Criterion C: Development

Table of Contents

Description

EasyAccounts is a Java-based program for employees. It allows office employees and owners to modify profile information and add and remove events from a calendar. For office workers and office managers, it allows them to enter different accounts into appropriate tables which can be viewed in a balance sheet or profit and loss statement format. It allows owners to manage employees.

Techniques used:

Parsing a data stream

Data parsing is a process in which one string of data gets converted into a different type of data.

```
editProfilePictureButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        CurrentUser CU1 = new CurrentUser(UsernameTextField.getText());
        CU1.GetCUInfo(UsernameTextField.getText());

        JFileChooser file = new JFileChooser();
        file.setCurrentDirectory(new File(System.getProperty("user.home"), child: "Pictures")); //opens window with user's pictures
        FileNameExtensionFilter filter = new FileNameExtensionFilter( description: "*.Images",
                ...extensions: "jpg","gif","png"); //ensure filetype is appropriate
        file.addChoosableFileFilter(filter); //add filter
        int result = file.showSaveDialog( parent: null);
        if (result == JFileChooser.APPROVE_OPTION){
            File selectedFile = file.getSelectedFile();
            String fileName = selectedFile.getName(); //save name of file
            if (fileName.endsWith("jpg") || fileName.endsWith("png") || fileName.endsWith("PNG") || fileName.endsWith("JPG") ||
                    fileName.endsWith("GIF") || fileName.endsWith("gif")) {
                path = selectedFile.getAbsolutePath(); // save path of file
                ProfilePic.setIcon(ResizeImage(path)); // set profile pic to selected file
                CU1.SaveImage(path, UsernameTextField.getText()); //save image to database
            }
            else
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Please choose an appropriate filetype.",
                        title: "Invalid File Type", JOptionPane.WARNING_MESSAGE); //error message
        }
    }
});
```

Figure 1 - JFileChooser() being used to select a file by the user and get the absolute file path

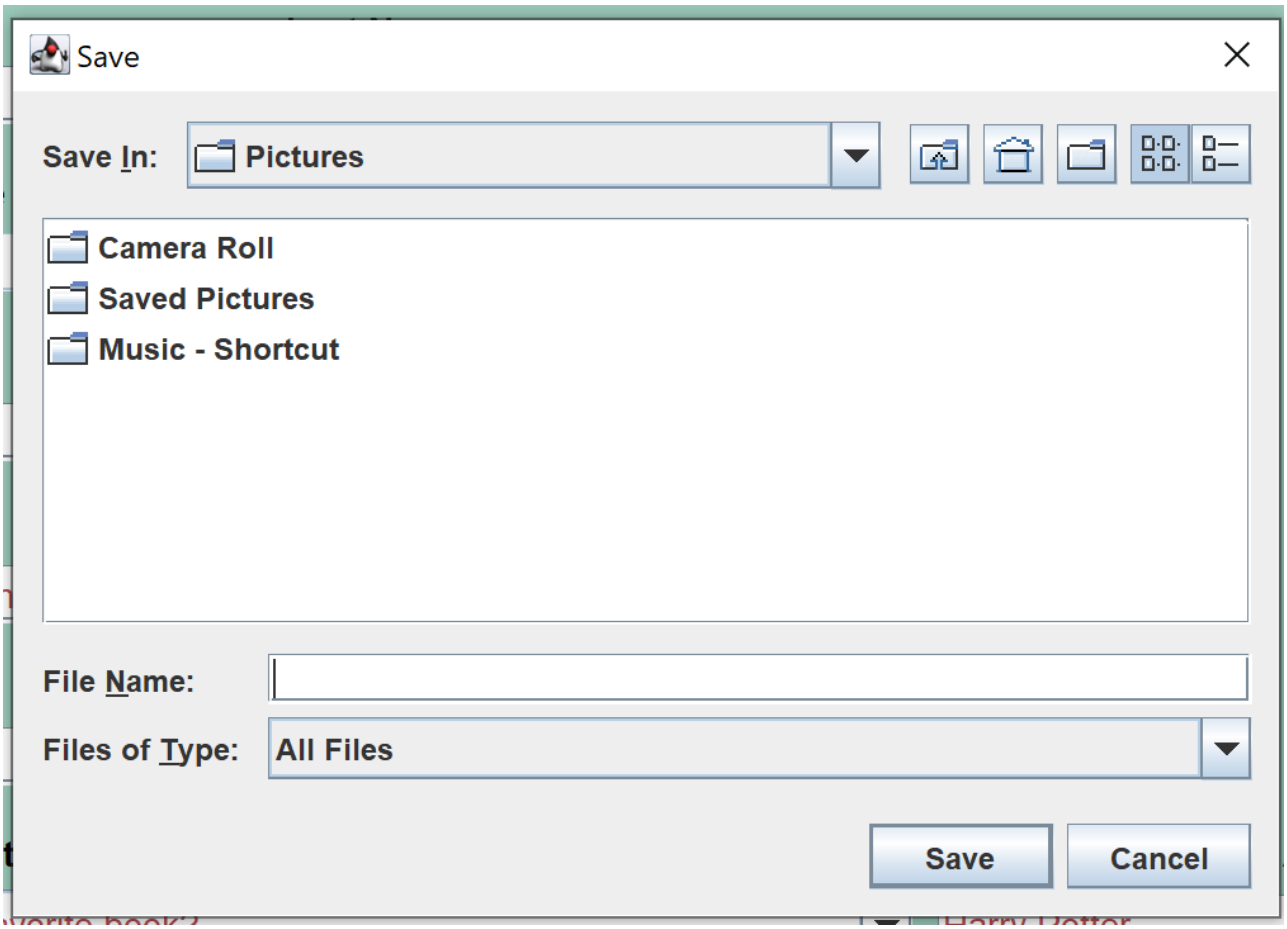

Figure 2 - JFileChooser

JFileChooser is used to get the absolute file path of the user's chosen picture. The program then sends it to the CurrentUser class to be saved into the database.

```
public void SaveImage(String path, String Username) {
    try {
        String host = "jdbc:mysql://Jay-PC:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); // create connection to database

        File file = new File(path); // create new file using path of selected image
        FileInputStream fis = new FileInputStream(file); // new file inputstream to add image to database
        PreparedStatement ps = con.prepareStatement( sql: "UPDATE easyaccounts_db.ea_users SET Profile_Pic = ? WHERE Username='" +
                "" + Username + "';");
        ps.setBinaryStream( parameterIndex: 1, fis, (int) file.length()); // convert to blob
        ps.executeUpdate(); // save image

        ps.close();
        fis.close();
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
} // set new profile pic for registered user
```

Figure 3 - The absolute file path being saved in the database using FileInputStream()

In the CurrentUser class, a method called SaveImage() uses the absolute file path string to create a file which is converted to a binary stream. That is saved in the database as a Binary Large Object (BLOB).

Adding data to an instance of the RandomAccessFile
Instances of RandomAccessFile support both reading and writing to a file.

```
    else {
        Statement stmt2 = con.createStatement();
        String SQL2 = "INSERT INTO easyaccounts_db.ea_users (Username,First_Name,Last_Name,Designation," +
                "Password,Gender,Date_Of_Birth,Email_Address,Security_Question_1,Security_Answer_1," +
                "Security_Question_2,Security_Answer_2) VALUES ('" + NUName + "','" + NFName + "','" +
                "" + NLName + "','" + NDesignation + "','" + NPassword + "','" + NGender + "','" + NDOB + "'," +
                "'" + NEmail + "','" + NSQ1 + "','" + NSQ1A + "','" + NSQ2 + "','" + NSQ2A + "');";
        stmt2.executeUpdate(SQL2);//Inserting new user into database

        File file = new File( pathname: "C:\\Users\\Temp\\IdeaProjects\\" +
                "EasyAccounts\\src\\com\\codebuilding\\Icons\\Account.png"); //getting default profile picture
        FileInputStream fis = new FileInputStream(file);
        PreparedStatement ps = con.prepareStatement( sql: "UPDATE easyaccounts_db.ea_users " +
                "SET Profile_Pic = ? WHERE Username='" + NUName + "';"); //saving default profile pic in database
        ps.setBinaryStream( parameterIndex: 1, fis, (int) file.length());
        ps.executeUpdate();

        ps.close();
        fis.close();
        con.close();
    }

} catch (SQLException err) {
    System.out.println(err.getMessage());
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
```

Figure 4 - Section of code adding a profile picture to the database using FileInputStream

FileInputStream() is used to enter files into the database. The program uses FileInputStream() to enter a default profile picture into the database which is used as the profile picture for new users.

```
    try {
        int blobLength = (int) ProfilePic.length(); //length of blob from database
        byte[] blobAsBytes = ProfilePic.getBytes( pos: 1, blobLength); //blob as byte array
        bufferedImage = ImageIO.read(new ByteArrayInputStream(blobAsBytes)); //saving blob as buffered image
    } catch (IOException e) {
        e.printStackTrace();
    }

} else{
    JOptionPane.showMessageDialog( parentComponent: null, message: CUName + " is not a user.",
            title: "Invalid Input", JOptionPane.WARNING_MESSAGE);
    Username = null;
}
```

Figure 5 - Portion of getting user information method which uses ImageIO

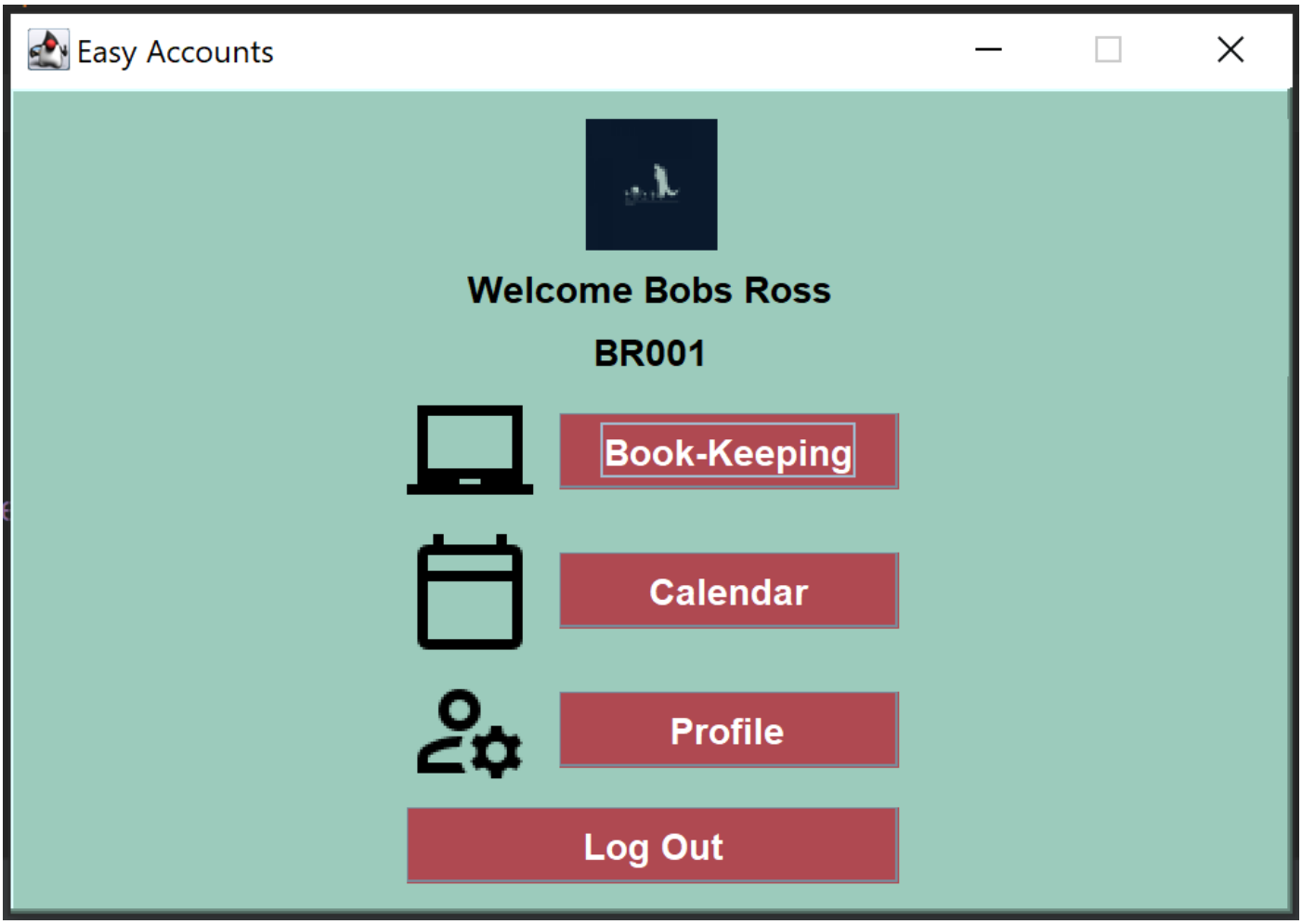

Figure 6 - GUI using buffered image saved as an Image Icon in a JLabel

ImageIO is used to read the converted BLOB and store it in a buffered image for the

profile picture feature. The buffered image is then converted to an ImageIcon so that it can be stored in a JLabel to be displayed for the user.

Deleting data from an instance of the RandomAccessFile

```
public TableModel RemoveEvent(int selectedRow, String username) {
    String EventName = GetEvents(username).getValueAt(selectedRow, columnIndex: 0).toString(); //getting event names
    String EventType = GetEvents(username).getValueAt(selectedRow, columnIndex: 1).toString(); //getting event types
    java.sql.Date EventDate = java.sql.Date.valueOf(GetEvents(username).getValueAt(selectedRow, columnIndex: 2)
            .toString()); //getting event types
    String EventDescription = GetEvents(username).getValueAt(selectedRow, columnIndex: 3).toString(); //getting event description

    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); // making connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "DELETE FROM userevents WHERE Username = '" + username + "'" +
                "AND EventName = '" + EventName + "' " + "AND EventDate = '" + EventDate + "' " +
                "AND EventDescription = '" + EventDescription + "' AND EventType = '" + EventType + "'";
        stmt1.executeUpdate(SQL1); //delete selected event from database

        Statement stmt2 = con.createStatement();
        String SQL2 = "SELECT * FROM userevents WHERE Username = '" + Username + "'";
        ResultSet rs1 = stmt2.executeQuery(SQL2); //updated events

        String[] columnNames = {"Event Name", "Event Type", "Event Date", "Event Description"};
        this.tableModel = new DefaultTableModel(columnNames, rowCount: 0); //set column names
```

Figure 7 - Method used to remove an event from the database (Part 1)

```
        while(rs1.next()){ //get event info
            String EventName1 = rs1.getString( columnLabel: "EventName");
            String EventType1 = rs1.getString( columnLabel: "EventType");
            String EventDate1 = rs1.getDate( columnLabel: "EventDate").toString();
            String EventDescription1 = rs1.getString( columnLabel: "EventDescription");

            String[] data = {EventName1,EventType1,EventDate1, EventDescription1};

            tableModel.addRow(data); //add each event to update
        }

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
    return tableModel;
}
```

Figure 8 - Method used to remove an event from the database (Part 2)

```
public TableModel RemoveEmployee(int selectedRow) {
    String Username = GetEmployees().getValueAt(selectedRow, columnIndex: 0).toString(); //getting employee username
    String FirstName = GetEmployees().getValueAt(selectedRow, columnIndex: 1).toString(); //getting employee first name
    String LastName = GetEmployees().getValueAt(selectedRow, columnIndex: 2).toString(); //getting employee last
    String Designation = GetEmployees().getValueAt(selectedRow, columnIndex: 3).toString(); //getting employee username

    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //making connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "DELETE FROM ea_users WHERE Username = '" + Username + "'AND First_Name = '" +
                "" + FirstName + "' AND Last_Name = '" + LastName + "' AND Designation = '" + Designation + "'";
        stmt1.executeUpdate(SQL1); //deleting employee

        GetEmployees(); // updating table

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
    return tableModel;
} //remove selected employees
```

Figure 9 - Method used to remove an employee from the database

The program gets the data in the selected row, then deletes it from the database, then refreshes the TableModel, then sets the updated model as the JTable's model.

Inserting data into an ordered sequential file

A sequential file is a file containing and storing data in chronological order.

```
public TableModel GetPNL (String Username){
    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); // make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "SELECT * FROM user_tables WHERE Username = '" + Username + "' " +
                "AND ElementType = 'Revenue' AND ElementTable = 'Profit and Loss Statement'";
        ResultSet rs1 = stmt1.executeQuery(SQL1); //get accounts labelled as revenue

        Statement stmt2 = con.createStatement();
        String SQL2 = "SELECT * FROM user_tables WHERE Username = '" + Username + "' " +
                "AND ElementType = 'Cost of Goods Sold' AND ElementTable = 'Profit and Loss Statement'";
        ResultSet rs2 = stmt2.executeQuery(SQL2); //get accounts labelled as COGS

        Statement stmt3 = con.createStatement();
        String SQL3 = "SELECT * FROM user_tables WHERE Username = '" + Username + "' " +
                "AND ElementType = 'Expenses' AND ElementTable = 'Profit and Loss Statement'";
        ResultSet rs3 = stmt3.executeQuery(SQL3); //get accounts labelled as Expenses

        String[] columnNames = {"Nominal Account", "Value"};
        this.tableModel = new DefaultTableModel(columnNames,  rowCount: 0); //add column names

        String[] Space = {null, null}; //space
```

Figure 10 - Creating Profit and Loss Statement by entering each row of data into a DefaultTableModel one after the other (Part 1)

```
String[] title = {"Revenue", null};
tableModel.addRow(title); //heading for revenue
tableModel.addRow(Space); //space

double Rtotal = 0; //revenue total

while(rs1.next()){ //get and add revenue accounts
    String RName = rs1.getString( columnLabel: "ElementName");
    String RValue = rs1.getString( columnLabel: "ElementValue");
    Rtotal = Rtotal + rs1.getDouble( columnLabel: "ElementValue");

    String[] data = {RName,RValue};

    tableModel.addRow(data);
}

tableModel.addRow(Space); //space
String[] title1 = {"Total Revenue", String.valueOf(Rtotal)}; // row with revenue total
tableModel.addRow(title1);
tableModel.addRow(Space); //space

String[] title2 = {"Cost of Goods Sold", null}; //COGS title
tableModel.addRow(title2);
tableModel.addRow(Space); //space

double COGStotal = 0; //COGS total
```

Figure 11 - Creating Profit and Loss Statement by entering each row of data into a DefaultTableModel one after the other (Part 2)

```
while(rs2.next()){ //get and add COGS accounts
    String COGSName = rs2.getString( columnLabel: "ElementName");
    String COGSValue = rs2.getString( columnLabel: "ElementValue");
    COGStotal = COGStotal + rs2.getDouble( columnLabel: "ElementValue");

    String[] data = {COGSName,COGSValue};

    tableModel.addRow(data);
}

tableModel.addRow(Space); //space
String[] title3 = {"Total Cost of Goods Sold", String.valueOf(COGStotal)}; //row with COGS total
tableModel.addRow(title3);
tableModel.addRow(Space); //space

String[] title4 = {"Gross Profit", String.valueOf(Rtotal-COGStotal)}; //row with gross profit
tableModel.addRow(title4);
tableModel.addRow(Space); //space

String[] title5 = {"Expenses", null}; //expenses title
tableModel.addRow(title5);
tableModel.addRow(Space); //space

double Exptotal = 0; //expenses total
```

Figure 12 - Creating Profit and Loss Statement by entering each row of data into a DefaultTableModel one after the other (Part 3)

```
        while(rs3.next()){ //get and add expense accounts
            String ExpName = rs3.getString( columnLabel: "ElementName");
            String ExpValue = rs3.getString( columnLabel: "ElementValue");
            Exptotal = Exptotal + rs3.getDouble( columnLabel: "ElementValue");

            String[] data = {ExpName,ExpValue};

            tableModel.addRow(data);
        }

        tableModel.addRow(Space);//space
        String[] title6 = {"Total Expenses", String.valueOf(Exptotal)}; //row with expenses total
        tableModel.addRow(title6);
        tableModel.addRow(Space);//space

        String[] title7 = {"Net Profit", String.valueOf((Rtotal-COGStotal)-Exptotal)}; //net profit row
        tableModel.addRow(title7);

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
    return tableModel;
} // format of profit and loss statement and totals
```

Figure 13 - Creating Profit and Loss Statement by entering each row of data into a DefaultTableModel one after the other (Part 4)

Each row of the balance sheet and profit and loss format is retrieved or calculated and added in a sequential manner to the TableModel using a while loop. It stores each account value from the database in a new row in the JTable.

Deleting data from a sequential file

```
public TableModel RemoveEmployee(int selectedRow) {
    String Username = GetEmployees().getValueAt(selectedRow, columnIndex: 0).toString(); //getting employee username
    String FirstName = GetEmployees().getValueAt(selectedRow, columnIndex: 1).toString(); //getting employee first name
    String LastName = GetEmployees().getValueAt(selectedRow, columnIndex: 2).toString(); //getting employee last
    String Designation = GetEmployees().getValueAt(selectedRow, columnIndex: 3).toString(); //getting employee username

    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //making connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "DELETE FROM ea_users WHERE Username = '" + Username + "'AND First_Name = '" +
                "" + FirstName + "' AND Last_Name = '" + LastName + "' AND Designation = '" + Designation + "'";
        stmt1.executeUpdate(SQL1); //deleting employee

        GetEmployees(); // updating table

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
    return tableModel;
} //remove selected employees
```

Figure 14 - Method getting selected row from JTable and the deleting it from the database and reuploading the updated TableModel

A query in MySQL is created using the data from the selected row and then deletes it from the database. The same process of creating the balance sheet format is then used to update the TableModel.

Searching

This refers to looking for specific data in a large data set and terminating the search once the data has been found.

```
public void GetCUInfo(String CUName) {
    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "SELECT * FROM ea_users WHERE Username = '" + CUName + "'";
        ResultSet rs1 = stmt1.executeQuery(SQL1); //get info for specific user

        if (rs1.next()) { //get user info
            Username = rs1.getString( columnLabel: "Username");
            FName = rs1.getString( columnLabel: "First_Name");
            LName = rs1.getString( columnLabel: "Last_Name");
            Designation = rs1.getString( columnLabel: "Designation");
            Password = rs1.getString( columnLabel: "Password");
            Gender = rs1.getString( columnLabel: "Gender");
            Email = rs1.getString( columnLabel: "Email_Address");
            SQ1 = rs1.getString( columnLabel: "Security_Question_1");
            SQ1A = rs1.getString( columnLabel: "Security_Answer_1");
            SQ2 = rs1.getString( columnLabel: "Security_Question_2");
            SQ2A = rs1.getString( columnLabel: "Security_Answer_2");
            DOB = rs1.getDate( columnLabel: "Date_Of_Birth");
            ProfilePic = rs1.getBlob( columnLabel: "Profile_Pic");
```

Figure 15 - Method used to search for a specific user when logging into the program (Part 1)

```
            try { // get profile pic from database
                int blobLength = (int) ProfilePic.length(); //length of blob from database
                byte[] blobAsBytes = ProfilePic.getBytes( pos: 1, blobLength); //blob as byte array
                bufferedImage = ImageIO.read(new ByteArrayInputStream(blobAsBytes)); //saving blob as buffered image
            } catch (IOException e) {
                e.printStackTrace();
            }

        } else{
            JOptionPane.showMessageDialog( parentComponent: null, message: CUName + " is not a user.",
                    title: "Invalid Input", JOptionPane.WARNING_MESSAGE);
            Username = null;
        }

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
}
```

Figure 16 - Method used to search for a specific user when logging into the program (Part 2)

```
public void SetNUInfo(String NUName, String NFName, String NLName, String NPassword, String NGender, String NEmail,
                      String NSQ1, String NSQ1A, String NSQ2, String NSQ2A, String NDesignation, Date NDOB) {
    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "SELECT * FROM ea_users WHERE Username = '" + NUName + "'";
        ResultSet rs1 = stmt1.executeQuery(SQL1);  //check if user exists

        if (rs1.next()) { //error if user exists
            JOptionPane.showMessageDialog( parentComponent: null,  message: NUName + " has already been taken. Please choose another username.");
        }
        else {
```

Figure 17 - Method used to determine whether the entered username already exists when creating an account

```
1 •   SELECT * FROM easyaccounts_db.ea_users;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Username	First_Name	Last_Name	Designation	Password	Gender	Date_Of_Birth	Email_Address	Security_Question_1	Security_Answer_1	Security_Qu
AL001	Anis	Lewis	Owner	Al001#1993	Male	1993-04-28	AL001@gmail.com	What Is your favorite book?	Harry Potter	What is the r
BR001	Bobs	Ross	Office Worker	Br001#1...	Male	1993-12-04	BR@gmail.com	What Is your favorite book?	HP	What is the r
JA001	Janes	Andersons	Office Worker	Ja001#1...	Female	1993-07-27	JA001@gmail.com	What Is your favorite book?	Harry Potter	What is the r
RR001	Robs	Ross	Office Worker	Rr001#1...	Male	1993-12-04	RR@gmail.com	What Is your favorite book?	HP	What is the r
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 18 - MySQL database showing ea_users table

Linear search is used to determine if a username already exists when registering. When logging in, linear search is used to determine if the entered username is a registered user. The program uses a while loop until the username is found or until the result set has finished.

Structured Query Language (SQL)
SQL is used to communicate with a database through connections.

```
public void SetNewPassword(String CUsername, String NPassword) {
    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "UPDATE easyaccounts_db.ea_users SET Password='" + NPassword + "' " +
                "WHERE Username='" + CUsername + "';"; //set new password
        stmt1.executeUpdate(SQL1);

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
} // set new password for registered user
```

Figure 19 - Method using SQL syntax

MySQL is used to store and retrieve user data, and to manipulate the database. Four SQL commands were used: “INSERT” to add data, “SELECT” to look for specific data, “UPDATE” to make changes to data, and “REMOVE” to remove data.

Implementing a hierarchical composite data structure

```
        while(rs1.next()){ //get event info
            String EventName1 = rs1.getString( columnLabel: "EventName");
            String EventType1 = rs1.getString( columnLabel: "EventType");
            String EventDate1 = rs1.getDate( columnLabel: "EventDate").toString();
            String EventDescription1 = rs1.getString( columnLabel: "EventDescription");

            String[] data = {EventName1,EventType1,EventDate1, EventDescription1};

            tableModel.addRow(data); //add each event to update
        }

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
    return tableModel;
}
```

Figure 20 - Using arrays to fill in the different rows of a DefaultTableModel for events

```
public TableModel GetEmployees(){
    try {
        String host = "jdbc:mysql://Jay-PC:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); // make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "SELECT * FROM ea_users";
        ResultSet rs1 = stmt1.executeQuery(SQL1); // get employees

        String[] columnNames = {"Username", "First Name", "Last Name", "Designation"};
        this.tableModel = new DefaultTableModel(columnNames,  rowCount: 0); // set column names

        while(rs1.next()){ //get employee info
            if (!rs1.getString( columnLabel: "Designation").equals("Owner")) {
                String Username = rs1.getString( columnLabel: "Username");
                String FirstName = rs1.getString( columnLabel: "First_Name");
                String LastName = rs1.getString( columnLabel: "Last_Name");
                String Designation = rs1.getString( columnLabel: "Designation");

                String[] data = {Username, FirstName, LastName, Designation};

                tableModel.addRow(data); // add each employee info
            }
        }
        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
```

Figure 21 - Using arrays to fill in the different rows of a DefaultTableModel for employees

A while loop is used to retrieve and store each row from the database in an array which is set as a new row in the DefaultTableModel. This TableModel is then set as the model of the JTables.

```
private int checkPasswordStrength(String password) {
    int strengthPercentage=0;
    String[] partialRegexChecks = { ".*[a-z]+.*", // lowercase
            ".*[A-Z]+.*", // uppercase
            ".*[\\d]+.*", // digits
            ".*[@#$%]+.*" // symbols
    };


    if (password.matches(partialRegexChecks[0])) { //has lowercase
        strengthPercentage+=25;
    }
    if (password.matches(partialRegexChecks[1])) { //has uppercase
        strengthPercentage+=25;
    }
    if (password.matches(partialRegexChecks[2])) { //has digits
        strengthPercentage+=25;
    }
    if (password.matches(partialRegexChecks[3])) { //has symbols
        strengthPercentage+=24;
    }
    if (password.length() >= 7) { //has or is longer than 7 characters
        strengthPercentage+=1;
    }

    return strengthPercentage;
}
```

Figure 22 - Using arrays to check the strength of an entered password

The checkPasswordStrength() method contains an initialized String array with the different criterion of the password. It checks the password against each criterion to ensure that the password is strong.

Inheritance

Inheritance allows for child classes to inherit the methods of the parent class.

```
public class EasyAccountsMain {
    public static JFrame EasyAccountsMainFrame; //JFrame to be used by the whole program

    public static void main(String[] args){
        LoginPage LPObject = new LoginPage(); // inheriting login page

        EasyAccountsMainFrame = new JFrame( title: "Easy Accounts");
        EasyAccountsMainFrame.setContentPane(LPObject.LoginPanel); //setting panel in frame
        EasyAccountsMainFrame.setVisible(true);
        EasyAccountsMainFrame.pack();
        EasyAccountsMainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //end program when the x button is clicked
        EasyAccountsMainFrame.setSize( width: 700, height: 500); //setting size of frame
        EasyAccountsMainFrame.setResizable(false);
        EasyAccountsMainFrame.setLocationRelativeTo(null); //setting location to the middle of the screen
    }
}
```

Figure 23 - JFrame inheriting JPanel and Main class inheriting Login Page

The program allows the main class to inherit the different classes so that a new instance of the main class does not need to be created every time the JPanel needs to be changed.

```
public class OfficeWorkerHomePage extends LoginPage {
    public JPanel OfficeWorkerHomePanel;
    public JLabel WelcomeFullNameLabel;
    public JLabel WelcomeUsernameLabel;
    private JButton BookKeepingButton;
    private JButton CalendarButton;
    private JButton ProfileButton;
    public JLabel ProfilePicLabel;
    private JButton logOutButton;
```

Figure 24 - OfficeWorkerHomePage class extending LoginPage class

The program also allows classes to extend other classes which allows for the methods from one class to be used in another class. This reduces the need to copy and paste the same code over and over again.

```
editProfilePictureButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        CurrentUser CU1 = new CurrentUser(UsernameTextField.getText());
        CU1.GetCUInfo(UsernameTextField.getText());

        JFileChooser file = new JFileChooser();
        file.setCurrentDirectory(new File(System.getProperty("user.home"), child: "Pictures")); //opens window with user's pictures
        FileNameExtensionFilter filter = new FileNameExtensionFilter( description: "*.Images",
                ...extensions: "jpg","gif","png"); //ensure filetype is appropriate
        file.addChoosableFileFilter(filter); //add filter
        int result = file.showSaveDialog( parent: null);
        if (result == JFileChooser.APPROVE_OPTION){
            File selectedFile = file.getSelectedFile();
            String fileName = selectedFile.getName(); //save name of file
            if (fileName.endsWith("jpg") || fileName.endsWith("png") || fileName.endsWith("PNG") || fileName.endsWith("JPG") ||
                    fileName.endsWith("GIF") || fileName.endsWith("gif")) {
                path = selectedFile.getAbsolutePath(); // save path of file
                ProfilePic.setIcon(ResizeImage(path)); // set profile pic to selected file
                CU1.SaveImage(path, UsernameTextField.getText()); //save image to database
            }
            else
                JOptionPane.showMessageDialog( parentComponent: null,  message: "Please choose an appropriate filetype.",
                        title: "Invalid File Type", JOptionPane.WARNING_MESSAGE); //error message
        }
    }
});
```

Figure 25 - Edit profile button in which System.getProperty inherits child component

JFileChooser usually opens a window to user.home, but since it inherits a child component called Pictures, the window opens the user's pictures folder. This results in less work for the user.

Encapsulation

Encapsulation refers to the bundling of data and methods that work on the data within one area, such as a class.

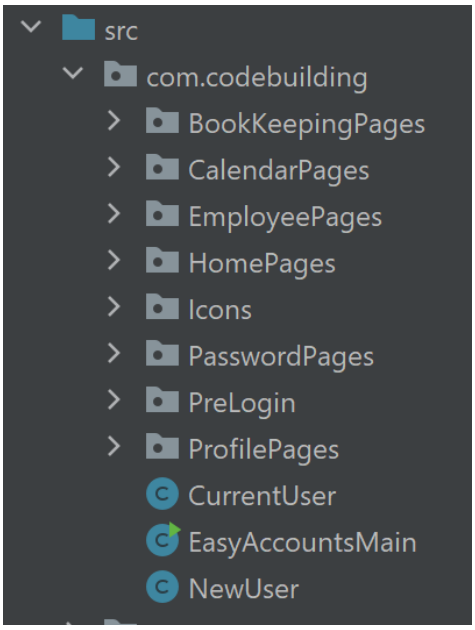

Figure 26 - Packages used to separate the different classes.

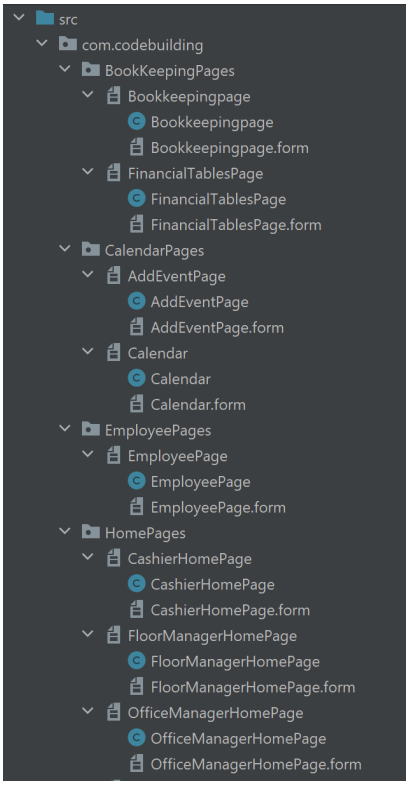

Figure 27 - Packages expanded (Part 1)

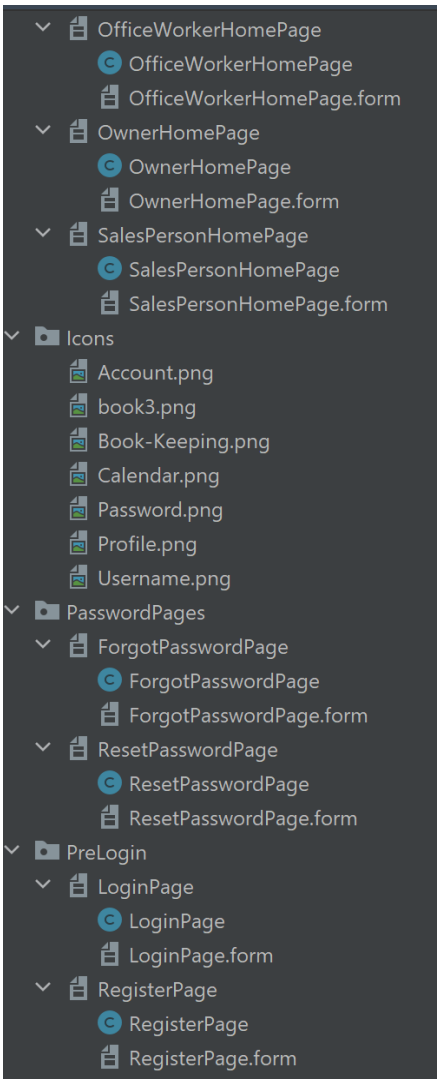

Figure 28 - Packages expanded (Part 2)

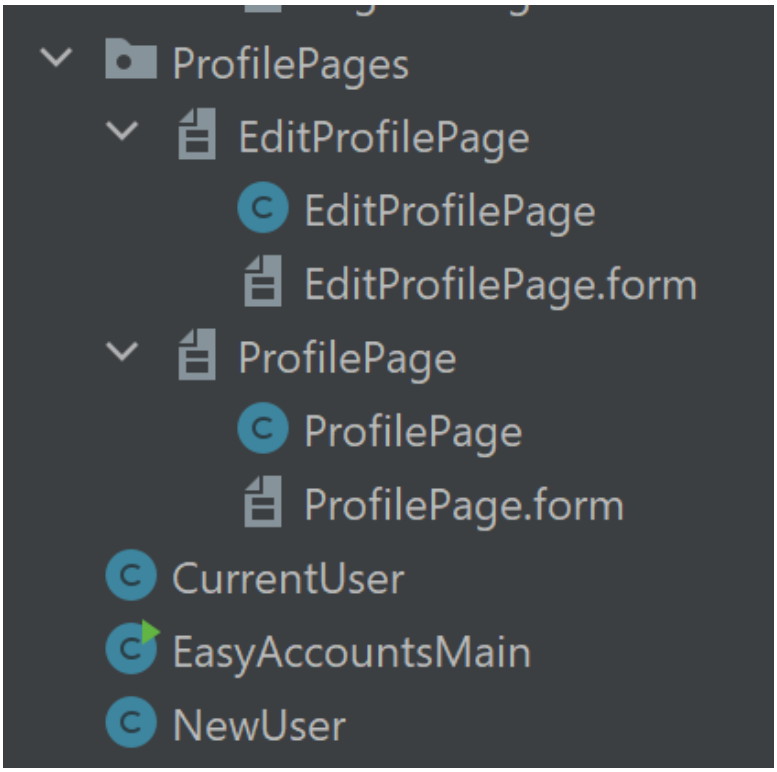

Figure 29 - Packages expanded (Part 3)

The different files are arranged appropriately into packages for less confusion. When another person is making changes to the program it will be easier for them to locate all of the classes. For example, the classes could be the login class or the homepage class.

```
public class CurrentUser {
    String Username, FName, LName, Password, Gender, Email, SQ1, SQ1A, SQ2, SQ2A, Designation;
    Date DOB;
    public Blob ProfilePic;
    BufferedImage bufferedImage;
    private DefaultTableModel tableModel;

    public CurrentUser(String CUName) { this.Username = CUName; } // set CU

    public void GetCUInfo(String CUName) {...} // get user info

    public String GetUsername() { return Username; } //return username

    public String GetFName() { return FName; } // return user first name

    public String GetLName() { return LName; } // return user last name

    public String GetDesignation() { return Designation; } // return user designation or role

    public String GetPassword() { return Password; } // return user password

    public String GetGender() { return Gender; } // return user gender

    public String GetEmail() { return Email; } // return user email

    public String GetSQ1() { return SQ1; } // return user security question 1
```

Figure 30 - CurrentUser class with all of the methods to be used (Part 1)

```
public String GetSQ1A() { return SQ1A; } // return user security question answer 1

public String GetSQ2() { return SQ2; } // return user security question 2

public String GetSQ2A() { return SQ2A; } // return user security question answer 2

public Date GetDOB() { return DOB; } // return user date of birth

public void SetNewPassword(String CUsername, String NPassword) {...} // set new password for registered user

public void SaveImage(String path, String Username) {...} // set new profile pic for registered user

public BufferedImage GetProfilePic() { return bufferedImage; } // return user profile pic

public void SaveUpdate(String FName, String LName, String OUsername, String Gender, String DOB,
                       String Email, String Password, String SQ1,
                       String SQ1A, String SQ2, String SQ2A) {...} // save changes made to user profile

public void AddEvent(String Username, String EventName, String EventDate, String EventDescription,
                     String EventType) {...} // add event to calendar

public TableModel RemoveEvent(int selectedRow, String username) {...} // remove event from calendar

public TableModel GetEvents(String Username) {...} // return user events

public TableModel GetTable (String Username, String type, String Table){...} // return accounts in each financial table

public void UpdateTable (String Username, TableModel currentTable, String type, String Table){...} //save changes made to tables
```

Figure 31 - CurrentUser class with all of the methods to be used (Part 2)

```
    public TableModel GetBS (String Username){...} // format of balance sheet and totals

    public TableModel GetPNL (String Username){...} // format of profit and loss statement and totals

    public TableModel GetEmployees(){...} // return employees in business

    public TableModel RemoveEmployee(int selectedRow) {...} //remove selected employees
}
```

Figure 32 - CurrentUser class with all of the methods to be used (Part 3)

The CurrentUser class houses all the methods which are used throughout the program. It allows for code to be able to be reused without needing to be rewritten.

```
public class NewUser {
    String Username;

    public NewUser(String NUName) { this.Username = NUName; } // set new user name

    public void SetNUInfo(String NUName, String NFName, String NLName, String NPassword, String NGender, String NEmail,
                          String NSQ1, String NSQ1A, String NSQ2, String NSQ2A, String NDesignation, Date NDOB) {...} // add new user

    public void SetTables(String Username){...} // set default tables for new user
}
```

Figure 33 - NewUser class with all of the methods to be used

The NewUser class houses the methods to be used during the registration process of the program. It allows for the reuse of code without needing to rewrite it.

Additional Libraries

Libraries are collections of classes which have already been created by someone else. These collections of classes can be imported into a program in order to avoid writing the same code over again.

```
import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import java.awt.image.BufferedImage;
import java.io.*;
import java.sql.*;
```

Figure 34 - Imported Packages (Part 1)

```
import com.codebuilding.CurrentUser;
import com.codebuilding.EasyAccountsMain;
import com.codebuilding.PreLogin.LoginPage;

import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
```

Figure 35 - Imported Packages (Part 2)

The javax.imageio.ImageIO is used to save profile pictures.

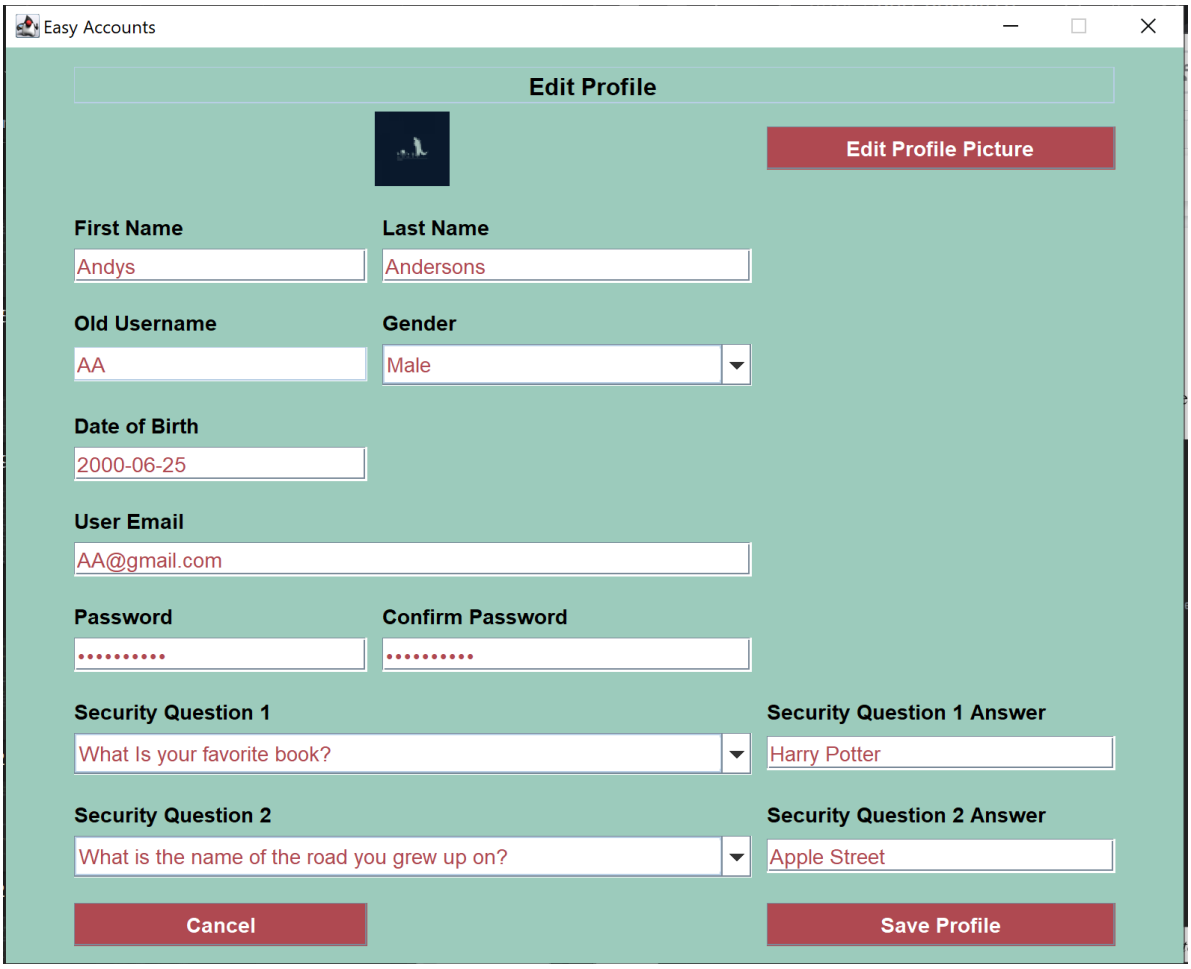

Figure 28 - Edit profile page

The javax.swing.* package is used for the creation of a JFrame, JPanels, DefaultTableModels, and the components on the JPanels for the Graphical User Interface.
The java.sql.* package is used for connections to the MySQL database, inserting data, removing data, updating data, searching for data, and for the conversion from string to sql appropriate formats in the program itself.
The java.awt.* and java.io.* packages were used to retrieve files from the users computer and to use action and focus listeners for JButtons and Jtables.

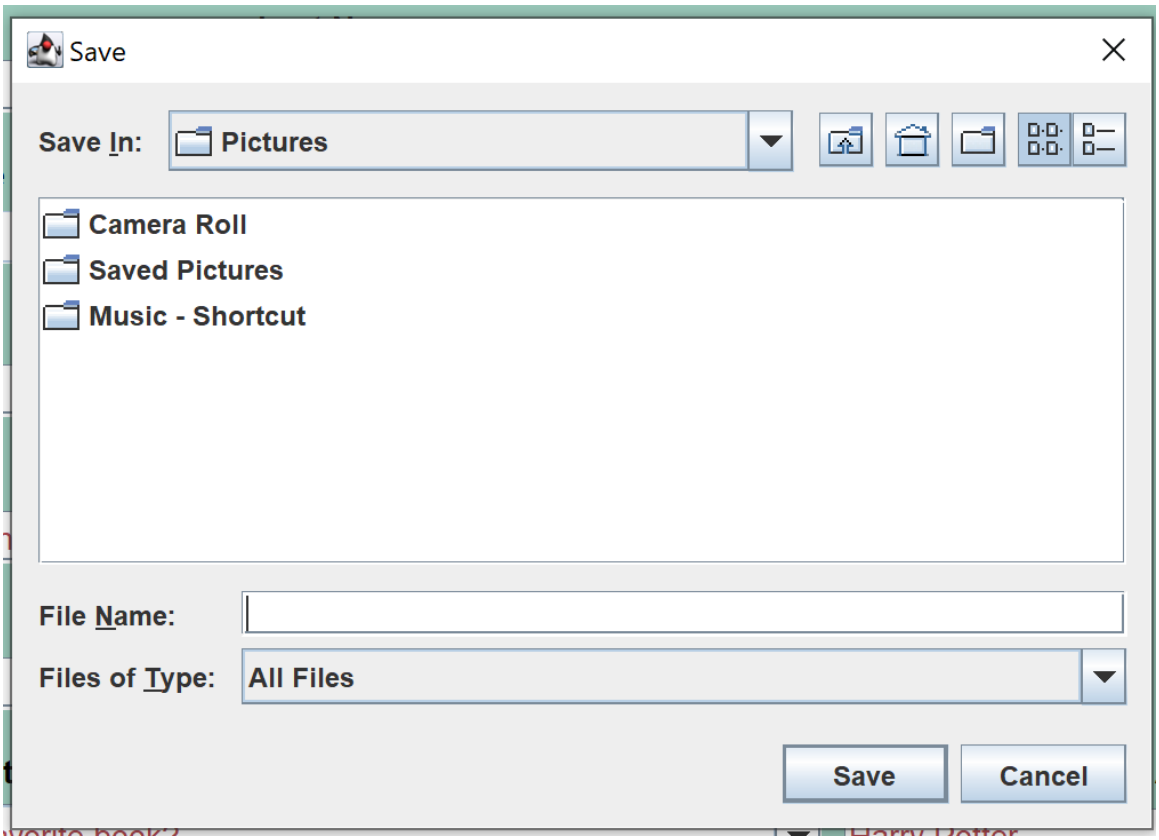

Figure 29 – JfileChooser

The javax.swing.JFileChooser and javax.swing.JFileChooser.FileNameExtensionFilter packages were used to retrieve and store file names along with the type of file to ensure it was a certain format.

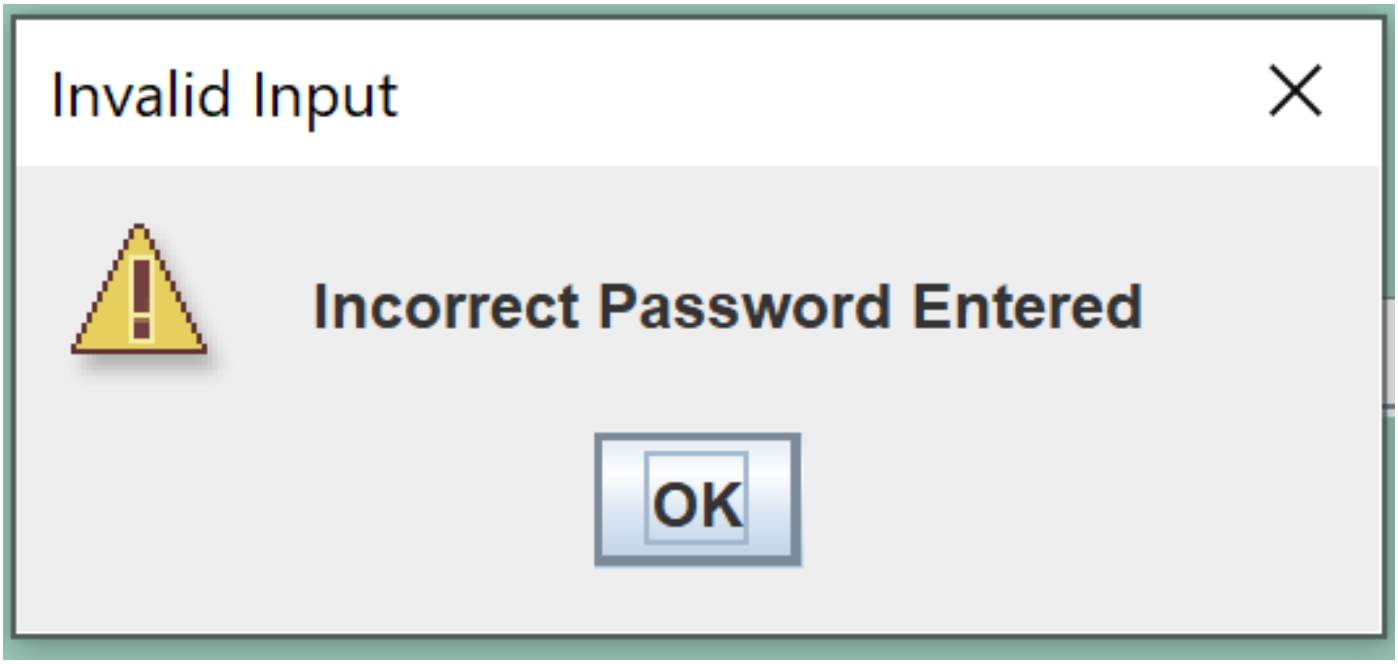

Figure 30 - Warning Message

The java.swing.JOptionPane package was used to create dialog boxes to present the user with messages such as error messages or informational messages.

Java Swing Components (GUI) and Action, Focus, and Key Listeners

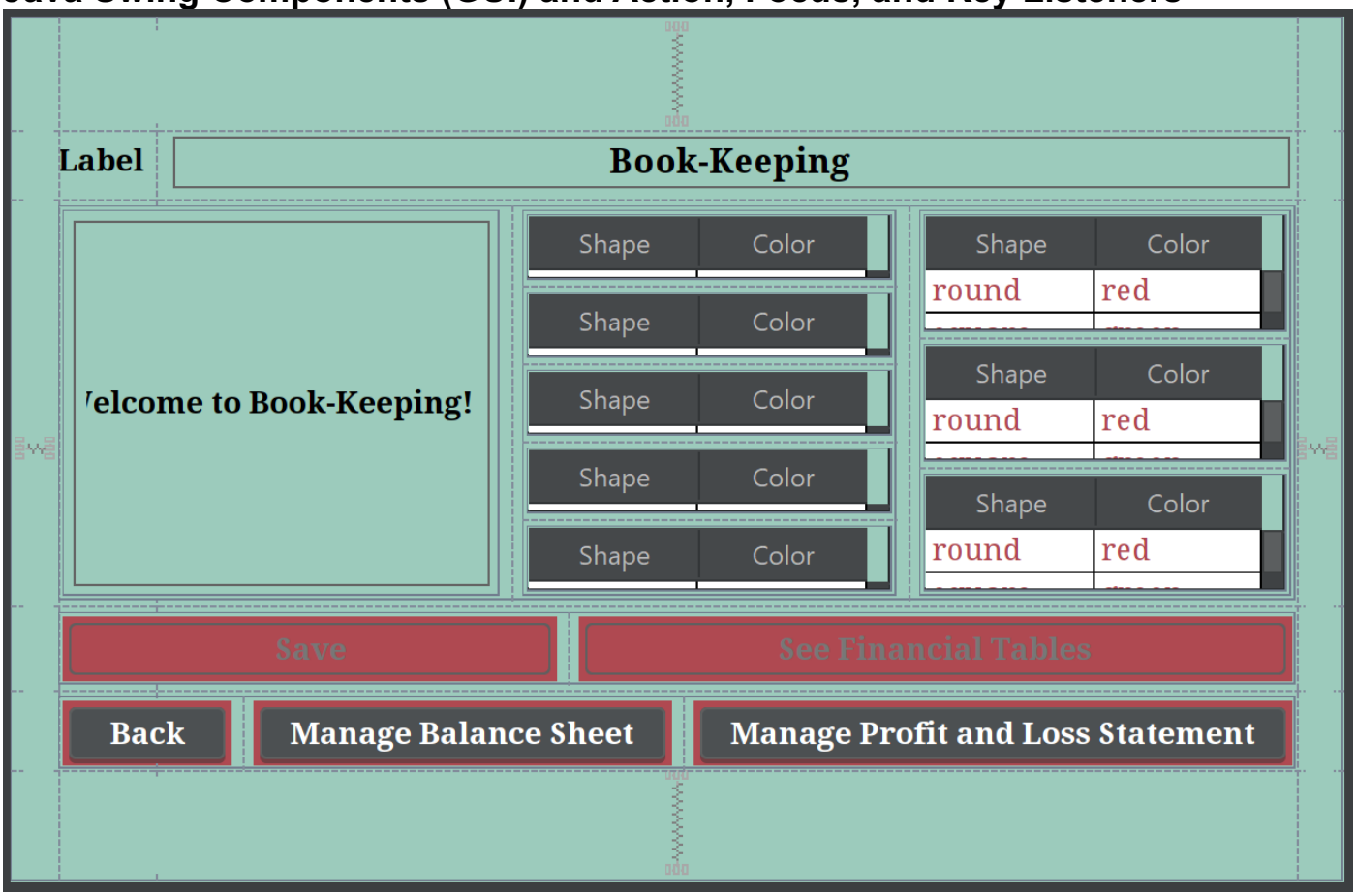

Figure 37 - Book-Keeping page GUI

```
public class EasyAccountsMain {
    public static JFrame EasyAccountsMainFrame; //JFrame to be used by the whole program

    public static void main(String[] args){
        LoginPage LPObject = new LoginPage(); // inheriting login page

        EasyAccountsMainFrame = new JFrame( title: "Easy Accounts");
        EasyAccountsMainFrame.setContentPane(LPObject.LoginPanel); //setting panel in frame
        EasyAccountsMainFrame.setVisible(true);
        EasyAccountsMainFrame.pack();
        EasyAccountsMainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //end program when the x button is clicked
        EasyAccountsMainFrame.setSize( width: 700, height: 500); //setting size of frame
        EasyAccountsMainFrame.setResizable(false);
        EasyAccountsMainFrame.setLocationRelativeTo(null); //setting location to the middle of the screen
    }
}
```

Figure 38 - Centering, Closing, Resizing JFrame

JFrame and JPanels

One main JFrame was created. For each page, the visibility of the JPanels were changed allowing different pages to be opened without needing to create a new

JFrame each time.

Centering JFrame
This technique of centering the JFrame was used throughout the program.

Closing JFrame
This technique was used to close a JFrame when a button was clicked.

Resizing JFrame
This technique of resizing the JFrame was used throughout the program.

```
BSCLTable.addFocusListener((FocusAdapter) focusGained(e) → { //checks for focus on JTable
        super.focusGained(e);
        BSCLTable.addKeyListener((KeyAdapter) keyPressed(e) → { // checks which key is clicked
                if (e.getKeyCode() == KeyEvent.VK_ENTER) //adds row if enter is clicked
                {
                    DefaultTableModel dtm = (DefaultTableModel) BSCLTable.getModel();
                    Vector Rowdata = null;
                    dtm.addRow(Rowdata);
                }

                if (e.getKeyCode() == KeyEvent.VK_BACK_SPACE) //removes row if backspace/delete is clicked.
                {
                    DefaultTableModel dtm = (DefaultTableModel) BSCLTable.getModel();
                    dtm.removeRow(BSCLTable.getSelectedRow());
                }
        });
});
```

Figure 39 - Focus Listener and Key Listener working together to add or remove rows

Adding and removing rows from JTable
Focus and Key listeners were used to add and remove rows from JTables.

```
manageProfitAndLossButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        CurrentUser CU1 = new CurrentUser(UsernameLabel.getText());
        CU1.GetCUInfo(UsernameLabel.getText()); //create instance of current user

        EasyAccountsMain.EasyAccountsMainFrame.setSize( width: 700, height: 650); //resize JFrame
        EasyAccountsMain.EasyAccountsMainFrame.setLocationRelativeTo(null); //set location of JFrame to center

        saveButton.setEnabled(true); //enable button
        financialTablesButton.setEnabled(true); //enable button
        WPanel.setVisible(false); //set panel invisible
        BSPanel.setVisible(false); //set panel invisible
        PNLPanel.setVisible(true); //set panel visible

        PNLRTable.setModel(CU1.GetTable(UsernameLabel.getText(),  type: "Revenue",
                Table: "Profit and Loss Statement")); //put data from database in table
        PNLCOGSTable.setModel(CU1.GetTable(UsernameLabel.getText(),  type: "Cost of Goods Sold",
                Table: "Profit and Loss Statement")); //put data from database in table
        PNLExpTable.setModel(CU1.GetTable(UsernameLabel.getText(),  type: "Expenses",
                Table: "Profit and Loss Statement")); //put data from database in table
    }
});
```

Figure 40 -Action listener used to activate JPanels and Jbuttons

Action Listeners
This was used for JButton clicks throughout the program.

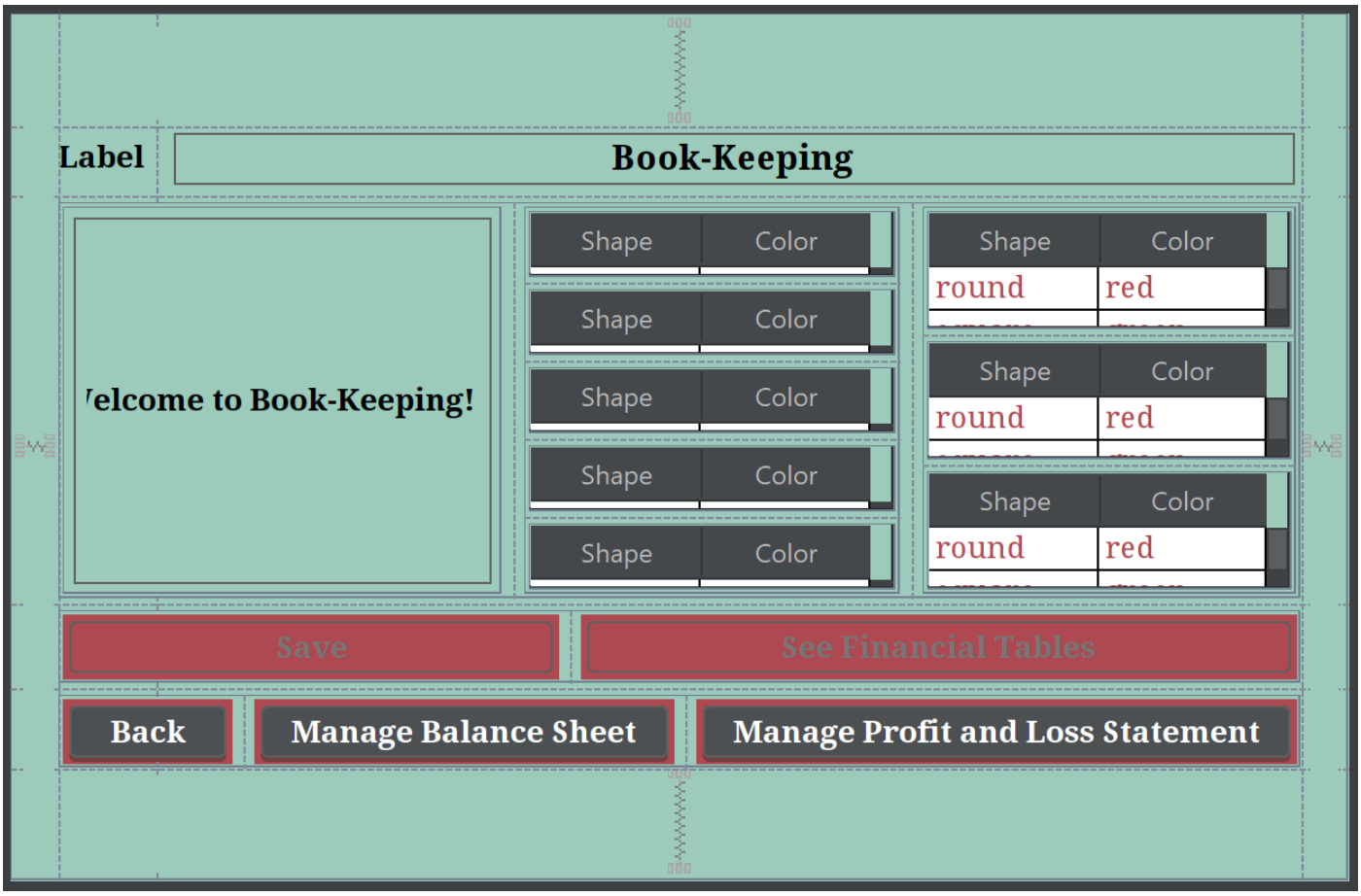

Figure 41 - Book-Keeping page GUI

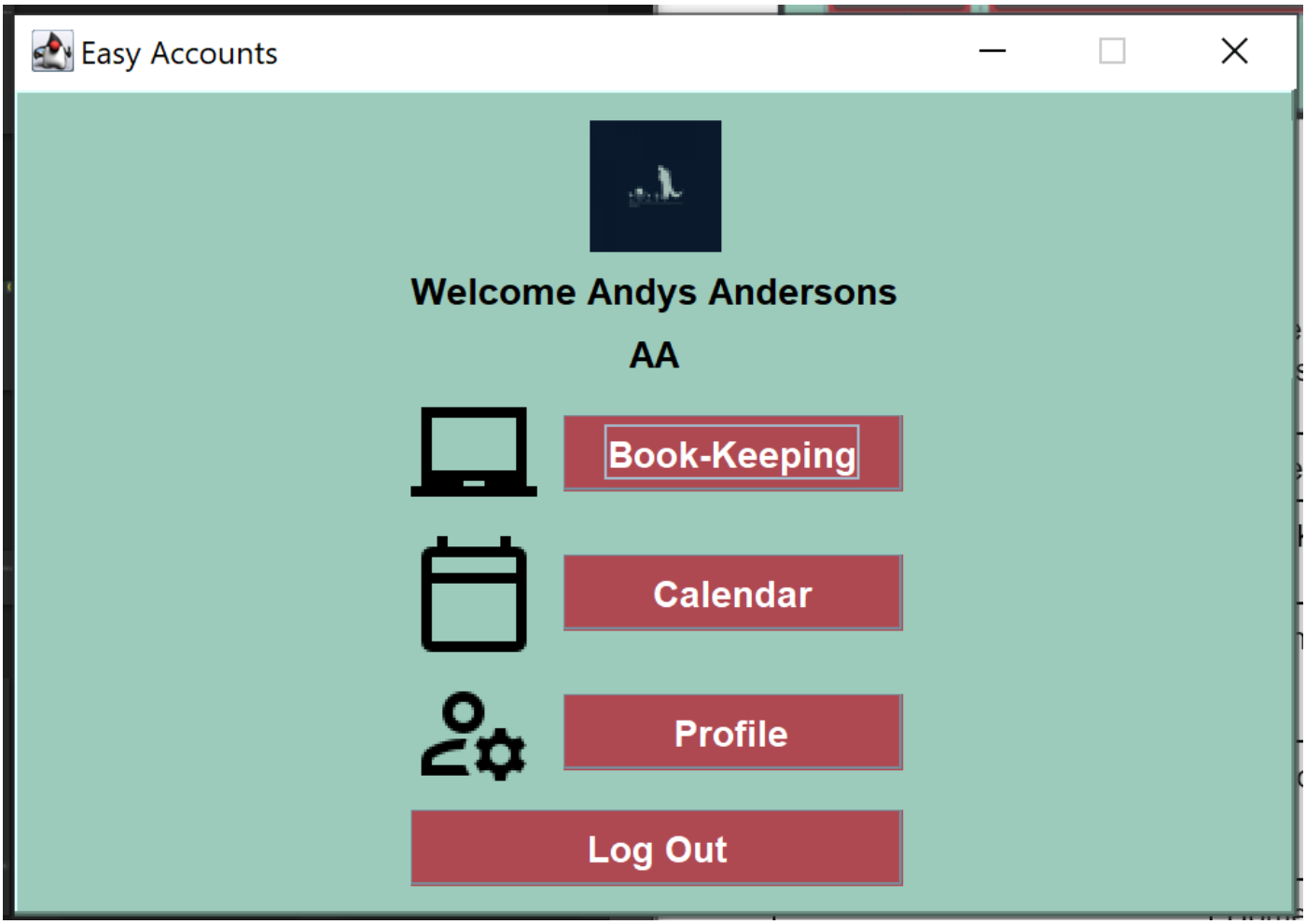

Figure 42 - Office worker home page

Other Components
Data entered by the user needed to be extracted from the front-end user interface. This was done by using javax.swing components.

Component	Folders/Classes	Role
JButton	BookKeepingPages (Folder)	To switch to different pages or to save as a PDF.
	CalendarPages (Folder)	To switch to different pages, to remove events, add events or to save as a PDF.
	EmployeePages (Folder)	To switch to different pages, to remove events or to save as a

		PDF.
	HomePages (Folder)	To switch to different pages.
	PasswordPages (Folder)	To switch to different pages or to change a password.
	PreLogin (Folder)	To switch to different pages or to create a new user.
	ProfilePages (Folder)	To switch to different pages or to make changes to a user's account.
JTextField	AddEventPage (Class)	To allow the user to enter information for a new event.
	ForgotPasswordPage (Class)	To allow the user to enter answers to the security questions.
	PreLogin (Folder)	To fill in personal information when registering and username when logging in.
	ProfilePages (Folder)	To show and edit personal information.
JPasswordField	ResetPasswordPage (Class)	To enter a new password.
	RegisterPage (Class)	To enter the password.
JComboBox	RegisterPage (Class)	To choose the security questions and gender.

	ForgotPasswordPage (Class)	To choose the security questions.
	EditProfilePage (Class)	To choose the security questions and gender.
JRadioButton	AddEventPage (Class)	To choose the type of event.
JTable	Calendar (Class)	To display events.
	BookKeepingPages (Folder)	To display different financial tables.
	EmployeePages (Folder)	To display employees.

Try and Catch Exception handling statements

```
public void SetNewPassword(String CUsername, String NPassword) {
    try {
        String host = "jdbc:mysql://localhost:3306/easyaccounts_db";
        String uName = "User";
        String uPass = "User1234";
        Connection con = DriverManager.getConnection(host, uName, uPass); //make connection to database

        Statement stmt1 = con.createStatement();
        String SQL1 = "UPDATE easyaccounts_db.ea_users SET Password='" + NPassword + "' " +
                "WHERE Username='" + CUsername + "';"; //set new password
        stmt1.executeUpdate(SQL1);

        con.close();
    } catch (SQLException err) {
        System.out.println(err.getMessage());
    }
} // set new password for registered user
```

Figure 43 - Method using Try and Catch to handle SQLExceptions

Try and Catch Exception statements were used in portions of the code to catch and define any SQLExceptions or IOExceptions.

Words: 1078 (without headings, figure titles, and table)

Works Cited

aceraven777. “How to insert images in blob in mysql table using only sql syntax (without PHP)?” *Stack Overflow*, Stack Exchange Inc, 17 December 2015, https://stackoverflow.com/questions/34328601/how-to-insert-images-in-blob-in-mysql-table-using-only-sql-syntax-without-php. Accessed 5 April 2022.

Atapoor, Noor Mohammad. “converting blob to an image stream and assign it to jLabel.” *Stack Overflow*, Stack Exchange Inc, 25 May 2013, https://stackoverflow.com/questions/16212169/converting-blob-to-an-image-stream-and-assign-it-to-jlabel. Accessed 5 April 2022.

Braun, Matthias. “java - Scroll JScrollPane to bottom.” *Stack Overflow*, Stack Exchange Inc, 9 July 2015, https://stackoverflow.com/questions/5147768/scroll-jscrollpane-to-bottom/5150437. Accessed 5 April 2022.

camickr. “java - Invisible components still take up space JPanel.” *Stack Overflow*, Stack Exchange Inc, 11 October 2011, https://stackoverflow.com/questions/7727728/invisible-components-still-take-up-space-jpanel. Accessed 5 April 2022.

Developer, director. *Generate pdf in java*. YouTube, 2017. *YouTube*, https://www.youtube.com/watch?v=TAJoVkaVv-8. Accessed 5 April 2022.

DimaSan. “How to go about saving an image in blob format to MySQL in Java.” *Stack Overflow*, Stack Exchange Inc, 28 December 2016, https://stackoverflow.com/questions/41369055/how-to-go-about-saving-an-image-in-blob-format-to-mysql-in-java. Accessed 5 April 2022.

“Error 1265 (01000) data truncated for column at row - MySQL.” *TablePlus*, TablePlus Inc, 23 August 2019, https://tableplus.com/blog/2019/08/error-1265-data-truncated-for-column-at-row.html. Accessed 5 April 2022.

“Export JTable data to PDF file in Java using iText.” *Istime4j*, Istime4j, 12 January 2014, http://istime4j.blogspot.com/2014/01/export-jtable-data-to-pdf-file-in-java.html. Accessed 5 April 2022.

fthiella. “MySQL delete multiple rows in one query conditions unique to each row.” *Stack Overflow*, Stack Exchange Inc, 15 February 2013, https://stackoverflow.com/questions/14904067/mysql-delete-multiple-rows-in-one-query-conditions-unique-to-each-row. Accessed 5 April 2022.

GorvGoyle. “MySQL Workbench Edit Table Data is read only.” *Stack Overflow*, Stack Exchange Inc, 4 March 2016, https://stackoverflow.com/questions/10815029/mysql-workbench-edit-table-data-is-read-only. Accessed 5 April 2022.

“Java JScrollBar - javatpoint.” *Javatpoint*, www.javatpoint.com, https://www.javatpoint.com/java-jscrollbar. Accessed 5 April 2022.

Javatpoint. “How to Return an Array in Java.” *Javatpoint*, Javatpoint, https://www.javatpoint.com/how-to-return-an-array-in-java. Accessed 5 April 2022.

Javatpoint. “Java JTable - javatpoint.” *Javatpoint*, Javatpoint, https://www.javatpoint.com/java-jtable. Accessed 5 April 2022.

KonfHub. “Programming by Example: Generating PDFs in Java.” *KonfHub*, KonfHub, 29 October 2019, https://konfhub.medium.com/programming-by-example-generating-pdfs-in-java-f02e961b42b1. Accessed 5 April 2022.

mcswizz. “mysql DELETE rows when multiple conditions are met.” *Stack Overflow*, Stack Exchange Inc, 17 July 2017, https://stackoverflow.com/questions/35109495/mysql-delete-rows-when-multiple-conditions-are-met. Accessed 5 April 2022.

Mishra, Neeraj. “Save and Retrieve Image from MySql Database Using Java.” *The Crazy Programmer*, The Crazy Programmer, https://www.thecrazyprogrammer.com/2016/01/save-and-retrieve-image-from-mysql-database-using-java.html. Accessed 5 April 2022.

Muhammad. “java - Autoresize the image using imageicon on jlabel.” *Stack Overflow*, Stack Exchange Inc, 23 November 2014, https://stackoverflow.com/questions/27089290/autoresize-the-image-using-imageicon-on-jlabel. Accessed 5 April 2022.

nbz. “java - Enable one JButton and disable another.” *Stack Overflow*, Stack Exchange Inc, 14 October 2014, https://stackoverflow.com/questions/26366331/enable-one-jbutton-and-disable-another. Accessed 5 April 2022.

Pask. “"Column count doesn't match value count at row" but it does.” *Stack Overflow*, 24 October 2012, https://stackoverflow.com/questions/13044901/column-count-doesnt-match-value-count-at-row-but-it-does. Accessed 5 April 2022.

php. “Insert auto increment primary key to existing table.” *Stack Overflow*, Stack Exchange Inc, 27 January 2014, https://stackoverflow.com/questions/9070764/insert-auto-increment-primary-key-to-existing-table. Accessed 5 April 2022.

pietv8x. “java - return a double value from a Jtable.” *Stack Overflow*, Stack Exchange Inc, 30 November 2014, https://stackoverflow.com/questions/27218687/return-a-double-value-from-a-jtable. Accessed 5 April 2022.

Porwal, Arpit. “Appending a new row to a JTable at runtime.” *Stack Overflow*, Stack Exchange Inc, 9 January 2017, https://stackoverflow.com/questions/41542077/appending-a-new-row-to-a-jtable-at-runtime. Accessed 5 April 2022.

Sinha, Bharat. “MySQL Insert query doesn't work with WHERE clause.” *Stack Overflow*, Stack Exchange Inc, 11 August 2012, https://stackoverflow.com/questions/485039/mysql-insert-query-doesnt-work-with-where-clause. Accessed 5 April 2022.

Sololearn. “Why is the method ignored in this case?” *Sololearn*, Sololearn, 8 April 2019, https://www.sololearn.com/Discuss/1752152/why-is-the-method-ignored-in-this-case. Accessed 5 April 2022.

Stack Overflow. “"Column count doesn't match value count at row" but it does.” *Stack Overflow*, Stack Overflow, 24 October 2012, https://stackoverflow.com/questions/13044901/column-count-doesnt-match-value-count-at-row-but-it-does. Accessed 5 April 2022.

Stack Overflow. “Creating array of custom objects in java.” *Stack Overflow*, Stack Overflow, 5 December 2013, https://stackoverflow.com/questions/20402764/creating-array-of-custom-objects-in-java. Accessed 5 April 2022.

Stack Overflow. “How display in JTable the data from MySQL.” *Stack Overflow*, Stack Overflow, 26 August 2017, https://stackoverflow.com/questions/45896437/how-display-in-jtable-the-data-from-mysql. Accessed 5 April 2022.

Stack Overflow. “How do I get which JRadioButton is selected from a ButtonGroup.” *Stack Overflow*, Stack Overflow, 14 October 2008, https://stackoverflow.com/questions/201287/how-do-i-get-which-jradiobutton-is-selected-from-a-buttongroup. Accessed 5 April 2022.

Stack Overflow. “How Do I Make Sure Only One JRadioButton Is Selected In Java?” *Stack Overflow*, Stack Overflow, 10 March 2020, https://stackoverflow.com/questions/60611779/how-do-i-make-sure-only-one-jradiobutton-is-selected-in-java. Accessed 5 April 2022.

Stack Overflow. “How to print Two-Dimensional Array like table.” *Stack Overflow*, Stack Overflow, 11 October 2012, https://stackoverflow.com/questions/12845208/how-to-print-two-dimensional-array-like-table. Accessed 5 April 2022.

Stack Overflow. “java - Adding JTable Model (IntelliJ Forms).” *Stack Overflow*, Stack Overflow, 2 February 2017, https://stackoverflow.com/questions/41993361/adding-jtable-model-intellij-forms/41993732. Accessed 5 April 2022.

Stack Overflow. “java - How to add row in JTable?” *Stack Overflow*, Stack Overflow, https://stackoverflow.com/questions/3549206/how-to-add-row-in-jtable. Accessed 5 April 2022.

Studytonight. “How to Return an Array in Java.” *Studytonight*, Studytonight,

https://www.studytonight.com/java-examples/how-to-return-an-array-in-java. Accessed 5 April 2022.

user1709952. “How to add a row in a JTable when click focus is out of the Table and <ENTER> key is pressed.” *Stack Overflow*, Stack Exchange Inc, 27 January 2014, https://stackoverflow.com/questions/21378390/how-to-add-a-row-in-a-jtable-when-click-focus-is-out-of-the-table-and-enter-ke. Accessed 5 April 2022.

user552245. “How2: Add a JPanel to a Document then export to PDF.” *Stack Overflow*, Stack Overflow, 23 December 2010, https://stackoverflow.com/questions/4517907/how2-add-a-jpanel-to-a-document-then-export-to-pdf. Accessed 5 April 2022.