# BANK MANAGEMENT SYSTEM



An

Object-Oriented Programming through Java Course Project Report in partial fulfilment of the degree

**Bachelor of Technology**

In

**Computer Science & Engineering**

**By**

CH.ASHWINI 2103A52028

P.CHAITHRA SRI 2103A52027

B.SHASHANKA 2103A52005

**Submitted to**

**Gotte Ranjith Kumar**

**Assistant Professor**

**Department of Computer Science And Engineering**



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

**This is to certify that the Object Oriented Programming through Java – Course Project Report entitled “BANK MANAGEMENT SYSTEM is a record of bonafide work carried out by the student ASHWINI ,CHAITHRA SRI AND SHASHANKA *bearing Roll No(s) 2103A52028 ,2103A52027 AND 2103A52005 during the academic* year 2023-2024in partial fulfillment of the award of the degree of *Bachelor of Technology* in Computer Science & Engineering by the SR UNIVERSITY,WARANGAL.**

**Lab In-charge Head of the Department**

# TABLE OF CONTENTS

# CONTENTS PAGE NO.

# Abstract 4

# Objective of the project 5

# Definition of the elements used in the project 6

# Design 7-11

# Implementation 12-16

# Result screen 17

# Conclusion 18

**ABSTRACT**

Bank management system can be considered as a most important thing in economic world.In the present scenario the banking sector is the common need in everyday life.In day to day life we face the problems and then we realize something is not done in this sector like we want to change the location (branch) of our account then we need to fill the application and then some day waiting to complete bank process. In this process amount of time is more as well as here occur manual work which is increases man power. Also in current scenario aadhar card linking is must with bank account and it is possible through the ATM but if in urgent we want to link aadhar it may be not possible there is no ATM are available in that case we provide this facility through the our project i.e. Bank management system.

# OBJECTIVE OF THE PROJECT

The project’s main goal is to create an online banking system for banks. All banking work is done manually in the current system. To withdraw or deposit money, the user must go to the bank. Today, it is also hard to find account information for people who have accounts in the banking system.

The main goal of the bank management system project is to create an organic and optimal software of interaction between the various banking components. This is to maximize the profit of the banking mechanism. The implementation of competent bank management procedures is significantly responsible for the successful optimization of the bank’s productivity and activities.

# DEFINITIONS OF THE ELEMENTS THAT ARE USED IN THE PROJECT

SWINGS: Java Swing is a lightweight Java graphical user interface (GUI) widget toolkit that includes a rich set of widgets.

PACKAGES: A package is a namespace that organizes a set of related classes and interfaces.

ACTIONLISTENER: the listener interface for receiving action events .

ACTIONPERFORMER: An action event occurs, whenever an action is performed by the user.

JPANEL: Panel, a part of the Java Swing package, is a container that can store a group of components.

JBUTTON: The JButton class is used to create a labeled button that has platform independent implementation.

JFRAME: JFrame is a top-level container that provides a window on the screen

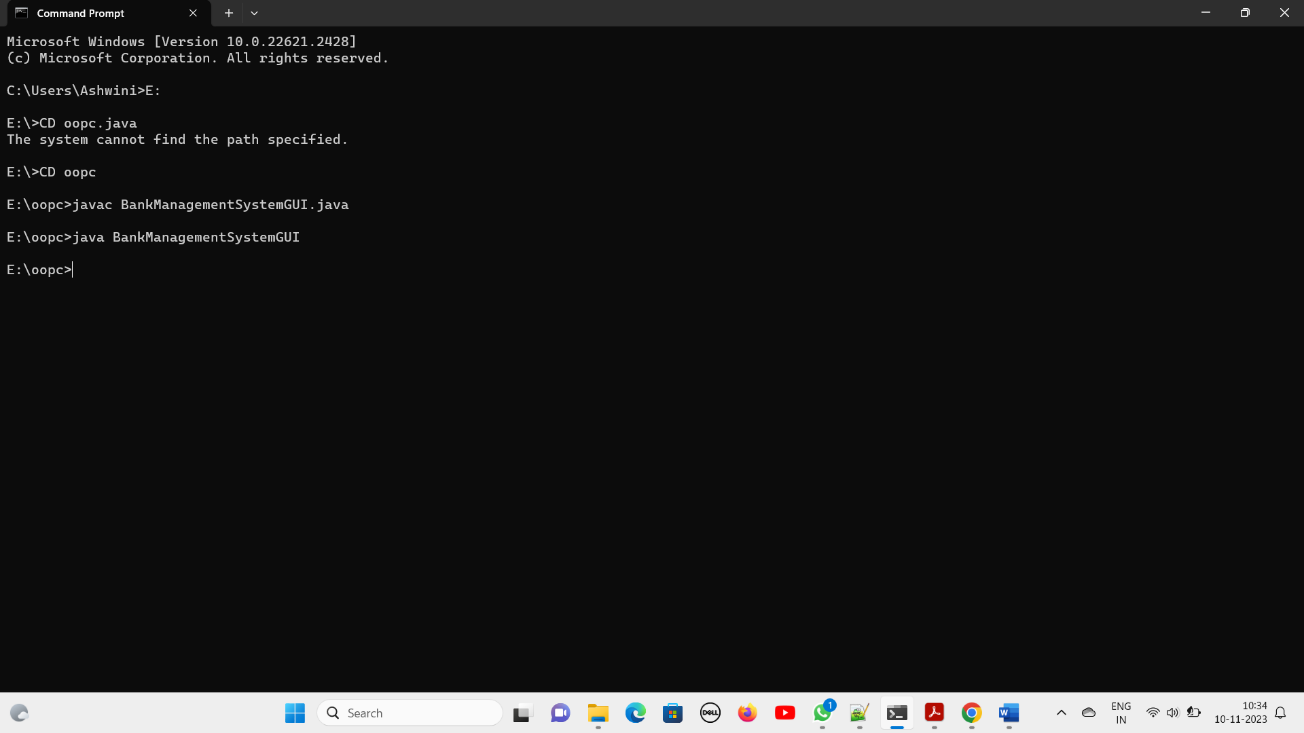
JLABEL:JLabel is a class of java Swing,JLabel is used to display a short string or an image icon.

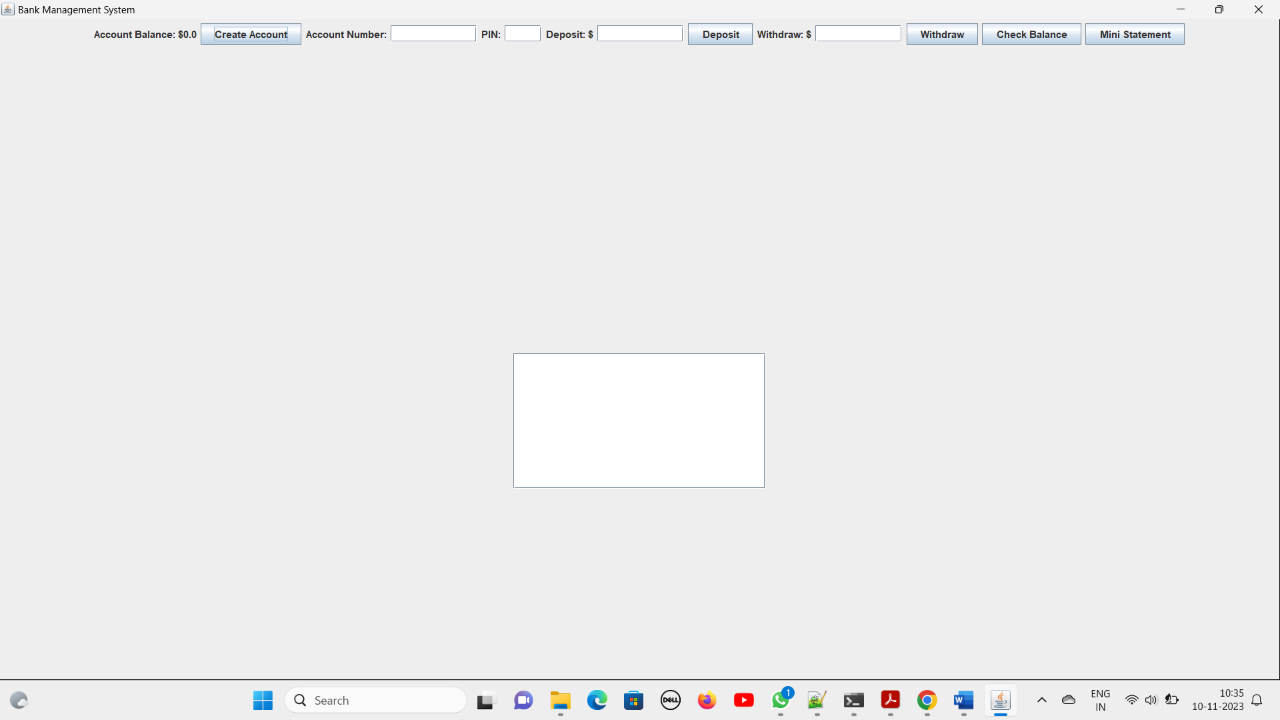
JTEXTFIELD: JTextField is a lightweight component that allows the editing of a single line of text.

6

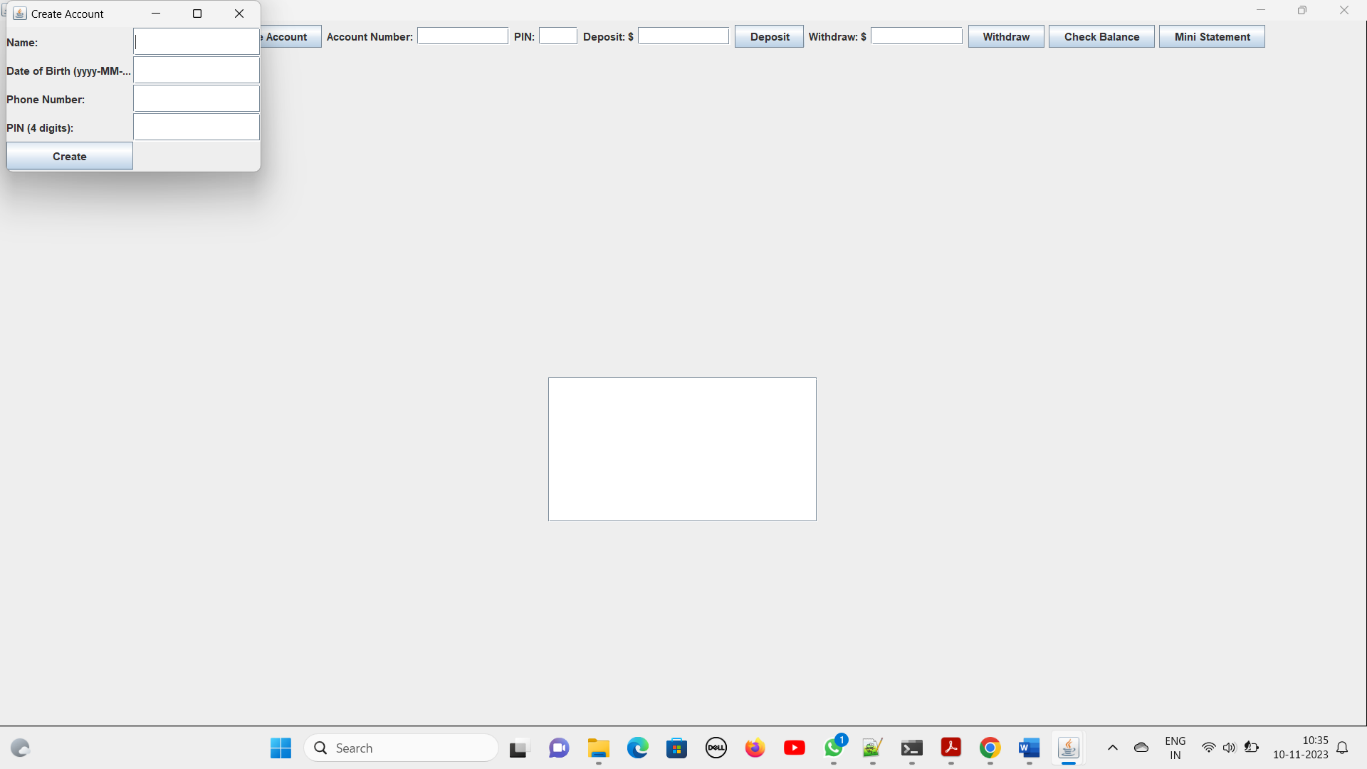
# DESIGN

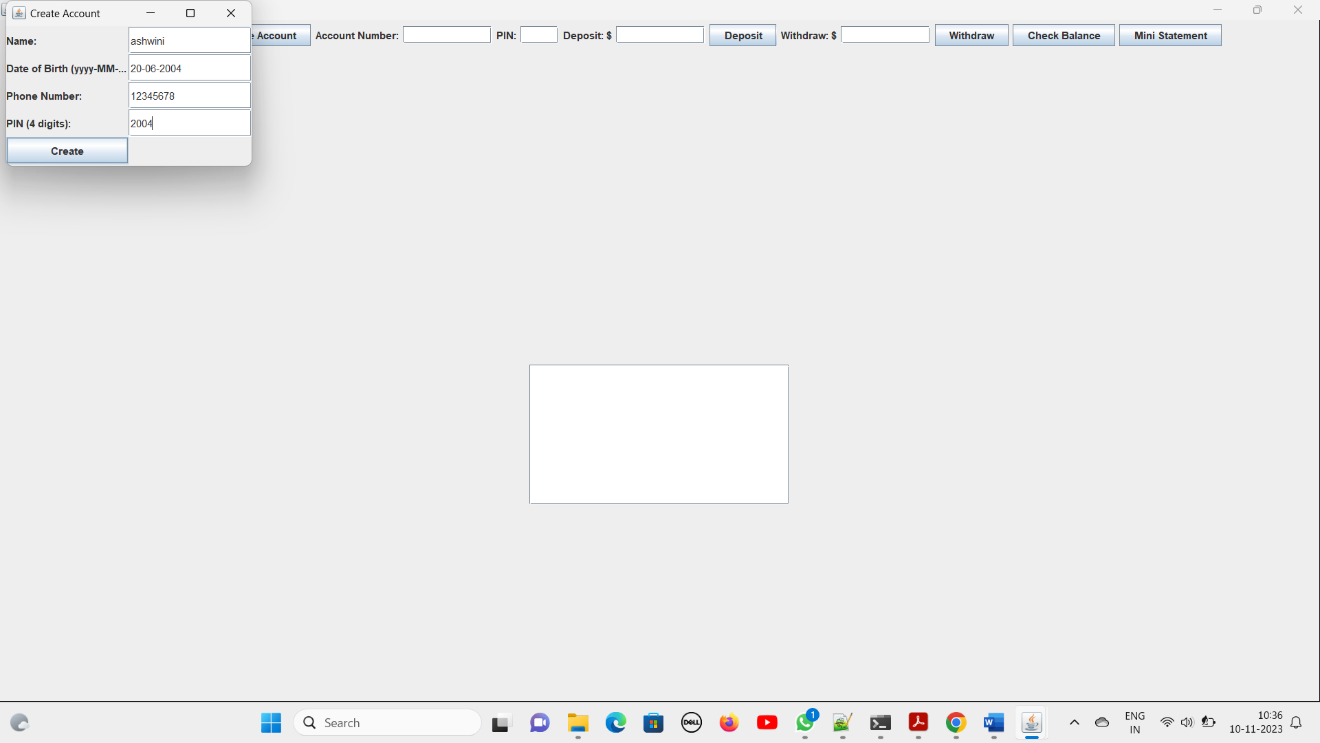
**SCREENS:**



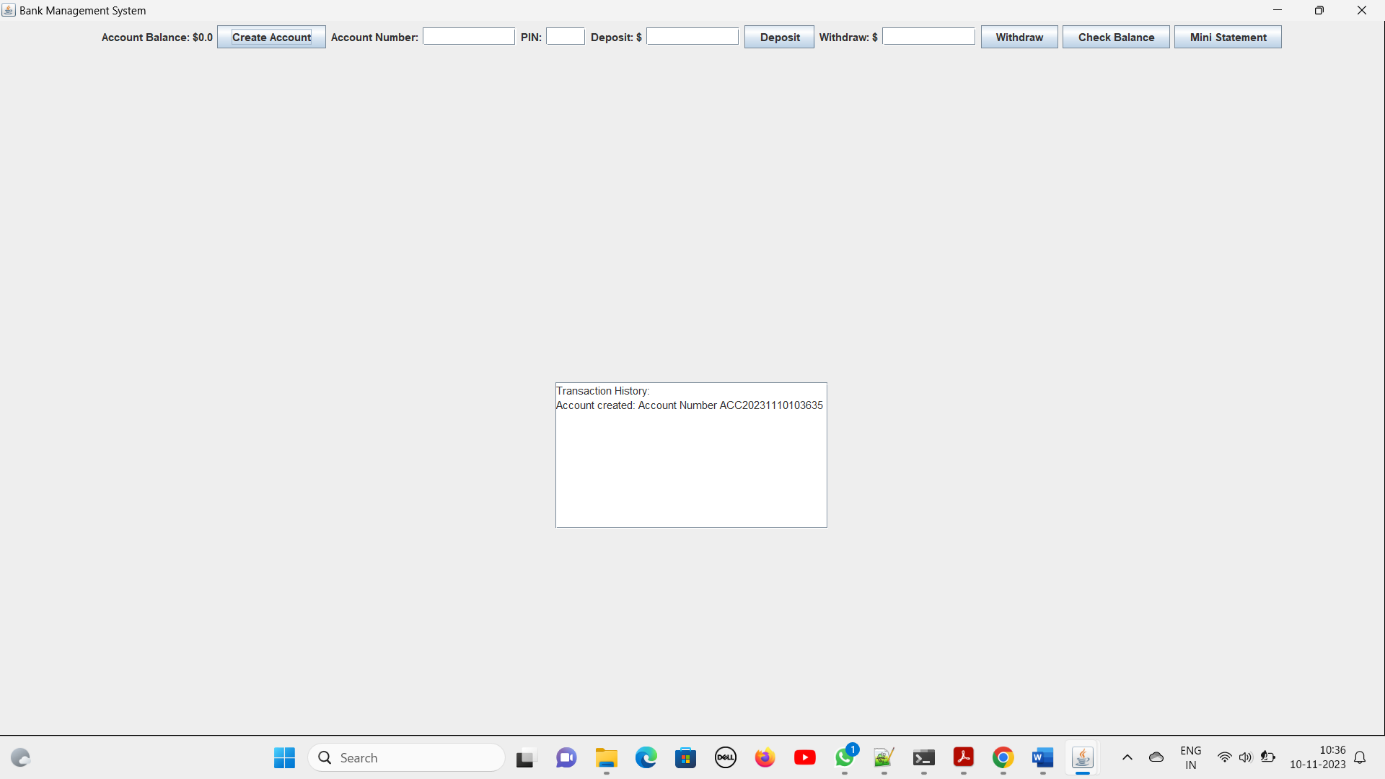
****

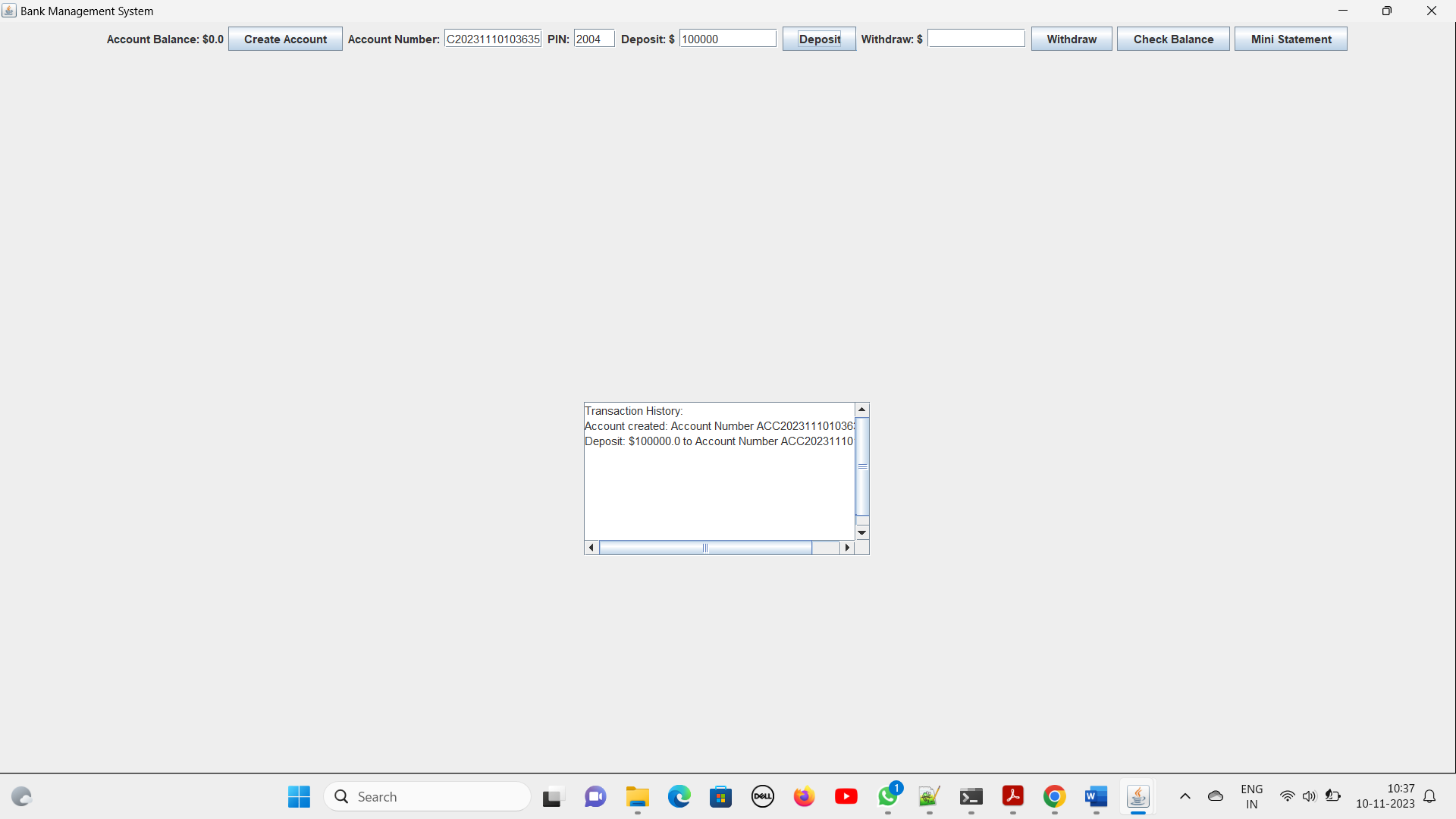
7



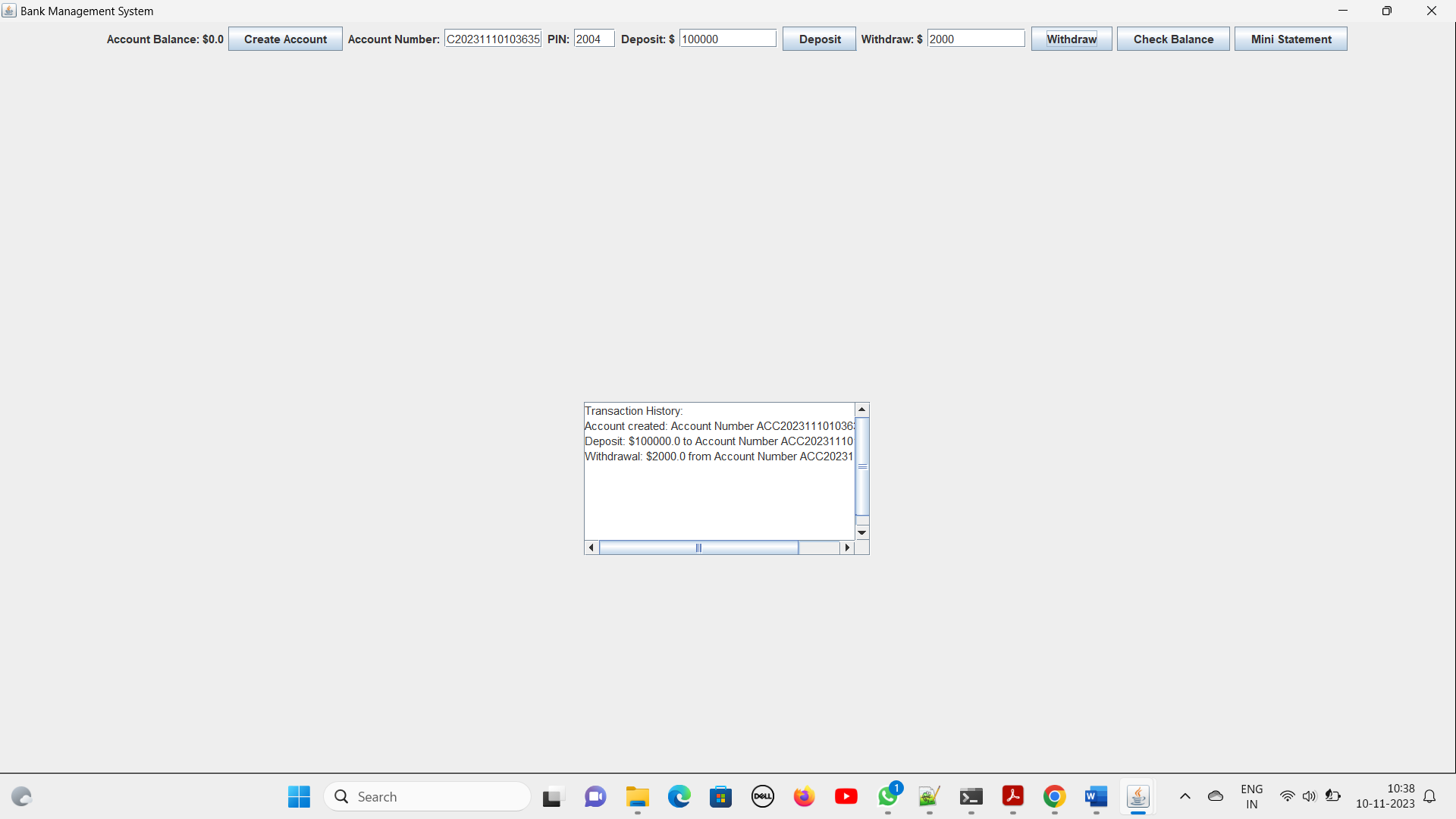


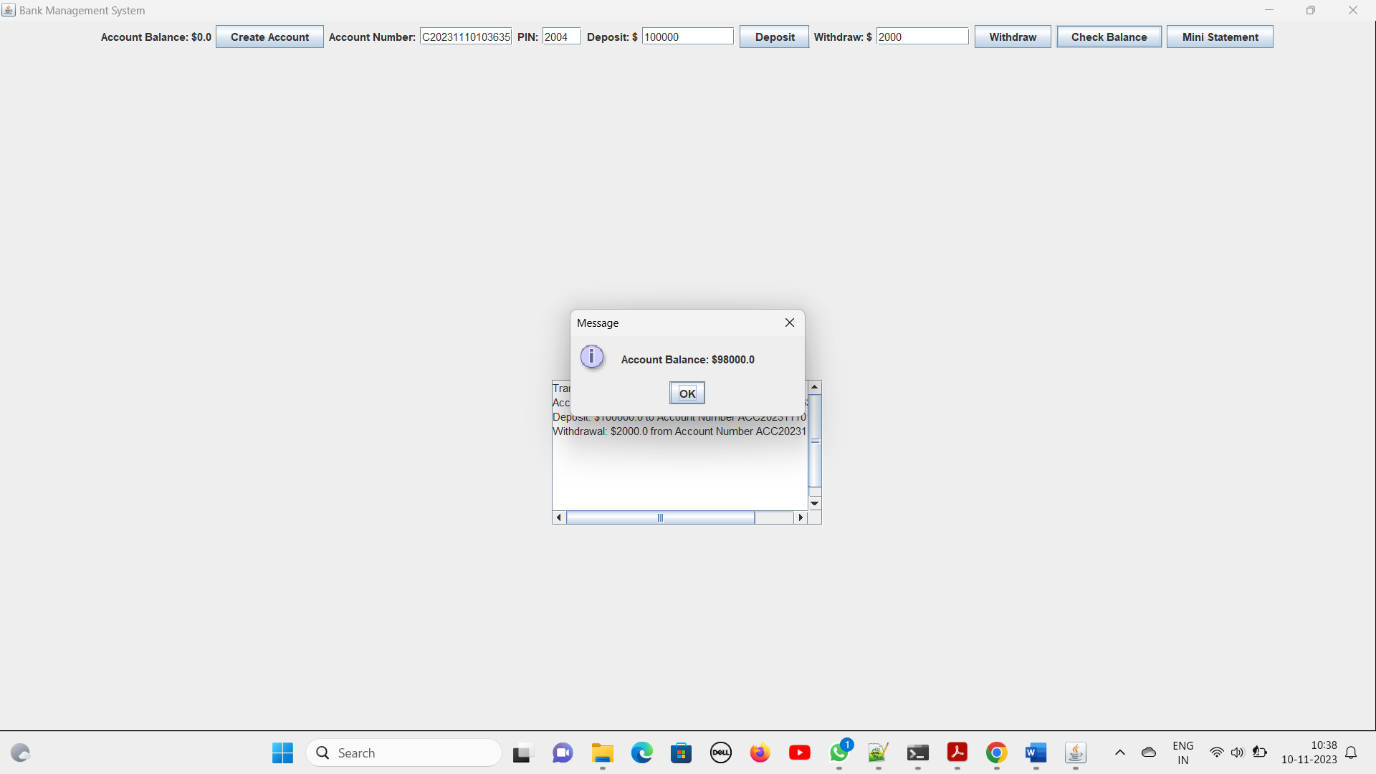
8

****

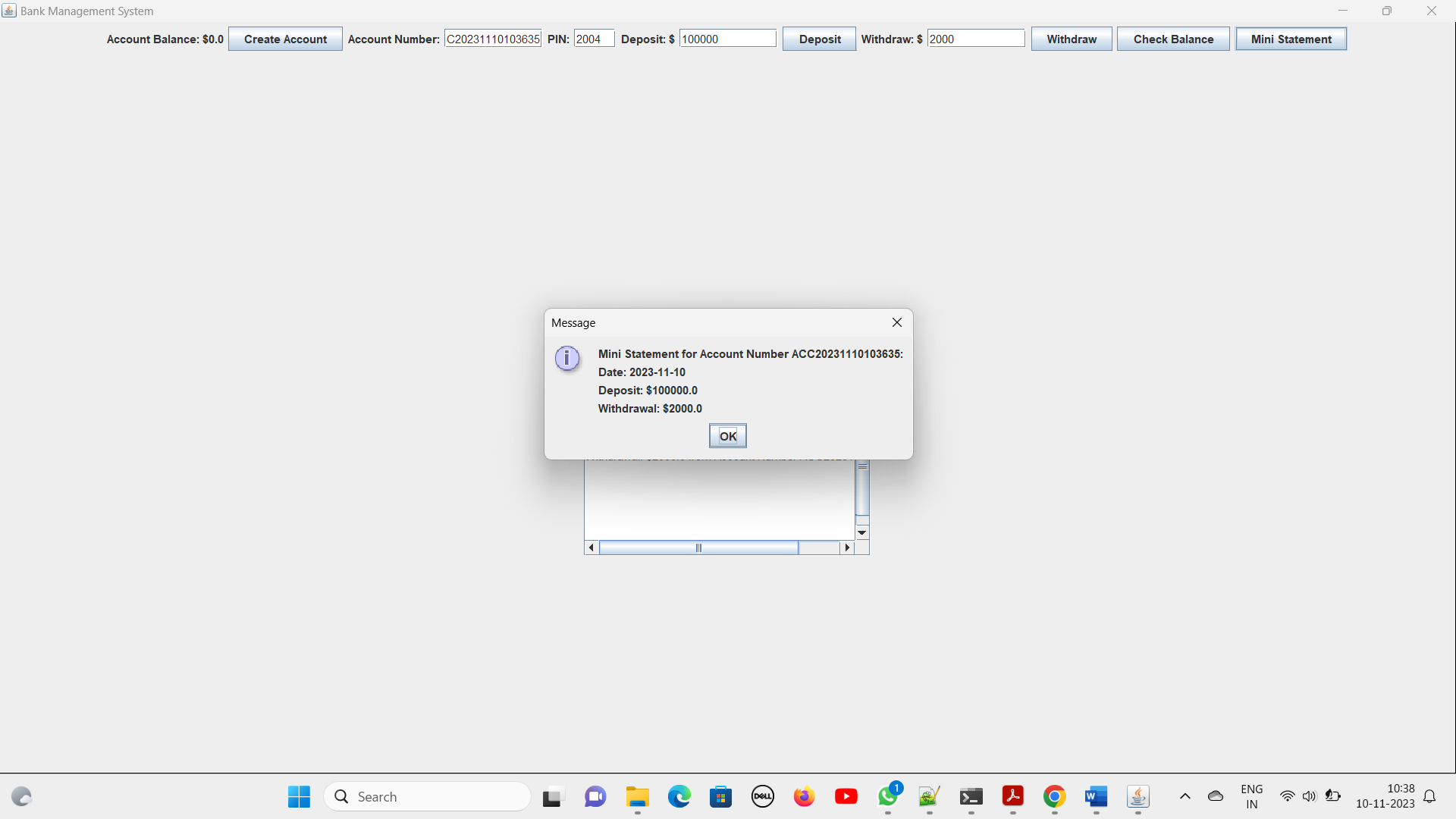


9



****

10

****

11

# IMPLEMENTATION

**CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.List;

public class BankManagementSystemGUI {

private List<Account> accounts = new ArrayList<>();

private List<String> transactionHistory = new ArrayList<>();

private JTextArea transactionHistoryArea;

private JTextField pinField;

private JTextField withdrawField; // Added withdrawField

public BankManagementSystemGUI() {

JFrame frame = new JFrame("Bank Management System");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 400);

JLabel balanceLabel = new JLabel("Account Balance: $0.0");

JButton createAccountButton = new JButton("Create Account");

JButton depositButton = new JButton("Deposit");

JButton withdrawButton = new JButton("Withdraw");

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

JButton miniStatementButton = new JButton("Mini Statement");

JTextField depositField = new JTextField(10);

withdrawField = new JTextField(10); // Initialize withdrawField

JTextField accountNumberField = new JTextField(10);

pinField = new JTextField(4);

transactionHistoryArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(transactionHistoryArea);

createAccountButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

createAccountDialog();

}

});

depositButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String accountNumber = accountNumberField.getText();

double amount = Double.parseDouble(depositField.getText());

performTransaction(accountNumber, "Deposit", amount);

}

});

withdrawButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String accountNumber = accountNumberField.getText();

String enteredPin = pinField.getText();

performWithdrawal(accountNumber, enteredPin);

}

});

checkBalanceButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String accountNumber = accountNumberField.getText();

displayBalance(accountNumber);

}

});

miniStatementButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String accountNumber = accountNumberField.getText();

displayMiniStatement(accountNumber);

}

});

JPanel panel = new JPanel();

panel.setLayout(new FlowLayout());

panel.add(balanceLabel);

panel.add(createAccountButton);

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

panel.add(accountNumberField);

panel.add(new JLabel("PIN:"));

panel.add(pinField);

panel.add(new JLabel("Deposit: $"));

panel.add(depositField);

panel.add(depositButton);

panel.add(new JLabel("Withdraw: $"));

panel.add(withdrawField);

panel.add(withdrawButton);

panel.add(checkBalanceButton);

panel.add(miniStatementButton);

JPanel historyPanel = new JPanel();

historyPanel.setLayout(new FlowLayout());

historyPanel.add(scrollPane);

frame.add(panel);

frame.add(historyPanel);

frame.setLayout(new GridLayout(2, 1));

frame.setVisible(true);

}

private void createAccountDialog() {

JFrame dialogFrame = new JFrame("Create Account");

dialogFrame.setSize(300, 200);

JPanel dialogPanel = new JPanel(new GridLayout(5, 2));

dialogPanel.add(new JLabel("Name:"));

JTextField nameField = new JTextField(20);

dialogPanel.add(nameField);

dialogPanel.add(new JLabel("Date of Birth (yyyy-MM-dd):"));

JTextField dobField = new JTextField(20);

dialogPanel.add(dobField);

dialogPanel.add(new JLabel("Phone Number:"));

JTextField phoneField = new JTextField(20);

dialogPanel.add(phoneField);

dialogPanel.add(new JLabel("PIN (4 digits):"));

JTextField pinField = new JTextField(4);

dialogPanel.add(pinField);

JButton createButton = new JButton("Create");

createButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String name = nameField.getText();

String dob = dobField.getText();

String phone = phoneField.getText();

String pin = pinField.getText();

String accountNumber = generateAccountNumber();

accounts.add(new Account(accountNumber, name, dob, phone, pin));

recordTransaction("Account created: Account Number " + accountNumber);

dialogFrame.dispose();

}

});

dialogPanel.add(createButton);

dialogFrame.add(dialogPanel);

dialogFrame.setVisible(true);

}

private String generateAccountNumber() {

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

Date now = new Date();

return "ACC" + sdf.format(now);

}

private void performTransaction(String accountNumber, String transactionType, double amount) {

for (Account account : accounts) {

if (account.getAccountNumber().equals(accountNumber)) {

if ("Deposit".equals(transactionType)) {

account.deposit(amount);

recordTransaction("Deposit: $" + amount + " to Account Number " + accountNumber);

}

return;

}

}

JOptionPane.showMessageDialog(null, "Account not found.");

}

private void performWithdrawal(String accountNumber, String enteredPin) {

for (Account account : accounts) {

if (account.getAccountNumber().equals(accountNumber) && account.checkPin(enteredPin)) {

try {

double amount = Double.parseDouble(withdrawField.getText());

if (account.withdraw(amount)) {

recordTransaction("Withdrawal: $" + amount + " from Account Number " + accountNumber);

} else {

JOptionPane.showMessageDialog(null, "Insufficient funds.");

}

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(null, "Invalid amount entered.");

}

return;

}

}

JOptionPane.showMessageDialog(null, "Account not found or incorrect PIN.");

}

private void displayBalance(String accountNumber) {

for (Account account : accounts) {

if (account.getAccountNumber().equals(accountNumber)) {

JOptionPane.showMessageDialog(null, "Account Balance: $" + account.getBalance());

return;

}

}

JOptionPane.showMessageDialog(null, "Account not found.");

}

private void displayMiniStatement(String accountNumber) {

for (Account account : accounts) {

if (account.getAccountNumber().equals(accountNumber)) {

StringBuilder statement = new StringBuilder("Mini Statement for Account Number " + accountNumber + ":\n");

Map<String, List<String>> transactionMap = account.getTransactionHistory();

for (Map.Entry<String, List<String>> entry : transactionMap.entrySet()) {

String date = entry.getKey();

List<String> transactions = entry.getValue();

statement.append("Date: ").append(date).append("\n");

for (String transaction : transactions) {

statement.append(transaction).append("\n");

}

}

JOptionPane.showMessageDialog(null, statement.toString());

return;

}

}

JOptionPane.showMessageDialog(null, "Account not found.");

}

private void recordTransaction(String transaction) {

transactionHistory.add(transaction);

updateTransactionHistoryArea();

}

private void updateTransactionHistoryArea() {

StringBuilder historyText = new StringBuilder("Transaction History:\n");

for (String transaction : transactionHistory) {

historyText.append(transaction).append("\n");

}

transactionHistoryArea.setText(historyText.toString());

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new BankManagementSystemGUI());

}

private class Account {

private String accountNumber;

private String name;

private String dob;

private String phone;

private String pin;

private double balance;

private Map<String, List<String>> transactionHistory = new HashMap<>();

public Account(String accountNumber, String name, String dob, String phone, String pin) {

this.accountNumber = accountNumber;

this.name = name;

this.dob = dob;

this.phone = phone;

this.pin = pin;

this.balance = 0.0;

}

public String getAccountNumber() {

return accountNumber;

}

public double getBalance() {

return balance;

}

public boolean checkPin(String enteredPin) {

return pin.equals(enteredPin);

}

public Map<String, List<String>> getTransactionHistory() {

return transactionHistory;

}

public void deposit(double amount) {

balance += amount;

recordTransaction("Deposit: $" + amount);

}

public boolean withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

recordTransaction("Withdrawal: $" + amount);

return true;

}

return false;

}

public void recordTransaction(String transaction) {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

String date = sdf.format(new Date());

List<String> transactionsForDate = transactionHistory.get(date);

if (transactionsForDate == null) {

transactionsForDate = new ArrayList<>();

transactionHistory.put(date, transactionsForDate);

}

transactionsForDate.add(transaction);

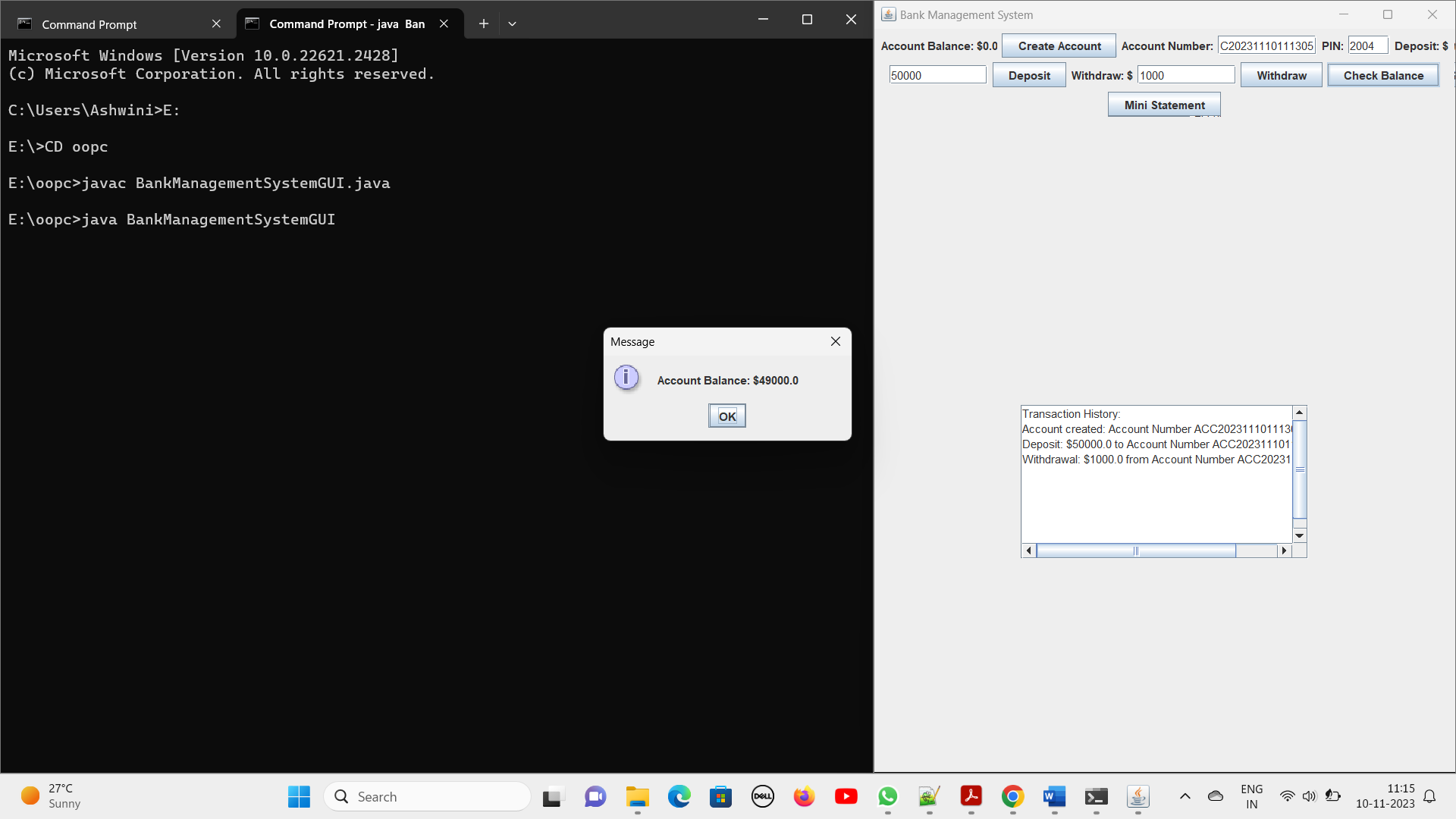
}

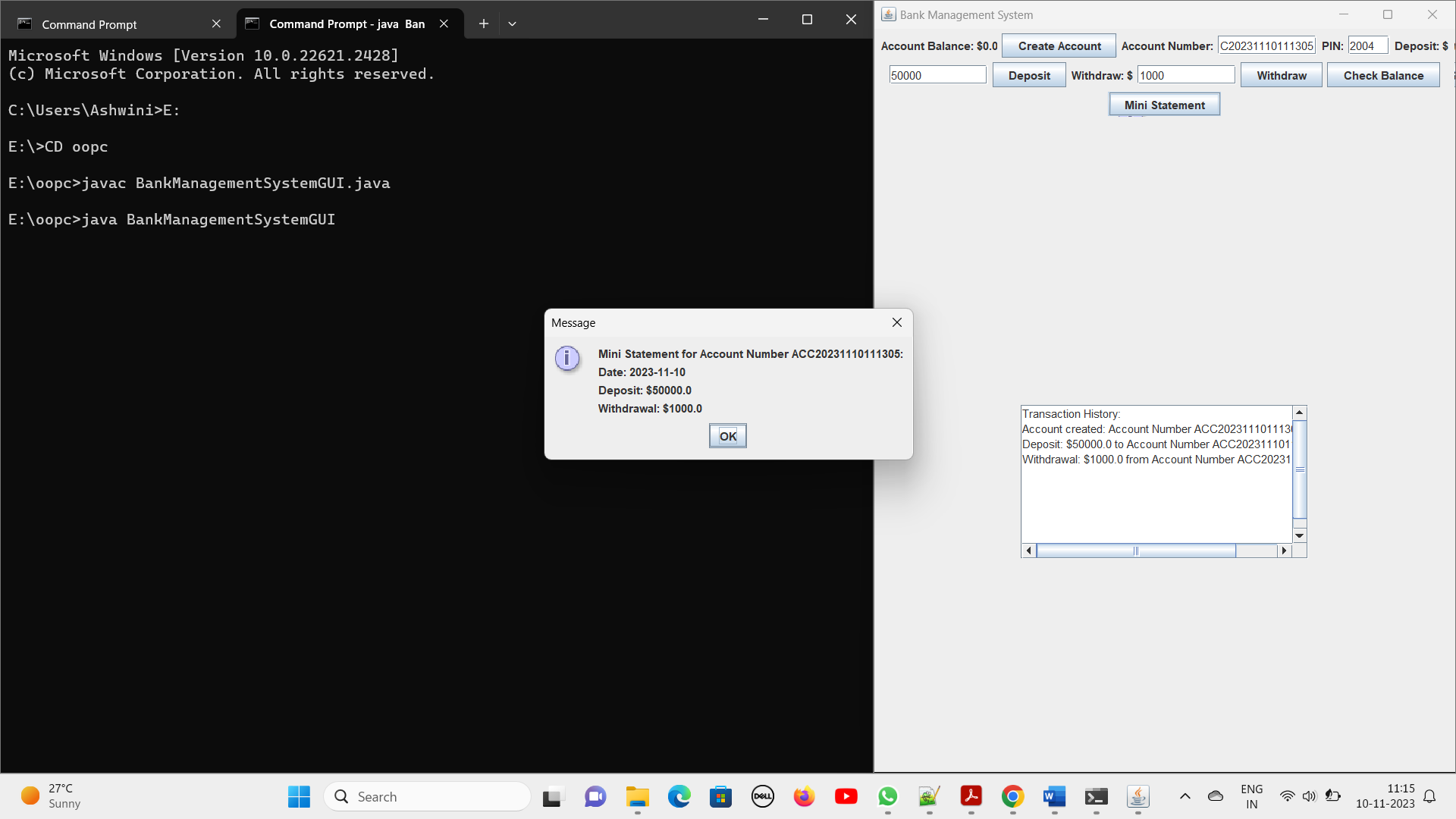
}

}

16

# RESULT SCREEN

****



17

**CONCLUSION**

We finally conclude that using this project we can provide a great interface between the user and the banking environment, thus satisfying the requirements of multiple users. It provides an efficient ways for people to involve in on-line transactions. We are providing a monitoring mechanism for admin which is having the ultimate power .Finally the users will be satisfied with our service.

18