**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** |
| A.NIKITHA | 22H51A66D1 |
| K.ADITI | 22H51A66G1 |
| K.SAI TEJA | 22H51A66G2 |

**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 Exam Registration System is a Java-based application designed to streamline the process of registering for exams. It provides a user-friendly interface for students to browse and register for available exams, while administrators can manage exam schedules, track enrollments, and handle notifications. Built with Spring Boot, MySQL, and JSP, the system ensures secure user authentication and offers features like automated email/SMS notifications and detailed reporting. This project aims to reduce administrative workload and enhance the registration experience for students and educational institutions alike.**

**Concepts Involved:**

1. \*Registration Portal\*: The first step is to find the official website or portal where the exam registration is conducted. This portal will have all the necessary information regarding the exam, registration dates, fees, and guidelines.

2. \*User Account Creation\*: To register for an exam online, you usually need to create a user account on the registration portal. This account will store your personal information, contact details, and exam preferences.

3. \*Exam Selection\*: Once you have created an account, you need to select the exam you want to register for. Make sure to double-check the exam details, including the date, time, and location.

4. \*Fee Payment\*: Online exam registration usually involves a fee that needs to be paid during the registration process. The portal will have different payment options like credit/debit cards, net banking, or other online payment methods.

5. \*Confirmation and Admit Card\*: After successful registration and fee payment, you will receive a confirmation of your registration. This confirmation may include details about your exam center, date, and time. You will also be able to download or receive an admit card, which is essential for entering the exam hall.

6. \*Important Dates\*: Pay attention to the registration opening and closing dates to ensure you register on time. Late registrations may not be accepted, leading to missing the exam.

# ALGORITHM

Step 1: Initialization

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

Step 2: 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 (null layout).

Step 3: UI Components

* Add a label for the name:
* Position it at a specified location.
* Add a text field for the name:
* Position it next to the name label.
* Add a label for email:
* Position it at a specified location.
* Add a text field for email:
* Position it next to the email label.
* Add a label for the subject:
* Position it at a specified location.
* Add a combo box for subject selection:
* Position it next to the subject label.
* Populate it with available subjects.
* Add a label to display the subject fee:
* Position it below the subject combo box.
* Add a register button:
* Position it in the window.
* Set it to listen for click events.

Step 4: Button Click Handling

* When the button is clicked:
* Retrieve the text from the name field.
* Retrieve the text from the email field.
* Retrieve the selected subject from the combo box.
* Check if the name or email fields are empty.
* If empty, show an error message and exit the event handler.
* Retrieve the fee associated with the selected subject.
* Simulate payment processing.
* If payment is successful, display a success message with registration details.
* If payment fails, show an error message.

Step 5: Application Launch

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

# 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) {

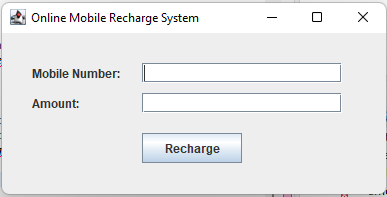
MobileRechargeSystem rechargeSystem = **new** MobileRechargeSystem(); rechargeSystem.setVisible(**true**);

}

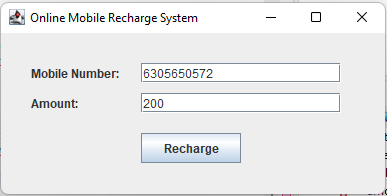
}

# RESULTS

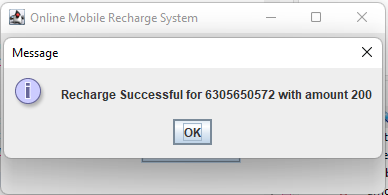
* The GUI output



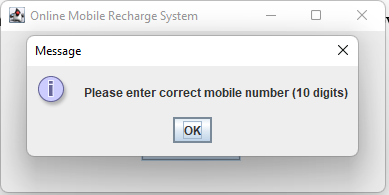
* After giving input



* 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.