```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
    private TextField userNameText;

    @FXML
    private PasswordField passwordText;

    @FXML
    private Button login;

    @FXML
    private Button signUp;

    @FXML
    private TextField userNameTextS;
    @FXML
    private TextField passwordTextS;

    @FXML
    private TextField checkPasswordText;

    @FXML
    private Button addTransaction;

    @FXML
    private TextField expenseName;

    @FXML
    private ChoiceBox<String> expenseType = new ChoiceBox<>();

    @FXML
    private TextField priceText;

    @FXML
    private TextArea description;

    @FXML
    private Button submit;

    @FXML
    private DatePicker expenseDate = new DatePicker();

    @FXML
    public ListView<String> ExpenseList = new ListView<String>();

    @FXML
    Button refresh;

    private Statement statement;

    @FXML
    private HBox expenseBox = new HBox(ExpenseList);

    @FXML
    private PieChart incomePieChart = new PieChart();

    @FXML
    private PieChart expensePieChart = new PieChart();

    @FXML
    private Label incomeLabel = new Label();

    @FXML
    private Label expenseLabel = new Label();

    @FXML
    private Label oweLabel = new Label();

    @FXML
    private Label owedLabel = new Label();

    @FXML
    private TableView<OweData> oweTable = new TableView<>();
    @FXML
    private TableColumn<OweData, String> oweUser = new TableColumn<>();
    @FXML
    private TableColumn<OweData, Double> oweAmount = new TableColumn<>();

    @FXML
    private TableView<OwedData> owedTable = new TableView<>();
    @FXML
    private TableColumn<OwedData, String> owedUser = new TableColumn<>();
    @FXML
    private TableColumn<OwedData, Double> owedAmount = new TableColumn<>();

    @FXML
    private TableView<ExpenseData> transactionsTable = new TableView<>();
    @FXML
    private TableColumn<ExpenseData, String> titleColumn = new TableColumn<>();
    @FXML
    private TableColumn<ExpenseData, String> typeColumn = new TableColumn<>();
    @FXML
    private TableColumn<ExpenseData, Double> amountColumn = new TableColumn<>();
    @FXML
    private TableColumn<ExpenseData, String> descriptionColumn = new TableColumn<>();

    @FXML
    private RadioButton incomeChartButton = new RadioButton();
    @FXML
    private RadioButton expenseChartButton = new RadioButton();
```

```java

    String currentUser = "";
    Parent root;
    Stage primaryStage = new Stage();

    ArrayList<OwedData> owedList = new ArrayList<>();
    ArrayList<OweData> oweList = new ArrayList<>();

    public TextField getUserNameText() {
        return userNameText;
    }

    public void setUserNameText(TextField userNameText) {
        this.userNameText = userNameText;
    }

    /**
     * initialize  - sets up everything for the program to function properly.
     */
    @Override
    public void initialize(URL arg0, ResourceBundle arg1) {
        this.expenseType.getItems().addAll("Food", "Income", "Other");
        oweAmount.setCellValueFactory(new PropertyValueFactory<OweData, Double>("oweAmount"));
        oweUser.setCellValueFactory(new PropertyValueFactory<OweData, String>("oweUser"));
        owedAmount.setCellValueFactory(new PropertyValueFactory<OwedData, Double>("owedAmount"));
        owedUser.setCellValueFactory(new PropertyValueFactory<OwedData, String>("owedUser"));

        titleColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("titleColumn"));
        typeColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("typeColumn"));
        amountColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, Double>("amountColumn"));
        descriptionColumn.setCellValueFactory(new PropertyValueFactory<ExpenseData, String>("descriptionColumn"));

        initializeExpenses();
        initializeAmountExpected();
        initializeAmountOwe();
        initializeTotalExpense();
        initializeTotalIncome();

        simplifyExpenses();
        incomePieChart();
        expensePieChart();

        ToggleGroup group = new ToggleGroup();
        incomeChartButton.setToggleGroup(group);
        expenseChartButton.setToggleGroup(group);

        incomePieChart.setOpacity(1);
        expensePieChart.setOpacity(0);


    }
    /**
     * SimplifyExpense  - it displays the expense data in a smaller table.
     */
    private void simplifyExpenses() {
        HashMap<String, Double> simplified = new HashMap<>();
        for(OweData data:this.oweList) {
            if (simplified.containsKey(data.getOweUser())) {
                simplified.put(data.getOweUser(), simplified.get(data.getOweUser()) - data.getOweAmount());
            } else {
                simplified.put(data.getOweUser(), -1*data.getOweAmount());
            }
        }
        for(OwedData owed:this.owedList) {
            if (simplified.containsKey(owed.getOwedUser())) {
                simplified.put(owed.getOwedUser(), simplified.get(owed.getOwedUser()) + owed.getOwedAmount());
            } else {
                simplified.put(owed.getOwedUser(), owed.getOwedAmount());
            }
        }
        this.owedList = new ArrayList<>();
        this.oweList =  new ArrayList<>();

        Double owed = (double) 0;

        Double owe = (double) 0;

        for (Map.Entry<String,Double> entry : simplified.entrySet()) {
            if(entry.getValue() > 0) {
                owed += entry.getValue();
                this.owedList.add(new OwedData(entry.getKey(), entry.getValue()));
            } else {
                owe += (-1 * entry.getValue());
                this.oweList.add(new OweData(entry.getKey(), -1 * entry.getValue()));
            }
        }

        this.owedTable.getItems().addAll(this.owedList);
        this.oweTable.getItems().addAll(this.oweList);;

        owedLabel.setText(owed.toString());
        oweLabel.setText(owe.toString());


    }

    /**
     * Login method - searches mysql database MYAPP for matches of Username and Password
     * it launches MainPage.fxml if the the executed Result Set is true.
     * @param Event
     * @throws Exception
     */
    public void login(ActionEvent Event) throws Exception {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                    "password");

            String sql = "SELECT count(*) FROM MYAPP.USERS where username = ? and password = ? ";
            PreparedStatement pst = conn.prepareStatement(sql);
            pst.setString(1, getUserNameText().getText());
            pst.setString(2, passwordText.getText());
            ResultSet rs = pst.executeQuery();
            this.currentUser = getUserNameText().getText();
            System.out.println(this.currentUser);
            if (rs.next()) {

                User u = new User();
                u.name = userNameText.getText();
                UserHolder userholder = UserHolder.getInstance();
                userholder.setUser(u);
                userholder.setName(userNameText.getText());

                BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("MainPage.fxml"));

                Scene scene = new Scene(root, 1251, 856);
                scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());

                primaryStage.setScene(scene);
                primaryStage.show();

            } else {
                System.out.println("Login Failed");
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
    }
    /**
     * SignUpButton method - opens SignUp.fxml when signUp button is clicked.
     * @param Event
     * @throws Exception
     */
    public void signUpButton(ActionEvent Event) throws Exception {
        BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("SignUp.fxml"));
        Scene scene = new Scene(root, 334, 361);
        scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
        primaryStage.setScene(scene);
```

```java
        primaryStage.show();
    }

    /**
     * SignUp method - it inserts the data entered from the signUp.fxml into the mysql database
     * @param Event
     * @throws Exception
     */
    public void signUp(ActionEvent Event) throws Exception {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                    "password");
            PreparedStatement pst = conn.prepareStatement("INSERT INTO MYAPP.USERS values(?,?)");
            pst.setString(1, userNameTextS.getText());
            pst.setString(2, passwordTextS.getText());
            if (passwordTextS.getText().equals(checkPasswordText.getText())) {
                pst.executeUpdate();
                System.out.println("Data Registerded");
            } else
                System.out.println("something went wrong");
        } catch (Exception e) {
            System.out.println("Exception: " + e.toString());
        }
    }
    /**
     * AddTransaction - opens AddTransaction.fxml
     * @param Event
     * @throws Exception
     */
    public void addTransaction(ActionEvent Event) throws Exception {
        BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("AddTransaction.fxml"));
        Scene scene = new Scene(root, 519, 425);
        scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
        // user.setUserName(currentUser);
    }
    /**
     * initializedExpenses - it executes a  throught the Details table
     * to get all the data, then takes the total amount for each and
     * inserts them into the transaction table.
     * @return
     */
    public ListView<String> initializeExpenses() {
        try {
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                    "password");
            PreparedStatement ps = conn.prepareStatement("SELECT DETAILS.detailId, DETAILS.title, " + "CASE "
                    + "WHEN TRANSACTIONS.fromUser = ? AND TRANSACTIONS.toUser = 'NONE' THEN 'Expense' "
```

```java
                        + "WHEN TRANSACTIONS.toUser = ? AND TRANSACTIONS.fromUser = 'NONE' THEN 'Income' "
                        + "ELSE 'Shared' " + "END AS TRANSACTION_TYPE, "
                        + "MAX(TRANSACTIONS.transactionDate) AS TRANSACTION_DATE, SUM(TRANSACTIONS.amount) AS TRANSACTION_AMOUNT "
                        + "FROM `MYAPP`.TRANSACTIONS, `MYAPP`.DETAILS " + "WHERE DETAILS.detailId = TRANSACTIONS.detailid "
                        + "AND (TRANSACTIONS.fromUser = ? OR TRANSACTIONS.toUser = ? )"
                        + "GROUP BY DETAILS.detailId, DETAILS.title, TRANSACTION_TYPE " + "ORDER BY DETAILS.detailid");

                ps.setString(1, UserHolder.getInstance().getName());
                ps.setString(2, UserHolder.getInstance().getName());
                ps.setString(3, UserHolder.getInstance().getName());
                ps.setString(4, UserHolder.getInstance().getName());
                statement = conn.createStatement();
                ResultSet rs = ps.executeQuery();
                ObservableList<ExpenseData> expenses = FXCollections.observableArrayList();
                while (rs.next()) {
                    ExpenseData expenseData = new ExpenseData();
                    expenseData.setTitleColumn(rs.getString("title"));
                    expenseData.setTypeColumn(rs.getString("TRANSACTION_TYPE"));
                    expenseData.setDescriptionColumn(rs.getDate("TRANSACTION_DATE").toString());
                    expenseData.setAmountColumn(rs.getDouble("TRANSACTION_AMOUNT"));
                    expenses.add(expenseData);
                }
                transactionsTable.setItems(expenses);
            } catch (Exception e) {

                e.printStackTrace();
            }
            return ExpenseList;
        }
        /**
         * initializeTotalExpense - takes data from TRANSACTIONS table stored in the mySQL database.
         * then takes the amount under any expense category and adds them together.
         */
        public void initializeTotalExpense() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn
                        .prepareStatement("SELECT SUM(TRANSACTIONS.amount) AS TOTAL_EXPENSES " + "FROM MYAPP.TRANSACTIONS "
                                + "WHERE TRANSACTIONS.fromUser = ? " + "AND TRANSACTIONS.touser = 'NONE'");


                ps.setString(1, UserHolder.getInstance().getName());
                statement = conn.createStatement();
                ResultSet rs = ps.executeQuery();

                while (rs.next()) {
                    System.out.println(rs.getString("TOTAL_EXPENSES"));
                    this.expenseLabel.setText(rs.getString("TOTAL_EXPENSES"));
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        /**
         * initializeTotalIncome - takes data from TRANSACTION table stored in the mySQL database.
         * then takes the amount under any income category and adds them together.
         */
        public void initializeTotalIncome() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn
                        .prepareStatement("SELECT SUM(TRANSACTIONS.amount) AS TOTAL_INCOME " + "FROM MYAPP.TRANSACTIONS "
                                + "WHERE TRANSACTIONS.touser = ? " + "AND TRANSACTIONS.fromUser = 'NONE'");


                ps.setString(1, UserHolder.getInstance().getName());
                statement = conn.createStatement();
                ResultSet rs = ps.executeQuery();

                while (rs.next()) {
                    System.out.println(rs.getString("TOTAL_INCOME"));
                    this.incomeLabel.setText(rs.getString("TOTAL_INCOME"));
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /**
         * initializeAmountExpected - finds the amount of money owed by other user
         */
        public void initializeAmountExpected() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn.prepareStatement("SELECT touser, SUM(TRANSACTIONS.amount) AMOUNT_EXPECTED "
                        + "FROM MYAPP.TRANSACTIONS " + "WHERE fromuser != 'NONE' AND touser != 'NONE' "
                        + "AND fromUser = ? " + "GROUP BY touser");


                ps.setString(1, UserHolder.getInstance().getName());
                statement = conn.createStatement();
                ResultSet rs = ps.executeQuery();
                while (rs.next()) {
                    OwedData currentRow = new OwedData();
                    currentRow.setOwedUser(rs.getString("touser"));
                    currentRow.setOwedAmount(rs.getDouble("AMOUNT_EXPECTED"));
                    owedList.add(currentRow);
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        /**
         * initializeAmountOwe - finds the amount you own to other users
         */
        public void initializeAmountOwe() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn.prepareStatement("SELECT fromuser AS USER, SUM(TRANSACTIONS.amount) AMOUNT_OWED "
                        + "FROM MYAPP.TRANSACTIONS " + "WHERE fromuser != 'NONE' AND touser != 'NONE' " + "AND touser = ? "
                        + "GROUP BY fromuser");


                ps.setString(1, UserHolder.getInstance().getName());
                statement = conn.createStatement();
                ResultSet rs = ps.executeQuery();
                while (rs.next()) {
                    OweData currentRow = new OweData();
                    currentRow.setOweUser(rs.getString(1));
                    currentRow.setOweAmount(rs.getDouble("AMOUNT_OWED"));
                    this.oweList.add(currentRow);

                    System.out.println(rs.getString(1) + " " + rs.getDouble("AMOUNT_OWED"));
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /**
         * refresh - refreshes main page after pressing the refresh button.
         * @param event
         */
        public void refresh(ActionEvent event) {
            Stage stage = (Stage) ((Button) event.getSource()).getScene().getWindow();
            stage.close();
            try {
                BorderPane root = (BorderPane) FXMLLoader.load(getClass().getResource("MainPage.fxml"));

                Scene scene = new Scene(root, 1003, 745);
                scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());

                primaryStage.setScene(scene);
                primaryStage.show();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        /**
         * expensePieChart - Inserts data into the expense pie chart.
         */
        public void expensePieChart() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn.prepareStatement("SELECT subType, SUM(amount) AS sum_amount FROM MYAPP.DETAILS, 
    MYAPP.TRANSACTIONS "
                        + "WHERE TRANSACTIONS.detailId = DETAILS.detailId and toUser = 'NONE' and fromUser = ? GROUP by subType");
                ps.setString(1, UserHolder.getInstance().getName());
                ResultSet rs = ps.executeQuery();
                ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
                while(rs.next()) {

                        pieChartData.add(new PieChart.Data(rs.getString("subType"), rs.getDouble("sum_amount")));

                }

                expensePieChart.getData().addAll(pieChartData);



            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        /**
         * incomePieChart - Inserts data into the income pie chart.
         */
        public void incomePieChart() {
            try {
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/MYAPP", "root",
                        "password");
                PreparedStatement ps = conn.prepareStatement("SELECT subType, SUM(amount) AS sum_amount FROM MYAPP.DETAILS, 
    MYAPP.TRANSACTIONS "
                        + "WHERE TRANSACTIONS.detailId = DETAILS.detailId and fromUser = 'NONE' and toUser = ? GROUP by subType");
                ps.setString(1, UserHolder.getInstance().getName());
                ResultSet rs = ps.executeQuery();
```

```java
            ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
            while(rs.next()) {

                        pieChartData.add(new PieChart.Data(rs.getString("subType"), rs.getDouble("sum_amount")));

            }
            incomePieChart.getData().addAll(pieChartData);


        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    /**
     * incomeChartButton - Displays Income PieChart
     * @param event
     */
    public void incomeChartButton(ActionEvent event) {
        incomePieChart.setOpacity(1);
        expensePieChart.setOpacity(0);
    }
    /**
     * expenseChartButton  - Displays Expense PieChart
     * @param event
     */
    public void expenseChartButton(ActionEvent event) {
        incomePieChart.setOpacity(0);
        expensePieChart.setOpacity(1);
    }
}

```