

DISASTER RECOGNITION AND MANAGEMENT USING API

A PROJECT REPORT

Submitted by,

Bhavana B A - 20211ISR0078

Disha R - 20211ISR0038

Monika P – 20211ISR0021

Under the guidance of,

Dr. Sivaramakrishnan.S

Associate Professor

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

**INFORMATION SCIENCE ENGINEERING
[ARTIFICIAL INTELLIGENCE AND ROBOTICS]**

At



PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report “**Disaster Recognition and Management using API**” being submitted by “Bhavana B A, Disha R, Monika P” bearing roll number(s) “20211ISR0078, 20211ISR0038, 20211ISR0021” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Information Science Engineering [Artificial Intelligence and Robotics] is a bonafide work carried out under my supervision.

Dr. Sivaramakrishnan.S
Associate Professor
School of CSE&IS
Presidency University

Dr. Zafar Ali Khan
Professor & HoD
School of CSE&IS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Disaster Recognition and Management using API** in partial fulfillment for the award of Degree of **Bachelor of Technology in Information Science Engineering [Artificial Intelligence and Robotics]** , is a record of our own investigations carried under the guidance of **Dr. Sivaramakirishnan.S, Associate Professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Bhavana B A - 20211ISR0078

Disha R - 20211ISR0038

Monika P – 20211ISR0021

ABSTRACT

The Disaster Information Aggregation Dashboard is an AI-enabled real-time platform to identify, interpret, and visualize real-time disaster events worldwide. The platform leverages the GPT-4 model of OpenAI to process natural language queries and respond with structured disaster data without using any traditional database. The system uses sophisticated methods like prompt-based querying, zero-shot learning, and natural language understanding to get valuable information out of unstructured sources and to monitor disasters accurately and dynamically.

The platform architecture consists of three main parts: the frontend, the backend, and the AI integration layer. The frontend has been implemented using React.js and features an interactive user interface with real-time maps developed using Leaflet.js and visualizations rendered using Chart.js. The frontend enables users to search for disaster data by type, severity, and location. The backend has been implemented using Node.js and Express.js and interacts with incoming queries by building optimized prompts, passing them to GPT-4, and formatting the responses received into something consumable by the frontend. Secure communication between the parts is enabled using Axios and CORS and configuration settings accessed using environment variables.

The AI engine is driven by OpenAI GPT-4, which has zero-shot learning ability. This makes it possible for the system to answer questions it has never been trained on before without any training on the particular task. GPT-4 looks at key disaster characteristics like type, severity, location (latitude and longitude), and sentiment when it reads a question. The system transforms all of this information into map and chart visuals and presents real-time situational awareness without storing any of it permanently.

Using real-time AI inference and obviating the necessity of traditional databases, the dashboard is both scalable as well as agile. The dashboard is capable of supporting a vast variety of disaster-triggered queries and providing up-to-date information in a responsive and secure manner. The employment of newer generation web technologies, efficient APIs, and AI inference makes the dashboard a robust tool when it comes to disseminating awareness about disasters, response planning, and communication during emergencies. The dashboard could have further support added to it in the future with other data feeds like multilingual querying and integration with real-time alert systems for particular types of disasters like floods, earthquakes, and storms.

ACKNOWLEDGE

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and Dr. “Zafar Ali Khan”, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr.Sivaramakrishnan.S Associate Professor** School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman**, department Project Coordinators “**Dr. Afroz Pasha**” and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Bhavana B A

Disha R

Monika P

LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 7.1.1	Timeline For Execution Of Project	25

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 6.1.1	System Design And Implementation Gantt Chart	20
2	Fig1.1	Disaster represented on world map	45
3	Fig1.2	Pie chart and Bar graph representation	45
4	Fig1.3	Disaster types	46
5	Fig1.4	Location where disaster has taken place	46
6	Fig 1.5	Latitude and Longitude for better geoloaction	47

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	ACKNOWLEDGMENT	v
	LIST OF TABLES	vi
	LIST OF FIGURES	vii
1.	INTRODUCTION	1-2
2.	LITERATURE REVIEW	3-5
3.	RESEARCH GAPS OF EXISTING METHODS	6-9
4.	PROPOSED METHODOLOGY	10-16
5.	OBJECTIVES	17-19
6.	SYSTEM DESIGN & IMPLEMENTATION	20-24
7.	TIMELINE FOR EXECUTION	25
8.	OUTCOMES	26-27
9.	RESULTS AND DISCUSSIONS	28-30
10.	CONCLUSION	31
11.	REFERENCES	32
12.	APPENDIX-A PSEUDOCODE	33-44
13.	APPENDIX-B	45-47
14.	APPENDIX-C	48

TABLE OF CONTENTS

6.		20-24
7.		25
8.		26-27
9.		28-31
10.		32
11.		40
12.		41-53
13.	ENCLOSURES	57

CHAPTER-1

INTRODUCTION

Natural and man-made disasters have the potential to risk life, infrastructure, and the ecosystem. Quick exposure to up-to-date and precise disaster information is essential to maintain public safety, facilitate coordinated responses to emergencies, and reduce losses. Conventional systems rely on ordered databases, official news feeds from government agencies, and piecemeal news reports. These are reliable but are often prone to delays or are non-localized and inaccessible in parts. Information is now available in excess but unstructured in an information-intensive world. There is a corresponding necessity to have smart systems to process real-time data and provide succinct and localized insights with minimal interference by humans.

To overcome this problem, a real-time Disaster Information Aggregation Dashboard has been implemented. The smart platform is engineered to deal with natural language requests and fetch real-time disaster information with the aid of large language models' capabilities. The system takes the shape of a framework atop OpenAI's GPT-4 and is able to process user requests using zero-shot learning mechanisms. The methodology has the ability to comprehend and answer requests without any requirement of custom training per task and has high adaptability as well as scalability in dynamic environments. Employing the technique of prompt engineering makes it possible to produce quality, formatted responses directly from unstructured data available on the web.

The architecture of the platform includes a strong backend developed with Node.js and Express.js to deal with API calls and formulate carefully crafted prompts to the GPT-4 model. The frontend was built using React.js and includes a simple-to-use interface to facilitate interactions between the user and the system in real time. The data is displayed

using interactive maps drawn using Leaflet.js and dynamic charts created with Chart.js to make it simple to visualize complicated disaster data. In contrast to traditional systems requiring data to be constantly stored, this system retrieves data in real-time on each request to avoid having a dedicated database and to maintain up-to-date information displayed.

The dashboard also includes Natural Language Understanding (NLU) to pull out such attributes as type of disaster, severity, location affected, and sentiment from the results of a query. The ability to process in real time makes the system particularly useful in times of emergencies, trip planning amidst environmental disasters, or policy-making. By providing a lean but efficient infrastructure, the platform is a contemporary response to disaster monitoring and situational awareness using a blend of artificial intelligence, real-time web fundamentals, and smart data visualization.

CHAPTER-2

LITERATURE SURVEY

The technologies of Artificial Intelligence (AI), Natural Language Processing (NLP), and geospatial have revolutionized the paradigm of disaster management in the last ten years. Several research studies have discussed ways to automate data retrieval, increase the precision of predictions and warnings related to disasters, and provide real-time situational insights. The review presented here identifies central research work on which the project was based and which guided its architecture and operations.

Patel et al. also suggested an AI-based earthquake response system to study real-time seismographic data. Their system utilized deep learning to classify events and showed the efficiency of utilizing models such as the BERT (Bidirectional Encoder Representations from Transformers) model to analyze text related to disasters. This emphasizes the importance of automatic mechanisms of classification, more so when it comes to large-scale and fast-changing scenarios of a disaster.

Kumbam and Vejre presented FloodLense, a real-time flood-detection framework using the ChatGPT to process real-time text data in the form of flood reports and news articles. The presentation highlighted the capability of the transformer architecture of the NLP models to extract useful insights from unstructured text information. This proves the methodology of employing large language models like GPT-4 in dynamic environments to make inferences when structured data might not be available.

Tripathy et al. addressed safe data sharing in cloud-IoT systems and proposed a cryptographic scheme to guarantee fine-grained access control with both forward and

backward security. While their work was more focused on security in resource-constrained IoT systems, it had an impact on the design of this project by underlining the significance of efficient, scalable, and safe data flows.

Saha et al. also came up with an architecture using IoT to monitor in real-time and send early warnings of natural disasters like floods and earthquakes. They used sensor and cloud computation to demonstrate the power of distributed systems in detecting disasters. This informed the real-time direction of the project while the solution proposed here substitutes real-time web data with AI-interpreted ones instead of sensor data.

Mody et al. developed Distress, an emergency response mobile application. Users could send alerts in real time and their location using GPS to both responders and emergency contacts. User-centric design and accessibility were priorities emphasized by their work, and it was the motivation to design an intuitive interface in this dashboard through React.js and map visualization using Leaflet.js.

Xu's work analyzed the use of sentiment analysis and natural language processing (NLP) methods such as VADER (Valence Aware Dictionary and sEntiment Reasoner) on data related to disasters. It highlighted how sentiment patterns are indicative of the severity and public perception of disasters and was incorporated into this system's real-time sentiment analysis to add an emotional context to reports of disasters.

Rajasekharan et al. drew attention to the significance of properly organized databases in enhancing the organisation, retrieval, and decision-making process using data related to disasters. While their study focused on the use of stable storage systems, the current dashboard takes a different stance by eschewing any database and directly retrieving

real-time data using GPT-4. The compromise comes with greater data freshness and less complexity in alignment with rapidly changing scenarios during disasters.

Rajabifard et al. focused on the importance of spatial data infrastructures in the visualization and analysis of disasters. Their research was in line with the concept of representing disasters on a map using location data (latitude and longitude), which forms the core aspect of mapping functionality in this dashboard via Leaflet.js.

Wallace and De Balogh discussed the efficiency of decision-support dashboards in managing emergencies and demonstrated how the visualization of real-time data significantly accelerates response rates and coordination. Their research necessitated the integration of a responsive and interactive dashboard to graphically depict real-time disaster occurrences with interactive features.

Kalidoss and Ravi also presented a model of managing disasters using IoT, cloud, and big data analytics to enhance emergency response. Their focus on instant data collection and low delay emphasized the necessity to build systems capable of handling and visualizing up-to-the-second information on disasters—a concept aligned with the employment of GPT-4 to fetch live data in the project. Collectively, these studies serve as a solid basis for establishing a system that is responsive as well as scalable and also endowed with the ability to comprehend, organize as well as visualize information on disasters via AI and NLP. In combining the insights of these studies, the existing dashboard system is a convergence of established methods and innovative technologies optimized towards real-time disaster intelligence.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

AI-driven disaster detection systems have been used in numerous studies like those conducted by Patel et al. and Kumbam & Vejre, in which AI models such as ChatGPT and BERT were utilized to classify and learn from disaster data. These models tend to specialize in a particular type of disaster, like earthquakes or floods, and rely on curated datasets or handcrafted data pipelines. Although the models yield good insights, the models are required to fine-tune to a particular type of disaster and thus are less resilient to novel, unexpected queries or evolving disaster scenarios. Consequently, they are less capable of reacting adaptively to emerging patterns of disasters or delivering generalized insights.

Concurrently, IoT and sensor systems proposed by Saha et al. and Kalidoss & Ravi rely on sensors and cloud computing to identify natural hazards. The systems are capable of early warning by monitoring changes in the environment but are rendered impractical in locations where the required infrastructure does not exist. The dependency on hardware deployment also reduces their scalability and usability in low-resource or rural areas with less available IoT infrastructure.

Emergency response mobile applications like Distress (by Mody et al.) are a user-focused system through which individuals are able to send out SOS messages and geolocation information. Although mobile applications are a responsive system to utilize in emergency situations on the go, the apps are centered on user alerts and do not include the data aggregation and analysis on multiple disasters required to get a better understanding of the overall impact of the disaster.

Static dashboards and disaster databases, as explored by Rajasekharan et al. and Rajabifard et al., focus on structured databases for disaster information retrieval. While these systems ensure data accuracy, they rely on static or scheduled updates, making them less effective in situations where real-time data is essential. This reliance on manual updates or database syncing can delay the delivery of critical information, leading to potential gaps in situational awareness during rapidly evolving disaster events.

Sentiment analyzing tools like VADER (employed in Xu's research) are typically used to analyze public mood through social media. They are good at sentiment extraction but poor at capturing a total disaster context like the type of the disaster, magnitude of the disaster, or location of the disaster. They are hence lacking in the in-depth analytic insights required to have a total comprehension of the prevailing crisis.

Research Gap

Despite significant advancements in the fields of AI, NLP, and geospatial analytics, existing disaster information systems face notable limitations. One of the primary challenges is their dependence on predefined datasets, which restricts the adaptability of AI models. These models often require labeled datasets for training, which can limit their ability to handle new disaster types or user queries with unfamiliar phrasing. This rigidity hinders the system's flexibility to respond dynamically to novel disaster scenarios or unanticipated questions.

Another main hurdle is the hardware and infrastructure dependency of IoT-based systems. Although such systems provide useful early warnings through their valuable addition to the predictive power of weather forecasts, they are unsuitable for broader global distribution because they are hardware and infrastructure-dependent. In low-resource and rural communities with restricted or non-existent IoT coverage, different non-infrastructure-intensive solutions are required.

Many existing systems also lack the ability to process free-form natural language queries, which severely limits their accessibility. For non-technical users, navigating complex dashboards or interfaces can be challenging. Most systems focus either on data visualization or sentiment analysis but fail to integrate these aspects into a unified, interactive platform that can provide a holistic view of a disaster event, including its type, severity, sentiment, and geographical location, all in one cohesive interface.

How the Project Fills the Gap

The Disaster Information Aggregation Dashboard is intended to overcome such gaps by providing a real-time AI system that offers dynamic on-demand information on disasters. The system takes advantage of GPT-4's zero-shot learning ability and question-based querying to answer natural language questions on any type of disaster without prior training or predetermined datasets. This makes the system very adaptable and capable of dealing with unfamiliar or newly occurring disaster scenarios and having a range of possible queries to understand and support a variety of ways to answer it without requiring fine-tuning or adaptation to specific tasks.

By retrieving data in real-time and thus bypassing the delays inherent in static databases and manual updates, the system makes the data presented always current and consistent with the most recent available disaster circumstances essential to informed decision-making in rapidly evolving disaster scenarios. The absence of a database also lessens the danger of outdated data being shown to users and makes the platform more responsive and timely.

The system consolidates multiple features into a unified interface by combining dynamic sentiment and statistical graphics (with Chart.js) with interactive geospatial mapping (utilising Leaflet.js). The unified interface enables users to gain a holistic perspective on disaster events by viewing location and impact of the event as well as public sentiments around it. The system's capability to map disaster data and sentiments in a single location provides a better representation than systems providing raw data or sentiment separately.

Additionally, the Disaster Information Aggregation Dashboard does not necessitate any physical infrastructure and is deployable on any web browser globally. This makes it possible to have the platform available in any region or area regardless of available infrastructure, hence a universally deployable tool for managing disasters.

As far as user accessibility goes, the dashboard is crafted to handle natural language questions freely, which means non-technical individuals can simply use and comprehend the data on the disasters. The system makes it simple to understand even unstructured user questions using advanced natural language techniques. The system is far more user-friendly compared to other systems that rely on technical skills to operate.

CHAPTER-4

PROPOSED METHADODOLOGY

The methodology proposed for the Disaster Information Aggregation Dashboard is intended to disseminate real-time AI-based disaster data by using novel technologies such as GPT-4, geospatial visualization, and sentiment analysis. The methodology makes certain that the platform presents reliable, real-time, and coherent insights on active disaster incidents and is user-friendly and responsive to different types of user queries. The methodology is organized into a number of main stages, which include data retrieval, natural language processing, dynamic visualization, and a user interface.

1. Data Fetching and Integration The central theme of the Disaster Information Aggregation Dashboard is fetching disaster data in real time. Rather than using predefined data sets or static databases, the system fetches the most up-to-date disaster information using APIs from real-time data sources on the web. The data concerns the type of the disaster (e.g. earthquake, flood, fire), the severity of the disaster, its location, the areas affected by it, and the state of the event.

The system fetches data dynamically to avoid the necessity of scheduled syncs or manual updates. The information displayed to users is up-to-date and mirrors the live changes in the condition of a disaster. The platform benefits from using APIs from a variety of sources like news agencies, disaster organizations, and public databases to tap into a variety of data. This provides users with a full understanding of the circumstances.

2. Natural Language Processing and Zero-Shot Learning: One of the innovations in the project is the employment of the zero-shot learning and natural language processing (NLP) abilities of GPT-4. The system may be queried using informal natural language by the users, without any requirement to formulate queries in a particular manner.

Users may pose questions like "What is the recent news about the earthquake in Nepal?" or "How serious is the flood in India?"

Utilising GPT-4's ability to respond to prompts in real time, the system is able to comprehend and respond to such queries accurately even in the absence of any prior training or datasets. The system is also adept at dealing with a range of queries, ranging from unplanned or new disaster incidents to unorthodox phrasing. The ability to adapt to varied types of queries enables the dashboard to return dynamic and appropriate responses to a large number of different user queries.

Zero-shot learning makes it possible to avoid having to fine-tune the system on a per-disaster basis. This makes it extremely scalable and adaptable and allows it to answer a range of different types of questions related to a disaster without the necessity to constantly retrain the model.

3. Geospatial Visualisation : To further improve the user experience, the Disaster Information Aggregation Dashboard features interactive geospatial visualization with Leaflet.js. The system enables users to look at disaster events on a clickable map and receive real-time information on the location and severity of the events. The map shows geospatial markers on affected locations like cities, towns, and regions and offers a solid, graphical impression of the impact of the disaster.

Along with highlighting the affected locations, the platform also enables zooming in and out of the map to look at a particular area or zoom out to gain a wider look at the scene. The interactive map is also updated in real-time as fresh disaster data is pulled in to maintain the up-to-the-minute and precise visualization. The geospatial visualization also facilitates better comprehension of the scale and magnitude of the disaster event by the end user and enables informed decision-making and required actions.

3. Sentiment Analysis : In addition to factual information on the catastrophe, the system also features a component of public sentiment assessment on the event. The system uses tools such as VADER to analyze the tone and sentiment in news and social media posts and compute real-time sentiment scores from the most recent content available.

The sentiment scores are represented on dynamic charts generated by Chart.js. The charts give an indication of overall public mood, i.e., if the public is worried, hopeful, or perplexed. Sentiment analysis may also reveal public levels of awareness and responses to the crisis and is helpful to the authorities and responders.

The sentiment analysis feature is integrated with the overall disaster data display, allowing users to view the disaster's severity, location, and public sentiment all in one place. This provides a more comprehensive understanding of the disaster, helping users assess not only the factual data but also the emotional and social context surrounding the event.

5. User Interface and Interaction Design: The user is at the center of the Disaster Information Aggregation Dashboard. The interface has a simple and intuitive layout and combines all the data and the visualizations into a cohesive experience. The interface makes it possible to respond to disaster data using a variety of different methods of input: free-form natural language queries, map interactions, and sentiment analysis charts.

The dashboard is completely responsive and available on any web browser, hence it is universally accessible from any location and on any device. There are no dependencies on hardware and therefore it has been implemented both in high-resource urban settlements and low-resource rural settings.

6. Real-Time Updates and Notifications: To further increase real-time functionality of the platform, the Disaster Information Aggregation Dashboard supports real-time updates and alerts. Users are instantaneously alerted when a fresh disaster event happens, or when dramatic changes are noted in the case of active disasters. Such alerts are made available through the platform's interface so that users are constantly aware of the most up-to-date developments.

Notifications are also configurable so that the user has the option to define preferences depending on the types of disasters they are interested in viewing, geographic location, or severity of concern. This makes it so users are getting the most relevant information at the best possible time so they are informed and may take required action as things are happening.

7. System Scalability and Adaptability The system is built to scale as it is capable of holding growing volumes of data as more incidents are recorded. The cloud-based infrastructure allows the platform to support growing volumes of data and user interactions without degrading performance. The platform also has a modular design to support simple integration of new data feeds, enhanced functions, or other tools related to disasters as they are developed. The methodology also provides flexibility so the system can adapt as disaster patterns and user requirements change. The platform becomes open to new data sources and emerging technologies at any time and is updated and enhanced accordingly to maintain and deliver real-time and precise insights to manage disasters.

The Main Advantages of APIs and Other Programs

1. APIs to Provide Real-Time Data: APIs are critical to retrieving real-time information about disasters from online sources like news websites, government bodies, and systems for reporting disasters. With APIs, the site is capable of delivering minute-by-minute updates regarding disasters without the necessity for human input to enter data or the usage of static databases.

APIs enable the dashboard to receive updated disaster data in real time, so the user always has the most recent information. APIs facilitate the aggregating of various data sources to provide a wide array of disaster information, enabling the platform to scale to handle an increasing number of data inputs. APIs also eliminate the human element of potentially entering incorrect data or missing data, which helps the platform to maintain reliability.

2. GPT-4 for Natural Language Processing: With GPT-4's NLP function, the site is capable of reading free-form user requests and delivering associated disaster data in real time. The zero-shot learning function is also capable of responding to novel or unknown queries without previous training, which is extremely useful for situations involving disasters where unforeseen circumstances may arise.

The feature enhances the system's accessibility since users are not required to have technical knowledge to use the system. They can just pose questions in natural language and get responses, a feature that makes the system available to many users, including laymen. The system is capable of processing multiple queries and adapting to unforeseen disaster events since the system is flexible in responding to them. Additionally, GPT-4's zero-shot learning feature does away with the requirement of predefined data or model-specific training for a specific task, lessening the requirement for continuous human intervention.

3. Real-Time Sentiment Analysis with VADER: Sentiment analysis is employed to measure the emotional reaction of the people to a calamity through the analysis of social media updates, news reports, and other textual information. VADER sentiment analysis software analyzes the data in real time to provide sentiment scores, which are visualized for the user in dynamic charts produced using Chart.js.

Sentiment analysis is also useful in revealing how the public is perceiving a disaster, enabling the concerned authorities to gauge the level of public awareness, public concerns, and the emotional tenor around the disaster. The input is continually updated with fresh data received, enabling decision-making to change strategy or communication based on shifting perceptions of the public. Sentiment analysis is supplemented with other types of disaster data, enabling these to provide a single interface overview of the severity of the disaster, where the disaster is occurring, and the public opinion,

4. Geospatial Visualization using Leaflet.js: The geospatial visualization aspect of the platform employs Leaflet.js to represent the disaster events in real time. The user is capable of visualizing the disaster data on an interactive map, getting information about affected area, severity levels, and geographical spread of the event.

Leaflet.js gives the user experience a boost through the provision of a graphic, intuitive representation of the disaster occurrences. The user can now zoom in from the map to focus on specific areas or pan out for an overview. The geospatial visualization assists the users in visualizing the scale and scope of the disaster in a better way, enabling them to decide based on reliable, real-time information.

5. Scalability and Cloud Infrastructure: The system is developed on a cloud infrastructure that supports seamless scalability with increasing data volumes and user activity. The cloud infrastructure ensures the system is capable of accommodating increasing data loads, while high availability and reliability are offered to users across various geographies.

Wait, The cloud infrastructure makes the platform scalable and reliable. It is capable of holding large amounts of disaster data, serving many real-time queries, and coping with high traffic for large-scale disaster occurrences. The cloud services also offer flexibility in the introduction of custom features, the incorporation of other data sources, as well as supporting more users without decreasing the performance level.

6. Cross-Platform Accessibility The Aggregation Dashboard for Disasters is available for viewing on any browser, so that it can run on many devices without the need to install special software or hardware. Cross-platform compatibility guarantees that the system can operate anywhere in the world, even where there are few resources available. The fact that the platform is accessible from any internet browser makes the platform universally accessible, enabling people from various settings and geographies (high-resource urban or low-resource rural settings) to receive information about disasters. The ease of the system ensures that the users can acquire critical information about a disaster from anywhere with an internet

CHAPTER-5

OBJECTIVES

The main objective of our projects are:

Provide Real-Time Disaster Information

The main aim of the project is to design a platform that provides real-time information regarding disasters. This shall cover the type of disaster (flood, earthquake, etc.), the severity of the disaster, the location that is affected, and the state the disaster is in right now. The data shall be automatically retrieved from authorized online sources so that the users are provided with the most updated information.

Facilitate Simple Interactions with Natural Language Requests

The platform will enable people to pose questions using natural language (as if communicating with a human) to receive information about disasters. Whether one is looking for the most recent information about an earthquake or the condition of a flood at a specific location, the system shall interpret the questions using cutting-edge AI (GPT-4) to reply with the right information without the input of predefined data or earlier training.

Provide Visual Disaster Mapping

The site will have an interactive map displaying the affected places due to disasters. With the help of tools like Leaflet.js, users can observe a visual overview of the spread of the disaster, along with the severity level, to understand the affected places. The map shall give the users a better sense of the geographical extent of the respective events in real time.

Analyzing Public Perception of Disasters

An important objective is also to offer information about how people feel about a disaster. The platform shall utilize sentiment tools such as VADER to scan public social media posts, news coverage, and other materials. It shall provide an indication of the public emotions and concerns about the disaster, enabling the authorities and the public at large to have a better understanding of public responses and needs.

Maintain Scalability and Flexibility

The platform would be developed on cloud technology, so the system is perfectly capable of processing large volumes of data and increasing users. With the emergence of new data sources or the platform increasing in size, the system is scalable without compromising on the level of performance or reliability. The platform is thus future-proofed and responsive to emerging needs.

Establish a Central Location for All Relevant Information

The initiative is to create an easy-to-use dashboard where anyone can have all the information about the disasters at their fingertips. The user can see the information about the disasters, interact with the map, read the sentiment analysis reports, and receive real-time updates without accessing multiple platforms or tools.

Individualized User Experience

The platform will enable users to customize their experience. They can specify their preference for the disasters they want to learn about the most, choose the places they wish to receive updates on, and receive updates or notifications according to their preferences. This way, the users receive only the most appropriate and timely information.

Provide Cross-Platform Accessibility

The platform can be operated on any internet-connected device, including smartphones, laptops, or tablets, and is supported on all modern web browsers. It is thus provided to a large number of users irrespective of where they are located or the device they have, so that individuals both in the cities and in the villages are provided with information on the disasters.

Promote Informed Choice for Disaster Response The platform will assist government agencies, response teams, and civilians in making improved choices in the midst of a disaster. With the accurate, real-time data provided, the platform will facilitate responses to disasters that are both faster and more informed, ultimately leading to lives saved and enhanced recovery efforts.

Promote awareness and readiness The project also seeks to enhance worldwide awareness regarding disasters so that individuals can prepare for potential future occurrences. By providing readily available information on the latest disasters, the platform will promote greater readiness and faster responses from citizens, governments, and relief groups. The main aim of the project is to design a platform that provides real-time information regarding disasters. This shall cover the type of disaster (flood, earthquake, etc.), the severity of the disaster, the location that is affected, and the state the disaster is in right now. The data shall be automatically retrieved from authorized online sources so that the users are provided with the most updated information.

Facilitate Simple Interactions with Natural Language Requests

The platform will enable people to pose questions using natural language (as if communicating with a human) to receive information about disasters. Whether one is looking for the most recent information about an earthquake or the condition of a flood at a specific location, the system shall interpret the questions using cutting-edge AI (GPT-4) to reply with the right information without the input of predefined data or earlier training.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

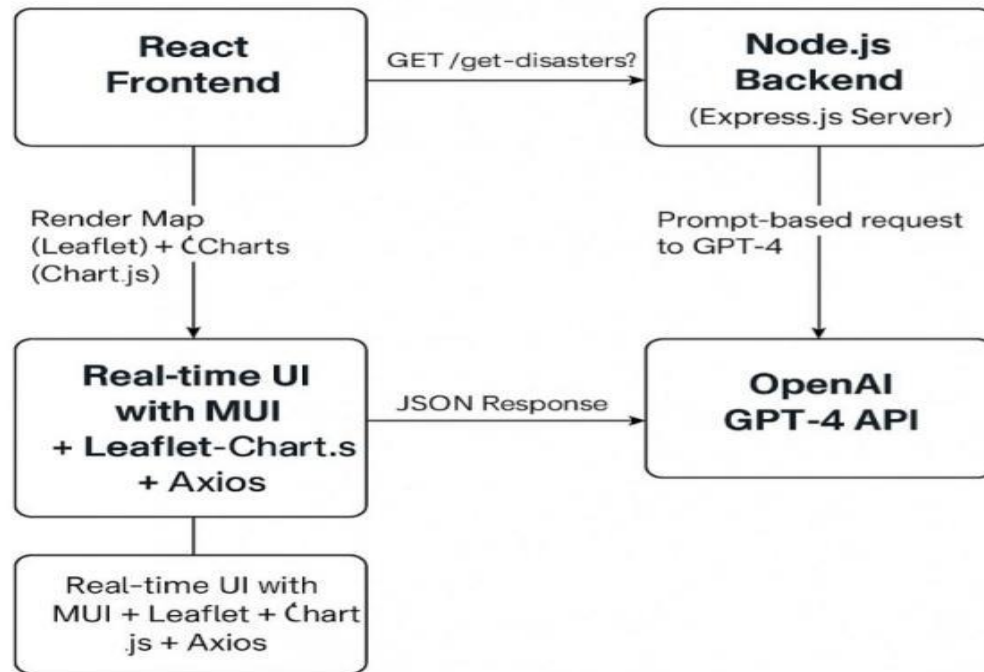


Fig 6.1.1

System Architecture

The architecture of the Disaster Information Aggregation Dashboard is based on the model of three tiers, comprising the frontend or user interface, the backend or server communication, and the OpenAI GPT-4 API or the AI model to process the queries. All these components are crucial in making the platform efficient and delivering real-time information on disasters.

1. Frontend (User Interface):

The frontend of the Disaster Information Aggregation Dashboard is developed using the React.js framework. This is where the user engages with the system to input natural language queries about the disasters. For instance, the user can pose questions like, “Where is the closest flood occurring?”

The frontend is capable of accepting queries from the user and receiving responses in real-time. The responses are presented using visualizations and maps that respond to user interaction. The system utilizes Leaflet.js to present geographical data about disasters on an interactive map while Chart.js is utilized to generate statistical visualizations that portray data regarding the disasters, including severity, impacted territories, among other data. This makes the system have an intuitive user interface that offers the user the information regarding the disasters in an understandable format.

2. Server and API Communication (Backend):

The backend for the Disaster Information Aggregation Dashboard is implemented using Node.js and Express.js. The backend serves to process the requests sent from the frontend, form suitable questions for the GPT-4 API provided by OpenAI, and process the responses from GPT-4.

When a user query is submitted, the backend formulates a well-structured prompt that captures the data to be used to retrieve the appropriate disaster data. The prompt is submitted to the OpenAI GPT-4 API, which translates the prompt and performs zero-shot learning to process the query. After GPT-4 has generated the response, the backend transforms and formats the information so that it is ready for displaying. The formatted information is then forwarded to the frontend for rendering.

3. OpenAI GPT-4 (Query Processing AI Model):

The foundation of the system is based upon OpenAI GPT-4, which utilizes zero-shot learning to both process and respond to queries for information concerning disasters. GPT-4 is not trained on specific data for disasters. Rather, it leverages its overall knowledge and natural language understanding to process the inquiry and return accurate, appropriate answers.

When the prompt is provided to GPT-4, the AI model responds based on the knowledge that is already present along with the inferencing capacity of the model. The system leverages the zero-shot learning approach wherein GPT-4 is able to respond with appropriate disaster knowledge based on general language understanding without specific task-oriented training or the use of predefined datasets. This makes the system adaptable and scalable to a large spectrum of queries regarding disasters.

4.Real-time data updates

To keep the system updated with the most recent information about the disasters, the architecture uses WebSockets or long polling. These technologies enable the frontend to have an open, real-time connection to the backend. When there is new data or an update, the frontend is able to show the user the most recent information about the disasters without the user needing to reload the page. This provides the user with the most recent and up-to-date data, which is extremely important for rapidly developing disasters.

Why This Building Works?

It is very effective for a number of reasons:

The system gives live information through the direct fetching of data from OpenAI GPT-4, so the information provided is always up to date and precise. It does away with static databases or schedule-driven updates, which are normally utilized in conventional systems of disaster management. The system is flexible and scalable since it does not draw from a predetermined dataset. With the application of zero-shot learning and prompt engineering, the system is capable of answering any form of query on disasters, independent of the wording or the kind of disaster.

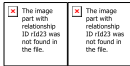
The user-friendly mapping interface developed using React.js, Leaflet.js, and Chart.js simplifies the entire user experience, enabling people to browse through and understand the data regarding the disasters.

The system utilizes WebSockets or long polling to keep users updated with the most recent disaster information at all times, even where data tends to change at high velocities in cases of emergencies.

Implementation in Your Project Within the project, this architecture is utilized in the following way: The frontend is developed using React.js for the user interface. It makes HTTP requests to the backend to send queries and render the responses. The backend, which is written using Node.js and Express.js, handles the process of queries and then communicates with the OpenAI GPT-4 API. The GPT-4 model, which is included through the backend, processes the prompts with the help of zero-shot learning to give real-time information on disasters. The frontend then visualizes the processed information through interactive maps provided using Leaflet.js and charts with the help of Chart.js. Real-time updates are provided using WebSockets or long polling so that the system gives recent data to the user whenever available. The architecture is designed to keep the system dynamic, flexible, and scalable while delivering reliable and timely information on disasters in an intuitive form that is easy to access.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



Task	Duration	Start Date	End Date
1. Project Planning	1 week	11-Jan	17-Jan
2. Backend Development	3 weeks	8-Feb	28-Feb
3. Frontend Development	3 weeks	13-Mar	2-Apr
4. Testing & QA	2 weeks	13-Apr	27-Apr
5. Delivery Deployment & Launch	1 week	5-May	11-May

Table 7.1.1

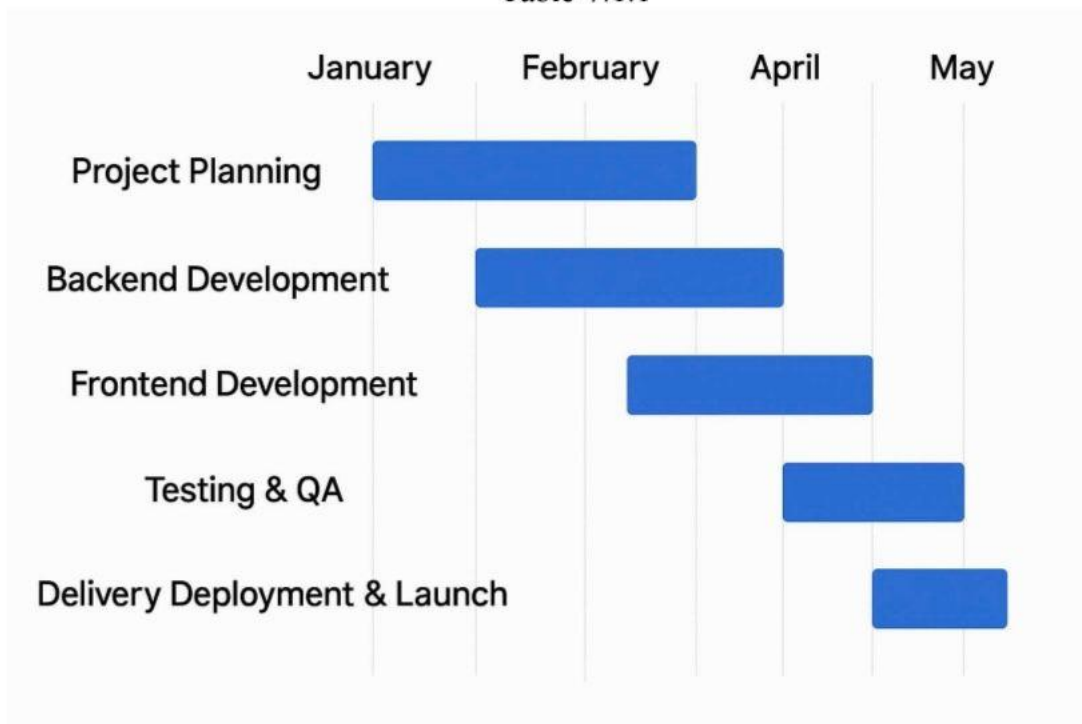


Table 7.1.1

CHAPTER-8

OUTCOMES

The Disaster Information Aggregation Dashboard is able to achieve the goal of delivering real-time, accurate, and thorough information regarding disasters using advanced AI, the GPT-4 model offered by OpenAI, and advanced natural language processing techniques. The project outcomes are manifold and cross several facets that include data accuracy, system efficiency, user experience, and scalability.

Real-time data processing

One of the major outputs of this project is the capacity of the system to offer real-time data about disasters without the use of conventional databases. The deployment of Prompt-based Querying and Zero-shot Learning guarantees that the system is capable of dynamic parsing of user queries and the provision of recent disaster data. Since every query retrieves fresh data directly from GPT-4, there is no latency incurred due to database synchronization or updates. The system is capable of supplying precise and timely data due to this real-time data process, which is critical in the case of a disaster where information has to be communicated at the earliest.

Reliable and Relevant Information about Disasters

The combination of GPT-4 with its high capability for Natural Language Understanding will enable the system to provide accurate, context-specific disaster data. From the question regarding the type, location, severity, or date of a natural disaster, GPT-4 is capable of perceiving the context of the query and responding with informative, well-formatted answers. This precision is critical in the management of disasters since the user can have confidence that the information offered is accurate as well as timely.

The Zero-shot Learning feature of GPT-4 also enhances the system's versatility to respond to queries regarding novel disaster incidents or worded differently. GPT-4 does not have to undergo re-training on specific domain data like other traditional models since the pre-trained knowledge of GPT-4 can answer a large variety of queries, making the system versatile for different situations.

Interactive data visualization

Another significant outcome is the development of an intuitive and interactive user interface for visualizing disaster data. Using Leaflet.js for map-based visualization and Chart.js for dynamic statistical charts, the system effectively presents data in a user-friendly manner. Users can view real-time disaster locations on the map, represented by geographical coordinates, and explore related statistics through interactive pie and bar charts. This graphical representation enhances the clarity and comprehensibility of the data, making it accessible to a broader audience, including non-experts in disaster management.

The sentiment chart, based on social media postings and news coverage of the disaster, offers a second level of understanding. The feature helps to measure the public mood toward in-progress disasters, potentially pinpointing an area of public concern or public interest in the midst of critical events. This feature adds an aspect of considering both factual data along with emotional perspective to the understanding of the situation of a disaster.

CHAPTER-9

RESULTS AND DISCUSSIONS

The Disaster Information Aggregation Dashboard has been extremely effective in handling real-time queries for disasters due to the implementation of GPT-4 along with advanced Natural Language Processing techniques. The system is capable of identifying the incidents that are currently happening, aggregating the data in an easy-to-use format, and displaying the vital information like the severity of the disasters, the location, and the distribution of the events without the support of a conventional database. The methodology helps keep the data up to date at all times, making the system extremely responsive and adaptable.

The crux of the system is based on two primary techniques: Zero-shot Learning and Prompt-based Querying both of which are driven by GPT-4. These two techniques complement each other perfectly to enable the system to grasp the user queries' context and respond with accurate, satisfactory responses irrespective of how new or specific the query is.

Prompt-Based Querying is about issuing properly framed prompts to GPT-4 to instruct it to generate particular forms of information. Upon input of a query from the user, the backend, based on Node.js and Express.js, passes on the prompt to GPT-4. The prompt may demand extensive information about ongoing disasters like the nature of the disaster, location, date, severity, and the corresponding latitude and longitude. GPT-4 process the query and responds in the specific format requested, supplying organized and formatted information regarding the disasters.

Aside from prompt-based questioning, the system also utilizes Zero-shot Learning. This method enables GPT-4 to respond to questions that have not been specially trained for. Since GPT-4 has been pre-trained on a massive dataset, it can generalize to respond to a large variety of questions on disasters without needing to undergo task-specific training. This makes the system highly versatile, capable of delivering precise answers to both common and unforeseen disaster situations, even where the questions are worded in unexpected ways.

GPT-4, founded on a transformer model, relies on mechanisms like self-attention, position encoding, and dense layers to learn the word relations and generate meaningful sentences. It produces responses sequentially with the next word predicted based on the previous context, which is important to guaranteeing accurate and coherent responses to disaster queries. The system's natural language understanding (NLU) capabilities enable it to recognize the type, location, and severity of the disaster without explicit training on these features.

The outputs of these processes are presented through an easy-to-use user interface developed using Leaflet.js and React.js. The dashboard visualizes the data of disasters on an interactive map based on the latitude and longitude coordinates provided by GPT-4 for marking the disasters on the map. Chart.js is also utilized to generate graphical representations of the data, including bar charts and pie charts, to represent the occurrences, severity, and spread of the disasters. A sentiment analysis chart based on the textual reports of the disaster is also provided to represent the emotional tone of the events, as indicated in the sentiment analysis flow

One of the most significant strengths of the system is that it does not store data in a standard database. Rather, every user query initiates a new, real-time call to GPT-4, so that the information presented is always up to date. This has the advantage of giving

real-time data but also minimizes privacy and security issues since data is not permanently retained. Frontend to backend communication is safely controlled using technologies like Axios and CORS.

In short, through the combination of Prompt Engineering, Zero-shot Learning, and Transformer-based Natural Language Understanding, the system produces very accurate real-time disaster intelligence. The architecture is efficient, lightweight, and state-of-the-art, rendering the system a potent tool for information aggregation and disaster monitoring.

CHAPTER-10

CONCLUSION

The Disaster Information Aggregation Dashboard is a contemporary and efficient way of real-time disaster tracking and information delivery. By utilising state-of-the-art technologies like OpenAI GPT-4, Prompt-based Querying, Zero-shot Learning, and Natural Language Understanding (NLU), the system can scan the internet for information on disasters without the need for any traditional databases. Like this, the system not only gains in terms of speed and scalability but also guarantees that the information reached is always up to date and right according to context.

The use of a React.js frontend and a Node.js + Express.js backend provides the user with an effortless interface to the platform. Sophisticated visualization solutions such as Leaflet.js and Chart.js give the user understanding of the events of the disaster in the form of maps, graphs, and sentiment analysis. The entire architecture facilitates dynamic data flow from the user queries to the visualization layer in real time.

By employing secure protocols and dynamic questioning, data privacy is ensured while maintaining high responsiveness. The dynamic adaptability of GPT-4 in dealing with novel queries through Zero-shot Learning ensures the dashboard is capable of accommodating novel disasters and changing information requirements without the need for domain-specific training.

In summary, the project illustrates the way that prompt engineering and artificial intelligence can be utilized effectively to create scalable, intelligent, and user-centric solutions for the management of information in disasters. It lays the groundwork for the future advancements in AI-supported systems for disaster response and has high potential for incorporation with international early warning systems and global platforms for emergency management.

REFERENCES

- [1] R. Kumbam and K. M. Vejre, "FloodLense: A Framework for ChatGPT-based Real-time Flood Detection," arXiv, 2024.
- [2] D. Le, "Disaster Tweets Classification using BERT- Based Language Model," arXiv, 2022.
- [3] A. K. Tripathy, S. K. Sahu, and A. K. Turuk, "An efficient and secure data sharing scheme for cloud-based Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 197–204, Jan. 2021,
- [4] H. N. Saha, S. Auddy, S. Pal, S. Kumar, S. Pandey, and R. Singh, "Disaster management using Internet of Things," in *Proc. 2017 8th Annu. Ind. Autom. Electromech. Eng. Conf. (IEMECON)*, Bangkok, Thailand, Aug. 2017, pp.
- [5] V. Mody, V. Mody, and S. Parekh, "Distress – An application for emergency response and disaster management," in *Proc. 2020 Int. Conf. Smart Electron. Commun. (ICOSEC)*, Trichy, India, Sep. 2020, pp.
- [6] K. Chanda, "Efficacy of BERT embeddings on predicting disaster from Twitter data," arXiv, 2021.
- [7] R. S. Rajasekharan, R. Bestak, and G. Dusserre, "A study on disaster management databases," in *Proc. 2021 Int. Conf. Military Technol. (ICMT)*, Brno, Czech Republic, Jun. 2021.
- [8] Patel, P. Bhattacharjee, A. Reza, and P. Pradhan, "Earthquake Response Analysis with AI," arXiv, 2025.
- [9] Bass, P. Clements, and R. Kazman, "Software Architecture in Practice," 2nd ed., Pearson, 2003.
- [10] T. J. Kalidoss and S. Ravi, "Disaster management system leveraging the emerging digital technologies," in *Proc. 2016 IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Chennai, India, Dec. 2016.

APPENDIX-A

PSUEDOCODE

BACKEND:

```
// Load environment variables from .env file
require('dotenv').config();

// Import required modules
const express = require('express');
const cors = require('cors');
const OpenAI = require('openai');

// Create an Express app instance
const app = express();

// Define server port
const PORT = 5000;

// Setup OpenAI with API Key
const openai = new OpenAI({
  apiKey: process.env.OPENAI_API_KEY,
});

// Enable JSON request parsing
app.use(express.json());
```

```
// Enable Cross-Origin Resource Sharing
```

```
app.use(cors());
```

```
const mockDisasterData = [
```

```
{
```

```
  type: "Earthquake",
```

```
  location: "San Francisco, USA",
```

```
  date: "2025-04-15",
```

```
  severity: 6,
```

```
  latitude: 37.7749,
```

```
  longitude: -122.4194,
```

```
  description: "Moderate earthquake causing minor  
structural damage"
```

```
},
```

```
{
```

```
  type: "Flood",
```

```
  location: "Dhaka, Bangladesh",
```

```
  date: "2025-04-10",
```

```
  severity: 8,
```

```
  latitude: 23.8103,
```

```
  longitude: 90.4125,
```

```
  description: "Monsoon rains caused severe flooding  
in low-lying areas"
```

```
}
```

```
];
```

```
app.get('/get-disasters', async (req, res) => {
```

```
  try {
```

```
    // Step 1: Get query or use default
```

```
    const query = req.query.query || "all disasters";
```

```
    console.log(`❏ Fetching disasters related to:  
"${query}"`);
```

```
// Step 2: Prepare GPT prompt

const prompt = `Provide a list of 5 recent disasters
from the last 2 weeks worldwide.

Return ONLY a valid JSON array without any
extra text or explanations.

Format strictly as:

[
  {
    "type": "Disaster Type",
    "location": "City, Country",
    "date": "YYYY-MM-DD",
    "severity": 1-10,
    "latitude": Decimal,
    "longitude": Decimal,
    "description": "Short summary (20-30 words)"
  }
];

// Step 3: Send request to GPT-4

const response = await
openai.chat.completions.create({
  model: 'gpt-4',
  messages: [
    { role: 'system', content: 'You are a helpful
assistant that outputs JSON.' },
    { role: 'user', content: prompt }
  ],
  max_tokens: 1000,
  temperature: 0.3
});

// Step 4: Get raw response from GPT
```

```

    const gptContent =
response.choices[0].message.content.trim();

    console.log("❑ Raw GPT response:\n",
gptContent);

let data;

try {

    // Step 5: Try parsing GPT response to JSON

    data = JSON.parse(gptContent);


    // Step 6: If it's a JSON object, try to extract
array

    if (data && typeof data === 'object' &&
!Array.isArray(data)) {

        const arrayKey = Object.keys(data).find(key
=> Array.isArray(data[key]));

        if (arrayKey) {

            data = data[arrayKey];

        }

    }


    // Step 7: Check if valid array, else throw error

    if (!Array.isArray(data)) {

        throw new Error("Response did not contain a
valid array");

    }


} catch (parseError) {

    console.error("❌JSON parsing failed:",
parseError.message);

    console.warn("❑ Falling back to mock data");

    data = mockDisasterData;

}

```

```

// Step 8: Final check — if no data, use fallback
if (!data || data.length === 0) {
  console.warn("❑ No disaster data found, using
mock data");
  data = mockDisasterData;
}
// Step 9: Send response to frontend
res.json(data);

} catch (error) {
  console.error('✖Error fetching disasters:', error);
  console.warn("❑ Falling back to mock data due to
error");
  res.json(mockDisasterData);
}
});
app.listen(PORT, () => {
  console.log(`✔Server running on
http://localhost:${PORT}`);
});

```

FRONTEND:

```

import React, { useEffect, useState } from 'react';
import axios from 'axios';
import {
  MapContainer, TileLayer, Marker, Popup
} from 'react-leaflet';
import {
  Box, Typography, Card, CardContent, Grid, AppBar, Toolbar, Table, TableHead, TableRow,
  TableCell, TableBody, Paper, LinearProgress, Chip, Divider, Tabs, Tab, CssBaseline
} from '@mui/material';
import { BarChart, Bar, XAxis, YAxis, Tooltip, ResponsiveContainer, PieChart, Pie, Cell } from

```

```
'recharts';

import 'leaflet/dist/leaflet.css';

const App = () => {
  const [disasters, setDisasters] = useState([]);

  const [loading, setLoading] = useState(true);
  const [tabValue, setTabValue] = useState(0);

  useEffect(() => {
    fetchDisasters();
  }, []);

  const fetchDisasters = async () => {
    try {
      setLoading(true);
      const response = await axios.get('http://localhost:5000/get-disasters');
      const disasterData = Array.isArray(response.data) ? response.data : [];
      setDisasters(disasterData);
    } catch (error) {
      console.error('Error fetching data:', error);
    } finally {
      setLoading(false);
    }
  };

  const disasterCountsByType = disasters.reduce((acc, disaster) => {
    acc[disaster.type] = (acc[disaster.type] || 0) + 1;
    return acc;
  }, {});

  const barChartData = Object.entries(disasterCountsByType).map(([type, count]) => ({
    type,
```

```

    count
  }));

const pieChartData = Object.entries(disasterCountsByType).map(([type, count]) => ({
  name: type,
  value: count
}));

const COLORS = ['#0088FE', '#00C49F', '#FFBB28', '#FF8042', '#8884D8', '#82ca9d', '#a4de6c',
'#d0ed57'];

const severityColor = (severity) => {
  if (severity >= 8) return 'error';
  if (severity >= 5) return 'warning';
  return 'success';
};

const handleTabChange = (event, newValue) => {
  setTabValue(newValue);
};

return (

<Box sx={{ minHeight: '100vh', bgcolor: 'background.default' }}>
  <CssBaseline />
  <AppBar position="static">
    <Toolbar>
      <Typography variant="h6" sx={{ flexGrow: 1 }}>
        Disaster Dashboard
      </Typography>
    </Toolbar>
  </AppBar>

```

```

{loading ? (
  <LinearProgress color="secondary" />
): (
  <Box sx={{ p: 2 }}>
    { /* Map Section */ }
    <Card sx={{ mb: 3, height: '50vh' }}>
      <CardContent sx={{ height: '100%', p: 0 }}>
        <MapContainer
          center={[20, 0]}
          zoom={2}
          style={{ height: '100%', width: '100%' }}
        >
          <TileLayer
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
            attribution='&copy; OpenStreetMap contributors'
          />
          {disasters.map((disaster, index) => (
            <Marker
              key={` ${disaster.type} - ${disaster.location} - ${index} `}
              position={[parseFloat(disaster.latitude), parseFloat(disaster.longitude)]}
            >
              <Popup>
                <Typography variant="subtitle1" sx={{ fontWeight: 'bold' }}>
                  {disaster.type}
                </Typography>
                <Chip
                  label={`Severity: ${disaster.severity}/10`}
                  color={severityColor(disaster.severity)}
                  size="small"
                  sx={{ mb: 1 }}
                />
                <Typography variant="body2">
                  <strong>Location:</strong> {disaster.location}
                </Typography>
              </Popup>
            </Marker>
          ))}
        </TileLayer>
      </CardContent>
    </Card>
  </Box>
)

```

```

        <Typography variant="body2">
            <strong>Date:</strong> {disaster.date}
        </Typography>
    </Popup>
</Marker>

)}}

    </MapContainer>
</CardContent>
</Card>

{ /* Summary Cards */ }
<Grid container spacing={2} sx={{ mb: 3 }}>
    <Grid item xs={12} sm={6} md={3}>
        <Card>
            <CardContent>
                <Typography color="textSecondary">Total Disasters</Typography>
                <Typography variant="h5">{disasters.length}</Typography>
            </CardContent>
        </Card>
    </Grid>
    <Grid item xs={12} sm={6} md={3}>
        <Card>
            <CardContent>
                <Typography color="textSecondary">Disaster Types</Typography>
                <Typography variant="h5">{Object.keys(disasterCountsByType).length}</Typography>
            </CardContent>
        </Card>
    </Grid>
    <Grid item xs={12} sm={6} md={3}>
        <Card>
            <CardContent>
                <Typography color="textSecondary">Avg Severity</Typography>

```

```

    <Typography variant="h5">
      {(disasters.reduce((sum, d) => sum + d.severity, 0) / disasters.length || 0).toFixed(1)}
    </Typography>
  </CardContent>
</Card>
</Grid>
<Grid item xs={12} sm={6} md={3}>
  <Card>
    <CardContent>
      <Typography color="textSecondary">Last Updated</Typography>
      <Typography variant="h5">{new Date().toLocaleDateString()}</Typography>
    </CardContent>
  </Card>
</Grid>
</Grid>

{/* Tab Section */}
<Card>
  <Tabs value={tabValue} onChange={handleTabChange} variant="fullWidth">
    <Tab label="Data Visualization" />
    <Tab label="Disaster List" />
    <Tab label="Sentiment Analysis" />
  </Tabs>
  <Divider />
  <CardContent>
    {/* Tab 1: Charts */}
    {tabValue === 0 && (
      <Grid container spacing={3}>
        <Grid item xs={12} md={6}>
          <Typography variant="h6" gutterBottom>Disaster Distribution</Typography>
          <ResponsiveContainer width="100%" height={300}>
            <PieChart>
              <Pie
                data={pieChartData}

```

```

    cx="50%"
    cy="50%"
    labelLine={ false}
    outerRadius={ 80}
    fill="#8884d8"
    dataKey="value"
    label={({ name, percent }) => `${name} ${(percent * 100).toFixed(0)}%`}
  >
    {pieChartData.map((entry, index) => (
      <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
    ))}
  </Pie>
  <Tooltip />
</PieChart>
</ResponsiveContainer>
</Grid>
<Grid item xs={ 12} md={ 6}>
  <Typography variant="h6" gutterBottom>Disaster Count by Type</Typography>
  <ResponsiveContainer width="100%" height={ 300}>
    <BarChart data={barChartData}>
      <XAxis dataKey="type" />
      <YAxis />
      <Tooltip />
      <Bar dataKey="count" fill="#1976d2" radius={[4, 4, 0, 0]} />
    </BarChart>
  </ResponsiveContainer>
</Grid>
</Grid>
)}

{/* Tab 2: Disaster Table */}
{tabValue === 1 && (
  <Paper sx={{ overflowX: 'auto' }}>
    <Table>

```

```

<TableHead>
  <TableRow>
    <TableCell>Type</TableCell>
    <TableCell>Location</TableCell>
    <TableCell>Date</TableCell>
    <TableCell>Severity</TableCell>
    <TableCell>Description</TableCell>
  </TableRow>
</TableHead>
<TableBody>
  {disasters.map((disaster, index) => (
    <TableRow key={` ${disaster.type} - ${index}`} hover>
      <TableCell>{disaster.type}</TableCell>
      <TableCell>{disaster.location}</TableCell>
      <TableCell>{disaster.date}</TableCell>
      <TableCell>
        <Chip
          label={` ${disaster.severity}/10` }
          color={severityColor(disaster.severity)}
          size="small"
        />
      </TableCell>
      <TableCell>{disaster.description}</TableCell>
    </TableRow>
  ) )}
</TableBody>
</Table>
</Paper>
)}

```

```

{/* Tab 3: Sentiment Analysis */}
{tabValue === 2 && (
  <Paper sx={{ overflowX: 'auto' }}>
    <Table>

```

```

    <TableHead>
      <TableRow>
        <TableCell>Type</TableCell>
        <TableCell>Location</TableCell>
        <TableCell>Sentiment</TableCell>
      </TableRow>
    </TableHead>
    <TableBody>
      {disasters.map((disaster, index) => (
        <TableRow key={`sentiment-${index}`} hover>
          <TableCell>{disaster.type}</TableCell>
          <TableCell>{disaster.location}</TableCell>
          <TableCell>{disaster.sentiment || 'N/A'}</TableCell>
        </TableRow>
      ))}
    </TableBody>
  </Table>
</Paper>
)}
</CardContent>
</Card>
</Box>
)}
</Box>
);
};

export default App;

```

APPENDIX-B

SCREENSHOTS OF THE OUTPUT

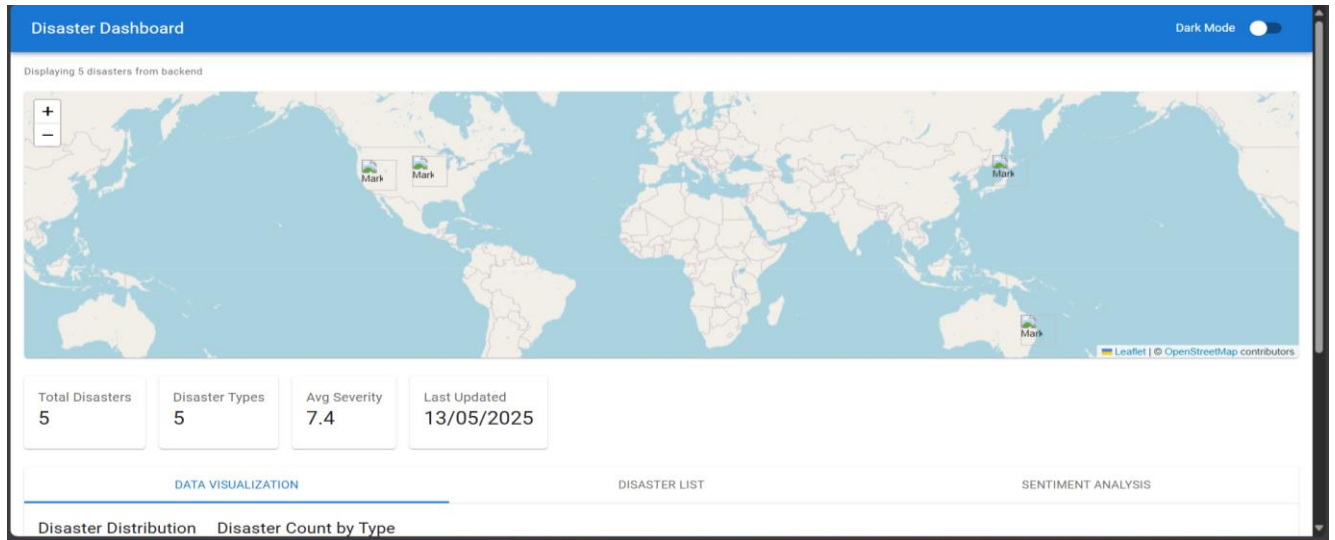


Fig1.1 Disaster represented on world map

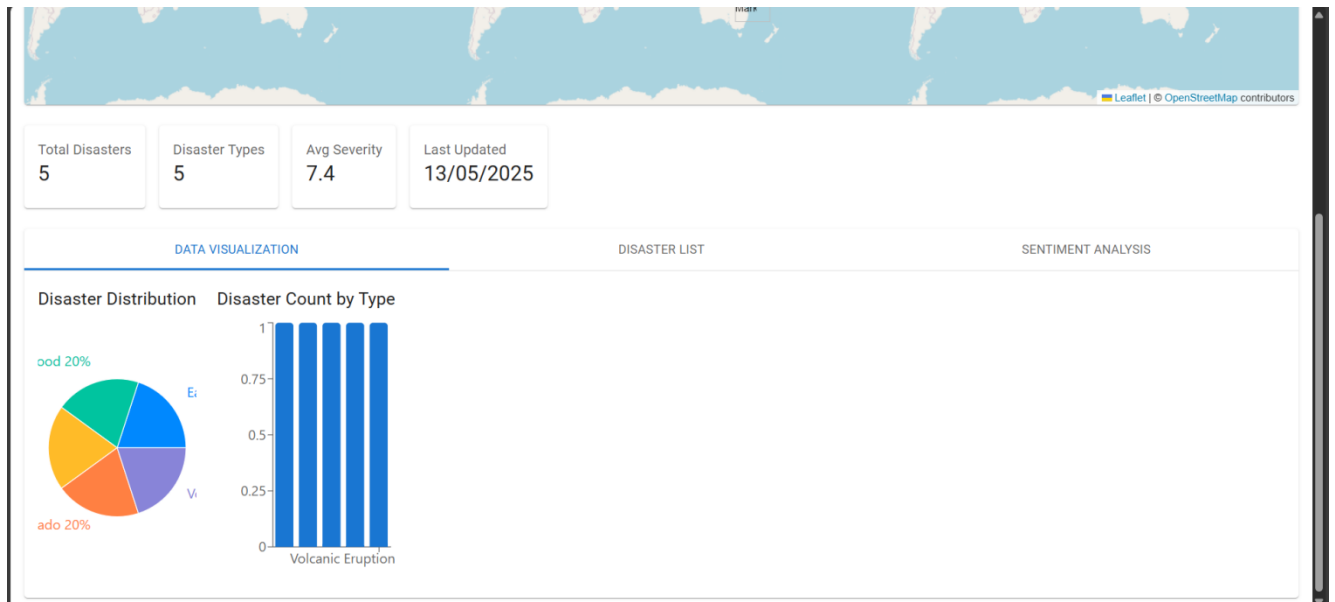


Fig1.2 Pie chart and Bar graph representation

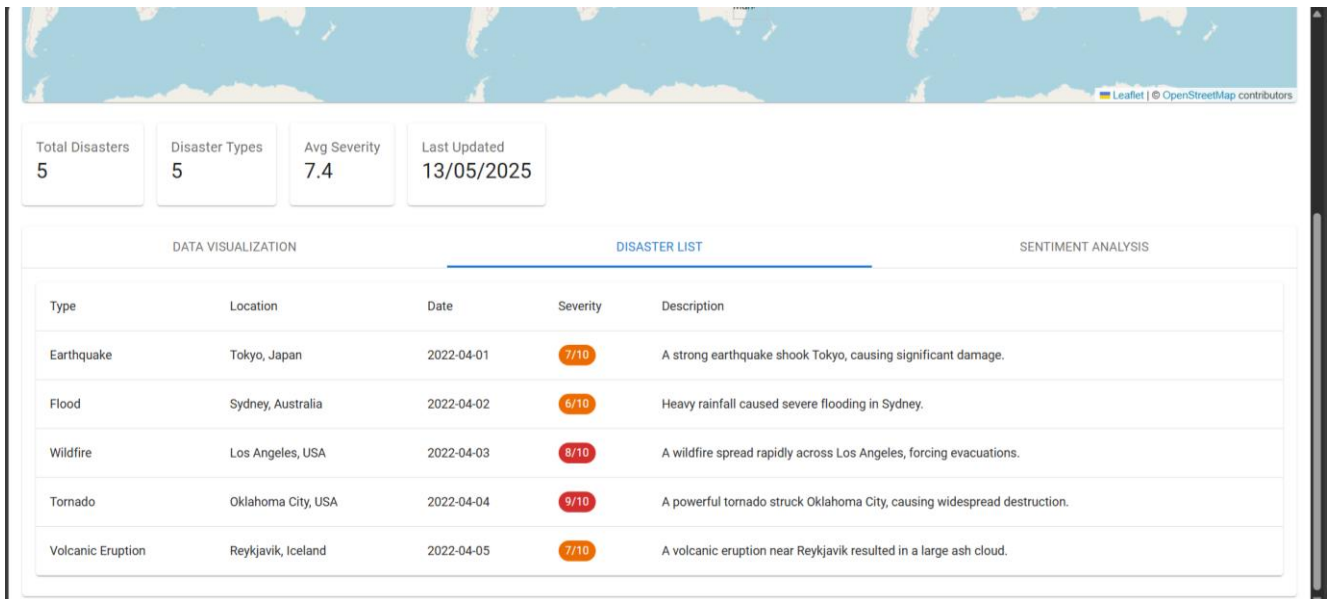


Fig1.3 Disaster types

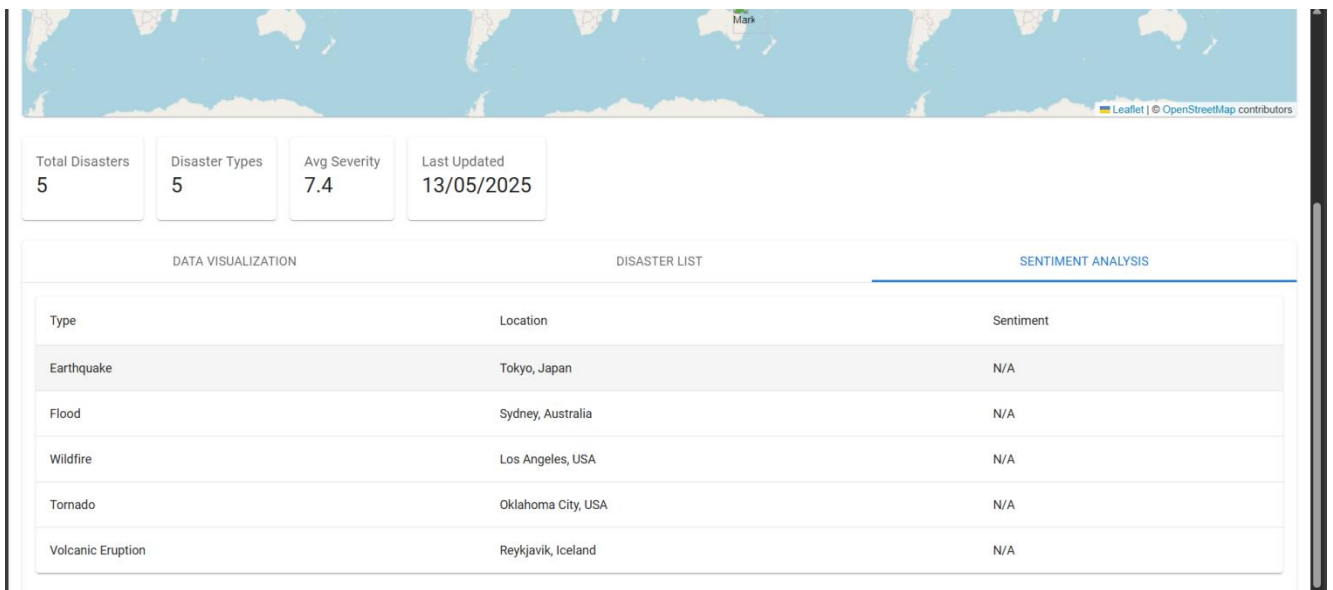
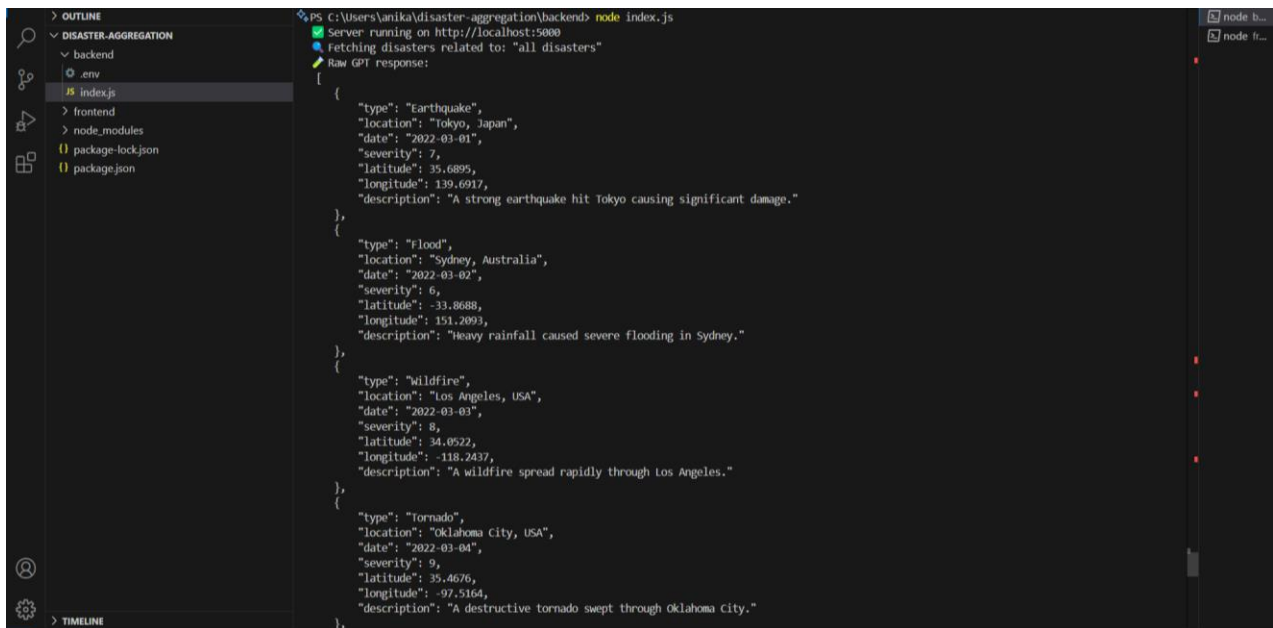


Fig1.4 Location where disaster has taken place



```
PS C:\Users\anika\disaster-aggregation\backend> node index.js
Server running on http://localhost:5000
Fetching disasters related to: "all disasters"
Raw GPT response:
[
  {
    "type": "Earthquake",
    "location": "Tokyo, Japan",
    "date": "2022-03-01",
    "severity": 7,
    "latitude": 35.6895,
    "longitude": 139.6917,
    "description": "A strong earthquake hit Tokyo causing significant damage."
  },
  {
    "type": "Flood",
    "location": "Sydney, Australia",
    "date": "2022-03-02",
    "severity": 6,
    "latitude": -33.8688,
    "longitude": 151.2093,
    "description": "Heavy rainfall caused severe flooding in Sydney."
  },
  {
    "type": "Wildfire",
    "location": "Los Angeles, USA",
    "date": "2022-03-03",
    "severity": 8,
    "latitude": 34.0522,
    "longitude": -118.2437,
    "description": "A wildfire spread rapidly through Los Angeles."
  },
  {
    "type": "Tornado",
    "location": "Oklahoma City, USA",
    "date": "2022-03-04",
    "severity": 9,
    "latitude": 35.4676,
    "longitude": -97.5164,
    "description": "A destructive tornado swept through Oklahoma City."
  }
],
```


Fig 1.5 Latitude and Longitude for better geoloaction

APPENDIX-C

ENCLOSURES

- 1. Journal publication/Conference Paper Presented Certificates of all students.**

2. Similarity Index / Plagiarism Check report.

Page 2 of 68 - Integrity OverviewSubmission ID trn:oid::1:3248710330





14% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- Bibliography

Match Groups

-  **49 Not Cited or Quoted 14%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **2 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 11%  Internet sources
- 5%  Publications
- 12%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

3. Details of mapping the project with the Sustainable Development Goals (SDGs).



SDG 13 – Climate Action

Relevance: Our project leverages cutting-edge technologies like GIS, sentiment analysis, and AI to visualize disaster data in real-time. It promotes innovation in disaster risk monitoring and communication. This supports resilient infrastructure and informed decision-making for emergency planning.

SDG 9 – Industry, Innovation and Infrastructure

Relevance: The dashboard strengthens adaptive capacity by tracking climate-related hazards such as floods, cyclones, and wildfires. It empowers authorities and communities with timely information for proactive response. This contributes directly to climate resilience and disaster risk reduction.
