# Rangamati Science and Technology University

Department: Computer Science And Engineering

1st Year 2nd Semester

Subject: Object Oriented Programming Lab

Submitted By:

Arindam Datta-2201011015

Iftakher Ibne Ilias-2201011024

Abdur Rahman-2201011033

Mohammed Imtiaz Aziz-2201011023

Submitted to:

Dr. Muhammed Jamshed Alam Patwary

Postdoctoral Fellow, Nottingham Trent University, England

Research Assistant Professor, IICT

Chittagong University of Engineering & Technology

# Project Name: Banking Management System

The **Bank Management System** is a desktop application built using Java Swing that allows a bank administrator to create, modify, delete, and manage customer bank accounts. The system provides a graphical interface for interaction, along with data persistence through file-based serialization.

To develop a GUI-based banking application that:

- Allows basic account operations (Create, Deposit, Withdraw, Transfer)
- Saves data persistently
- Provides an admin panel with real-time data display
- Is packaged as a standalone Windows `.exe` for ease of use

## apparatus:

- **Programming Language:** Java
- **GUI Framework:** Java Swing
- **Data Storage:** File-based serialization (`accounts.dat`)
- **Executable Wrapper:** Launch4j (for converting `.jar` to `.exe`)
- **IDE:** NetBeans (recommended)

## System Requirements:

- **Operating System:** Windows / Linux
- **Java Version:** JDK 8 or higher
- **Memory:** Minimum 2 GB RAM
- **Disk:** 10 MB for JRE + Application

- **Bundled JRE (optional):** For .exe without needing system Java

## Features:

- Create new bank accounts with unique IDs
- Deposit and withdraw money
- Transfer between accounts
- Search by name or ID
- View all accounts
- Track total number of transactions
- Delete accounts
- User-friendly Admin Panel GUI
- Data saved locally in serialized form

## System Design:

- GUI components include:
  - Title bar with icon
  - Buttons panel (GridLayout)
  - Output panel (JTextPane)
- All user interactions are handled via `JOptionPane` dialogs
- Backend account logic encapsulated in a `BankAccount` class
- Account data saved in `accounts.dat`

## Class Structure:

**1. `BankAccount` class**

- Fields: `name`, `address`, `type`, `uid`, `balance`
- Static: `no_of_trans` (tracks transactions)
- Implements `Serializable`

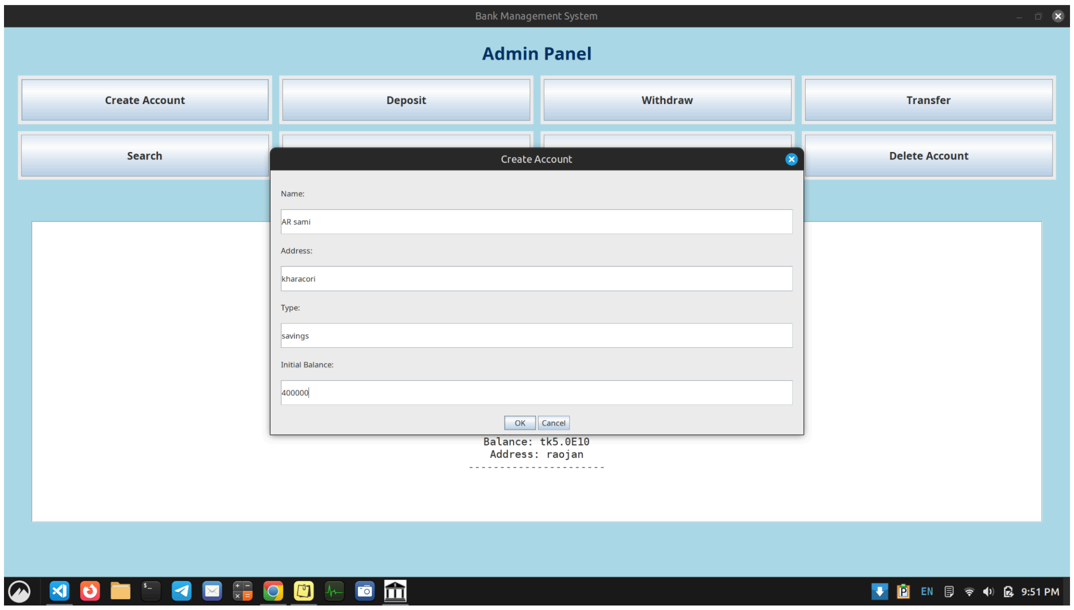
**2. `BankGUI` class**

- Extends `JFrame`
- Contains:
  - GUI setup in constructor
  - Button action listeners
  - Account operations (create, deposit, etc.)
  - Methods to load/save accounts using object streams

## Functionality Description:

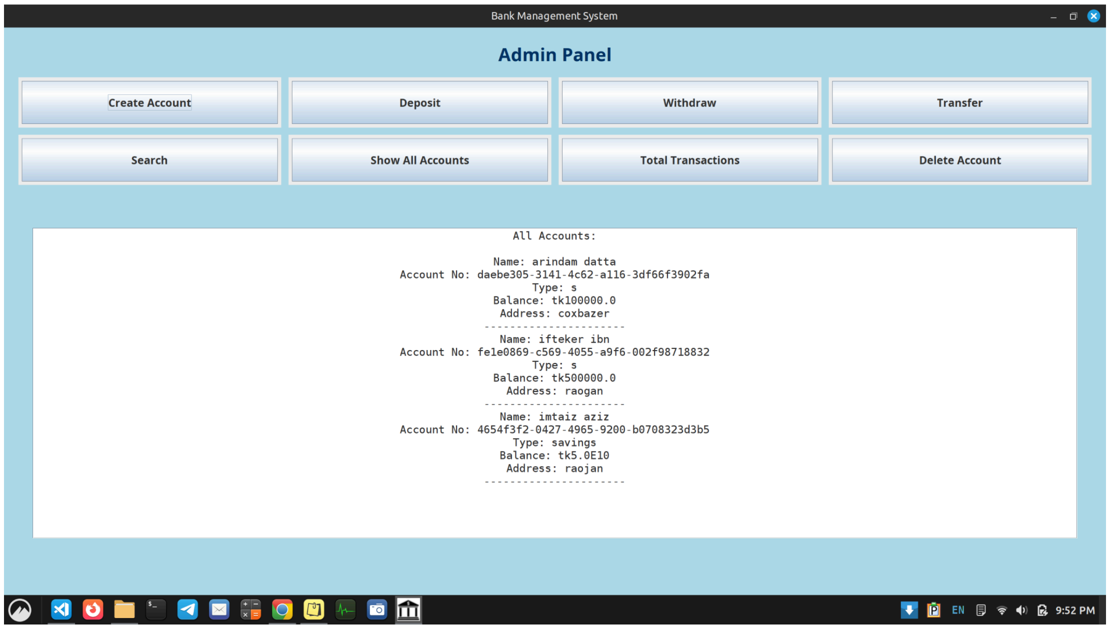
### Account Creation:

Screenshot:



Name: AR sami
Address: kharacori
Type: savings
Initial Balance: 400000

```
Balance: tk5.0E10
Address: raojan
--------------------
```

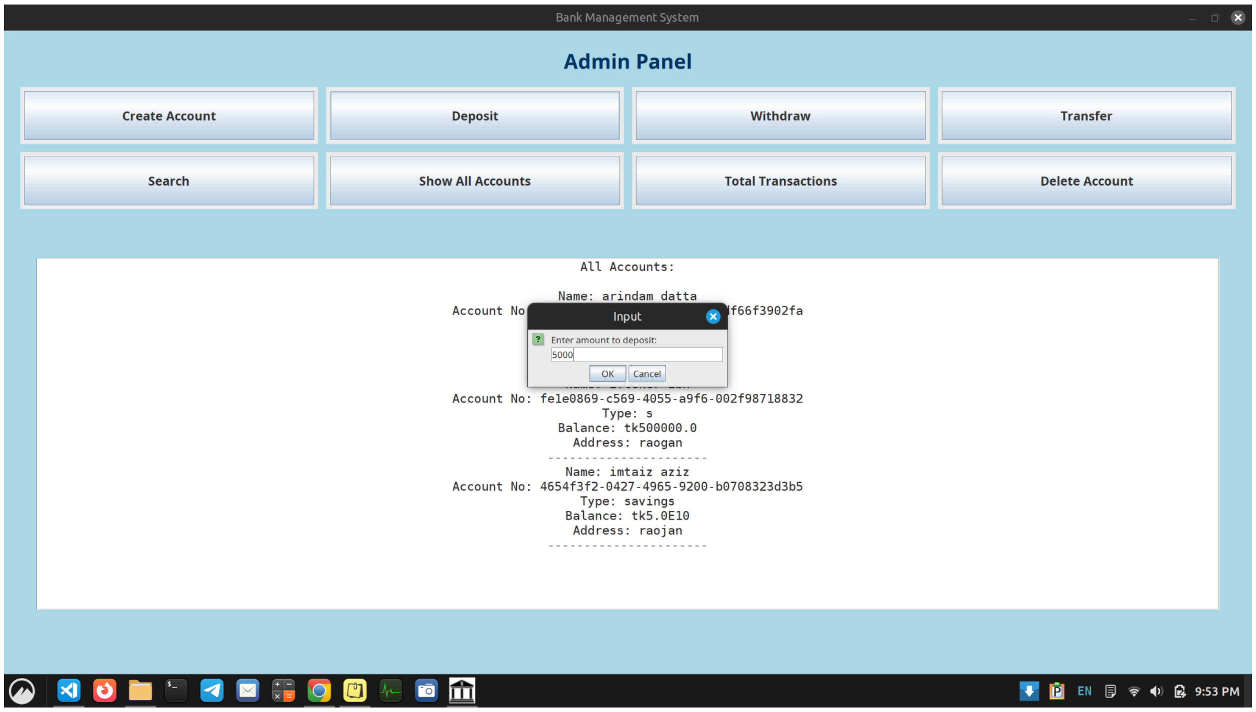


```
                      All Accounts:

                  Name: arindam datta
Account No: daebe305-3141-4c62-a116-3df66f3902fa
                        Type: s
                  Balance: tk100000.0
                   Address: coxbazer
                 ----------------------
                   Name: ifteker ibn
Account No: fe1e0869-c569-4055-a9f6-002f98718832
                        Type: s
                  Balance: tk500000.0
                    Address: raogan
                 ----------------------
                   Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
                     Type: savings
                   Balance: tk5.0E10
                    Address: raojan
                 ----------------------
```

- Prompts for name, address, type, and balance
- Auto-generates UID
- Displays confirmation

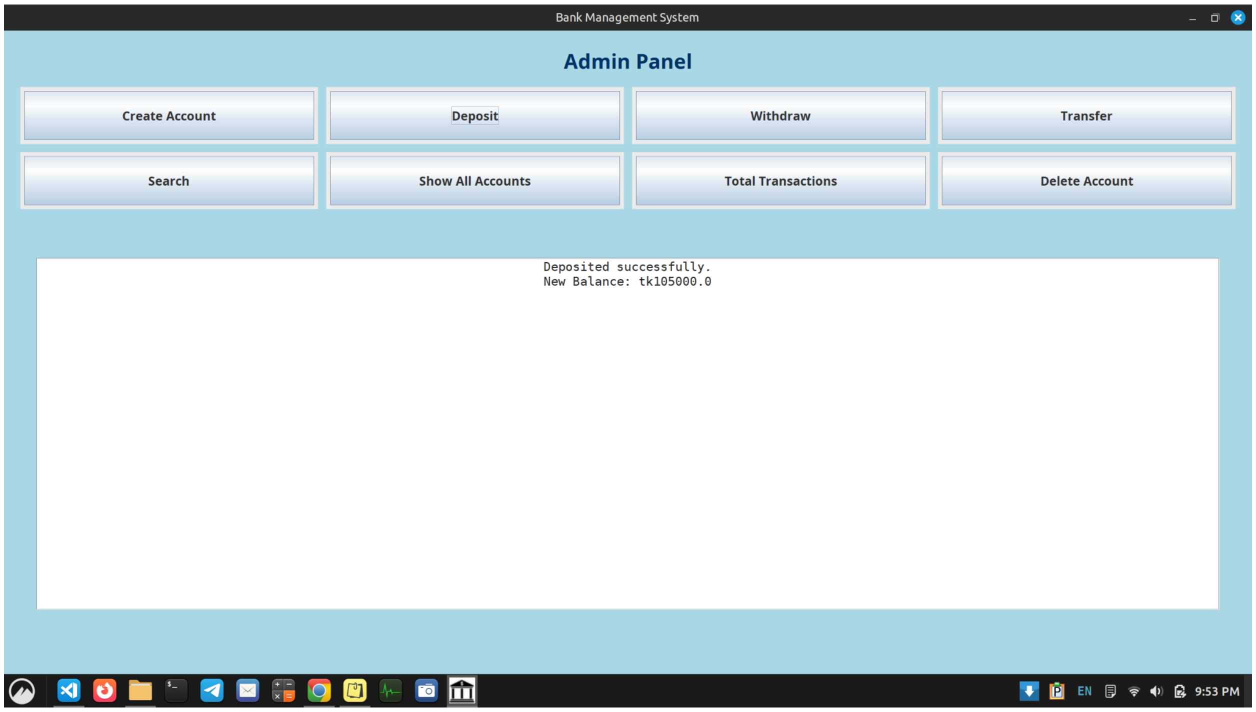
## Deposit:

### Screenshot:



```
All Accounts:

Name: arindam datta
Account No                              f66f3902fa

Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk500000.0
Address: raogan
----------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
----------------------
```

```
5000
```

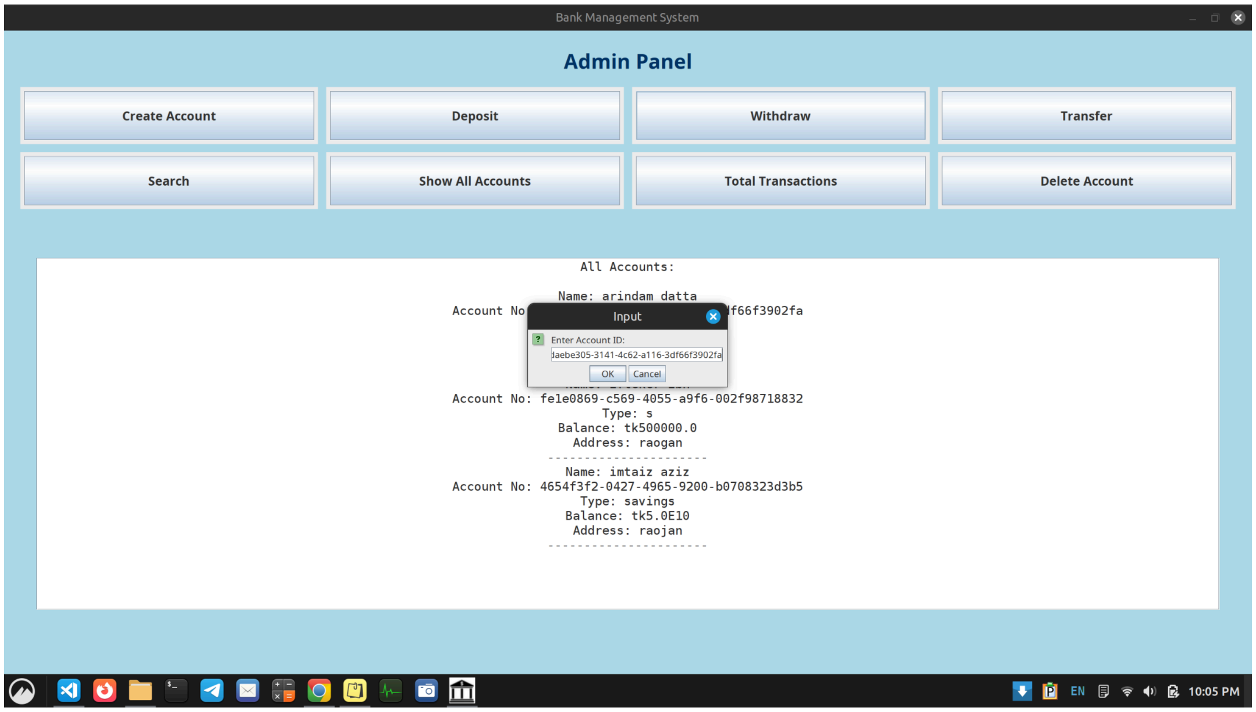


```
Deposited successfully.
New Balance: tk105000.0
```

- Input account ID and amount
- Updates balance and transaction count

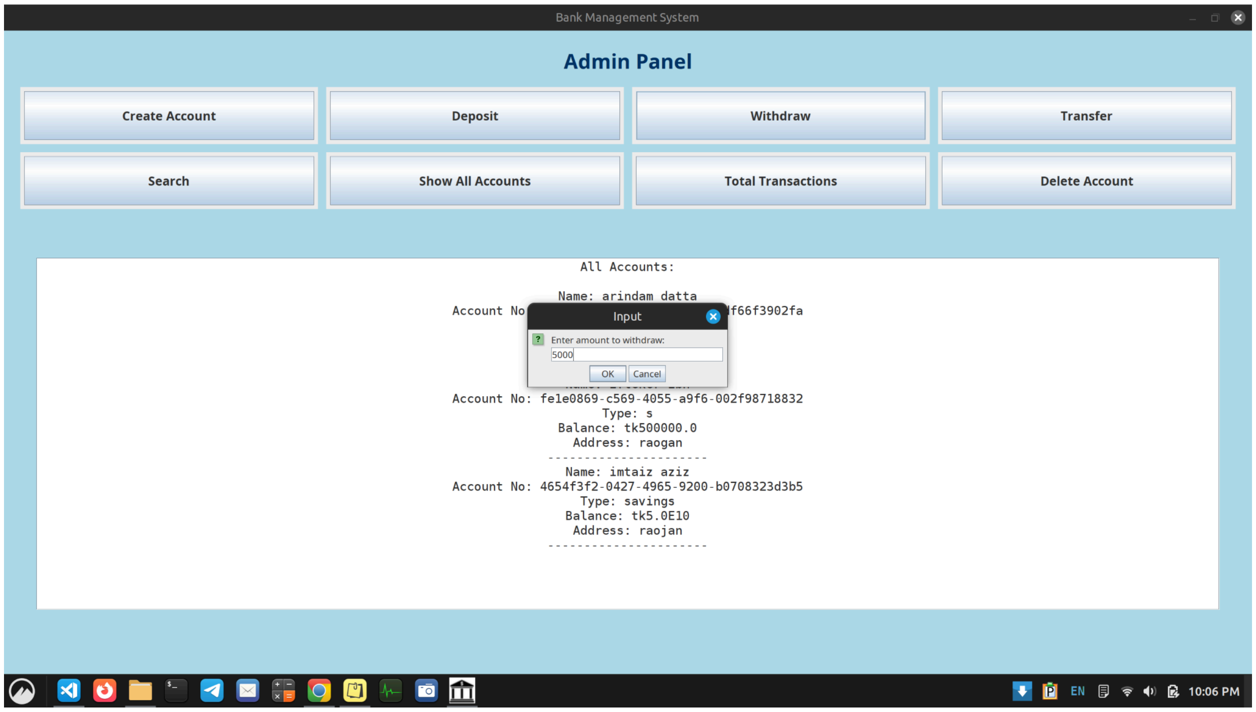
## Withdraw:

## Screenshot:



```
All Accounts:

Name: arindam datta
Account No                          lf66f3902fa

Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk500000.0
Address: raogan
---------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
---------------------
```

Enter Account ID: daebe305-3141-4c62-a116-3df66f3902fa



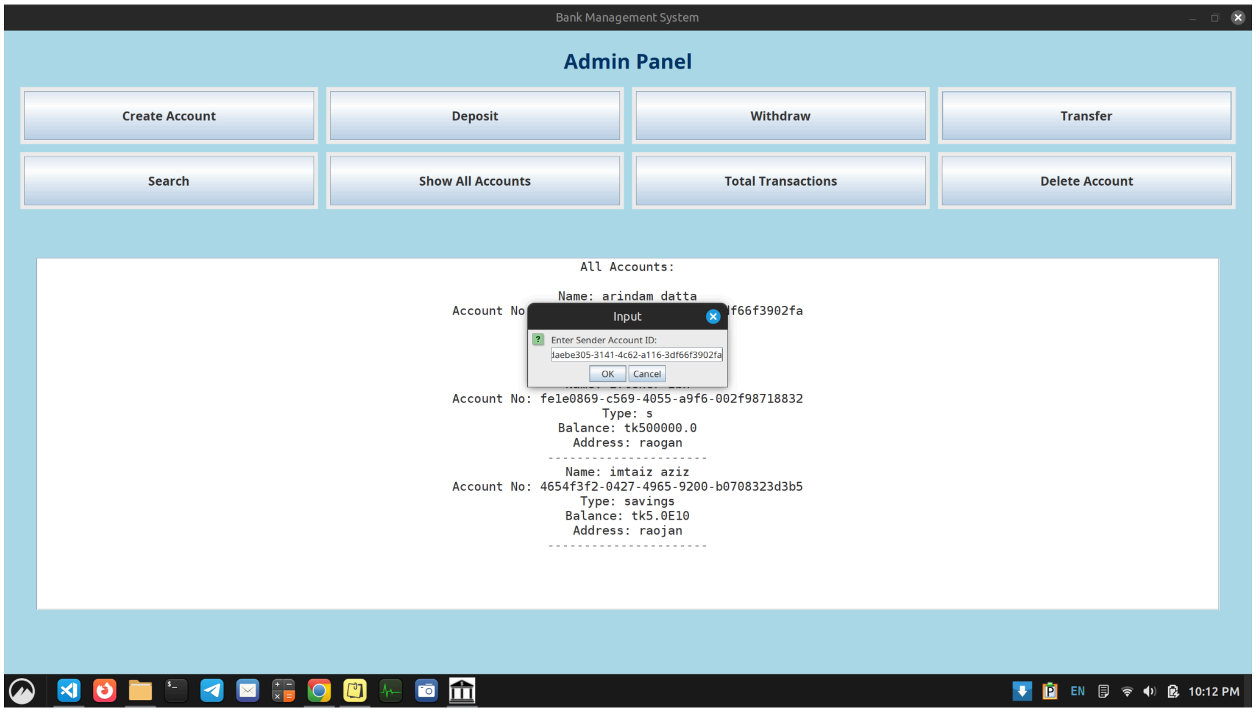
```
All Accounts:

Name: arindam datta
Account No                          lf66f3902fa

Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk500000.0
Address: raogan
---------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
---------------------
```

Enter amount to withdraw: 5000

- Verifies balance before deduction

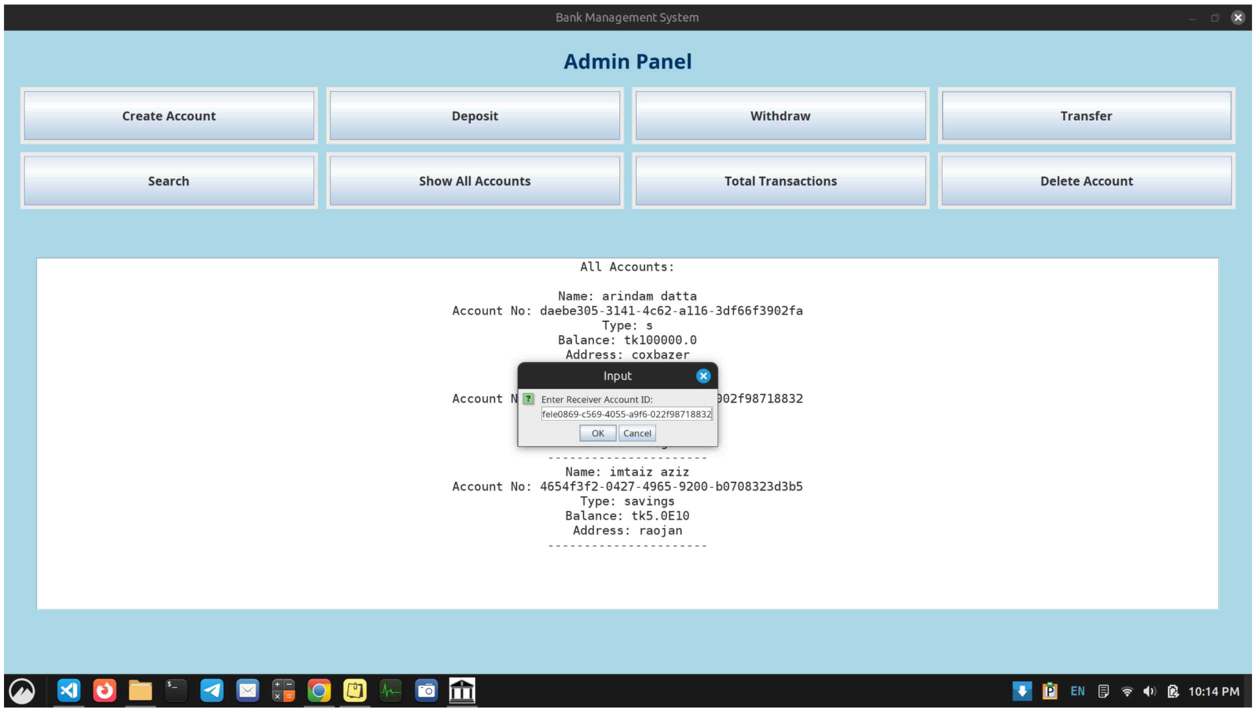
## **Transfer:**

## **Screenshot:**



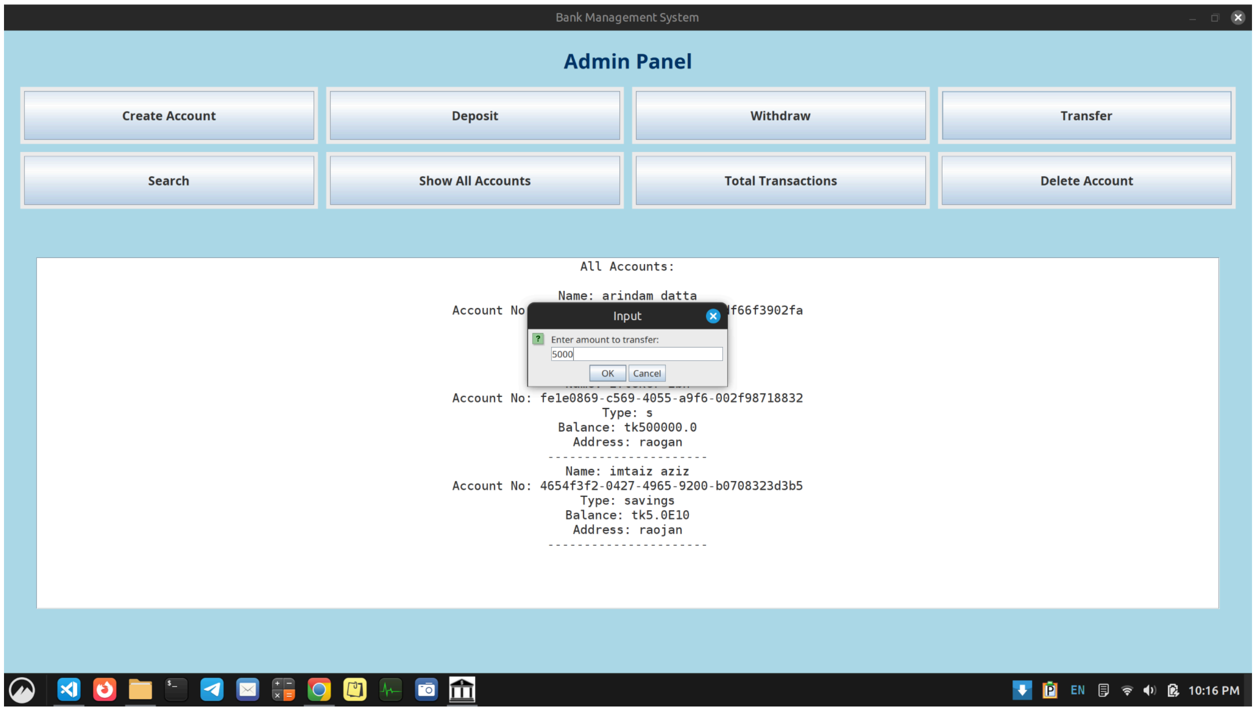
```
All Accounts:

Name: arindam datta
Account No                    f66f3902fa

Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk500000.0
Address: raogan
---------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
---------------------
```

Enter Sender Account ID: daebe305-3141-4c62-a116-3df66f3902fa



```
All Accounts:

Name: arindam datta
Account No: daebe305-3141-4c62-a116-3df66f3902fa
Type: s
Balance: tk100000.0
Address: coxbazer

Account N                    002f98718832

---------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
---------------------
```

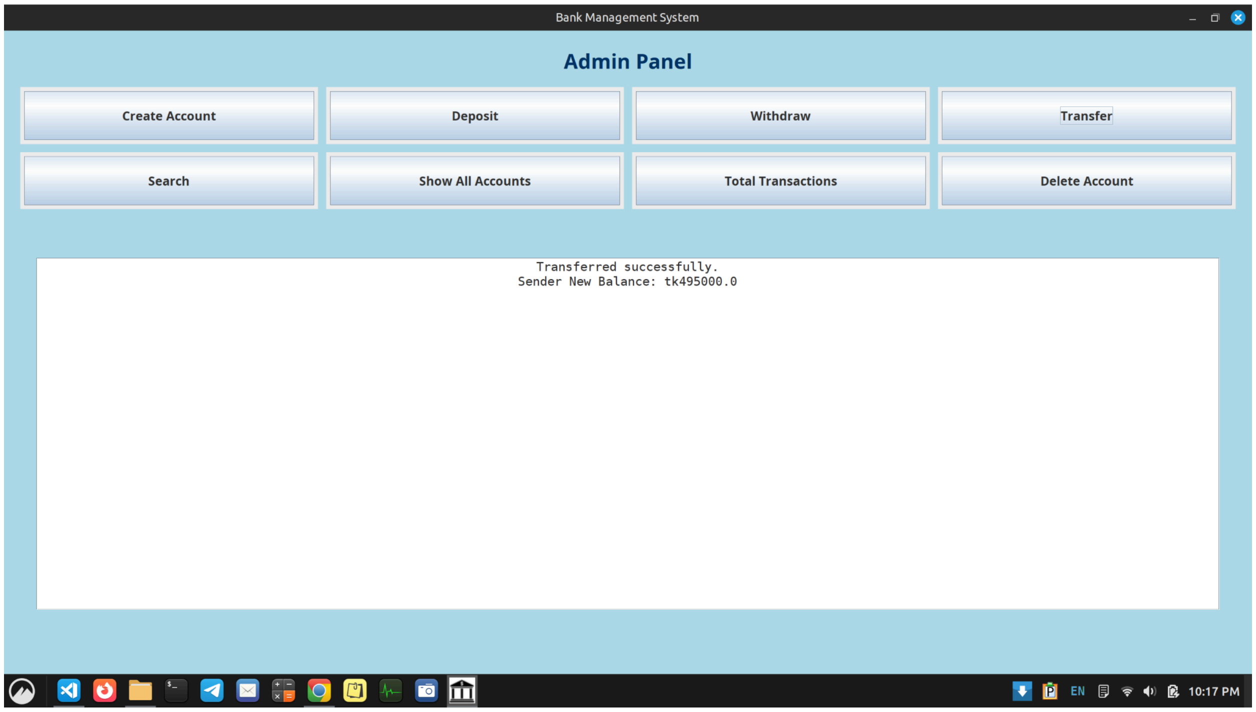
Enter Receiver Account ID: fele0869-c569-4055-a9f6-022f98718832



```
All Accounts:

Name: arindam datta
Account No                              df66f3902fa
```

Enter amount to transfer: 5000

```
Account No: fe1e0869-c569-4055-a9f6-002f98718832
                    Type: s
              Balance: tk500000.0
               Address: raogan
            ----------------------
              Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
                 Type: savings
               Balance: tk5.0E10
               Address: raojan
            ----------------------
```

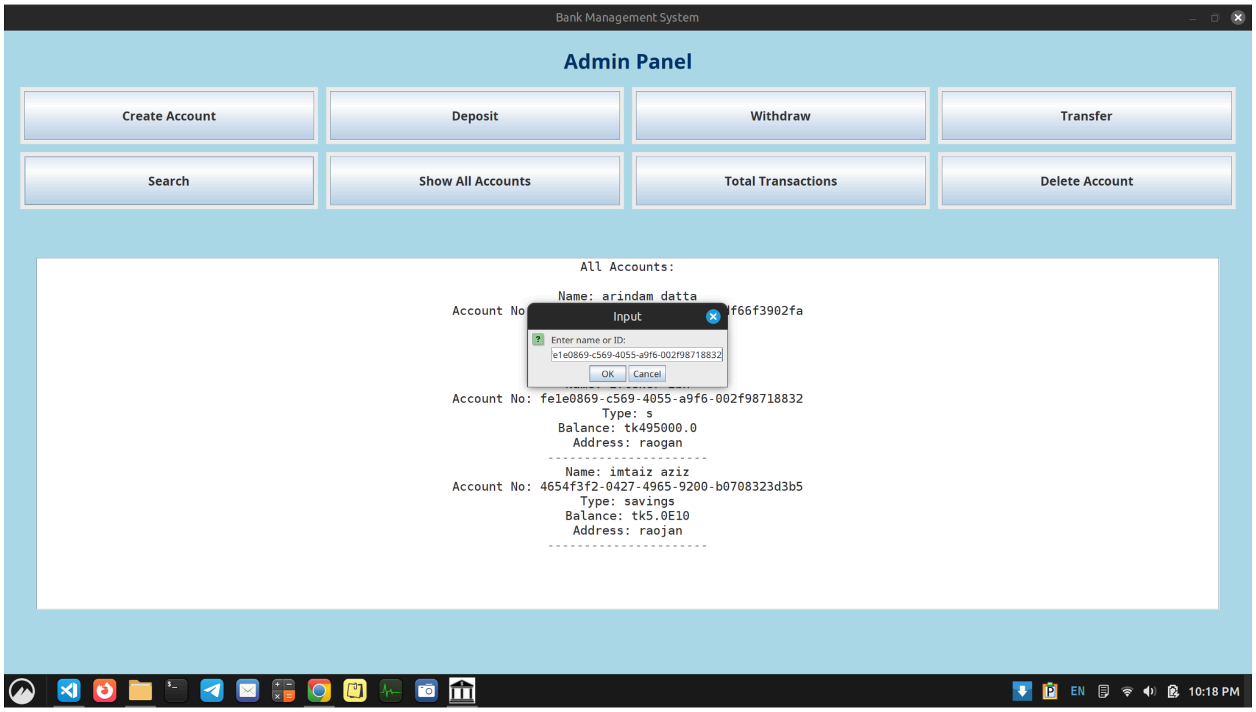


```
Transferred successfully.
Sender New Balance: tk495000.0
```

- Input sender and receiver IDs
- Validates balance before transfer

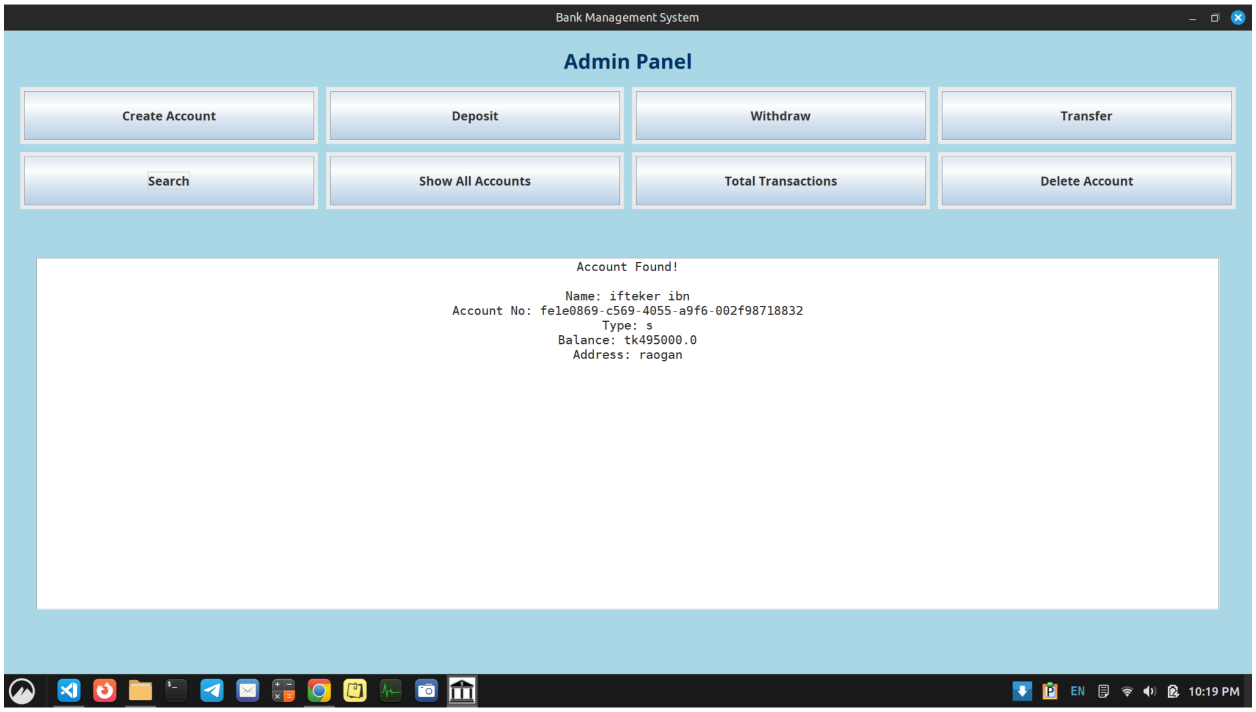
## Search;

## Screenshot:



```
All Accounts:

Name: arindam datta
Account No                          f66f3902fa

Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk495000.0
Address: raogan
----------------------
Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
Type: savings
Balance: tk5.0E10
Address: raojan
----------------------
```

Enter name or ID: e1e0869-c569-4055-a9f6-002f98718832

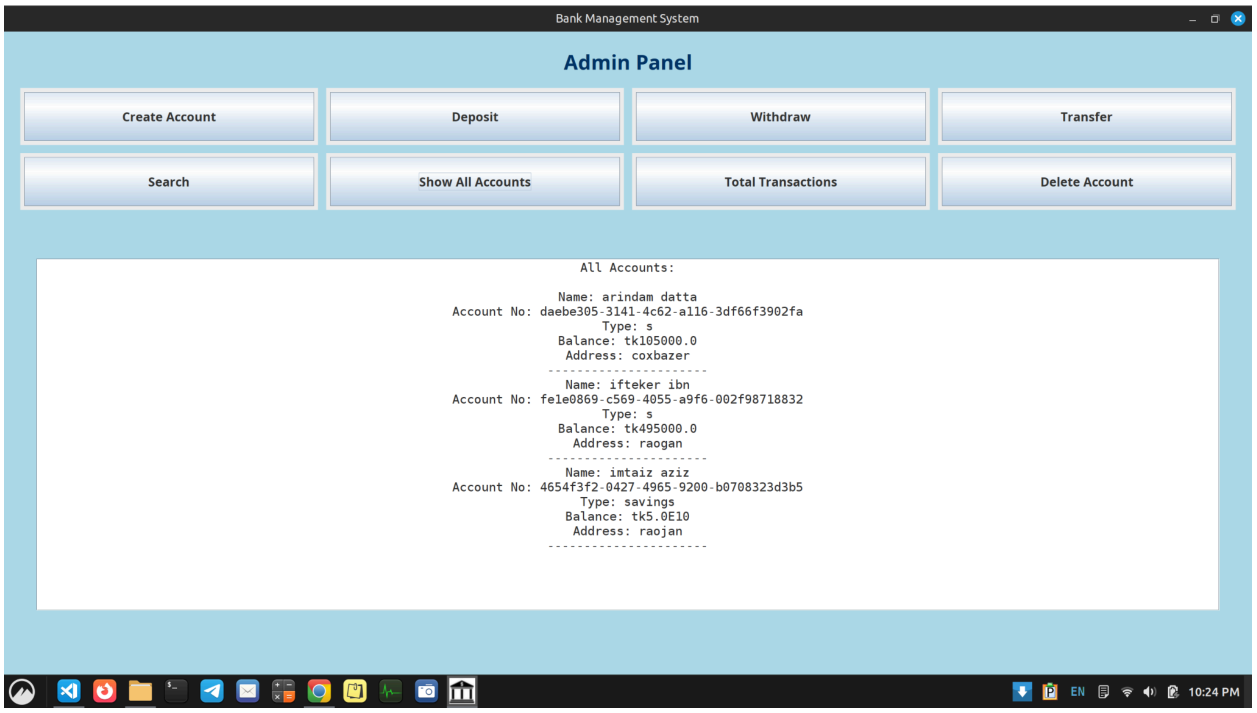


```
Account Found!

Name: ifteker ibn
Account No: fe1e0869-c569-4055-a9f6-002f98718832
Type: s
Balance: tk495000.0
Address: raogan
```

- Search by name or account ID

## Show All:

## Screenshot:

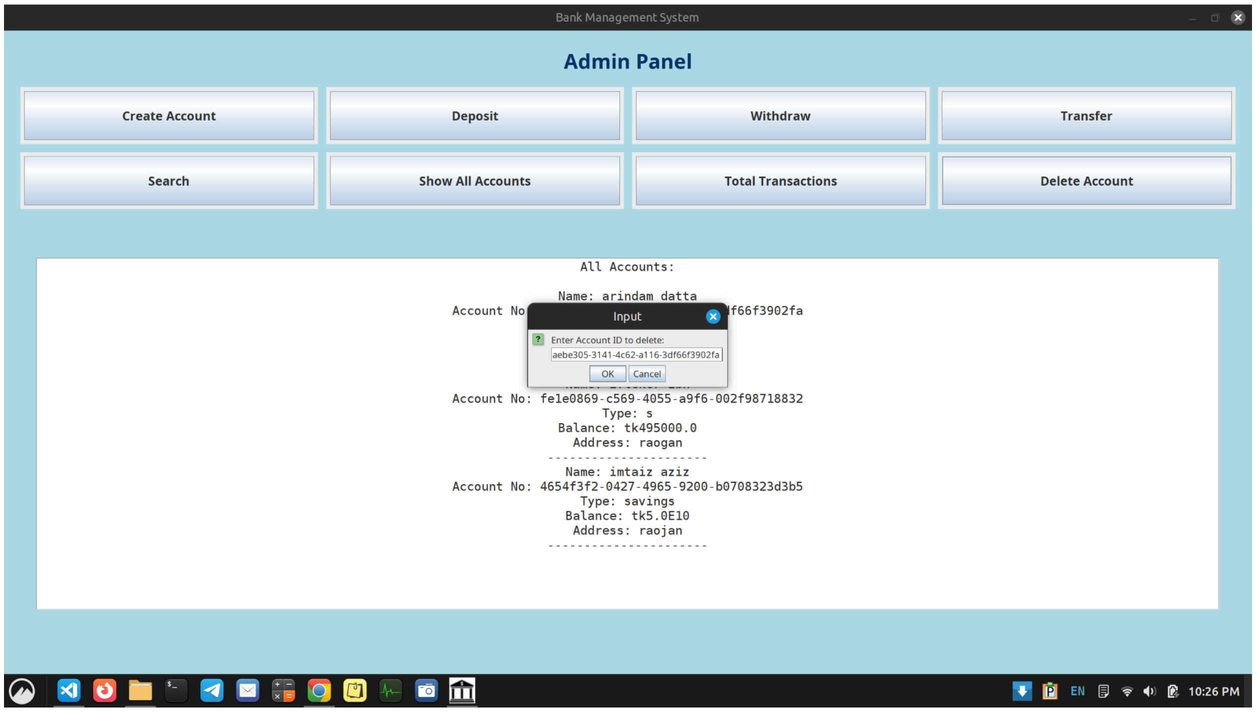


```
                    All Accounts:

                 Name: arindam datta
Account No: daebe305-3141-4c62-a116-3df66f3902fa
                       Type: s
                 Balance: tk105000.0
                  Address: coxbazer
                ---------------------
                  Name: ifteker ibn
Account No: fe1e0869-c569-4055-a9f6-002f98718832
                       Type: s
                 Balance: tk495000.0
                   Address: raogan
                ---------------------
                  Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
                    Type: savings
                  Balance: tk5.0E10
                   Address: raojan
                ---------------------
```

- Lists all accounts with details

## delete Account:

## Screenshot:



Bank Management System

**Admin Panel**

Create Account | Deposit | Withdraw | Transfer

Search | Show All Accounts | Total Transactions | Delete Account

```
                All Accounts:

              Name: arindam datta
Account No                        f66f3902fa
```

Input

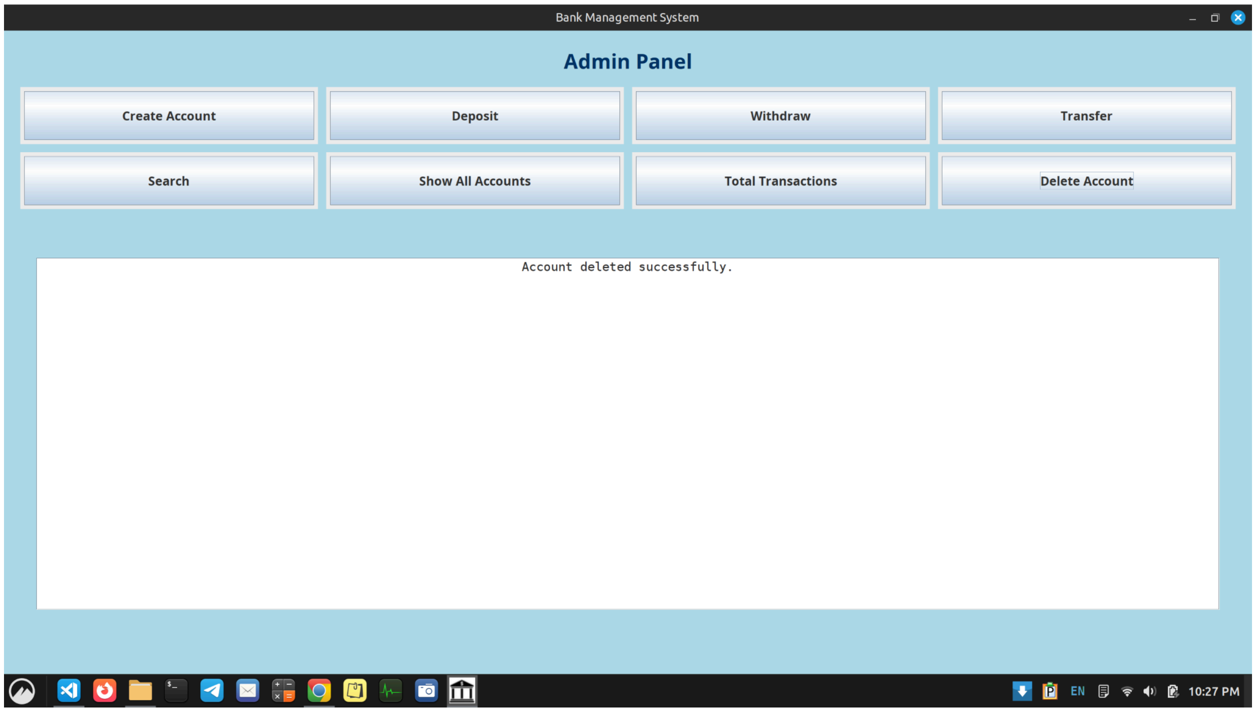
Enter Account ID to delete:

aebe305-3141-4c62-a116-3df66f3902fa

OK | Cancel

```
Account No: fe1e0869-c569-4055-a9f6-002f98718832
                    Type: s
              Balance: tk495000.0
                Address: raogan
             ----------------------
               Name: imtaiz aziz
Account No: 4654f3f2-0427-4965-9200-b0708323d3b5
                 Type: savings
               Balance: tk5.0E10
                Address: raojan
             ----------------------
```

EN 10:26 PM



Bank Management System

**Admin Panel**

Create Account | Deposit | Withdraw | Transfer

Search | Show All Accounts | Total Transactions | Delete Account

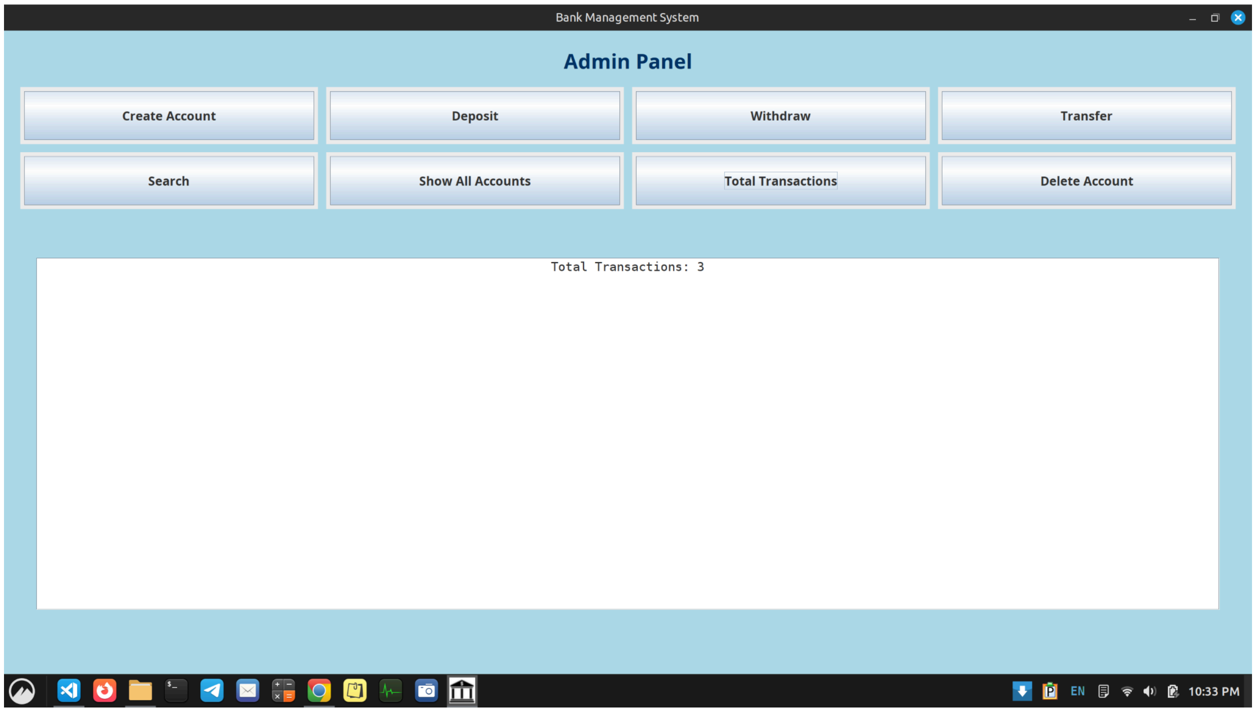
```
Account deleted successfully.
```

EN 10:27 PM

- Prompts for confirmation before deletion

## Total transaction:

## Screenshot:



```
Total Transactions: 3
```

## Data Persistence:

- File: `accounts.dat`
- Format: Java object serialization (`ObjectOutputStream`)
- Data automatically loaded on application startup
- Safe saving on each operation

## Executable Conversion (JAR to EXE):

### Step 1: Export JAR

- Build your project in NetBeans
- Ensure `BankGUI` is the main class
- Include resources (e.g. `icon.png`) in JAR

### Step 2: Convert JAR to EXE using Launch4j

- Open Launch4j
- Set:
  - **Output file:** `BankApp.exe`
  - **Jar:** `BankGUI.jar`
  - **Bundled JRE path:** (optional, e.g. `jre`)
  - **Classpath:** leave as is if JAR is standalone
- Press **Build Wrapper**

### Step 3: Run

- Double-click `BankApp.exe`
- App launches with GUI and functionality

---

## How to Run the Project:

**With JAR**

bash
Copy code
```
java -jar BankGUI.jar
```

**With EXE**

- Double-click `BankApp.exe`
- Ensure `accounts.dat` and `icon.png` are in the same folder (if not inside JAR)

## Source code:

### BankAccount Class:

```
import java.io.Serializable;
import java.util.*;
public class BankAccount implements Serializable {
   //Creating global Scanner object
   public static Scanner scr = new Scanner(System.in);

   @Override
   public String toString() {
       return "Name: " + name +
           "\nAccount No: " + uid +
           "\nType: " + type +
           "\nBalance: tk" + balance +
           "\nAddress: " + address;
   }

    public String uid;
    String name;
   String address;
   String type;
   double balance;
    static int no_of_trans = 0;

   //Default Constructor
   BankAccount(){
       uid=null;
       name=address=type=null;
   }

   BankAccount(String name,String address,String type,double balance){
       this.uid = UUID.randomUUID().toString();
       this.name = name;
       this.address = address;
       this.type = type;
       this.balance = balance;
   }


   void transfer(BankAccount receiver) {
       System.out.print("Enter amount to transfer : tk");
       double amt = Double.parseDouble(scr.nextLine());
```

```java
        if (amt <= this.balance) {
            this.balance -= amt;
            receiver.balance += amt;
            System.out.println("Transfer Successful......");
            no_of_trans++;
        } else {
            System.out.println("Insufficient Balance!!!!!!");
        }
    }

    //Deposit amount in account
    void deposit(){
        System.out.println("");
        System.out.println("Account No : "+this.uid);
        System.out.println("Name : "+this.name);
        System.out.print("Enter amount to be Deposited : tk");
        double blnc = Double.parseDouble((scr.nextLine()));
        this.balance = this.balance + blnc;
        System.out.println("");
        System.out.println("Amount Credited Successfully...");
        System.out.println("");
        no_of_trans++;
    }

    //Withdraw from account
    void withdraw(){
        System.out.println("");
        System.out.println("Account No : "+this.uid);
        System.out.println("Name : "+this.name);
        System.out.print("Enter amount to be Withdrawn : tk");
        double blnc = Double.parseDouble((scr.nextLine()));
        this.balance = this.balance - blnc;
        System.out.println("");
        System.out.println("Amount Debited Successfully...");
        System.out.println("");
        no_of_trans++;
    }

    //Change address of depositer
    void changeAddress(){
        System.out.println("");
        System.out.println("Account No : "+this.uid);
        System.out.println("Name : "+this.name);
        System.out.print("Enter New Address : ");
        this.address = scr.nextLine();
        System.out.println("");
        System.out.println("Address Successfully Changed...");
        System.out.println("");
    }

    void checkBalance() {
        System.out.println("Account No : " + this.uid);
        System.out.println("Available Balance : tk" + this.balance);
    }
}
```

**BankGUI Class:**

```
import javax.swing.*;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;
import java.awt.*;
import java.util.ArrayList;
import java.io.*;
public class BankGUI extends JFrame {

private void saveAccounts() {
    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("accounts.dat"))) {
        oos.writeObject(accounts);
        System.out.println("Accounts saved successfully.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void loadAccounts() {
    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("accounts.dat"))) {
        accounts = (ArrayList<BankAccount>) ois.readObject();
        System.out.println("Accounts loaded successfully.");
```

```java
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("No previous data found or failed to load.");
    }
}

    private void setOutput(String text) {
    output.setText(text);
    StyledDocument doc = output.getStyledDocument();
    SimpleAttributeSet center = new SimpleAttributeSet();
    StyleConstants.setAlignment(center, StyleConstants.ALIGN_CENTER);
    doc.setParagraphAttributes(0, doc.getLength(), center, false);
    }


    private static void setUIFont(Font font) {
    java.util.Enumeration<Object> keys = UIManager.getDefaults().keys();
    while (keys.hasMoreElements()) {
        Object key = keys.nextElement();
        Object value = UIManager.get(key);
        if (value instanceof Font) {
            UIManager.put(key, font);
            }
        }
    }

    private ArrayList<BankAccount> accounts = new ArrayList<>();
    private JTextPane output = new JTextPane();

    public BankGUI() {
        Color lightBlue = new Color(173, 216, 230); // Light blue color
        getContentPane().setBackground(lightBlue);

        setTitle("Bank Management System");
        loadAccounts();
        ImageIcon icon = new ImageIcon(getClass().getResource("icon.png"));
        setIconImage(icon.getImage());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH); // Fullscreen window
        setLocationRelativeTo(null);
        output.setEditable(false);
        output.setFont(new Font("Monospaced", Font.PLAIN, 30)); // Larger
font
        StyledDocument doc = output.getStyledDocument();
        SimpleAttributeSet center = new SimpleAttributeSet();
        StyleConstants.setAlignment(center, StyleConstants.ALIGN_CENTER);
        doc.setParagraphAttributes(0, doc.getLength(), center, false);

        JScrollPane scrollPane = new JScrollPane(output);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(2, 4, 20, 20)); // 2 rows, 4
columns

        String[] btnLabels = {
            "Create Account", "Deposit", "Withdraw",
            "Transfer", "Search", "Show All Accounts", "Total
Transactions", "Delete Account"
        };

        JButton[] buttons = new JButton[btnLabels.length];

        for (int i = 0; i < btnLabels.length; i++) {
            buttons[i] = new JButton(btnLabels[i]);
            buttons[i].setFont(new Font("Arial", Font.BOLD, 30));
            buttons[i].setPreferredSize(new Dimension(120, 120)); // Square
shape

             // Create a wrapper panel with padding
            JPanel wrapper = new JPanel(new BorderLayout());
            wrapper.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10)); // top, left, bottom, right
            wrapper.add(buttons[i], BorderLayout.CENTER);

            buttonPanel.add(wrapper);
        }

        buttons[0].addActionListener(e -> createAccount());
        buttons[1].addActionListener(e -> deposit());
        buttons[2].addActionListener(e -> withdraw());
```

```
       buttons[3].addActionListener(e -> transfer());
       buttons[4].addActionListener(e -> search());
       buttons[5].addActionListener(e -> showAll());
       buttons[6].addActionListener(e -> output.setText("Total
Transactions: " + BankAccount.no_of_trans));    JPanel topPanel = new
JPanel(new BorderLayout());
       buttons[7].addActionListener(e -> deleteAccount());
       topPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,
20));

       // Create and configure title label
     // Create and configure title label
       JLabel titleLabel = new JLabel("Admin Panel");
       titleLabel.setFont(new Font("Arial", Font.BOLD, 50));
       titleLabel.setForeground(new Color(0, 51, 102));
       titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT); // Center in
BoxLayout

       // Create a vertical box layout panel for title and buttons
       JPanel verticalPanel = new JPanel();
       verticalPanel.setLayout(new BoxLayout(verticalPanel,
BoxLayout.Y_AXIS));
       verticalPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20,
20));

       verticalPanel.add(titleLabel);
       verticalPanel.add(Box.createVerticalStrut(30)); // space between
title and buttons
       verticalPanel.add(buttonPanel);

       topPanel.add(verticalPanel, BorderLayout.CENTER);

       add(topPanel, BorderLayout.NORTH);
       JPanel outputPanel = new JPanel(new BorderLayout());
       outputPanel.setBorder(BorderFactory.createEmptyBorder(80,80, 160,
80)); // adds padding (top, left, bottom, right)
       outputPanel.add(scrollPane, BorderLayout.CENTER);
       outputPanel.setBackground(lightBlue); // match background

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

```
        topPanel.setBackground(lightBlue);
        verticalPanel.setBackground(lightBlue);
        buttonPanel.setBackground(lightBlue);
        output.setBackground(Color.WHITE);

        setVisible(true);
    }

    // All method definitions below remain unchanged...

    private void createAccount() {
    // Create text fields
    JTextField name = new JTextField();
    name.setFont(new Font("Arial", Font.PLAIN, 22));
    JTextField address = new JTextField();
    address.setFont(new Font("Arial", Font.PLAIN, 22));
    JTextField type = new JTextField();
    type.setFont(new Font("Arial", Font.PLAIN, 22));
    JTextField balance = new JTextField();
    balance.setFont(new Font("Arial", Font.PLAIN, 22));

    // Create a panel with vertical layout and spacing
    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(8, 1, 0, 10)); // 8 rows, 10px vertical
gap
    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20)); //
outer padding

    // Add label and field pairs
    panel.add(new JLabel("Name:"));
    panel.add(name);
    panel.add(new JLabel("Address:"));
    panel.add(address);
    panel.add(new JLabel("Type:"));
    panel.add(type);
    panel.add(new JLabel("Initial Balance:"));
    panel.add(balance);

    // Create JOptionPane and dialog
```

```
   JOptionPane optionPane = new JOptionPane(panel,
JOptionPane.PLAIN_MESSAGE, JOptionPane.OK_CANCEL_OPTION);
   JDialog dialog = optionPane.createDialog(this, "Create Account");

   // Set dialog size to half the window
   int width = getWidth() / 2;
   int height = getHeight() / 2;
   dialog.setSize(width, height);
   dialog.setLocationRelativeTo(this);
   dialog.setVisible(true);

   // Process input if OK clicked
   Object selectedValue = optionPane.getValue();
   if (selectedValue != null &&
selectedValue.equals(JOptionPane.OK_OPTION)) {
       try {
           BankAccount acc = new BankAccount(
               name.getText(),
               address.getText(),
               type.getText(),
               Double.parseDouble(balance.getText())
           );
           accounts.add(acc);
           saveAccounts();
           setOutput("Account Created!\n\n" + acc.toString());
       } catch (Exception ex) {
           setOutput("Error creating account.");
       }
   }
}
   private BankAccount selectAccount(String prompt) {
       String uid = JOptionPane.showInputDialog(this, prompt);
       for (BankAccount acc : accounts) {
           if (acc.uid.equalsIgnoreCase(uid)) return acc;
       }
       output.setText("Account not found.");
       return null;
   }
   private void deposit() {
       BankAccount acc = selectAccount("Enter Account ID:");
```

```
        if (acc != null) {
            String amt = JOptionPane.showInputDialog(this, "Enter amount to
deposit:");
            acc.balance += Double.parseDouble(amt);
            BankAccount.no_of_trans++;
            output.setText("Deposited successfully.\nNew Balance: tk" +
acc.balance);
        }
        saveAccounts();
    }
    private void withdraw() {
        BankAccount acc = selectAccount("Enter Account ID:");
        if (acc != null) {
            String amt = JOptionPane.showInputDialog(this, "Enter amount to
withdraw:");
            double amount = Double.parseDouble(amt);
            if (amount <= acc.balance) {
                acc.balance -= amount;
                BankAccount.no_of_trans++;
                output.setText("Withdrawn successfully.\nNew Balance: tk" +
acc.balance);
            } else {
                output.setText("Insufficient balance.");
            }
            saveAccounts();
        }
    }

    private void transfer() {
        BankAccount sender = selectAccount("Enter Sender Account ID:");
        if (sender != null) {
            BankAccount receiver = selectAccount("Enter Receiver Account
ID:");
            if (receiver != null) {
                String amt = JOptionPane.showInputDialog(this, "Enter
amount to transfer:");
                double amount = Double.parseDouble(amt);
                if (amount <= sender.balance) {
                    sender.balance -= amount;
                    receiver.balance += amount;
```

```
                   BankAccount.no_of_trans++;
                   output.setText("Transferred successfully.\nSender New
Balance: tk" + sender.balance);
               } else {
                   output.setText("Insufficient balance.");
               }
           }
       }
       saveAccounts();
   }

   private void search() {
       String query = JOptionPane.showInputDialog(this, "Enter name or
ID:");
       for (BankAccount acc : accounts) {
           if (acc.name.equalsIgnoreCase(query) ||
acc.uid.equalsIgnoreCase(query)) {
               output.setText("Account Found!\n\n" + acc.toString());
               return;
           }
       }
       output.setText("Account not found.");
   }

   private void showAll() {
       StringBuilder sb = new StringBuilder("All Accounts:\n\n");
       for (BankAccount acc : accounts) {
           sb.append(acc.toString()).append("\n-----------------------\n");
       }
       output.setText(sb.toString());
   }
   private void deleteAccount() {
   BankAccount acc = selectAccount("Enter Account ID to delete:");
   if (acc != null) {
       int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you
want to delete this account?", "Confirm Delete",
JOptionPane.YES_NO_OPTION);
       if (confirm == JOptionPane.YES_OPTION) {
           accounts.remove(acc);
           output.setText("Account deleted successfully.");
```

```java
        } else {
            output.setText("Account deletion cancelled.");
        }
    }
    saveAccounts();
}
@SuppressWarnings("unchecked")
private void loadAccountsFromFile() {
    File file = new File("accounts.dat");
    if (!file.exists()) {
        System.out.println("No previous data found.");
        return;
    }

    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(file))) {
        accounts = (ArrayList<BankAccount>) ois.readObject();
        System.out.println("Loaded " + accounts.size() + " accounts.");
    } catch (Exception e) {
        System.out.println("Error reading saved accounts: " +
e.getMessage());
    }
}

private void saveAccountsToFile() {
    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("accounts.dat"))) {
        oos.writeObject(accounts);
        System.out.println("Accounts saved.");
    } catch (IOException e) {
        System.out.println("Error saving accounts: " + e.getMessage());
    }
}

    public static void main(String[] args) {
    setUIFont(new Font("Arial", Font.PLAIN, 22)); // Set font size to 22
for all components
    new BankGUI();  // Now create the GUI
    }
}
```

## Conclusion:

The Bank Management System provides a simple, robust platform for managing customer bank data through a user-friendly GUI. Its portability via `.exe` conversion and persistent data storage makes it a practical mini-project for academic or personal learning purposes.