ATM SIMULATION SYSTEM

A PROJECT REPORT

Submitted by

SRIVATHSAN S (2303811724321108)

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **" ATM SIMULATION SYSTEM"** is the bonafide work of **SRIVATHSAN S (2303811724321108)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

[illegible]

Signature

Dr. T. AVUDAIAPPAN M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy -621 112.

S. Geetha

Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24

S. Geetha

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**ATM SIMULATION SYSTEM** ” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING.**

S. Srivathsan

Signature

SRIVATHSAN S

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **"K. Ramakrishnan College of Technology (Autonomous)",** for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.,** for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D**., Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E**., Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conductive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

This project presents a simulation of an Automated Teller Machine (ATM) system developed using Java. The program replicates fundamental banking operations to provide users with a practical, interactive experience. Key features include user authentication, balance inquiries, cash withdrawals, deposits, and transaction histories, designed to mimic the functionality of a real-world ATM.

The system is built with a modular and object-oriented approach, ensuring scalability and maintainability. It utilizes Java's core libraries for functionality and incorporates error handling mechanisms to provide a robust user experience. The program also emphasizes security by using PIN-based authentication, minimizing unauthorized access.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

An ATM (Automated Teller Machine) Simulation System is a software application designed to emulate the behavior and functionality of a real-world ATM. It is often used for educational purposes, software testing, or as part of banking application development.

Scenarios to Simulate:

- Successful and failed login attempts.
- Handling of overdrafts.
- Daily withdrawal limits.
- Connectivity issues (e.g., with the database).

1.2 OBJECTIVE

To design and implement an ATM simulation system that provides a user-friendly interface for performing banking transactions, including balance inquiries, cash withdrawals, deposits, and account management, while ensuring security, accuracy, and efficiency in a simulated environment.

CHAPTER 2
PROJECT METHODOLOGY

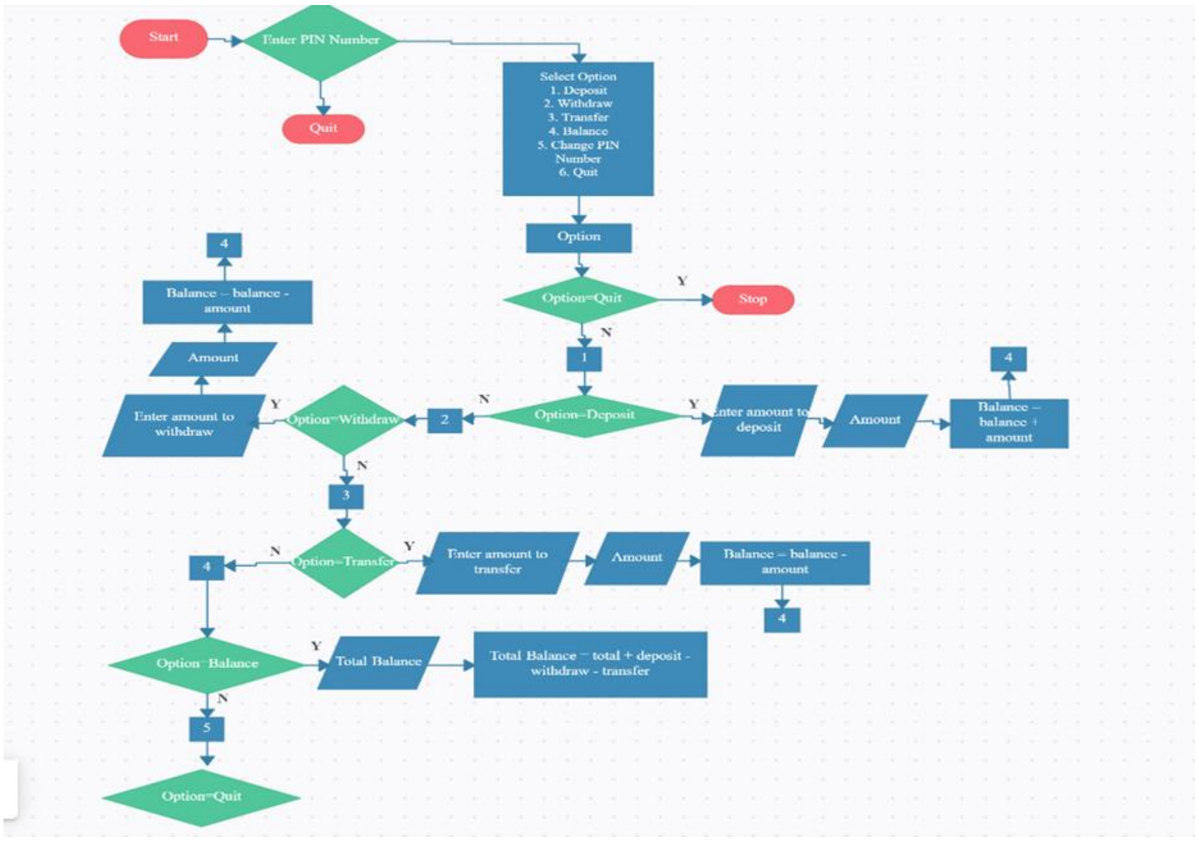
2.1Proposed Work

- The ATM Simulation program is designed to replicate the basic functionalities of an Automated Teller Machine (ATM). This program provides users with an interactive interface to perform essential banking operations such as checking account balance, withdrawing money, depositing funds, and exiting the session.
- **The primary goals of this project are to:**

1. **Enhance Understanding of Object-Oriented Programming (OOPs):** The program demonstrates the use of Java classes, objects, and methods to model real-world entities and processes.
2. **Implement Error Handling:** The program incorporates input validation and exception handling to ensure a smooth user experience.
3. **Improve Logical Thinking:** Logical conditions and control flow structures are used to simulate decision-making in an ATM.

This simulation is an excellent way to practice programming concepts while modeling a real-world application. It can be expanded to include advanced features like multiple accounts, database integration, or a graphical user interface (GUI) for a more immersive experience.

2.2 Block Diagram



CHAPTER 3
JAVA PROGRAMMING CONCEPTS

3.1 OBJECT-ORIENTED PROGRAMMING (OOP)

- Classes and Objects: Represent entities like the ATM, user account, and bank.
- Encapsulation: Use private fields with public getters and setters to protect sensitive data (e.g., account balance, PIN).

3.2 INPUT/OUTPUT OPERATIONS

Console input/Output: Use scanner for reading user input and system.out.println for displaying messages.

File Handling: Store user detail or transaction logs persistently in files using FILE and FileWriter.

3.3 EXCEPTION HANDLING

Try-catch Blocks: Handle errors like invalid inputs, insufficient funds, or file not found.

Custom Exceptions: Define exception such as InsufficientFundsException for specific errors.

3.4 CONTROL FLOW

Loops: Repeatedly display menus or retry on invalid input

Conditional Statements: Handle choices and actions like deposit, withdrawl, balance check, or PIN validation if-else or Switch

3.5 MULTI-THREADING

Simulate concurrent ATM access using threads if advanced simulation is required.

CHAPTER 4
MODULE DESCRIPTION

4.1 Module 1 User Interface Module

Handles interactions between the user and the system.

4.2 Module 2 Authentication Module

1)Ensures secure access to user accounts.

2)User login with card number and PIN verification.

4.3 Module 3 Account Management Module

1)Manages user accounts and related operations.

2)View account balance.

3)Update user account details.

4.4 Module 4 Transaction Processing Module

1)Handle all financial transaction in the system

2) Withdrawal with overdraft protection.

3) Deposit processing and balance updates.

4)Transfer funds between accounts.

4.5 Module 5 Error Handlin Module

Manages errors and exceptions that occur during system operation includes Error Logging ,User Notifications ,Recovery

CHAPTER 5
CONCLUSION

The ATM simulation program in Java has successfully implemented the basic features of an ATM system, such as user authentication, account management, and transaction processing. The system works efficiently for the use cases it was designed for and provides an excellent foundation for future enhancements. While the program is simple and text-based, it effectively demonstrates how the key components of an ATM work together. Future improvements could focus on making the system more secure, user-friendly, and scalable to accommodate a larger number of users and transactions.

REFERENCES:

Books on Java Programming:

- **"Head First Java" by Kathy Sierra and Bert Bates** – A great book for beginners to understand core Java concepts and object-oriented programming.
- **"Effective Java" by Joshua Bloch** – A reference for more advanced concepts, including best practices in Java programming.

Online Tutorials:

- **GeeksforGeeks**: A popular programming tutorial website with examples and explanations for many Java concepts, such as handling user input, arrays, and collections: GeeksforGeeks Java Tutorials
- **W3Schools**: A beginner-friendly site for learning basic Java syntax, concepts, and examples: W3Schools Java Tutorial

Online Courses:

- **Coursera** and **Udemy** offer many Java courses that cover everything from beginner to advanced topics, including OOP and Java application development.

APPENDIX-A SOURCE CODE

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.util.HashMap;

import java.util.Map;


public class ATMSystem extends JFrame {

    private static final Map<String, String> users = new HashMap<>(); // Store
account numbers and PINs

    private String currentAccountNumber;

    private int balance = 100000;


    private JTextField accountNumberField;

    private JPasswordField pinField;

    private JTextArea outputArea;


    public ATMSystem() {

        // Predefined users (You can add more users here)

        users.put("123456", "1234");

        users.put("987654", "4321");
        // Set up the GUI components
        setTitle("ATM System");
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Set background color
        getContentPane().setBackground(new Color(240, 240, 240));

        // Account Login Panel
        JPanel loginPanel = new JPanel();
        loginPanel.setLayout(new GridLayout(3, 2));
        loginPanel.setBackground(new Color(220, 220, 255)); // Light blue
background

        loginPanel.add(new JLabel("Account Number:"));
        accountNumberField = new JTextField();
        loginPanel.add(accountNumberField);

        loginPanel.add(new JLabel("PIN:"));
        pinField = new JPasswordField();
        loginPanel.add(pinField);
```

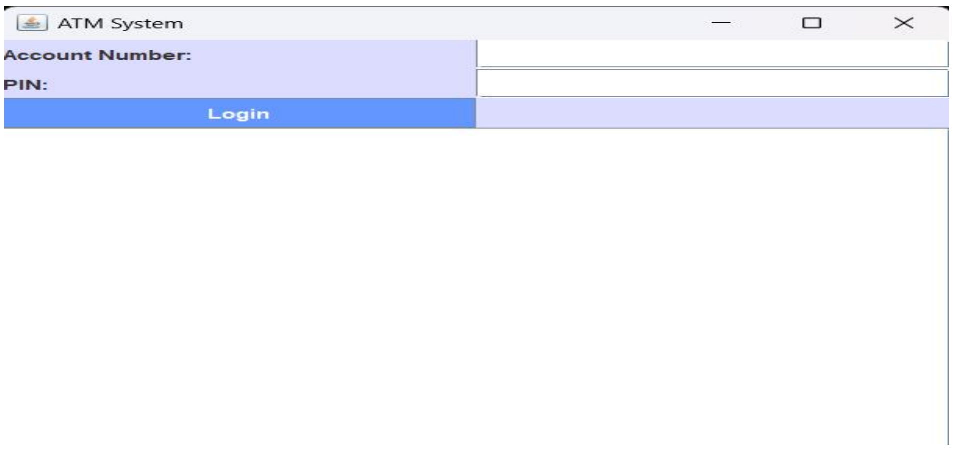
```
JButton loginButton = new JButton("Login");
loginButton.setBackground(new Color(100, 150, 255)); // Light blue button
loginButton.setForeground(Color.WHITE);
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        authenticate();
    }
});
loginPanel.add(loginButton);

// Output Area for results
outputArea = new JTextArea();
outputArea.setEditable(false);
outputArea.setFont(new Font("Arial", Font.PLAIN, 18)); // Increased font
size
outputArea.setBackground(Color.WHITE);
outputArea.setForeground(Color.BLACK);
JScrollPane scrollPane = new JScrollPane(outputArea);

// Add components to the frame
add(loginPanel, BorderLayout.NORTH);
```

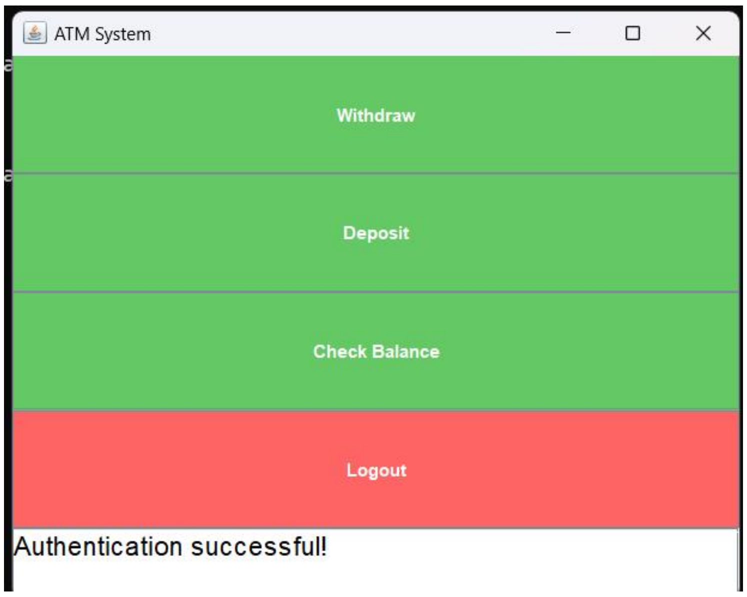
```
        add(scrollPane, BorderLayout.CENTER);


        setVisible(true);

    }


    // Method to authenticate the user

    private void authenticate() {

        String enteredAccountNumber = accountNumberField.getText();

        String enteredPin = new String(pinField.getPassword());


        if (users.containsKey(enteredAccountNumber) &&
users.get(enteredAccountNumber).equals(enteredPin)) {

            currentAccountNumber = enteredAccountNumber;

            outputArea.setText("Authentication successful!\n");

            displayMainMenu();

        } else {

            outputArea.setText("Authentication failed! Invalid account number or
PIN.\n");

        }

    }


    // Method to display the main menu after successful authentication

    private void displayMainMenu() {
        // Remove previous components (login screen)

        getContentPane().removeAll();

        revalidate();

        repaint();


        // Main menu panel

        JPanel menuPanel = new JPanel();

        menuPanel.setLayout(new GridLayout(4, 1));

        menuPanel.setBackground(new Color(220, 255, 220)); // Light green
background


        JButton withdrawButton = new JButton("Withdraw");

        withdrawButton.setBackground(new Color(100, 200, 100)); // Light green
button

        withdrawButton.setForeground(Color.WHITE);

        withdrawButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                withdraw();

            }

        });


        JButton depositButton = new JButton("Deposit");
        depositButton.setBackground(new Color(100, 200, 100)); // Light green
button

        depositButton.setForeground(Color.WHITE);

        depositButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                deposit();

            }

        });


        JButton checkBalanceButton = new JButton("Check Balance");

        checkBalanceButton.setBackground(new Color(100, 200, 100)); // Light
green button

        checkBalanceButton.setForeground(Color.WHITE);

        checkBalanceButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                checkBalance();

            }

        });


        JButton logoutButton = new JButton("Logout");

        logoutButton.setBackground(new Color(255, 100, 100)); // Red button for
logout

        logoutButton.setForeground(Color.WHITE);
        logoutButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                logout();
            }
        });

        menuPanel.add(withdrawButton);
        menuPanel.add(depositButton);
        menuPanel.add(checkBalanceButton);
        menuPanel.add(logoutButton);

        add(menuPanel, BorderLayout.CENTER);

        // Output area at the bottom
        JScrollPane scrollPane = new JScrollPane(outputArea);
        add(scrollPane, BorderLayout.SOUTH);

        revalidate();
        repaint();
    }

    private void withdraw() {
        String amountStr = JOptionPane.showInputDialog("Enter amount to
withdraw:");
        if (amountStr != null) {
            try {
                int amount = Integer.parseInt(amountStr);
                if (amount > 0 && amount <= balance) {
                    balance -= amount;
                    outputArea.setText("Withdrawal successful! New balance: "
+ balance);
                } else {
                    outputArea.setText("Insufficient balance or invalid
amount.");
                }
            } catch (NumberFormatException e) {
                outputArea.setText("Invalid amount. Please enter a valid
number.");
            }
        }
    }

    private void deposit() {
        String amountStr = JOptionPane.showInputDialog("Enter amount to
deposit:");
        if (amountStr != null) {
            try {
                int amount = Integer.parseInt(amountStr);
                if (amount > 0) {
                    balance += amount;
                    outputArea.setText("Deposit successful! New balance: " +
balance);
                } else {
                    outputArea.setText("Invalid amount. Please enter a positive
number.");
                }
            } catch (NumberFormatException e) {
                outputArea.setText("Invalid amount. Please enter a valid
number.");
            }
        }
    }

    private void checkBalance() {
        outputArea.setText("Current balance: " + balance);
    }

    private void logout() {
        currentAccountNumber = null;
        outputArea.setText("You have been logged out.\n");

        accountNumberField.setText("");

        pinField.setText("");

        displayLoginScreen();

    }


    // Method to display login screen

    private void displayLoginScreen() {

        // Remove previous components (main menu)

        getContentPane().removeAll();

        revalidate();

        repaint();


        JPanel loginPanel = new JPanel();

        loginPanel.setLayout(new GridLayout(3, 2));

        loginPanel.setBackground(new Color(220, 220, 255)); // Light blue
background


        loginPanel.add(new JLabel("Account Number:"));

        accountNumberField = new JTextField();

        loginPanel.add(accountNumberField);
```

```
loginPanel.add(new JLabel("PIN:"));
pinField = new JPasswordField();
loginPanel.add(pinField);

JButton loginButton = new JButton("Login");
loginButton.setBackground(new Color(100, 150, 255)); // Light blue button
loginButton.setForeground(Color.WHITE);
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        authenticate();
    }
});
loginPanel.add(loginButton);

outputArea.setText("");

add(loginPanel, BorderLayout.NORTH);
add(new JScrollPane(outputArea), BorderLayout.CENTER);

revalidate();
repaint();
```

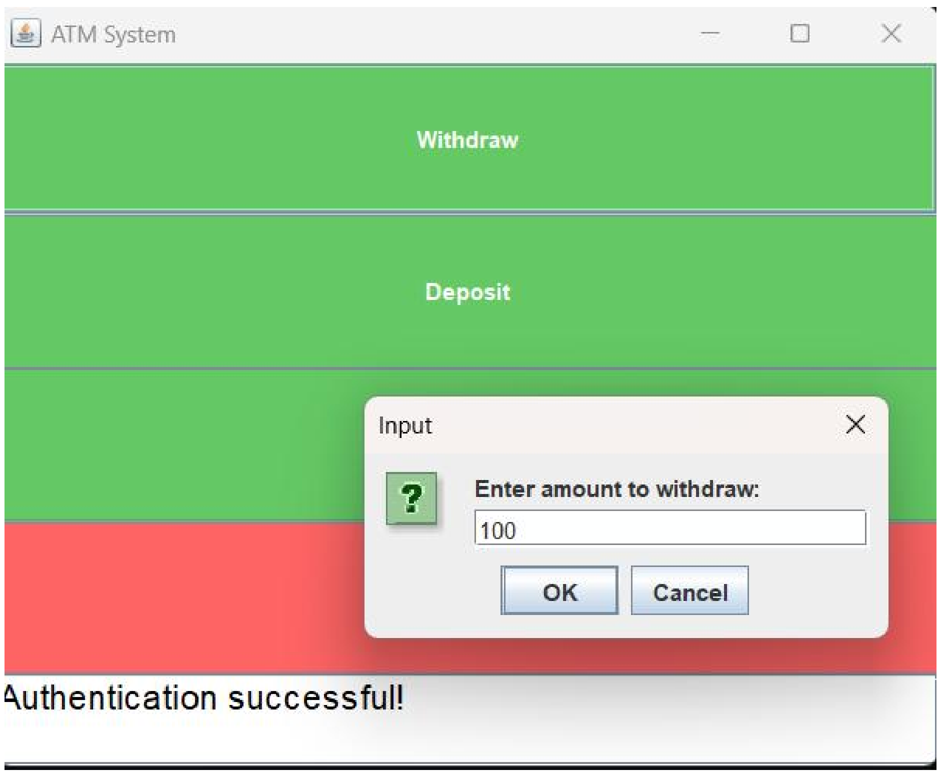
```
        }


        public static void main(String[] args) {
                SwingUtilities.invokeLater(new Runnable() {
                        @Override
                        public void run() {
                                new ATMSystem();
                        }
                });
        }
}
```

APPENDIX-B SCREEN SHOT

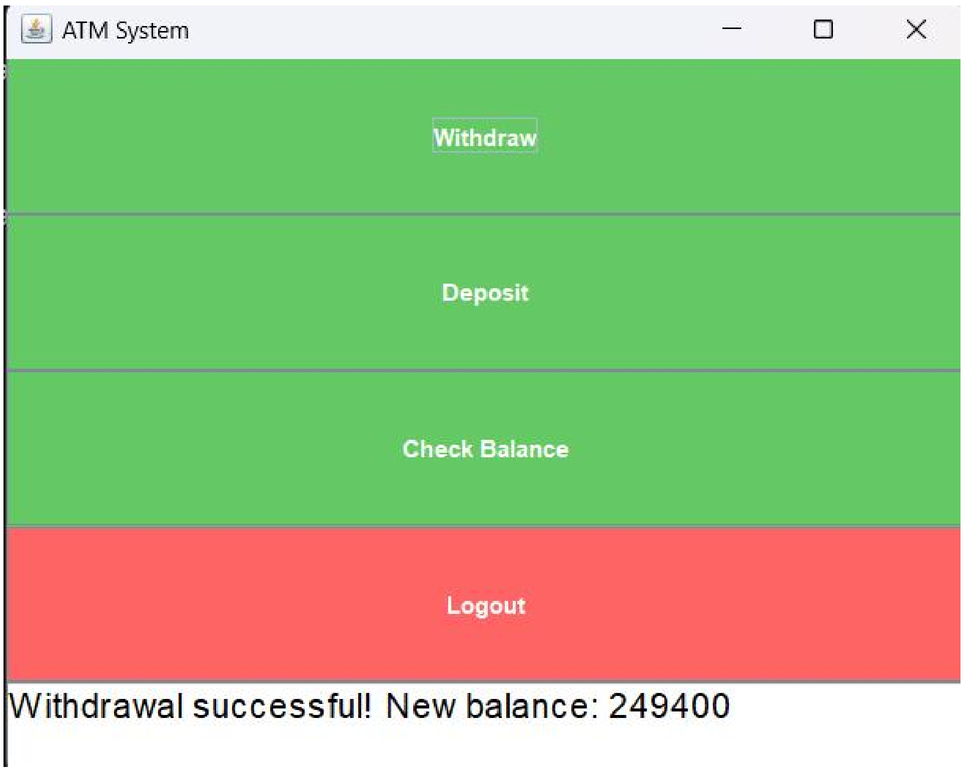




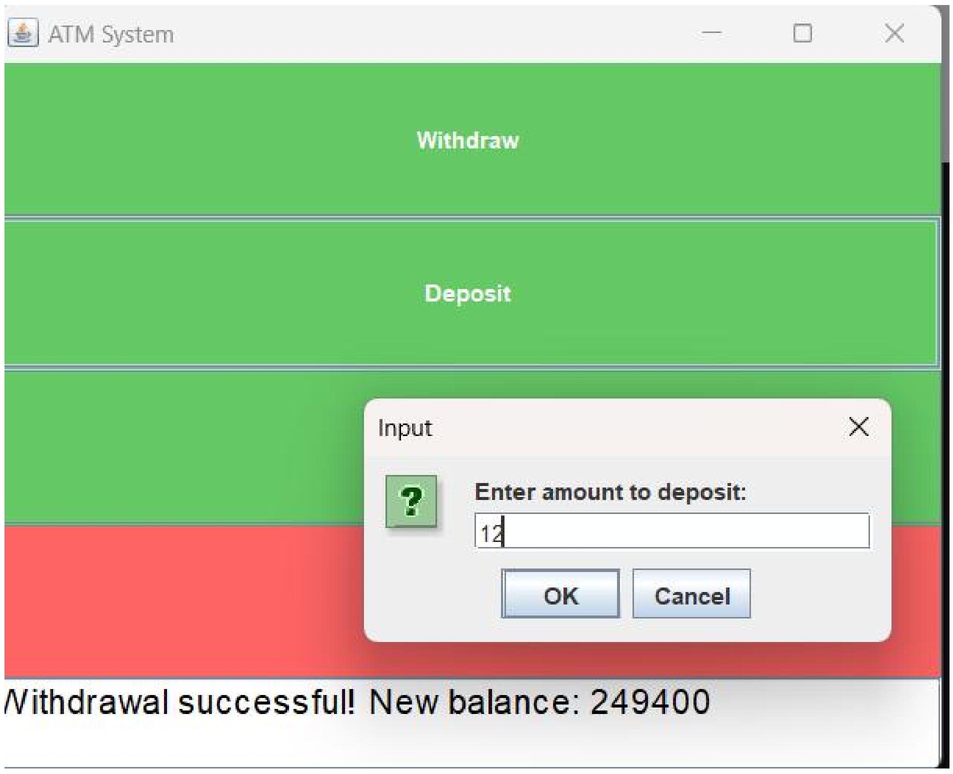
Authentication successful!



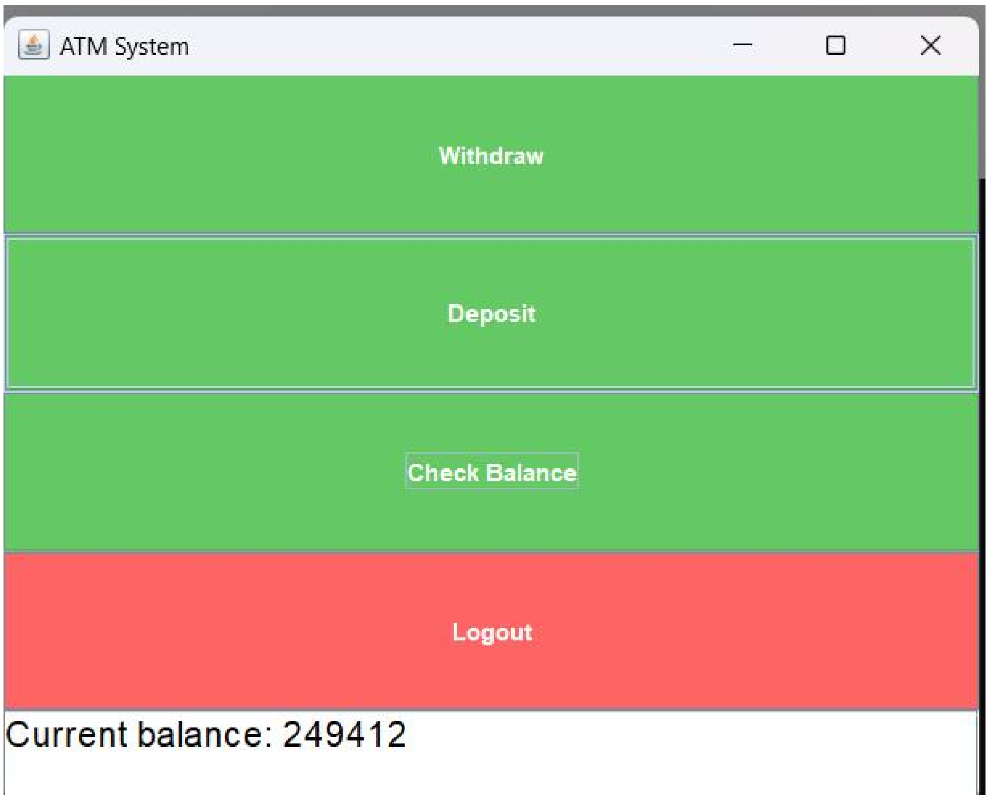
Authentication successful!



Withdrawal successful! New balance: 249400



Withdrawal successful! New balance: 249400



Current balance: 249412