



# **DISASTER MANAGEMENT SYSTEM**



## **A PROJECT REPORT**

*Submitted by*

**RAGHUL S (23081171021126)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 - JAVA PROGRAMMING**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

**SAMAYAPURAM-621112**

**NOVEMBER- 2024**

**K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY  
(AUTONOMOUS)**

**SAMAYAPURAM-621112**

**BONAFIDECERTIFICATE**

Certified that this project report on “**DISASTER MANAGEMENT SYSTEM** ” is the bonafide work of **RAGHUL S(2303811710421126)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING  
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,  
HEAD OF THE DEPARTMENT  
PROFESSOR

**SIGNATURE**

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

**HEAD OF THE DEPARTMENT**

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING  
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),  
SUPERVISOR  
ASSISTANT PROFESSOR

**SIGNATURE**

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.),

**SUPERVISOR**

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 03.12.2024

CGB1201- JAVA PROGRAMMING  
Mr.MANJARMANNAN A, M.E.,  
INTERNAL EXAMINER  
ASSISTANT PROFESSOR

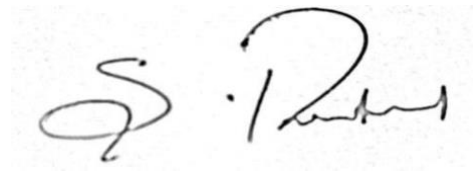
**INTERNAL EXAMINER**

CGB1201-JAVA PROGRAMMING  
Mrs.AARUNA PRIYA, M.E.,  
EXTERNAL EXAMINER  
ASSISTANT PROFESSOR  
8104-DSEC, PERAMBALUR.

**EXTERNAL EXAMINER**

## DECLARATION

I declare that the project report on “**DIASTER MANAGEMENT SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201-JAVA PROGRAMMING**.



---

RAGHUL S

Place: Samayapuram

Date: 03/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequateduration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patiencewhich motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards

## **MISSION OF THE INSTITUTION**

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

## **VISION OF DEPARTMENT**

To be a center of eminence in creating competent software professionals with research and innovative skills.

## **MISSION OF DEPARTMENT**

**M1: Industry Specific:** To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

**M2: Research:** To prepare students for research-oriented activities.

**M3: Society:** To empower students with the required skills to solve complex technological problems of society.

## **PROGRAM EDUCATIONAL OBJECTIVES**

### **1. PEO1: Domain Knowledge**

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

### **2. PEO2: Employability Skills and Research**

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

### **3. PEO3: Ethics and Values**

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

#### **PSO 1: Domain Knowledge**

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

#### **PSO 2: Quality Software**

To apply software engineering principles and practices for developing quality software for scientific and business applications.

#### **PSO 3: Innovation Ideas**

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

### **PROGRAM OUTCOMES (POs)**

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## ABSTRACT

The **Disaster Management System** is a Java-based desktop application designed using Swing, providing an efficient solution for managing disaster records and coordinating rescue teams. The application allows users to add, view, and filter disaster records based on name, location, and severity levels (Low, Medium, or High). It also supports rescue team management, enabling users to register teams with a name and validated 10-digit contact number. The system offers features such as viewing all recorded disasters and rescue teams, filtering disasters by severity, displaying statistical summaries (including the total number of disasters and their severity breakdown), and clearing all data with user confirmation. The user interface is built with a tabbed design, separating disaster and rescue team management for clarity and ease of use. A central display area provides outputs like added records, filtered results, and statistics. This application exemplifies the effective use of Java's Swing library, Object-Oriented Programming (OOP) principles, and collections like ArrayList and Stream API to deliver an organized and interactive management solution.



### ABSTRACT WITH POs AND PSOs MAPPING

#### CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Disaster Management System is a Java-based desktop application designed using Swing, providing an efficient solution for managing disaster records and coordinating rescue teams. The application allows users to add, view, and filter disaster records based on name, location, and severity levels (Low, Medium, or High). It also supports rescue team management, enabling users to register teams with a name and validated 10-digit contact number. The system offers features such as viewing all recorded disasters and rescue teams, filtering disasters by severity, displaying statistical summaries (including the total number of disasters and their severity breakdown), and clearing all data with user confirmation. The user interface is built with a tabbed design, separating disaster and rescue team management for clarity and ease of use.	<b>PO1 -3</b> <b>PO2 -3</b> <b>PO3 -3</b> <b>PO5 -3</b> <b>PO6 -3</b> <b>PO8 -3</b> <b>PO9 -3</b> <b>PO10 -3</b> <b>PO11-3</b> <b>PO12 -3</b>	<b>PSO1 -3</b> <b>PSO2 -3</b> <b>PSO3 -3</b>

Note: 1- Low, 2-Medium, 3- High

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	viii
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
<b>2</b>	<b>PROJECT METHODOLOGY</b>	3
	2.1 Proposed Work	3
	2.2 Block Diagram	5
<b>3</b>	<b>MODULE DESCRIPTION</b>	6
	3.1 Add Disaster Module	6
	3.2 View Disaster Module	6
	3.3 Add Rescue Team Module	7
	3.4 View Rescue Team Module	7
	3.5 View Statistics Module	7
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	8
	4.1 Conclusion	8
	4.2 Future Scope	9
	<b>REFERENCES</b>	23
	<b>APPENDIX A (SOURCE CODE)</b>	10
	<b>APPENDIX B (SCREENSHOTS)</b>	20

## CHAPTER 1

### INTRODUCTION

#### 1.1 Objective

The objective of this code is to develop an interactive desktop application for managing disaster information and coordinating rescue teams. The application aims to streamline the process of recording, viewing, filtering, and analyzing data related to disasters and rescue operations. It facilitates efficient organization and retrieval of disaster records by severity and location, enables rescue team registration with validated contact information, and provides users with actionable insights through statistics. The system also offers a user-friendly interface to simplify data management tasks while ensuring robust data handling and flexibility for future extensions.

#### 1.2 Overview

The Disaster Management System is a Java-based application built using the Swing framework to manage disaster-related information and rescue team data. It features a tabbed interface that separates disaster and rescue team management, providing a clear and user-friendly layout. Users can record details of disasters, including their name, location, and severity (Low, Medium, or High), as well as register rescue teams with names and validated contact numbers. The application supports several operations, such as viewing all recorded disasters and teams, filtering disasters by severity, and displaying statistical summaries, including counts of disasters by severity and the total number of rescue teams. A dedicated option allows users to clear all stored data, ensuring easy data management. The central display area dynamically updates with the results of these operations. Built on Object-Oriented Programming principles, the system uses Java's ArrayList for data storage and the Stream API for filtering and statistical calculations. This program demonstrates efficient data handling

## 1.3 Java Programming Concepts

### ❖ Object-Oriented Programming (OOP):

- **Encapsulation:** The Disaster and RescueTeam classes encapsulate related properties (name, location, severity, etc.) and provide access to them via methods like getSeverity() and toString().
- **Abstraction:** The classes abstract the details of disasters and rescue teams, simplifying their management and interaction.

### ❖ Swing Framework:

- Used for creating a **Graphical User Interface (GUI)**.
- Components like JTabbedPane, JTextField, JTextArea, JLabel, and JButton are used to build the interface.

### ❖ Event Handling:

- **ActionListeners** are implemented to handle button clicks, enabling interactivity (e.g., adding records, filtering data, and clearing records).

### ❖ Collections Framework:

- **ArrayList** is used to store lists of Disaster and RescueTeam objects, enabling dynamic data storage and retrieval.
- The **Stream API** is used for filtering and statistical calculations on the disaster data.

### ❖ Data Validation:

- Input validation ensures only valid data is added (e.g., checking for empty fields and validating a 10-digit contact number for rescue teams).

### ❖ String Formatting and Presentation:

- The String.format method and toString methods are used to format object data for display in the text area.

### ❖ Error Handling:

- Input validation combined with JOptionPane messages provides user feedback for errors, preventing invalid data entry.

## CHAPTER 2

### PROJECT METHODOLOGY

#### 2.1 Proposed Work

##### Integration with External Data Sources

- **Disaster API Integration:** Fetch real-time disaster data from external APIs (e.g., global weather or disaster APIs) to keep the system updated.
- **Database Support:** Transition from in-memory ArrayList storage to a database (e.g., MySQL, SQLite) for persistent storage of disaster and rescue team data.

##### Enhanced Disaster Management

- **Detailed Disaster Records:** Add fields like date, time, and affected population for each disaster.
- **Categorization:** Enable categorization of disasters (e.g., natural disasters like earthquakes, floods, or man-made disasters like fires, accidents).
- **Geolocation:** Integrate with a map API (e.g., Google Maps) to show disaster locations visually.

##### Rescue Team Coordination

- **Availability Status:** Add a status indicator for rescue teams (e.g., *Available*, *On Duty*, *Unavailable*).
- **Assignment to Disasters:** Allow rescue teams to be assigned to specific disasters, with tracking for active missions.

##### Reporting and Analytics

- **Advanced Statistics:**
  - Visual representation of disaster frequency and severity over time using charts.
  - Reports on team utilization and disaster response times.

## User Interface Improvements

- **Modern UI Framework:** Upgrade the Swing-based UI to a more modern Java framework like JavaFX for a richer user experience.
- **Responsive Design:** Optimize the layout for better usability on different screen sizes.
- **Theming Options:** Provide multiple themes (e.g., light and dark modes) for improved user experience.

## Mobile and Web Accessibility

- **Web Application:** Develop a web-based version of the system for remote access.
- **Mobile App:** Build a mobile application to enable field personnel to access and update data in real-time.

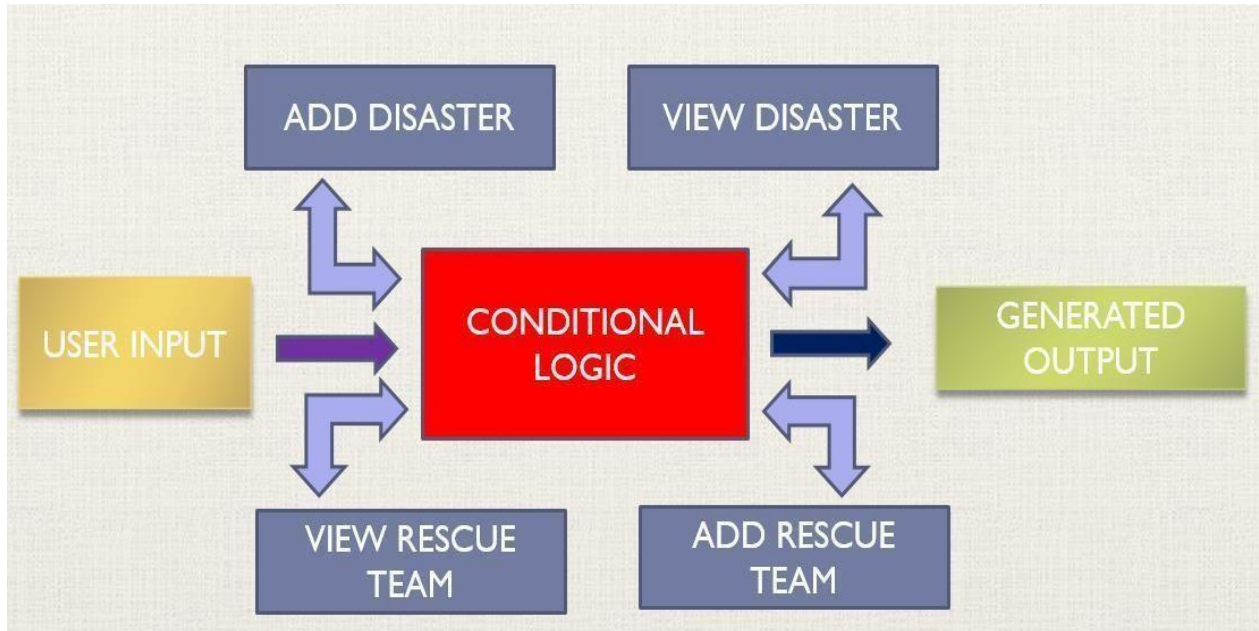
## Data Security and Backup

- **Encryption:** Secure sensitive data (e.g., rescue team contacts) using encryption techniques.
- **Data Backup:** Implement automated backups to prevent data loss.
- **Audit Logs:** Maintain logs of user actions for accountability and troubleshooting.

## Notifications and Alerts

- **Real-Time Alerts:** Send email or SMS notifications to rescue teams when new disasters are recorded.
- **Severity-Based Alerts:** Prioritize notifications based on disaster severity.

## 2.2 Block Diagram



## CHAPTER 3

### MODULE DESCRIPTION

#### 3.1 Add Disaster Module

##### Input Fields:

- **Disaster Name:** A (JTextField) to enter the name of the disaster.
- **Location:** A (JTextField) to specify the disaster's location.
- **Severity:** A (JComboBox) to select the disaster's severity level (Low, Medium, High).

##### Validation:

- Ensures all fields are filled.
- Displays an error message (JOptionPane) for missing or invalid inputs.

##### Data Storage:

- Adds valid disaster records to an ArrayList<Disaster> for dynamic storage.

#### 3.2 View Disaster Module

##### Access to Disaster Records:

- Displays all disasters stored in the ArrayList<Disaster>.

##### Formatted Output:

- Presents disaster details in a user-friendly format with fields such as name, location, and severity.

##### Error Handling:

- Notifies the user if no disasters have been recorded yet.



### 3.3 Add Rescue Team Module

#### Input Fields:

- **Team Name:** A JTextField for entering the rescue team's name.
- **Contact:** A JTextField for the team's 10-digit contact number.

#### Validation:

- Ensures all fields are filled.
- Checks that the contact number matches a valid 10-digit format.

### 3.4 View Rescue Team Module

#### Formatted Output:

- The rescue team details (name and contact) are presented clearly for user review.

#### Error Handling:

- Notifies the user if no rescue teams have been recorded yet.

### 3.5 View Statistics Module

#### Disaster Metrics:

- Displays the total number of disasters.
- Breaks down the disasters by severity (Low, Medium, High).

#### Rescue Team Metrics:

- Displays the total number of rescue teams recorded.

#### User Feedback:

- Displays the calculated statistics in a structured format within the JTextArea.

## CHAPTER 4

### CONCLUSION AND FUTURE SCOPE

#### 4.1 CONCLUSION

The **Disaster Management System** is a Java-based GUI application designed using the Swing framework to effectively manage disasters and rescue teams. It allows users to add, view, and manage disaster records by specifying details such as name, location, and severity (Low, Medium, High). Similarly, it enables the addition and review of rescue team details, including team names and 10-digit contact numbers. The program offers functionality for viewing disaster statistics, including the total number of disasters, the count of rescue teams, and the breakdown of disasters by severity. Users can filter disaster records based on severity and clear all stored data when necessary. The interface is user-friendly, featuring interactive elements like validation for input fields and pop-up dialogs for error messages and confirmation prompts. While the program is functional for basic management tasks, it can be enhanced further by integrating a database for persistent storage, adding advanced search options, enabling real-time updates, or supporting multi-user access. Overall, this application provides a solid framework for disaster and rescue management, with potential for expansion to meet more complex needs.

## 4.2 FUTURE SCOPE

The **Disaster Management System** has a promising future scope with numerous opportunities for enhancement and scalability. Integrating a database like MySQL or PostgreSQL would allow for persistent data storage, ensuring that disaster and rescue team records are retained beyond runtime. Expanding the application into a web-based or mobile platform would increase accessibility, enabling real-time updates and monitoring from any device. Features such as real-time alerts and updates, powered by APIs like OpenWeather or USGS, could significantly improve the system's responsiveness to ongoing disasters. Enhanced search and filter functionalities, including sorting by location, date, or severity, would make data management more efficient. The addition of role-based access control (RBAC) could enable secure, multi-user access, catering to administrators, managers, and rescue team members. Geolocation integration using tools like Google Maps API could provide visual representation of disaster locations and rescue operations, aiding in planning and resource allocation.

## APPENDIX A

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.stream.Collectors;

class Disaster {
    private String name;
    private String location;
    private String severity;

    public Disaster(String name, String location, String severity) {
        this.name = name;
        this.location = location;
        this.severity = severity;
    }

    public String getSeverity() {
        return severity;
    }

    @Override
    public String toString() {
        return String.format("Name: %s\nLocation: %s\nSeverity: %s", name,
location, severity);
    }
}
```

```

}

class RescueTeam {
    private String name;
    private String contact;

    public RescueTeam(String name, String contact) {
        this.name = name;
        this.contact = contact;
    }

    @Override
    public String toString() {
        return String.format("Name: %s%nContact: %s", name, contact);
    }
}

public class DisasterManagementSystem extends JFrame {
    private final ArrayList<Disaster> disasters = new ArrayList<>();
    private final ArrayList<RescueTeam> rescueTeams = new ArrayList<>();

    private JTextField disasterNameField, locationField, teamNameField,
contactField;
    private JComboBox<String> severityDropdown;
    private JTextArea displayArea;

    public DisasterManagementSystem() {
        setTitle("Disaster Management System");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```
setSize(900, 700);

setLayout(new BorderLayout(10, 10));

JTabbedPane tabs = new JTabbedPane();

// Disaster Tab
JPanel disasterTab = new JPanel(new BorderLayout(10, 10));
disasterTab.add(createDisasterInputPanel(), BorderLayout.NORTH);
disasterTab.add(createDisplayPanel(), BorderLayout.CENTER);
tabs.add("Manage Disasters", disasterTab);

// Rescue Team Tab
JPanel rescueTab = new JPanel(new BorderLayout(10, 10));
rescueTab.add(createRescueTeamInputPanel(), BorderLayout.NORTH);
rescueTab.add(createDisplayPanel(), BorderLayout.CENTER);
tabs.add("Manage Rescue Teams", rescueTab);

// Add tabs to frame
add(tabs, BorderLayout.CENTER);
add(createButtonPanel(), BorderLayout.SOUTH);

setVisible(true);
}

private JPanel createDisasterInputPanel() {
    JPanel panel = new JPanel(new GridLayout(5, 2, 5, 5));
    panel.setBorder(BorderFactory.createTitledBorder("Add Disaster"));

    disasterNameField = new JTextField();
```

```
locationField = new JTextField();  
  
severityDropdown = new JComboBox<>(new String[] {"Low", "Medium",  
"High"});
```

```
panel.add(new JLabel("Name:"));  
panel.add(disasterNameField);  
panel.add(new JLabel("Location:"));  
panel.add(locationField);  
panel.add(new JLabel("Severity:"));  
panel.add(severityDropdown);
```

```
JButton addButton = new JButton("Add Disaster");  
addButton.addActionListener(e -> addDisaster());  
panel.add(new JLabel());  
panel.add(addButton);
```

```
return panel;  
}
```

```
private JPanel createRescueTeamInputPanel() {  
    JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));  
    panel.setBorder(BorderFactory.createTitledBorder("Add Rescue Team"));  
  
    teamNameField = new JTextField();  
    contactField = new JTextField();  
  
    panel.add(new JLabel("Name:"));  
    panel.add(teamNameField);  
    panel.add(new JLabel("Contact:"));
```

```
panel.add(contactField);
```

```
        JButton addButton = new JButton("Add Team");  
        addButton.addActionListener(e -> addRescueTeam());  
        panel.add(new JLabel());  
        panel.add(addButton);  
  
        return panel;  
    }  
  
    private JPanel createButtonPanel() {  
        JPanel panel = new JPanel(new GridLayout(1, 5, 10, 10));  
  
        JButton viewDisastersButton = new JButton("View Disasters");  
        JButton viewTeamsButton = new JButton("View Teams");  
        JButton statisticsButton = new JButton("View Statistics");  
        JButton clearDataButton = new JButton("Clear All Data");  
        JButton filterButton = new JButton("Filter Disasters");  
  
        viewDisastersButton.addActionListener(e -> viewDisasters());  
        viewTeamsButton.addActionListener(e -> viewRescueTeams());  
        statisticsButton.addActionListener(e -> viewStatistics());  
        clearDataButton.addActionListener(e -> clearAllData());  
        filterButton.addActionListener(e -> filterDisasters());  
  
        panel.add(viewDisastersButton);  
        panel.add(viewTeamsButton);  
        panel.add(statisticsButton);  
        panel.add(clearDataButton);
```



```
panel.add(filterButton);
```

```
return panel;
```

```
}
```

```
private JPanel createDisplayPanel() {
```

```
    JPanel panel = new JPanel(new BorderLayout());
```

```
    displayArea = new JTextArea();
```

```
    displayArea.setEditable(false);
```

```
    displayArea.setFont(new Font("Arial", Font.PLAIN, 14));
```

```
    panel.add(new JScrollPane(displayArea), BorderLayout.CENTER);
```

```
    return panel;
```

```
}
```

```
private void addDisaster() {
```

```
    String name = disasterNameField.getText().trim();
```

```
    String location = locationField.getText().trim();
```

```
    String severity = (String) severityDropDown.getSelectedItem();
```

```
    if (name.isEmpty() || location.isEmpty()) {
```

```
        JOptionPane.showMessageDialog(this, "All fields must be filled.", "Error",
```

```
JOptionPane.ERROR_MESSAGE);
```

```
        return;
```

```
    }
```

```
    disasters.add(new Disaster(name, location, severity));
```

```
    displayArea.setText("Disaster added successfully!");
```

```
    disasterNameField.setText("");
```

```
    locationField.setText("");
```

}

```
private void addRescueTeam() {  
    String name = teamNameField.getText().trim();  
    String contact = contactField.getText().trim();  
  
    if (name.isEmpty() || !contact.matches("\\d{10}")) {  
        JOptionPane.showMessageDialog(this, "Enter a valid name and a 10-digit  
contact number.", "Error", JOptionPane.ERROR_MESSAGE);  
        return;  
    }  
  
    rescueTeams.add(new RescueTeam(name, contact));  
    displayArea.setText("Rescue Team added successfully!");  
    teamNameField.setText("");  
    contactField.setText("");  
}  
  
private void viewDisasters() {  
    if (disasters.isEmpty()) {  
        displayArea.setText("No disasters recorded.");  
        return;  
    }  
  
    StringBuilder output = new StringBuilder("Disasters:\n");  
    for (Disaster d : disasters) {  
        output.append(d).append("\n\n");  
    }  
    displayArea.setText(output.toString());  
}
```

}

```
private void viewRescueTeams() {
```

```
    if (rescueTeams.isEmpty()) {
```

```
        displayArea.setText("No rescue teams recorded.");
```

```
        return;
```

```
    }
```

```
    StringBuilder output = new StringBuilder("Rescue Teams:\n");
```

```
    for (RescueTeam team : rescueTeams) {
```

```
        output.append(team).append("\n\n");
```

```
    }
```

```
    displayArea.setText(output.toString());
```

```
}
```

```
private void viewStatistics() {
```

```
    int totalDisasters = disasters.size();
```

```
    int lowSeverity = (int) disasters.stream().filter(d ->  
d.getSeverity().equalsIgnoreCase("Low")).count();
```

```
    int mediumSeverity = (int) disasters.stream().filter(d ->  
d.getSeverity().equalsIgnoreCase("Medium")).count();
```

```
    int highSeverity = (int) disasters.stream().filter(d ->  
d.getSeverity().equalsIgnoreCase("High")).count();
```

```
    int totalTeams = rescueTeams.size();
```

```
    String stats = String.format("Total Disasters: %d\nTotal Rescue Teams:  
%d\nLow Severity Disasters: %d\nMedium Severity Disasters: %d\nHigh  
Severity Disasters: %d",
```

```
        totalDisasters, totalTeams, lowSeverity, mediumSeverity, highSeverity);
```

}

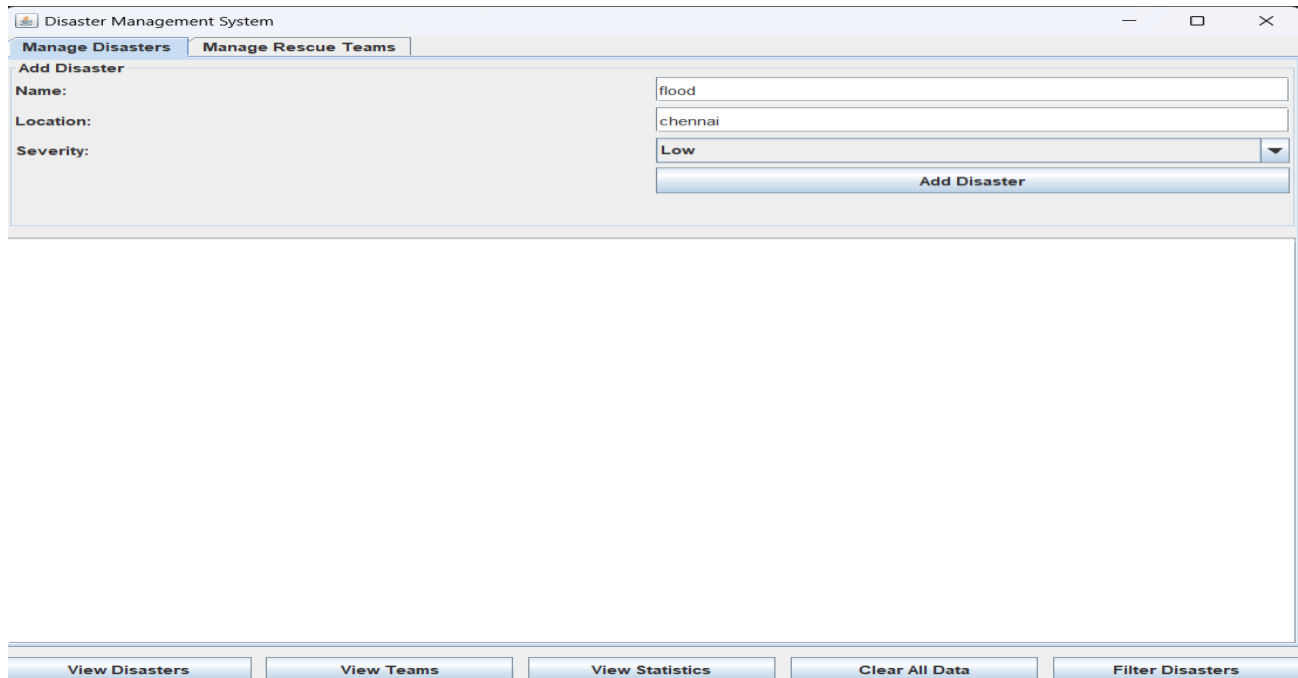
```
private void clearAllData() {  
    int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to  
clear all data?", "Confirm", JOptionPane.YES_NO_OPTION);  
    if (confirm == JOptionPane.YES_OPTION) {  
        disasters.clear();  
        rescueTeams.clear();  
        displayArea.setText("All data has been cleared.");  
    }  
}
```

```
private void filterDisasters() {  
    String severity = JOptionPane.showInputDialog(this, "Enter severity to filter  
(Low, Medium, High):");  
    if (severity == null || severity.isEmpty()) return;  
  
    String filtered = disasters.stream()  
        .filter(d -> d.getSeverity().equalsIgnoreCase(severity))  
        .map(Disaster::toString)  
        .collect(Collectors.joining("\n\n"));  
  
    if (filtered.isEmpty()) {  
        displayArea.setText("No disasters found with severity: " + severity);  
    } else {  
        displayArea.setText("Filtered Disasters:\n" + filtered);  
    }  
}
```

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(DisasterManagementSystem::new);  
}  
}
```

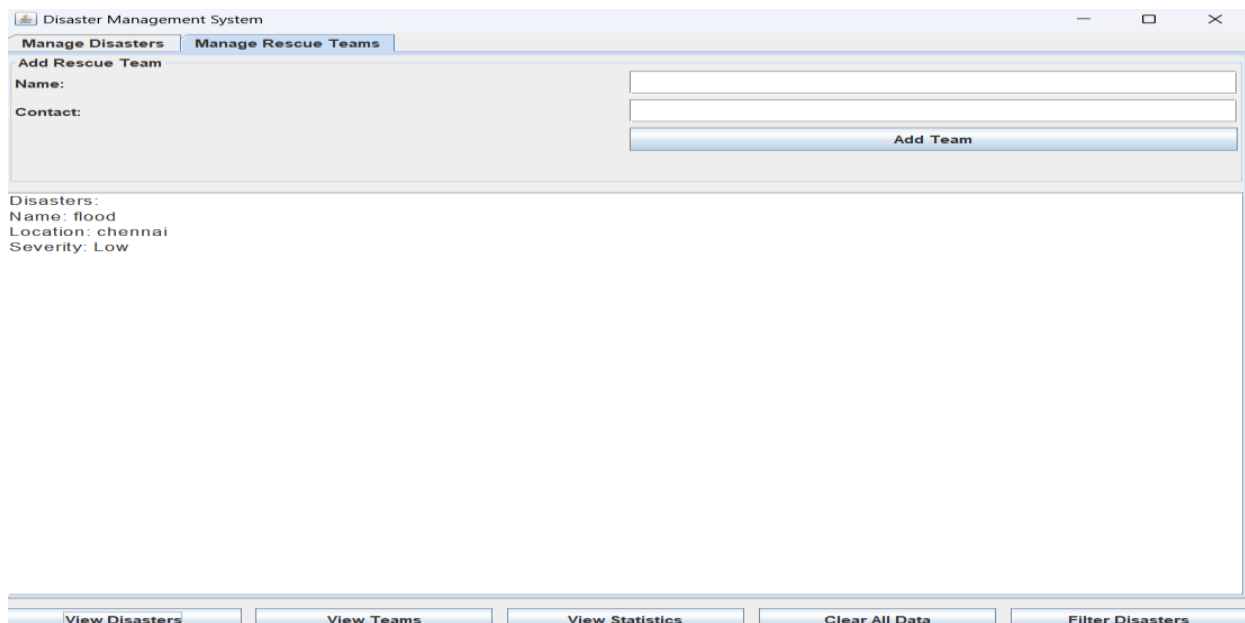
## APPENDIX B

### Add Disaster Screen



The screenshot shows the 'Disaster Management System' window with the 'Manage Disasters' tab selected. The 'Add Disaster' section contains three input fields: 'Name' with the value 'flood', 'Location' with the value 'chennai', and 'Severity' with a dropdown menu set to 'Low'. An 'Add Disaster' button is located below these fields. At the bottom of the window, there are five buttons: 'View Disasters', 'View Teams', 'View Statistics', 'Clear All Data', and 'Filter Disasters'.

### View Disaster Screen



The screenshot shows the 'Disaster Management System' window with the 'Manage Rescue Teams' tab selected. The 'Add Rescue Team' section contains two input fields: 'Name' and 'Contact', with an 'Add Team' button below them. Below this section, the 'Disasters' section displays the details of the added disaster: 'Name: flood', 'Location: chennai', and 'Severity: Low'. At the bottom of the window, there are five buttons: 'View Disasters', 'View Teams', 'View Statistics', 'Clear All Data', and 'Filter Disasters'.

## Add Rescue Team Screen

Disaster Management System

Manage Disasters
Manage Rescue Teams

Add Rescue Team

Name:
Contact:

chennai rescue team
8270347668

Add Team

View Disasters
View Teams
View Statistics
Clear All Data
Filter Disasters

## View Rescue Team Screen

Disaster Management System

Manage Disasters
Manage Rescue Teams

Add Rescue Team

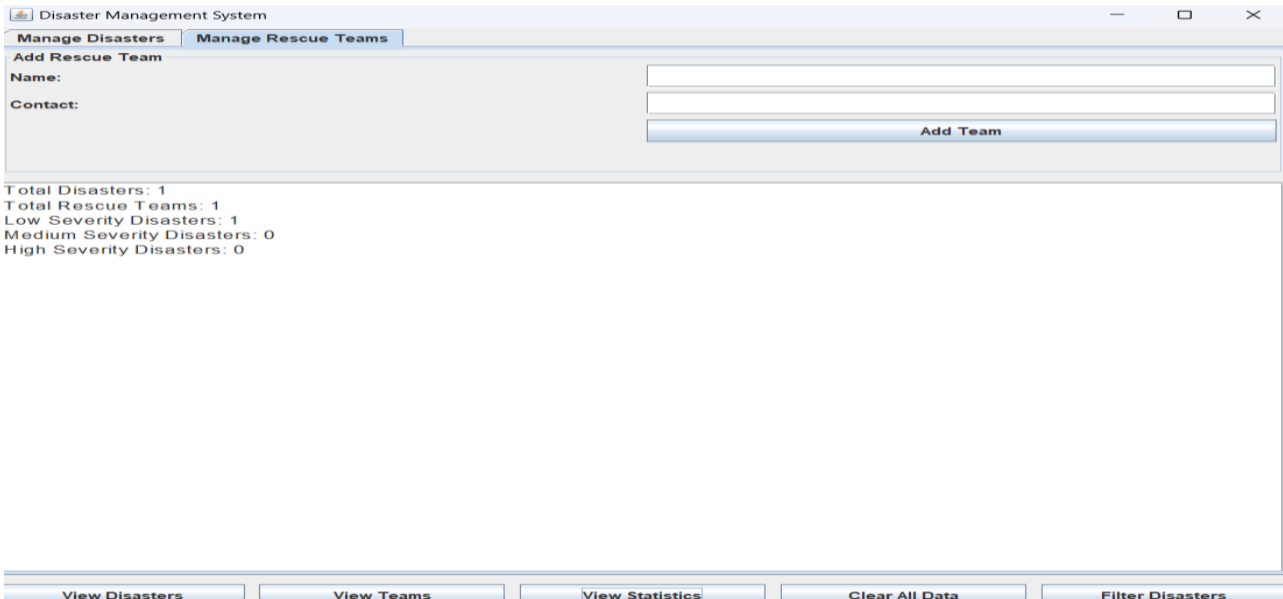
Name:
Contact:

Add Team

Rescue Teams:  
Name: chennai rescue team  
Contact: 8270347668

View Disasters
View Teams
View Statistics
Clear All Data
Filter Disasters

## View Statistics Screen



Disaster Management System

Manage Disasters | Manage Rescue Teams

Add Rescue Team

Name:

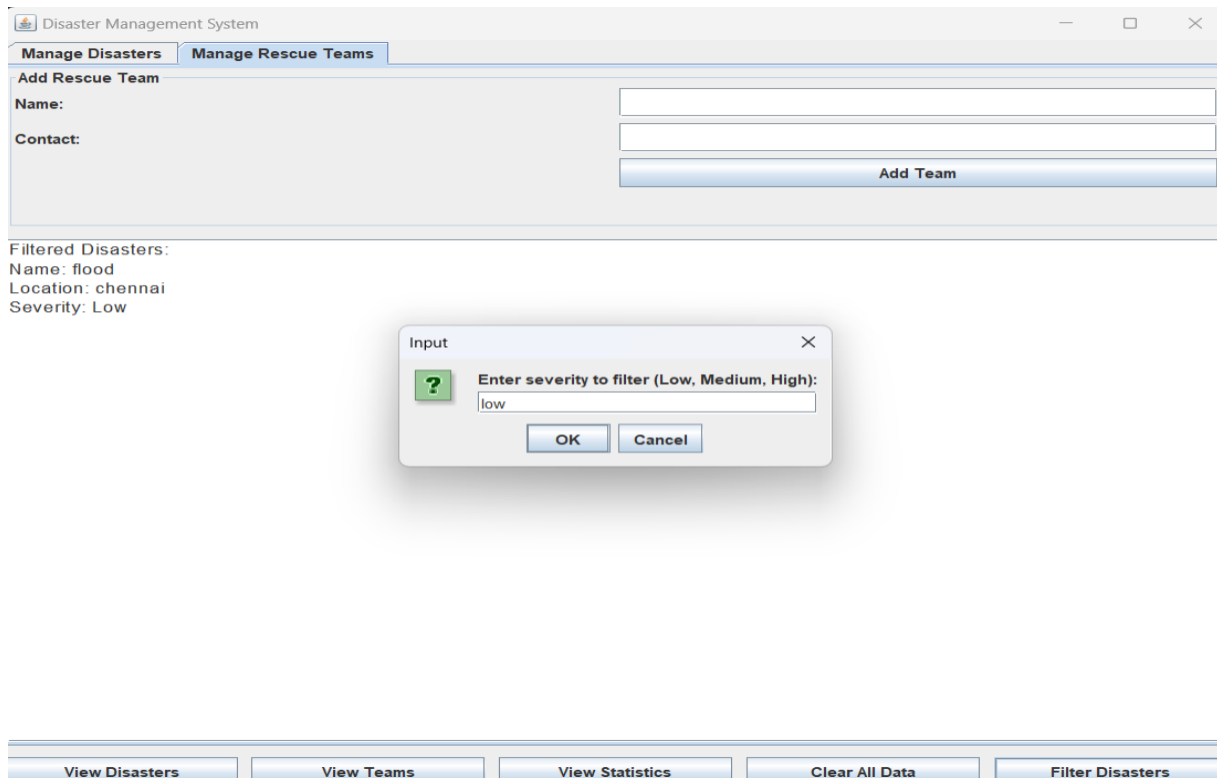
Contact:

Add Team

Total Disasters: 1  
Total Rescue Teams: 1  
Low Severity Disasters: 1  
Medium Severity Disasters: 0  
High Severity Disasters: 0

View Disasters | View Teams | View Statistics | Clear All Data | Filter Disasters

## Filtered Disaster Screen



Disaster Management System

Manage Disasters | Manage Rescue Teams

Add Rescue Team

Name:

Contact:

Add Team

Filtered Disasters:  
Name: flood  
Location: chennai  
Severity: Low

Input

Enter severity to filter (Low, Medium, High):

OK Cancel

View Disasters | View Teams | View Statistics | Clear All Data | Filter Disasters



## REFERENCES

### WEBSITES

- <https://www.disastermanagement.com/>
- <https://projectsgeek.com/>
- <https://stackoverflow.com/questions/1176282/java-open-source-helpdesk-workflow-project>

### YOUTUBE LINK

- <https://www.youtube.com/watch?v=sV3cigTJvG4>
- <https://www.youtube.com/watch?v=EBy-FOFddW8>