**[This is system is built to manage the details of customer accounts, transactions, balance, statement. It provides a simpler alternative to conventional banking system as the customer is not required to be at the banking to complete any transaction.]**

[Online Banking]



**[ Franck Davy Tchienkoua ]**

# Online Banking System

Major：Computer Science & Artificial Intelligence

Name：Franck Davy Tchienkoua

Student Number：19511160012

Due Date：28/10/2021

# Wenzhou University

# ABSTRACT

This project **“Online Banking System”** provides us a simple interface for maintenance of a customer’s account at a bank. It can be used by bank institutions to provide a convenient channel for their customers to complete simple transactions.

Online Banking is one of the most important financial activities which will be carried out by any person who holds a bank account. There are various activities that can be carried out once you log in to your bank account. Once a user logs in he or she can check the bank balance, check bank account transaction history or account summary, transfer funds to another account. Whenever we deal with a banking system main concern should be the security related to banking transactions and account login activity.

# APPROACH

Throughout the project the focus has been on presenting information in an easy and intelligible manner. The project could be of help for those who want to know about the fundamentals of online banking systems and want to develop software based on the same concept.

The proposed project uses a MVC (Model - View - Controller) approach to built the system. The use of MVC architecture is preferred for small to large scale projects. It effectively separate the GUI from the back-end and allow for easy maintenance of the project. With this approach we make sure that our code is as DRY (Don’t Repeat Yourself) as possible. Errors are less likely to occur and can be dealt with without affecting the entire project.

## Objectives

1. Online opening of accounts
2. Online banking
3. Money Deposit
4. Money Withdrawal
5. Money Transfer
6. Transactions History
7. Balance Checking
8. Cash-In & Cash-Out

## User Views

* Customers

# Platform

## Operating System:

* Mac-Os

## Technologies Used:

* **JavaFx**
* **FXML**
* **PostgreSQL**
* **CSS**

## Software Requirements:

* **OpenJDK-16.0.2**
* **FXML-16 (Scene Builder)**
* **JavaFX-SDK-16**
* **PostgreSQL Server**
* **PostgresSQL JDBC 42.2.22**

SOFTWARE REQUIRMENT SPECIFICATION

Introduction

**1.1 Purpose:**

The objective of the **Online Banking System** is to allow customers of any bank providing the system to be able to complete basic transactions that would normally require them to physically be at the bank. It’ll also facilitate keeping all records of deposits, withdrawals and transfers and add extra features like profile update. In this manner, all transactions information can be available in a few seconds.

Overall, it’ll make online banking system an easier job for the administrator and the customer of any bank. The system can be extended to include other more serious aspects of banking. Due to a change of project and time constraint, the system only priorities the basic features of an online banking system. We will later on work on the administrator aspect of the software and add more features for customers such as adding adding additional cards, changing profile picture, request for bank statement, etc.

The main purpose of this document is to illustrate the requirements of the project **Online Banking System** and is intended to help any organization to maintain and manage its customers’ transactions.

**1.2 Scope :**

Without a **Online Banking System**, banks are limited to traditional banking methods (in-person), and with that comes some quirks.  
The Online Banking system will store all the details of the customers including their personal information, accounts details, and transactions.

**Register module**: Register module will help in creating new accounts with the bank. This module will simplify the task of on paper registration. After a successful registration, the customer is given an account number and customer id among other details to uniquely identify them in ten system. They can also update basic information.

**Login module**: Login module will help in authentication of user accounts. Users who have valid login username and password can log into their respective accounts.

**Account module**: Account module provides the user with a glimpse of transactions they have made over time and history of what’s been done in the system. They can quickly look check their balance and account number.

**Deposits module**: Deposits module helps with giving the user a way to deposit money into their account. It checks if the user has entered a valid amount. If they have, the transaction is successful and their balance is automatically updated along with how much they have deposited over time.

**Withdrawals module**: Withdrawals module helps with giving the user a way to withdraw money from their account. If the user has sufficient funds, the transaction is successful and their balance is automatically updated along with how much they have withdrawn over time.

**Transfers module**: Transfers module will the user transfer money from one account to another. It will check for account and amount validity.

**Transactions module**: Suppose there are hundreds of thousands of transactions and the user is looking for one in particular or transactions at a given time period. In a manual system this is a tedious task, but using this module we can easily search the transaction by specifying the transaction type, id, amount, date, etc in the search criteria.

**Settings module**: Settings module will help the customer change his username or password. Maybe they have been compromised or are not strong enough.

**Logout module**: Logout module will help the customer sign out of the system. The customer could be using the system on someone’s laptop for example and will be bad if they didn’t have a way to logout once they are done using the system.

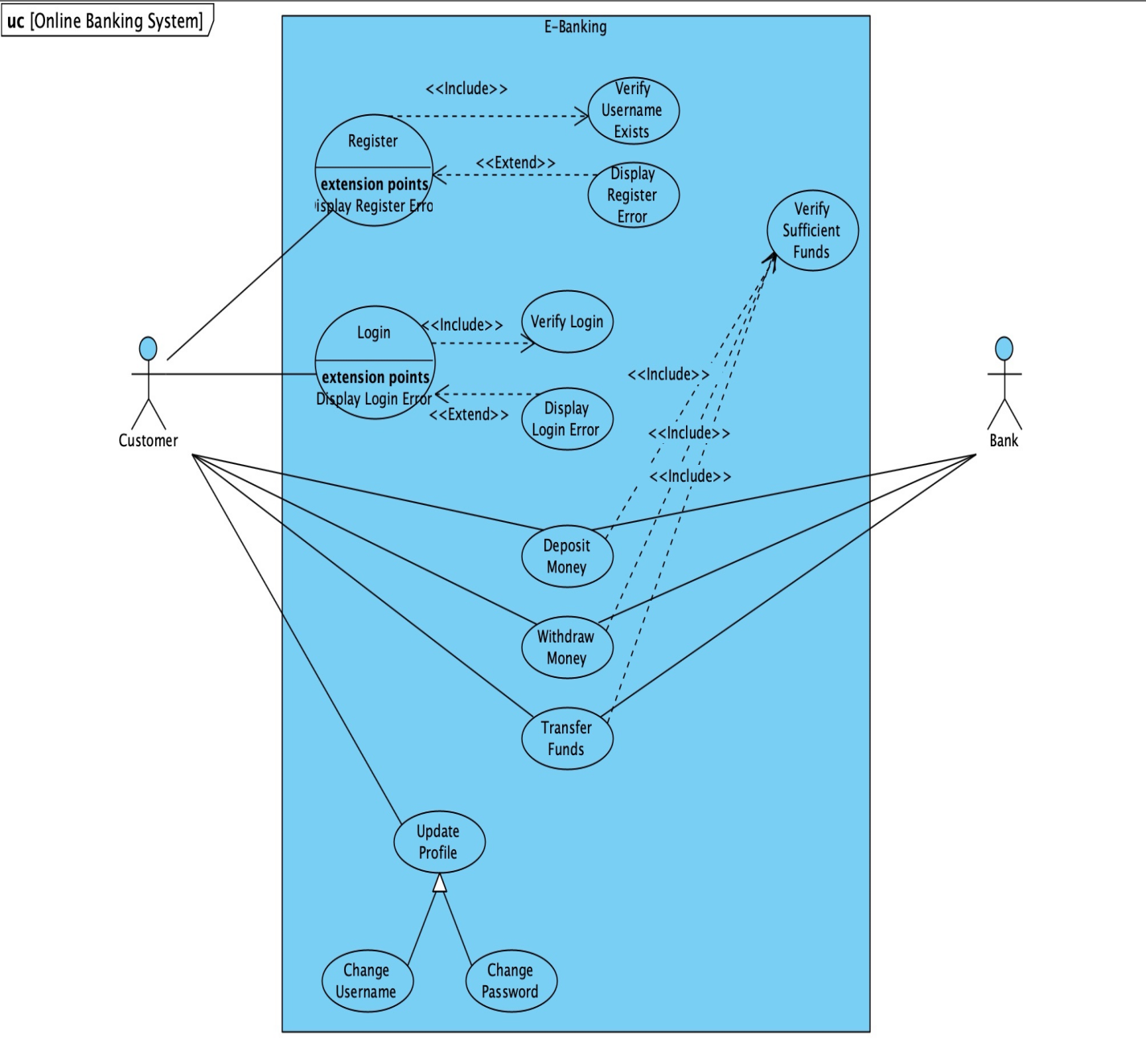
**2.1 Product Functions :**

There is only one user who will be using this product: A bank’s customer.

The features that are available to the Administrator are:

* Checking Balance
* Update username or password
* Deposit money
* Withdraw money
* Transfer money
* Checking personal information
* Checking Transactions and Account History

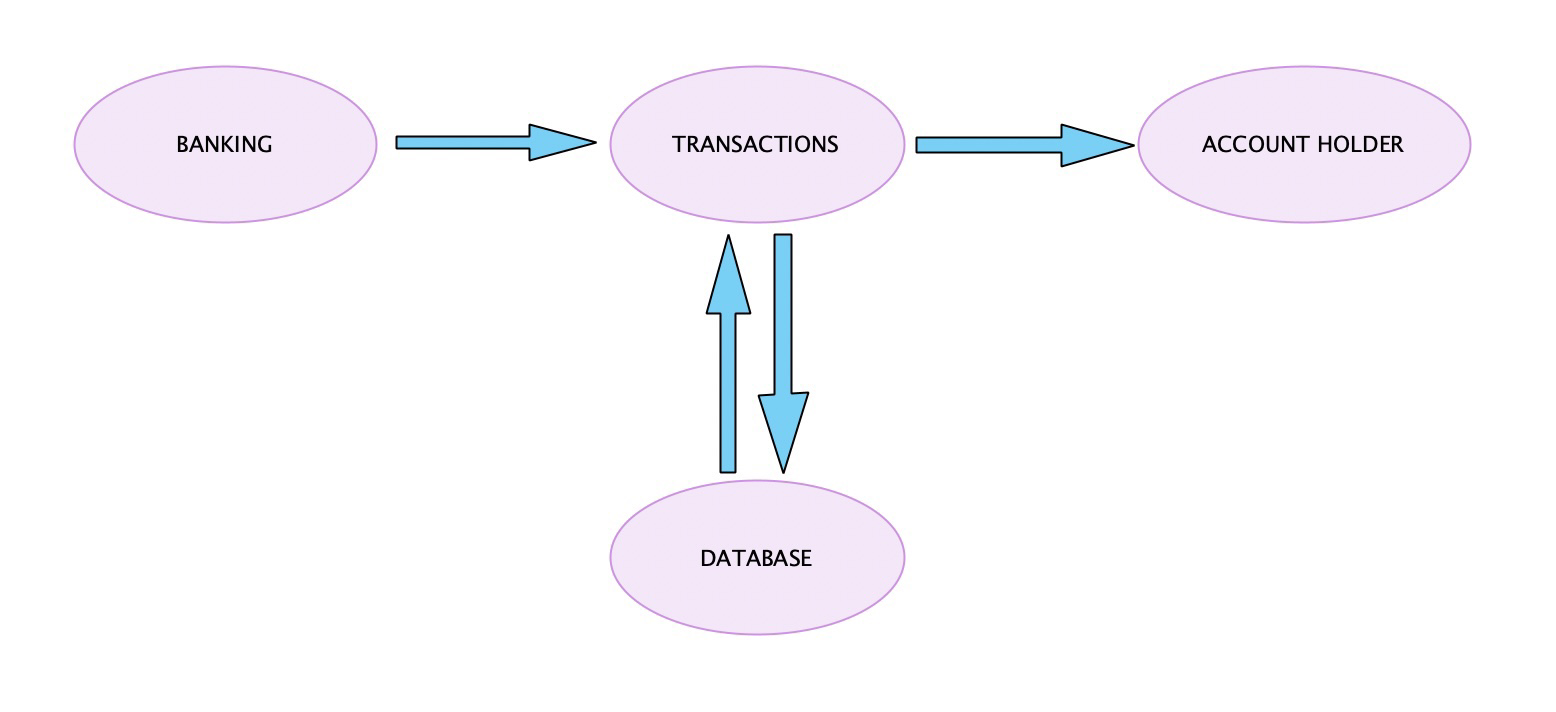
**2.2 USE CASE MODEL :**



**2.3 ACTIVITY DIAGRAM :**



**2.4 DATAFLOW :**



**2.5 DATABASE :**



**DESIGN PHASE**

**Introduction  
1.1 Scope and purpose**

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself.

The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

**1.2 Overall System Design Objectives**

The overall system design objective is to provide an efficient, modular design that will reduce the system’s complexity, facilitate change and result in an easy implementation. This will be accomplished by designing strongly cohesion system with minimal coupling. In addition, this document will provide interface design models that are consistent, user friendly and will provide straight forward transition through the various system functions.

**1.3 Project Structure**

Database Package: this package contains database configuration classes for the system.

Views Package: this package contains everything related to the graphical aspect of the system. It is responsible for rendering what the user sees when using the software.

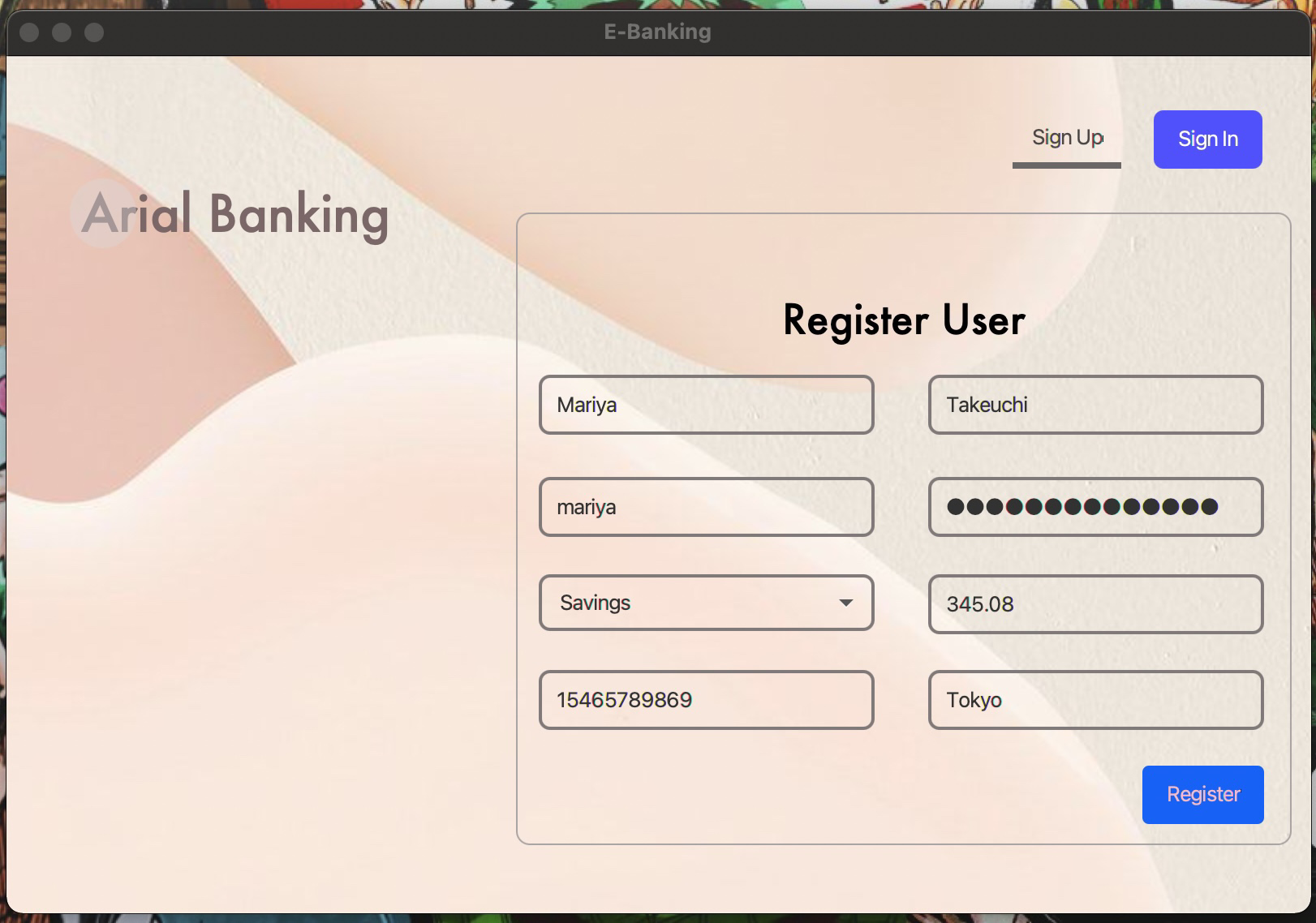
Controllers Package: this package as the name implies controls the behavior of the system. More precisely, it is composed of several controller classes for each view template in the system. It makes sure that every action taken by the user is answered with the correct information.

Models Package: this package is more abstract compared to others in the system. It contains sets of rules that the Controllers package need in order to function. It designed to eliminate data redundancy in the project and allow for better security. It is easily manageable and scalable to meet the programmer’s needs.

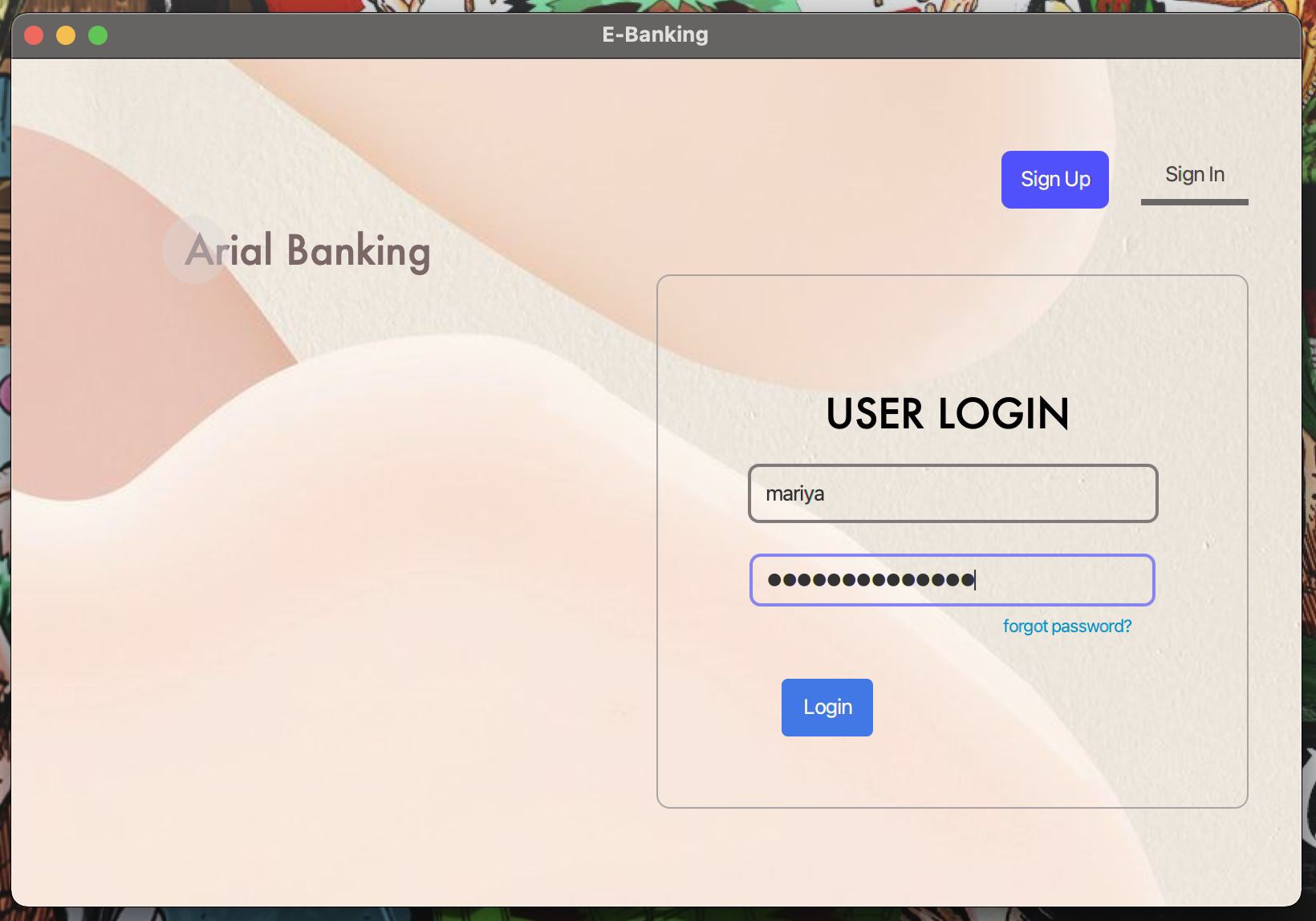
Public Package: this package is responsible for running the program and switching between pages. It is the main part of the system.

**USER INTERFACE**

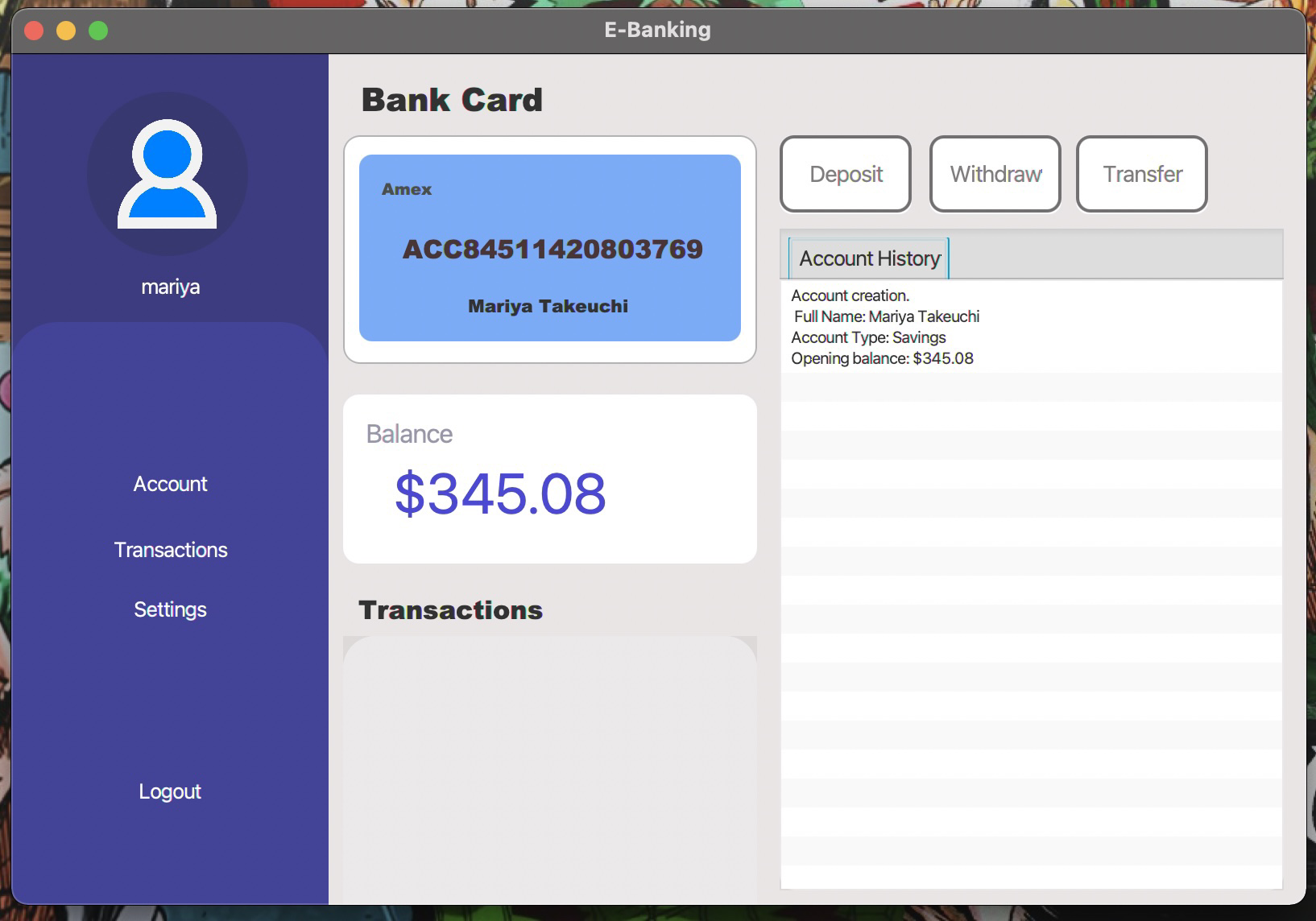
***User Registration***



***User Login***



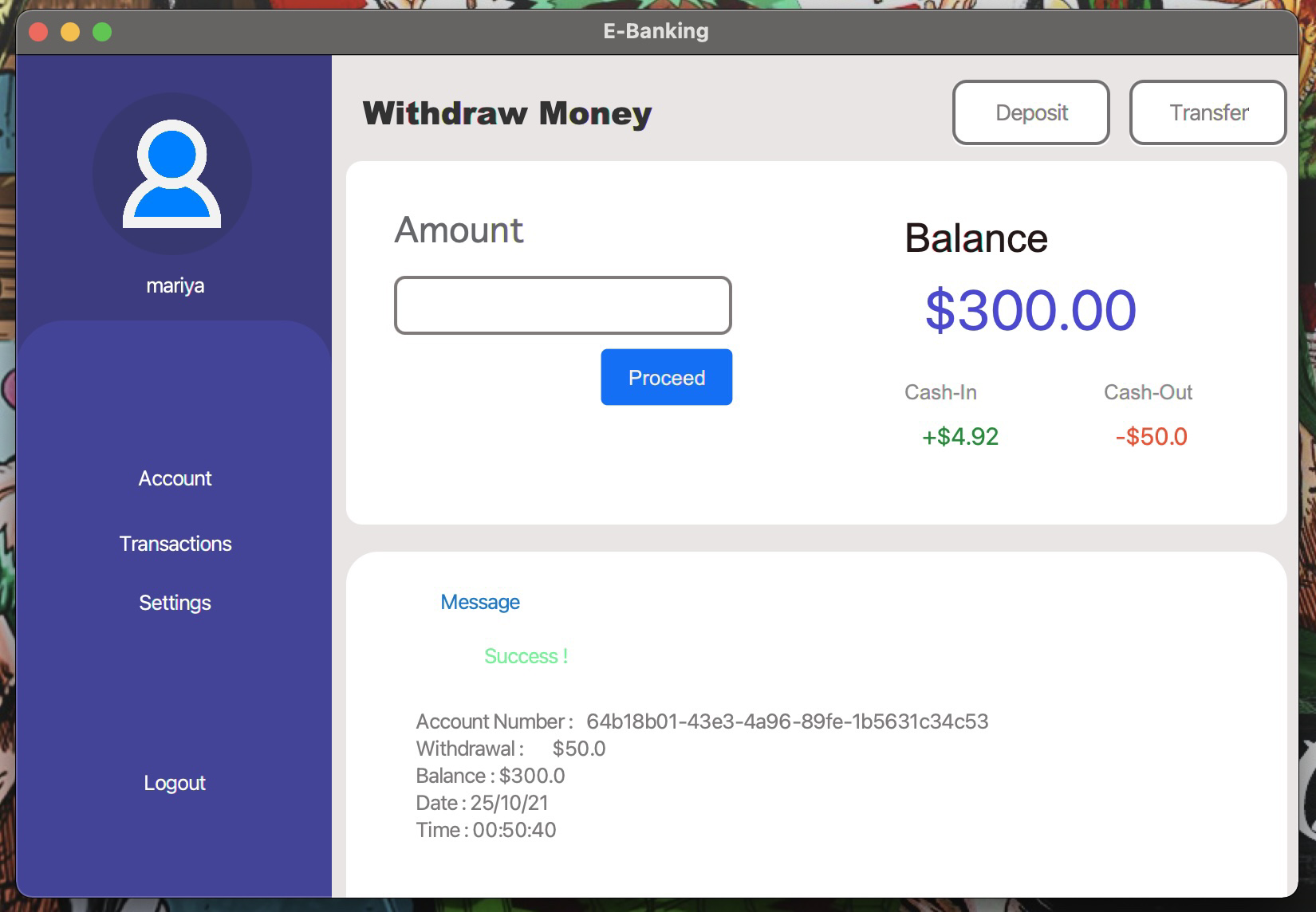
***User Account***



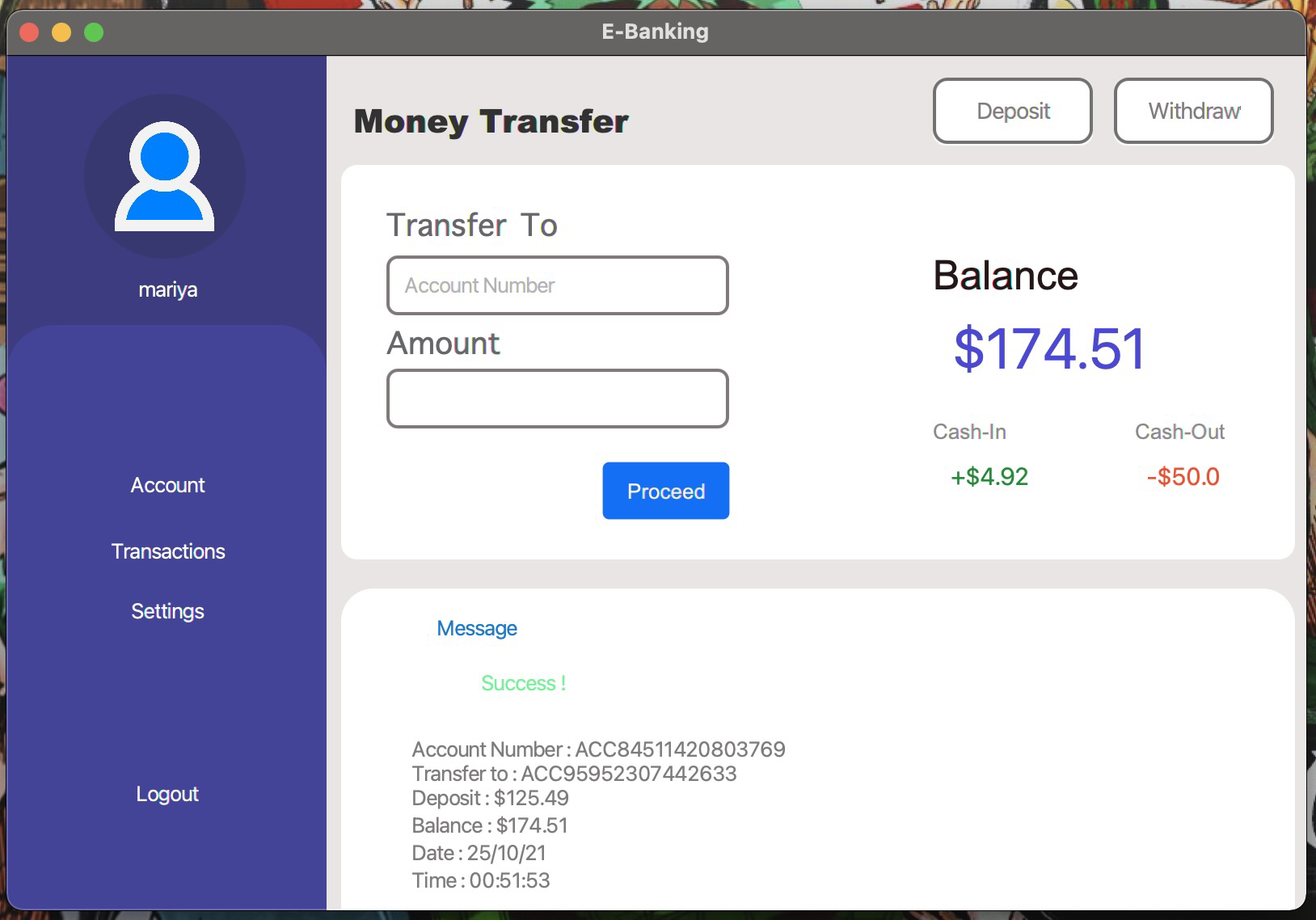
***Make Deposit***



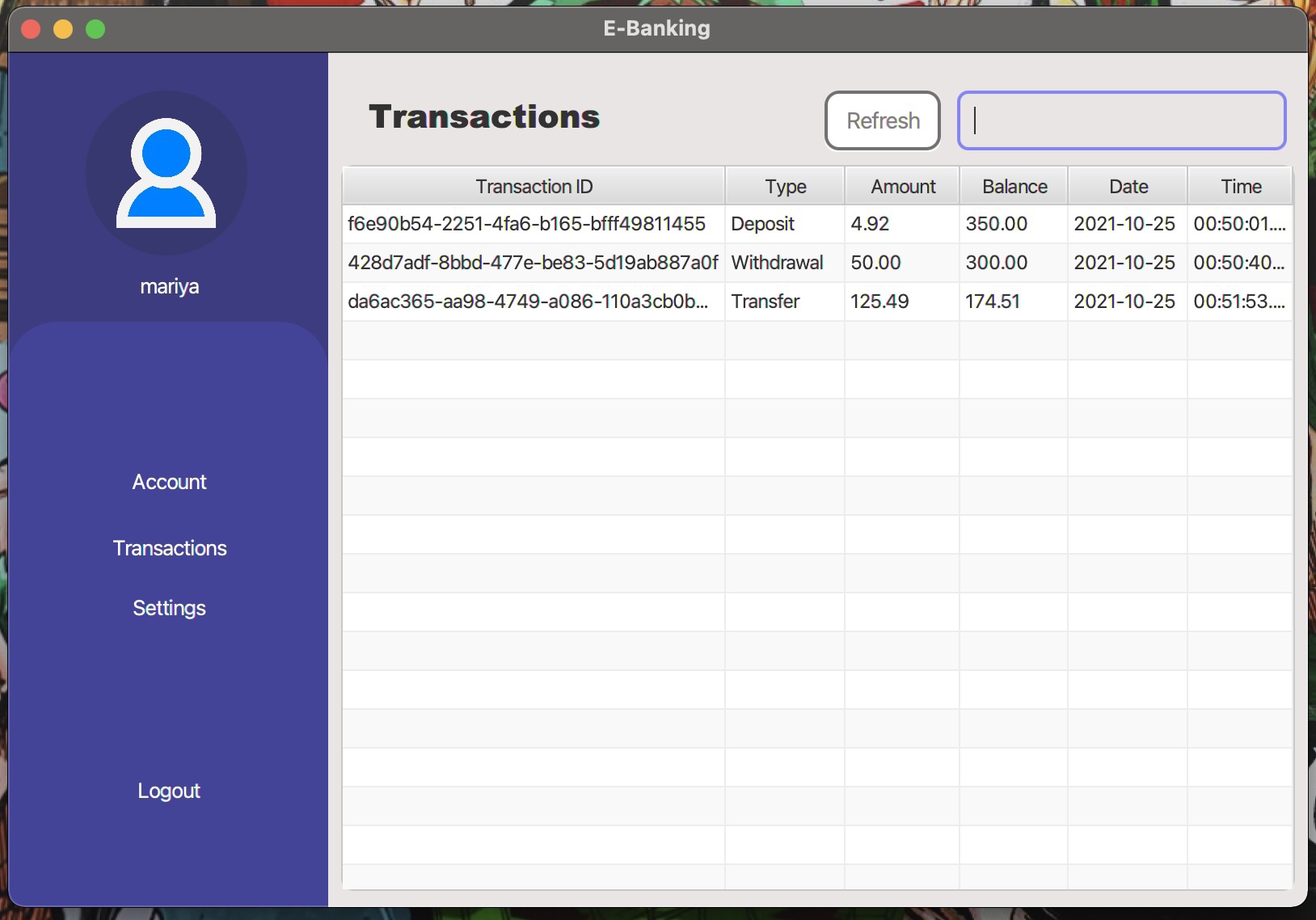
***Withdraw Money***



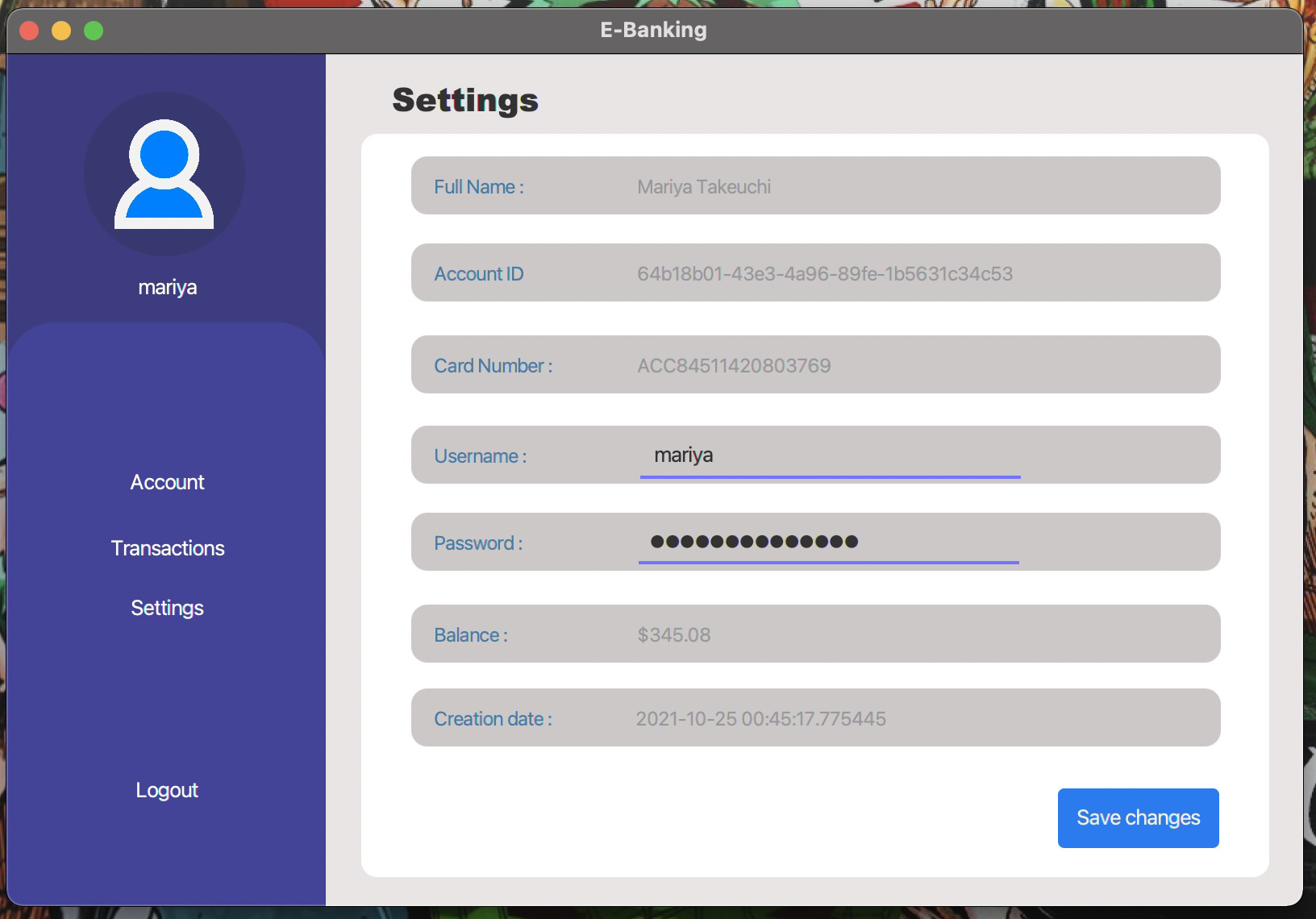
***Transfer Funds***



***Transactions Table***



***Account Settings***



**CODING**

***Database Conf Interface***

package Database;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
  
public interface Conf {  
 String  
 *DB\_DRIVER* = "org.postgresql.Driver",  
 *DB\_DSN* = "jdbc:postgresql://localhost:5432/jproj",  
 *DB\_USERNAME* = "kv.kn",  
 *DB\_PASSWORD* = "";  
  
  
 Connection connect ();  
 Connection disconnect () throws SQLException;  
 void initialize() throws SQLException;  
}

***Database Connect Class***

package Database;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
public class Connect implements Conf {  
 Connection conn;  
  
 @Override  
 public Connection connect () {  
 try { Class.*forName*(*DB\_DRIVER*); conn = DriverManager.*getConnection*(*DB\_DSN*, *DB\_USERNAME*, *DB\_PASSWORD*); }  
 catch (SQLException | ClassNotFoundException e) { e.printStackTrace(); }  
 return conn;  
 }  
  
 @Override  
 public Connection disconnect () throws SQLException {  
 if ( conn != null && !conn.isClosed() ) conn.close();  
 return conn;  
 }  
  
 @Override  
 public void initialize () throws SQLException {}  
}

***Register Class Controller***

package Controllers;  
  
import Database.Connect;  
import Models.AlertMessage;  
import Models.RegisterValidator;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.ChoiceBox;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.\*;  
import java.util.Random;  
import java.util.UUID;  
  
  
public class Register extends Connect {  
 static String *\_branchID* = null;  
 static int *count*;  
  
 StringBuilder account\_number () {  
 Random account\_number = new Random();  
 StringBuilder card = new StringBuilder("ACC");  
  
 for (int i = 0; i < 14; i++) {  
 int n = account\_number.nextInt(10);  
 card.append(n);  
 }  
  
 return card;  
 }  
  
 public void initialize () {  
 acc\_type.getItems().addAll(account\_type);  
 }  
  
 private String get\_type () {  
 return acc\_type.getValue();  
 }  
  
 private final String[] account\_type = { "Savings", "Regular" };  
 private static final String *cust\_query* = "INSERT INTO customers VALUES (?::uuid, ?::uuid, ?::varchar, ?::varchar, ?::varchar, ?::varchar, ?::varchar, ?::varchar)";  
 private static final String *acc\_query* = "INSERT INTO accounts VALUES (?::uuid, ?::uuid, ?::text, ?::varchar, ?::numeric)";  
 private static final String *hist\_query* = "INSERT INTO history VALUES (?::uuid, ?::uuid, ?::varchar, ?::varchar)";  
 private static final String *fetch\_query* = "SELECT \* FROM branches LIMIT 1";  
  
 @FXML  
 private Text message\_status;  
  
 @FXML  
 private TextField balance, firstname, lastname, username, password, city, contact;  
  
 @FXML  
 private ChoiceBox<String> acc\_type;  
  
 @FXML  
 void loginBtn () throws IOException { Switcher.*switcher*("/Views/login.fxml"); }  
  
 @FXML  
 void register () {  
 Connection conn = connect();  
  
 // ------------------------------------------------------  
 String cust = String.*valueOf*( UUID.*randomUUID*() );  
 String fname = firstname.getText();  
 String lname = lastname.getText();  
 String user = username.getText();  
 String pass = password.getText();  
 String cty = city.getText();  
 String contct = contact.getText();  
  
 final String SQL = "SELECT COUNT(\*) AS count FROM customers WHERE username = '" + user + "'";  
  
 RegisterValidator.*validate*(fname, lname, pass, cty, contct, firstname, lastname, password, city, contact, message\_status);  
  
 if ( user.length() == 0 ) {  
 username.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("field cannot be empty.");  
 AlertMessage.*message*(message\_status);  
 }  
 else {  
 username.setStyle(null);  
  
 try ( Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(SQL) ) {  
  
 while ( rs.next() ) *count* = rs.getInt("count");  
  
 if (*count* > 0) {  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("This username is already taken.");  
 AlertMessage.*message*(message\_status);  
 }  
 else {  
 try ( Statement \_fetch = conn.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(*fetch\_query*);  
 while ( res.next() ) *\_branchID* = res.getString(1);  
  
 try ( PreparedStatement \_stmt = conn.prepareStatement(*cust\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 \_stmt.setString(1, cust);  
 \_stmt.setString(2, *\_branchID*);  
 \_stmt.setString(3, fname);  
 \_stmt.setString(4, lname);  
 \_stmt.setString(5, user);  
 \_stmt.setString(6, pass);  
 \_stmt.setString(7, cty);  
 \_stmt.setString(8, contct);  
 \_stmt.executeUpdate();  
  
 //================================================================================================================================//  
 //================================================================================================================================//  
 //================================================================================================================================//  
 //================================================================================================================================//  
  
 String acc = String.*valueOf*( UUID.*randomUUID*() );  
 String card = String.*valueOf*( account\_number() );  
 String type = get\_type();  
 double amount = Double.*parseDouble*( balance.getText() );  
  
 try ( PreparedStatement stmt\_ = conn.prepareStatement(*acc\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 stmt\_.setString(1, acc);  
 stmt\_.setString(2, cust);  
 stmt\_.setString(3, card);  
 stmt\_.setString(4, type);  
 stmt\_.setDouble(5, amount);  
 stmt\_.executeUpdate();  
  
 //================================================================================================================================//  
 //================================================================================================================================//  
 //================================================================================================================================//  
 //================================================================================================================================//  
  
 String id = String.*valueOf*( UUID.*randomUUID*() );  
 String status = "Success";  
 String message = "Account creation.\n Full Name: " + fname + " " + lname + "\nAccount Type: " + type + "\nOpening balance: $" + amount;  
  
 try ( PreparedStatement \_history\_ = conn.prepareStatement(*hist\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 \_history\_.setString(1, id);  
 \_history\_.setString(2, acc);  
 \_history\_.setString(3, message);  
 \_history\_.setString(4, status);  
 \_history\_.executeUpdate();  
 }  
  
 } catch (SQLException err) { err.getStackTrace(); }  
  
 } catch (SQLException err) { err.getStackTrace(); }  
  
 } catch (SQLException err) { err.getStackTrace(); }  
  
 Switcher.*switcher*("/Views/login.fxml");  
 }  
  
 } catch (SQLException | IOException error) { error.printStackTrace(); }  
  
 }  
  
 }  
}

***Login Class Controller***

package Controllers;  
  
import Database.Connect;  
import Models.LoginValidator;  
import Models.RegistrationAlertMessage;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.Statement;  
import java.sql.SQLException;  
  
public class Login extends Connect {  
  
 protected static String *\_accountID*, *\_customerID*, *\_cardNumber*, *\_accountType*, *\_accountBalance*, *\_accountDate*, *\_firstname*, *\_lastname*,  
 *\_username*, *\_password*, *\_city*, *\_contact*, *\_depositID*, *\_deposit*, *\_depositBalance*, *\_depositDate*, *\_depositTime*,  
 *\_withdrawalID*, *\_withdrawal*, *\_withdrawalBalance*, *\_withdrawalDate*, *\_withdrawalTime*, *\_transferID*,  
 *\_accountFrom*, *\_accountTo*, *\_transferAmount*, *\_transferBalance*, *\_transferDate*, *\_transferTime*,  
 *\_historyMessage*, *\_historyStatus*, *\_historyDate*, *\_historyTime*;  
  
 protected static double *CASH\_IN*, *CASH\_OUT*, *WITHDRW*, *TRANS*;  
  
 @FXML  
 private TextField username, password;  
  
 @FXML  
 void registerBtn () throws IOException { Switcher.*switcher*("/Views/register.fxml"); }  
  
 @FXML  
 private Text message\_status;  
  
 @FXML  
 void login () {  
  
 Connection conn = connect();  
 String uname = username.getText();  
 String pass = password.getText();  
  
 LoginValidator.*validate* (uname, pass, username, password, message\_status);  
  
 if ( pass.length() >= 3 ) {  
  
 final String SQL = "SELECT \* FROM accounts a, customers c WHERE a.customer\_id = c.customer\_id AND username = '" + uname + "' AND password = '" + pass + "'";  
  
 try ( Statement stmt = conn.createStatement() ) {  
 ResultSet res = stmt.executeQuery(SQL);  
  
 while ( res.next() ) {  
 *\_accountID* = res.getString("acc\_id");  
 *\_customerID* = res.getString("customer\_id");  
 *\_cardNumber* = res.getString("card\_number");  
 *\_accountType* = res.getString("acc\_type");  
 *\_accountBalance* = res.getString("balance");  
 *\_accountDate* = res.getString("creation\_date");  
 *\_firstname* = res.getString("firstname");  
 *\_lastname* = res.getString("lastname");  
 *\_username* = res.getString("username");  
 *\_password* = res.getString("password");  
 *\_city* = res.getString("city");  
 *\_contact* = res.getString("contact");  
 }  
  
 if ( uname.equals(*\_username*) && pass.equals(*\_password*) ) Switcher.*switcher*("/Views/account.fxml");  
  
 else { message\_status.setStyle("-fx-fill: #f66262;"); message\_status.setText("username or password is wrong or both."); RegistrationAlertMessage.*message*(message\_status); }  
  
 } catch (SQLException | IOException err) { err.getStackTrace(); }  
  
 }  
  
 else { message\_status.setStyle("-fx-fill: #f66262;"); message\_status.setText("password should be more than 3 characters long."); RegistrationAlertMessage.*message*(message\_status); }  
  
 }  
  
}

***Account Class Controller***

package Controllers;  
  
import Models.Initializer;  
import Public.\*;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.ListView;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.SQLException;  
  
public class Account extends Login {  
 @FXML  
 private Label account\_user, account\_holder, account\_card;  
  
 @FXML  
 private Text account\_balance;  
  
 @FXML  
 private ListView<String> account\_history, account\_transactions;  
  
 public void initialize () throws SQLException { Initializer.*init\_accounts* ( account\_user, account\_holder, account\_card, account\_balance, account\_history, account\_transactions, connect() ); }  
  
 @FXML  
 void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 void depositBtn () throws IOException { Switcher.*switcher*("/Views/deposit.fxml"); }  
  
 @FXML  
 void withdrawalBtn () throws IOException { Switcher.*switcher*("/Views/withdrawal.fxml"); }  
  
 @FXML  
 void transferBtn () throws IOException { Switcher.*switcher*("/Views/transfer.fxml"); }  
  
 @FXML  
 void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
}

***Deposit Class Controller***

package Controllers;  
  
import Models.AlertMessage;  
import Models.Initializer;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.\*;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.UUID;  
  
public class Deposit extends Login {  
 private static final String *dep\_query* = "INSERT INTO deposits (transac\_id, acc\_id, cust\_id, deposit, balance) VALUES (?::uuid, ?::uuid, ?::uuid, ?::numeric, ?::numeric)";  
 private static final String *hist\_query* = "INSERT INTO history VALUES (?::uuid, ?::uuid, ?::varchar, ?::varchar)";  
  
 @FXML  
 private Label account\_user;  
  
 @FXML  
 private Text account\_balance, balance\_deposit, balance\_withdrawal, message\_status, message\_account, message\_deposit, message\_balance, message\_date, message\_time;  
  
 @FXML  
 private TextField depot;  
  
 @FXML  
 void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 void withdrawalBtn () throws IOException { Switcher.*switcher*("/Views/withdrawal.fxml"); }  
  
 @FXML  
 void transferBtn () throws IOException { Switcher.*switcher*("/Views/transfer.fxml"); }  
  
 @FXML  
 void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
  
 @Override  
 public void initialize () throws SQLException { Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() ); }  
  
 @FXML  
 void proceed () {  
 Connection conn = connect();  
  
 String transac\_id = String.*valueOf*( UUID.*randomUUID*() );  
 String id = String.*valueOf*(UUID.*randomUUID*());  
 String status = "Success";  
 String acc = *\_accountID*;  
 String cust = *\_customerID*;  
 double dep = Double.*parseDouble*( depot.getText() );  
 double bal = 0;  
 String message = "Money deposit: \t$" + dep;  
  
 final String balance\_query = "SELECT \* FROM accounts WHERE acc\_id = '" + acc + "'";  
  
 if ( depot.getText().length() == 0 ) depot.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
 else depot.setStyle(null);  
  
 try ( Statement \_fetch = conn.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(balance\_query);  
 while ( res.next() )  
 bal = res.getDouble("balance");  
  
 if ( dep > 0 ) {  
  
 try ( PreparedStatement stmt = conn.prepareStatement(*dep\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 stmt.setString(1, transac\_id);  
 stmt.setString(2, acc);  
 stmt.setString(3, cust);  
 stmt.setDouble(4, dep);  
 stmt.setDouble(5, bal + dep);  
 stmt.executeUpdate();  
 } catch (SQLException err) { err.getStackTrace(); }  
  
  
 final String acc\_query = "UPDATE accounts SET balance = balance + '" + dep + "' WHERE acc\_id = '" + acc + "'";  
 try ( Statement \_stmt = conn.createStatement() ) { \_stmt.executeUpdate(acc\_query); } catch (SQLException err) { err.getStackTrace(); }  
  
  
 try ( PreparedStatement \_history\_ = conn.prepareStatement(*hist\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 \_history\_.setString(1, id);  
 \_history\_.setString(2, acc);  
 \_history\_.setString(3, message);  
 \_history\_.setString(4, status);  
 \_history\_.executeUpdate();  
 } catch (SQLException err) { err.getStackTrace(); }  
  
 Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() );  
  
 DateFormat date\_format = new SimpleDateFormat("dd/MM/yy");  
 Calendar cal1 = Calendar.*getInstance*();  
 String date = date\_format.format( cal1.getTime() );  
  
 DateFormat time\_format = new SimpleDateFormat("HH:mm:ss");  
 Calendar cal2 = Calendar.*getInstance*();  
 String time = time\_format.format( cal2.getTime() );  
  
 message\_status.setStyle("-fx-fill: #60ee88;");  
  
 message\_status.setText("\t\t\t\tSuccess !");  
 message\_account.setText("\t\tAccount Number : \t" + acc);  
 message\_deposit.setText("\t\tDeposit : \t$" + dep);  
 bal += dep;  
 Withdrawal.*message\_alert* (bal, date, time, message\_balance, message\_date, message\_time, message\_status, message\_account, message\_deposit);  
  
 }  
 else {  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("\t\t\t\tInvalid amount, please try again !");  
 AlertMessage.*message*(message\_status);  
 }  
  
 depot.clear();  
  
 } catch (SQLException err) { err.getStackTrace(); }  
  
 }  
  
}

***Withdrawal Class Controller***

package Controllers;  
  
import Models.AlertMessage;  
import Models.Initializer;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.\*;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.UUID;  
  
public class Withdrawal extends Login {  
 private static final String *with\_query* = "INSERT INTO withdrawals (transac\_id, acc\_id, cust\_id, withdrawal, balance) VALUES (?::uuid, ?::uuid, ?::uuid, ?::numeric, ?::numeric)";  
 private static final String *hist\_query* = "INSERT INTO history VALUES (?::uuid, ?::uuid, ?::varchar, ?::varchar)";  
  
 @FXML  
 private TextField withdrw;  
  
 @FXML  
 private Label account\_user;  
  
 @FXML  
 private Text account\_balance, balance\_deposit, balance\_withdrawal, message\_status, message\_account, message\_deposit, message\_balance, message\_date, message\_time;  
  
 @FXML  
 void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 void depositBtn () throws IOException { Switcher.*switcher*("/Views/deposit.fxml"); }  
  
 @FXML  
 void transferBtn () throws IOException { Switcher.*switcher*("/Views/transfer.fxml"); }  
  
 @FXML  
 void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
  
 @Override  
 public void initialize () throws SQLException { Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() ); }  
  
 @FXML  
 void proceed () {  
 Connection conn = connect();  
  
 String transac\_id = String.*valueOf*( UUID.*randomUUID*() );  
 String id = String.*valueOf*( UUID.*randomUUID*() );  
 String status = "Success";  
 String cust = *\_customerID*, acc = *\_accountID*;  
 double with = Double.*parseDouble*( withdrw.getText() );  
 double bal = 0;  
 String message = "Money withdrawal: \t$" + with;  
  
 final String balance\_query = "SELECT \* FROM accounts WHERE acc\_id = '" + acc + "'";  
  
 if ( withdrw.getText().length() == 0 ) withdrw.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
 else withdrw.setStyle(null);  
  
 try ( Statement \_fetch = conn.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(balance\_query);  
 while ( res.next() )  
 bal = res.getDouble(5);  
  
 if ( with > 0 && bal > 0 && bal >= with ) {  
  
 try ( PreparedStatement stmt = conn.prepareStatement(*with\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 stmt.setString(1, transac\_id);  
 stmt.setString(2, acc);  
 stmt.setString(3, cust);  
 stmt.setDouble(4, with);  
 stmt.setDouble(5, bal - with);  
 stmt.executeUpdate();  
 } catch (SQLException err) { err.getStackTrace(); }  
  
  
 final String query = "UPDATE accounts SET balance = balance - '" + with + "' WHERE acc\_id = '" + acc + "'";  
 try ( Statement stmt = conn.createStatement() ) { stmt.executeUpdate(query); } catch (SQLException err) { err.printStackTrace(); }  
  
  
 try ( PreparedStatement \_history\_ = conn.prepareStatement(*hist\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 \_history\_.setString(1, id);  
 \_history\_.setString(2, acc);  
 \_history\_.setString(3, message);  
 \_history\_.setString(4, status);  
 \_history\_.executeUpdate();  
 } catch (SQLException err) { err.printStackTrace(); }  
  
 Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() );  
  
 String f = "dd/MM/yy";  
 DateFormat df = new SimpleDateFormat(f);  
 Calendar cal1;  
 cal1 = Calendar.*getInstance*();  
 String date;  
 date = df.format( cal1.getTime() );  
  
 DateFormat time\_format = new SimpleDateFormat("HH:mm:ss");  
 Calendar cal2 = Calendar.*getInstance*();  
 String time = time\_format.format( cal2.getTime() );  
  
 message\_status.setStyle("-fx-fill: #60ee88;");  
  
 message\_status.setText("\t\t\t\tSuccess !");  
 message\_account.setText("\t\tAccount Number : \t" + acc);  
 message\_deposit.setText("\t\tWithdrawal : \t$" + with);  
 bal -= with;  
 *message\_alert*(bal, date, time, message\_balance, message\_date, message\_time, message\_status, message\_account, message\_deposit);  
  
 }  
 else {  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("\t\t\t\tSorry, you don't have sufficient funds to proceed with this transaction !");  
 AlertMessage.*message*(message\_status);  
 }  
  
 withdrw.clear();  
  
 } catch (SQLException err) { err.printStackTrace(); }  
  
 }  
  
 static void message\_alert (double bal, String date, String time, Text message\_balance, Text message\_date, Text message\_time, Text message\_status, Text message\_account, Text message\_deposit) {  
 message\_balance.setText("\t\tBalance : $" + bal);  
 message\_date.setText("\t\tDate : " + date);  
 message\_time.setText("\t\tTime : " + time);  
  
 AlertMessage.*message*(message\_status);  
 AlertMessage.*message*(message\_status);  
 AlertMessage.*message*(message\_account);  
 AlertMessage.*message*(message\_deposit);  
 AlertMessage.*message*(message\_balance);  
 AlertMessage.*message*(message\_date);  
 AlertMessage.*message*(message\_time);  
 }  
}

***Transfer Class Controller***

package Controllers;  
  
import Models.AlertMessage;  
import Models.Initializer;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.\*;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.UUID;  
  
public class Transfer extends Login {  
 static String *accID*;  
 private static final String *transfer\_query* = "INSERT INTO transfers (balance, transfer\_id, acc\_from, acc\_to, amount) VALUES (?::numeric, ?::uuid, ?::text, ?::text, ?::numeric)";  
 private static final String *hist\_query* = "INSERT INTO history VALUES (?::uuid, ?::uuid, ?::varchar, ?::varchar)";  
  
 @FXML  
 private TextField acc\_to, amount;  
  
 @FXML  
 private Label account\_user;  
  
 @FXML  
 private Text account\_balance, balance\_deposit, balance\_withdrawal, message\_status, message\_account, mess\_transfer\_to, message\_deposit, message\_balance, message\_date, message\_time;  
  
 @FXML  
 void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 void depositBtn () throws IOException { Switcher.*switcher*("/Views/deposit.fxml"); }  
  
 @FXML  
 void withdrawalBtn () throws IOException { Switcher.*switcher*("/Views/withdrawal.fxml"); }  
  
 @FXML  
 void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
  
 @Override  
 public void initialize () throws SQLException { Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() ); }  
  
 @FXML  
 void proceed () {  
 Connection conn = connect();  
  
 String d = "dd/MM/yy";  
 Calendar \_c;  
 DateFormat df = new SimpleDateFormat(d);  
 \_c = Calendar.*getInstance*();  
 String dt;  
 dt = df.format( \_c.getTime() );  
  
 DateFormat time\_format = new SimpleDateFormat("HH:mm:ss");  
 Calendar cal2 = Calendar.*getInstance*();  
 String time = time\_format.format( cal2.getTime() );  
  
 String transfer\_id = String.*valueOf*( UUID.*randomUUID*() );  
 String id = String.*valueOf*( UUID.*randomUUID*() );  
 String status = "Success";  
 String from = *\_cardNumber*;  
 String to = acc\_to.getText();  
 double amnt = Double.*parseDouble*( amount.getText() );  
 double bal = 0;  
 String message = "Money transfer : \t$" + amnt + " to Account Number: " + to;  
  
 final String balance\_query = "SELECT \* FROM accounts WHERE card\_number = '" + from + "'";  
  
 if ( to.length() == 0 ) acc\_to.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
 else acc\_to.setStyle(null);  
  
 if ( amount.getText().length() == 0 ) amount.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
 else amount.setStyle(null);  
  
 try ( Statement fetch = conn.createStatement(); ResultSet res = fetch.executeQuery(balance\_query) ) {  
 while ( res.next() ) {  
 *accID* = res.getString("acc\_id");  
 bal = res.getDouble("balance");  
 }  
  
 if ( amnt > 0 && bal > 0 && bal >= amnt ) {  
  
 try ( PreparedStatement stmt = conn.prepareStatement(*transfer\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 stmt.setDouble(1, bal - amnt);  
 stmt.setString(2, transfer\_id);  
 stmt.setString(3, from);  
 stmt.setString(4, to);  
 stmt.setDouble(5, amnt);  
 stmt.executeUpdate();  
 } catch (SQLException error) { error.printStackTrace(); }  
  
  
 final String deduction = "UPDATE accounts SET balance = balance - '" + amnt + "' WHERE card\_number = '" + from + "'";  
 try ( Statement stmt = conn.createStatement() ) { stmt.executeUpdate(deduction); } catch (SQLException err) { err.printStackTrace(); }  
  
 final String addition = "UPDATE accounts SET balance = balance + '" + amnt + "' WHERE card\_number = '" + to + "'";  
 try { Statement stmt = conn.createStatement(); stmt.executeUpdate(addition); } catch (SQLException err) { err.printStackTrace(); }  
  
  
 try ( PreparedStatement \_history\_ = conn.prepareStatement(*hist\_query*, Statement.*RETURN\_GENERATED\_KEYS*) ) {  
 \_history\_.setString(1, id);  
 \_history\_.setString(2, *accID*);  
 \_history\_.setString(3, message);  
 \_history\_.setString(4, status);  
 \_history\_.executeUpdate();  
 } catch (SQLException err) { err.getCause(); }  
  
 Initializer.*init* ( account\_user, account\_balance, balance\_deposit, balance\_withdrawal, connect() );  
  
 message\_status.setStyle("-fx-fill: #60ee88;");  
  
 message\_status.setText("\t\t\t\tSuccess !");  
 message\_account.setText("\t\tAccount Number : " + from);  
 mess\_transfer\_to.setText("\t\tTransfer to : " + to);  
 message\_deposit.setText("\t\tDeposit : $" + amnt);  
 AlertMessage.*message*(mess\_transfer\_to);  
 bal -= amnt;  
 Withdrawal.*message\_alert*(bal, dt, time, message\_balance, message\_date, message\_time, message\_status, message\_account, message\_deposit);  
  
 }  
 else {  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("\t\t\t\tSorry, you don't have sufficient funds to proceed with this transaction !");  
 AlertMessage.*message*(message\_status);  
 }  
  
 acc\_to.clear(); amount.clear();  
  
 } catch (SQLException err) { err.printStackTrace(); }  
 }  
}

***Transactions Class Controller***

package Controllers;  
  
import Models.Initializer;  
import Models.TabViewHelper;  
import Models.TransactionsHistory;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.TextField;  
import javafx.scene.control.cell.PropertyValueFactory;  
  
import java.io.IOException;  
import java.sql.SQLException;  
  
  
public class Transactions extends Login {  
  
 @FXML  
 private Label account\_user;  
  
 @FXML  
 private TableView<TransactionsHistory> transactions\_view;  
  
 @FXML  
 private TableColumn<TransactionsHistory, String> TRANS\_ID, TRANS\_TYPE, TRANS\_AMOUNT, TRANS\_BALANCE, TRANS\_DATE, TRANS\_TIME;  
  
 @FXML  
 private TextField searching;  
  
 public void initialize () throws SQLException {  
 Initializer.*init\_transactions* ( account\_user, connect() );  
  
 TabViewHelper.*table\_view*(transactions\_view);  
  
 TRANS\_ID.setCellValueFactory( new PropertyValueFactory<>("trans\_id"));  
 TRANS\_TYPE.setCellValueFactory( new PropertyValueFactory<>("trans\_type"));  
 TRANS\_AMOUNT.setCellValueFactory( new PropertyValueFactory<>("trans\_amount"));  
 TRANS\_BALANCE.setCellValueFactory( new PropertyValueFactory<>("trans\_balance"));  
 TRANS\_DATE.setCellValueFactory( new PropertyValueFactory<>("trans\_date"));  
 TRANS\_TIME.setCellValueFactory( new PropertyValueFactory<>("trans\_time"));  
  
 TabViewHelper.*table\_search*(transactions\_view, searching);  
 }  
  
 @FXML  
 private void refresh () throws SQLException { TabViewHelper.*table\_refresh*(transactions\_view); searching.clear(); TabViewHelper.*table\_search*(transactions\_view, searching); }  
  
 @FXML  
 private void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 private void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 private void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
  
}

***Settings Class Controller***

package Controllers;  
  
import Models.Initializer;  
import Models.RegistrationAlertMessage;  
import Public.Switcher;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.io.IOException;  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class Settings extends Login {  
  
 @FXML  
 private Text message\_status;  
  
 @FXML  
 private Label account\_fullname, account\_id, account\_card, account\_user, account\_balance, account\_creation;  
  
 @FXML  
 private TextField account\_username, account\_password;  
  
 @FXML  
 void accountBtn () throws IOException { Switcher.*switcher*("/Views/account.fxml"); }  
  
 @FXML  
 void transactionsBtn () throws IOException { Switcher.*switcher*("/Views/transactions.fxml"); }  
  
 @FXML  
 void settingsBtn () throws IOException { Switcher.*switcher*("/Views/settings.fxml"); }  
  
 @FXML  
 private void logoutBtn () throws IOException, SQLException { disconnect(); Switcher.*switcher*("/Views/login.fxml"); }  
  
 public void initialize () throws SQLException {  
 account\_user.setText(*\_username*);  
 account\_fullname.setText(*\_firstname* + " " + *\_lastname*);  
 account\_id.setText(*\_accountID*);  
 account\_card.setText(*\_cardNumber*);  
  
 Initializer.*init\_settings* ( account\_username, account\_password, connect() );  
  
 account\_balance.setText("$" + *\_accountBalance*);  
 account\_creation.setText(*\_accountDate*);  
 }  
  
 @FXML  
 void save\_changes () throws SQLException {  
 Connection conn = connect();  
 String user = account\_username.getText();  
 String pass = account\_password.getText();  
  
 final String sql = "UPDATE customers SET username = '" + user + "', password = '" + pass + "' WHERE customer\_id = '" + *\_customerID* + "'";  
  
 try ( Statement stmt = conn.createStatement() ) { stmt.executeUpdate(sql); } catch (SQLException err) { err.getStackTrace(); }  
  
 Initializer.*init\_settings* ( account\_username, account\_password, connect() );  
  
 message\_status.setStyle("-fx-fill: #125f25;");  
 message\_status.setText("Saved changes. \tusername : " + user + "\t\tpassword : " + pass);  
 RegistrationAlertMessage.*message*(message\_status);  
 }  
}

***AlertMessage Interface***

package Models;  
  
import javafx.animation.KeyFrame;  
import javafx.animation.KeyValue;  
import javafx.animation.Timeline;  
import javafx.beans.property.DoubleProperty;  
import javafx.scene.text.Text;  
import javafx.util.Duration;  
import org.jetbrains.annotations.NotNull;  
  
public interface AlertMessage {  
  
 static void message (@NotNull Text text) {  
 DoubleProperty opacity = text.opacityProperty();  
 Timeline fadeOut =  
 new Timeline (  
 new KeyFrame ( Duration.*ZERO*, new KeyValue(opacity, 1) ),  
 new KeyFrame ( Duration.*seconds*(40), new KeyValue(opacity, 0) )  
 );  
 fadeOut.play();  
 }  
  
}

***Intializer Class***

package Models;  
  
import Controllers.Login;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.scene.control.Label;  
import javafx.scene.control.ListView;  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Collections;  
  
public class Initializer extends Login {  
  
 public static ObservableList<TransactionsHistory> *HistoryList* = FXCollections.*observableArrayList*();  
  
 public static void init (Label account\_user, Text account\_balance, Text balance\_deposit, Text balance\_withdrawal, Connection connect) throws SQLException {  
  
 final String q = "SELECT \* FROM accounts a LEFT JOIN customers c ON c.customer\_id = a.customer\_id WHERE acc\_id = '" + *\_accountID* + "'";  
  
 try ( Statement \_fetch = connect.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(q);  
 while ( res.next() ) {  
 account\_user.setText( res.getString("username") );  
 account\_balance.setText( "$" + res.getString( "balance") );  
 }  
 }  
  
 final String cash\_in = "SELECT SUM(deposit) AS CASH\_IN FROM deposits WHERE acc\_id = '" + *\_accountID* + "'";  
 try ( ResultSet r = connect.createStatement().executeQuery(cash\_in) ) { while ( r.next() ) *CASH\_IN* = r.getDouble("CASH\_IN"); balance\_deposit.setText("+$" + *CASH\_IN*); }  
  
 final String w = "SELECT SUM(withdrawal) AS WITHDRW FROM withdrawals WHERE acc\_id = '" + *\_accountID* + "'";  
 try ( ResultSet r = connect.createStatement().executeQuery(w) ) { while ( r.next() ) *WITHDRW* = r.getDouble("WITHDRW"); }  
  
 final String t = "SELECT SUM(amount) AS TRANS FROM transfers WHERE acc\_from = '" + *\_accountID* + "'";  
 try ( ResultSet r = connect.createStatement().executeQuery(t) ) { while ( r.next() ) *TRANS* = r.getDouble("TRANS"); }  
  
 *CASH\_OUT* = *WITHDRW* + *TRANS*;  
 balance\_withdrawal.setText("-$" + *CASH\_OUT*);  
 }  
  
  
 public static void init\_settings (TextField account\_username, TextField account\_password, Connection connect) throws SQLException {  
  
 final String q = "SELECT username, password FROM customers WHERE customer\_id = '" + *\_customerID* + "'";  
  
 try ( Statement \_fetch = connect.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(q);  
 while ( res.next() ) {  
 account\_username.setText( res.getString("username") );  
 account\_password.setText( res.getString("password") );  
 }  
 }  
  
 }  
  
 public static void init\_transactions (Label account\_user, Connection connect) throws SQLException {  
  
 final String q = "SELECT username FROM customers WHERE customer\_id = '" + *\_customerID* + "'";  
  
 try ( Statement \_fetch = connect.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(q);  
 while ( res.next() ) account\_user.setText( res.getString("username") );  
 }  
  
 }  
  
 public static void init\_accounts (Label account\_user, Label account\_holder, Label account\_card, Text account\_balance, ListView<String> account\_history, ListView<String> account\_transactions, Connection connect) throws SQLException {  
  
 final String q = "SELECT \* FROM accounts a LEFT JOIN customers c ON c.customer\_id = a.customer\_id WHERE acc\_id = '" + *\_accountID* + "'";  
  
 try ( Statement \_fetch = connect.createStatement() ) {  
 ResultSet res = \_fetch.executeQuery(q);  
 while ( res.next() ) {  
 account\_user.setText( res.getString("username") );  
 account\_holder.setText( res.getString("firstname") + " " + res.getString("lastname") );  
 account\_card.setText( res.getString("card\_number") );  
 account\_balance.setText("$" + res.getDouble("balance") );  
 }  
 }  
  
 final String \_\_SQL = "SELECT \* FROM history WHERE acc\_id = '" + *\_accountID* + "'";  
  
 try {  
 Statement \_\_stmt = connect.createStatement(); ResultSet r = \_\_stmt.executeQuery(\_\_SQL);  
  
 while ( r.next() ) {  
 *\_historyMessage* = r.getString("message");  
 *\_historyStatus* = r.getString("status");  
 *\_historyDate* = r.getString("date");  
 *\_historyTime* = r.getString("time");  
 account\_history.getItems().addAll( Collections.*singleton*(*\_historyMessage*) );  
 }  
 } catch (SQLException error) { error.printStackTrace(); }  
  
  
 final String \_SQL = "SELECT DISTINCT d.transac\_id D\_ID, d.deposit DEP, d.balance D\_BALANCE, d.date D\_DATE, d.time D\_TIME, w.transac\_id W\_ID, " +  
 "w.withdrawal WITHD, w.balance W\_BALANCE, w.date W\_DATE, w.time W\_TIME, t.transfer\_id T\_ID, t.acc\_from T\_FROM, t.acc\_to T\_TO, " +  
 "t.amount T\_AMOUNT, t.balance T\_BALANCE, t.date T\_DATE, t.time T\_TIME FROM accounts a JOIN transfers t on a.card\_number = t.acc\_from " +  
 "FULL JOIN deposits d on d.cust\_id = a.customer\_id FULL JOIN withdrawals w ON d.acc\_id = w.acc\_id WHERE d.acc\_id = '" + *\_accountID* + "'";  
  
 try ( Statement \_stmt = connect.createStatement() ) {  
 ResultSet rs = \_stmt.executeQuery(\_SQL);  
  
 while ( rs.next() ) {  
 *\_depositID* = rs.getString("D\_ID");  
 *\_deposit* = rs.getString("DEP");  
 *\_depositBalance* = rs.getString("D\_BALANCE");  
 *\_depositDate* = rs.getString("D\_DATE");  
 *\_depositTime* = rs.getString("D\_TIME");  
  
 *\_withdrawalID* = rs.getString("W\_ID");  
 *\_withdrawal* = rs.getString("WITHD");  
 *\_withdrawalBalance* = rs.getString("W\_BALANCE");  
 *\_withdrawalDate* = rs.getString("W\_DATE");  
 *\_withdrawalTime* = rs.getString("W\_TIME");  
  
 *\_transferID* = rs.getString("T\_ID");  
 *\_accountFrom* = rs.getString("T\_FROM");  
 *\_accountTo* = rs.getString("T\_TO");  
 *\_transferAmount* = rs.getString("T\_AMOUNT");  
 *\_transferBalance* = rs.getString("T\_BALANCE");  
 *\_transferDate* = rs.getString("T\_DATE");  
 *\_transferTime* = rs.getString("T\_TIME");  
  
 String dep = "Deposit: $" + *\_deposit* + " \tdate: " + *\_depositDate* + "\ttime: " + *\_depositTime*;  
 String with = "Withdrawal: $" + *\_withdrawal* + " \tdate: " + *\_withdrawalDate* + "\ttime: " + *\_withdrawalTime*;  
 String trans = "Transferred $" + *\_transferAmount* + " to " + *\_accountTo* + " . current balance: " + *\_transferBalance*;  
  
 if (*\_deposit* != null) {  
 account\_transactions.getItems().addAll( Collections.*singleton*(dep) );  
  
 *HistoryList*.add ( new TransactionsHistory( *\_depositID*, "Deposit", *\_deposit*, *\_depositBalance*, *\_depositDate*, *\_depositTime*) );  
 }  
  
 if (*\_withdrawal* != null) {  
 account\_transactions.getItems().addAll( Collections.*singleton*(with) );  
  
 *HistoryList*.add ( new TransactionsHistory( *\_withdrawalID*, "Withdrawal", *\_withdrawal*, *\_withdrawalBalance*, *\_withdrawalDate*, *\_withdrawalTime*) );  
 }  
 if (*\_transferAmount* != null) {  
 account\_transactions.getItems().addAll( Collections.*singleton*(trans) );  
  
 *HistoryList*.add ( new TransactionsHistory( *\_transferID*, "Transfer", *\_transferAmount*, *\_transferBalance*, *\_transferDate*, *\_transferTime*) );  
 }  
  
 }  
  
 }  
  
 }  
}

***LoginValidator Interface***

package Models;  
  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
public interface LoginValidator {  
  
 static void validate (String uname, String pass, TextField username, TextField password, Text message\_status) {  
  
 if ( uname.length() == 0 ) {  
 username.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("username is required!");  
 RegistrationAlertMessage.*message*(message\_status);  
 }  
 else username.setStyle(null);  
  
 if ( pass.length() < 3 ) {  
 password.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("password is required!");  
 RegistrationAlertMessage.*message*(message\_status);  
 }  
 else password.setStyle(null);  
  
 }  
}

***RegisterValidator Interface***

package Models;  
  
import javafx.scene.control.TextField;  
import javafx.scene.text.Text;  
  
public interface RegisterValidator {  
  
 static void validate (String fname, String lname, String pass, String cty, String contct, TextField firstname, TextField lastname, TextField password, TextField city, TextField contact, Text message\_status) {  
  
 *fname*(fname, lname, firstname, lastname, message\_status);  
  
 if ( pass.length() < 8 ) {  
  
 if ( pass.length() == 0 ) {  
 password.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("field cannot be empty.");  
 AlertMessage.*message*(message\_status);  
 }  
 else password.setStyle(null);  
  
 password.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("password must be more than 8 characters.");  
 AlertMessage.*message*(message\_status);  
 }  
 else password.setStyle(null);  
  
 *fname*(cty, contct, city, contact, message\_status);  
  
 }  
  
 private static void fname (String fname, String lname, TextField firstname, TextField lastname, Text message\_status) {  
 if ( fname.length() == 0 ) {  
 firstname.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("field cannot be empty.");  
 AlertMessage.*message*(message\_status);  
 }  
 else firstname.setStyle(null);  
  
 if ( lname.length() == 0 ) {  
 lastname.setStyle("-fx-border-color: red; -fx-border-width: 2px");  
  
 message\_status.setStyle("-fx-fill: #f66262;");  
 message\_status.setText("field cannot be empty.");  
 RegistrationAlertMessage.*message*(message\_status);  
 }  
 else lastname.setStyle(null);  
 }  
  
}

***RegistrationAlertMessage Class***

package Models;  
  
import javafx.animation.KeyFrame;  
import javafx.animation.KeyValue;  
import javafx.animation.Timeline;  
import javafx.beans.property.DoubleProperty;  
import javafx.scene.text.Text;  
import javafx.util.Duration;  
import org.jetbrains.annotations.NotNull;  
  
public class RegistrationAlertMessage {  
  
 public static void message (@NotNull Text message\_status) {  
 DoubleProperty opacity = message\_status.opacityProperty();  
 Timeline fadeOut =  
 new Timeline (  
 new KeyFrame( Duration.*ZERO*, new KeyValue(opacity, 1) ),  
 new KeyFrame ( Duration.*seconds*(15), new KeyValue(opacity, 0) )  
 );  
 fadeOut.play();  
 }  
}

***ScreenTransition Interface***

package Models;  
  
import javafx.animation.KeyFrame;  
import javafx.animation.KeyValue;  
import javafx.animation.Timeline;  
import javafx.beans.property.DoubleProperty;  
import javafx.scene.layout.Pane;  
import javafx.util.Duration;  
import org.jetbrains.annotations.NotNull;  
  
public interface ScreenTransition {  
 static void fadeIn (@NotNull Pane root) {  
 DoubleProperty opacity = root.opacityProperty();  
 Timeline fadeIn =  
 new Timeline (  
 new KeyFrame ( Duration.*ZERO*, new KeyValue(opacity, 0.8) ),  
 new KeyFrame ( new Duration(800), new KeyValue(opacity, 1.0) )  
 );  
 fadeIn.play();  
 }  
}

***TabViewHelper Class***

package Models;  
  
import Controllers.Login;  
import Database.Connect;  
import javafx.collections.transformation.FilteredList;  
import javafx.collections.transformation.SortedList;  
import javafx.scene.control.TableView;  
import javafx.scene.control.TextField;  
  
import java.sql.\*;  
  
import static Models.Initializer.*HistoryList*;  
  
  
public abstract class TabViewHelper extends Login {  
  
 public static void table\_view (TableView<TransactionsHistory> transactions\_view) { transactions\_view.setItems(*HistoryList*); }  
  
 public static void table\_refresh (TableView<TransactionsHistory> transactions\_view) throws SQLException {  
  
 final String \_SQL = "SELECT d.transac\_id DEP\_ID, d.deposit DEP, d.balance DEP\_BALANCE, d.date DEP\_DATE, d.time DEP\_TIME, w.transac\_id WITH\_ID, " +  
 "w.withdrawal WITHD, w.balance WITH\_BALANCE, w.date WITH\_DATE, w.time WITH\_TIME, t.transfer\_id TRANS\_ID, t.amount TRANS\_AMOUNT," +  
 "t.balance TRANS\_BALANCE, t.date TRANS\_DATE, t.time TRANS\_TIME FROM deposits d FULL JOIN withdrawals w ON d.acc\_id = w.acc\_id " +  
 "FULL JOIN transfers t on d.acc\_id = t.acc\_from WHERE d.acc\_id = '" + *\_accountID* + "'";  
  
 Connection conn = new Connect().connect();  
  
 try (PreparedStatement \_stmt = conn.prepareStatement(\_SQL); ResultSet rs = \_stmt.executeQuery() ) {  
 *HistoryList*.clear();  
  
 while ( rs.next() ) {  
  
 if ( rs.getString("DEP") != null) {  
 *HistoryList*.add (  
 new TransactionsHistory (  
 rs.getString("DEP\_ID"),  
 "Deposit",  
 rs.getString("DEP"),  
 rs.getString("DEP\_BALANCE"),  
 rs.getString("DEP\_DATE"),  
 rs.getString("DEP\_TIME")  
 )  
 );  
 transactions\_view.setItems(*HistoryList*);  
 }  
  
 if ( rs.getString("WITHD") != null) {  
 *HistoryList*.add (  
 new TransactionsHistory (  
 rs.getString("WITH\_ID"),  
 "Withdawal",  
 rs.getString("WITHD"),  
 rs.getString("WITH\_BALANCE"),  
 rs.getString("WITH\_DATE"),  
 rs.getString("WITH\_TIME")  
 )  
 );  
 transactions\_view.setItems(*HistoryList*);  
 }  
  
 if ( rs.getString("TRANS\_AMOUNT") != null) {  
 *HistoryList*.add (  
 new TransactionsHistory (  
 rs.getString("TRANS\_ID"),  
 "Transfer",  
 rs.getString("TRANS\_AMOUNT"),  
 rs.getString("TRANS\_BALANCE"),  
 rs.getString("TRANS\_DATE"),  
 rs.getString("TRANS\_TIME")  
 )  
 );  
 transactions\_view.setItems(*HistoryList*);  
 }  
  
 }  
  
 // add refresh table-view fade out  
 }  
 }  
  
 public static void table\_search (TableView<TransactionsHistory> transactions\_view, TextField searching) {  
 FilteredList<TransactionsHistory> filteredData = new FilteredList<>(*HistoryList*, b -> true);  
  
 searching.textProperty().addListener(  
 ( (observableValue, oldValue, newValue) ->  
 filteredData.setPredicate( TransactionsHistory ->  
 {  
 if ( newValue.isEmpty() || newValue.isBlank() ) return true;  
  
 String searchKeyword = newValue.toLowerCase();  
  
 if ( TransactionsHistory.getTrans\_id().toLowerCase().contains(searchKeyword) ) return true;  
 else if ( TransactionsHistory.getTrans\_type().toLowerCase().contains(searchKeyword) ) return true;  
 else if ( TransactionsHistory.getTrans\_amount().toLowerCase().contains(searchKeyword) ) return true;  
 else if ( TransactionsHistory.getTrans\_balance().toLowerCase().contains(searchKeyword) ) return true;  
 else if ( TransactionsHistory.getTrans\_date().toLowerCase().contains(searchKeyword) ) return true;  
 else return TransactionsHistory.getTrans\_time().toLowerCase().contains(searchKeyword);  
 }  
 )  
 )  
 );  
  
 SortedList<TransactionsHistory> sortData = new SortedList<>(filteredData);  
 sortData.comparatorProperty().bind( transactions\_view.comparatorProperty() );  
  
 transactions\_view.setItems(sortData);  
 }  
  
}

***TransactionsHistory Class***

package Models;  
  
public class TransactionsHistory {  
  
 String trans\_id, trans\_type, trans\_amount, trans\_balance, trans\_date, trans\_time;  
  
 public TransactionsHistory (String trans\_id, String trans\_type, String trans\_amount, String trans\_balance, String trans\_date, String trans\_time) {  
 this.trans\_id = trans\_id;  
 this.trans\_type = trans\_type;  
 this.trans\_amount = trans\_amount;  
 this.trans\_balance = trans\_balance;  
 this.trans\_date = trans\_date;  
 this.trans\_time = trans\_time;  
 }  
  
 public String getTrans\_id () { return trans\_id; }  
 public String getTrans\_type () { return trans\_type; }  
 public String getTrans\_amount () { return trans\_amount; }  
 public String getTrans\_balance () { return trans\_balance; }  
 public String getTrans\_date () { return trans\_date; }  
 public String getTrans\_time () { return trans\_time; }  
  
}

**FUTURE SCOPE OF THE PROJECT**

* The Online Banking System can be enhanced to include some other functionality like bank statement.
* An administrator view can be added to manage the system.
* The system can be expanded to integrate different branches of the bank.
* Integrate other aspects of banking into the system. Some more abstract essential for a fully function-able bank.
* Support for other banks API can be added.