**Bank Management System using Java Swing**

**A Project Report**

***Submitted by***

**Akash Bahuguna**

**UID: 22BCA10080**

***in partial fulfillment for the award of the degree of***

**Bachelors in Computer Application**

**IN**

**University Institute of Computing**



**Chandigarh University**

**April 2025**



# **BONAFIDE CERTIFICATE**

Certified that this project report “**Bank management System using Java Swing**” is the Bonafide work of **Akash Bahuguna** who carried out the project work under my/our supervision.

……………………. .……………………..

(Signature of HOD) (Signature of Supervisor)

**Mrs. Kavita Gupta Ms. Preeti Kaur**

**University Institute of Computing**

Submitted for the project viva voce examination held on

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# **Abstract**

The **Bank Management System using Java Swing** is a simple yet functional desktop application that allows users to manage basic banking operations through a graphical user interface (GUI). Developed using **Java AWT and Swing**, the system provides an interactive platform to perform key operations such as **creating accounts**, **depositing funds**, **withdrawing money**, and **viewing account details**.

The application maintains an in-memory list of accounts using Java collections, ensuring quick access and updates. Each account is uniquely identified by an account number and holds essential information like the account holder’s name and balance. The interface is designed using components like JFrame, JPanel, JLabel, JTextField, JButton, and JTextArea, offering a user-friendly layout for seamless interaction.

This project demonstrates the practical implementation of object-oriented programming (OOP) principles such as encapsulation and abstraction, and emphasizes the use of event-driven programming using action listeners. It serves as a foundational project for understanding Java GUI development and banking logic integration.

**Introduction & Objective**

**Introduction**

The **Bank Management System using Java Swing** is a GUI-based application designed to simulate basic banking operations in a user-friendly environment. The system enables users to create bank accounts, deposit and withdraw money, and view account details through an interactive graphical interface. Built with Java's AWT and Swing libraries, the project emphasizes the importance of event-driven programming and the practical application of object-oriented principles.

This project is ideal for beginners in Java who wish to understand how GUI applications function and how backend logic (account handling) can be integrated with frontend components (buttons, forms, and displays). The application is entirely standalone and runs without a database, storing account information in memory using an ArrayList.

**Objectives**

The main objectives of this project are:

1. **To develop a GUI-based application** for managing simple banking operations using Java Swing and AWT.
2. **To implement core banking features** such as account creation, deposit, withdrawal, and account detail viewing.
3. **To demonstrate object-oriented programming** concepts such as encapsulation, abstraction, and class-object interactions.
4. **To provide hands-on experience** with Java Swing components, layout managers, and event handling.
5. **To ensure user-friendly interaction** by using intuitive labels, buttons, text fields, and messages.
6. **To simulate a real-time banking environment** with responsive feedback and basic error handling.

**Flow Diagram**

**┌───────────────┐**

**│ Start Program │**

**└──────┬────────┘**

**▼**

**┌────────────────────┐**

**│ Enter Account Info │**

**└──────┬──────┬──────┘**

**▼ ▼**

**[Create] [Actions]**

**/ | \**

**▼ ▼ ▼**

**[Deposit] [Withdraw] [View]**

**\ | /**

**▼ ▼ ▼**

**Show Confirmation**

**▼**

**┌────────────┐**

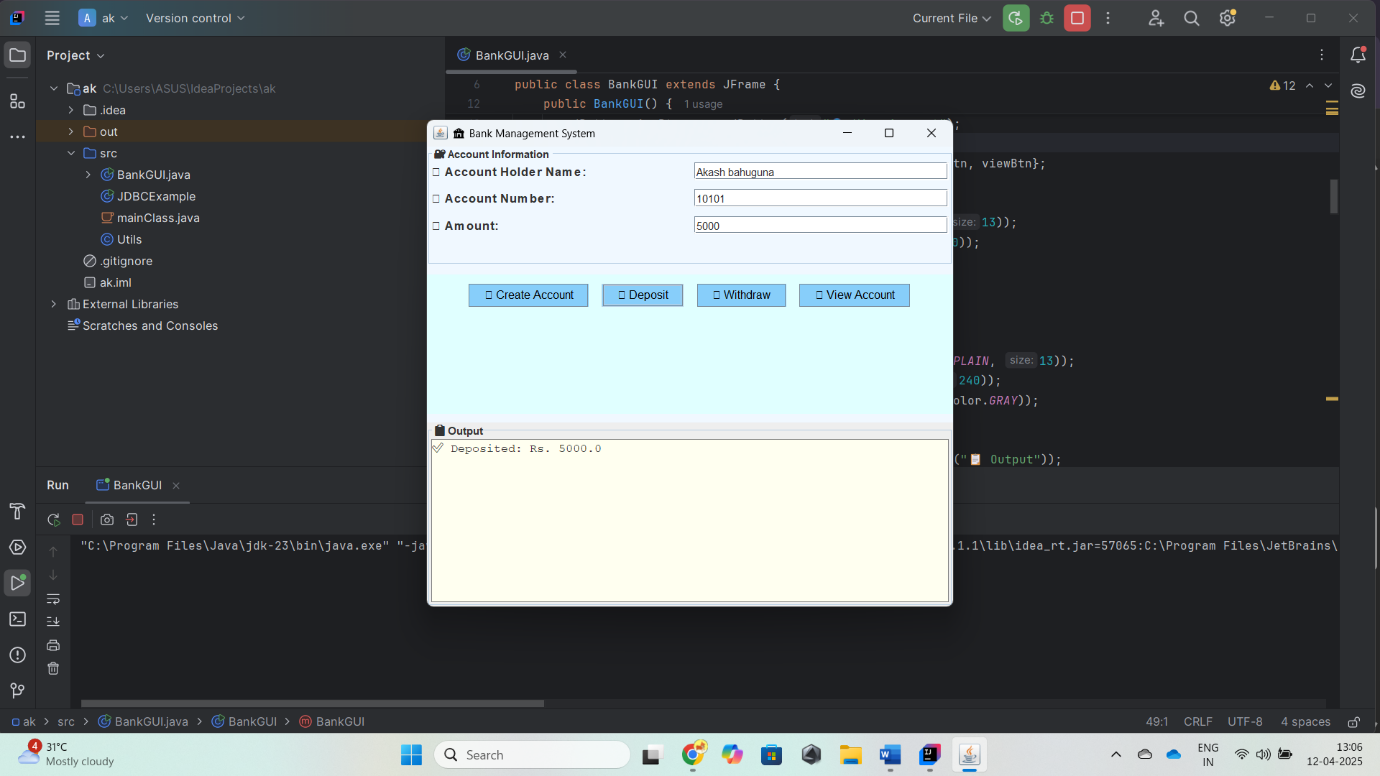
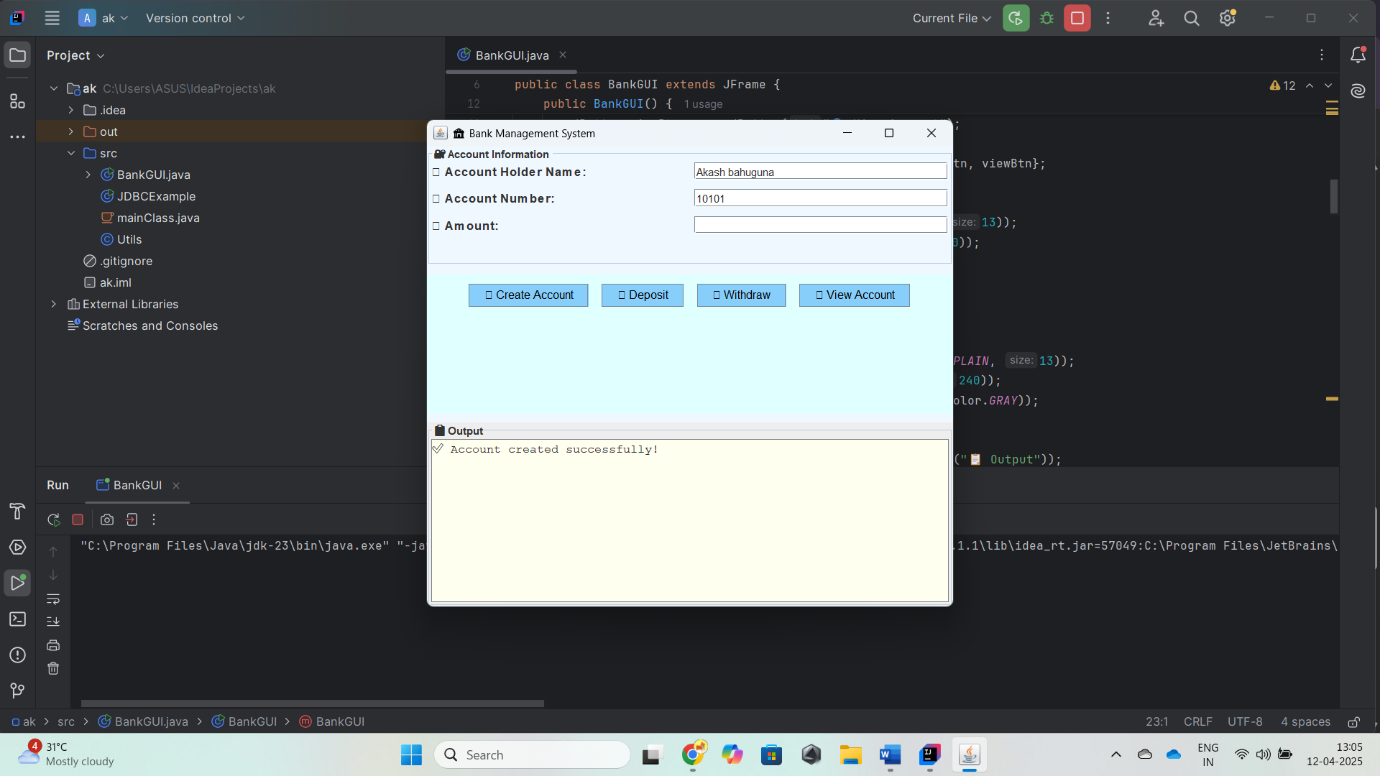
**│ Exit │**

**└────────────┘**

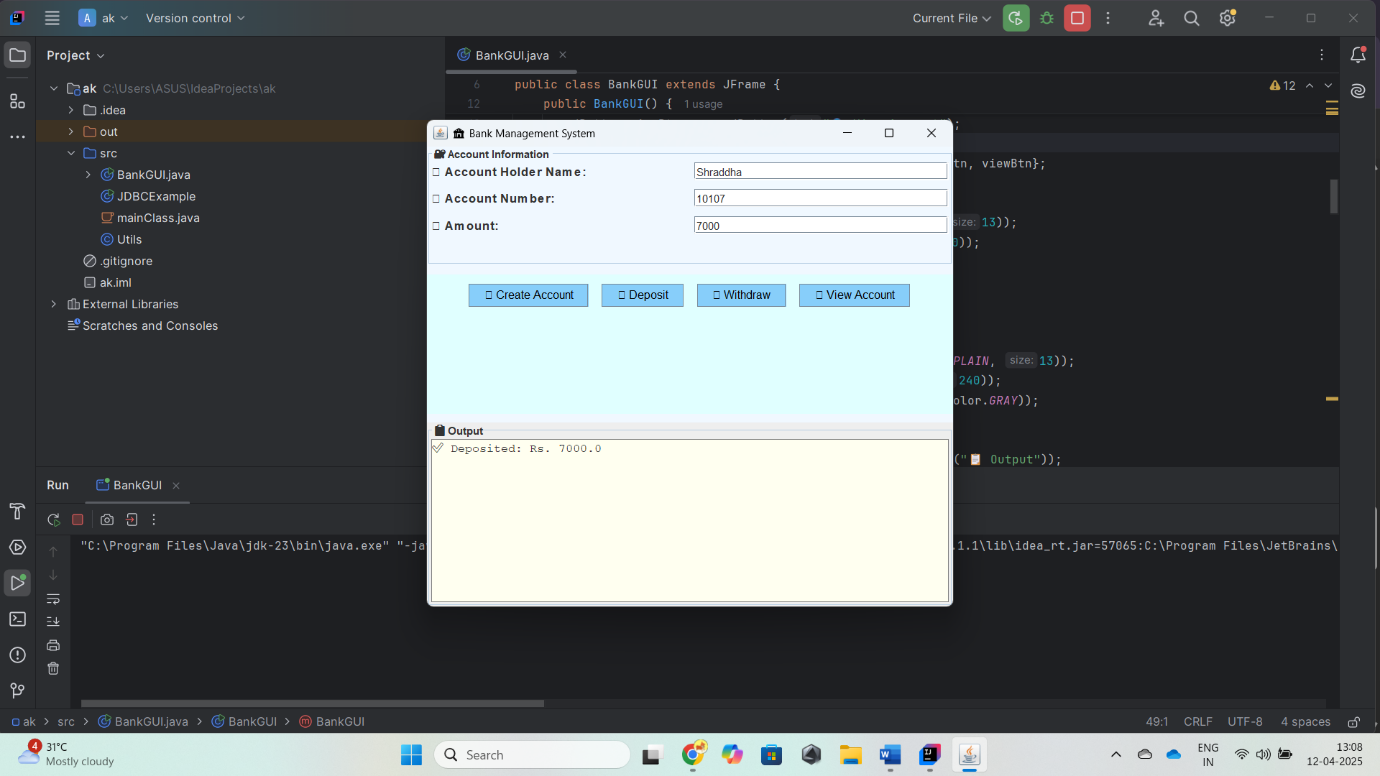
**Implementation**

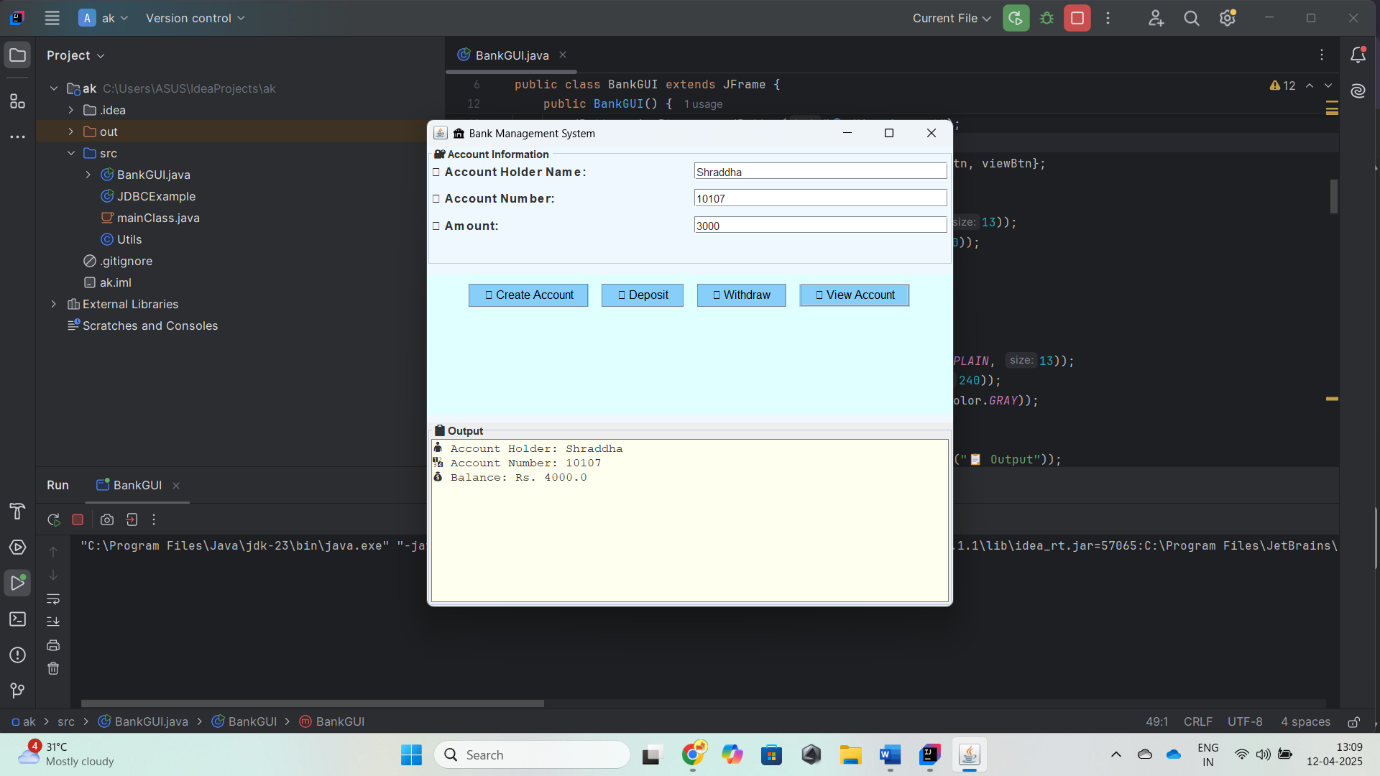
**Sample Code Snippet:**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.ArrayList;  
  
public class BankGUI extends JFrame {  
 private ArrayList<BankAccount> accounts = new ArrayList<>();  
  
 private JTextField nameField, numberField, amountField;  
 private JTextArea outputArea;  
  
 public BankGUI() {  
 setTitle("🏦 Bank Management System");  
 setSize(600, 550);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
 setLayout(new BorderLayout(10, 10));  
 getContentPane().setBackground(new Color(240, 248, 255));  
  
 JPanel topPanel = new JPanel(new GridLayout(4, 2, 10, 10));  
 topPanel.setBorder(BorderFactory.*createTitledBorder*("🔐 Account Information"));  
 topPanel.setBackground(new Color(240, 248, 255));  
  
 JLabel nameLabel = new JLabel("👤 Account Holder Name:");  
 nameLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 topPanel.add(nameLabel);  
 nameField = new JTextField();  
 topPanel.add(nameField);  
  
 JLabel numberLabel = new JLabel("🔢 Account Number:");  
 numberLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 topPanel.add(numberLabel);  
 numberField = new JTextField();  
 topPanel.add(numberField);  
  
 JLabel amountLabel = new JLabel("💵 Amount:");  
 amountLabel.setFont(new Font("Arial", Font.*BOLD*, 14));  
 topPanel.add(amountLabel);  
 amountField = new JTextField();  
 topPanel.add(amountField);  
  
 JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.*CENTER*, 15, 10));  
 buttonPanel.setBackground(new Color(224, 255, 255));  
  
 JButton createBtn = new JButton("➕ Create Account");  
 JButton depositBtn = new JButton("💰 Deposit");  
 JButton withdrawBtn = new JButton("💸 Withdraw");  
 JButton viewBtn = new JButton("🔍 View Account");  
  
 JButton[] buttons = {createBtn, depositBtn, withdrawBtn, viewBtn};  
 for (JButton btn : buttons) {  
 btn.setFocusPainted(false);  
 btn.setFont(new Font("Arial", Font.*PLAIN*, 13));  
 btn.setBackground(new Color(135, 206, 250));  
 btn.setForeground(Color.*BLACK*);  
 buttonPanel. Add(btn);  
 }  
  
 outputArea = new JTextArea(10, 40);  
 outputArea.setFont(new Font("Monospaced", Font.*PLAIN*, 13));  
 outputArea.setBackground(new Color(255, 255, 240));  
 outputArea.setBorder(BorderFactory.*createLineBorder*(Color.*GRAY*));  
 outputArea.setEditable(false);  
 JScrollPane scrollPane = new JScrollPane(outputArea);  
 scrollPane.setBorder(BorderFactory.*createTitledBorder*("📋 Output"));  
  
 add(topPanel, BorderLayout.*NORTH*);  
 add(buttonPanel, BorderLayout.*CENTER*);  
 add(scrollPane, BorderLayout.*SOUTH*);  
  
 createBtn.addActionListener(e -> createAccount());  
 depositBtn.addActionListener(e -> deposit());  
 withdrawBtn.addActionListener(e -> withdraw());  
 viewBtn.addActionListener(e -> viewAccount());  
  
 setVisible(true);  
 }  
  
 private void createAccount() {  
 String name = nameField.getText();  
 if (name.isEmpty() || numberField.getText().isEmpty()) {  
 outputArea.setText("❗ Please enter all details to create an account.\n");  
 return;  
 }  
  
 try {  
 int number = Integer.*parseInt*(numberField.getText());  
  
 for (BankAccount acc : accounts) {  
 if (acc.getAccountNumber() == number) {  
 outputArea.setText("⚠️ Account already exists!\n");  
 return;  
 }  
 }  
  
 accounts.add(new BankAccount(name, number));  
 outputArea.setText("✅ Account created successfully!\n");  
 } catch (NumberFormatException e) {  
 outputArea.setText("❌ Invalid account number. Please enter digits only.\n");  
 }  
 }  
  
 private void deposit() {  
 try {  
 int number = Integer.*parseInt*(numberField.getText());  
 double amt = Double.*parseDouble*(amountField.getText());  
  
 for (BankAccount acc : accounts) {  
 if (acc.getAccountNumber() == number) {  
 acc.deposit(amt);  
 outputArea.setText("✅ Deposited: Rs. " + amt + "\n");  
 return;  
 }  
 }  
 outputArea.setText("❌ Account not found!\n");  
 } catch (NumberFormatException e) {  
 outputArea.setText("❌ Please enter valid numbers in account number and amount fields.\n");  
 }  
 }  
  
 private void withdraw() {  
 try {  
 int number = Integer.*parseInt*(numberField.getText());  
 double amt = Double.*parseDouble*(amountField.getText());  
  
 for (BankAccount acc : accounts) {  
 if (acc.getAccountNumber() == number) {  
 if (acc.getBalance() >= amt) {  
 acc.withdraw(amt);  
 outputArea.setText("✅ Withdrawn: Rs. " + amt + "\n");  
 } else {  
 outputArea.setText("❗ Insufficient balance.\n");  
 }  
 return;  
 }  
 }  
 outputArea.setText("❌ Account not found!\n");  
 } catch (NumberFormatException e) {  
 outputArea.setText("❌ Please enter valid numbers in account number and amount fields.\n");  
 }  
 }  
  
 private void viewAccount() {  
 try {  
 int number = Integer.*parseInt*(numberField.getText());  
  
 for (BankAccount acc : accounts) {  
 if (acc.getAccountNumber() == number) {  
 outputArea.setText("👤 Account Holder: " + acc.getAccountHolderName() +  
 "\n🔢 Account Number: " + acc.getAccountNumber() +  
 "\n💰 Balance: Rs. " + acc.getBalance() + "\n");  
 return;  
 }  
 }  
 outputArea.setText("❌ Account not found!\n");  
 } catch (NumberFormatException e) {  
 outputArea.setText("❌ Invalid account number format.\n");  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(BankGUI::new);  
 }  
}  
  
class BankAccount {  
 private String accountHolderName;  
 private int accountNumber;  
 private double balance;  
  
 public BankAccount(String name, int number) {  
 this.accountHolderName = name;  
 this.accountNumber = number;  
 this.balance = 0.0;  
 }  
  
 public void deposit(double amount) {  
 balance += amount;  
 }  
  
 public void withdraw(double amount) {  
 if (amount <= balance) {  
 balance -= amount;  
 }  
 }  
  
 public String getAccountHolderName() {  
 return accountHolderName;  
 }  
  
 public int getAccountNumber() {  
 return accountNumber;  
 }  
  
 public double getBalance() {  
 return balance;  
 }  
}

**OUTPUT** A screenshot of a computer

AI-generated content may be incorrect.A computer screen with a white box

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**✅ Conclusion**

This project, titled "Bank Management System using Java Swing", effectively implements a simple banking application that allows users to perform basic operations such as creating a bank account, depositing money, withdrawing funds, and viewing account details through a graphical user interface.

Using Java Swing components like JFrame, JPanel, JTextField, JTextArea, and JButton, the system offers an interactive and user-friendly interface. The application maintains account information using an ArrayList of custom BankAccount objects, ensuring proper object-oriented structure.

All operations are performed in real time without the need for backend databases, demonstrating a practical and lightweight solution suitable for learning GUI development and event handling in Java.

The project showcases:

* Good use of event-driven programming via ActionListeners.
* Proper input validation and error handling.
* Aesthetic UI elements like emojis and custom fonts for enhanced user experience.