Spring Master Project Proposal

Class: APCS-A

Period (Circle): 2-Schenk 6-Schenk

Name (FULL): Lokesh Budda Sankar Narayan

Executive Summary

I'm creating an expense app similar to Splitwise but it can also track your own income/expenses. Users can add their income or expenses in the app. Income can come from different sources. Similarly expenses can be of different types. It will allow users to create expenses and split them among people. In this app users can create an account, they can add expenses and share it with other users of the app. The app will be able to show the total amount owed by the user. The app will also be able to simplify expenses between the same set of users, for example if Tom owes $10 to Sam and Sam owes $6 to Tom then the app will simplify it as Tom owes $4 and Sam owes nothing. Users will be able to see a summary of their income vs expenses in a given time period.

Specifics to Assist in Project Approval

- **GUI - Enter income or expenses, View Summary**
- **Database - Login information, Income/Expenses information**
- **Inheritance - Types of Incomes, Types of Expenses**

I understand that all code submitted under my Spring project must be of my own authoring. No one else may generate or otherwise write any algorithms for me. Any code submitted that is 3rd party will be presented to the teacher for prior approval if specific algorithms or libraries are required outside the libraries available in class as part of our normal lab work or language libraries provided.

Lokesh B.S. 9/1/2021

Signature Date

Project is: Approved Unapproved (See teacher) or

Approved as modified below

```java
package application;
/**
 *  Singleton Class to hold data through the different pane changes.s
 * @author lokeshbudda
 *
 */
public final class UserHolder {

    private User user;
    private String name = "";
    private final static UserHolder INSTANCE = new UserHolder();

    private UserHolder() {
    }

    public static UserHolder getInstance() {
        return INSTANCE;
      }

    public void setUser(User u) {
        this.user = u;
    }

    public User getUser() {
        return this.user;
    }
```

```
27
28      public void setName(String name) {
29          this.name = name;
30      }
31
32      public String getName() {
33          return this.name;
34      }
35  }
```

```java
package application;
/**
 * poco class
 * @author lokeshbudda
 *
 */
public class User {
    public String name;
    public User(String name) {
        this.name = name;
    }

    public User() {
        this.name = "";
    }
}

```

```java
package application;

import java.util.ArrayList;

public class Transactions {
    ArrayList<Transaction> s  = new ArrayList<Transaction>();
    User user;


    public Transactions(ArrayList<Transaction> s, User user) {
        super();
        this.s = s;
        this.user = user;
    }

    public ArrayList<Transaction> getS() {
        return s;
    }
    public void setS(ArrayList<Transaction> s) {
        this.s = s;
    }
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
```

```
27    }
28
29}
30
```

```java
package application;

import java.sql.Date;

public abstract class Transaction {
    public Date date;
    public String title;
    public Double amount;
    public String Type;
    public String description;
    public String UserName;
    public Boolean pending;

    public Transaction() {
    }

    public Transaction(Date date, String title, Double amount, String type, String description, String userName , Boolean pending) {
        this.date = date;
        this.title = title;
        this.amount = amount;
        Type = type;
        this.description = description;
        UserName = userName;
        this.pending = pending;
    }
    public Date getDate() {
        return date;
    }
    public void setDate(Date date) {
        this.date = date;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public Double getAmount() {
        return amount;
    }
    public void setAmount(Double amount) {
        this.amount = amount;
    }
    public String getType() {
        return Type;
    }
    public void setType(String type) {
        Type = type;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getUserName() {
```

```
58          return UserName;
59      }
60
61      public void setUserName(String userName) {
62          UserName = userName;
63      }
64      public Boolean getPending() {
65          return pending;
66      }
67
68      public void setPending(Boolean pending) {
69          this.pending = pending;
70      }
71      public abstract void showTransaction();
72      public abstract ArrayList<String> getCategoryOptions();
73
74  }
75
```

```java
package application;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class SharedExpense extends Expense implements Presistable{
    private String payer;
    private Double amount;
    private List<String> SharedWith;

    /*
     * getters and setters
     */
    public String getPayer() {
        return payer;
    }
    public void setPayer(String payer) {
        this.payer = payer;
    }
    public Double getAmount() {
        return amount;
    }
    public void setAmount(Double amount) {
        this.amount = amount;
    }

    public List<String> getSharedWith() {
        return SharedWith;
    }
    public void setSharedWith(List<String> sharedWith) {
        SharedWith = sharedWith;
    }
    /**
     * Save - Saves all the data into the database.
     */
    @Override
    public void save() {
        Connection conn;
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/LoginDatabase", "root", "password");

            PreparedStatement ps = conn.prepareStatement(
                    "INSERT INTO `MYAPP`.`DETAILS` (title, description, subType) VALUES (?, ?, ?)", Statement.RETURN_GENERATED_KEYS);

            ps.setString(1, this.getTitle());
            ps.setString(2, this.getDescription());
            ps.setString(3, this.getExpenseCatagory());
            ps.executeUpdate();
            ResultSet rs = ps.getGeneratedKeys();
            if(rs.next()) {
```

```
57                  Integer detailId = (int) rs.getLong(1);
58                  Double splitAmount = this.amount / (this.SharedWith.size() + 1);
59                  PreparedStatement ps1 = conn.prepareStatement(
60                          "INSERT INTO `MYAPP`.`TRANSACTIONS` (transactionDate, fromUser, toUser, amount, detailId) VALUES (?, ?, ?, ?, ?)");
61
62                  ps1.setDate(1, this.getDate());
63                  ps1.setString(2, this.getUserName());
64                  ps1.setString(3, "NONE");
65                  ps1.setDouble(4, splitAmount);
66                  ps1.setInt(5, detailId);
67                  ps1.executeUpdate();
68
69                  for(String user:this.SharedWith) {
70                      PreparedStatement ps2 = conn.prepareStatement(
71                              "INSERT INTO `MYAPP`.`TRANSACTIONS` (transactionDate, fromUser, toUser, amount, detailId) VALUES (?, ?, ?, ?, ?)");
72
73                      ps2.setDate(1, this.getDate());
74                      ps2.setString(2, this.getUserName());
75                      ps2.setString(3, user);
76                      ps2.setDouble(4, splitAmount);
77                      ps2.setInt(5, detailId);
78                      ps2.executeUpdate();
79                  }
80              }
81          } catch (SQLException e) {
82              e.printStackTrace();
83          }
84      }
85      //abstract class method implementation
86      @Override
87      public void showTransaction() {
88
89
90      }
91      @Override
92      public ArrayList<String> getCategoryOptions() {
93
94          return null;
95      }
96      /**
97       * getAvailableUsers - gets all the user names and displays them on a list.
98       * @return
99       */
100     public static ArrayList<String> getAvailableUsers() {
101         ArrayList<String> users = new ArrayList<String>();
102         try {
103             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/LoginDatabase", "root", "password");
104             PreparedStatement ps = conn.prepareStatement("SELECT UserName FROM MYAPP.USERS where UserName != ?");
105             ps.setString(1, UserHolder.getInstance().getName());
106             ResultSet rs = ps.executeQuery();
107             while(rs.next()) {
108                 users.add(rs.getString("UserName"));
109             }
110
111             System.out.println(users);
112
```

```
113         } catch (Exception e) {
114
115             e.printStackTrace();
116         }
117         return users;
118     }
119
120 }
121
```

```java
package application;
/**
 * Presistable Interface
 * @author lokeshbudda
 *
 */
public interface Presistable {
    public void save();
    public Transaction read();
}

```

```java
package application;
/**
 * poco class for getting owe data.
 * @author lokeshbudda
 *
 */
public class OwedData {

    private String owedUser;
    private Double owedAmount;
    public String getOwedUser() {
        return owedUser;
    }
    public OwedData() {
        super();
    }
    public OwedData(String owedUser, Double owedAmount) {
        super();
        this.owedUser = owedUser;
        this.owedAmount = owedAmount;
    }
    public void setOwedUser(String owedUser) {
        this.owedUser = owedUser;
    }
    public Double getOwedAmount() {
        return owedAmount;
```

```
27     }
28     public void setOwedAmount(Double owedAmount) {
29         this.owedAmount = owedAmount;
30     }
31 
32 }
33 
```

```java
package application;
/**
 * poco class for getting owe data.
 * @author lokeshbudda
 *
 */
public class OweData {
    private String oweUser;
    private Double oweAmount;

    public OweData(String oweUser, Double oweAmount) {
        super();
        this.oweUser = oweUser;
        this.oweAmount = oweAmount;
    }


    public String getOweUser() {
        return oweUser;
    }
    public void setOweUser(String oweUser) {
        this.oweUser = oweUser;
    }
    public OweData() {
        super();
    }
```

```
27      public Double getOweAmount() {
28          return oweAmount;
29      }
30      public void setOweAmount(Double oweAmount) {
31          this.oweAmount = oweAmount;
32      }
33
34  }
35
```

```java
package application;

import java.sql.Connection;
/**
 * Income  -  stores all the income Transactions into the database;
 * @author lokeshbudda
 *
 */
public class Income extends Transaction implements Presistable {

    private String incomeCatagory;
    public static ArrayList<String> categoryOptions = new ArrayList<String>(List.of("salary", "Side Business", "Gift/Awards"));
    private Statement statement;

    public Income(Date date, String title, Double amount, String type, String description, String userName,
            String incomeCateogry) {
        this.incomeCatagory = incomeCateogry;
    }

    public Income() {

    }
    /**
     * Save - saves all the income transaction into the right columns in the Transaction table
     */
    @Override
    public void save() {
        Connection conn;
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/LoginDatabase", "root", "password");

            PreparedStatement ps = conn.prepareStatement(
                    "INSERT INTO `MYAPP`.`DETAILS` (title, description, subType) VALUES (?, ?, ?)", Statement.RETURN_GENERATED_KEYS);

            ps.setString(1, this.getTitle());
            ps.setString(2, this.getDescription());
            ps.setString(3, this.getIncomeCatagory());
            ps.executeUpdate();
            ResultSet rs = ps.getGeneratedKeys();
            if(rs.next()) {
                int detailId = (int) rs.getLong(1);
                PreparedStatement ps1 = conn.prepareStatement(
                        "INSERT INTO `MYAPP`.`TRANSACTIONS` (transactionDate, fromUser, toUser, amount, detailId) VALUES (?, ?, ?, ?, ?)");

                ps1.setDate(1, this.getDate());
                ps1.setString(2, "NONE");
                ps1.setString(3, this.getUserName());
                ps1.setDouble(4, this.amount);
                ps1.setInt(5, detailId);
                ps1.executeUpdate();
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    /**
     * gets all the data from database and reads it.
     */
    @Override
    public Income read() {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/LoginDatabase", "root", "password");
            PreparedStatement ps = conn.prepareStatement("SELECT Username, Title, Catagory, Description, Date, Amount  FROM LoginDatabase.Income where UserName = ?");
            ArrayList<ExpenseData> tableData = new ArrayList<ExpenseData>();
```

```
74              System.out.println(UserHolder.getInstance().getName());
75
76              ps.setString(1, UserHolder.getInstance().getName());
77              statement = conn.createStatement();
78              ResultSet rs = ps.executeQuery();
79              System.out.println("run");
80              while(rs.next()) {
81
82                  this.title = rs.getString("Title");
83                  this.UserName = rs.getString("Username");
84                  this.incomeCatagory = rs.getString("Catagory");
85                  this.description = rs.getString("Description");
86                  this.date = rs.getDate("Date");
87                  this.amount = rs.getDouble("Amount");
88              }
89
90          } catch (Exception e) {
91              e.printStackTrace();
92          }
93          return this;
94      }
95      //abstract method implementation.
96      @Override
97      public void showTransaction() {
98
99      }
100     //Getters and setters.
101     public String getIncomeCatagory() {
102         return incomeCatagory;
103     }
104
105     public void setIncomeCatagory(String incomeCatagory) {
106         this.incomeCatagory = incomeCatagory;
107     }
108
109     @Override
110     public ArrayList<String> getCategoryOptions() {
111         return this.categoryOptions;
112     }
113
114 }
115
```

```java
package application;

import java.sql.Date;

public class ExpenseData {
    //Fields
    private String titleColumn;
    private String typeColumn;
    private Double amountColumn;
    private String descriptionColumn;
    //Getters and setters
    public String getTitleColumn() {
        return titleColumn;
    }
    public void setTitleColumn(String titleColumn) {
        this.titleColumn = titleColumn;
    }
    public String getTypeColumn() {
        return typeColumn;
    }
    public void setTypeColumn(String typeColumn) {
        this.typeColumn = typeColumn;
    }
    public Double getAmountColumn() {
        return amountColumn;
    }
```

```
27      public void setAmountColumn(Double amountColumn) {
28          this.amountColumn = amountColumn;
29      }
30      public String getDescriptionColumn() {
31          return descriptionColumn;
32      }
33      public void setDescriptionColumn(String descriptionColumn) {
34          this.descriptionColumn = descriptionColumn;
35      }
36
37  }
38
```

```java
package application;


import java.net.URL;
import java.sql.Date;
import java.util.ResourceBundle;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.ListView;
import javafx.scene.control.RadioButton;
import javafx.scene.control.SelectionMode;
import javafx.scene.control.SelectionModel;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.AnchorPane;

public class AddTransactionController {
    @FXML
    private TextField transactionTitle;

    @FXML
    private TextField transactionDes;

    @FXML
    private TextField transactionAmount;

    @FXML
    private ChoiceBox<String> incomeCB;

    @FXML
    private Button incomeSubmit;

    @FXML
    private ChoiceBox<String> expenseCB;

    @FXML
```

```
41      private Button expenseSubmit;
42
43      @FXML
44      private ChoiceBox<String> SharedCB;
45
46      @FXML
47      private Button sharedSubmit;
48
49      @FXML
50      private ListView<String> sharedList;
51
52      @FXML
53      private DatePicker transactionDate;
54      @FXML
55      private AnchorPane incomePane;
56
57      @FXML
58      private AnchorPane expensePane;
59
60      @FXML
61      private AnchorPane sharedPane;
62
63      @FXML
64      private RadioButton incomeButton;
65
66      @FXML
67      private RadioButton expenseButton;
68
69      @FXML
70      private RadioButton sharedButton;
71
72
73     /**
74      * initialize - sets up AddTransaction.fxml
75      */
76      @FXML
77      private void initialize() {
78          ToggleGroup group = new ToggleGroup();
79          incomeButton.setToggleGroup(group);
80          expenseButton.setToggleGroup(group);
```

```
 81         sharedButton.setToggleGroup(group);
 82
 83     incomePane.setVisible(true);
 84     expensePane.setVisible(false);
 85     sharedPane.setVisible(false);
 86
 87         this.incomeCB.getItems().addAll(Income.categoryOptions);
 88         this.expenseCB.getItems().addAll(Expense.categoryOptions);
 89         this.SharedCB.getItems().addAll(Expense.categoryOptions);
 90         this.sharedList.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);;
 91         this.sharedList.getItems().addAll(SharedExpense.getAvailableUsers());
 92     }
 93
 94     /**
 95      * incomePane - displays incomePane and hides the rest
 96      * @param event
 97      */
 98     public void incomePane(ActionEvent event) {
 99     incomePane.setVisible(true);
100     expensePane.setVisible(false);
101     sharedPane.setVisible(false);
102     }
103     /**
104      * expensePane - displays expanePane and hides the rest
105      * @param event
106      */
107     public void expensePane(ActionEvent event) {
108     incomePane.setVisible(false);
109     expensePane.setVisible(true);
110     sharedPane.setVisible(false);
111     }
112     /**
113      * shaprePane - displays sharePane and hides the rest
114      * @param event
115      */
116     public void sharePane(ActionEvent event) {
117     incomePane.setVisible(false);
118     expensePane.setVisible(false);
119     sharedPane.setVisible(true);
120     }
```

```
121
122     /**
123      * SubmitIncome  - adds all the data entered into the database correctly.
124      * @param event
125      */
126     public void SubmitIncome(ActionEvent event) {
127         System.out.println("****************");
128         Income income = new Income();
129         income.setTitle(this.transactionTitle.getText());
130         income.setDate(Date.valueOf(this.transactionDate.getValue()));
131         income.setAmount(Double.valueOf(transactionAmount.getText()));
132         income.setDescription(this.transactionDes.getText());
133         income.setIncomeCatagory(this.incomeCB.getValue());
134         income.setUserName(UserHolder.getInstance().getName());
135         System.out.println("****************");
136         income.save();
137
138     }
139     /**
140      * SubmitExpense  - adds all the data entered into the database correctly.
141      * @param event
142      */
143     public void SubmitExpense(ActionEvent event) {
144         System.out.println("*******E*********");
145         Expense expense = new Expense();
146         expense.setTitle(this.transactionTitle.getText());
147         expense.setDate(Date.valueOf(this.transactionDate.getValue()));
148         expense.setAmount(Double.valueOf(transactionAmount.getText()));
149         expense.setDescription(this.transactionDes.getText());
150         expense.setExpenseCatagory(this.incomeCB.getValue());
151         expense.setUserName(UserHolder.getInstance().getName());
152     expense.save();
153
154     }
155     /**
156      * SubmitSharedExpense  - adds all the data entered into the database correctly.
157      * @param event
158      */
159     public void SubmitSharedExpense(ActionEvent event) {
160         System.out.println("********SE********");
```

```
161         SharedExpense expense = new SharedExpense();
162         expense.setTitle(this.transactionTitle.getText());
163         expense.setDate(Date.valueOf(this.transactionDate.getValue()));
164         expense.setAmount(Double.valueOf(transactionAmount.getText()));
165         expense.setDescription(this.transactionDes.getText());
166         expense.setExpenseCatagory(this.incomeCB.getValue());
167         expense.setUserName(UserHolder.getInstance().getName());
168         expense.setSharedWith(this.sharedList.getSelectionModel().getSelectedItems());
169         expense.save();
170     }
171 }
172
173
```

```java
package application;

import java.sql.Connection;
/**
 * expense - adds to existing catagories
 * @author lokeshbudda
 *
 */
public class Expense extends Transaction implements Presistable{

    private String expenseCatagory;
    public static ArrayList<String> categoryOptions = new ArrayList<String>(List.of("Food", "Transportation", "Entertainment", "Gym", "Shopping", "Other"));
    public Expense() {
        super();

    }

    public Expense(Date date, String title, Double amount, String type, String description, String userName, Boolean pending) {
        super(date, title, amount, type, description, userName, pending);

    }
    /**
     * save(Abstract method) - saves all data to the correct categories.
     */
    @Override
    public void save() {
        Connection conn;
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/LoginDatabase", "root", "password");

            PreparedStatement ps = conn.prepareStatement(
                    "INSERT INTO `MYAPP`.`DETAILS` (title, description, subType) VALUES (?, ?, ?)", Statement.RETURN_GENERATED_KEYS);

            ps.setString(1, this.getTitle());
            ps.setString(2, this.getDescription());
            ps.setString(3, this.getExpenseCatagory());
            ps.executeUpdate();
            ResultSet rs = ps.getGeneratedKeys();
            if(rs.next()) {
                int detailId = (int) rs.getLong(1);
                PreparedStatement ps1 = conn.prepareStatement(
```

```
50                          "INSERT INTO `MYAPP`.`TRANSACTIONS` (transactionDate, fromUser, toUser, amount, detailId) VALUES
   (?, ?, ?, ?, ?)");
51
52                  ps1.setDate(1, this.getDate());
53                  ps1.setString(2, this.getUserName());
54                  ps1.setString(3, "NONE");
55                  ps1.setDouble(4, this.amount);
56                  ps1.setInt(5, detailId);
57                  ps1.executeUpdate();
58              }
59          } catch (SQLException e) {
60              // TODO Auto-generated catch block
61              e.printStackTrace();
62          }
63      }
64      //Abstract method implementation
65      @Override
66      public Expense read() {
67          return null;
68
69
70      }
71      //Abstract method implementation
72      @Override
73      public void showTransaction() {
74
75
76      }
77      //getter and setter
78      public String getExpenseCatagory() {
79          return expenseCatagory;
80      }
81
82      public void setExpenseCatagory(String expenseCatagory) {
83          this.expenseCatagory = expenseCatagory;
84      }
85
86      @Override
87      public ArrayList<String> getCategoryOptions() {
88          return this.categoryOptions;
89      }
90
91  }
92
```

```java
package application;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Side;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.Button;
import javafx.scene.control.CheckBox;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.MenuItem;
import javafx.scene.control.PasswordField;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;

import java.net.URL;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.ResourceBundle;

public class MainController implements Initializable {
    @FXML
```

```
50      private TextField userNameText;
51
52      @FXML
53      private PasswordField passwordText;
54
55      @FXML
56      private Button login;
57
58      @FXML
59      private Button signUp;
60
61      @FXML
62      private TextField userNameTextS;
63      @FXML
64      private TextField passwordTextS;
65
66      @FXML
67      private TextField checkPasswordText;
68
69      @FXML
70      private Button addTransaction;
71
72      @FXML
73      private TextField expenseName;
74
75      @FXML
76      private ChoiceBox<String> expenseType = new ChoiceBox<>();
77
78      @FXML
79      private TextField priceText;
80
81      @FXML
82      private TextArea description;
83
84      @FXML
85      private Button submit;
86
87      @FXML
88      private DatePicker expenseDate = new DatePicker();
89
90      @FXML
91      public ListView<String> ExpenseList = new ListView<String>();
92
93      @FXML
94      Button refresh;
95
96      private Statement statement;
97
98      @FXML
```

```
 99     private HBox expenseBox = new HBox(ExpenseList);
100
101     @FXML
102     private PieChart incomePieChart = new PieChart();
103
104     @FXML
105     private PieChart expensePieChart = new PieChart();
106
107     @FXML
108     private Label incomeLabel = new Label();
109
110     @FXML
111     private Label expenseLabel = new Label();
112
113     @FXML
114     private Label oweLabel = new Label();
115
116     @FXML
117     private Label owedLabel = new Label();
118
119     @FXML
120     private TableView<OweData> oweTable = new TableView<>();
121     @FXML
122     private TableColumn<OweData, String> oweUser = new TableColumn<>();
123     @FXML
124     private TableColumn<OweData, Double> oweAmount = new TableColumn<>();
125
126     @FXML
127     private TableView<OwedData> owedTable = new TableView<>();
128     @FXML
129     private TableColumn<OwedData, String> owedUser = new TableColumn<>();
130     @FXML
131     private TableColumn<OwedData, Double> owedAmount = new TableColumn<>();
132
133     @FXML
134     private TableView<ExpenseData> transactionsTable = new TableView<>();
135     @FXML
136     private TableColumn<ExpenseData, String> titleColumn = new TableColumn<>();
137     @FXML
138     private TableColumn<ExpenseData, String> typeColumn = new TableColumn<>();
139     @FXML
140     private TableColumn<ExpenseData, Double> amountColumn = new TableColumn<>();
141     @FXML
142     private TableColumn<ExpenseData, String> descriptionColumn = new TableColumn<>();
143
144     @FXML
145     private RadioButton incomeChartButton = new RadioButton();
146     @FXML
147     private RadioButton expenseChartButton = new RadioButton();
```

```
148
149     String currentUser = "";
150     Parent root;
151     Stage primaryStage = new Stage();
152
153     ArrayList<OwedData> owedList = new ArrayList<>();
154     ArrayList<OweData> oweList = new ArrayList<>();
155
156     public TextField getUserNameText() {
157         return userNameText;
158     }
159
160     public void setUserNameText(TextField userNameText) {
161         this.userNameText = userNameText;
162     }
163
164     /**
165      * initialize  - sets up everything for the program to function properly.
166      */
167     @Override
168     public void initialize(URL arg0, ResourceBundle arg1) {
169         this.expenseType.getItems().addAll("Food", "Income", "Other");
170         oweAmount.setCellValueFactory(new PropertyValueFactory<OweData, Double>("oweAmount"));
171         oweUser.setCellValueFactory(new PropertyValueFactory<OweData, String>("oweUser"));
172         owedAmount.setCellValueFactory(new PropertyValueFactory<OwedData, Double>("owedAmount"));
173         owedUser.setCellValueFactory(new PropertyValueFactory<OwedData, String>("owedUser"));
174
175         titleColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("titleColumn"));
176         typeColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("typeColumn"));
177         amountColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, Double>("amountColumn"));
178         descriptionColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("descriptionColumn"));
179
180         initializeExpenses();
181         initializeAmountExpected();
182         initializeAmountOwe();
183         initializeTotalExpense();
184         initializeTotalIncome();
185
186         simplifyExpenses();
187         incomePieChart();
188         expensePieChart();
189
190         ToggleGroup group = new ToggleGroup();
191         incomeChartButton.setToggleGroup(group);
192         expenseChartButton.setToggleGroup(group);
193
194         incomePieChart.setOpacity(1);
195         expensePieChart.setOpacity(0);
196
```

```
197
198     }
199     /**
200      * SimplifyExpense  - it displays the expense data in a smaller table.
201      */
202     private void simplifyExpenses() {
203         HashMap<String, Double> simplified = new HashMap<>();
204         for(OweData data:this.oweList) {
205             if (simplified.containsKey(data.getOweUser())) {
206                 simplified.put(data.getOweUser(), simplified.get(data.getOweUser()) - data.getOweAmount());
207             } else {
208                 simplified.put(data.getOweUser(), -1*data.getOweAmount());
209             }
210         }
211         for(OwedData owed:this.owedList) {
212             if (simplified.containsKey(owed.getOwedUser())) {
213                 simplified.put(owed.getOwedUser(), simplified.get(owed.getOwedUser()) + owed.getOwedAmount());
214             } else {
215                 simplified.put(owed.getOwedUser(), owed.getOwedAmount());
216             }
217         }
218         this.owedList = new ArrayList<>();
219         this.oweList =  new ArrayList<>();
220
221         Double owed = (double) 0;
222
223         Double owe = (double) 0;
224
225         for (Map.Entry<String,Double> entry : simplified.entrySet()) {
226             if(entry.getValue() > 0) {
227                 owed += entry.getValue();
228                 this.owedList.add(new OwedData(entry.getKey(), entry.getValue()));
229             } else {
230                 owe += (-1 * entry.getValue());
231                 this.oweList.add(new OweData(entry.getKey(), -1 * entry.getValue()));
232             }
233         }
234
235         this.owedTable.getItems().addAll(this.owedList);
236         this.oweTable.getItems().addAll(this.oweList);;
237
238         owedLabel.setText(owed.toString());
239         oweLabel.setText(owe.toString());
240
241
242     }
243
244     /**
245      * Login method - searches mysql database MYAPP for matches of Username and Password
```

```
246      * it launches MainPage.fxml if the the executed Result Set is true.
247      * @param Event
248      * @throws Exception
249      */
250     public void login(ActionEvent Event) throws Exception {
251         try {
252             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
253                     "password");
254 
255             String sql = "SELECT count(*) FROM MYAPP.USERS where username = ? and password = ? ";
256             PreparedStatement pst = conn.prepareStatement(sql);
257             pst.setString(1, getUserNameText().getText());
258             pst.setString(2, passwordText.getText());
259             ResultSet rs = pst.executeQuery();
260             this.currentUser = getUserNameText().getText();
261             System.out.println(this.currentUser);
262             if (rs.next()) {
263 
264                 User u = new User();
265                 u.name = userNameText.getText();
266                 UserHolder userholder = UserHolder.getInstance();
267                 userholder.setUser(u);
268                 userholder.setName(userNameText.getText());
269 
270                 BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("MainPage.fxml"));
271 
272                 Scene scene = new Scene(root, 1251, 856);
273                 scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
274 
275                 primaryStage.setScene(scene);
276                 primaryStage.show();
277 
278             } else {
279                 System.out.println("Login Failed");
280             }
281         } catch (Exception e) {
282             System.out.println("Exception: " + e.toString());
283         }
284     }
285     /**
286      * SignUpButton method - opens SignUp.fxml when signUp button is clicked.
287      * @param Event
288      * @throws Exception
289      */
290     public void signUpButton(ActionEvent Event) throws Exception {
291         BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("SignUp.fxml"));
292         Scene scene = new Scene(root, 334, 361);
293         scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
294         primaryStage.setScene(scene);
```

```
295         primaryStage.show();
296     }
297
298     /**
299      * SignUp method - it inserts the data entered from the signUp.fxml into the mysql database
300      * @param Event
301      * @throws Exception
302      */
303     public void signUp(ActionEvent Event) throws Exception {
304         try {
305             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
306                     "password");
307             PreparedStatement pst = conn.prepareStatement("INSERT INTO MYAPP.USERS values(?,?)");
308             pst.setString(1, userNameTextS.getText());
309             pst.setString(2, passwordTextS.getText());
310             if (passwordTextS.getText().equals(checkPasswordText.getText())) {
311                 pst.executeUpdate();
312                 System.out.println("Data Registerded");
313             } else
314                 System.out.println("something went wrong");
315         } catch (Exception e) {
316             System.out.println("Exception: " + e.toString());
317         }
318     }
319     /**
320      * AddTransaction - opens AddTransaction.fxml
321      * @param Event
322      * @throws Exception
323      */
324     public void addTransaction(ActionEvent Event) throws Exception {
325         BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("AddTransaction.fxml"));
326         Scene scene = new Scene(root, 519, 425);
327         scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
328         primaryStage.setScene(scene);
329         primaryStage.show();
330         // user.setUserName(currentUser);
331     }
332     /**
333      * initializedExpenses - it executes a  throught the Details table
334      * to get all the data, then takes the total amount for each and
335      * inserts them into the transaction table.
336      * @return
337      */
338     public ListView<String> initializeExpenses() {
339         try {
340             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
341                     "password");
342             PreparedStatement ps = conn.prepareStatement("SELECT DETAILS.detailId, DETAILS.title, " + "CASE "
343                     + "WHEN TRANSACTIONS.fromUser = ? AND TRANSACTIONS.toUser = 'NONE' THEN 'Expense' "
```

```
344                     + "WHEN TRANSACTIONS.toUser = ? AND TRANSACTIONS.fromUser = 'NONE' THEN 'Income' "
345                     + "ELSE 'Shared' " + "END AS TRANSACTION_TYPE, "
346                     + "MAX(TRANSACTIONS.transactionDate) AS TRANSACTION_DATE, SUM(TRANSACTIONS.amount) AS TRANSACTION_AMOUNT "
347                     + "FROM `MYAPP`.TRANSACTIONS, `MYAPP`.DETAILS " + "WHERE DETAILS.detailId = TRANSACTIONS.detailid "
348                     + "AND (TRANSACTIONS.fromUser = ? OR TRANSACTIONS.toUser = ? )"
349                     + "GROUP BY DETAILS.detailId, DETAILS.title, TRANSACTION_TYPE " + "ORDER BY DETAILS.detailid");
350
351             ps.setString(1, UserHolder.getInstance().getName());
352             ps.setString(2, UserHolder.getInstance().getName());
353             ps.setString(3, UserHolder.getInstance().getName());
354             ps.setString(4, UserHolder.getInstance().getName());
355             statement = conn.createStatement();
356             ResultSet rs = ps.executeQuery();
357             ObservableList<ExpenseData> expenses = FXCollections.observableArrayList();
358             while (rs.next()) {
359                 ExpenseData expenseData = new ExpenseData();
360                 expenseData.setTitleColumn(rs.getString("title"));
361                 expenseData.setTypeColumn(rs.getString("TRANSACTION_TYPE"));
362                 expenseData.setDescriptionColumn(rs.getDate("TRANSACTION_DATE").toString());
363                 expenseData.setAmountColumn(rs.getDouble("TRANSACTION_AMOUNT"));
364                 expenses.add(expenseData);
365             }
366             transactionsTable.setItems(expenses);
367         } catch (Exception e) {
368
369             e.printStackTrace();
370         }
371         return ExpenseList;
372     }
373     /**
374      * initializeTotalExpense - takes data from TRANSACTIONS table stored in the mySQL database.
375      * then takes the amount under any expense category and adds them together.
376      */
377     public void initializeTotalExpense() {
378         try {
379             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
380                     "password");
381             PreparedStatement ps = conn
382                     .prepareStatement("SELECT SUM(TRANSACTIONS.amount) AS TOTAL_EXPENSES " + "FROM MYAPP.TRANSACTIONS "
383                             + "WHERE TRANSACTIONS.fromUser = ? " + "AND TRANSACTIONS.touser = 'NONE'");
384
385
386             ps.setString(1, UserHolder.getInstance().getName());
387             statement = conn.createStatement();
388             ResultSet rs = ps.executeQuery();
389
390             while (rs.next()) {
391                 System.out.println(rs.getString("TOTAL_EXPENSES"));
392                 this.expenseLabel.setText(rs.getString("TOTAL_EXPENSES"));
```

```
393             }
394
395         } catch (Exception e) {
396             e.printStackTrace();
397         }
398     }
399     /**
400      * initializeTotalIncome - takes data from TRANSACTION table stored in the mySQL database.
401      * then takes the amount under any income category and adds them together.
402      */
403     public void initializeTotalIncome() {
404         try {
405             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
406                     "password");
407             PreparedStatement ps = conn
408                     .prepareStatement("SELECT SUM(TRANSACTIONS.amount) AS TOTAL_INCOME " + "FROM MYAPP.TRANSACTIONS "
409                             + "WHERE TRANSACTIONS.touser = ? " + "AND TRANSACTIONS.fromUser = 'NONE'");
410
411
412             ps.setString(1, UserHolder.getInstance().getName());
413             statement = conn.createStatement();
414             ResultSet rs = ps.executeQuery();
415
416             while (rs.next()) {
417                 System.out.println(rs.getString("TOTAL_INCOME"));
418                 this.incomeLabel.setText(rs.getString("TOTAL_INCOME"));
419             }
420
421         } catch (Exception e) {
422             e.printStackTrace();
423         }
424     }
425
426     /**
427      * initializeAmountExpected - finds the amount of money owed by other user
428      */
429     public void initializeAmountExpected() {
430         try {
431             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
432                     "password");
433             PreparedStatement ps = conn.prepareStatement("SELECT touser, SUM(TRANSACTIONS.amount) AMOUNT_EXPECTED "
434                     + "FROM MYAPP.TRANSACTIONS " + "WHERE fromuser != 'NONE' AND touser != 'NONE' "
435                     + "AND fromUser = ? " + "GROUP BY touser");
436
437
438             ps.setString(1, UserHolder.getInstance().getName());
439             statement = conn.createStatement();
440             ResultSet rs = ps.executeQuery();
441             while (rs.next()) {
```

```
442                 OwedData currentRow = new OwedData();
443                 currentRow.setOwedUser(rs.getString("touser"));
444                 currentRow.setOwedAmount(rs.getDouble("AMOUNT_EXPECTED"));
445                 owedList.add(currentRow);
446             }
447 
448         } catch (Exception e) {
449             e.printStackTrace();
450         }
451     }
452     /**
453      * initializeAmountOwe - finds the amount you own to other users
454      */
455     public void initializeAmountOwe() {
456         try {
457             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
458                     "password");
459             PreparedStatement ps = conn.prepareStatement("SELECT fromuser AS USER, SUM(TRANSACTIONS.amount) AMOUNT_OWED "
460                     + "FROM MYAPP.TRANSACTIONS " + "WHERE fromuser != 'NONE' AND touser != 'NONE' " + "AND touser = ? "
461                     + "GROUP BY fromuser");
462 
463 
464             ps.setString(1, UserHolder.getInstance().getName());
465             statement = conn.createStatement();
466             ResultSet rs = ps.executeQuery();
467             while (rs.next()) {
468                 OweData currentRow = new OweData();
469                 currentRow.setOweUser(rs.getString(1));
470                 currentRow.setOweAmount(rs.getDouble("AMOUNT_OWED"));
471                 this.oweList.add(currentRow);
472 
473                 System.out.println(rs.getString(1) + " " + rs.getDouble("AMOUNT_OWED"));
474             }
475 
476         } catch (Exception e) {
477             e.printStackTrace();
478         }
479     }
480 
481     /**
482      * refresh - refreshes main page after pressing the refresh button.
483      * @param event
484      */
485     public void refresh(ActionEvent event) {
486         Stage stage = (Stage) ((Button) event.getSource()).getScene().getWindow();
487         stage.close();
488         try {
489             BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("MainPage.fxml"));
490 
```

```
491             Scene scene = new Scene(root, 1003, 745);
492             scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
493 
494             primaryStage.setScene(scene);
495             primaryStage.show();
496         } catch (Exception e) {
497             e.printStackTrace();
498         }
499     }
500     /**
501      * expensePieChart - Inserts data into the expense pie chart.
502      */
503     public void expensePieChart() {
504         try {
505             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
506                     "password");
507             PreparedStatement ps = conn.prepareStatement("SELECT subType, SUM(amount) AS sum_amount FROM MYAPP.DETAILS,
    MYAPP.TRANSACTIONS "
508                     + "WHERE TRANSACTIONS.detailId = DETAILS.detailId and toUser = 'NONE' and fromUser = ? GROUP by subType");
509             ps.setString(1, UserHolder.getInstance().getName());
510             ResultSet rs = ps.executeQuery();
511             ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
512             while(rs.next()) {
513 
514                         pieChartData.add(new PieChart.Data(rs.getString("subType"), rs.getDouble("sum_amount")));
515 
516             }
517 
518             expensePieChart.getData().addAll(pieChartData);
519 
520 
521 
522         } catch (SQLException e) {
523             // TODO Auto-generated catch block
524             e.printStackTrace();
525         }
526     }
527     /**
528      * incomePieChart - Inserts data into the income pie chart.
529      */
530     public void incomePieChart() {
531         try {
532             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
533                     "password");
534             PreparedStatement ps = conn.prepareStatement("SELECT subType, SUM(amount) AS sum_amount FROM MYAPP.DETAILS,
    MYAPP.TRANSACTIONS "
535                     + "WHERE TRANSACTIONS.detailId = DETAILS.detailId and fromUser = 'NONE' and toUser = ? GROUP by subType");
536             ps.setString(1, UserHolder.getInstance().getName());
537             ResultSet rs = ps.executeQuery();
```

```
538             ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
539             while(rs.next()) {
540 
541                         pieChartData.add(new PieChart.Data(rs.getString("subType"), rs.getDouble("sum_amount")));
542 
543             }
544             incomePieChart.getData().addAll(pieChartData);
545 
546 
547         } catch (SQLException e) {
548             // TODO Auto-generated catch block
549             e.printStackTrace();
550         }
551 
552     }
553     /**
554      * incomeChartButton - Displays Income PieChart
555      * @param event
556      */
557     public void incomeChartButton(ActionEvent event) {
558         incomePieChart.setOpacity(1);
559         expensePieChart.setOpacity(0);
560     }
561     /**
562      * expenseChartButton  - Displays Expense PieChart
563      * @param event
564      */
565     public void expenseChartButton(ActionEvent event) {
566         incomePieChart.setOpacity(0);
567         expensePieChart.setOpacity(1);
568     }
569 }
570 
```

```java
/*
 * Lokesh Budda
 * Period 6
 * APCSA
 * Schenk
 * Master Project
 */
package application;

import javafx.application.Application;
/**
 * Main  - starts the application
 * @author lokeshbudda
 * @since 2021-05-10
 */
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) {

        try {
            BorderPane root = (BorderPane)FXMLLoader.load(getClass().getResource("Login.fxml"));
            Scene scene = new Scene(root,341,372);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

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

```

Logout

SUMMARY:

Income : 5150.0

Expenses : 18675.5

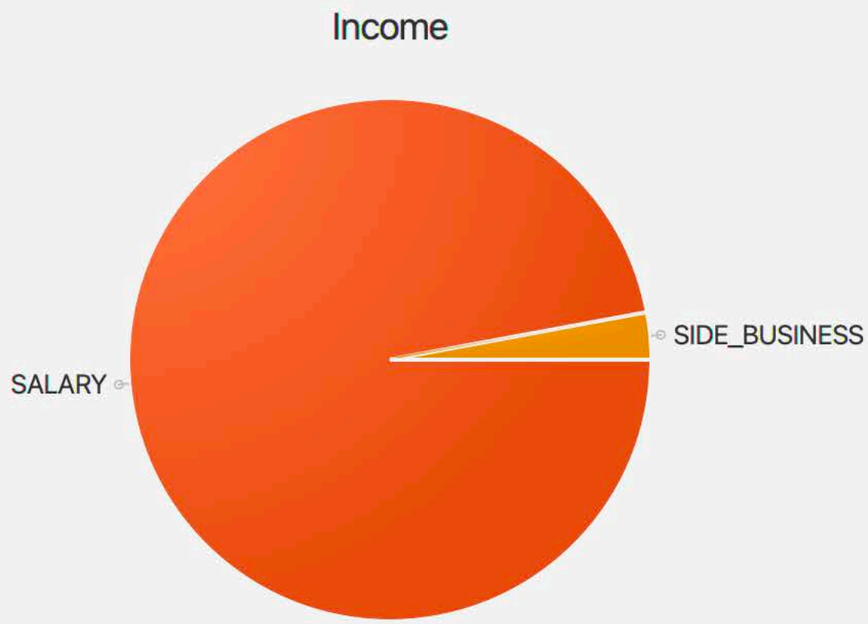
You owe : 5000.0

User	Amount
lokesh	5000.0

You are owed : 3581.5

User	Amount
USER_B	3469.0
USER_C	112.5

Chart: Income Chart Expense Chart



Title	Type	Amount	Date
My paycheck	Income	5000.0	2021-05-01
My divident	Income	150.0	2021-05-01
My Rent	Expense	1800.0	2021-05-01
Pasta	Expense	1800.0	2021-05-01
Bus	Expense	12.0	2021-05-01
Car Parking	Expense	50.0	2021-05-01
Pizza	Expense	25.0	2021-05-01
Uber	Expense	37.0	2021-05-01
Dinner at XYZ	Expense	100.0	2021-05-01
Dinner at XYZ	Shared	200.0	2021-05-01

Refresh

Add Transaction