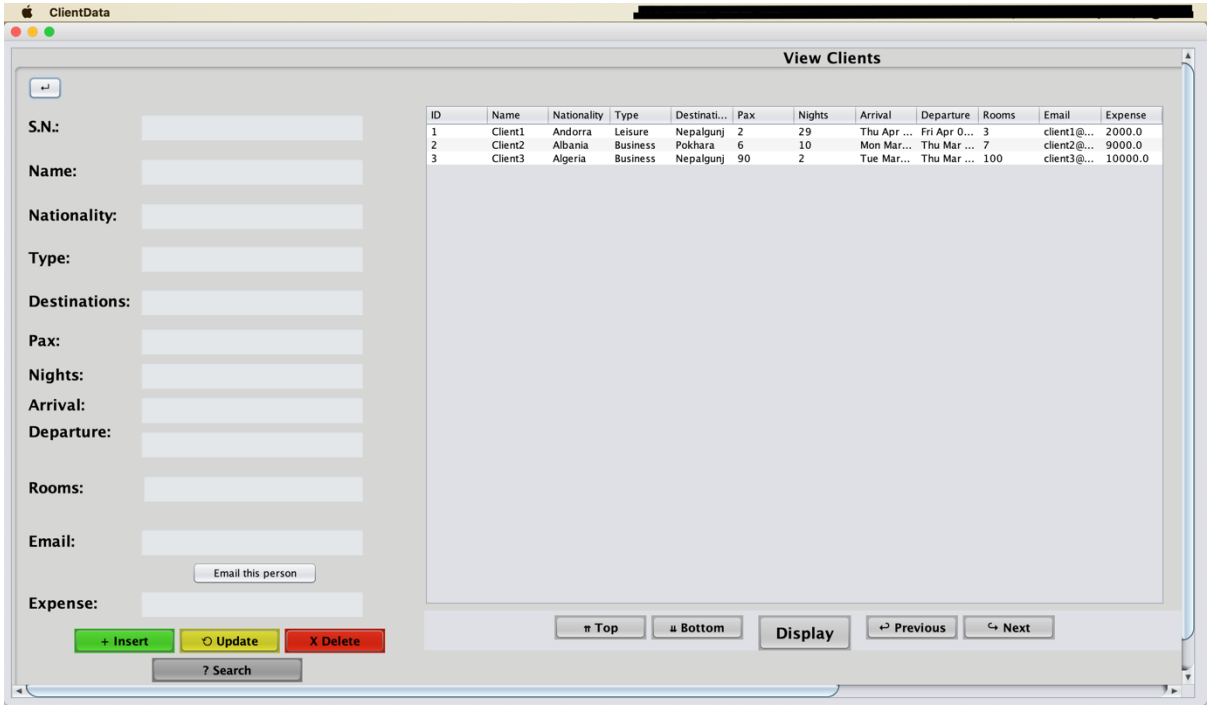
[3] Technique no.1: Database

```
private void jclick1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jtextnationality.setText(jcombonationality.getSelectedItem().toString());
}
```



The date chooser allows to select the date for arrival and departure for that particular destination. Uses the datechooser and calendar library.

4. “View Clients”: The ‘View’ button in this JFrame and the ‘View Clients’ from the Main window serve the same purpose of taking the user to the main window for clients



The table above autofill itself with the data from the database. Used rs2xml.jar library and to make it possible, this method is utilized as getters, they are public, accessible to the entire program.

[4]

```
public class Clients {

    private int sn;
    private String name;
    private String nationality;
    private String type;
    private String destinations;
    private int pax;
    private int nights;
    private String arrival;
    private String departure;
    private int rooms;
    private String email;
    private float expense;
```

[4] Technique no.4: Methods

5

```
public Clients(int psn, String pname, String pnationality, String ptype, String pdestinations, int ppax, int pnights,

    this.sn = psn;
    this.name = pname;
    this.nationality = pnationality;
    this.type = ptype;
    this.destinations = pdestinations;
    this.pax = ppax;
    this.nights = pnights;
    this.arrival = parrival;
    this.departure = pdeparture;
    this.rooms = prooms;
    this.email = pemail;
    this.expense = pexpense;

}
```

```
public int getSN(){
    return sn;
}

public String getName(){
    return name;
}

public String getNationality(){
    return nationality;
}

public String getType(){
    return type;
}

public String getDestinations(){
    return destinations;
}

public int getPax(){
    return pax;
}
```

The above method is called here:

[6]Try-catch exception handling is used throughout to ensure smooth running of the program

```
   public ArrayList<Clients> getClientList(){

try{
            ArrayList<Clients> clientList = new ArrayList<Clients>();
            Connection conn = DriverManager.getConnection ("jdbc:mysql://localhost:8889/ClientDatabase","root","root");

            String query = "Select * from ClientInformation";

            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery(query);
            Clients client;

            while (rs.next()){
                client = new Clients(rs.getInt("sn"), rs.getString("name"), rs.getString("nationality"), rs.getString("type"), rs.ge
                clientList.add(client);
            }

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

[5] Technique no.8: Accessor Methods
[6] Technique no.5: Exception Handling

This code functions to copy the table data from the database to an ArrayList because arrays are easy to populate, edit upon and transfer data to and from.

[7]

```
public void Show_Clients_In_JTable(){

    ArrayList <Clients> list = getClientList();
    DefaultTableModel model = (DefaultTableModel)JTable_Clients.getModel();

    //Clear JTable
    model.setRowCount(0);
    Object[] row = new Object [12];
    for (int i = 0; i < list.size(); i++){

        row[0] = list.get(i).getSN();
        row[1] = list.get(i).getName();
        row[2] = list.get(i).getNationality();
        row[3] = list.get(i).getType();
        row[4] = list.get(i).getDestinations();
        row[5] = list.get(i).getPax();
        row[6] = list.get(i).getNights();
        row[7] = list.get(i).getArrival();
        row[8] = list.get(i).getDeparture();
        row[9] = list.get(i).getRooms();
        row[10] = list.get(i).getEmail();
        row[11] = list.get(i).getExpense();

        model.addRow(row);

    }

}
```

Now the populated ArrayList is then transferred onto the table in the JFrame row by row and column by column.

[7] Technique no.7: ArrayList

```
public void ShowItem(int index){
    txt_sn.setText(Integer.toString(getClientList().get(index).getSN()));
    txt_name.setText(getClientList().get(index).getName());
    txt_nationality.setText(getClientList().get(index).getNationality());
    txt_type.setText(getClientList().get(index).getType());
    txt_destinations.setText(getClientList().get(index).getDestinations());
    txt_pax.setText(Integer.toString(getClientList().get(index).getPax()));
    txt_nights.setText(Integer.toString(getClientList().get(index).getNights()));
    txt_arrival.setText(getClientList().get(index).getArrival());
    txt_departure.setText(getClientList().get(index).getDeparture());
    txt_rooms.setText(Integer.toString(getClientList().get(index).getRooms()));
    txt_email.setText(getClientList().get(index).getEmail());
    txt_expense.setText(Float.toString(getClientList().get(index).getExpense()));
}
```

As soon as a row item is selected, its values will also be shown in the jTextFields. This ShowItem() method will do that and will further allow to make a receipt for the client.

Under the table, I kept some navigation buttons like ‘Top’ to change the rows selected.

```
private void Btn_NextActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    id2 = txt_sn.getText();
    int id3 = Integer.parseInt(id2);
    int pos = id3;

    if (pos >= getClientList().size()){
        pos = pos+1;
        pos = getClientList().size()-1;
    }
    ShowItem(pos);

}

private void Btn_PreviousActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    pos2--;

    if (pos2<0){
        pos2 = 0;
    }

    ShowItem(pos2);
}

private void Btn_BottomActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    pos1 = getClientList().size()-1;
    ShowItem(pos1);

}

private void Btn_TopActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    pos1 = 0;
    ShowItem(pos1);

}
```

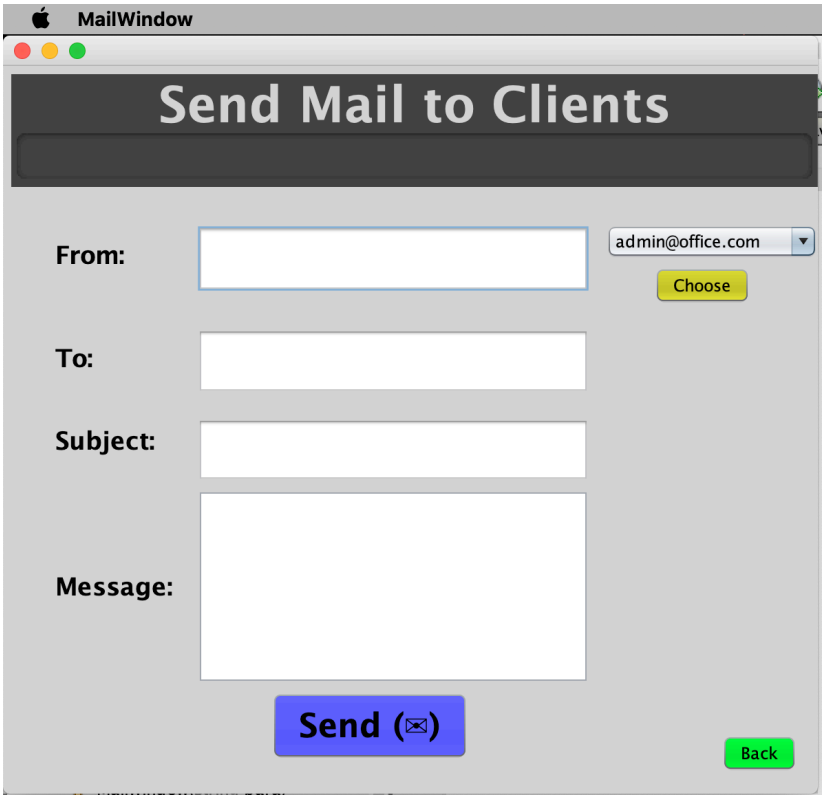
The Display button will take the items in the jTextfields on the left and paste it onto the receipt on the right.

```
private void Btn_DisplayActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String sn0 = (txt_sn.getText());
    String name0 = (txt_name.getText());
    String nationality0 = (txt_nationality.getText());
    String type0 = (txt_type.getText());
    String destinations0 = (txt_destinations.getText());
    String pax0 = (txt_pax.getText());
    String nights0 = (txt_nights.getText());
    String arrival0 = (txt_arrival.getText());
    String departure0 = (txt_departure.getText());
    String rooms0 = (txt_rooms.getText());
    String email0 = (txt_email.getText());
    String expense0 = (txt_expense.getText());



    area_receipt.append("\t     Client Information Pvt Ltd\n"
            +"\n ========================================="
            +"\nS.N.:\t                                "+sn0
            +"\nName:\t                                "+name0
            +"\nNationality:\t                               "+nationality0
            +"\nType:\t                                "+ type0
            +"\nDestinations:\t                                 "+destinations0
            +"\nPax:\t                               "+pax0
            +"\nNights:\t                                  "+nights0
            +"\nDate of Arrival:\t                                      "+arrival0
            +"\nDate of Departure:\t    "+departure0
            +"\nRooms:                                                  "+ rooms0
            +"\nEmail:                                                     "+ email0
            +"\nExpense:                                                "+ expense0
            +"\n ========================================="
            +"\n\n                                  Always at your service\n\t                  Office Firm"
            +"\n\n\n"
    )
            ;
}
```

The ‘Print’ button will print the receipt in hard copy form from a printer. The .print() method allows to open up the window already preset in NetBeans.

```
private void Btn_PrintActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{

        //if (!area_receipt.equals("")){
    area_receipt.print();

    }
    //}
    catch(Exception ex){
        System.out.println("ERROR..");
    }
}
```

‘Email this person’ button opens up a new JFrame shown below.




```
private void jBtnEmailActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (txt_email.getText().trim().isEmpty()){
        JOptionPane.showMessageDialog(null, "Please select a client to message", "No Email", JOptionPane.ERROR_MESSAGE);
    }

    else{
    String Receiver = txt_email.getText();
    new MailWindow(Receiver).setVisible(true);
    }

}
```

It takes the email address from the textfield, and inserts into the ‘To:’ field of this jFrame.

```
else{

System.out.println("Preparing to send email");
Properties prop = new Properties();

prop.put("mail.smtp.auth", "true");
prop.put("mail.smtp.starttls.enable", "true");
prop.put("mail.smtp.host", "smtp.gmail.com");
prop.put("mail.smtp.port", "587");

Session session = Session.getDefaultInstance(prop, new Authenticator() {
    @Override
    protected PasswordAuthentication getPasswordAuthentication(){
        return new PasswordAuthentication (Sender, Password);
    }
     });
try{
    MimeMessage message = new MimeMessage(session);
    message.setFrom(new InternetAddress(Sender));
    message.setRecipient (Message.RecipientType.TO, new InternetAddress(Receiver));
    message.setSubject(Subject);
    message.setText(Contents);
    Transport.send(message);
    System.out.println("Success");

}catch(Exception ex){
    System.out.println("" + ex);
}
ClientData cube  = new ClientData();
cube.setVisible(true);
dispose();
}
```

The ‘Send’ button uses Java Mail API (along with its library) to send mail to the inbox. SMTP is used for gmail and its port to authenticate. MimeMessage takes message and subject of the email and Transports to the inbox. This requires the sender email to ‘on’ the less secure apps feature on their account.

In ClientData.java, The buttons on the bottom left enables the user to edit the client data.

‘Insert’ button opens up the ClientData.java JFrame

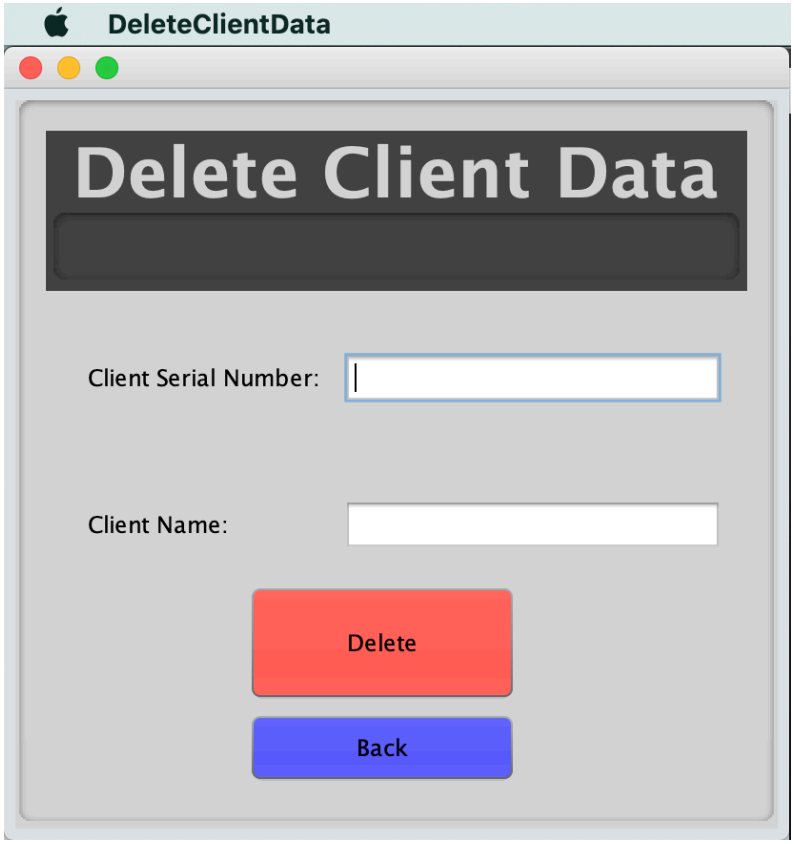
‘Update’ button opens up UpdateClientDetails.java.

The field taken for reference is the primary key for this table from which other details can be changed.

```
String sql = "update ClientInformation set  name = ?, "
        + "nationality = ?, "
        + "type = ?, "
        + "destinations = ?, "
        + "pax = ?, "
        + "nights = ?, "
        + "arrival = ?, "
        + "departure = ?, "
        + "rooms = ?, "
        + "email = ?, "
        + "expense = ? "
        + "where sn = ?";
```

[8]SQL commands used to edit table.

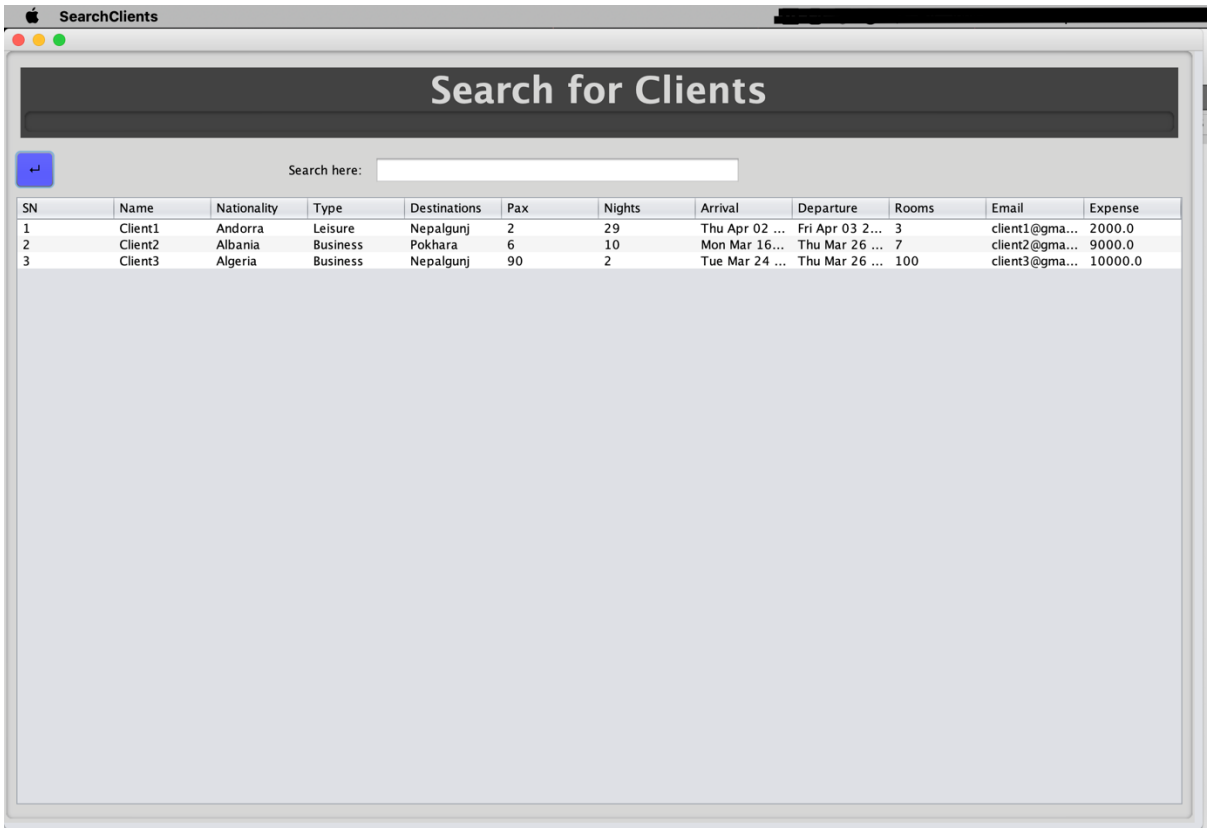
The **'Delete'** button opens up the following window where the user has to keep the S.N. and name of the client they want to delete



[8] Technique no.6: SQL Commands

```
private void jbtnDeleteActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        Connection conn=DriverManager.getConnection ("jdbc:mysql://localhost:8889/ClientDatabase","root","root");
        String sql = "Delete from ClientInformation WHERE sn = ? AND name = ?";
        String sn = jSN.getText();
        String name = jName.getText();
        PreparedStatement st=conn.prepareStatement (sql);
        st.setString(1, sn);
        st.setString(2, name);
        st.executeUpdate();

        System.out.println(sn + ": " + name + " tuple deleted");
        JOptionPane.showMessageDialog(null, sn + ": " + name + " tuple deleted", "Success!", JOptionPane.ERROR_MESSAGE);
        jSN.setText("");
        jName.setText("");

        st.close();
        conn.close();
    }
        catch (SQLException ex) {
        System.out.println(ex);
    }
}
```

The **'Search'** button opens up the following window and allows the user to search for a particular client from the entire client table based on what they type in the search bar.

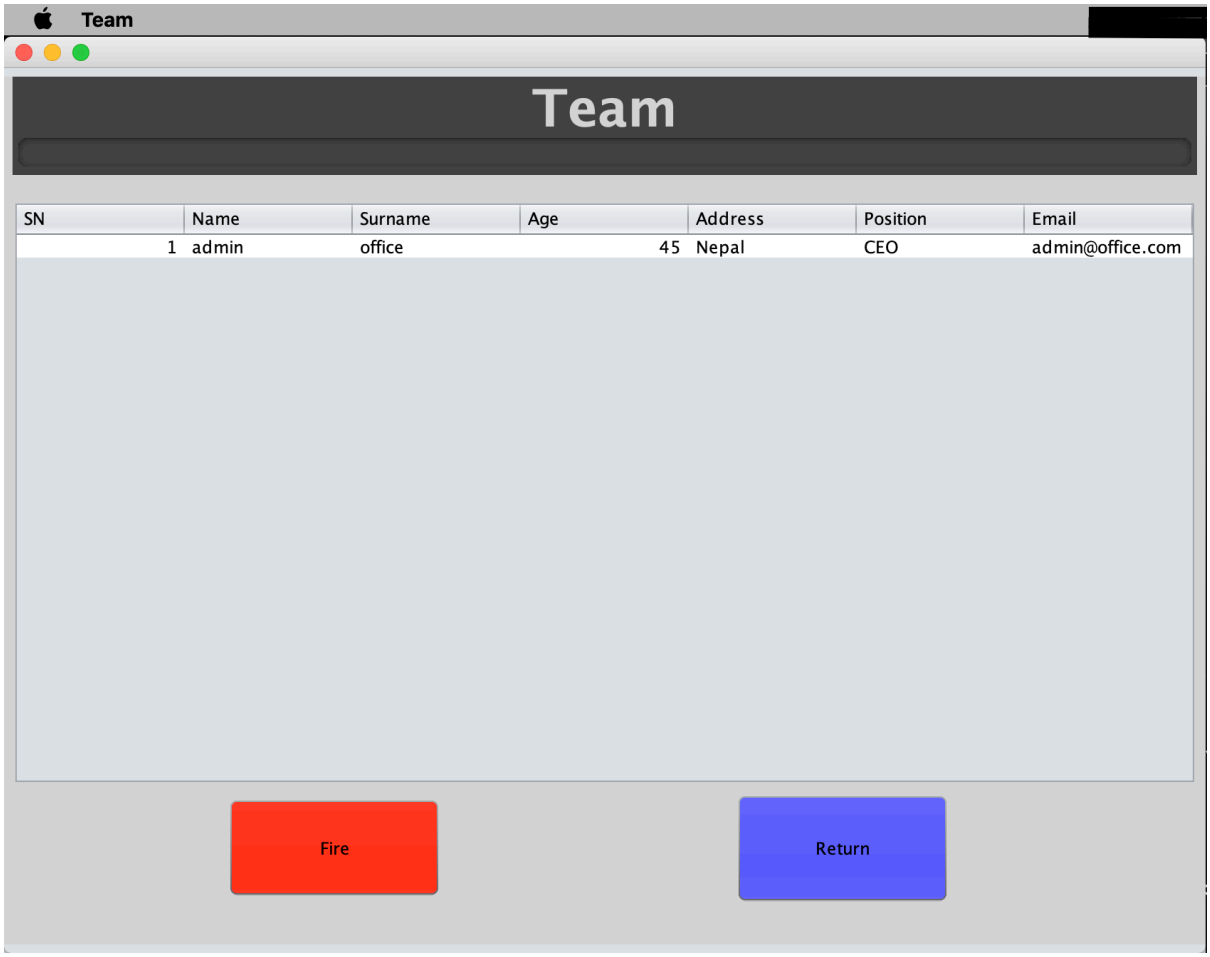


When the user starts to type in the search bar, the table below is dynamically filtered.

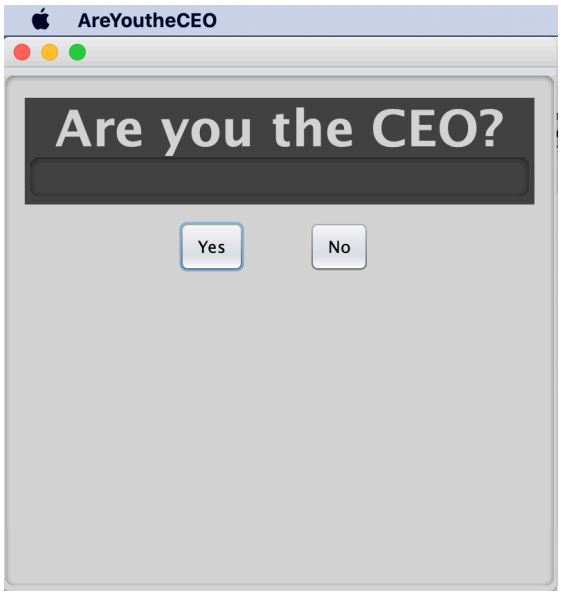
```
private void Txt_SearchKeyReleased(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:

    DefaultTableModel table = (DefaultTableModel)JTable_Clients.getModel();
    String search = Txt_Search.getText();
    TableRowSorter<DefaultTableModel> tr = new TableRowSorter<DefaultTableModel>(table);
    JTable_Clients.setRowSorter(tr);
    tr.setRowFilter(RowFilter.regexFilter(search));

}
```

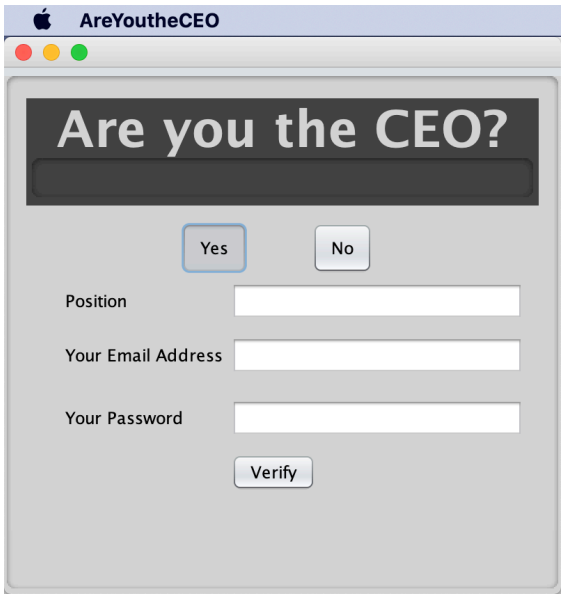
It utilizes a regex already in the library to remove invalid search results.

5. "Meet the team" allows the user to see other users.



The **'Fire'** button is special because only the admin (CEO) is allowed to fire staffs.

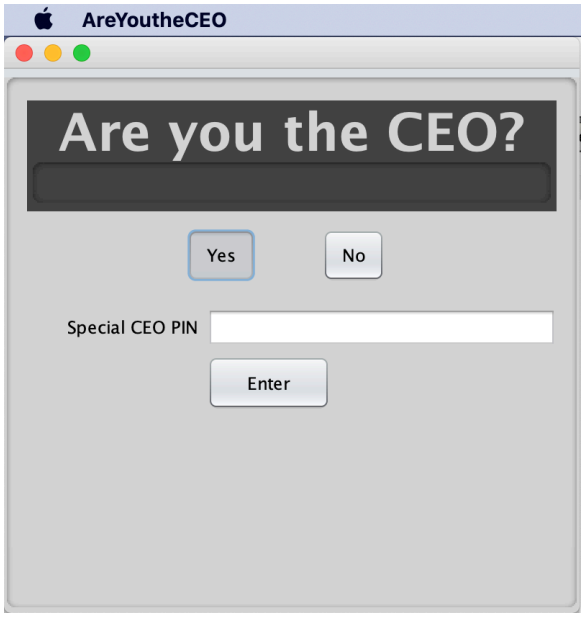




```
private void jYESActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    jLabel1.setVisible(true);
    jLabel2.setVisible(true);
    jLabel3.setVisible(true);
    jTextField1.setVisible(true);
    jTextField2.setVisible(true);
    jPasswordField1.setVisible(true);
    jVerify.setVisible(true);


}
```

.setVisible() is used to show or hide those fields.

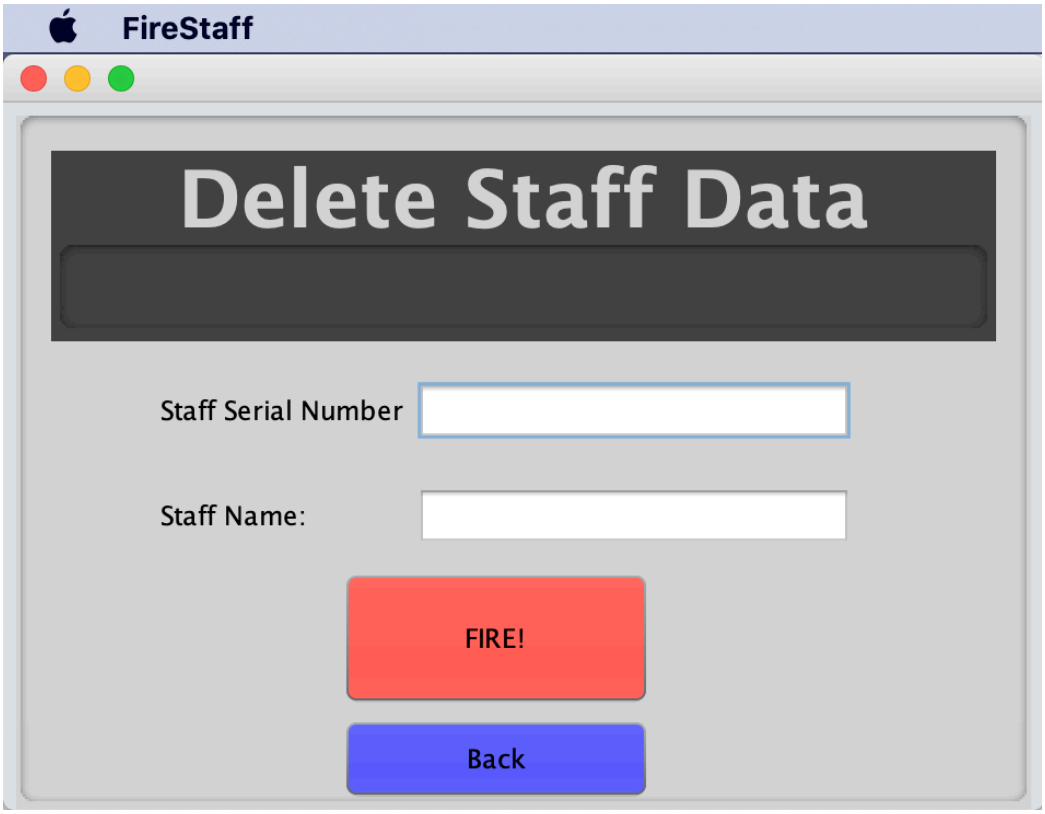
The **'Verify'** uses the same codes as the ones for the StaffLogin.java



The PIN is only known by the CEO. It retrieves the PIN from a table and matches. If true, then the final JFrame pops up.

```
rs = st.executeQuery("Select PIN from SecretPIN");


while (rs.next()){
    if (rs.getString(1).equals(a)){
        log = 0;
        break;
    }
}
if (log==0){


   FireStaff car = new FireStaff();
   car.setVisible(true);
   dispose();

}
```

After all that, the CEO gets access to the following frame, where staff data can be deleted.



Same codes as deleting client details apply here

Word Count: 993

Bibliography

1. Benett, S., 2013. *Android - Email Validation*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/18909746/android-email-validation/18909955

2. YouTube. 2015. *How To Bind Jtable From Mysql Database Using Arraylist In Java Netbeans [With Source Code]*. [online] Available at: https://www.youtube.com/watch?v=2d4i6BXQPFA

3. YouTube. 2017. *JAVA | Search In Jtable Without Jbutton [Source On Description]*. [online] Available at: https://www.youtube.com/watch?v=DJEXpgLyAtQ&t

4. YouTube. 2018. *Java Swing Application For Sending E-Mail - Studyviral*. [online] Available at: https://www.youtube.com/watch?v=TJXxImu0eVk