

A Project Report

On

**“Real-Time Disaster Information Aggregation Software”**

Batch Details

|  |  |  |
| --- | --- | --- |
| Sl. No. | Roll Number | Student Name |
| 1 | 20211ISR0021 | MONIKA P |
| 2 | 20211ISR0038 | DISHA R |
| 3 | 20211ISR0078 | BHAVANA B A |

# Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.

Under the guidance of,

**Dr. Sivaramakrishnan**

Presidency School of Information Science, Presidency University, Bengaluru

## CONTENTS

1. **Introduction about Project**
2. **Literature Review**
3. **Objectives**
4. **Methodology**
5. **Timeline for Execution of Project**
6. **Expected Outcomes**
7. **Conclusion**
8. **References**

### INTRODUCTION

Effective disaster response requires timely and accurate information. While government alerts and news reports provide crucial updates, social media platforms have become a valuable source of real-time data during crises. However, manually monitoring and analyzing this vast amount of information is inefficient and time-consuming.

To address this challenge, a software solution is needed to automate the aggregation and categorization of disaster-related data from social media, news portals, and open sources. By leveraging machine learning and natural language processing (NLP), the system will filter relevant information and present it on a user-friendly dashboard. This will enable disaster response teams to make quick, informed decisions, improving coordination and response time.

By streamlining data collection and analysis, this solution aims to enhance disaster management operations, ultimately saving lives and minimizing damage.

The core objectives of this platform include:

* **Automated Data Aggregation**
* **Efficient Categorization**
* **Real-time Insights**
* **User-friendly Dashboard**

### LITERATURE REVIEW

Developing a disaster response software solution requires an understanding of existing technologies and methodologies in data aggregation, real-time processing, and user interface design. Several studies and documentation sources provide valuable insights into these aspects.

Bostock (2024) discusses **D3.js**, a JavaScript library for visualizing data using HTML, SVG, and CSS. D3.js enables dynamic and interactive data visualizations, making it an essential tool for presenting real-time disaster data effectively on dashboards 111.

For real-time data processing, **Node.js** plays a crucial role. Pixel Free Studio (2024) highlights how Node.js is used to handle real-time data efficiently, particularly in scenarios requiring continuous data streaming and event-driven architecture. This capability is essential for disaster response systems that rely on instant updates from multiple sources 222.

The **Datalib JavaScript library**, as detailed by Vega Team (2024), provides various utilities for data transformation and analysis. It simplifies data handling, filtering, and statistical operations, making it useful for structuring disaster-related information into meaningful categories 333.

Webix, a **JavaScript UI library**, is another important tool for developing intuitive and responsive user interfaces. Webix Team (2024) emphasizes its capability to create dashboards with real-time data visualization, making it suitable for disaster management applications 444.

Additionally, GeeksforGeeks (2024) presents a study on building a **disaster management system using the MERN stack**. This tutorial provides practical insights into integrating MongoDB, Express.js, React.js, and Node.js for creating a scalable and efficient disaster response web application. The MERN stack ensures seamless front-end and back-end communication, enabling effective real-time data aggregation and decision-making support 555.

By leveraging these technologies, the proposed disaster response software can efficiently collect, process, and present disaster-related data, thereby improving emergency response operations.

### OBJECTIVES

· **Automated Data Aggregation** – Develop a system to collect disaster-related information from multiple sources, including social media and news portals, in real time.

· **Efficient Categorization** – Implement machine learning and NLP techniques to classify data based on disaster type, location, severity, and urgency.

· **Real-time Insights** – Provide disaster response teams with up-to-date, actionable intelligence for swift decision-making.

· **User-friendly Dashboard** – Design an intuitive interface to display categorized data, enabling quick analysis and response.

· **Improved Response Time** – Reduce the delay in identifying, verifying, and acting on emergency situations through automation.

· **Enhanced Coordination** – Facilitate seamless communication and collaboration among disaster response agencies via a centralized system.

### · METHODOLOGY

**DESIGN PROCEDURE**

**| 1. Requirement Analysis|**

**↓**

**| 2. Technology Selection |**

**↓**

**| 3. System Architecture |**

**↓**

**| 4. Data Collection & Preprocessing |**

**↓**

**| (Web Scraping, APIs, NLP) |**

**↓**

**| 5. Machine Learning Model |**

**↓**

**| (Classification, Sentiment Analysis) |**

**↓**

**| 6. Backend & API Development |**

**↓**

**| (Node.js, Express, MongoDB) |**

**↓**

**| 7. Dashboard Development |**

**↓**

**| (React.js, D3.js, Webix) |**

**↓**

**| 8. Testing & Deployment |**

### ****Explanation of the Design Procedure Flowchart****

The design procedure starts with **Requirement Analysis**, where key stakeholders, data sources, and system functionalities are identified. The system must collect, categorize, and visualize disaster-related data for efficient response.

Next, in **Technology Selection**, React.js is used for the frontend, Node.js with Express.js for the backend, and MongoDB for data storage. Python-based NLP and ML models handle classification, while APIs and web scraping tools extract data from social media and news sources.

The **System Architecture Design** includes four layers: data collection (scraping and APIs), processing (NLP and ML models), storage (MongoDB), and presentation (dashboard visualization). **Data Collection & Preprocessing** involves filtering and cleaning extracted information to ensure accuracy.

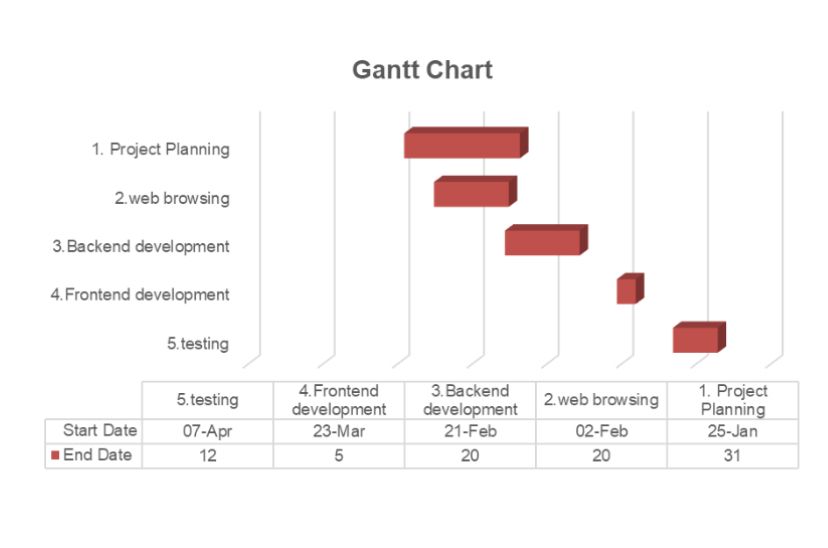
**Machine Learning Model Implementation** categorizes disasters, assesses urgency through sentiment analysis, and extracts geolocation data. **Backend & API Development** enables smooth data retrieval, real-time updates via WebSockets, and secure system communication.

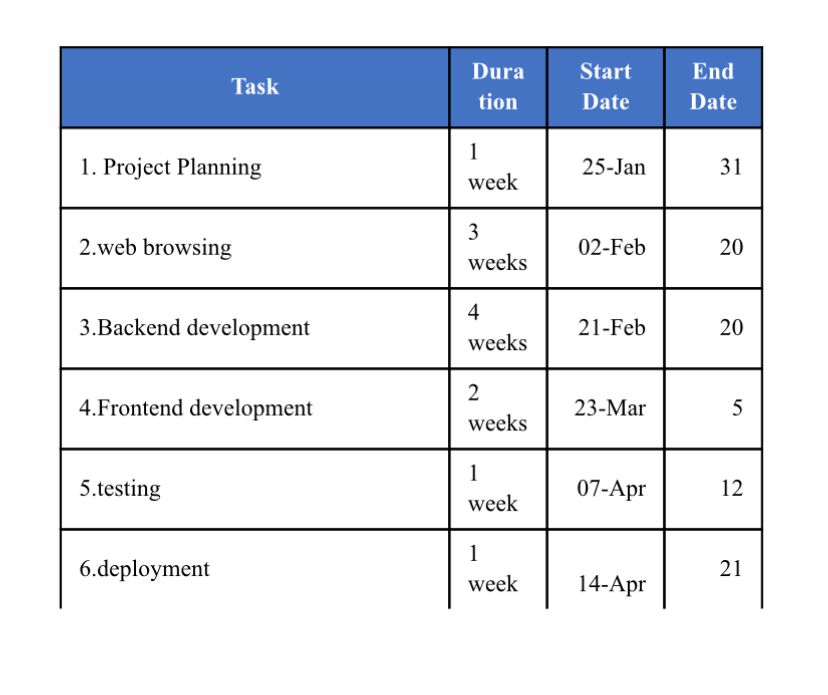
Finally, the **Dashboard Development** phase integrates interactive visualizations, followed by **Testing & Deployment** to ensure reliability, scalability, and seamless system performance.

### OUTCOMES

* **Faster Response Time:** Automates data collection, reducing delays in identifying and responding to disasters.
* **Accurate Disaster Classification:** Uses machine learning to categorize disasters based on severity, type, and location.
* **Real-time Situational Awareness:** Provides up-to-date insights through a user-friendly dashboard for emergency teams.
* **Improved Coordination:** Enhances communication between disaster response agencies for better decision-making.
* **Geolocation Mapping:** Identifies affected regions accurately, leading to quicker resource allocation.
* **Reduced Manual Effort:** Automates data processing, allowing agencies to focus on relief efforts rather than monitoring.
* **Enhanced Disaster Preparedness:** Ensures timely and well-informed actions, minimizing damage and saving lives.

### TIMELINE OF THE PROJECT/ PROJECT EXECUTION PLAN

****



1. **CONCLUSION**

The disaster response data aggregation system will enhance emergency response by automating the collection and classification of disaster-related information from social media, news portals, and other sources. This will **reduce response time** and provide real-time situational awareness for disaster management agencies.

By leveraging **machine learning models**, the system will accurately categorize disasters based on **type, severity, and location**, ensuring that emergency responders receive only relevant and verified data. The integration of **natural language processing (NLP)** will further refine data accuracy by filtering out misinformation and irrelevant content.

A **real-time interactive dashboard** will present disaster-related insights in an intuitive manner, allowing emergency teams to quickly assess ongoing crises. **Geolocation mapping** will pinpoint affected areas, facilitating **faster resource allocation and rescue operations**. Automated alerts will notify responders of critical developments, ensuring immediate action can be taken.

### REFERENCES

1. M. Bostock, “D3.js – Data-Driven Documents,” D3.js Official Documentation, 2024. [Online]. Available: https://en.wikipedia.org/wiki/D3.js. [Accessed: Jan. 30, 2025].
2. Pixel Free Studio, “How to Use Node.js for Real-Time Data Processing,” Pixel Free Studio Blog, 2024. [Online]. Available: https://blog.pixelfreestudio.com/how-to-use-node-js-for-real-time-data-processing. [Accessed: Jan. 30, 2025].
3. Vega Team, “Datalib – JavaScript Data Utility Library,” Vega Project Documentation, 2024. [Online]. Available: https://vega.github.io/datalib. [Accessed: Jan. 30, 2025].
4. Webix Team, “Webix: JavaScript UI Library,” Webix Official Documentation, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Webix. [Accessed: Jan. 30, 2025].
5. GeeksforGeeks, “Disaster Management Website using MERN,” GeeksforGeeks Tutorials, 2024. [Online]. Available: https://www.geeksforgeeks.org/disaster-management-website-using-mern. [Accessed: Jan. 30, 2025].