VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI 590018

Project Report on

"ATM SIMULATION"

By

Nikhitha S (1BM24CS189)
Nikitha M N (1BM24CS190)
Panchami P (1BM24CS196)
Harsh Bafna (1BM24CS357)

Under the Guidance of

Prof. Monisha H M

Professor, Department of CSE
BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering
BMS College of Engineering
(Autonomous college under VTU)
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

2025-2026

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOPS with JAVA project titled "**ATM SIMULATION"** has been carried out by Nikhitha S (1BM24CS189), Nikitha M N (1BM24CS190), Panchami P (1BM24CS196), Harsh Bafna (1BM24CS357) during the academic year 2025-2026.

Signature of the guide
Prof. Monisha H M

Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Nikhitha S (1BM24CS189), Nikitha M N (1BM24CS190), Panchami P (1BM24CS196), Harsh Bafna (1BM24CS357), students of 3rd Semester, 'I' Section, B.E., Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this project work entitled "ATM SIMULATION" has been carried out by us under the guidance of Prof. Monisha H M, Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Sep-Dec 2025. We also declare that to the best of our knowledge and belief, the project reported here is not from part of any other report by any other students.

Signature of the Candidates

Nikhitha S (1BM24CS189)

Nikitha M N (1BM24CS190)

Panchami P (1BM24CS196)

Harsh Bafna (1BM24CS357)

PROBLEM STATEMENT:

Traditional ATM systems are widely used for performing basic banking operations such as balance inquiry, cash withdrawal, and deposit. However, understanding the internal working of such systems requires practical implementation using programming concepts. Many students learn Java concepts theoretically but lack exposure to real-time application development.

The problem is to design and develop a Java-based ATM Simulation System that mimics the working of a real ATM machine. The system should provide a secure login mechanism using a PIN and allow the user to perform basic banking operations through a graphical interface. The application must be user-friendly, reliable, and should handle invalid inputs gracefully without crashing.

INTRODUCTION:

An Automated Teller Machine (ATM) is an electronic banking outlet that enables customers to complete basic transactions without the assistance of a bank representative. ATMs have become an essential part of modern banking due to their convenience and availability.

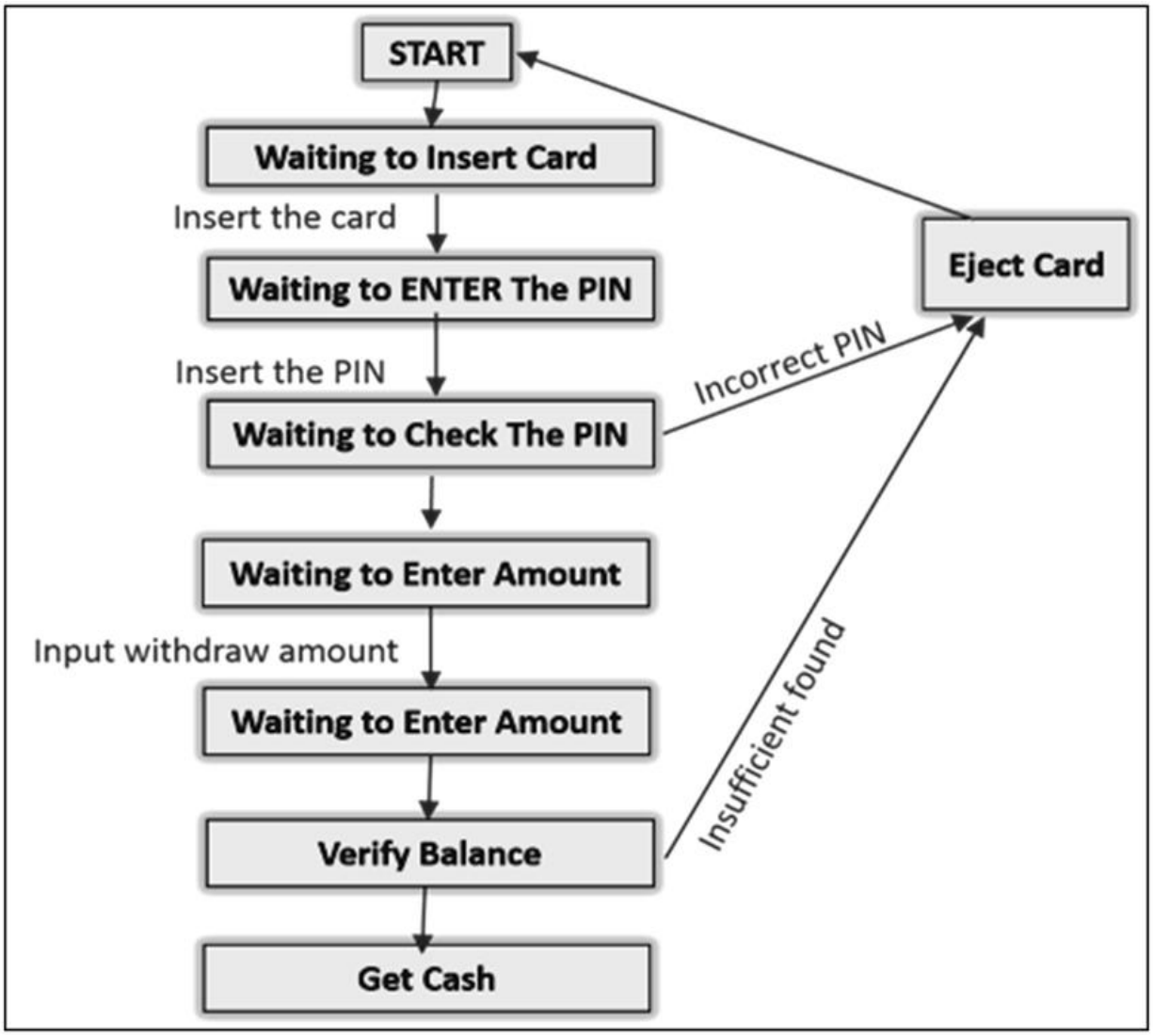
This project titled **"ATM SIMULATION"** is a graphical user interface (GUI) based application developed using Java Swing. It gives you real time visualization of fundamental operations of an ATM such as PIN authentication, balance inquiry, cash deposit, and withdrawal. The project emphasizes the practical implementation of **Object-Oriented Programming (OOP) concepts**, event-driven programming, and exception handling in Java.

The main aim of this project is to provide hands-on experience in developing a real-world application using Java and to enhance understanding of GUI components and user interaction.

OVERVIEW OF THE PROJECT:

The ATM Simulation Project consists of multiple functional modules that work together to provide seamless interaction with the user.

Here is the Block Diagram of the modules:



Explanation

- The user first interacts with the **Login Module** by entering a PIN.
- After successful authentication, the user is redirected to the **ATM Menu**.
- The menu allows the user to select operations like checking balance, depositing money, or withdrawing cash.
- The system updates the balance dynamically and displays appropriate messages.
- The user can exit the system safely at any time.

TOOLS USED:

The following tools and technologies were used to develop this project:

- **Java (JDK)** – Core programming language used for application logic
- **Java Swing** – Used for creating graphical user interface components
- **AWT (Abstract Window Toolkit)** – Used for layouts, events, and graphics
- **VS CODE Editor**– Used for writing, compiling, and executing code
- **Windows Operating System** – Platform used for development and testing

OOPS CONCEPTS, TECHNIQUES AND MECHANISMS USED FROM JAVA:

This project extensively uses Object Oriented Programming concepts along with other Java mechanisms, which are explained below:

Encapsulation:

Encapsulation is achieved by declaring variables such as balance and PIN as private members of the class. This prevents direct access to sensitive data and ensures data security.

Inheritance:

The main class ATMSimulation extends the JFrame class, inheriting properties of a window-based application.

Abstraction:

The internal implementation details of ATM operations are hidden from the user. The user interacts only through buttons and dialogs.

Polymorphism:

Method overriding is used in handling events such as actionPerformed() where different actions are performed based on the source of the event.

Interfaces:

The ActionListener interface is implemented to handle button click events.

Exception Handling:

Try-catch blocks are used to handle invalid inputs such as non-numeric PINs or invalid transaction amounts, preventing application crashes.

Java Swing

Java Swing is a GUI toolkit used to create window-based applications. In this project, Swing components such as JFrame, JButton, JLabel, JPasswordField, JPanel, and JOptionPane are used to design the ATM interface. Swing allows the application to provide a user-friendly and interactive experience similar to a real ATM machine.

Abstract Window Toolkit (AWT)

AWT is a Java library that provides basic graphical components, layouts, and event handling mechanisms. In this project, AWT is used for layout management (BorderLayout, GridBagLayout), event handling (ActionListener), colors, fonts, and graphics rendering. AWT works along with Swing to manage user interactions and visual appearance.

Java Event Handling

Event handling is a mechanism that controls user interactions such as button clicks. The ActionListener interface is implemented in this project to handle actions performed by ATM buttons like login, deposit, withdrawal, and exit. Each user action triggers an event that is processed by the application.

Java Graphics (Graphics & Graphics2D)

The Graphics and Graphics2D classes are used to create custom visual effects. In this project, they are used to design a gradient background for the ATM interface, enhancing the visual appearance of the application.

Java Layout Managers

Layout managers are used to control the arrangement of components in a window. The project uses BorderLayout and GridBagLayout to place buttons, labels, and input fields in a structured and responsive manner.

Java Swing Dialogs (JOptionPane)

JOptionPane is used to display dialog boxes for user interaction. It is used in the project to show messages such as balance information, transaction success messages, error alerts, and input dialogs for deposit and withdrawal amounts.

IMPLEMENTATION:

The ATM Simulation is implemented as a Java Swing application using event-driven programming. The main components include:

- **Login Screen** using JPasswordField for secure PIN entry
- **Menu Screen** with buttons for ATM operations
- **Dialog Boxes** using JOptionPane to interact with the user

The application maintains the account balance internally and updates it based on deposit and withdrawal transactions. Each user action triggers an event handled by the actionPerformed() method.

CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ATMSimulation extends JFrame implements ActionListener {

   private final int PIN = 1234;
   private int balance = 10000;

   private JPasswordField pinField;
   private JButton loginBtn;

   private JButton balanceBtn, depositBtn, withdrawBtn, exitBtn;

   public ATMSimulation() {
      setTitle("ATM Simulation");
      setExtendedState(JFrame.MAXIMIZED_BOTH);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(new BorderLayout());
```

```
    showLoginScreen();
    setVisible(true);
}

// ================= LOGIN SCREEN =================
void showLoginScreen() {
    getContentPane().removeAll();
    add(new GradientPanel(createLoginCard()), BorderLayout.CENTER);
    revalidate();
    repaint();
}

// ================= MENU SCREEN =================
void showMenuScreen() {
    getContentPane().removeAll();
    add(new GradientPanel(createMenuCard()), BorderLayout.CENTER);
    revalidate();
    repaint();
}

// ================= LOGIN CARD =================
JPanel createLoginCard() {
    JPanel panel = baseCard();

    JLabel title = titleLabel("ATM MACHINE");
    JLabel pinLabel = label("Enter PIN:");

    pinField = new JPasswordField(10);
    pinField.setFont(new Font("Segoe UI", Font.PLAIN, 16));

    loginBtn = primaryButton("LOGIN");
    loginBtn.addActionListener(this);
```

```
        addToPanel(panel, title, 0, 0, 2);
        addToPanel(panel, pinLabel, 0, 1, 1);
        addToPanel(panel, pinField, 1, 1, 1);
        addToPanel(panel, loginBtn, 0, 2, 2);

        return panel;
    }
// ================= MENU CARD =================
    JPanel createMenuCard() {
        JPanel panel = baseCard();

        JLabel title = titleLabel("ATM MENU");

        balanceBtn = primaryButton("Check Balance");
        depositBtn = primaryButton("Deposit");
        withdrawBtn = primaryButton("Withdraw");
        exitBtn = dangerButton("Exit");

        balanceBtn.addActionListener(this);
        depositBtn.addActionListener(this);
        withdrawBtn.addActionListener(this);
        exitBtn.addActionListener(this);

        addToPanel(panel, title, 0, 0, 2);
        addToPanel(panel, balanceBtn, 0, 1, 2);
        addToPanel(panel, depositBtn, 0, 2, 2);
        addToPanel(panel, withdrawBtn, 0, 3, 2);
        addToPanel(panel, exitBtn, 0, 4, 2);


    return panel;
    }
```

```
// ================= BUTTON ACTIONS =================
    public void actionPerformed(ActionEvent e) {

        // LOGIN
        if (e.getSource() == loginBtn) {
            try {
                int enteredPin = Integer.parseInt(
                        new String(pinField.getPassword()));

                if (enteredPin == PIN) {
                    showMenuScreen();
                } else {
                    message("Invalid PIN ✖");
                }
            } catch (Exception ex) {
                message("Enter numeric PIN only");
            }
        }

        // CHECK BALANCE
        if (e.getSource() == balanceBtn) {
            message("💰 Current Balance: ₹" + balance);
        }

        // DEPOSIT
        if (e.getSource() == depositBtn) {
            String input = JOptionPane.showInputDialog(this,
                    "Enter amount to deposit:");
            try {
                int amt = Integer.parseInt(input);
                if (amt > 0) {
```

```
            balance += amt;
            message("₹" + amt + " deposited successfully");
          }
        } catch (Exception ex) {
          message("Invalid amount");
        }
      }

      // WITHDRAW
      if (e.getSource() == withdrawBtn) {
        String input = JOptionPane.showInputDialog(this,
            "Enter amount to withdraw:");
        try {
          int amt = Integer.parseInt(input);
          if (amt > balance) {
            message("Insufficient balance ❌");
          } else if (amt > 0) {
            balance -= amt;
            message("Please collect cash 💵\nBalance: ₹" + balance);
          }
        } catch (Exception ex) {
          message("Invalid amount");
        }
      }

      // EXIT
      if (e.getSource() == exitBtn) {
        message("Thank you for using ATM 😊");
        System.exit(0);
      }
    }
```

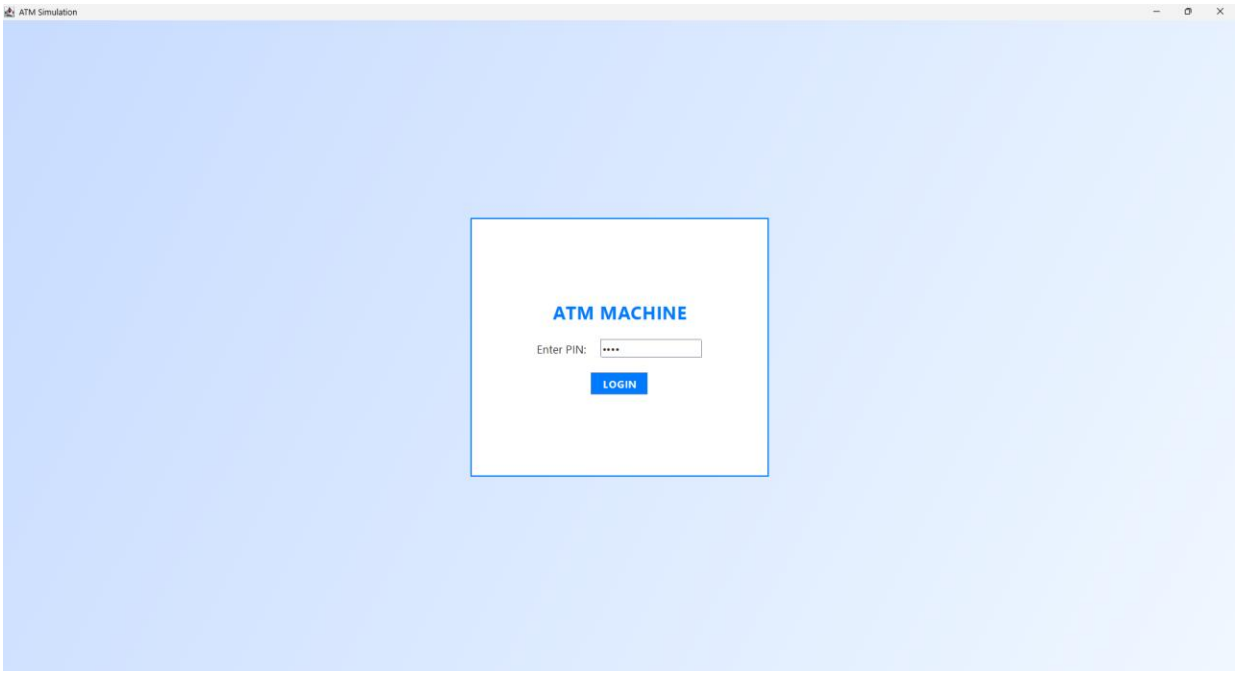
```
// -----UI HELPERS-----
JPanel baseCard() {
    JPanel panel = new JPanel(new GridBagLayout());
    panel.setPreferredSize(new Dimension(420, 360));
    panel.setBackground(Color.WHITE);
    panel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(new Color(0, 123, 255), 2),
            BorderFactory.createEmptyBorder(20, 20, 20, 20)
    ));
    return panel;
}

JLabel titleLabel(String text) {
    JLabel lbl = new JLabel(text);
    lbl.setFont(new Font("Segoe UI", Font.BOLD, 26));
    lbl.setForeground(new Color(0, 123, 255));
    return lbl;
}

JLabel label(String text) {
    JLabel lbl = new JLabel(text);
    lbl.setFont(new Font("Segoe UI", Font.PLAIN, 16));
    return lbl;
}

JButton primaryButton(String text) {
    JButton btn = new JButton(text);
    btn.setFont(new Font("Segoe UI", Font.BOLD, 15));
    btn.setBackground(new Color(0, 123, 255));
    btn.setForeground(Color.WHITE);
    btn.setFocusPainted(false);
    return btn;
```

```
    }

    JButton dangerButton(String text) {
        JButton btn = primaryButton(text);
        btn.setBackground(new Color(220, 53, 69));
        return btn;
    }

    void addToPanel(JPanel p, Component c, int x, int y, int w) {
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.gridx = x;
        gbc.gridy = y;
        gbc.gridwidth = w;
        gbc.insets = new Insets(10, 10, 10, 10);
        p.add(c, gbc);
    }

    void message(String msg) {
        JOptionPane.showMessageDialog(this, msg, "ATM",
                JOptionPane.INFORMATION_MESSAGE);
    }
// ================= GRADIENT BACKGROUND =================
    class GradientPanel extends JPanel {
        JPanel card;
        GradientPanel(JPanel card) {
            this.card = card;
            setLayout(new GridBagLayout());
            add(card);
        }

        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
```

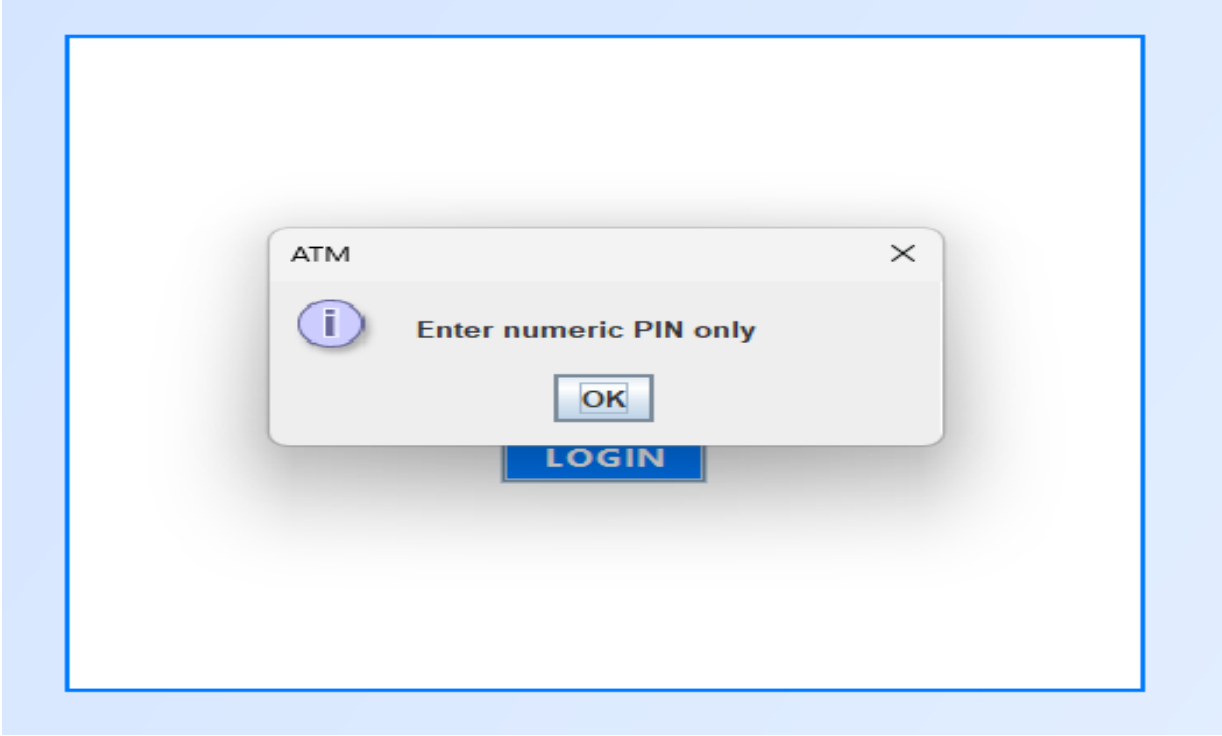
```
            Graphics2D g2 = (Graphics2D) g;
            g2.setPaint(new GradientPaint(
                  0, 0, new Color(200, 220, 255),
                  getWidth(), getHeight(), new Color(240, 248, 255)
            ));
            g2.fillRect(0, 0, getWidth(), getHeight());
        }
    }
    public static void main(String[] args) {
        new ATMSimulation();
    }
}
```

RESULTS AND OUTPUT:

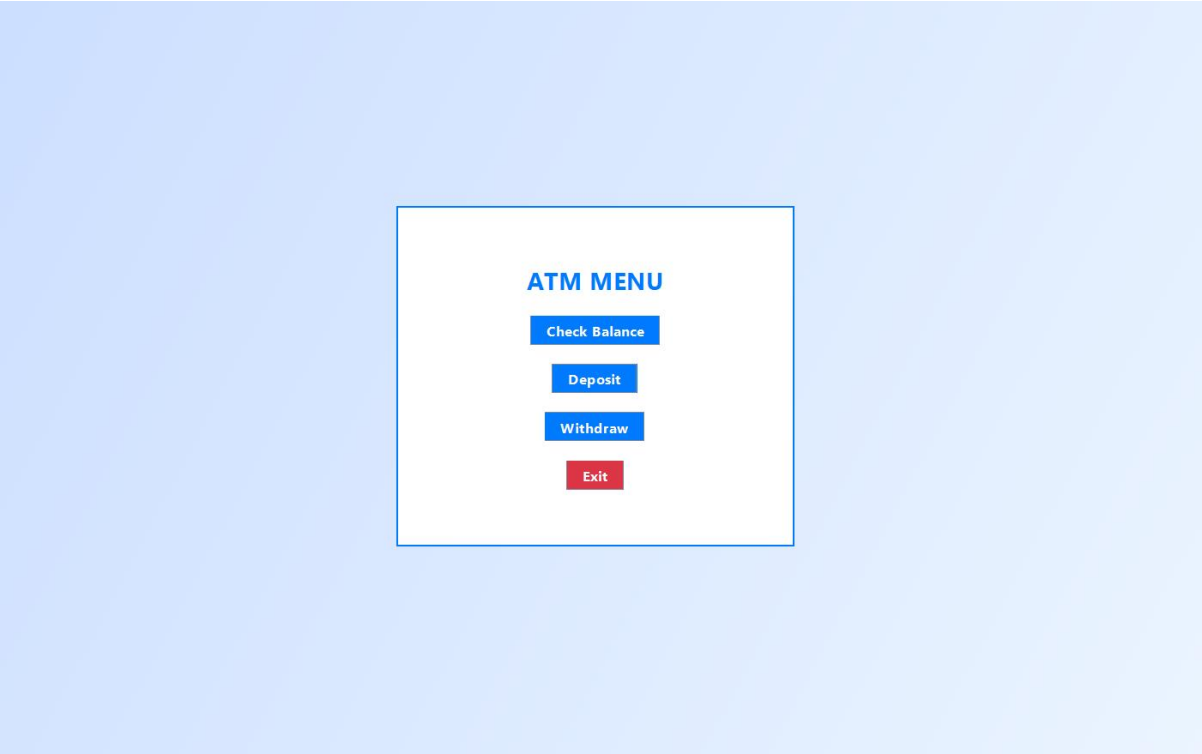
LOGIN SCREEN



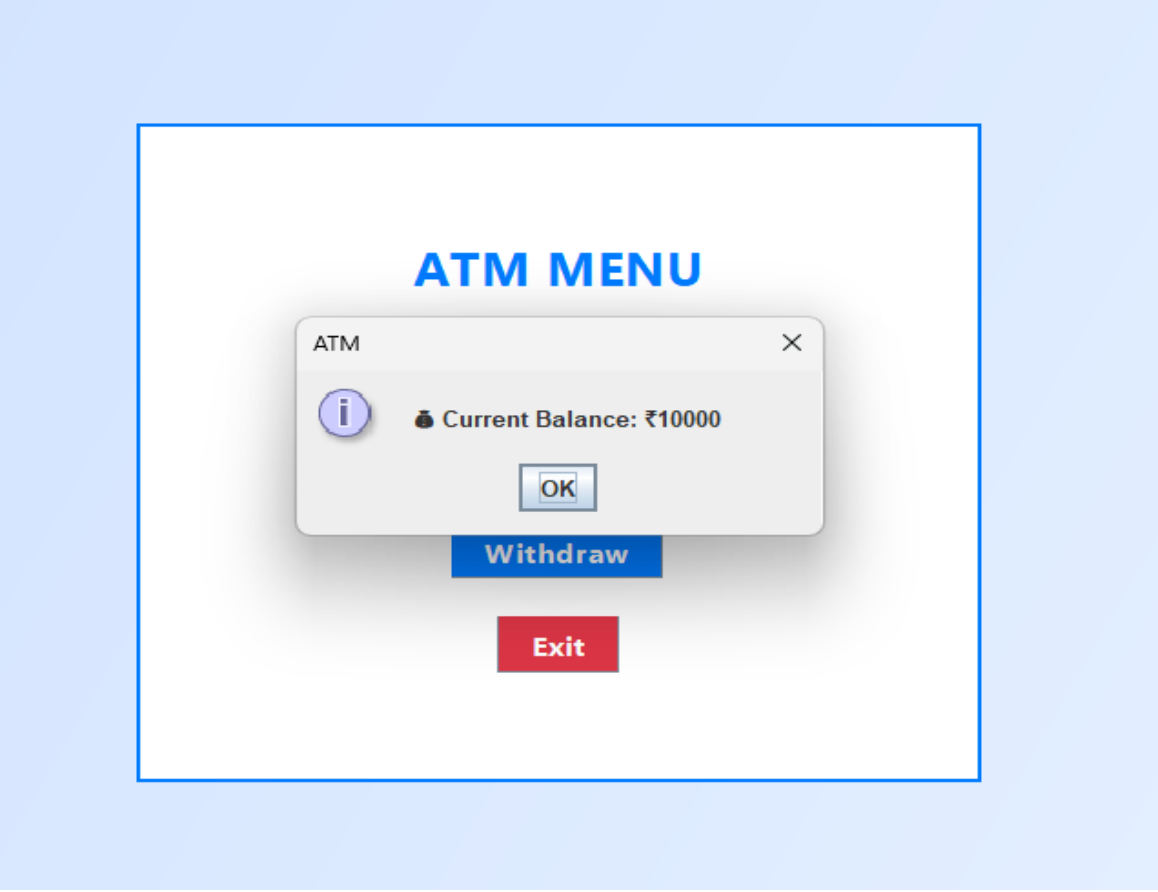
ERROR MESSAGE DISPLAYED IF NON- NUMERIC PIN ENTERED



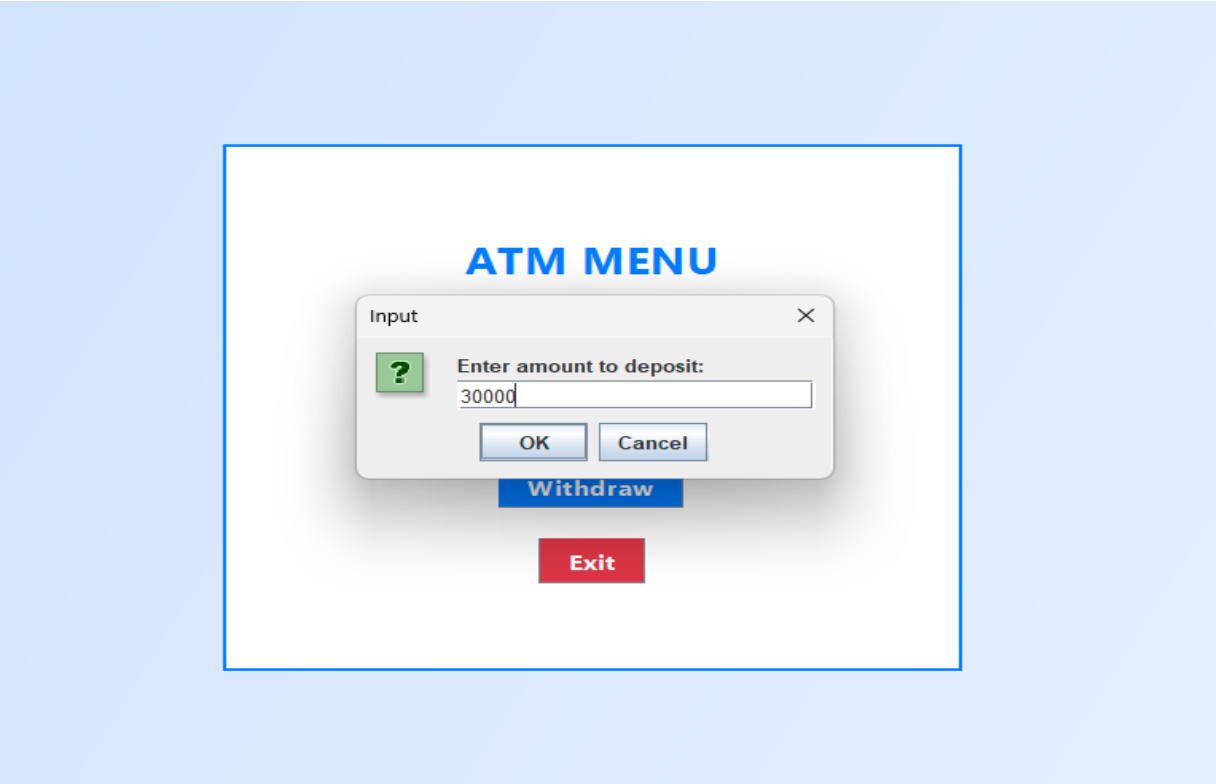
MENU SCREEN



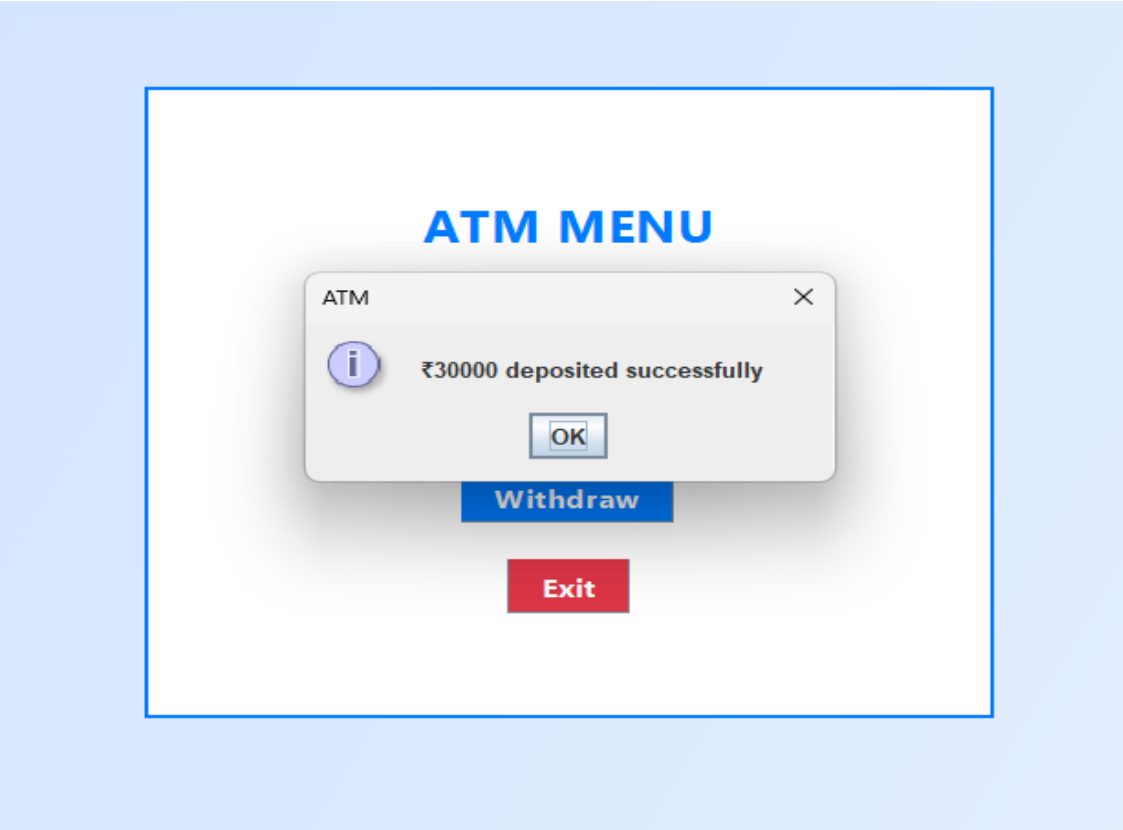
CHECK BALANCE



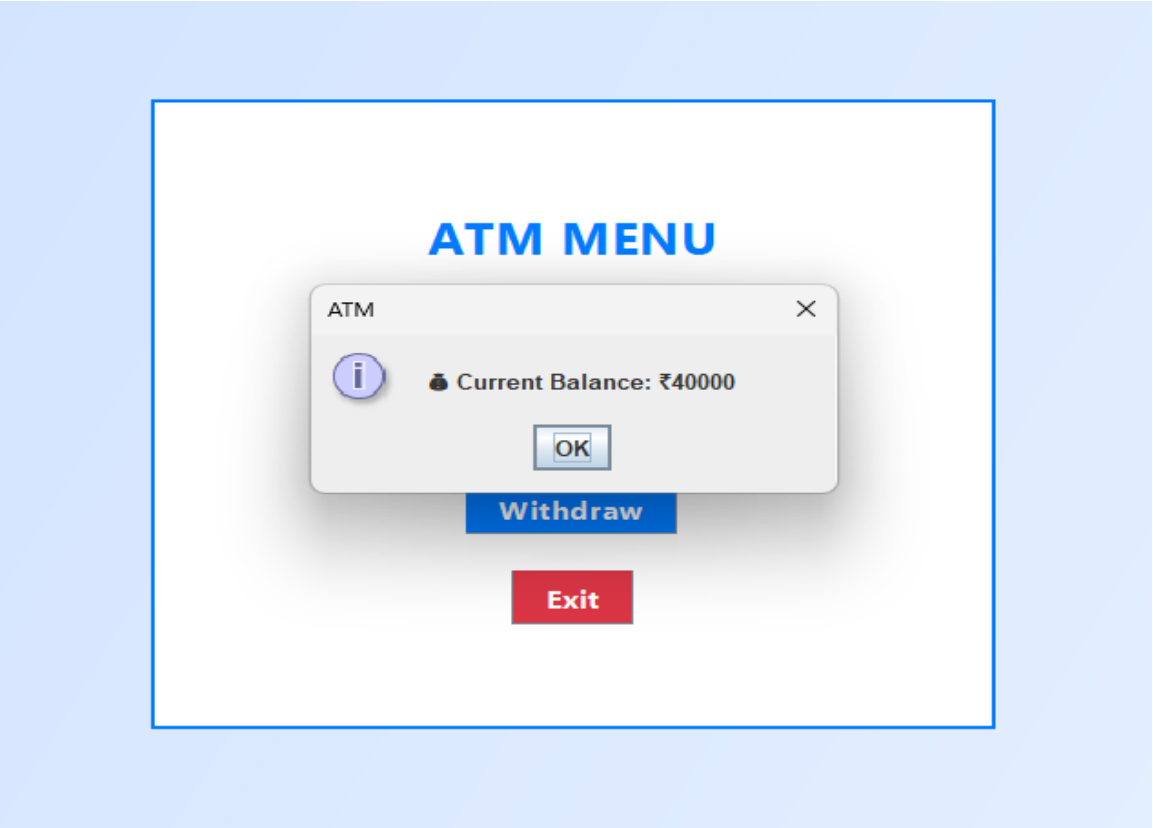
DEPOSIT



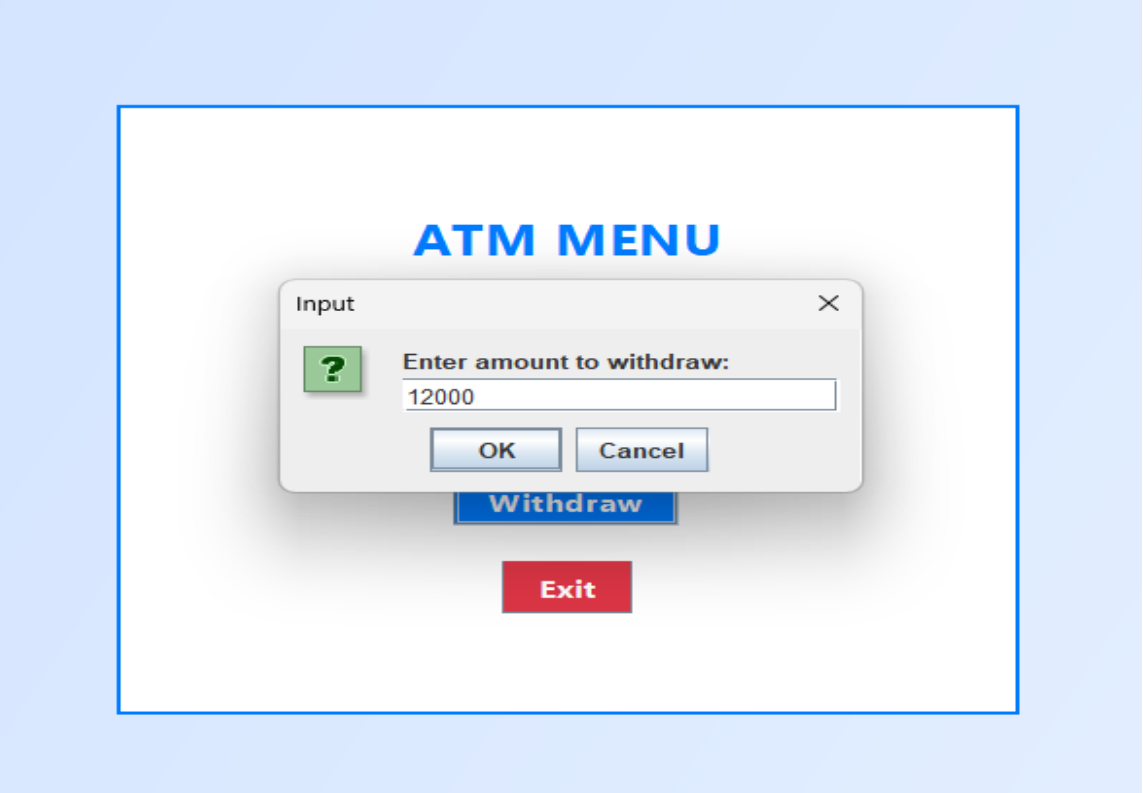
SUCCESS MESSAGE DISPLAYED AFTER DEPOSIT OF MONEY



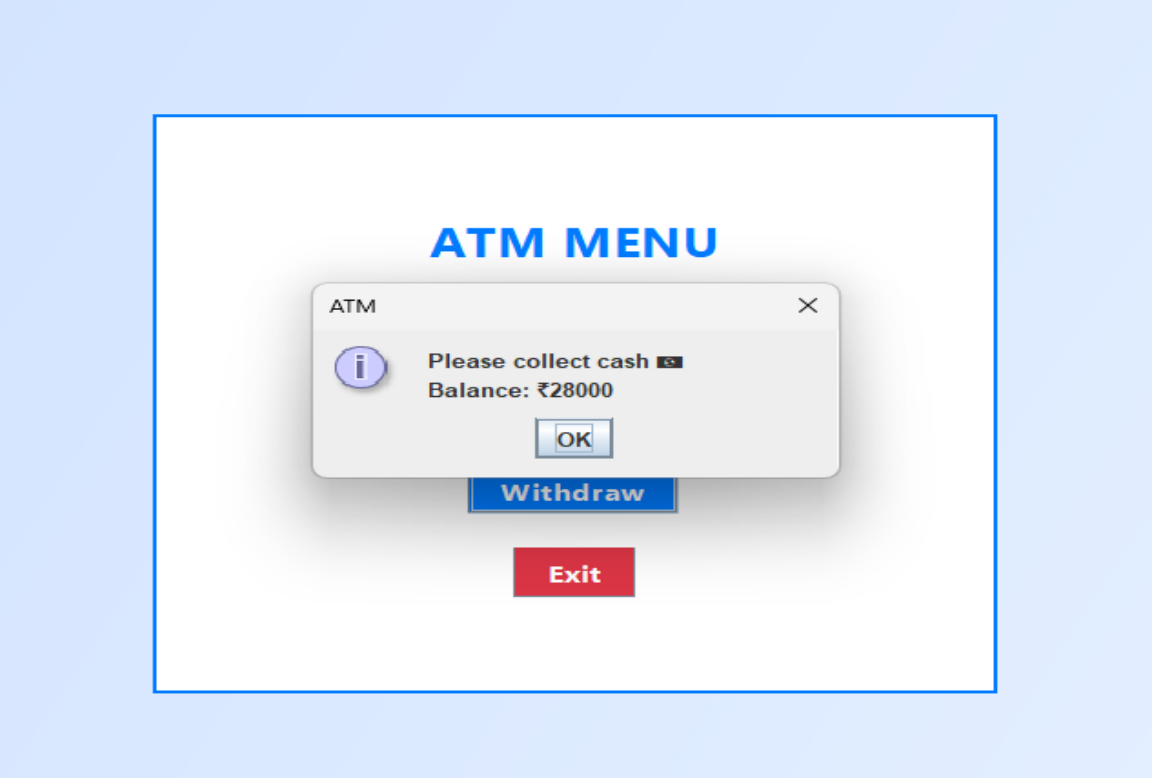
BALANCE UPDATED AFTER DEPOSTING MONEY



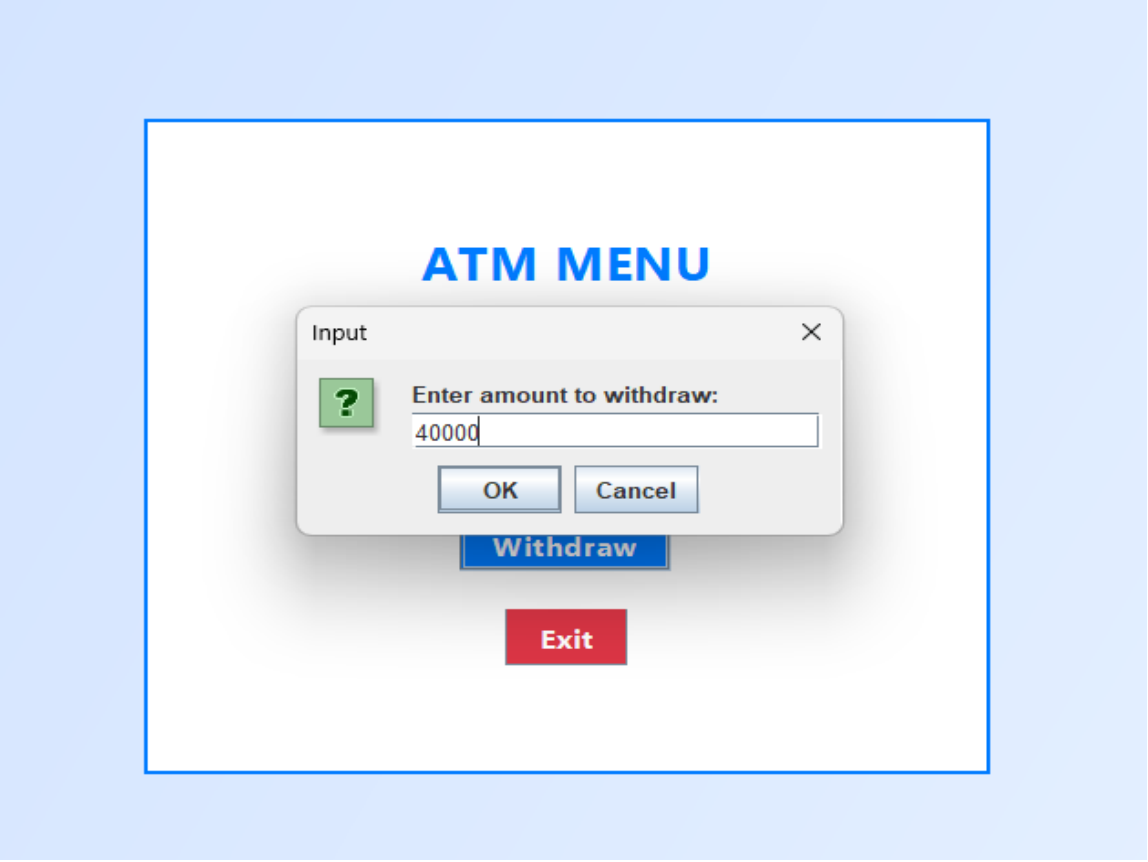
WITHDRAW



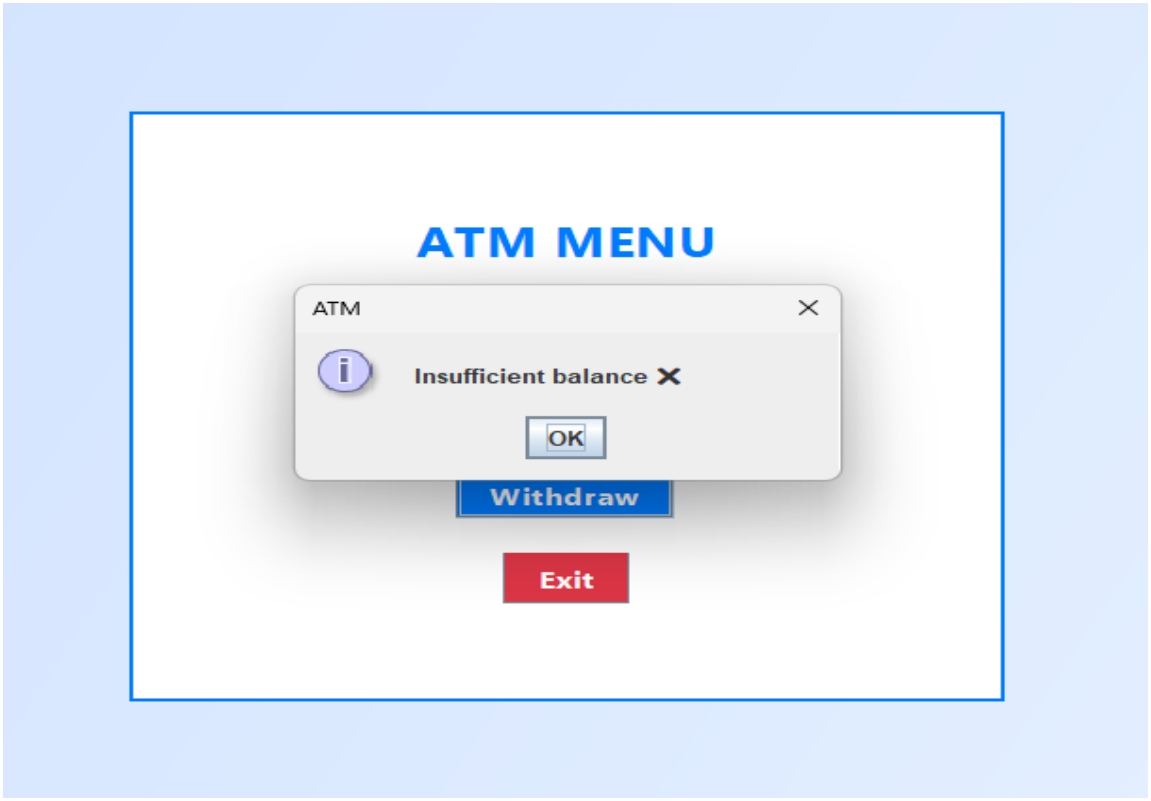
MESSAGE DISPLAYED TO COLLECT CASH WITH UPDATED BALANCE



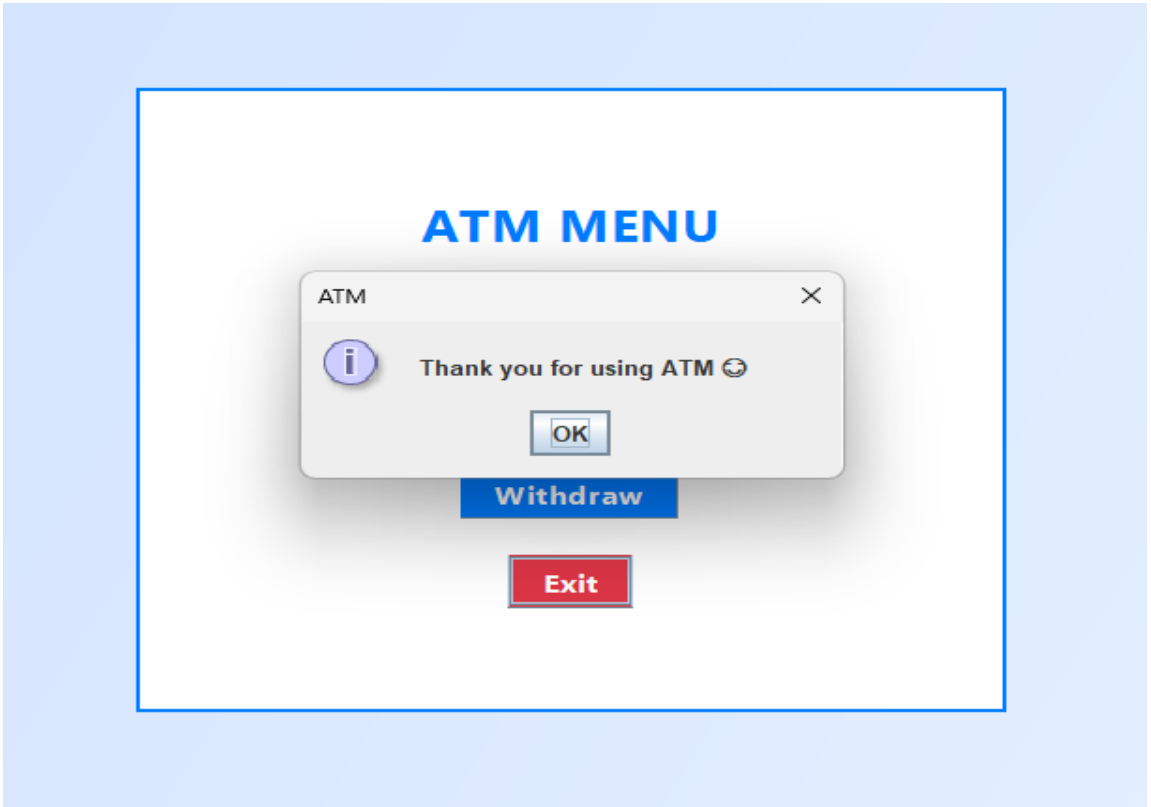
IF REQUEST TO WITHDRAW IS MORE THAN AVAILABLE BALANCE



ERROR MESSAGE DISPLAYED



EXIT



REFERENCES:

1. **Herbert Schildt, *Java: The Complete Reference*, McGraw-Hill**
2. **Java Swing Tutorials – https://docs.oracle.com/javase/tutorial/uiswing/**
3. **Lecture notes and course materials**