**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

**(NAAC Accredited with ‘A+’ Grade & NBA Accredited) (Approved by AICTE, Permanently Affiliated to JNTU Hyderabad)**

**KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501401**



|  |  |
| --- | --- |
| **STUDENT NAME** | **ROLL NUMBER** |
| K.SHASHANK | 22H51A7330 |
| K.MOUNIKA | 22H51A7331 |
| K.SHARATH NAIK | 22H51A7334 |

**Contents**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Particulars** | **Page No** |
| 1 | Introduction | 3 |
| 2 | Algorithm / Flowchart / Procedure | 4 |
| 3 | Requirements (Hardware and Software) | 5 |
| 4 | Implementation | 6 |
| 5 | Results | 8 |
| 6 | Conclusion & References | 9 |

# INTRODUCTION

The Online Mobile Recharge System is a convenient and user-friendly application designed to facilitate the process of recharging mobile phone accounts. This system eliminates the need for physical vouchers or visiting retail stores, allowing users to recharge their mobile numbers anytime and anywhere using an internet-connected device. The primary objective is to provide a seamless and efficient experience for users to maintain their mobile account balances.

In this application, key concepts from graphical user interface (GUI) programming are employed to create an interactive and intuitive user experience. Utilizing the Swing framework in Java, the system provides a visually appealing and functional interface where users can input their mobile number and desired recharge amount.

**Concepts Involved:**

1. GUI Design: The application uses the Swing framework to create a window with labels, text fields, and buttons, ensuring a smooth user interaction.
2. Event Handling: Implements ActionListener to handle button click events, making the application responsive to user actions.
3. Input Validation: Validates user input to ensure the mobile number is exactly 10 digits long and the recharge amount is one of the predefined packages (100, 150, 200, 500, 1000).
4. Error Handling: Uses dialog boxes to provide feedback to users when invalid input is detected, enhancing usability.
5. Layouts and Positioning: Utilizes null layout to manually position components within the window, offering precise control over the UI design.
6. JFrame: Extends JFrame to create the main application window, setting properties such as title, size, and default close operation.
7. JLabel: Used to label the input fields for mobile number and recharge amount, guiding user input.
8. JTextField: Allows users to enter their mobile number and the amount they wish to recharge.
9. JButton: A button labeled "Recharge" that users click to initiate the recharge process.
10. JOptionPane: Provides pop-up messages for user feedback, informing them of successful recharges or input errors.

The Online Mobile Recharge System combines these concepts to create an effective and reliable solution for managing mobile recharges, leveraging the power of Java Swing for a robust and interactive application. This system exemplifies how modern GUI applications can enhance user convenience and operational efficiency in everyday tasks.

**Step1:-** Initialization:

# ALGORITHM

* + Create a class for the application.
  + Extend the GUI framework's main window class.
  + Implement the interface for handling button click events.

**Step2:-** Frame Setup:

* + Set the window title.
  + Define window dimensions (width and height).
  + Specify that the application should exit when the window is closed.
  + Choose a layout manager that allows manual placement of components**. Step3:-** UI Components:
  + Add a label for mobile number:
  + Position it at a specified location.
  + Add a text field for mobile number:
  + Position it next to the mobile number label.
  + Add a label for amount:
  + Position it at a specified location.
  + Add a text field for amount:
  + Position it next to the amount label.
  + Add a recharge button:
  + Position it in the window.
  + Set it to listen for click events.

**Step4:-** Button Click Handling:

* + When the button is clicked:
  + Retrieve the text from the mobile number field.
  + Retrieve the text from the amount field.
  + Check if the mobile number has exactly 10 digits.
  + If not, show an error message and exit the event handler.
  + Try to convert the amount text to an integer.
  + If conversion fails, show an error message and exit the event handler.
  + If the amount is not 100, 150, 200, 500, or 1000, show an error message and exit the event handler.
  + Display a success message with the mobile number and recharge amount.

**Step5:-** Application Launch:

* + Create an instance of the main class.
  + Make the application window visible to the user.

This algorithm breaks down the process of creating and running the mobile recharge system application step by step, ensuring each component and interaction is clearly defined.

# REQUIREMENTS

1. A computer/Laptop compatible with Eclipse
2. Eclipse IDE
3. JAVA JDK
4. JAVA JRE

# IMPLEMENTATION / SOURCE CODE

**package** project1; **import** javax.swing.\*; **import** java.awt.event.\*;

**public class** MobileRechargeSystem **extends** JFrame **implements** ActionListener {

**private** JLabel mobileNumberLabel, amountLabel; **private** JTextField mobileNumberField, amountField; **private** JButton rechargeButton;

**public** MobileRechargeSystem() { setTitle("Online Mobile Recharge System"); setSize(400, 200);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); setLayout(**null**);

mobileNumberLabel = **new** JLabel("Mobile Number:"); mobileNumberLabel.setBounds(30, 30, 100, 20); add(mobileNumberLabel);

mobileNumberField = **new** JTextField(); mobileNumberField.setBounds(140, 30, 200, 20); add(mobileNumberField);

amountLabel = **new** JLabel("Amount:"); amountLabel.setBounds(30, 60, 100, 20); add(amountLabel);

amountField = **new** JTextField(); amountField.setBounds(140, 60, 200, 20); add(amountField);

rechargeButton = **new** JButton("Recharge"); rechargeButton.setBounds(140, 100, 100, 30); rechargeButton.addActionListener(**this**); add(rechargeButton);

}

**public void** actionPerformed(ActionEvent e) {

**if** (e.getSource() == rechargeButton) {

String mobileNumber = mobileNumberField.getText(); String amountText = amountField.getText();

// Validate mobile number length

**if** (mobileNumber.length() != 10) {

JOptionPane.*showMessageDialog*(**this**, "Please enter correct mobile number (10 digits)");

**return**;

}

// Validate the amount

**int** amount;

**try** {

amount = Integer.*parseInt*(amountText);

} **catch** (NumberFormatException ex) { JOptionPane.*showMessageDialog*(**this**, "Please enter a valid amount"); **return**;

}

**if** (amount != 100 && amount != 150 && amount != 200 && amount != 500 && amount != 1000) {

1000)");

}

JOptionPane.*showMessageDialog*(**this**, "Please enter a valid amount (100, 150, 200, 500,

**return**;

// Here, you can add code to perform the recharge operation

// For simplicity, let's just display a message

JOptionPane.*showMessageDialog*(**this**, "Recharge Successful for " + mobileNumber + " with amount " + amount);

}

}

**public static void** main(String[] args) {

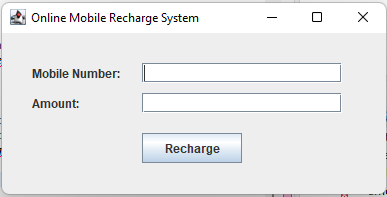
MobileRechargeSystem rechargeSystem = **new** MobileRechargeSystem(); rechargeSystem.setVisible(**true**);

}

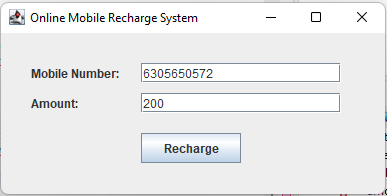
}

# RESULTS

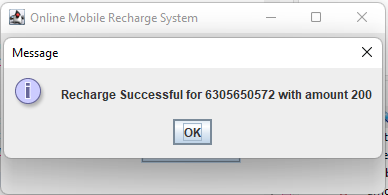
* The GUI output



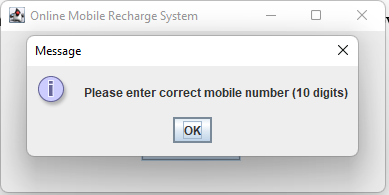
* After giving input



* Click on the recharge ,then it shows recharge successfull



* If you enter the mobile number more than or less than 10 digits , then it shows



# CONCLUSION

The Mobile Recharge System is a straightforward and efficient Java application utilizing the Swing framework to provide a user-friendly interface for recharging mobile phones. It ensures input validation for both mobile numbers and recharge amounts, enhancing user experience and preventing errors. By automating the recharge process, it saves time and offers convenience. This application effectively demonstrates the integration of basic GUI components and event handling in Java. Overall, it simplifies mobile recharges and improves operational efficiency.

# REFERENCES

1. Java the complete reference, 8th Editon, Herbert Schildt, TMH.
2. Java How to Program, H. M. Dietel and P. J. Dietel, 6th Edition, Pearson Education/PHI.
3. Introduction to Java programming, Y. Daniel Liang, Pearson Education.