ALL PROJECT FILES

1.DynamicTable.java

```
package application;

import java.sql.Connection;
import java.sql.ResultSet;
import java.util.concurrent.atomic.AtomicLong;

import Dao.DBConnect;
import javafx.application.Platform;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableColumn.CellDataFeatures;
import javafx.scene.control.TableView;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import javafx.util.Callback;

public class DynamicTable {

    // TABLE VIEW AND DATA OBJECT CREATIONS
    @SuppressWarnings("rawtypes")
    private ObservableList<ObservableList> data;
    @SuppressWarnings("rawtypes")
    private TableView tableview;

    // CONNECTION 2 DATABASE
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public void buildData(String sql) {
        tableview = new TableView();
        DBConnect c = new DBConnect();
        data = FXCollections.observableArrayList();
        try {
            Connection conn = c.getConnection();
            // SQL FOR SELECTING DATA
            String SQL = sql;
            // ResultSet object
            ResultSet rs = conn.createStatement().executeQuery(SQL);

            /**********************************
             * TABLE COLUMN ADDED DYNAMICALLY *
             **********************************/
            for (int i = 0; i < rs.getMetaData().getColumnCount(); i++) {

                // We are using non property style for making dynamic table
                final int j = i;
                TableColumn  col  =  new  TableColumn(rs.getMetaData().getColumnName(i  +
1));

                /**
                 * Build an ObservableList for column headings as you iterate thru meta data
                 * setup callback Api for column retrievals, works with call method to return
                 * heading names
                 */
                col.setCellValueFactory(
                        new         Callback<CellDataFeatures<ObservableList,          String>,
ObservableValue<String>>() {

                            public                              ObservableValue<String>
call(CellDataFeatures<ObservableList, String> param) {
                                return                                             new
SimpleStringProperty(param.getValue().get(j).toString());
                            }

                        });
                // add each column name to tableview object
                tableview.getColumns().add(col);
                // display column names to console as they are added to table dynamically
                System.out.println("Column      ["      +      i      +      "]      added      ["      +
rs.getMetaData().getColumnName(i + 1) + "]");
            }

            /********************************************
             * Data added to ObservableList dynamically *
             ********************************************/
            int ridx = 0; // track a row index to display to console added rows to table
            while (rs.next()) {
                // Iterate Row
                ObservableList<String> row = FXCollections.observableArrayList();
                for (int i = 1; i <= rs.getMetaData().getColumnCount(); i++) {
                    // Iterate Column, grab data
                    row.add(rs.getString(i));
                }
                System.out.println("Row [" + (ridx++) + "] added " + row);
                data.add(row);
            }
            // automatically adjust width of columns depending on their content
            tableview.setColumnResizePolicy((param) -> true);
            Platform.runLater(() -> customResize(tableview));

            // add data to tableview object
            tableview.setItems(data);

            Scene secondScene = new Scene(tableview, 600, 400);

            Stage secondStage = new Stage();
            secondStage.setTitle("New Stage");
            secondStage.setScene(secondScene);

            secondStage.show();

            Main.stage.setOnCloseRequest((WindowEvent event1) -> {

                System.out.println("Main window closed");

            });

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Error on Building Data");
        }
    }

    public void customResize(TableView<?> view) {

        AtomicLong width = new AtomicLong();
        view.getColumns().forEach(col -> {
            width.addAndGet((long) col.getWidth());
        });
        double tableWidth = view.getWidth();

        if (tableWidth > width.get()) {
            view.getColumns().forEach(col -> {
                col.setPrefWidth(col.getWidth()    +    ((tableWidth    -    width.get())    /
view.getColumns().size()));
                        });
                }
        }


}
```

2. Main.java

```
package application;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class Main extends Application {

        public static Stage stage; // set global stage object!!!

        @Override
        public void start(Stage primaryStage) {

                try {
                        stage = primaryStage;
                        Parent                                          root                                            =
FXMLLoader.load(getClass().getResource("/views/LoginView.fxml"));
                        Scene scene = new Scene(root);
                        stage.setTitle("Login View");
                        stage.setScene(scene);
                        stage.show();
                } catch (Exception e) {
                        System.out.println("Error occured while inflating view: " + e);
                }
        }

        public static void main(String[] args) {

                launch(args);
        }
}
```

3. Utils.java

```
package application;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Utils {

    public static String encrypt2MD5(String password){
        String hexStr = "";
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            byte[] bytes = md5.digest(password.getBytes("utf-8"));
            StringBuffer stringBuffer = new StringBuffer();
            for (byte b : bytes) {
                int bt = b&0xff;
                if (bt < 16) {
                    stringBuffer.append(0);
                }
                stringBuffer.append(Integer.toHexString(bt));
            }
            hexStr = stringBuffer.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return hexStr;
    }

}
```

4. AdminController.java

```
package controllers;

import Dao.DBConnect;
import application.Utils;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import models.UserModel;
import models.BankModel;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class AdminController {

        // Declare DB objects
        Statement stmt = null;
        Connection conn = null;

        @FXML
        private TextField txtUserName;

        @FXML
        private TextField txtBankName;

        @FXML
        private ComboBox<String> cmbRole;

        @FXML
        private TextField txtAddress;

        @FXML
        private TableView<BankModel> bankTable;

        @FXML
        private TableView<UserModel> userTable;

        @FXML
        private TableColumn<UserModel,String> colUserId;

        @FXML
        private TableColumn<UserModel,String> colUserName;

        @FXML
        private TableColumn<UserModel,String> colUserRole;

        @FXML
        private TableColumn<BankModel,String> colBankId;
```

```
@FXML
private TableColumn<BankModel,String> colBankName;

@FXML
private TableColumn<BankModel,String> colBankAddress;

@FXML
private Label lblBankError;

@FXML
private Label lblUserError;

@FXML
private Label lbUserId;

@FXML
private Label lbBankId;

public AdminController() throws SQLException {
	DBConnect dbConnect = new DBConnect();
	conn = dbConnect.getConnection();
	stmt = conn.createStatement();
}

@FXML
void initialize() throws SQLException {
	colUserId.setCellValueFactory(new PropertyValueFactory("id"));
	colUserName.setCellValueFactory(new PropertyValueFactory("uname"));
	colUserRole.setCellValueFactory(new PropertyValueFactory("admin"));
	colBankId.setCellValueFactory(new PropertyValueFactory("bankId"));
	colBankName.setCellValueFactory(new PropertyValueFactory("bankName"));
	colBankAddress.setCellValueFactory(new PropertyValueFactory("bankAddress"));

	refreshUserData();
	refreshBankData();
	cmbRole.getItems().addAll("Admin", "User");
}


public void addUser(ActionEvent event) {
	System.out.println("Inserting records into the table...");
	String username = txtUserName.getText();
	if(null == username || username.isEmpty()){
		lblUserError.setText("The User Name is Empty!");
```

```
                return;
            }
            String role = cmbRole.getValue();
            if(null == role || role.isEmpty()){
                lblUserError.setText("The role is Empty!");
                return;
            } else {
                if(role.equals("Admin")){
                    role = "1";
                } else {
                    role = "0";
                }
            }
            String password = Utils.encrypt2MD5("123456");
            try{
                int userId = getMaxUserId() + 1;
                String insertSQL = "insert into jpapa_users(id,uname,passwd,admin) values ('" +
userId + "','"+ username + "','" + password + "','" +
                        role + "')";
                stmt.executeUpdate(insertSQL);
                System.out.println("User Record created");

                String insertAccountSQL = "insert into
jpapa_accounts(type,balance,userid,amount) values ('" + "Create" + "','" + "0.00" + "','"
                        + userId + "','" + "0.00" + "')";
                stmt.executeUpdate(insertAccountSQL);
                System.out.println("Account Record created");

                refreshUserData();


            } catch (SQLException e){
                System.out.println("User creation failed");
                e.printStackTrace();
            }



        }

        public void updateUser(ActionEvent event) {
            System.out.println("Updating records into the table...");
            String username = txtUserName.getText();
            if(null == username || username.isEmpty()){
```

```
                        lblUserError.setText("The User Name is Empty!");
                        return;
                }
                String role = cmbRole.getValue();
                if(null == role || role.isEmpty()){
                        lblUserError.setText("The role is Empty!");
                        return;
                }
                String accountId = lbUserId.getText();
                if(null == accountId || accountId.isEmpty()){
                        lblUserError.setText("Please select one!");
                        return;
                }
                try{
                        String updateSQL = "update jpapa_users set uname = '" + username + "', admin =
'" + role + "' where id = '" + accountId + "'";
                        stmt.executeUpdate(updateSQL);
                        refreshUserData();
                        System.out.println("User Record created");
                } catch (SQLException e){
                        System.out.println("User creation failed");
                        e.printStackTrace();
                }
        }

        public void deleteUser(ActionEvent event) {
                String accountId = lbUserId.getText();
                if(null == accountId || accountId.isEmpty()){
                        lblUserError.setText("Please select one!");
                        return;
                }
                try{
                        String updateSQL = "delete from jpapa_users where id = '" + accountId + "'";
                        stmt.executeUpdate(updateSQL);
                        System.out.println("User Record deleted");
                        txtUserName.setText("");
                        cmbRole.setValue("");
                        refreshUserData();
                } catch (SQLException e){
                        System.out.println("User Record delete failed");
                        e.printStackTrace();
                }
        }
```

```
	public void cancelUser(ActionEvent event) {
		txtUserName.setText("");
		cmbRole.setValue("");
	}

	public void addBank(ActionEvent event) {
		System.out.println("Inserting records into the table...");
		String bankName = txtBankName.getText();
		String address = txtAddress.getText();
		if(null == bankName || bankName.isEmpty()){
			lblBankError.setText("The bank name is Empty!");
			return;
		}
		if(null == address || address.isEmpty()){
			lblBankError.setText("The address is Empty!");
			return;
		}

		try{
			String insertSQL = "insert into jpapa_bank(name,address) values ('" + bankName +
"','" + address + "')";
			stmt.executeUpdate(insertSQL);
			System.out.println("Bank Record created");
			refreshBankData();
		} catch (SQLException e){
			System.out.println("Bank Record creation failed");
			e.printStackTrace();
		}

	}

	public void updateBank(ActionEvent event) {
		System.out.println("Updating records into the table...");
		String bankName = txtBankName.getText();
		String address = txtAddress.getText();
		if(null == bankName || bankName.isEmpty()){
			lblBankError.setText("The bank name is Empty!");
			return;
		}
		if(null == address || address.isEmpty()){
			lblBankError.setText("The address is Empty!");
			return;
		}
```

```
        String bankId = lbBankId.getText();
        if(null == bankId || bankId.isEmpty()){
            lblBankError.setText("Please select one!");
            return;
        }
        try{
            String updateSQL = "update jpapa_bank set name = '" + bankName + "',address =
'" + address + "' where id = '" + bankId + "'";
            stmt.executeUpdate(updateSQL);
            System.out.println("Bank Record updated");
            refreshBankData();
        } catch (SQLException e){
            System.out.println("Bank Record update failed");
            e.printStackTrace();
        }
    }

    public void deleteBank(ActionEvent event) {
        String bankId = lbBankId.getText();
        if(null == bankId || bankId.isEmpty()){
            lblBankError.setText("Please select one!");
            return;
        }
        try{
            String updateSQL = "delete from jpapa_bank where id = '" + bankId + "'";
            stmt.executeUpdate(updateSQL);
            System.out.println("Bank Record deleted");
            txtBankName.setText("");
            txtAddress.setText("");
            refreshBankData();
        } catch (SQLException e){
            System.out.println("Bank Record delete failed");
            e.printStackTrace();
        }
    }

    public void cancelBank(ActionEvent event) {
        txtBankName.setText("");
        txtAddress.setText("");
    }

    public void selectUser(MouseEvent mouseEvent) {
        UserModel account = userTable.getSelectionModel().getSelectedItem();
        txtUserName.setText(account.getUname());
        if(account.getAdmin().equals("Admin")){
                cmbRole.setValue("Admin");
        } else {
                cmbRole.setValue("User");
        }
        lbUserId.setText(account.getId());
}

public void selectBank(MouseEvent mouseEvent) {
        BankModel bank = bankTable.getSelectionModel().getSelectedItem();
        txtBankName.setText(bank.getBankName());
        txtAddress.setText(bank.getBankAddress());
        lbBankId.setText(String.valueOf(bank.getBankId()));
}

private void refreshUserData() throws SQLException {
        userTable.getItems().clear();
        String accountQuerySQL = "SELECT id,uname,admin FROM jpapa_users";
        ObservableList<UserModel> accountList = FXCollections.observableArrayList();
        ResultSet accountRs = stmt.executeQuery(accountQuerySQL);
        while (accountRs.next()) {
                UserModel account = new UserModel();
                String id = accountRs.getString("id");
                String userName = accountRs.getString("uname");
                String admin = accountRs.getString("admin");
                if(admin.equals("1")){
                        account.setAdmin("Admin");
                } else {
                        account.setAdmin("User");
                }
                account.setId(id);
                account.setUname(userName);
                accountList.add(account);
        }
        userTable.setItems(accountList);
}

private void refreshBankData() throws SQLException {
        bankTable.getItems().clear();
        ObservableList<BankModel> bankList = FXCollections.observableArrayList();
        String bankQuerySQL = "SELECT id,name,address FROM jpapa_bank";
        ResultSet bankRs = stmt.executeQuery(bankQuerySQL);
        while (bankRs.next()) {
                int id = bankRs.getInt("id");
                String name = bankRs.getString("name");
                String address = bankRs.getString("address");
                BankModel bank = new BankModel(id,name,address);
                bankList.add(bank);
            }
            bankTable.setItems(bankList);
        }

        private int getMaxUserId() throws SQLException {
            String querySQL = "SELECT MAX(id) maxId FROM jpapa_users";
            ResultSet rs = stmt.executeQuery(querySQL);
            int id = 0;
            while (rs.next()) {
                id = rs.getInt("maxId");
            }
            return id;
        }
}
```

5. ClientController.java

```
package controllers;

import Dao.DBConnect;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseEvent;
import models.AccountModel;
import models.BankModel;
import models.UserModel;

import java.net.URL;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ResourceBundle;

public class ClientController implements Initializable {
```

```
private Statement stmt = null;
private Connection conn = null;

private static int userId;

@FXML
public TableColumn<AccountModel,String> colId;
@FXML
public TableColumn<AccountModel,String> colAmount;
@FXML
public TableColumn<AccountModel,String> colType;
@FXML
public TableView<AccountModel> accountTableView;
@FXML
public TextField txtAmount;
@FXML
public ComboBox<String> cmbType;
@FXML
public TextField txtBalance;
@FXML
public Button btnCancel;
@FXML
public Label accountId;
@FXML
public Button btnConfirm;
@FXML
public Label lblError;

public ClientController() throws SQLException {
        DBConnect dbConnect = new DBConnect();
        conn = dbConnect.getConnection();
        stmt = conn.createStatement();
}

public void initialize(URL location, ResourceBundle resources) {
        colId.setCellValueFactory(new PropertyValueFactory("id"));
        colAmount.setCellValueFactory(new PropertyValueFactory("type"));
        colType.setCellValueFactory(new PropertyValueFactory("amount"));

        refreshAccData();
        cmbType.getItems().addAll("Withdraw", "Deposit");
}
```

```
	public void confirm(ActionEvent event) {
		System.out.println("Updating records into the table...");
		String amount = txtAmount.getText();
		String type = cmbType.getValue();
		if(null == amount || amount.isEmpty()){
			lblError.setText("The amount is Empty!");
			return;
		}
		if(null == type || type.isEmpty()){
			lblError.setText("The type is Empty!");
			return;
		}

		String id = accountId.getText();
		if(null == id || id.isEmpty()){
			lblError.setText("Please select one!");
			return;
		}
		String balanceTxt = txtBalance.getText();
		if(null == balanceTxt || balanceTxt.isEmpty()){
			balanceTxt = "0";
		}
		Double balanceD = Double.valueOf(balanceTxt);
		if (type.equals("Withdraw")){
			balanceD = balanceD - Double.valueOf(amount);
		} else {
			balanceD = balanceD + Double.valueOf(amount);
		}
		try{
			String insertSQL = "insert into jpapa_accounts(type,balance,userid,amount) values
('" + type + "','" + balanceD + "','"
					+ userId + "','" + amount + "')";
			stmt.executeUpdate(insertSQL);
			System.out.println("Account Record updated");
			refreshAccData();
		} catch (SQLException e){
			System.out.println("Account Record update failed");
			e.printStackTrace();
		}
		txtBalance.setText(String.valueOf(balanceD));

	}

	public void cancel(ActionEvent event) {
```

```
            txtAmount.setText("");
            cmbType.setValue("");
        }

        public void getAccountItem(MouseEvent mouseEvent) {
            AccountModel                              accountModel                              =
accountTableView.getSelectionModel().getSelectedItem();
            accountId.setText(String.valueOf(accountModel.getId()));
        }

        private void refreshAccData(){
            accountTableView.getItems().clear();
            ObservableList<AccountModel> accountModels = FXCollections.observableArrayList();
            String  accQuerySQL  =  "SELECT  id,type,balance,userid,amount  FROM  jpapa_accounts
WHERE userid = " + "'" +userId + "'" ;
            ResultSet accRs = null;
            try {
                accRs = stmt.executeQuery(accQuerySQL);
                while (accRs.next()) {
                    int id = accRs.getInt("id");
                    String type = accRs.getString("type");
                    String amount = accRs.getString("amount");
                    String balance = accRs.getString("balance");
                    AccountModel  accountModel  =  new  AccountModel(id,     type,  amount,
balance);
                    accountModels.add(accountModel);
                    txtBalance.setText(String.valueOf(balance));
                }
                accountTableView.setItems(accountModels);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        public static void setUserId(int user_id) {
            userId = user_id;
            System.out.println("Welcome id " + userId);
        }
}
```

6. LoginController.java

```
package controllers;

import application.Main;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import models.LoginModel;

public class LoginController {

	@FXML
	private TextField txtUsername;

	@FXML
	private PasswordField txtPassword;

	@FXML
	private Label lblError;

	private LoginModel model;

	public LoginController() {
		model = new LoginModel();
	}

	public void login() {

		lblError.setText("");
		String username = this.txtUsername.getText();
		String password = this.txtPassword.getText();

		// Validations
		if (username == null || username.trim().equals("")) {
			lblError.setText("Username Cannot be empty or spaces");
			return;
		}
		if (password == null || password.trim().equals("")) {
			lblError.setText("Password Cannot be empty or spaces");
			return;
		}
		if (username == null || username.trim().equals("") && (password == null || 
password.trim().equals(""))) {
			lblError.setText("User name / Password Cannot be empty or spaces");
```

```
            return;
        }

        // authentication check
        checkCredentials(username, password);

    }

    public void checkCredentials(String username, String password) {
        Boolean isValid = model.getCredentials(username, password);
        if (!isValid) {
            lblError.setText("User does not exist!");
            return;
        }
        try {
            Parent root;
            if (model.isAdmin() && isValid) {
                // If user is admin, inflate admin view

                root =    FXMLLoader.load(getClass().getResource("/views/AdminView.fxml"));
                Main.stage.setTitle("Admin View");

            } else {
                // If user is customer, inflate customer view
                // ***Set user ID acquired from db****
                int user_id = model.getId();
                ClientController.setUserId(user_id);
                root =    FXMLLoader.load(getClass().getResource("/views/ClientView.fxml"));
                Main.stage.setTitle("Client View");
            }

            Scene scene = new Scene(root);
            Main.stage.setScene(scene);

        } catch (Exception e) {
            System.out.println("Error occured while inflating view: " + e);
        }

    }
}
```

7. DBConnect.java

```
package Dao;
```

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

public class DBConnect    {

        protected Connection connection;
        public Connection getConnection() {
                return connection;
        }

        private static String url = "jdbc:mysql://www.papademas.net:3307/510fp";
        private static String username = "fp510";
        private static String password = "510";

        public DBConnect() {
                try {
                        connection = DriverManager.getConnection(url, username, password);
                } catch (SQLException e) {
                        System.out.println("Error creating connection to database: " + e);
                        System.exit(-1);
                }
        }


}
```

8. AccountModel.java

```
package models;

public class AccountModel {

        private int id;

        private int userId;

        private String type;

        private String amount;

        private String balance;

        public AccountModel(int id, String type, String amount, String balance) {
```

```
        this.id = id;
        this.type = type;
        this.amount = amount;
        this.balance = balance;
}

public AccountModel() {

}

public int getId() {
        return id;
}

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

public int getUserId() {
        return userId;
}

public void setUserId(int userId) {
        this.userId = userId;
}

public String getType() {
        return type;
}

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

public String getAmount() {
        return amount;
}

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

public String getBalance() {
        return balance;
```

```
	}

	public void setBalance(String balance) {
		this.balance = balance;
	}
}
```

9.BankModel.java

```
package models;

public class BankModel {

	int bankId;
	String bankName;
	String bankAddress;

	public int getBankId() {
		return bankId;
	}

	public void setBankId(int bankId) {
		this.bankId = bankId;
	}

	public String getBankName() {
		return bankName;
	}

	public void setBankName(String bankName) {
		this.bankName = bankName;
	}

	public String getBankAddress() {
		return bankAddress;
	}

	public void setBankAddress(String bankAddress) {
		this.bankAddress = bankAddress;
	}

	public BankModel(Integer id, String name, String address) {
		this.bankId = id;
		this.bankName = name;
		this.bankAddress = address;
	}

	public BankModel() {
		// TODO Auto-generated constructor stub
	}

	public String toString() {

		String bankData = "Bank Account Affiliation:\nID- " + bankId;
		bankData += "\nName- " + bankName;
		bankData += "\nAddress-" + bankAddress + "\n";

		return bankData;
	}

}
```

10.LoginModel.java

```
package models;

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import Dao.DBConnect;
import application.Utils;

public class LoginModel extends DBConnect {

	private Boolean admin;
	private int id;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public Boolean isAdmin() {
		return admin;
	}
	public void setAdmin(Boolean admin) {
		this.admin = admin;
	}

	public Boolean getCredentials(String username, String password){
			password = Utils.encrypt2MD5(password);
			String query = "SELECT * FROM jpapa_users WHERE uname = ? and passwd = ?;";
			try(PreparedStatement stmt = connection.prepareStatement(query)) {
			     stmt.setString(1, username);
			     stmt.setString(2, password);
			     ResultSet rs = stmt.executeQuery();
			       if(rs.next()) {
				       setId(rs.getInt("id"));
				       setAdmin(rs.getBoolean("admin"));
				       return true;
			       }
			  }catch (SQLException e) {
			       e.printStackTrace();
			  }
			return false;
	}

}//end class
```

11.UserModel.java

```
package models;

public class UserModel {

	private String id;
	private String uname;
	private String password;
	private String admin;
	private String balance;

	public UserModel(){

	}

	public UserModel(String id, String uname, String password, String admin, String balance) {
		this.id = id;
		this.uname = uname;
		this.password = password;
		this.admin = admin;
		this.balance = balance;
	}
```

```
public String getId() {
        return id;
}

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

public String getUname() {
        return uname;
}

public void setUname(String uname) {
        this.uname = uname;
}

public String getPassword() {
        return password;
}

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

public String getAdmin() {
        return admin;
}

public void setAdmin(String admin) {
        this.admin = admin;
}

public String getBalance() {
        return balance;
}

public void setBalance(String balance) {
        this.balance = balance;
}

@Override
public String toString() {
        return "Account{" +
                        "id='" + id + '\'' +
                        ", uname='" + uname + '\'' +
                        ", password='" + password + '\'' +
                        ", admin='" + admin + '\'' +
                        ", balance='" + balance + '\'' +
                        '}';
        }
}
```

12. AdminView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.AdminController">
   <children>
      <TabPane prefHeight="406.0" prefWidth="600.0" tabClosingPolicy="UNAVAILABLE">
        <tabs>
          <Tab text="View User">
            <content>
                <VBox minHeight="0.0" minWidth="0.0" prefHeight="180.0"
prefWidth="200.0">
                   <children>
                      <VBox prefHeight="200.0" prefWidth="100.0">
                         <children>
                            <TableView fx:id="userTable"
onMouseClicked="#selectUser" prefHeight="249.0" prefWidth="600.0">
                              <columns>
                                <TableColumn fx:id="colUserId" prefWidth="75.0"
text="Id" />
                                <TableColumn fx:id="colUserName" prefWidth="75.0"
text="Name" />
                                  <TableColumn fx:id="colUserRole" prefWidth="75.0"
text="Role" />
                              </columns>
                            </TableView>
                         </children>
                      </VBox>
```

```
                        <VBox prefHeight="22.0" prefWidth="565.0">
                            <children>
                                <VBox prefHeight="150.0" prefWidth="600.0">
                                    <children>
                                        <GridPane>
                                           <columnConstraints>
                                              <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                              <ColumnConstraints hgrow="SOMETIMES"
minWidth="10.0" prefWidth="100.0" />
                                           </columnConstraints>
                                           <rowConstraints>
                                              <RowConstraints minHeight="10.0"
prefHeight="30.0" vgrow="SOMETIMES" />
                                           </rowConstraints>
                                            <children>
                                                <AnchorPane prefHeight="200.0"
prefWidth="200.0">
                                                    <children>
                                                        <Label layoutX="41.0" layoutY="8.0"
text="Name:" />
                                                        <TextField fx:id="txtUserName"
layoutX="98.0" layoutY="4.0" />
                                                        <Label fx:id="lbUserId"
layoutX="18.0" layoutY="8.0" text="Label" visible="false" />
                                                    </children>
                                                </AnchorPane>
                                                <AnchorPane prefHeight="200.0"
prefWidth="200.0" GridPane.columnIndex="1">
                                                    <children>
                                                        <Label layoutX="21.0" layoutY="8.0"
text="Role:" />
                                                        <ComboBox fx:id="cmbRole"
layoutX="75.0" layoutY="3.0" prefWidth="150.0" />
                                                    </children>
                                                </AnchorPane>
                                            </children>
                                        </GridPane>
                                    </children>
                                    <VBox.margin>
                                        <Insets top="20.0" />
                                    </VBox.margin></VBox>
                                <VBox prefHeight="200.0" prefWidth="100.0">
                                    <children>
```

```
                                                <AnchorPane prefHeight="0.0" prefWidth="570.0">
                                                    <children>
                                                        <Button layoutX="68.0" layoutY="22.0"
mnemonicParsing="false" onAction="#addUser" text="Add" />
                                                        <Button layoutX="184.0" layoutY="22.0"
mnemonicParsing="false" onAction="#updateUser" text="Update" />
                                                        <Button layoutX="308.0" layoutY="22.0"
mnemonicParsing="false" onAction="#deleteUser" text="Delete" />
                                                        <Button layoutX="426.0" layoutY="22.0"
mnemonicParsing="false" onAction="#cancelUser" text="Cancel" />
                                                    </children>
                                                </AnchorPane>
                                            </children>
                                            <VBox.margin>
                                                <Insets top="20.0" />
                                            </VBox.margin></VBox>
                                        <VBox alignment="CENTER" prefHeight="73.0"
prefWidth="600.0">
                                            <children>
                                                 <Label fx:id="lblUserError" prefHeight="24.0"
prefWidth="313.0" text="" textFill="#f71702">
                                                    <font>
                                                        <Font size="18.0" />
                                                    </font></Label>
                                            </children></VBox>
                                    </children></VBox>
                                </children></VBox>
                        </content>
                    </Tab>
                    <Tab text="View Bank">
                        <content>
                            <VBox minHeight="0.0" minWidth="0.0" prefHeight="180.0"
prefWidth="200.0">
                                <children>
                                    <VBox prefHeight="200.0" prefWidth="100.0">
                                        <children>
                                            <TableView fx:id="bankTable"
onMouseClicked="#selectBank" prefHeight="200.0" prefWidth="200.0">
                                                <columns>
                                                    <TableColumn fx:id="colBankId"
prefWidth="75.0" text="Id" />
                                                    <TableColumn fx:id="colBankName"
prefWidth="75.0" text="Name" />
                                                    <TableColumn fx:id="colBankAddress"
```

```
prefWidth="300.0" text="Address" />
                                        </columns>
                                    </TableView>
                                </children>
                            </VBox>
                            <VBox prefHeight="200.0" prefWidth="100.0">
                                <children>
                                    <VBox prefHeight="150.0" prefWidth="600.0">
                                        <children>
                                            <GridPane>
                                                <columnConstraints>
                                                    <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                                                    <ColumnConstraints
hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                                                </columnConstraints>
                                                <rowConstraints>
                                                    <RowConstraints  minHeight="10.0"
prefHeight="30.0" vgrow="SOMETIMES" />
                                                </rowConstraints>
                                                <children>
                                                    <AnchorPane     prefHeight="200.0"
prefWidth="200.0">
                                                        <children>
                                                            <Label       layoutX="41.0"
layoutY="8.0" text="Name:" />
                                                            <TextField
fx:id="txtBankName" layoutX="98.0" layoutY="4.0" />
                                                  <Label                 fx:id="lbBankId"
layoutX="14.0" layoutY="7.0" text="Label" visible="false" />
                                                        </children>
                                                    </AnchorPane>
                                                    <AnchorPane     prefHeight="200.0"
prefWidth="200.0" GridPane.columnIndex="1">
                                                        <children>
                                                            <Label        layoutY="5.0"
prefHeight="15.0" prefWidth="57.0" text="Address:" />
                                                  <TextField             fx:id="txtAddress"
layoutX="59.0" layoutY="-1.0" />
                                                        </children>
                                                    </AnchorPane>
                                                </children>
                                            </GridPane>
                                        </children>
```

```
                                                            <VBox.margin>
                                                                  <Insets top="20.0" />
                                                            </VBox.margin></VBox>
                                                      <VBox prefHeight="200.0" prefWidth="100.0">
                                                            <children>
                                                                  <AnchorPane                    prefHeight="200.0"
prefWidth="200.0">
                                                                        <children>
                                                                              <Button                   layoutX="68.0"
layoutY="22.0" mnemonicParsing="false" onAction="#addBank" text="Add" />
                                                                              <Button                  layoutX="184.0"
layoutY="22.0" mnemonicParsing="false" onAction="#updateBank" text="Update" />
                                                                              <Button                  layoutX="308.0"
layoutY="22.0" mnemonicParsing="false" onAction="#deleteBank" text="Delete" />
                                                                              <Button                  layoutX="426.0"
layoutY="22.0" mnemonicParsing="false" onAction="#cancelBank" text="Cancel" />
                                                                        </children>
                                                                  </AnchorPane>
                                                            </children>
                                                            <VBox.margin>
                                                                  <Insets top="20.0" />
                                                            </VBox.margin></VBox>
                                                      <VBox      alignment="CENTER"      prefHeight="200.0"
prefWidth="100.0">
                                                            <children>
                                                                  <Label                          fx:id="lblBankError"
prefHeight="24.0" prefWidth="297.0" text="" textFill="#ff0101">
                                                              <font>
                                                                  <Font size="18.0" />
                                                              </font></Label>
                                                            </children></VBox>
                                                </children></VBox>
                                    </children></VBox>
                        </content>
                  </Tab>
              </tabs>
          </TabPane>
      </children>
</VBox>
```

13. ClientView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
```

```
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox  maxHeight="-Infinity"  maxWidth="-Infinity"  minHeight="-Infinity"  minWidth="-Infinity"
prefHeight="400.0"                prefWidth="600.0"                xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.ClientController">
    <children>
        <VBox prefHeight="407.0" prefWidth="576.0">
            <children>
                <TableView      fx:id="accountTableView"      onMouseClicked="#getAccountItem"
prefHeight="278.0" prefWidth="596.0">
                    <columns>
                        <TableColumn fx:id="colId" prefWidth="75.0" text="id" />
                        <TableColumn fx:id="colAmount" prefWidth="75.0" text="Amount" />
                          <TableColumn fx:id="colType" prefWidth="75.0" text="Type" />
                    </columns>
                </TableView>
            </children></VBox>
        <VBox prefHeight="200.0" prefWidth="100.0">
            <children>
                <VBox prefHeight="200.0" prefWidth="100.0">
                    <children>
                        <GridPane alignment="CENTER" prefHeight="144.0" prefWidth="600.0">
                            <columnConstraints>
                                <ColumnConstraints       hgrow="SOMETIMES"       minWidth="10.0"
prefWidth="100.0" />
                                <ColumnConstraints       hgrow="SOMETIMES"       minWidth="10.0"
prefWidth="100.0" />
                            </columnConstraints>
                            <rowConstraints>
                                <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
                                <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
                                <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
                                  <RowConstraints          minHeight="10.0"          prefHeight="30.0"
vgrow="SOMETIMES" />
                            </rowConstraints>
                             <children>
                                  <Label          alignment="CENTER_RIGHT"          prefHeight="15.0"
prefWidth="300.0" text="Balance:">
                                    <padding>
```

```
                              <Insets right="20.0" />
                           </padding></Label>
                        <TextField fx:id="txtBalance" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefWidth="150.0" GridPane.columnIndex="1" />
                        <Label alignment="CENTER_RIGHT" prefHeight="15.0" prefWidth="304.0" text="Amount:" GridPane.rowIndex="1">
                           <padding>
                              <Insets right="20.0" />
                           </padding></Label>
                        <Label alignment="CENTER_RIGHT" contentDisplay="CENTER" prefHeight="15.0" prefWidth="301.0" text="Type:" GridPane.rowIndex="2">
                           <opaqueInsets>
                              <Insets />
                           </opaqueInsets>
                           <padding>
                              <Insets right="20.0" />
                           </padding></Label>
                        <TextField fx:id="txtAmount" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefWidth="150.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                        <ComboBox fx:id="cmbType" prefWidth="150.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />
                        <Button fx:id="btnCancel" mnemonicParsing="false" onAction="#cancel" text="Cancel" GridPane.columnIndex="1" GridPane.rowIndex="3">
                           <GridPane.margin>
                              <Insets left="30.0" />
                           </GridPane.margin></Button>
                        <VBox alignment="CENTER_RIGHT" prefHeight="200.0" prefWidth="100.0" GridPane.rowIndex="3">
                           <children>
                              <Button fx:id="btnConfirm" alignment="CENTER_RIGHT" mnemonicParsing="false" onAction="#confirm" text="Confirm" />
                           </children>
                           <padding>
                              <Insets right="30.0" />
                           </padding>
                        </VBox>
                        <Label fx:id="accountId" text="Label" visible="false" GridPane.rowIndex="2" />
                     </children>
                  </GridPane>
               </children>
            </VBox>
```

```
                    <VBox alignment="CENTER" prefHeight="61.0" prefWidth="600.0">
                        <children>
                            <Label fx:id="lblError" text="" />
                        </children></VBox>
                </children>
            </VBox>
        </children>
</VBox>
```

14. LoginView.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>
<?import java.net.URL?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
        prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/10.0.2-internal"
        xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.LoginController"
        styleClass="bgVBox">
    <stylesheets>
        <URL value="@styles.css" />
    </stylesheets>
    <children>
        <VBox prefHeight="200.0" prefWidth="100.0">
            <children>
                <Label     alignment="CENTER"     contentDisplay="CENTER"     prefHeight="133.0"
prefWidth="600.0" text="Welcome">
                    <font>
                        <Font size="36.0" />
                    </font>
                </Label>
            </children>
        </VBox>
        <VBox prefHeight="200.0" prefWidth="100.0">
            <children>
                <GridPane prefHeight="133.0" prefWidth="600.0">
                  <columnConstraints>
                    <ColumnConstraints          hgrow="SOMETIMES"          maxWidth="295.0"
minWidth="10.0" prefWidth="261.33333333333337" />
                    <ColumnConstraints          hgrow="SOMETIMES"          maxWidth="363.0"
minWidth="10.0" prefWidth="339.33333333333326" />
```

```
                    </columnConstraints>
                    <rowConstraints>
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
                      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
/>
                    </rowConstraints>
                     <children>
                        <Label alignment="CENTER_RIGHT" prefHeight="48.0" prefWidth="301.0"
text="User Name:">
                           <font>
                              <Font size="18.0" />
                           </font>
                           <GridPane.margin>
                              <Insets right="20.0" />
                           </GridPane.margin>
                        </Label>
                        <Label alignment="CENTER_RIGHT" prefHeight="24.0" prefWidth="301.0"
text="Password:" GridPane.rowIndex="1">
                           <font>
                              <Font size="18.0" />
                           </font>
                           <GridPane.margin>
                              <Insets right="20.0" />
                           </GridPane.margin>
                        </Label>
                        <PasswordField fx:id="txtPassword" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="30.0"
prefWidth="170.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />
                        <TextField fx:id="txtUsername" maxHeight="-Infinity"
maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="30.0"
prefWidth="170.0" GridPane.columnIndex="1" />
                     </children>
                  </GridPane>
               </children>
            </VBox>
            <VBox alignment="TOP_CENTER" prefHeight="200.0" prefWidth="100.0">
               <children>
                  <VBox alignment="CENTER" prefHeight="200.0" prefWidth="600.0">
                     <children>
                        <Button alignment="CENTER" contentDisplay="CENTER"
mnemonicParsing="false" onAction="#login" prefHeight="38.0" prefWidth="94.0"
text="Submit">
                           <opaqueInsets>
```

```
                                        <Insets />
                                    </opaqueInsets>
                                    <VBox.margin>
                                        <Insets />
                                    </VBox.margin>
                                </Button>
                            </children>
                        </VBox>
                        <VBox alignment="CENTER" prefHeight="200.0" prefWidth="100.0">
                            <children>
                                <Label fx:id="lblError" text="" />
                            </children></VBox>
                    </children>
                    <opaqueInsets>
                        <Insets />
                    </opaqueInsets>
                    <VBox.margin>
                        <Insets />
                    </VBox.margin>
                    <padding>
                        <Insets top="0" />
                    </padding></VBox>
        </children>
</VBox>
```

15.Styles.css

```
.bgVBox{
        -fx-background-image: url(bg.jpg);
        -fx-background-repeat: stretch;
        -fx-background-position: center center;
}
```