

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590 018, Karnataka



“Project Title”

*Report submitted in partial fulfillment of the requirements for
the Curriculum of III Semester*

OOP with JAVA-BCS306A

In the faculty of Artificial Intelligence and Machine Learning

By

Dhruva R

1MP23AI014

Yashas Gowda N

1MP23AI059

BGS College of Engineering and Technology
Bangalore – 560086



Department of Artificial Intelligence and Machine Learning

||Jai Sri Gurudev ||

BGSKH Education Trust (R.) – A unit of Sri Adichunchanagiri Shikshana Trust(R.)

BGS College of Engineering and Technology (BGSCET)

(Approved by AICTE, New Delhi and Affiliated to VTU, Belagavi)

Adjacent to Mahalakshmi Metro Station, Mahalakshmiapuram, West of Chord Road, Bengaluru -560 086, Karnataka

Table of Contents

| | |
|--------------------------------|----|
| Introduction | 5 |
| Methodology | 6 |
| Implementation | 7 |
| Algorithm | 7 |
| Inner Working | 8 |
| Main Class | 11 |
| Controller Class | 12 |
| SessionManager Class | 29 |
| DatabaseManagement Class | 32 |
| HibernateUtil Class | 33 |
| TransactionObj Class | 34 |
| TransactionDAO Class | 36 |
| UserObj Class | 39 |
| UserDAO Class | 41 |
| Output | 43 |
| Login Page | 44 |
| Register Page | 44 |
| Home Page | 44 |
| Expense Transaction Page | 46 |
| Income Transaction Page | 46 |
| Expense Overview Page | 47 |

Introduction

The **Expense Tracker Application** is a Java-based software designed to help users track their income, expenses, and savings. The project was developed using JavaFX, a framework for building graphical user interfaces (GUIs) in Java, and utilizes object-oriented principles to manage the data related to transactions. The application provides a user-friendly interface for managing daily finances by allowing users to log both their income and expenses, categorize transactions, and view detailed reports such as pie charts that represent the distribution of spending and savings.

This expense tracker aims to address the common issue of financial management by providing a simple yet efficient solution for users to monitor and control their finances. It not only helps in tracking transactions but also allows for monthly summaries, making it easier for individuals to understand where their money is being spent and whether they are meeting their savings goals. Additionally, users can easily visualize their financial data with the help of interactive charts.

The application also includes a login system to authenticate users and store their data in a session, ensuring that users can access their individual financial records securely. This allows each user to manage their own set of transactions, ensuring privacy and organization. The system calculates and displays the total expenses, income, and savings in real-time, with the added feature of saving the data for future reference.

By utilizing this application, users can gain a clear insight into their financial habits, improve budgeting practices, and make more informed financial decisions. The visual representation of their financial activity in the form of charts and summaries ensures that the user remains engaged with their financial goals.

Methodology

The methodology for the development of the Expense Tracker is divided into multiple phases: planning, design, implementation, and testing.

1.Planning:

- Identifying the core features such as expense tracking, income management, transaction history, and savings calculation.
- Defining the user interface structure to ensure a seamless user experience.

2.Design:

- Designing the architecture based on a Model-View-Controller (MVC) pattern to separate concerns and enhance maintainability.
- Using JavaFX for the frontend to ensure a smooth graphical interface.
- Implementing Hibernate ORM for handling database interactions efficiently.

3.Implementation:

- Developing the backend to handle user interactions, transaction management, and pie chart updates.
- Developing the frontend to display the data visually and allow for easy user interaction.

4.Testing:

- Conducting unit tests and integration tests to ensure the functionality and stability of the application.
- Gathering user feedback to improve the interface and overall usability.

Implementation

The implementation of the Expense Tracker involves several key components:

1. Frontend:

- Developed using JavaFX, the user interface includes text fields for transaction input, pie charts for financial breakdowns, and tables to display transaction history.
- The frontend communicates with the backend to update and display real-time data about expenses, income, and savings.

2. Backend:

- The backend consists of multiple classes that handle database operations, transaction logic, and financial calculations.
- A DAO (Data Access Object) pattern is used to interact with the database to create, read, update, and delete transactions.
- The 'SessionManager' class maintains the current user session and ensures that all user-specific data is handled correctly.

3. Database:

- A relational database stores user information and transaction records. The backend interacts with the database using Hibernate ORM to ensure efficient querying and storage of data.

Algorithm

The following is the overall algorithm for the Expense Tracker:

1. Initialization Phase:

- Load the necessary libraries and establish a session for the current user.
- Fetch and display user data (transactions, categories, savings) on the UI.

2. Transaction Management:

- User inputs transaction details (type, amount, category).
- Validate input data and create a new transaction object.
- Save the transaction to the database and update session data.
- Refresh pie charts and transaction history.

3. Transaction Display:

- Fetch all transactions and categorize them by type (income or expense).
- Display categorized data on pie charts.
- Show the transaction history with relevant details.

4. Savings Calculation:

- Calculate the total income and expenses for the current user.
- Display the calculated savings as the difference between income and expenses.

5. User Session and Data Persistence:

- Ensure transactions are saved and retrieved from the database.
- Maintain the user session throughout the application lifecycle.

6. UI and Interaction:

- Handle user interactions like adding transactions and viewing data.
- Update the UI dynamically based on user input.

7. Error Handling and Validation:

- Validate inputs and display error messages for invalid data.
- Handle database and UI errors gracefully.

8. Application Exit:

- Save data to the database and close the application.

Inner Working

The inner working of the Expense Tracker revolves around the interaction between the frontend, backend, and database.

1. Frontend:

- JavaFX components such as PieCharts, TextFields, and TextFlow are used to create the user interface.
- Users can add transactions, view charts, and navigate the application using button clicks and other interactions.

2. Backend:

- The `TransactionDAO` class handles the CRUD operations on transaction data.
- The `SessionManager` class manages user login and session persistence.
- Business logic for calculating expenses, income, and savings is implemented in backend classes, which update the frontend dynamically.

3. Database:

- Hibernate ORM is used to interact with the database. It abstracts SQL queries and allows for object-relational mapping, making it easier to manage transactions in the database.
- The database schema includes tables for users and transactions, with relationships between them based on user ID.

4. Real-Time Updates:

- The application listens for user input events and updates the UI in real-time, ensuring the user sees their financial data immediately after any changes.

5. Data Flow:

- The data flow starts from the user input on the frontend, passes through the backend for validation and processing, and is then stored/retrieved from the database as necessary.
- The UI components are updated based on the data received from the backend to display real-time information such as pie charts and transaction history.

Project Code

The project has multiple java classes :

1. Controller.java

- Manages user interface (UI) interactions.
- Updates pie charts, transaction history, and user details.
- Handles scene switching and user input for transaction management.

2. DatabaseManagement.java

- Manages the database connection and CRUD operations.
- Ensures that all transaction data is saved and retrieved from the database.
- Handles user authentication and data integrity.

3. HibernateUtil.java

- Provides the Hibernate session factory for database interactions.
- Configures database connection properties for the Hibernate framework.
- Enables object-relational mapping for storing and retrieving data.

4. Main.java

- Entry point for the application, initializing the JavaFX application.
- Loads the primary scene and sets up the main application window.
- Starts the UI and links it with the controller logic.

5. SessionManager.java

- Manages the current user session and their transaction data.
- Holds user-specific data, such as user ID, name, and transaction details.
- Provides methods to access and update session data across the application.

6. TransactionDAO.java

- Handles database operations related to transactions.
- Implements methods to create, read, update, and delete transaction records.
- Interacts with the database to store transaction data securely.

7. TransactionObj.java

- Defines the transaction object model, including type, amount, category, and date.
- Acts as a data container for individual transactions.
- Used for storing and manipulating transaction-related data.

8. UserDAO.java

- Manages database operations related to user information.
- Provides methods for creating and retrieving user data.
- Ensures proper handling of user credentials and authentication.

9. UserObj.java

- Defines the user object model, storing user-specific details like ID and name.
- Serves as a data structure for managing user information.
- Used by the application to manage sessions and user-related data.

10. FXML Files

- Define the UI layout and structure for different scenes in the application.
- Link UI components to controller methods to handle user actions.
- Provide an easy-to-maintain, modular approach for UI design.

Main Class

```
package com.example;

import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

public class Main extends Application {

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

    @Override
    public void start(Stage primaryStage) throws IOException {
        try {
            Parent root =
FXMLLoader.load(getClass().getResource("/com/example/LoginScene.fxml"));
            Scene scene = new Scene(root);
            primaryStage.initStyle(StageStyle.TRANSPARENT);
            scene.setFill(Color.TRANSPARENT);
            scene.getStylesheets().add(getClass().getResource("/com/example/styles.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (IOException e) {
            e.printStackTrace();
            System.err.println("Error loading FXML file: " + e.getMessage());
        }
    }
}
```

Controller Class

```
package com.example;

import java.io.IOException;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

import javafx.animation.FadeTransition;
import javafx.animation.RotateTransition;
import javafx.animation.TranslateTransition;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.scene.text.TextFlow;
import javafx.stage.Stage;
import javafx.util.Duration;

public class Controller {

    private Stage stage;
    private Scene scene;
    private Parent root;
```

```

@FXML
private AnchorPane rootNode;

@FXML
private Rectangle clip;

@FXML
private Text welcomeUserText;

SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
//~~~~~INITAILIZE~~~~~
@FXML
public void initialize() {
    enterTranshistTF();
    enterInTranshistTF();
    enterSavTranshistTF();
}

int i = 0;

//~~~~~THREADS~~~~~

class setCurrUserThread extends Thread {
    public void run() {
        SessionManager.setCurrUser(user);
    }
}

class setExpTransactionListThread extends Thread {
    public void run() {
        SessionManager.setExpTransactionList();
    }
}

class setInTransactionListThread extends Thread {
    public void run() {
        SessionManager.setInTransactionList();
    }
}

class setTransactionListThread extends Thread {
    public void run() {
        SessionManager.setTransactionList();
    }
}

class setSavingsTransactionListThread extends Thread {
    public void run() {
        SessionManager.setSavingsTransactionList();
    }
}

```

```

    }

    class setTotalOverviewThreadThread extends Thread {
        public void run() {
            SessionManager.setTotalOverview();
        }
    }

    //TITLE BAR BUTTONS

    @FXML
    private void closeWindow(MouseEvent event) {
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.close();
    }

    @FXML
    private void minimizeWindow(MouseEvent event) {
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setIconified(true);
    }

    @FXML
    private void maximizeWindow(MouseEvent event) {
    }

    //~~~~~LOGIN SCENE~~~~~

    @FXML
    private Text invalidLogin;

    @FXML
    private TextField lognamein;

    @FXML
    private TextField logpassin;

    @FXML
    void switchRegisterScene(MouseEvent event) {
        try {
            Parent root =
FXMLLoader.load(getClass().getResource("/com/example/RegisterScene.fxml"));
            stage = (Stage)((Node) event.getSource()).getScene().getWindow();
            scene = new Scene(root);
            scene.setFill(Color.TRANSPARENT);
            stage.setScene(scene);
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

    }

    UserObj user = new UserObj();

    @FXML
    void userLogin(MouseEvent event) throws SQLException, InterruptedException {

        UserDAO userDAO = new UserDAO(sessionFactory);
        user = userDAO.getUser(String.valueOf(lognamein.getText()));
        if(user==null)
        {
            System.out.println("INVALID USER");
            invalidLogin.setText("Check Name!!!");
            invalidLogin.setVisible(true);
        }
        else if(user.password.equals(String.valueOf(logpassin.getText())))
        {
            System.out.println("LOGIN SUCCESS");
            invalidLogin.setText("Logging in!!!");
            invalidLogin.setFill(Color.web("#ecedec"));
            invalidLogin.setVisible(true);

            SessionManager.setCurrUser(user);
            SessionManager.setExpTransactionList();
            SessionManager.setInTransactionList();
            SessionManager.setTransactionList();
            SessionManager.setSavingsTransactionList();
            SessionManager.setTotalOverview();
            LocalDate today = LocalDate.now();
            if(today.getDayOfMonth()==today.lengthOfMonth())
                SessionManager.updateSavingsTransactionList();

            try {
                FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/example/HomeScene.fxml"));
                Parent root = loader.load();
                Controller homeController = loader.getController();
                homeController.initializePieChart();
                homeController.initializehomeText();
                homeController.welcomeUserText.setText("Welcome,
"+String.valueOf(lognamein.getText()));
                stage = (Stage)((Node) event.getSource()).getScene().getWindow();
                scene = new Scene(root);
                scene.setFill(Color.TRANSPARENT);
                stage.setScene(scene);
                stage.centerOnScreen();
                stage.show();

            } catch (IOException e) {

```

```

        e.printStackTrace();
    }
}
else
{
    System.out.println("INCORRECT PASSWORD");
    invalidLogin.setText("Check Password!!!");
    invalidLogin.setVisible(true);
}
}

//~~~~~REGISTER SCENE~~~~~

@FXML
private Text invalidReg;

@FXML
private TextField regnamein;

@FXML
private TextField regpassin;

@FXML
private TextField regphnoin;

@FXML
void regNewUser(MouseEvent event) throws SQLException {
    UserDao userDao = new UserDao(sessionFactory);
    UserObj newUser = userDao.getUser(String.valueOf(regnamein.getText()));
    if(newUser==null)
    {
        newUser=new UserObj();
        newUser.setName(String.valueOf(regnamein.getText()));
        newUser.setPassword(String.valueOf(regpassin.getText()));
        newUser.setPhno(String.valueOf(regphnoin.getText()));
        userDao.createUser(newUser);
        sessionFactory.close();
        invalidReg.setText("User registered");
        invalidReg.setVisible(true);
        System.out.println("USER REGISTERED IN DATABASE");
    }
    else
    {
        invalidReg.setText("User already registered");
        invalidReg.setVisible(true);
        System.out.println("USER ALREADY REGISTERED IN DATABASE");
    }
}
}

```

```

@FXML
void switchLoginScene(MouseEvent event) {
    try {
        Parent root =
FXMLLoader.load(getClass().getResource("/com/example/LoginScene.fxml"));
        stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        scene.setFill(Color.TRANSPARENT);
        stage.setScene(scene);
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

//~~~~~HOME SCENE~~~~~

```

```

//Burger Menu Toggle

```

```

@FXML
private VBox burgerMenu;

```

```

@FXML
private Rectangle line1;

```

```

@FXML
private Rectangle line2;

```

```

@FXML
private Rectangle line3;

```

```

@FXML
private AnchorPane SideBar;

```

```

private boolean isMenuOpen = true;

```

```

@FXML
private Text homeExp;

```

```

@FXML
private Text homeIn;

```

```

@FXML
private Text homeSav;

```

```

@FXML
void toggleSidebar(MouseEvent event) {
    RotateTransition rotateL1 = new RotateTransition(Duration.seconds(0.3),line1);

```

```

TranslateTransition translateL1 = new TranslateTransition(Duration.seconds(0.3),line1);
RotateTransition rotateL3 = new RotateTransition(Duration.seconds(0.3),line3);
TranslateTransition translateL3 = new TranslateTransition(Duration.seconds(0.3),line3);
FadeTransition fadeL2 = new FadeTransition(Duration.seconds(0.1),line2);
TranslateTransition translateSB = new TranslateTransition(Duration.seconds(0.3),
Sidebar);
if(isMenuOpen)
{
    //Line 1 Transitions Settings
    rotateL1.setByAngle(45);
    translateL1.setByY(line1.getHeight()*2);
    //Line 2 Transitions Settings
    rotateL3.setByAngle(-45);
    translateL3.setByY(-line3.getHeight()*2);
    //Line 2 Transitions Settings
    fadeL2.setFromValue(1);
    fadeL2.setToValue(0);
    //Sidebar Transitions Settings
    translateSB.setByX(300);

    rotateL1.play();
    translateL1.play();
    fadeL2.play();
    rotateL3.play();
    translateL3.play();
    translateSB.play();

    isMenuOpen=false;
}
else
{
    //Line 1 Transitions
    rotateL1.setByAngle(-45);
    translateL1.setByY(-line1.getHeight()*2);
    //Line 2 Transitions
    rotateL3.setByAngle(45);
    translateL3.setByY(line3.getHeight()*2);
    //Line 2 Transitions
    fadeL2.setFromValue(0);
    fadeL2.setToValue(1);
    //Sidebar Transitions Settings
    translateSB.setByX(-300);

    rotateL1.play();
    translateL1.play();
    fadeL2.play();
    rotateL3.play();
    translateL3.play();
    translateSB.play();
}

```

```
        isMenuOpen = true;
    }
}
```

```
//Sidebar
```

```
@FXML
```

```
void switchColorBlack(MouseEvent event) {
    Node sourceNode = (Node) event.getSource();
    if (sourceNode instanceof Text)
    {
        Text t = (Text) sourceNode;
        t.setFill(Color.web("#ecedec"));
    }
    else if(isMenuOpen)
    {
        line1.setFill(Color.web("#ECEDEC"));
        line2.setFill(Color.web("#ECEDEC"));
        line3.setFill(Color.web("#ECEDEC"));

    }
    else
    {
        line1.setFill(Color.web("#ecedec"));
        line2.setFill(Color.web("#ecedec"));
        line3.setFill(Color.web("#ecedec"));
    }
}
```

```
@FXML
```

```
void switchColorRed(MouseEvent event) {
    Node sourceNode = (Node) event.getSource();
    if (sourceNode instanceof Text)
    {
        Text t = (Text) sourceNode;
        t.setFill(Color.web("#888888"));
    }
    else
    {
        line1.setFill(Color.web("#888888"));
        line2.setFill(Color.web("#888888"));
        line3.setFill(Color.web("#888888"));
    }
}
```

```
@FXML
```

```
void switchTransactionScene(MouseEvent event)
{
    try {
```

```

        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/example/ExpenseTransactionScene.fxml"));
        Parent root = loader.load();
        Controller homeController = loader.getController();
        homeController.initializePieChart();
        stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        scene.setFill(Color.TRANSPARENT);
        stage.setScene(scene);
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

@FXML
void switchExpenseOverviewScene(MouseEvent event) {
    try {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/example/ExpenseOverviewScene.fxml"));
        Parent root = loader.load();
        Controller homeController = loader.getController();
        homeController.initializeSavPieChart();
        stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        scene.setFill(Color.TRANSPARENT);
        stage.setScene(scene);
        stage.show();
        enterSavTranshistTF();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

@FXML
void switchIncomeTransactionScene(MouseEvent event) {
    try {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/example/IncomeTransactionScene.fxml"));
        Parent root = loader.load();
        Controller homeController = loader.getController();
        homeController.initializeincomePieChart();
        stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        scene = new Scene(root);
        scene.setFill(Color.TRANSPARENT);
        stage.setScene(scene);
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

    }

    @FXML
    void switchHomeScene(MouseEvent event) {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/example/HomeScene.fxml"));
            Parent root = loader.load();
            Controller homeController = loader.getController();
            homeController.initializePieChart();
            homeController.initializehomeText();
            homeController.welcomeUserText.setText("Welcome,
"+SessionManager.getCurrUser().name);
            stage = (Stage)((Node) event.getSource()).getScene().getWindow();
            scene = new Scene(root);
            scene.setFill(Color.TRANSPARENT);
            stage.setScene(scene);
            stage.centerOnScreen();
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```
//HOME MONTHLY OVERVIEW TEXT
```

```

void initializehomeText() {
    homeExp.setText(" Rs."+String.valueOf(SessionManager.totalExp));
    homeIn.setText(" Rs."+String.valueOf(SessionManager.totalIn));
    homeSav.setText(" Rs."+String.valueOf(SessionManager.totalIn-
SessionManager.totalExp));
}

```

```

//~~~~~EXPENSES
OVERVIEW~~~~~

```

```
//Pie Chart
```

```

@FXML
private PieChart homePieChart;

public void initializePieChart() {
    List<TransactionObj> transactions = SessionManager.getExpTransactionList();
    Map<String, Double> categoryTotals = new HashMap<>();

    // Accumulate amounts by category
    for (TransactionObj transaction : transactions) {
        String category = transaction.getCategory();
        double amount = transaction.getAmount();

```

```

        // Sum the amounts for each category {categoryTotals.put(category,
categoryTotals.getDefault(category, 0.0) + amount);}
        categoryTotals.put(category, categoryTotals.getDefault(category, 0.0) + amount);
    }

    // Create the PieChart data
    ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
    for (Map.Entry<String, Double> entry : categoryTotals.entrySet()) {
        pieChartData.add(new PieChart.Data(entry.getKey(), entry.getValue()));
    }

    // Set the data to the PieChart
    homePieChart.setData(pieChartData);
    homePieChart.setLabelsVisible(true);
}

@FXML
private Text textPC;

@FXML
public void switchHomePieChart(MouseEvent event) {
    if(i==0)
    {
        List<TransactionObj> transactions = SessionManager.getInTransactionList();
        Map<String, Double> categoryTotals = new HashMap<>();

        // Accumulate amounts by category
        for (TransactionObj transaction : transactions) {
            String category = transaction.getCategory();
            double amount = transaction.getAmount();

            // Sum the amounts for each category {categoryTotals.put(category,
categoryTotals.getDefault(category, 0.0) + amount);}
            categoryTotals.put(category, categoryTotals.getDefault(category, 0.0) + amount);
            i=1;
        }

        // Create the PieChart data
        ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
        for (Map.Entry<String, Double> entry : categoryTotals.entrySet()) {
            pieChartData.add(new PieChart.Data(entry.getKey(), entry.getValue()));
        }

        // Set the data to the PieChart
        homePieChart.setData(pieChartData);
        homePieChart.setLabelsVisible(true);
        textPC.setText("Income Chart");
    }
}
else

```

```

        {
            initializePieChart();
            textPC.setText("Expense Chart");
            i=0;
        }
    }

//~~~~~TRANSACTION SCENE~~~~~

//Insert New Transaction

@FXML
private TextField expAmtin;

@FXML
private TextField expCategoryin;

@FXML
void insertTransaction(MouseEvent event) throws InterruptedException {
    UserObj user = SessionManager.getCurrUser();
    TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
    TransactionObj newTransaction = new TransactionObj();
    newTransaction.setType(0);
    newTransaction.setCategory(String.valueOf(expCategoryin.getText()));
    newTransaction.setAmount(Integer.parseInt(expAmtin.getText()));
    newTransaction.setUid(user.id);
    newTransaction.setTransdate(new Date());
    transactionDAO.createTransaction(newTransaction);
    SessionManager.setExpTransactionList();
    SessionManager.setSavingsTransactionList();
    expCategoryin.setText("");
    expAmtin.setText("");
    initializePieChart();
    enterTranshistTF();
}

//Transaction History

@FXML
private TextFlow transhistTF;

void enterTranshistTF() {
    if(transhistTF==null)
    {
        System.err.println("Expense Text Flow is null");
        return;
    }
    transhistTF.getChildren().clear();

```

```

List<TransactionObj> transactions = SessionManager.getExpTransactionList();

for (int i = transactions.size() - 1; i >= 0; i--) {
    TransactionObj transaction = transactions.get(i);

    Text heading = new Text("Transaction ID: ");
    heading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
    heading.setFill(Color.web("#ecedec"));

    Text content = new Text(String.valueOf(transaction.getId()) + "\n");
    content.setFont(Font.font("Arial", 12));
    content.setFill(Color.web("#ecedec"));

    Text dateHeading = new Text("Transaction Date: ");
    dateHeading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
    dateHeading.setFill(Color.web("#ecedec"));

    Text dateContent = new Text(String.valueOf(transaction.getTransdate()) + "\n");
    dateContent.setFont(Font.font("Arial", 12));
    dateContent.setFill(Color.web("#ecedec"));

    Text categoryHeading = new Text("Category: ");
    categoryHeading.setFont(Font.font("Arial", FontWeight.BOLD, 14));
    categoryHeading.setFill(Color.web("#ecedec"));

    Text categoryContent = new Text(transaction.getCategory() + "\n");
    categoryContent.setFont(Font.font("Arial", 12));
    categoryContent.setFill(Color.web("#ecedec"));

    Text amountHeading = new Text("Amount: ");
    amountHeading.setFont(Font.font("Arial", FontWeight.BOLD, 14));
    amountHeading.setFill(Color.web("#ecedec"));

    Text amountContent = new Text(String.valueOf(transaction.getAmount()) + "\n");
    amountContent.setFont(Font.font("Arial", 12));
    amountContent.setFill(Color.web("#ecedec"));

    Line line = new Line(0, 0, 480, 0);
    line.setStrokeWidth(2);
    line.setStroke(Color.web("#ecedec"));

    Text ln = new Text("\n\n");

    transhistTF.getChildren().addAll(heading, content, dateHeading, dateContent,
    categoryHeading, categoryContent, amountHeading, amountContent, line, ln);
}
}

```

```

private TextFlow incomeHistTextFlow;

@FXML
private TextField incomeAmt;

@FXML
private TextField incomeCategory;

@FXML
private PieChart incomePiechart;

@FXML
void addIncomeAmt(MouseEvent event) {
    UserObj user = SessionManager.getCurrUser();
    TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
    TransactionObj newTransaction = new TransactionObj();
    newTransaction.setType(1);
    newTransaction.setCategory(String.valueOf(incomeCategory.getText()));
    newTransaction.setAmount(Integer.parseInt(incomeAmt.getText()));
    newTransaction.setUid(user.id);
    newTransaction.setTransdate(new Date());
    transactionDAO.createTransaction(newTransaction);
    SessionManager.setInTransactionList();
    incomeCategory.setText("");
    incomeAmt.setText("");
    SessionManager.setSavingsTransactionList();
    new setTotalOverviewThread().join();
    enterInTranshistTF();
    initializeincomePieChart();
}

void enterInTranshistTF() {
    if(incomeHistTextFlow == null)
    {
        System.err.println("Savings Text Flow is null");
        return;
    }

    incomeHistTextFlow.getChildren().clear();
    List<TransactionObj> transactions = SessionManager.getInTransactionList();

    for (int i = transactions.size() - 1; i >= 0; i--) {
        TransactionObj transaction = transactions.get(i);

        Text heading = new Text("Transaction ID: ");
        heading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
        heading.setFill(Color.web("#ecedec"));

```

```

        Text content = new Text(String.valueOf(transaction.getId()) + "\n");

```

```

content.setFont(Font.font("Arial", 12));
content.setFill(Color.web("#ecedec"));

Text dateHeading = new Text("Transaction Date: ");
dateHeading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
dateHeading.setFill(Color.web("#ecedec"));

Text dateContent = new Text(String.valueOf(transaction.getTransdate()) + "\n");
dateContent.setFont(Font.font("Arial", 12));
dateContent.setFill(Color.web("#ecedec"));

Text categoryHeading = new Text("Category: ");
categoryHeading.setFont(Font.font("Arial", FontWeight.BOLD, 14));
categoryHeading.setFill(Color.web("#ecedec"));

Text categoryContent = new Text(transaction.getCategory() + "\n");
categoryContent.setFont(Font.font("Arial", 12));
categoryContent.setFill(Color.web("#ecedec"));

Text amountHeading = new Text("Amount: ");
amountHeading.setFont(Font.font("Arial", FontWeight.BOLD, 14));
amountHeading.setFill(Color.web("#ecedec"));

Text amountContent = new Text(String.valueOf(transaction.getAmount()) + "\n");
amountContent.setFont(Font.font("Arial", 12));
amountContent.setFill(Color.web("#ecedec"));

Line line = new Line(0, 0, 480, 0);
line.setStrokeWidth(2);
line.setStroke(Color.web("#ecedec"));

Text ln = new Text("\n\n");

incomeHistTextFlow.getChildren().addAll(heading, content, dateHeading,
dateContent, categoryHeading, categoryContent, amountHeading, amountContent, line, ln);
}
}

@FXML
void initializeIncomePieChart() {
    List<TransactionObj> transactions = SessionManager.getInTransactionList();
    Map<String, Double> categoryTotals = new HashMap<>();

    // Accumulate amounts by category
    for (TransactionObj transaction : transactions) {
        String category = transaction.getCategory();
        double amount = transaction.getAmount();

        // Sum the amounts for each category {categoryTotals.put(category,
categoryTotals.getOrDefault(category, 0.0) + amount);}

```

```

        categoryTotals.put(category, categoryTotals.getDefault(category, 0.0) + amount);
    }

    // Create the PieChart data
    ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
    for (Map.Entry<String, Double> entry : categoryTotals.entrySet()) {
        pieChartData.add(new PieChart.Data(entry.getKey(), entry.getValue()));
    }

    // Set the data to the PieChart
    incomePiechart.setData(pieChartData);
    incomePiechart.setLabelsVisible(true);
}

```

```
//~~~~~EXPENSE OVERVIEW~~~~~
```

```
@FXML
private TextFlow savingTF;
```

```
@FXML
void enterSavTranshistTF() {
    if(savingTF == null)
    {
        System.err.println("Income Text Flow is null");
        return;
    }
}

```

```
savingTF.getChildren().clear();
List<TransactionObj> transactions = SessionManager.getSavingsTransactionList();

```

```
for (TransactionObj transaction : transactions)
{
    Text heading = new Text("Transaction ID: ");
    heading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
    heading.setFill(Color.web("#ecedec"));

    Text content = new Text(String.valueOf(transaction.getId()) + "\n");
    content.setFont(Font.font("Arial", 12));
    content.setFill(Color.web("#ecedec"));

    Text dateHeading = new Text("Transaction Date: ");
    dateHeading.setFont(Font.font("Arial", FontWeight.BOLD, 16));
    dateHeading.setFill(Color.web("#ecedec"));

    Text dateContent = new Text(String.valueOf(transaction.getTransdate()) + "\n");
    dateContent.setFont(Font.font("Arial", 12));
    dateContent.setFill(Color.web("#ecedec"));
}

```

```
Text categoryHeading = new Text("Month: ");
```

```

        categoryHeading.setFill(Color.web("#ecedec"));

        Text categoryContent = new Text(transaction.getCategory() + "\n");
        categoryContent.setFont(Font.font("Arial", 12));
        categoryContent.setFill(Color.web("#ecedec"));

        Text amountHeading = new Text("Amount: ");
        amountHeading.setFont(Font.font("Arial", FontWeight.BOLD, 14));
        amountHeading.setFill(Color.web("#ecedec"));

        Text amountContent = new Text(String.valueOf(transaction.getAmount()) + "\n");
        amountContent.setFont(Font.font("Arial", 12));
        amountContent.setFill(Color.web("#ecedec"));

        Line line = new Line(0, 0, 480, 0);
        line.setStrokeWidth(2);
        line.setStroke(Color.web("#ecedec"));

        Text ln = new Text("\n\n");

        savingTF.getChildren().addAll(heading, content, dateHeading, dateContent,
        categoryHeading, categoryContent, amountHeading, amountContent, line, ln);
    }
}

@FXML
private PieChart expOverviewPiechart;

@FXML
void initializeSavPieChart() {
    int[] total = SessionManager.getTotalOverview();
    Map<String, Double> categoryTotals = new HashMap<>();

    categoryTotals.put("Expense", (double) total[0]);
    categoryTotals.put("Savings", (double) (total[1] - total[0]));

    System.out.println("\n\n\nEXPENSE : "+total[0]+" SAVINGS : "+(total[1]-
    total[0])+"\n\n\n");

    ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList();
    for (Map.Entry<String, Double> entry : categoryTotals.entrySet()) {
        pieChartData.add(new PieChart.Data(entry.getKey(), entry.getValue()));
    }

    // Set the data to the PieChart
    expOverviewPiechart.setData(pieChartData);
    expOverviewPiechart.setLabelsVisible(true);
}

```

SessionManager Class

```
package com.example;

import java.time.LocalDate;
import java.time.format.TextStyle;
import java.util.Date;
import java.util.List;
import java.util.Locale;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class SessionManager {
    private static UserObj currUser;

    private static List<TransactionObj> expTransactions;
    private static List<TransactionObj> inTransactions;
    private static List<TransactionObj> Transactions;
    private static List<TransactionObj> SavTransactions;

    public static int totalExp=0,totalIn=0;

    public static void setCurrUser(UserObj user) {
        currUser = user;
    }

    public static UserObj getCurrUser() {
        return currUser;
    }

    public static void setExpTransactionList()
    {
        SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
        TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
        expTransactions = transactionDAO.getTransactions(currUser.id,0);
    }

    public static List<TransactionObj> getExpTransactionList()
    {
        return expTransactions;
    }

    public static void setInTransactionList() {
        SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
```

```

        TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
        inTransactions = transactionDAO.getTransactions(currUser.id,1);
    }

    public static List<TransactionObj> getInTransactionList() {
        return inTransactions;
    }

    public static void setTransactionList() {
        SessionFactory sessionFactory = new
        Configuration().configure().buildSessionFactory();
        TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
        Transactions = transactionDAO.getTransactions(currUser.id,3);
    }

    public static List<TransactionObj> getTransactionList()
    {
        return Transactions;
    }

    public static void updateSavingsTransactionList()
    {

        LocalDate today = LocalDate.now();

        int totalSave = totalIn-totalExp;

        SessionFactory sessionFactory = new
        Configuration().configure().buildSessionFactory();
        TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
        TransactionObj newTransaction = new TransactionObj();
        newTransaction.setType(2);
        String monthName = today.getMonth().getDisplayName(TextStyle.FULL,
        Locale.ENGLISH);
        int year = today.getYear();
        String Category = monthName+" "+year;
        newTransaction.setCategory(Category);
        newTransaction.setAmount(totalSave);
        newTransaction.setUid(currUser.id);
        newTransaction.setTransdate(new Date());
        transactionDAO.createTransaction(newTransaction);

        setSavingsTransactionList();
    }

    public static void setSavingsTransactionList() {
        SessionFactory sessionFactory = new
        Configuration().configure().buildSessionFactory();
        TransactionDAO transactionDAO = new TransactionDAO(sessionFactory);
        SavTransactions = transactionDAO.getTransactions(currUser.id,2);

```

```
}

public static List<TransactionObj> getSavingsTransactionList() {
    return SavTransactions;
}

public static void setTotalOverview() {
    for(TransactionObj x : expTransactions)
        totalExp+=x.getAmount();

    for(TransactionObj x : inTransactions)
        totalIn+=x.getAmount();
}

public static int[] getTotalOverview() {
    return new int[]{totalExp,totalIn};
}
}
```

DatabaseManagement Class

```
package com.example;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseManagement {

    public Connection con;

    public void establishConnection()
    {
        try {
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/expense_tracker",
"root", "root");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void disconnectConnection() {
        if (con != null) {
            try {
                con.close();
                System.out.println("Connection Closed Successfully");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("Connection was not established");
        }
    }
}
```

HibernateUtil Class

```
package com.example;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

TransactionObj Class

```
package com.example;

import java.util.Date;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "transactions")
public class TransactionObj {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private int uid;
    private String category;
    private int type;
    private Date transdate;
    private int amount;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUid() {
        return uid;
    }

    public void setUid(int uid) {
        this.uid = uid;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }
}
```

```
    }

    public int getType() {
        return type;
    }

    public void setType(int type) {
        this.type = type;
    }

    public Date getTransdate() {
        return transdate;
    }

    public void setTransdate(Date transdate) {
        this.transdate = transdate;
    }

    public int getAmount() {
        return amount;
    }

    public void setAmount(int amount) {
        this.amount = amount;
    }
}
```

TransactionDAO Class

```
package com.example;

import java.util.List;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

public class TransactionDAO {
    private final SessionFactory sessionFactory;

    public TransactionDAO(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    // Create
    public void createTransaction(TransactionObj transaction) {
        Transaction tx = null;
        try (Session session = sessionFactory.openSession()) {
            tx = session.beginTransaction();
            session.persist(transaction);
            tx.commit();
        } catch (Exception e) {
            if (tx != null) {
                tx.rollback();
            }
            e.printStackTrace();
        }
    }

    // Read
    public TransactionObj getTransaction(int id) {
        try (Session session = sessionFactory.openSession()) {
            return session.get(TransactionObj.class, id);
        }
    }

    public List<TransactionObj> getAllTransactions() {
        try (Session session = sessionFactory.openSession()) {
            Query<TransactionObj> query = session.createQuery("FROM TransactionObj",
TransactionObj.class);
            return query.getResultList();
        }
    }
}
```

```

// Update
public void updateTransaction(TransactionObj transaction) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        session.merge(transaction);
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
        }
        e.printStackTrace();
    }
}

// Delete
public void deleteTransaction(int id) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        TransactionObj transaction = session.get(TransactionObj.class, id);
        if (transaction != null) {
            session.remove(transaction); // Use remove instead of delete
        }
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
        }
        e.printStackTrace();
    }
}

public List<TransactionObj> getTransactions(int uid,int type) {
    try (Session session = sessionFactory.openSession()) {
        if(type==3)
        {
            Query<TransactionObj> query = session.createQuery(
                "FROM TransactionObj WHERE uid = :uid",
                TransactionObj.class
            );
            query.setParameter("uid", uid);

            return query.getResultList();
        }
        else
        {
            Query<TransactionObj> query = session.createQuery(
                "FROM TransactionObj WHERE uid = :uid AND type = :type",

```

```
        TransactionObj.class
    );
    query.setParameter("type", type);
    query.setParameter("uid", uid);

    return query.getResultList();
}
}
}
```

UserObj Class

```
package com.example;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "users")
public class UserObj {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    int id;
    String name;
    String password;
    String phno;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPhno() {
        return phno;
    }
}
```

```
    }  
  
    public void setPhno(String phno) {  
        this.phno = phno;  
    }  
}
```

UserDAO Class

```
package com.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;

public class UserDAO {

    private final SessionFactory sessionFactory;

    public UserDAO(SessionFactory sessionFactory)
    {
        this.sessionFactory=sessionFactory;
    }

    public void createUser(UserObj user)
    {
        Transaction tx=null;
        try(Session session=sessionFactory.openSession())
        {
            tx=session.beginTransaction();
            session.persist(user);
            tx.commit();
        }
        catch(Exception e)
        {
            if(tx!=null)
            {
                tx.rollback();
            }
            e.printStackTrace();
        }
    }

    public UserObj getUser(String name) {
        try (Session session = sessionFactory.openSession()) {
            String hql = "FROM UserObj WHERE name = :name";
            return session.createQuery(hql, UserObj.class)
                .setParameter("name", name)
                .uniqueResult();
        }
    }

    public void updateUser(UserObj user)
```

```

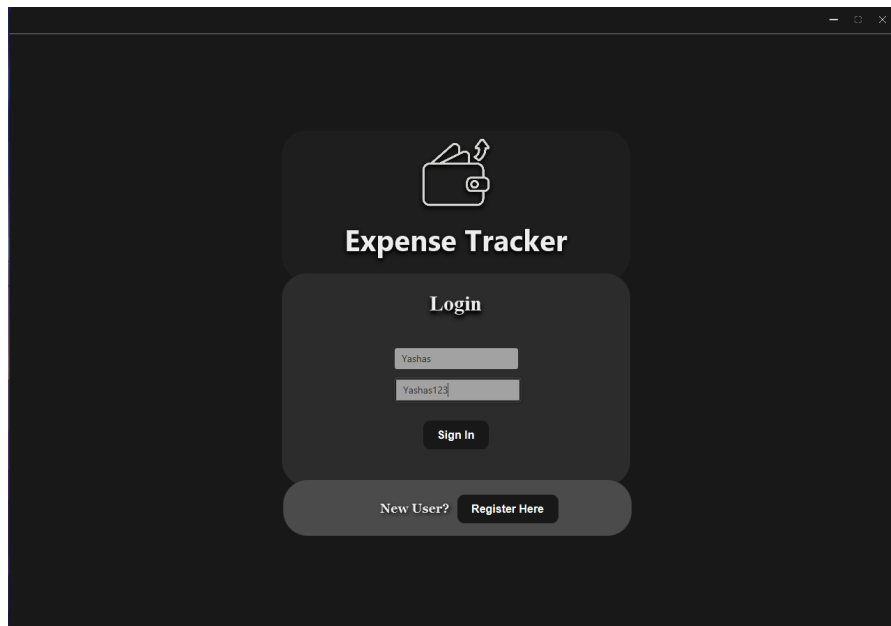
{
    Transaction tx = null;
    try(Session session=sessionFactory.openSession())
    {
        tx=session.beginTransaction();
        session.merge(user);
        tx.commit();
    }
    catch (Exception e)
    {
        if (tx != null)
        {
            tx.rollback();
        }
        e.printStackTrace();
    }
}

public void deleteUser(String name)
{
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        UserObj user = session.get(UserObj.class, name);
        if (user != null) {
            session.remove(user); // Use remove instead of delete
        }
        tx.commit();
    } catch (Exception e) {
        if (tx != null) {
            tx.rollback();
        }
        e.printStackTrace();
    }
}
}

```

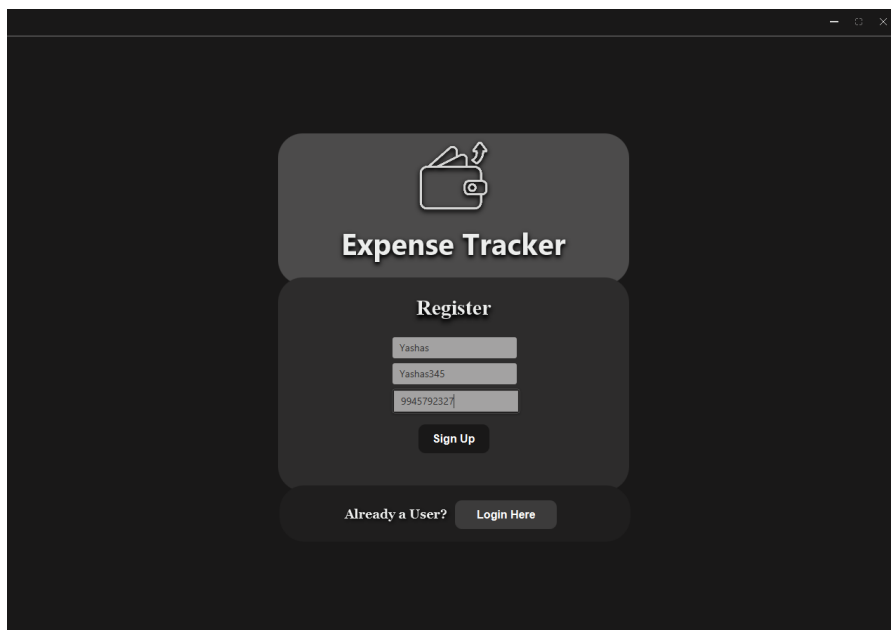
Output

Login Page



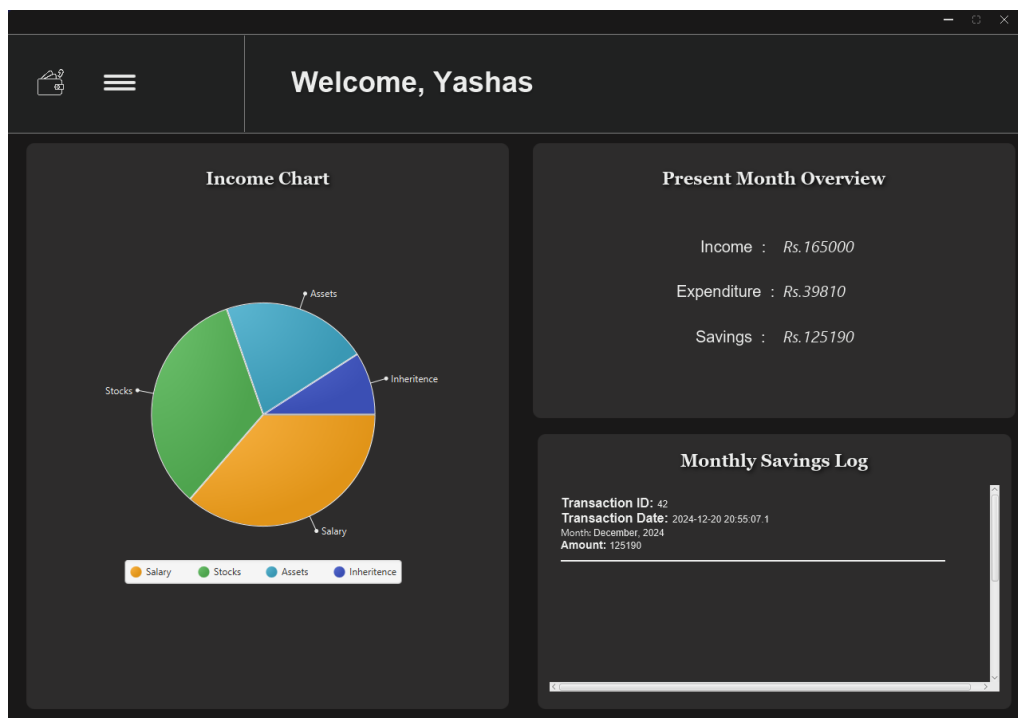
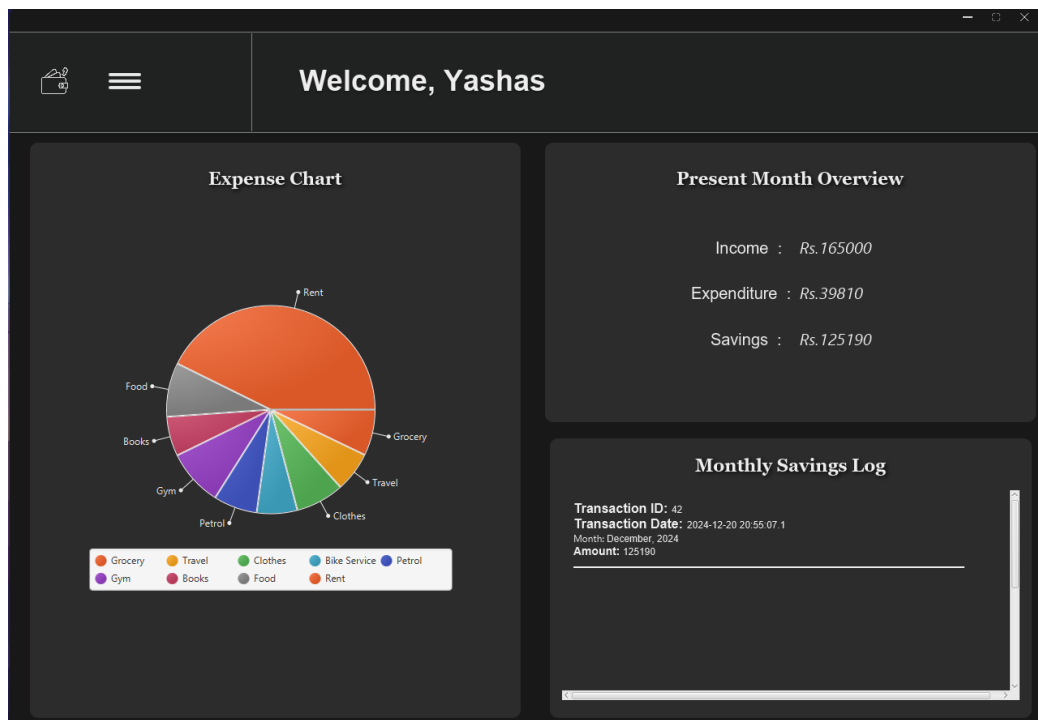
A screenshot of a web browser displaying the login page for an "Expense Tracker" application. The page has a dark background. At the top, there is a wallet icon with a dollar sign and the text "Expense Tracker". Below this is a "Login" section with two input fields: the first contains "Yashas" and the second contains "Yashas123". A "Sign In" button is positioned below the password field. At the bottom of the login section, there is a link "New User?" followed by a "Register Here" button.

Register Page

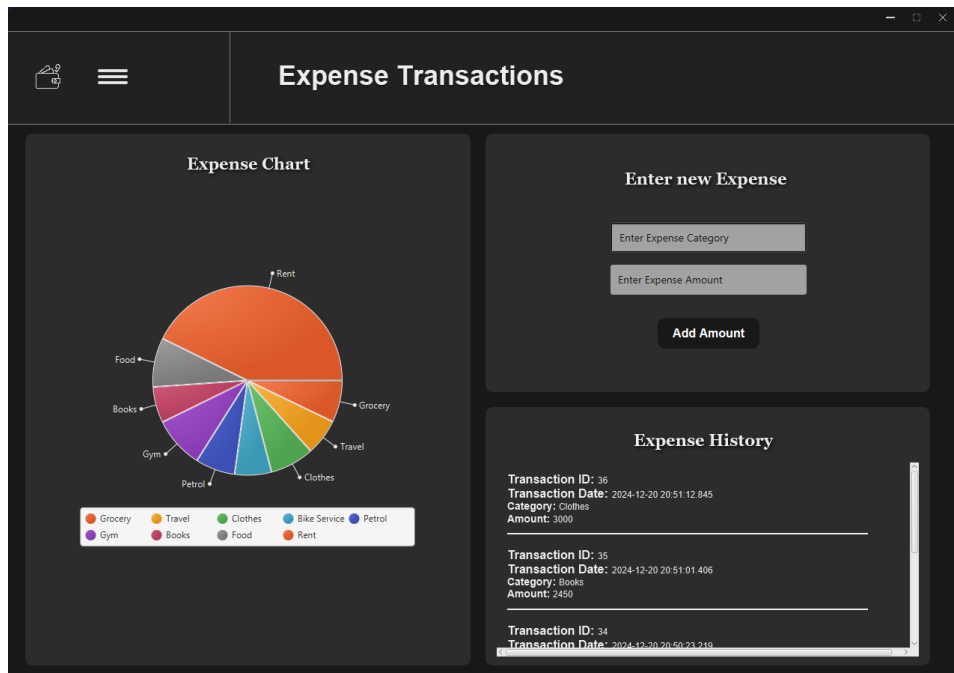


A screenshot of a web browser displaying the register page for an "Expense Tracker" application. The page has a dark background. At the top, there is a wallet icon with a dollar sign and the text "Expense Tracker". Below this is a "Register" section with three input fields: the first contains "Yashas", the second contains "Yashas345", and the third contains "9945792327". A "Sign Up" button is positioned below the phone number field. At the bottom of the register section, there is a link "Already a User?" followed by a "Login Here" button.

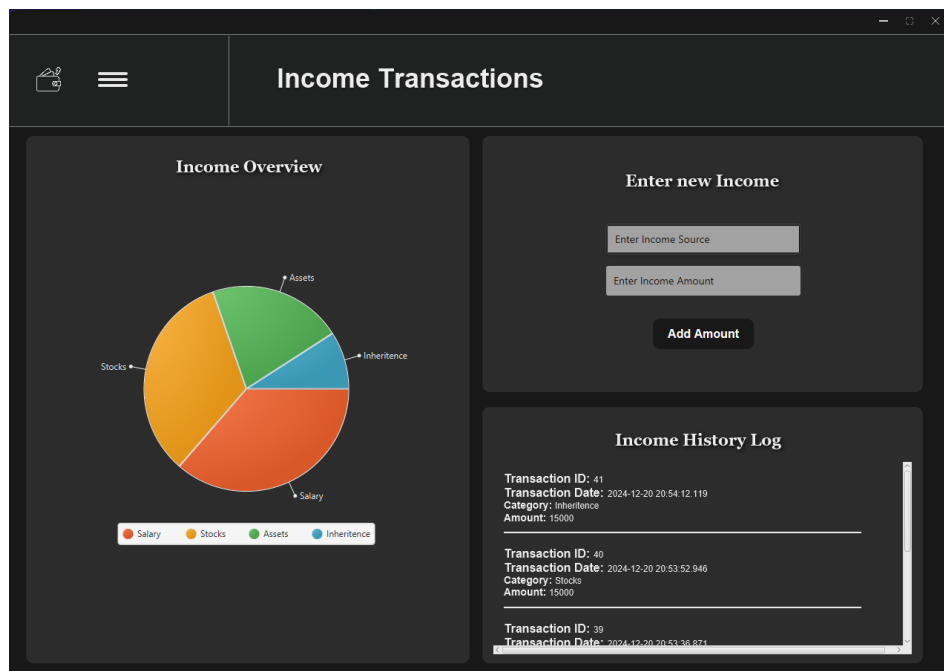
Home Page



Expense Transaction Page



Income Transaction Page



Expense Overview Page

