

**PROJECT REPORT**

**PROJECT TITLE: ATMAPP**

**PRESENTED BY: Sagar Dhondge**

**INTERN ID: VN-JD-4W289**

**1. Introduction**

The ATM Simulator Application is a GUI-based system designed to replicate the functionalities of an Automated Teller Machine (ATM). It allows users to perform banking transactions such as deposits, withdrawals, and balance inquiries in a user-friendly environment. The application is built using Java with Swing for the graphical user interface and integrates MySQL as the database through JDBC. In this app if we are not register then we can register ourself and further do login in the app once we login we will see three options that is withdraw deposit and balance once this is done we can do logout and go to the login page back. The login or register data is to be saved in the MYSQL Database.

**2. Technologies Used**

* **Java:** Core programming language used for application development.
* **Java Swing:** Used for creating the graphical user interface (GUI).
* **JDBC (Java Database Connectivity):** To establish a connection between Java and MySQL.
* **MySQL:** Relational database management system used for storing user accounts and transaction details.
* **AWT (Abstract Window Toolkit):** Used for basic UI components and event handling.

**3. Objective and Scope**

**Objectives:**

* To simulate an ATM interface that allows users to perform banking operations.
* To implement user authentication and secure banking transactions.
* To maintain a database of user accounts and transaction history.

**Scope:**

* Users can perform actions like deposit, withdraw, and check balance.
* User authentication is managed using a PIN-based login system.
* Future enhancements may include fund transfers and mini-statements.

**4. System Design**

**4.1 Architecture Pattern: MVC (Model-View-Controller)**

* **Model:** Handles database operations related to users and transactions.
* **View:** Swing-based GUI for user interaction.
* **Controller:** Manages user input, authentication, and banking operations.

**4.2 Flow of the Application**

1. User enters card details and PIN for authentication.
2. Upon successful authentication, the user is presented with transaction options.
3. The user can perform a deposit, withdrawal, or check balance.
4. The application updates the database and displays results accordingly.

**5. Implementation**

**5.1 User Interface**

* **Login Screen:** Allows users to enter account number and PIN.
* **Main Menu:** Provides options for withdrawal, deposit, and balance inquiry.
* **Transaction Screens:** Displays input fields and buttons for performing transactions.
* **Result Display:** Shows transaction success or failure messages.

**5.2 Database Integration**

* **Connection:** Uses JDBC to connect with the MySQL database.
* **Tables:**
  + users (id, username, Password, balance)
* **CRUD Operations:**
  + **Create** new users and transactions.
  + **Read** user details and transaction history.
  + **Update** account balances.

**6.Code Snippet**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.sql.\*;  
  
public class ATMApp {  
  
 private JFrame frame;  
 private JTextField usernameField;  
 private JPasswordField passwordField;  
 private DBHelper dbHelper;  
  
 public static void main(String[] args) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 ATMApp window = new ATMApp();  
 window.frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
  
 public ATMApp() {  
 dbHelper = new DBHelper();  
 initialize();  
 }  
  
 private void initialize() {  
 frame = new JFrame("ATM Simulator");  
 frame.setBounds(100, 100, 500, 400);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setResizable(false);  
 frame.getContentPane().setBackground(new Color(50, 50, 50));  
  
 showLoginPanel();  
 }  
  
 private void showLoginPanel() {  
 frame.getContentPane().removeAll();  
  
 // Login Panel  
 JPanel loginPanel = new JPanel();  
 loginPanel.setBackground(new Color(50, 50, 50));  
 frame.getContentPane().add(loginPanel, BorderLayout.*CENTER*);  
 loginPanel.setLayout(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(10, 10, 10, 10);  
  
 // Username label and field  
 JLabel lblUsername = new JLabel("Username:");  
 lblUsername.setForeground(Color.*WHITE*);  
 lblUsername.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 loginPanel.add(lblUsername, gbc);  
  
 usernameField = new JTextField(20);  
 usernameField.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 1;  
 gbc.gridy = 0;  
 loginPanel.add(usernameField, gbc);  
  
 // Password label and field  
 JLabel lblPassword = new JLabel("Password:");  
 lblPassword.setForeground(Color.*WHITE*);  
 lblPassword.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 loginPanel.add(lblPassword, gbc);  
  
 passwordField = new JPasswordField(20);  
 passwordField.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 1;  
 gbc.gridy = 1;  
 loginPanel.add(passwordField, gbc);  
  
 // Login Button  
 JButton btnLogin = new JButton("Login");  
 btnLogin.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnLogin.setBackground(new Color(0, 153, 51));  
 btnLogin.setForeground(Color.*WHITE*);  
 btnLogin.setFocusPainted(false);  
 btnLogin.setPreferredSize(new Dimension(150, 40));  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 gbc.gridwidth = 2;  
 loginPanel.add(btnLogin, gbc);  
  
 // Register Button  
 JButton btnRegister = new JButton("Register");  
 btnRegister.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnRegister.setBackground(new Color(0, 102, 204));  
 btnRegister.setForeground(Color.*WHITE*);  
 btnRegister.setFocusPainted(false);  
 btnRegister.setPreferredSize(new Dimension(150, 40));  
 gbc.gridx = 0;  
 gbc.gridy = 3;  
 gbc.gridwidth = 2;  
 loginPanel.add(btnRegister, gbc);  
  
 // Message Label  
 JLabel lblMessage = new JLabel("");  
 lblMessage.setForeground(Color.*RED*);  
 lblMessage.setFont(new Font("Arial", Font.*ITALIC*, 14));  
 gbc.gridx = 0;  
 gbc.gridy = 4;  
 gbc.gridwidth = 2;  
 loginPanel.add(lblMessage, gbc);  
  
 // ActionListener for Login button  
 btnLogin.addActionListener(e -> {  
 String username = usernameField.getText();  
 String password = new String(passwordField.getPassword());  
  
 if (dbHelper.authenticate(username, password)) {  
 showATMMenu(username);  
 } else {  
 lblMessage.setText("Invalid credentials! Please try again.");  
 }  
 });  
  
 // ActionListener for Register button  
 btnRegister.addActionListener(e -> showRegistrationForm());  
  
 frame.revalidate();  
 frame.repaint();  
 }  
  
 private void showRegistrationForm() {  
 // Create a new JFrame for registration  
 JFrame registerFrame = new JFrame("Register");  
 registerFrame.setBounds(100, 100, 500, 400); // Increased window size  
 registerFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
 registerFrame.setResizable(false);  
 registerFrame.getContentPane().setBackground(new Color(50, 50, 50));  
  
 JPanel panel = new JPanel();  
 panel.setBackground(new Color(50, 50, 50));  
 registerFrame.getContentPane().add(panel, BorderLayout.*CENTER*);  
 panel.setLayout(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(10, 10, 10, 10);  
  
 // Username field  
 JLabel lblUsername = new JLabel("Username:");  
 lblUsername.setForeground(Color.*WHITE*);  
 lblUsername.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 panel.add(lblUsername, gbc);  
  
 JTextField regUsernameField = new JTextField();  
 regUsernameField.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 regUsernameField.setPreferredSize(new Dimension(300, 40)); // Increased size  
 gbc.gridx = 1;  
 gbc.gridy = 0;  
 panel.add(regUsernameField, gbc);  
  
 // Password field  
 JLabel lblPassword = new JLabel("Password:");  
 lblPassword.setForeground(Color.*WHITE*);  
 lblPassword.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 panel.add(lblPassword, gbc);  
  
 JPasswordField regPasswordField = new JPasswordField();  
 regPasswordField.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 regPasswordField.setPreferredSize(new Dimension(300, 40)); // Increased size  
 gbc.gridx = 1;  
 gbc.gridy = 1;  
 panel.add(regPasswordField, gbc);  
  
 // Initial Balance field  
 JLabel lblBalance = new JLabel("Initial Balance:");  
 lblBalance.setForeground(Color.*WHITE*);  
 lblBalance.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 panel.add(lblBalance, gbc);  
  
 JTextField balanceField = new JTextField();  
 balanceField.setFont(new Font("Arial", Font.*PLAIN*, 16));  
 balanceField.setPreferredSize(new Dimension(300, 40)); // Increased size  
 gbc.gridx = 1;  
 gbc.gridy = 2;  
 panel.add(balanceField, gbc);  
  
 // Register Button  
 JButton btnRegister = new JButton("Register");  
 btnRegister.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnRegister.setBackground(new Color(0, 102, 204));  
 btnRegister.setForeground(Color.*WHITE*);  
 btnRegister.setFocusPainted(false);  
 btnRegister.setPreferredSize(new Dimension(200, 50)); // Increased button size  
 gbc.gridx = 0;  
 gbc.gridy = 3;  
 gbc.gridwidth = 2;  
 panel.add(btnRegister, gbc);  
  
 // ActionListener for Register button  
 btnRegister.addActionListener(e -> {  
 String username = regUsernameField.getText();  
 String password = new String(regPasswordField.getPassword());  
 double balance = Double.*parseDouble*(balanceField.getText());  
  
 if (dbHelper.registerUser(username, password, balance)) {  
 JOptionPane.*showMessageDialog*(registerFrame, "User registered successfully!");  
 registerFrame.dispose();  
 } else {  
 JOptionPane.*showMessageDialog*(registerFrame, "Registration failed. Try again.");  
 }  
 });  
  
 registerFrame.setVisible(true);  
 }  
  
 private void showATMMenu(final String username) {  
 frame.getContentPane().removeAll();  
  
 // ATM Panel  
 JPanel atmPanel = new JPanel();  
 atmPanel.setBackground(new Color(50, 50, 50));  
 frame.getContentPane().add(atmPanel, BorderLayout.*CENTER*);  
 atmPanel.setLayout(new GridBagLayout());  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(15, 15, 15, 15);  
  
 // Balance label and display  
 JLabel lblBalance = new JLabel("Balance: ");  
 lblBalance.setForeground(Color.*WHITE*);  
 lblBalance.setFont(new Font("Arial", Font.*PLAIN*, 18));  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 atmPanel.add(lblBalance, gbc);  
  
 JLabel balanceLabel = new JLabel(String.*valueOf*(dbHelper.getBalance(username)));  
 balanceLabel.setForeground(Color.*WHITE*);  
 balanceLabel.setFont(new Font("Arial", Font.*PLAIN*, 18));  
 gbc.gridx = 1;  
 gbc.gridy = 0;  
 atmPanel.add(balanceLabel, gbc);  
  
 // Deposit Button  
 JButton btnDeposit = new JButton("Deposit");  
 btnDeposit.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnDeposit.setBackground(new Color(0, 153, 51));  
 btnDeposit.setForeground(Color.*WHITE*);  
 btnDeposit.setFocusPainted(false);  
 btnDeposit.setPreferredSize(new Dimension(150, 40));  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 gbc.gridwidth = 2;  
 atmPanel.add(btnDeposit, gbc);  
  
 // Withdraw Button  
 JButton btnWithdraw = new JButton("Withdraw");  
 btnWithdraw.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnWithdraw.setBackground(new Color(204, 51, 51));  
 btnWithdraw.setForeground(Color.*WHITE*);  
 btnWithdraw.setFocusPainted(false);  
 btnWithdraw.setPreferredSize(new Dimension(150, 40));  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 gbc.gridwidth = 2;  
 atmPanel.add(btnWithdraw, gbc);  
  
 // Logout Button  
 JButton btnLogout = new JButton("Logout");  
 btnLogout.setFont(new Font("Arial", Font.*BOLD*, 16));  
 btnLogout.setBackground(new Color(102, 102, 102));  
 btnLogout.setForeground(Color.*WHITE*);  
 btnLogout.setFocusPainted(false);  
 btnLogout.setPreferredSize(new Dimension(150, 40));  
 gbc.gridx = 0;  
 gbc.gridy = 3;  
 gbc.gridwidth = 2;  
 atmPanel.add(btnLogout, gbc);  
  
 // Action for Deposit Button  
 btnDeposit.addActionListener(e -> {  
 String amountStr = JOptionPane.*showInputDialog*(frame, "Enter deposit amount:");  
 double amount = Double.*parseDouble*(amountStr);  
 dbHelper.deposit(username, amount);  
 balanceLabel.setText(String.*valueOf*(dbHelper.getBalance(username)));  
 });  
  
 // Action for Withdraw Button  
 btnWithdraw.addActionListener(e -> {  
 String amountStr = JOptionPane.*showInputDialog*(frame, "Enter withdrawal amount:");  
 double amount = Double.*parseDouble*(amountStr);  
 dbHelper.withdraw(username, amount);  
 balanceLabel.setText(String.*valueOf*(dbHelper.getBalance(username)));  
 });  
  
 // Action for Logout Button  
 btnLogout.addActionListener(e -> {  
 showLoginPanel();  
 });  
  
 frame.revalidate();  
 frame.repaint();  
 }  
}

// JDBC MYSQL Connection   
class DBHelper {  
  
 private static final String *URL* = "jdbc:mysql://localhost:3306/ATMSystem";  
 private static final String *USER* = "root";  
 private static final String *PASSWORD* = "Sagar@9075";  
  
 private Connection conn;  
  
 public DBHelper() {  
 try {  
 conn = DriverManager.*getConnection*(*URL*, *USER*, *PASSWORD*);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public boolean authenticate(String username, String password) {  
 String query = "SELECT \* FROM Users WHERE username = ? AND password = ?";  
 try (PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, username);  
 stmt.setString(2, password);  
 ResultSet rs = stmt.executeQuery();  
 return rs.next();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 public boolean registerUser(String username, String password, double initialBalance) {  
 String query = "INSERT INTO Users (username, password, balance) VALUES (?, ?, ?)";  
 try (PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, username);  
 stmt.setString(2, password);  
 stmt.setDouble(3, initialBalance);  
 int rowsAffected = stmt.executeUpdate();  
 return rowsAffected > 0;  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return false;  
 }  
  
 public double getBalance(String username) {  
 String query = "SELECT balance FROM Users WHERE username = ?";  
 try (PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, username);  
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 return rs.getDouble("balance");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return 0.0;  
 }  
  
 public void updateBalance(String username, double newBalance) {  
 String query = "UPDATE Users SET balance = ? WHERE username = ?";  
 try (PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setDouble(1, newBalance);  
 stmt.setString(2, username);  
 stmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void deposit(String username, double amount) {  
 double currentBalance = getBalance(username);  
 double newBalance = currentBalance + amount;  
 updateBalance(username, newBalance);  
 }  
  
 public void withdraw(String username, double amount) {  
 double currentBalance = getBalance(username);  
 if (currentBalance >= amount) {  
 double newBalance = currentBalance - amount;  
 updateBalance(username, newBalance);  
 } else {  
 System.*out*.println("Insufficient funds.");  
 }  
 }  
}

**7.SQL CODE**

CREATE DATABASE ATMSystem;

USE ATMSystem;

CREATE TABLE Users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

balance DOUBLE NOT NULL DEFAULT 0

);

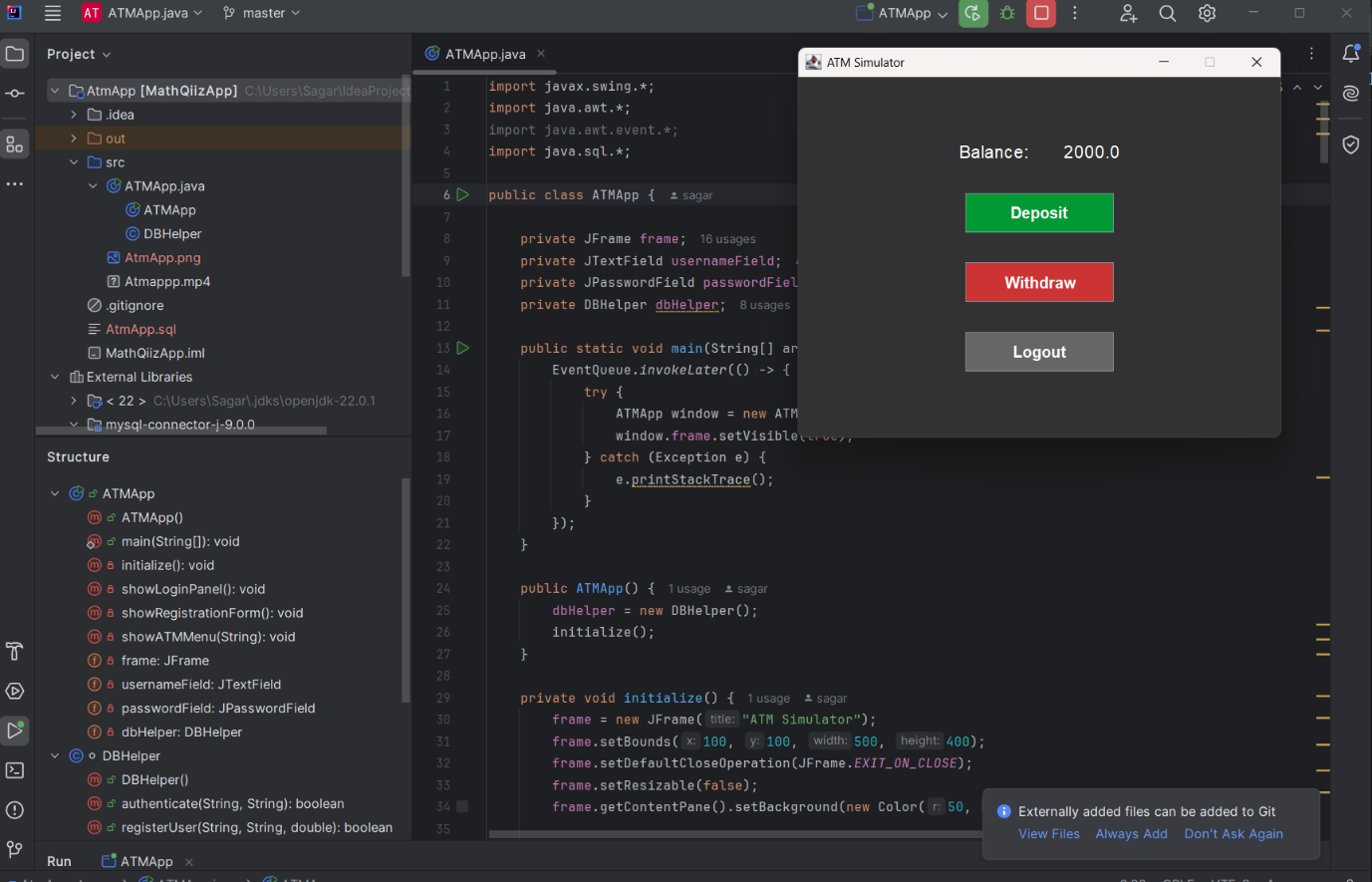
select\*from Users;

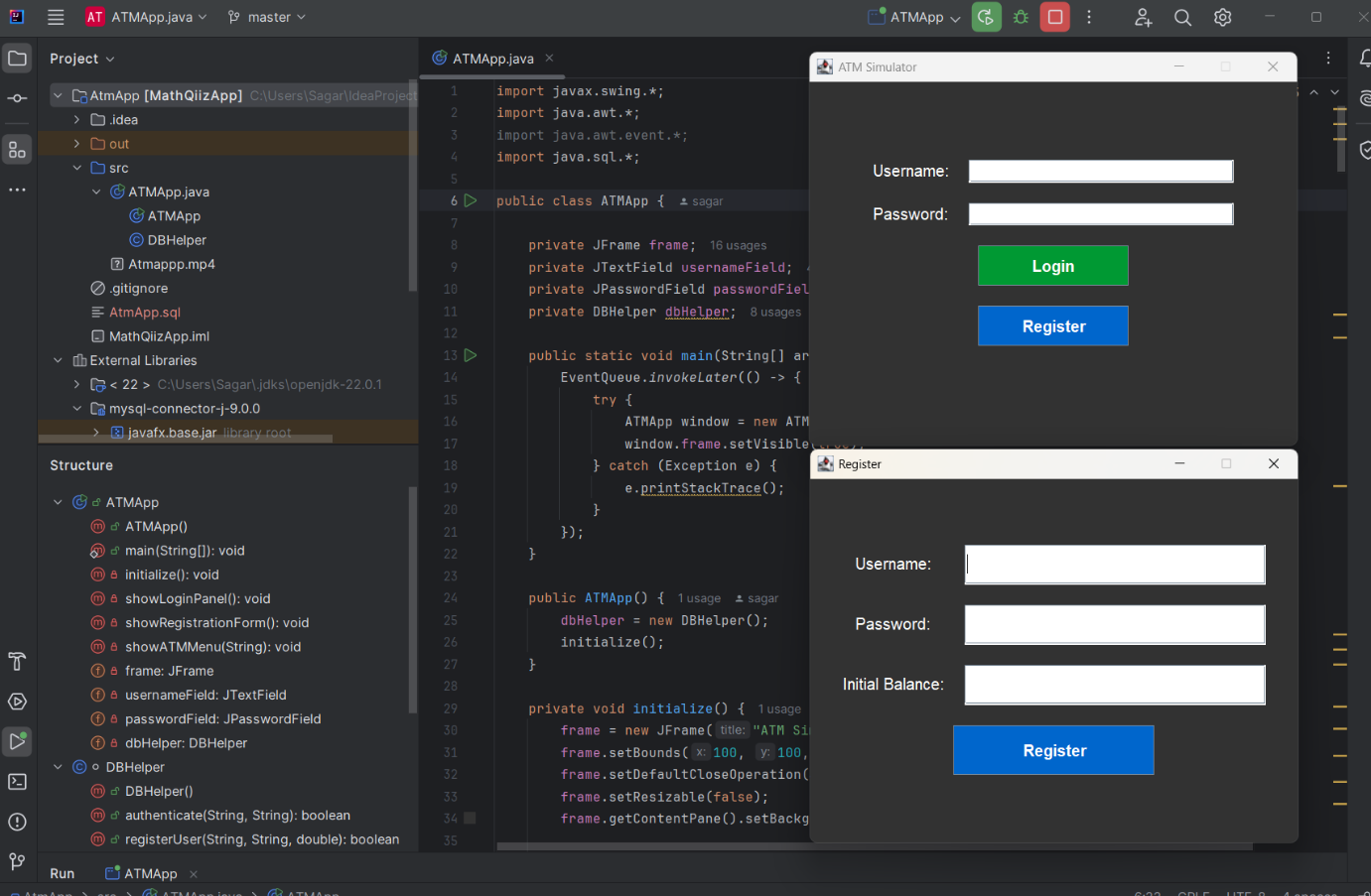
**8. Testing**

* **Unit Testing:** Tested individual functions like user authentication and balance updates.
* **Integration Testing:** Verified interactions between GUI and database.
* **User Testing:** Conducted usability testing to refine the UI and transaction processes.

**9. Results**

The ATM Simulator successfully replicates basic ATM functionalities, ensuring secure transactions and an intuitive user experience.





**10. Conclusion**

This project demonstrates the use of Java Swing and JDBC for developing a GUI-based ATM simulator. Future enhancements could include fund transfers, mini-statements, and biometric authentication.