INDUSTRIAL ORIENTED MINI PROJECT

Report

On

AI BASED HUMANITARIAN AID DISTRIBUTIOIN OPTIMISATION

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

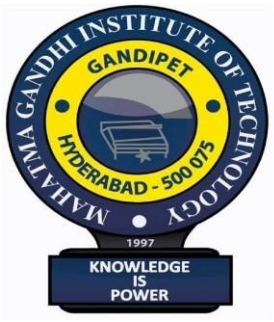
Junuthula Siri Lasya - 22261A1227

T.Sambhav - 22261A1251

Under the guidance of

Dr. B.Harika

Associate Professor, Department of IT



DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (AUTONOMOUS)

(Affiliated to JNTUH, Hyderabad; Eight UG Programs Accredited by NBA; Accredited by NAAC with 'A++' Grade)

Gandipet, Hyderabad, Telangana, Chaitanya Bharati (P.O), Ranga Reddy District, Hyderabad– 500075, Telangana

2024-2025

CERTIFICATE

This is to certify that the **Industrial Oriented Mini Project** entitled **AI BASED HUMANITARIAN AID DISTRIBUTIOIN OPTIMISATION** submitted by **Junuthula Siri Lasya (22261A1227), T.Sambhav (22261A1251)** in partial fulfillment of the requirements for the Award of the Degree of Bachelor of Technology in Information Technology as specialization is a record of the bona fide work carried out under the supervision of **Dr. B.Harika**, and this has not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Supervisor:

Dr. B.Harika
Associate Professor
Dept. of IT

IOMP Supervisor:

Dr. U. Chaitanya
Assistant Professor
Dept. of IT

EXTERNAL EXAMINAR

Dr. D. Vijaya Lakshmi
Professor and HOD
Dept. of IT

DECLARATION

We hear by declare that the **Industrial Oriented Mini Project** entitled **AI BASED HUMANITARIAN AID DISTRIBUTIOIN OPTIMISATION** is an original and bona fide work carried out by us as a part of fulfilment of Bachelor of Technology in Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of **Dr.B.Harika , Associate Professor**, Department of IT, MGIT.

No part of the project work is copied from books /journals/ internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other source.

Junuthula Siri Lasya - 22261A1227

T.Sambhav- 22261A1251

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the **Industrial Oriented Mini Project**.

We would like to express our sincere gratitude and indebtedness to our project guide **Dr. B.Harika , Associate Professor**, Dept. of IT, who has supported us throughout our project with immense patience and expertise.

We are also thankful to our honourable Principal of MGIT **Prof. G. Chandramohan Reddy** and **Dr. D. Vijaya Lakshmi**, Professor and HOD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this **Industrial Oriented Mini Project** successfully.

We are also extremely thankful to our Project Coordinator(s)**Dr. U. Chaitanya**, Assistant Professor, Department of IT, and senior faculty **Mrs.B.Meenakshi** Department of IT for their valuable suggestions and guidance throughout the course of this project.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support for completion of this work.

Junuthula Siri Lasya - 22261A1227

T.Sambhav- 22261A1251

ABSTRACT

An AI-based Humanitarian Aid Distribution System significantly improves crisis response by using cutting-edge technologies like real-time data analysis, predictive modeling, and automated decision-making. Unlike traditional systems that often face delays, resource mismanagement, and poor coordination, AI systems bring speed, precision, and adaptability to aid operations. This is especially crucial during natural disasters, conflicts, or pandemics where every second counts, and the needs of affected populations change rapidly.

For instance, by analyzing weather data, population density, terrain, and transportation networks, AI can help predict where the crisis might escalate next and what resources will be required there. These insights help in proactively deploying aid, preventing bottlenecks, and ensuring timely support.

Furthermore, AI optimizes resource allocation and routing by assigning urgency scores based on severity, vulnerability, and accessibility. Intelligent routing systems prioritize high-need areas, reducing wastage and ensuring that the most affected communities receive help first. These systems can also adjust routes in real time based on roadblocks, weather disruptions, or updated threat levels, making logistics more efficient and flexible compared to rigid manual plans.

Finally, AI-driven communication platforms enable better collaboration among stakeholders such as governments, NGOs, and international relief agencies. Shared dashboards, alert systems, and predictive tools create a unified response ecosystem where decisions are based on consistent, updated information. This reduces duplication of efforts, speeds up deployment, and ensures that all players are aligned in their mission to deliver life-saving aid efficiently and equitably.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 MOTIVATION

Humanitarian aid plays a vital role in addressing the urgent needs of populations affected by natural disasters, conflicts, and other emergencies. However, traditional relief operations often suffer from significant challenges such as delayed responses, inefficient resource allocation, and lack of coordination among agencies. These issues can result in critical gaps in aid delivery, leaving vulnerable communities without timely support. In rapidly evolving crisis scenarios, manual methods are often too slow and rigid to respond effectively, highlighting the urgent need for a smarter, more adaptive solution.

This project proposes the development of an AI-Based Humanitarian Aid Distribution System that integrates real-time data analysis, predictive modeling, and intelligent decision-making. The system will gather and analyze data on disaster severity, population density, infrastructure status, and resource availability from various sources such as satellite imagery, government databases, and humanitarian networks. Using machine learning algorithms, it will dynamically assess ongoing crisis conditions, forecast demand for relief supplies, and generate optimized distribution strategies. These models will continuously learn from historical crisis patterns to improve the accuracy and relevance of future predictions.

By providing actionable insights in real time, the AI system will empower governments, NGOs, and aid organizations to allocate resources more efficiently and prioritize areas with the highest need. Automated logistics planning will reduce delays and minimize waste, while intelligent routing will ensure that supplies reach remote or hard-to-access regions quickly. Ultimately, this project aims to create a scalable and sustainable AI-powered infrastructure that transforms humanitarian aid delivery into a faster, smarter, and more coordinated effort—making future crisis management more resilient and effective.

1.2 PROBLEM STATEMENT

In crisis zones, humanitarian aid distribution often faces serious challenges such as delays, resource misallocation, inefficiency, and difficulty in reaching remote or severely affected areas. Traditional aid systems rely on manual decision-making and limited data, making them slow and inadequate, especially in rapidly evolving situations. The lack of real-time

monitoring and predictive insights often results in poor coordination and mismanagement of vital resources, reducing the overall effectiveness of relief efforts.

To address these issues, an AI-based Humanitarian Aid Distribution System is proposed, designed to enhance the speed and accuracy of aid delivery. This system would leverage real-time data from sources like satellite imagery, weather reports, and on-ground sensors. Machine learning models would analyze this data to assess disaster severity, predict aid demand, and prioritize distribution. Geospatial analysis and route optimization would help in planning efficient logistics and reaching affected areas quickly and effectively.

Automation and AI-driven decisions would reduce human error and ensure resources are directed where they are needed most. This system would not only enhance operational efficiency but also promote transparency and accountability, offering a smarter, faster, and more coordinated approach to humanitarian crisis management.

1.3 EXISTING SYSTEM

The existing system of humanitarian aid distribution primarily depends on traditional, manual methods of planning and decision-making. Aid is typically allocated based on fixed criteria such as population counts, outdated census data, or localized reports from field personnel. While these inputs provide a basic understanding of the situation, they often lack the depth and speed required to respond effectively in dynamic and rapidly evolving crisis environments. As a result, relief efforts are frequently delayed, resources may not reach the most affected areas on time, and aid distribution becomes inconsistent or misaligned with actual needs on the ground.

One of the major limitations of this system is its reactive nature. Aid distribution decisions are made after a crisis has escalated, rather than anticipating needs based on early indicators. This approach makes it difficult to deploy resources proactively, especially in complex situations like widespread natural disasters or conflict zones where conditions can change within hours. Moreover, the absence of real-time monitoring tools means that decision-makers cannot track the progress or effectiveness of ongoing relief efforts, nor can they quickly adjust plans as new challenges emerge. As a result, resources are often wasted or underutilized, and many vulnerable populations remain underserved.

Another critical issue is the limited integration of technology and data analytics. Most traditional systems do not incorporate advanced tools such as machine learning, satellite imagery, or predictive modeling, which could provide actionable insights and improve

coordination. Communication between different stakeholders—governments, NGOs, and local responders—is often fragmented, leading to duplication of efforts or critical gaps in coverage. Furthermore, the logistical backbone of traditional systems tends to be slow and rigid, lacking intelligent routing or optimization capabilities. These inefficiencies highlight the urgent need for a more adaptive, real-time, and technology-driven approach to humanitarian aid distribution.

1.3.1 LIMITATIONS

- **Data Availability and Accuracy**
  The effectiveness of the AI system heavily depends on the availability and accuracy of real-time data such as disaster severity, population movements, and infrastructure status. In some regions, especially conflict zones or remote areas, reliable data may be scarce, delayed, or inconsistent, which can affect the system's decision-making accuracy.
- **Dependency on Technology Infrastructure**
  The proposed system relies on stable internet connectivity, cloud services, GPS, and power supply. In many disaster-struck areas, such infrastructure may be damaged or unavailable, making it difficult to collect or transmit real-time data, run predictive models, or execute optimized logistics.
- **Model Bias and Limitations in Prediction**
  AI and machine learning models are only as good as the data they are trained on. If the training data is biased or unrepresentative of certain types of crises or regions, the system’s predictions may be flawed, leading to unfair or inefficient distribution of aid.
- **Operational and Logistical Challenges**
  While the system can generate optimized routes and aid plans, actual implementation may face ground-level challenges such as blocked roads, political restrictions, or lack of vehicles and personnel. The system cannot always account for sudden, real-time disruptions in field operations.

1.4 PROPOSED SYSTEM

The proposed **AI-Based Humanitarian Aid Distribution System** is designed to enhance the speed, accuracy, and efficiency of crisis response by integrating real-time data from various sources such as satellites, drones, GPS, and social media. This multi-source data collection allows the system to continuously monitor disaster zones, assess severity, and identify the most affected regions. Unlike traditional methods that rely on manual assessments and static reports, this system dynamically updates its strategy based on evolving conditions, ensuring a more informed and timely response.

At its core, the system employs AI algorithms for route optimization and predictive analytics. By analyzing current data and historical crisis patterns, it forecasts aid requirements and allocates resources accordingly. The use of AI-powered drones further strengthens the system by providing aerial surveillance, mapping inaccessible areas, and delivering essential supplies where ground access is limited. These technologies work together to reduce delays, improve coverage, and ensure that aid reaches vulnerable populations efficiently, even in complex logistical environments.

To promote transparency and accountability, the system integrates blockchain technology for tracking and auditing every step of the aid distribution process. Each transaction—from dispatch to delivery—is securely recorded on a tamper-proof digital ledger. This not only prevents corruption and resource mismanagement but also builds trust among governments, NGOs, donors, and affected communities. By combining AI and blockchain, the system offers a robust, proactive, and transparent solution for managing humanitarian aid in crisis situations.

1.4.1 ADVANTAGES

• Real-Time Decision-Making

The system leverages real-time data from satellites, drones, sensors, and social media to assess crisis severity and needs instantly, enabling faster and more accurate decision-making.

• **Optimized Resource Allocation**

AI algorithms predict demand and allocate resources efficiently based on dynamic data, ensuring aid reaches the right people at the right time and minimizing waste or duplication.

•**Faster and Smarter Logistics**

Route optimization and predictive analytics reduce delivery time and fuel costs, allowing rapid deployment of aid, especially to remote or hard-to-reach areas.

1.5 OBJECTIVES

- **Real-time Crisis Assessment** – Analyze disaster severity and affected regions.
- **Optimized Resource Allocation** – Distribute aid efficiently based on demand.
- **Minimize Delays & Waste** – Use AI for faster and smarter deliveries.
- **Dynamic Decision-Making** – Adjust strategies using real-time data.
- **Enhanced Coordination** – Improve communication between relief agencies.
- **Supply Chain Optimization** – Automate logistics for better efficiency.

1.6 HARDWARE AND SOFTWARE REQUIREMENTS

1.6.1 SOFTWARE REQUIREMENTS

•Software:

The system is developed using **Visual Studio Code (VS Code)** as the primary integrated development environment (IDE). VS Code offers a versatile environment for writing, debugging, and managing the Python scripts used in real-time aid coordination and visualization.

•Primary Language:

The project is built primarily in Python, selected for its readability and support for data analytics, web development, and AI integration. Key Python libraries used include **pandas** for data manipulation, matplotlib and **seaborn** for data visualization, **pydeck** for geospatial mapping, random and datetime for dynamic generation and time-based logic.

• Frontend Framework:

The frontend interface is designed using **Streamlit**, which enables rapid development of responsive dashboards and data-driven web applications. Users can interact with the system through dropdowns, forms, charts, and maps all rendered via Streamlit.

• Backend Framework:

All core functionalities, including urgency scoring, resource allocation logic, and data filtering, are implemented in **Python** and executed through the Streamlit backend. This includes session management and user interaction routing.

• Database:

The system uses **SQLite3**, a serverless, lightweight relational database for storing:

- District and state mappings,
- Real-time NGO aid requests,
- Region-wise resource requirements and urgency scores.

• Frontend Technologies:

Although Streamlit abstracts most UI design, the application optionally supports:

- **HTML & CSS**: For minor UI customizations.
- **JavaScript**: For future integration of interactive charts or map components.
- **Bootstrap (if extended)**: For mobile responsiveness when integrating external templates.

1.6.2 HARDWARE REQUIREMENTS

- **Processor:** Intel Core i3 or higher
  A multi-core processor like Intel Core i3 or above is required to run backend services and AI computations smoothly.
- **RAM:** 4GB or more
  Minimum 4GB of RAM ensures smooth multitasking, especially for running the web interface and background processes.
- **Storage:** Minimum 500GB HDD or SSD
  Sufficient storage is needed to store application files, real-time data logs, and AI model files. SSD is preferred for better speed.
- **Internet Connection:** Required for data syncing and updates
  A stable internet connection is essential for real-time data fetching, system updates, and API communication.

2. LITERATURE SURVEY

M. C. Ramesh et al., 2022 This study explores the use of Deep Reinforcement Learning (DRL) for dynamic resource allocation in humanitarian aid. The approach allows real-time decision-making and adaptive responses to evolving crisis conditions. However, the model's complexity and high training time pose challenges for rapid deployment in real-world scenarios. The authors suggest developing simpler and more interpretable DRL models to make them practical and deployable in the field under urgent conditions [1].

Y. Zhang et al., 2023 Zhang and team introduce a Multi-Agent System to manage and coordinate humanitarian logistics among various aid organizations. This system enhances collaboration, reducing redundant deliveries and improving supply efficiency. Despite its advantages, the model depends heavily on inter-organizational data sharing, which can be limited due to privacy or policy restrictions. The study highlights the importance of secure and privacy-aware communication protocols between agents [2].

R. Banerjee et al., 2021 This research focuses on AI-driven geospatial analysis using satellite imagery to detect affected regions without the need for ground access. It enables rapid and wide-area assessment of disaster zones, improving the speed and accuracy of relief operations. However, the effectiveness of the system is constrained by the availability and quality of satellite images. The study recommends integrating drone imagery and real-time feedback from on-ground personnel for more reliable results [3].

A. Mohan et al., 2023 Mohan and colleagues propose a machine learning-based framework for forecasting crisis hotspots to anticipate aid demands and reduce waste. The system can help prioritize resources proactively. However, there is a risk of over-prioritizing predicted areas while overlooking others. The authors emphasize the need for balanced AI models that incorporate confidence levels and ethical considerations to ensure fair and effective resource allocation [4].

A. Gupta et al., 2022 This paper presents a hybrid AI system combining satellite imagery, on-ground reports, and weather data for real-time crisis impact mapping. The system improves decision speed and accuracy during floods and earthquakes. While promising, it relies on stable internet access and API availability in affected areas, which may not always

be feasible. The authors recommend incorporating offline-compatible backup models and hybrid cloud-edge architectures [5].

L. Kim et al., 2023 Kim's team proposed an urgency scoring algorithm that integrates socioeconomic vulnerability, past disaster patterns, and infrastructure risk. It helps prioritize not only based on physical impact but also on community resilience. However, the data collection process is labor-intensive and dependent on government cooperation. The study suggests using automated open-source scraping techniques to populate the models dynamically [6].

B. Singh et al., 2021 This study focuses on the use of blockchain in humanitarian aid tracking. It ensures transparency in resource distribution and prevents fraud. However, the model's integration is limited by infrastructure and user adoption challenges in remote areas. The authors propose hybrid systems where blockchain is paired with traditional data logs to ensure broader accessibility [7].

Table 2.1 Literature Survey of AI-Based Humanitarian Aid Distribution System

Ref	Author & year of publications	Journal / Conference	Method / Approach	Merits	Limitations	Research Gap
[1]	M. C. Ramesh et al., 2022	IEEE Access, doi: 10.1109/ ACCESS. 2022.3202113	Deep Reinforcement Learning for Dynamic Resource Allocation	Real-time decision-making and adaptability to crisis dynamics	High training time and complexity of model implementation in real -world crisis	Need for low-complexity and interpretable models for quick deployment in field conditions
[2]	Y. Zhang et al., 2023	arXiv preprint, doi: 2302.05643	Multi-Agent Systems for Coordinated Humanitarian Logistics	Promotes collaboration between aid agencies, prevents redundant supply drops	Requires inter-organizational data sharing which is often restricted	Development of secure and privacy-aware multi-agent communication protocols
[3]	R. Banerjee et al., 2021	Springer Humanitarian Tech Journal	AI-driven Geospatial Analysis using Satellite Imagery	Enables detection of affected regions without ground access, rapid area-wide assessments	Dependent on satellite image availability and quality	Integration of drone data and real-time feedback from field personnel
[4]	A.Mohan et al., 2023	Nature Human Behaviour, doi: 10.1038/s41562-023-01521-7	Machine Learning for Forecasting Crisis Hotspots	Anticipates aid demand, reduces resource wastage	May lead to over-prioritization of predicted zones and neglect of others	Balanced AI models that consider prediction confidence levels and ethical prioritization
[5]	A. Gupta et al., 2022	IEEE Sensors Letters	Hybrid AI system with weather + satellite + reports	Fast crisis detection with minimal delay	Needs stable internet and APIs in all zones	Offline-compatible backup models for remote deployment
[6]	L. Kim et al., 2023	Disaster Informatics Conf.	Socioeconomic & Infrastructure Risk Urgency Model	Prioritizes vulnerable communities, not just physical risk	Manual data gathering is time-intensive	Need for automated open-source scraping for real-time updates
[7]	B. Singh et al., 2021	Blockchain for Social Good	Blockchain-based Humanitarian Aid Tracking	Ensures transparency and prevents fraud	Adoption difficult in low-tech rural areas	Combine blockchain with fallback paper-based logs for redundancy

3. ANALYSIS AND DESIGN

The core objective of this project is to address the inefficiencies in traditional humanitarian aid distribution by integrating AI-driven solutions. Through comprehensive analysis, the system identifies key challenges such as delayed responses, lack of real-time data, and misallocation of resources. It evaluates crisis data like disaster severity, population density, logistics constraints, and available resources to understand the dynamics of aid requirements. This analysis helps identify vulnerable zones, predict future needs using historical patterns, and assess existing distribution mechanisms to highlight areas of improvement.

The system follows a modular architecture combining real-time data processing, AI-driven decision-making, and efficient communication interfaces. The backend, developed using Python and frameworks like Flask or Django, handles the AI algorithms and data processing. It incorporates machine learning models for predictive analytics and route optimization. The frontend, built with React.js and standard web technologies (HTML, CSS, JavaScript), presents an interactive interface for users such as government bodies, NGOs, and aid workers to monitor crises, approve aid deliveries, and track logistics in real time. A centralized database supports seamless data access, and APIs enable integration with satellite feeds, drone inputs, and NGO requests.

When a crisis occurs, the system collects real-time geospatial and logistical data through APIs or CSV uploads. The AI engine analyzes the data to generate urgency scores, identify affected zones, and suggest optimal resource allocation routes. Decision-makers can interact with the dashboard to validate, modify, or approve recommendations. The design also emphasizes transparency by integrating blockchain or audit logs to track aid movement and ensure accountability. The modularity of the design allows for future scalability and integration of additional technologies such as drone surveillance and IoT devices for field updates.

3.1 MODULES

1. Crisis Data Collection Module

Collects real-time data from diverse sources such as satellite imagery, social media feeds, news updates, and IoT-based on-ground sensors. This data helps detect crisis onset, scale, and impact across different geographic regions.

- **Input:** Satellite data, drone feeds, social media alerts, government bulletins, NGO reports, and sensor data.
- **Output:** Aggregated crisis data stream capturing disaster type, location, scale, and temporal changes for further analysis.

2. Preprocessing & Pattern Recognition Module

This module filters and organizes raw input data to make it suitable for analysis. It detects patterns like flood spread, food scarcity clusters, or medical emergencies using AI/ML algorithms.

- **Input:** Raw collected data including unstructured reports, images, location logs, and past crisis datasets.
- **Output:** Cleaned and structured data with extracted features such as crisis zones, severity levels, and affected populations.

3. Demand Forecasting & Prioritization Module

Uses historical data and real-time indicators to forecast aid demand in affected regions. It assigns urgency scores to each location based on population density, infrastructure damage, and scarcity indicators.

- **Input:** Pre-processed crisis data including demographic, logistic, and damage-related features.
- **Output:** Forecasted aid demand for each region along with priority rankings for immediate action.

4. Resource Allocation & Optimization Module

Implements AI-driven algorithms to optimize aid delivery routes and resource distribution across priority zones. It considers constraints like resource availability, transportation feasibility, and supply chain delays.

- **Input:** Demand predictions, current resource inventory, transport routes, and infrastructure data.
- **Output:** Optimized logistics plan including delivery schedules, resource allocation maps, and transport routing.

5. Multi-Agent Coordination Module

Enables seamless collaboration among government agencies, NGOs, and international aid groups using a shared digital interface. It ensures transparency, avoids duplication of efforts, and enables coordinated decision-making.

- **Input:** Resource plans, organizational roles, available personnel and assets, shared crisis data.
- **Output:** Real-time coordination dashboards, updated task assignments, and synchronized response workflows.

3.2 ARCHITECTURE

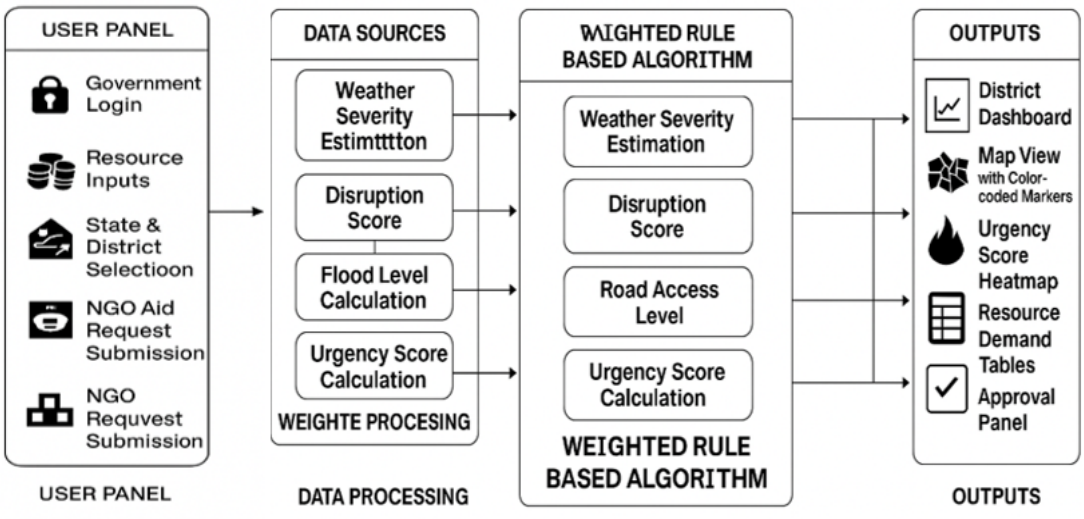


Fig. 3.2.1 Architecture of AI-Based Humanitarian Aid Coordination System

The architecture of the AI-Based Humanitarian Aid Coordination System is designed to ensure effective crisis management by integrating various data sources, applying intelligent rule-based algorithms, and providing actionable outputs through a user-friendly interface. The system begins with the User Panel, which serves as the primary interaction interface for stakeholders such as government officials and NGOs. Users can log in, input resource information, and submit aid requests. They can also select the specific state and district they are working with, allowing the system to filter data and provide localized insights.

Next, the Data Processing Layer handles inputs from multiple sources to build a robust understanding of the crisis environment. This includes weather severity estimation, disruption scores, flood levels, and urgency score calculation. These variables are collected from real-time feeds such as weather APIs, satellite images, and NGO field reports. This raw data is then processed through a weighting mechanism that assigns significance to each parameter based on predefined criteria, helping to prioritize the most critical information for decision-making.

The heart of the system lies in the Weighted Rule-Based Algorithm module. Here, each processed input is analyzed using logical rules that take into account the intensity of the disaster, road access levels, disruption to essential services, and regional vulnerability. These elements are combined to generate a final Urgency Score, which helps determine which districts or zones require immediate attention. This ensures that limited resources are allocated based on urgency and not just availability or location.

The final outputs are then displayed in a highly visual and actionable form through the Output Panel. This includes a District Dashboard showing key statistics, a Map View with color-coded markers to represent severity levels, a Heatmap of Urgency Scores, Resource Demand Tables, and an Approval Panel for validating and dispatching resources. These visual tools empower authorities and NGOs to make quick, informed decisions and monitor crisis evolution in real time.

Overall, this architecture seamlessly integrates data acquisition, intelligent processing, and dynamic visualization to streamline humanitarian response. By combining real-time inputs with predictive scoring and collaborative tools, the system greatly enhances situational awareness and resource management during crises.

3.3 UML DIAGRAMS

3.3.1 USE CASE DIAGRAMS

A Use Case Diagram is a crucial component of system design that helps visualize the interactions between external actors (users) and the system itself. It illustrates what the system does (use cases) from the perspective of users, offering a clear overview of its functional requirements. In the context of the AI-Based Humanitarian Aid Distribution System, the use case diagram showcases the roles of Government Officials and NGOs, and how they interact with the system to manage and streamline crisis resource allocation effectively.

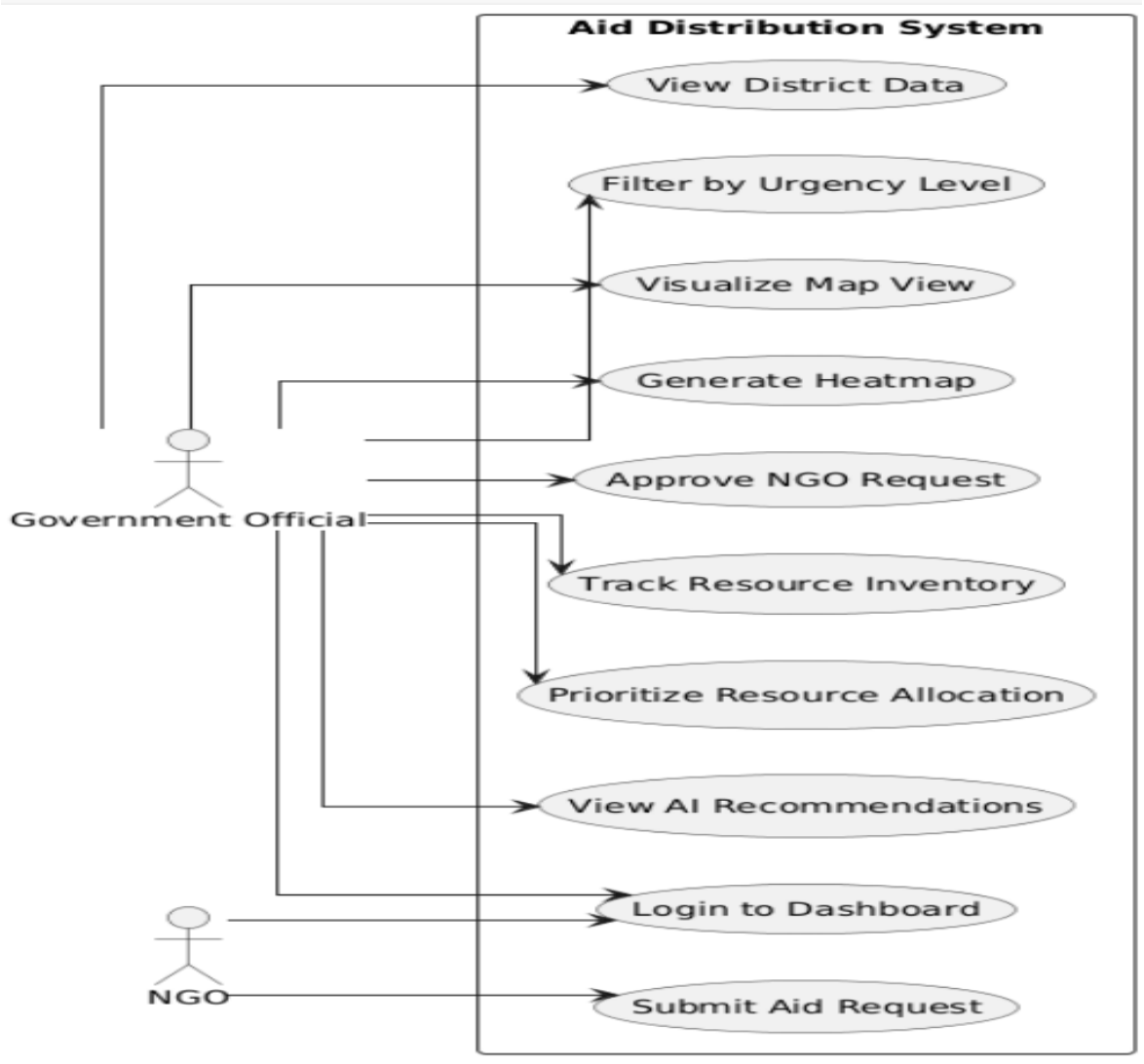


Fig. 3.3.1.1 Use Case Diagram

Actors:

1. **Government Official:**
   The primary actor responsible for managing and monitoring aid distribution. They interact with the system to analyze district-level crisis data, make decisions based on AI-generated insights, and approve or deny aid requests.
2. **NGO (Non-Governmental Organization):**
   Secondary actor that uses the system to submit aid requests for affected regions. NGOs collaborate with the government panel by reporting on-ground situations and requesting necessary resources.

Use Cases:

1. **Login to Dashboard:**
   Government officials and NGOs can securely access the system.
2. **View District Data:**
   Officials can view real-time district-wise crisis data.
3. **Filter by Urgency:**
   Allows filtering of districts by urgency level (e.g., Immediate, Urgent, Monitor).
4. **Visualize Map & Heatmap:**
   View crisis zones through color-coded map and urgency heatmaps.
5. **Submit Aid Request (NGO):**
   NGOs request resources specifying region and aid type.
6. **Approve NGO Request:**
   Officials review and approve or reject aid submissions.
7. **Track & Prioritize Resources:**
   Officials monitor available resource stock and allocate aid efficiently based on urgency.
8. **View AI Recommendations:**
   System displays AI-generated support strategies per region.

3.3.2 CLASS DIAGRAM

A class diagram is a fundamental component in object-oriented modeling that provides a blueprint of a system's structure. It shows classes, attributes, methods, and the relationships among different objects in the system. It is particularly valuable during software development for system design, database mapping, and implementation planning.

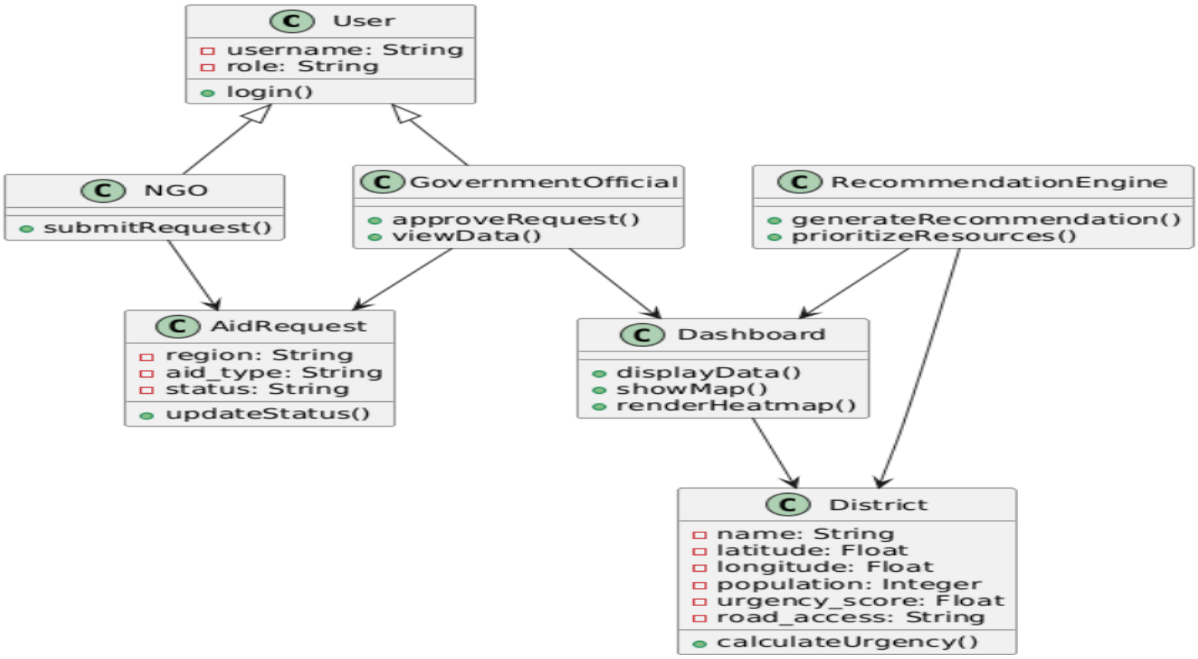


Fig. 3.3.2.1 Class Diagram

Relationships:

User

- Attributes: username, role
- Methods: login()
- A generic superclass inherited by both NGO and GovernmentOfficial.

NGO

- Inherits from User.
- Method: submitRequest()
- Submits aid requests to the system.

Government Official

- Inherits from User.
- Methods: approveRequest(), viewData()

- Has authority to approve aid requests and access district-level data.

Aid Request

- Attributes: region, aid_type, status
- Method: updateStatus()
- Contains all details related to aid requests made by NGOs.

Dashboard

- Methods: displayData(), showMap(), renderHeatmap()
- Acts as a visualization layer for government officials to analyze and act.

Recommendation Engine

- Methods: generateRecommendation(), prioritizeResources()
- Uses AI to provide optimized recommendations based on urgency and resource availability.

District

- Attributes: name, latitude, longitude, population, urgency_score, road_access
- Method: calculateUrgency()
- Each district contains geographic and crisis data used in prioritization and mapping.

System Flow:

NGO Interaction

- An NGO user logs in and uses submitRequest() to send a request to the system.
- The request is stored as an AidRequest object with appropriate attributes.

Government Official Role

- Logs in and uses viewData() to analyze regional crisis levels.
- Uses approveRequest() to either approve or reject NGO requests.

Visualization & Recommendations

- The Dashboard gathers data and visualizes it using showMap() and renderHeatmap().
- The RecommendationEngine processes real-time district data and gives resource allocation suggestions.

Urgency Calculation

- The District class uses attributes like population, road access, and disruption levels to compute urgency via calculateUrgency().

3.3.3 ACTIVITY DIAGRAM

An activity diagram is a type of behavioral diagram in UML (Unified Modeling Language) that visually represents the workflow of actions and activities in a system. It highlights the sequence of operations, conditional branching, parallel processes, and user/system interactions. Activity diagrams are especially useful for understanding business processes and system functionalities in dynamic behavior.

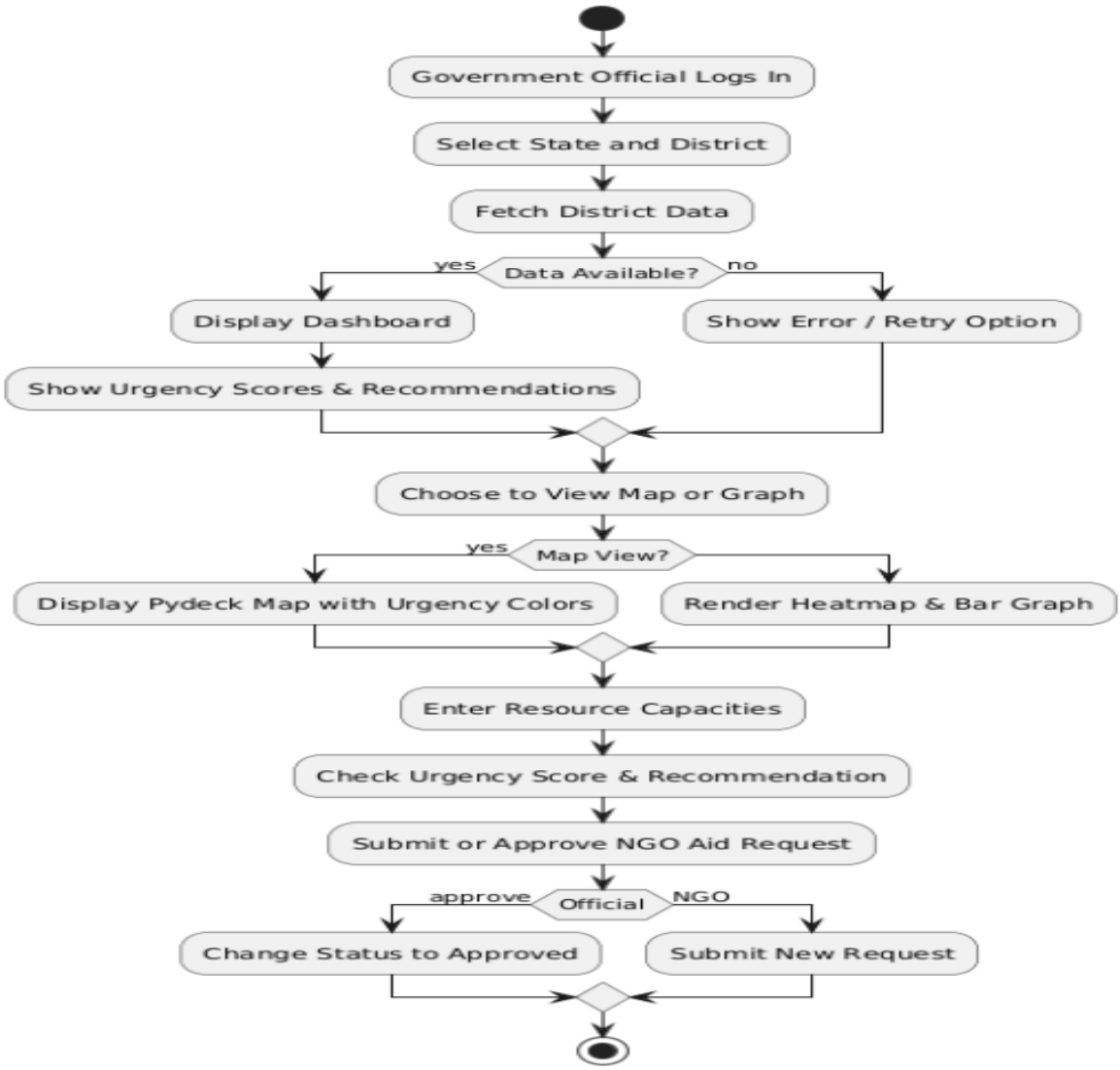


Fig. 3.3.3.1 Activity Diagram

Flow Explanation:

1. Government Official Logs In

- The process starts with the Government Official logging into the system.
- The user is authenticated and gains access to the dashboard functionalities.

2. Select State and District

- The official selects the geographic area of interest (state and district) to fetch relevant data.

3. Fetch District Data

- The system attempts to retrieve real-time district data (population, urgency score, disruption, etc.).

4. Conditional Check – Data Available?

- **Yes:** If data is available, it proceeds to show the dashboard.
- **No:** An error or retry option is displayed, allowing the official to try again or select a different region.

5. Display Dashboard

- The dashboard becomes visible, summarizing the region's crisis status.

6. Show Urgency Scores & Recommendations

- Urgency scores (calculated using calculateUrgency() in the District class) and AI-based resource recommendations are shown.

7. Choose to View Map or Graph

- The official decides whether to visualize data through a map or a graph.

8. Conditional Branch – Map View?

- **Yes:** Display Pydeck Map with urgency color coding for visual insight into regional crisis levels.
- **No:** Render a Heatmap and Bar Graph to view trends and comparative metrics.

9. Enter Resource Capacities

- The official inputs current available resources (e.g., food kits, medical supplies) for distribution planning.

10. Check Urgency Score & Recommendation

- The system cross-verifies the urgency of districts with available resources and AI recommendations.

11. Submit or Approve NGO Aid Request

- Two possible roles:
    - NGO submits new aid requests through their interface.
    - Government Official approves the pending requests after review.

12. Conditional Branch – Who Initiates?

- If initiated by the official, the system updates the aid request status to "Approved."
- If initiated by the NGO, the system logs it as a new request to be processed.

13. End of Activity

- The activity concludes after the submission or approval of aid requests, enabling the aid dispatch workflow.

3.3.4 SEQUENCE DIAGRAM

A sequence diagram illustrates the flow of interactions between actors and system components over time, as seen in Fig. 3.3.4.1, emphasizing the order in which messages are exchanged to achieve specific functionalities. Actors represent external entities that interact with the system, while lifelines depict the system components involved in the process. Messages are shown as arrows, indicating the flow of information or actions between these elements. By providing a step-by-step view of workflows, sequence diagrams help in understanding and designing the dynamic behaviour of a system.

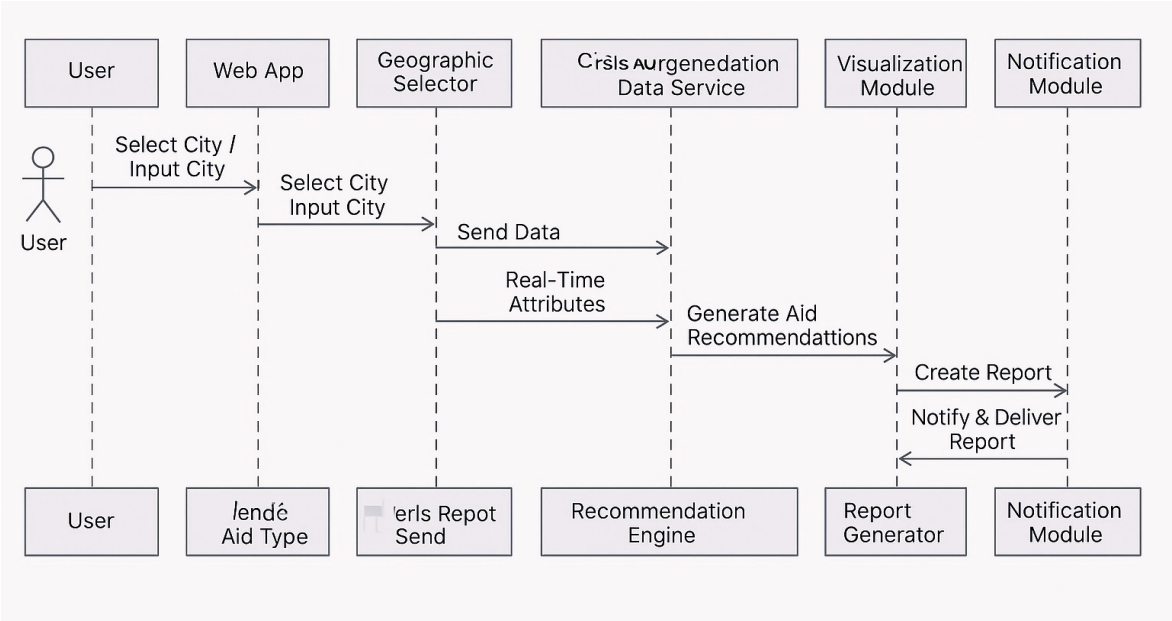


Fig. 3.3.4.1 Sequence Diagram

Key Interactions and Relationships

- **User and Web App:**
  - Select City / Input City: The user interacts with the Web App to choose or manually input a city for crisis analysis.
  - **Send Data:** The user submits the selected city data to the system for further processing.
- **Web App and Geographic Selector:**
  - Real-Time Attributes: The Web App sends the city data to the Geographic Selector to retrieve real-time attributes (e.g., location details, crisis-related data).
- **Geographic Selector and Crisis Augmentation Data Service:**
  - Generate Aid Recommendations: The Geographic Selector forwards the processed data to the Crisis Augmentation Data Service, which generates tailored aid recommendations based on the crisis context.
- **Crisis Augmentation Data Service and Visualization Module:**
  - Create Report: The service sends the aid recommendations to the Visualization Module, which compiles and formats the data into a comprehensive report**.**
- **Visualization Module and Notification Module:**
  - Notify & Deliver Report: The Visualization Module triggers the Notification Module to alert the user and deliver the finalized report.
- **User and System:**
  - View Report: The user receives the report and recommendations, completing the workflow.

This sequence diagram captures the end-to-end process of crisis aid recommendation generation, from user input to report delivery, highlighting the collaboration between system components.

3.3.5 COMPONENT DIAGRAM

A Component Diagram is a type of structural diagram used in software engineering to represent the components of a system and how they interact or depend on each other. It shows how the components (which could be software modules, subsystems, or other significant parts) are organized and connected within a system. In this diagram, each component encapsulates a set of related functionalities and interfaces, as shown in Fig. 3.3.5.1.

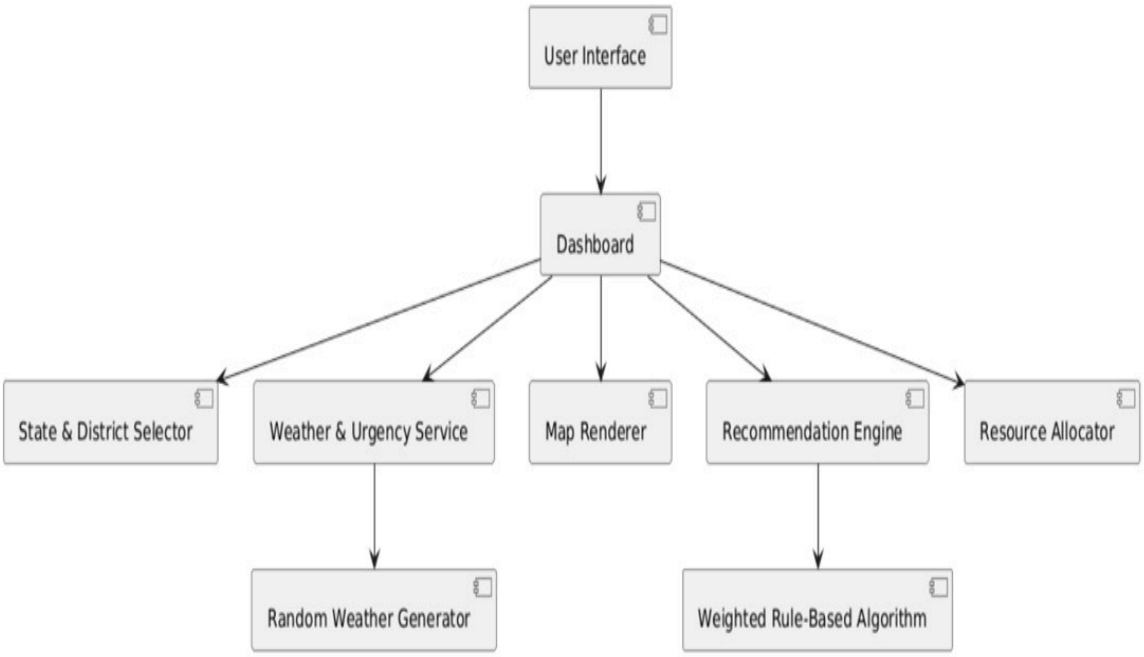


Fig. 3.3.5.1 Component Diagram

Main Components:

1. **User Interface (Crisis Dashboard - Streamlit Web App):**
    - **Description:** The front-end interface where users interact with the system. It allows users to fetch crisis data, request weather info, view AI-generated scores, submit aid requests, approve/reject requests, and download CSV reports.
2. **Backend Server:**
    - **Description:** The core server that processes user requests and coordinates interactions between other components. It includes sub-components like the Data Manager, Weather Fetcher, and AI Recommendation Engine.

3. **Data Manager:**
    - **Description:** Handles data operations such as saving new aid requests, updating request statuses, and managing crisis data storage and retrieval.
4. **Weather Fetcher:**
    - **Description:** Communicates with external APIs (e.g., OpenWeatherMap) to fetch live weather data relevant to crisis-affected regions.
5. **AI Recommendation Engine:**
    - **Description:** Processes crisis data and generates AI-driven scores and recommendations for aid distribution based on real-time inputs.
6. **NGO Request Handler:**
    - **Description:** Manages aid requests submitted by NGOs, including validation and forwarding to the Government Approval Panel.
7. **Government Approval Panel:**
    - **Description:** Reviews and approves/rejects aid requests based on predefined criteria and available resources.
8. **Report Generator:**
    - **Description:** Compiles data into structured reports (e.g., CSV files) for stakeholders, including crisis details, weather info, and AI recommendations.
9. **Database:**
    - **Description:** Stores all system data, including aid requests, session states, and crisis-related information. Sub-components:
        - Aid Requests Database: Tracks all aid requests and their statuses.
        - Session State Storage: Maintains user session data for real-time interactions.
10. **External Systems:**
    - **Description:** Integrates with third-party services like the OpenWeatherMap API to fetch live weather updates for crisis regions.

3.3.6 DEPLOYMENT DIAGRAM

The Deployment Diagram illustrates the physical deployment of software components across hardware nodes in the Humanitarian Aid distribution Optimisation system. It captures the interactions between the user device**,** server, and database, highlighting how system components communicate and are distributed in a real-world deployment scenario.

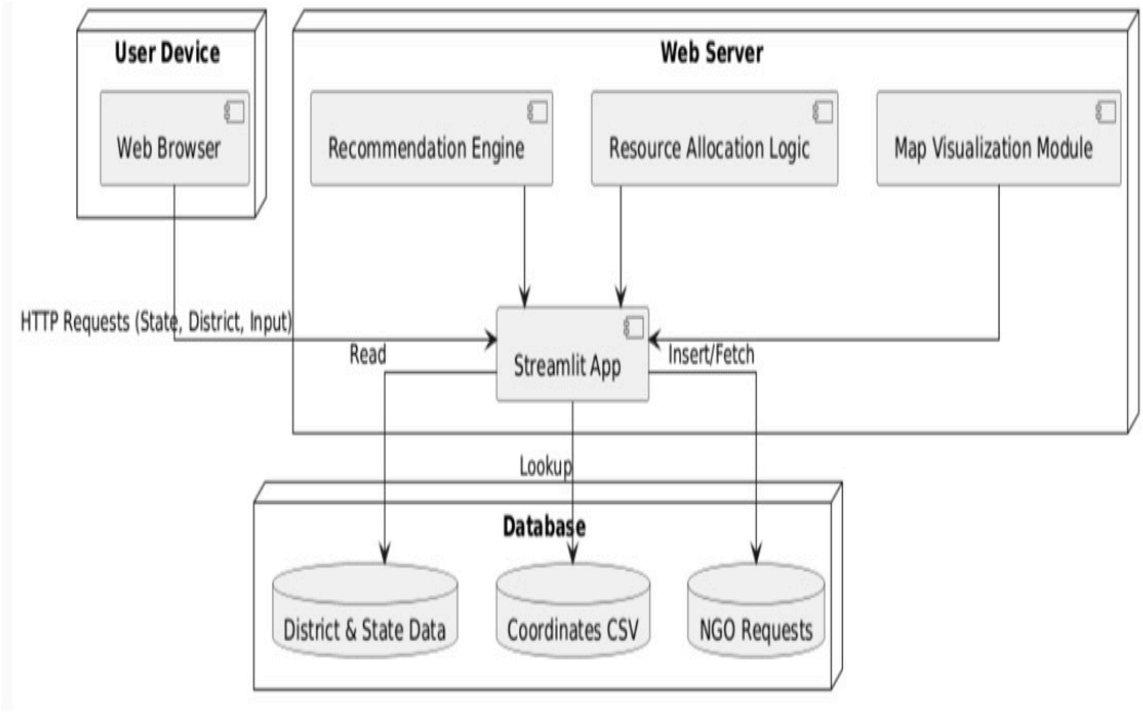


Fig. 3.3.6.1 Deployment Diagram

The deployment diagram shows how the system components are distributed across three main nodes:

- **UserDevice:**
  Operates through a web browser interface, allowing users (government officials or NGO members) to select states, input crisis data, and view recommendations. Communication with the server occurs via HTTP requests that transmit region-wise inputs.
- **Server:**
  Hosts the **Streamlit Web Application** and incorporates:
  - Recommendation Engine to classify urgency using rule-based logic, Resource Allocation Module to estimate food, shelter, and medical needs

based on population and urgency scores, Map Visualization Module to display colored urgency maps and heatmaps using Pydeck and Matplotlib. The server acts as the central processor, managing both business logic and UI rendering.

- **Database:**
  Built using **SQLite**, the database stores:
  - District and state mapping data (CSV),Geographical coordinates for map rendering, The server accesses and modifies this data via SQL queries (read, insert, update).

This architecture ensures seamless coordination between the front-end interface, AI-driven decision logic, and backend storage—enabling real-time crisis data visualization and actionable humanitarian aid decisions.

3.4 METHODOLOGY

Data Acquisition

- **Live Data Integration**
  • The system retrieves near real-time crisis-related data from multiple sources (e.g., government disaster databases, weather APIs, or manually updated CSVs).
  • Purpose: This dynamic data ensures that the urgency scoring and recommendations reflect the latest field conditions during crises such as floods, pandemics, or food shortages.
- **Historical & Static Data**
  • Some data such as district-level demographics, past aid requests, NGO details, and resource availability are preloaded or manually updated into the system.
  • Purpose: This baseline data helps improve accuracy in urgency score computation and AI recommendations.

Model Steps

- Data Preprocessing & Cleaning

  • The raw datasets are cleaned to handle missing values, inconsistent formats, or outdated records.

  • Key fields such as Disruption Level, Food Shortage, Flood Impact, and Population are normalized for fair comparison across districts.
- **Urgency Score Calculation**

  • A weighted scoring system combines multiple factors (e.g., disaster intensity, food shortage level, and disruption level).

  • Formula Example:

  Urgency Score = 0.4 * Disruption + 0.3 * Food Shortage + 0.2 * Flood Impact + 0.1 * Population Density (normalized)
- **Recommendation Logic**

  • AI-based logic (using simple rules or ML models) recommends types and quantities of resources needed based on urgency score thresholds.

  • Districts with higher scores are prioritized, and resources are allocated accordingly.

Visualization & Interaction

- **Map-Based Visualization (Pydeck)**

  • Pydeck maps render districts with color-coded urgency levels (e.g., red for high urgency).

  • Helps officials quickly identify critical areas.
- **Graph-Based Visualization**

  • Bar graphs and heatmaps are used to compare multiple districts across factors like food shortages and flood impact.

  • Enables analytical decision-making when approving NGO aid requests.

Decision & Workflow Automation

- NGO Request Submission
  NGOs submit aid requests specifying resource needs, affected areas, and urgency levels.
- Government Approval Interface
  Officials can view real-time recommendations and urgency data before approving or rejecting aid requests.
- Status Management
  Upon approval, the request status updates in the system (e.g., from *Pending* to *Approved*).
  Triggers the aid dispatch process in coordination with logistics partners.

Why This Methodology?

- Real-Time Response: Integrating live data ensures the system adapts to sudden disaster changes.
- AI-Powered Fairness: Prioritization based on urgency score removes bias and supports ethical distribution.
- Transparency: Officials and NGOs operate through a single shared dashboard, improving visibility and accountability.
- Scalability: The methodology supports multi-region use and can be extended to include more disaster types and predictive models in the future.

4. CODE AND IMPLEMENTATION

4.1 CODE

app.py

```
import streamlit as st
import pandas as pd
import random
import requests
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import pydeck as pdk

# Page setup
st.set_page_config(page_title="🌍 Crisis Coordination Dashboard", layout="centered")
st.title("🤝 AI-Based Humanitarian Aid Coordination")

# Load district coordinates
coords_df = pd.read_csv("all_state_district_coordinates.csv")
coords_map = dict(zip(coords_df["District"], zip(coords_df["Latitude"],
coords_df["Longitude"])))

# Government login
GOVT_USERNAME = "govt_user"
GOVT_PASSWORD = "secure123"
if "govt_logged_in" not in st.session_state:
    st.session_state.govt_logged_in = False

with st.sidebar:
    st.subheader("🔐 Government Login")
    if not st.session_state.govt_logged_in:
        username = st.text_input("Username")
        password = st.text_input("Password", type="password")
        if st.button("Login"):
            st.session_state.govt_logged_in = True
            st.success(f"Logged in as: {username}")
    else:
        st.info("Logged in as government official.")

    st.markdown("## 🔍 Navigate")
    section = st.radio("Go to Section", [
        "Dashboard", "Map View", "Recommendations",
        "Resource Prioritization", "NGO Requests", "Approval Panel"
    ])

    st.markdown("## 🛠️ Resource Capacity")
```

```
    total_food = st.number_input("Total Food Units", 0, 100000, 5000)
    total_medical = st.number_input("Total Medical Kits", 0, 10000, 1000)
    total_shelter = st.number_input("Total Shelter Units", 0, 10000, 500)

    st.markdown("## 🌟 Urgency Filter")
    urgency_level = st.selectbox(
        "Filter by Urgency",
        ["All", "🚨 Immediate Deployment", "⚠️ Urgent Support Required", "📊 Monitor Situation"]
    )

@st.cache_data
def get_districts_from_csv(state_name):
    df = pd.read_csv("full_indian_districts_updated.csv")
    return df[df["State"] == state_name]["District"].tolist()

@st.cache_data(ttl=1800)
def generate_district_data(state):
    districts = get_districts_from_csv(state)
    records = []
    for d in districts:
        lat, lon = coords_map.get(d, (None, None))
        severity = round(random.uniform(0.0, 1.0), 2)
        disruption = round(random.uniform(0.7, 1.0), 2)
        flood = min(10, int(disruption * 10))
        road = random.choices(["Blocked", "Low", "Medium", "High"], weights=[0.4, 0.2, 0.2, 0.2])[0]
        urgency = round(disruption * 0.4 + flood / 10 * 0.2 + severity * 0.2 + (0.2 if road == "Blocked" else 0), 2)
        recommendation = "🚨 Immediate Deployment" if urgency > 0.7 else "⚠️ Urgent Support Required" if urgency > 0.5 else "📊 Monitor Situation"
        records.append({
            "District": d,
            "Latitude": lat,
            "Longitude": lon,
            "Weather Severity": severity,
            "Disruption": disruption,
            "Flood": flood,
            "Road Access": road,
            "Urgency Score": urgency,
            "AI Recommendation": recommendation,
            "Population": random.randint(5000, 20000)
        })
    return pd.DataFrame(records)

selected_state = st.selectbox("Select a State", ["Telangana", "Maharashtra", "Delhi", "West Bengal", "Tamil Nadu", "Karnataka"])
if st.button("🔄 Refresh District Data"):
    st.session_state[selected_state] = generate_district_data(selected_state)
```

```
    st.session_state[f"{selected_state}_last_updated"] = datetime.now().strftime("%Y-%m-
%d %H:%M:%S")

if selected_state not in st.session_state:
    st.session_state[selected_state] = generate_district_data(selected_state)
    st.session_state[f"{selected_state}_last_updated"] = datetime.now().strftime("%Y-%m-
%d %H:%M:%S")

df_districts = st.session_state[selected_state]
last_updated = st.session_state.get(f"{selected_state}_last_updated", "Unknown")
st.markdown(f"🕒 **Last Updated**: `{last_updated}`")

if urgency_level != "All":
    df_districts = df_districts[df_districts["AI Recommendation"] == urgency_level]

if section == "Dashboard":
    st.subheader(f"📍 District-wise Data for {selected_state}")
    st.dataframe(df_districts)

    st.subheader("📉 Urgency Bar Chart")
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.bar(df_districts["District"], df_districts["Urgency Score"], color='tomato')
    ax.set_title("Urgency Score by District")
    ax.set_xlabel("District")
    ax.set_ylabel("Urgency Score")
    ax.set_ylim(0, 1)
    ax.grid(axis='y')
    plt.xticks(rotation=90)
    st.pyplot(fig)

elif section == "Map View":
    st.subheader(f"🗺️ District Map of {selected_state}")
    df_districts["Color"] = df_districts["AI Recommendation"].apply(
        lambda x: [255, 0, 0, 180] if "Immediate" in x else [255, 165, 0, 160] if "Urgent" in x
else [0, 128, 0, 120]
    )
    if not df_districts.empty and "Latitude" in df_districts.columns:
        map_layer = pdk.Layer(
            "ScatterplotLayer",
            data=df_districts.dropna(subset=["Latitude", "Longitude"]),
            get_position='[Longitude, Latitude]',
            get_color='Color',
            get_radius=20000,
            pickable=True
        )
        tooltip = {
            "html": "<b>{District}</b><br/>Urgency: {Urgency
Score}<br/>Recommendation: {AI Recommendation}",
            "style": {"backgroundColor": "steelblue", "color": "white", "padding": "5px"}
        }
```

```
        view_state = pdk.ViewState(
            latitude=df_districts["Latitude"].mean(),
            longitude=df_districts["Longitude"].mean(),
            zoom=6,
            pitch=0
        )
        st.pydeck_chart(pdk.Deck(layers=[map_layer], initial_view_state=view_state,
tooltip=tooltip))
    else:
        st.info("No coordinates available for districts in this state.")

elif section == "Recommendations":
    st.subheader("📊 AI Recommendations Summary")
    st.bar_chart(df_districts["AI Recommendation"].value_counts())
    st.line_chart(df_districts.sort_values(by="Urgency Score",
ascending=False).set_index("District")["Urgency Score"])

elif section == "Resource Prioritization":
    st.subheader("📦 Resource Prioritization")
    df_districts["Food Needed"] = (df_districts["Urgency Score"] *
df_districts["Population"] * 0.02).astype(int)
    df_districts["Shelter Needed"] = (df_districts["Urgency Score"] *
df_districts["Population"] * 0.01).astype(int)
    df_districts["Medical Needed"] = (df_districts["Urgency Score"] *
df_districts["Population"] * 0.015).astype(int)
    st.dataframe(df_districts[["District", "Urgency Score", "Population", "Food Needed",
"Shelter Needed", "Medical Needed"]])

elif section == "NGO Requests":
    st.subheader("📨 NGO Aid Request")
    aid_region = st.selectbox("Select Region for Aid", df_districts["District"].unique())
    aid_type = st.text_input("Type of Aid (e.g. food, shelter, medical)")
    if "aid_requests" not in st.session_state:
        st.session_state.aid_requests = []
    if st.button("📤 Submit Aid Request") and aid_type:
        st.session_state.aid_requests.append({"Region": aid_region, "Aid Type": aid_type,
"Status": "Pending"})
        st.success(f'Aid request submitted for {aid_region}!")

elif section == "Approval Panel":
    st.subheader("📬 Government Approval Panel")
    if st.session_state.govt_logged_in:
        requests_df = pd.DataFrame(st.session_state.aid_requests)
        if not requests_df.empty:
            for i, row in requests_df[requests_df["Status"] == "Pending"].iterrows():
                st.markdown(f"*Region: {row['Region']} | **Aid**: {row['Aid Type']}")
                if st.button(f"✅ Approve Request #{i}"):
                    st.session_state.aid_requests[i]["Status"] = "Approved"
                    st.success(f"Request for {row['Region']} approved.")
```

```
    else:
        st.warning("Login as government official to approve.")

st.subheader("🔥 Urgency Score Heatmap")
st.markdown(f"**Filtered Districts:** {len(df_districts)}")

if df_districts.empty:
    st.info("No districts match the selected urgency level. Try changing the filter to 'All'.")
elif len(df_districts) < 2:
    st.warning("Not enough data to generate a heatmap. At least two districts are needed.")
elif "Urgency Score" not in df_districts.columns:
    st.error("Urgency Score column missing in data.")
else:
    heat_df = df_districts[["District", "Urgency Score"]].copy()
    heat_df.set_index("District", inplace=True)
    fig, ax = plt.subplots(figsize=(10, len(heat_df) * 0.5))
    sns.heatmap(heat_df, annot=True, cmap="Reds", linewidths=0.5, fmt=".2f", ax=ax)
    st.pyplot(fig)
```

4.2 IMPLEMENTATION

Create a directory folder as following and copy relevant pieces of code into the required parts: -

```
AidDistributionSystem
│
├── app.py                    # Main Streamlit application
├── requirements.txt          # Python dependencies (Streamlit, pandas, matplotlib, seaborn, pydeck)
│
├── data/
│   ├── all_state_district_coordinates.csv    # District lat-long data
│   ├── full_indian_districts_updated.csv     # State → District mapping
│   └── aid_requests.db                       # SQLite database for aid request tracking
│
├── src/
│   ├── scoring.py            # Urgency score calculation (rule-based logic)
│   ├── resources.py          # Calculates food, shelter, and medical needs
│   ├── map_generator.py      # Pydeck map rendering logic
│   └── heatmap.py            # Seaborn heatmap generator
│
├── static/
│   └── style.css             # Optional CSS for custom Streamlit styling
│
├── templates/                # (Optional, used if you migrate to Flask)
│
├── README.md                 # Project summary and setup instructions (optional)
└── LICENSE                   # (Optional, add if open-source)
```

Installing Python Packages

To set up your environment and install all required Python libraries, follow these steps:

Open the integrated terminal in VS Code and activate your virtual environment.

Run the following command to install all required packages:

pip install streamlit pandas matplotlib seaborn pydeck

If you're using a requirements.txt, you can also run:

pip install -r requirements.txt

Setting Up SQLite

This project uses **SQLite**, a lightweight embedded database — no separate installation is required.

To View or Edit the Database:

1. Install DB Browser for SQLite
   📥 Download: https://sqlitebrowser.org/dl
2. After installation**:**
   - Open data/aid_requests.db
   - You'll be able to view or manually update NGO requests and government login data (if implemented)

Creating the Required Tables

Open the terminal in **VS Code** and launch the SQLite shell:

sqlite3 data/aid_requests.db

Paste the following SQL commands to create required tables:

```
-- Table for NGO aid requests
CREATE TABLE IF NOT EXISTS ngo_requests (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    district TEXT,
    aid_type TEXT,
    status TEXT DEFAULT 'Pending'
);

-- Optional: Table for government login (if login system is used)
CREATE TABLE IF NOT EXISTS officials (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT,
    password TEXT
);
```

To exit the SQLite shell:

```
.exit
```

Running the Application

1. Open terminal and navigate to your project root:

   ```
   cd AidDistributionSystem
   ```

2. Run the Streamlit application:

   ```
   streamlit run app.py
   ```

3. Open your browser and visit:

   http://localhost:8501

5. TESTING

5.1 INTRODUCTION TO TESTING

Testing is a critical phase in the software development lifecycle, ensuring that every component of the system behaves as expected across various use cases and data conditions. It validates the system's functionality, performance, reliability, and user experience under normal and extreme circumstances.

In the **AI-Based Humanitarian Aid Distribution System**, testing played a vital role in ensuring that all modules—from urgency scoring to resource prioritization—operate accurately and cohesively to support emergency response workflows.

Testing focused on the following core components:

- District selection and urgency score calculation
- Dashboard data presentation
- Map visualization and tooltip accuracy
- NGO Aid Request submission and status tracking
- Filtering and data responsiveness
- User session and state memory

5.2 TEST CASES:

Table 5.1 Test Cases of AI-Based Humanitarian Aid Distribution System

Test Case ID	Test case Name	Test Description	Expected Output	Actual Output	Remarks
TC_01	Home → Login	Government user enters credentials and logs in	Sidebar shows government sections	Sidebar with government-only features shown	Success
TC_02	State & District Selection	Select state (e.g., Telangana) from dropdown	District data loaded for the selected state	Districts and urgency data displayed	Success
TC_03	Urgency Scoring	Trigger generate_district_data() after selecting state	Urgency score calculated using weather, flood, disruption	Scores generated and displayed correctly	Success
TC_04	Map Visualization	Load map after district data generation	Pydeck map shows color-coded urgency circles	Map renders with hover tooltips for each district	Success
TC_05	Recommendation Panel	View recommendation tab after data load	Shows recommendation counts	Bar and line charts correctly rendered	Success
TC_06	Resource Prioritization	Calculate resources for each district	Food, medical, and shelter needs based on urgency and population	Table shows correct computed values	Success
TC_07	NGO Request Form	Submit a request for aid (e.g., Food at Hyderabad)	Request appears in pending approval li	Request logged and visible to government panel	Success
TC_08	Approval Panel	Approve pending NGO requests	Status of request changes to "Approved"	Status correctly updated in table	Success
TC_09	Heatmap Rendering	View urgency heatmap with ≥ 2 districts	Seaborn heatmap shows urgency scores per district	Heatmap renders with correct values and labels	Success
TC_10	Input Validation	Skip entering aid type and click Submit	Shows error for missing required input	Error displayed, submission blocked	Success

6. RESULTS

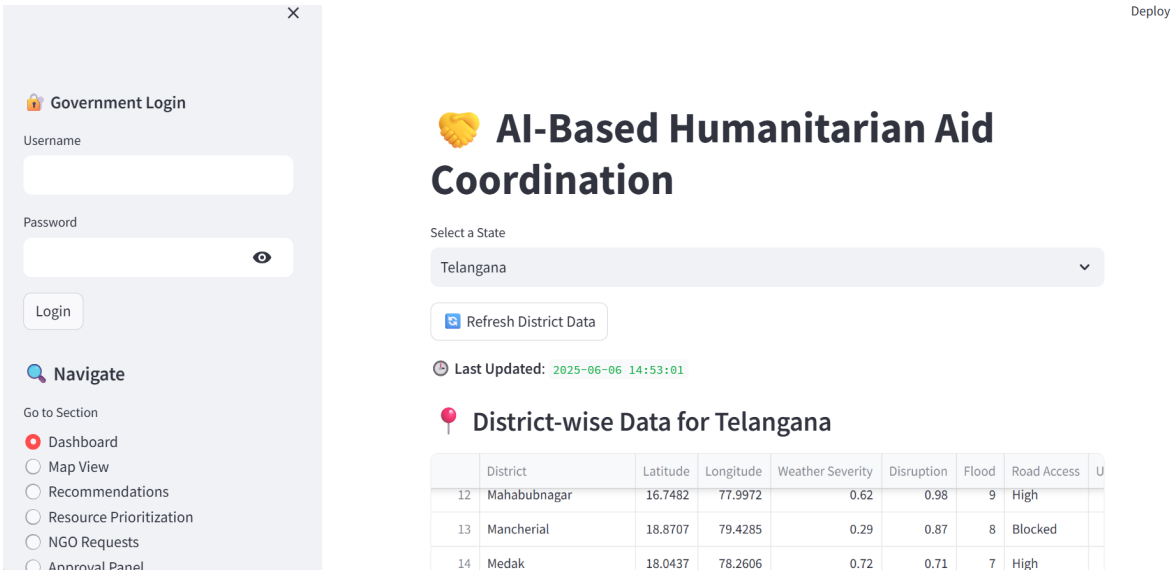


Fig. 6.1 Main Dashboard

Fig. 6.1 shows the main dashboard of the AI-Based Humanitarian Aid Coordination system. Upon login, government officials can select a state (e.g., Telangana) from the dropdown, refresh real-time or simulated data, and view district-wise urgency metrics. The left sidebar allows seamless navigation across various modules such as the map view, recommendations, resource prioritization, and the NGO approval panel.

	District	Latitude	Longitude	Weather Severity	Disruption	Flood	Road Access	Urgency Score	AI Recommendation	Population
0	Adilabad	19.6667	78.5333	0.22	0.85	8	Medium	0.54	⚠️ Urgent Support Required	14,559
1	Bhadradri Kothagudem	17.5532	80.641	0.85	0.73	7	Medium	0.6	⚠️ Urgent Support Required	16,660
2	Hyderabad	17.385	78.4867	0.25	0.82	8	High	0.54	⚠️ Urgent Support Required	17,868
3	Jagtial	18.7945	78.9166	0.46	0.96	9	Blocked	0.86	🚨 Immediate Deployment	15,389
4	Jangaon	17.7215	79.1522	0.33	0.99	9	Medium	0.64	⚠️ Urgent Support Required	11,576
5	Jayashankar Bhupalpally	18.4355	80.0617	0.16	0.88	8	Low	0.54	⚠️ Urgent Support Required	17,233
6	Jogulamba Gadwal	16.2326	77.8077	0.79	0.95	9	Medium	0.72	🚨 Immediate Deployment	18,416
7	Kamareddy	18.3206	78.3411	0.26	0.88	8	Blocked	0.76	🚨 Immediate Deployment	17,882
8	Karimnagar	18.4386	79.1288	0.88	0.82	8	Low	0.66	⚠️ Urgent Support Required	19,189
9	Khammam	17.2473	80.1514	0.55	0.94	9	Medium	0.67	⚠️ Urgent Support Required	6,062
10	Komaram Bheem	None	None	0.96	0.77	7	Blocked	0.84	🚨 Immediate Deployment	19,868
11	Mahabubabad	17.6078	80.0024	0.4	0.93	9	High	0.63	⚠️ Urgent Support Required	6,986
12	Mahabubnagar	16.7482	77.9972	0.46	0.79	7	Blocked	0.75	🚨 Immediate Deployment	18,840

Fig. 6.2 Urgency Analysis

Fig. 6.2 displays the district-wise urgency analysis table generated by the AI-Based Humanitarian Aid Coordination system. For each district, the table lists geospatial coordinates (latitude and longitude), weather severity, disruption index, flood level, and road access status. Using these parameters, the system calculates an overall Urgency Score**.**

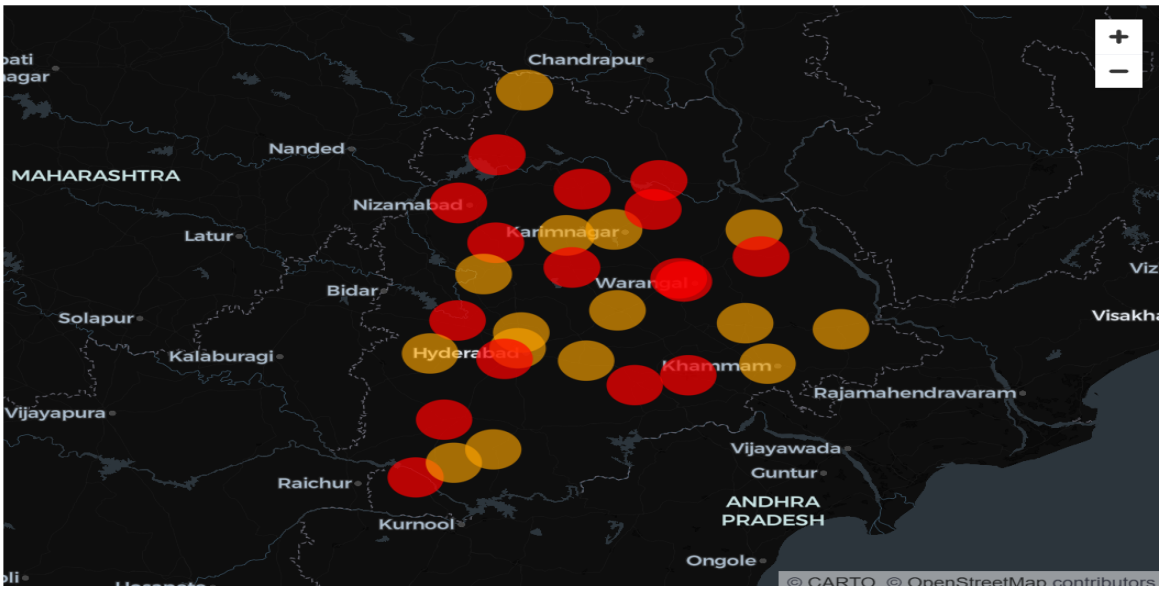


Fig. 6.3 District level Map

Fig. 6.3 showcases the interactive district-level map of Telangana generated using Pydeck in the AI-Based Humanitarian Aid Coordination system. Each circular marker represents a district, color-coded based on the AI-generated recommendation: red indicates Immediate Deployment, and orange denotes Urgent Support Required.

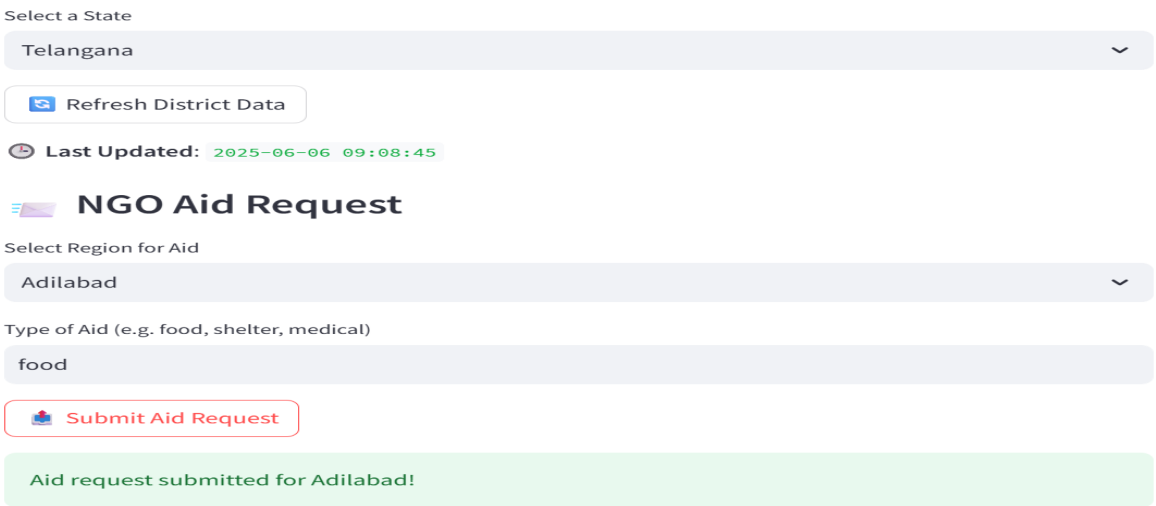


Fig. 6.4 NGO aid request

Fig. 6.4 illustrates the NGO Aid Request submission interface within the AI-Based Humanitarian Aid Coordination dashboard. Users can select a state and district, specify the type of aid required (e.g., food, shelter, medical), and submit the request.

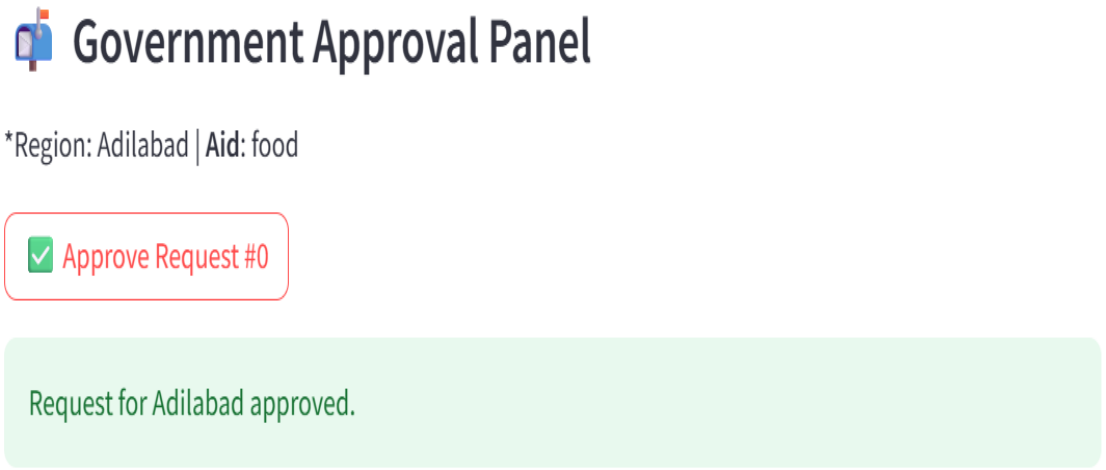


Fig. 6.5 Government approval panel

Fig. 6.5 depicts the Government Approval Panel in the AI-Based Humanitarian Aid Coordination system. It allows authorized officials to review and approve submitted NGO aid requests.

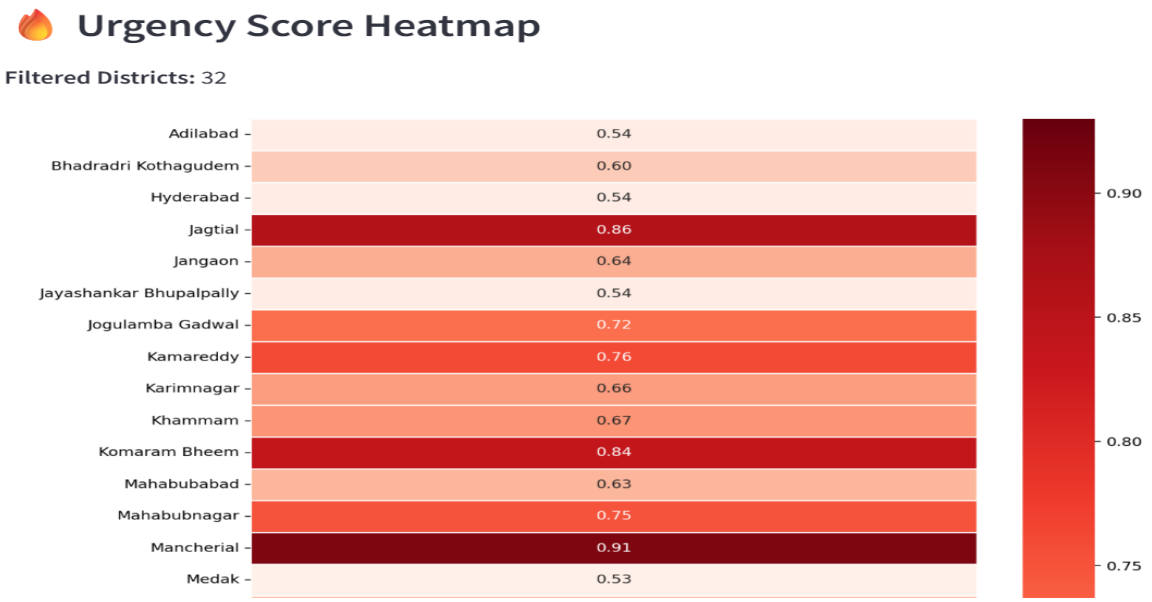


Fig. 6.6 Heatmap

Fig. 6.6 illustrates the Urgency Score Heatmap used in the AI-Based Humanitarian Aid Coordination system. It visually represents the urgency scores of various districts, with darker shades indicating higher urgency levels.

Fig. 6.7 Bar Chart

Fig. 6.7 displays the AI Recommendations Summary as a bar chart, categorizing districts based on their urgency level into two main recommendations: "⚠ Urgent Support Required" and "🚨 Immediate Deployment."

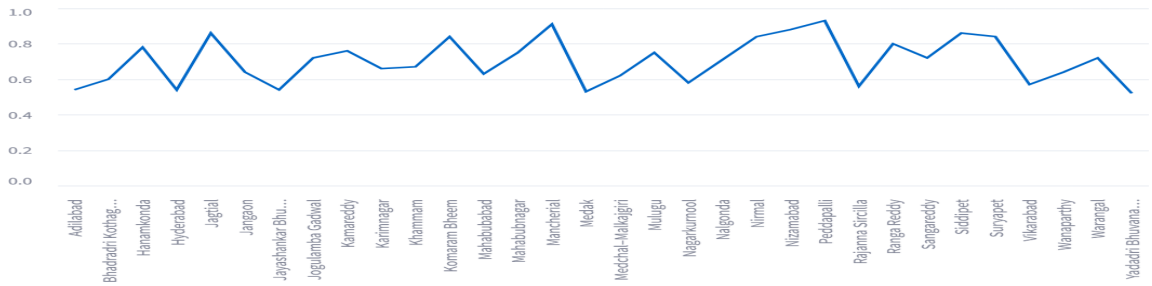


Fig. 6.8 Line chart visualisation

Fig. 6.8 displays the line chart visualization of urgency scores across various districts in Telangana. Each point on the chart represents a district's urgency score calculated by the system's rule-based algorithm.

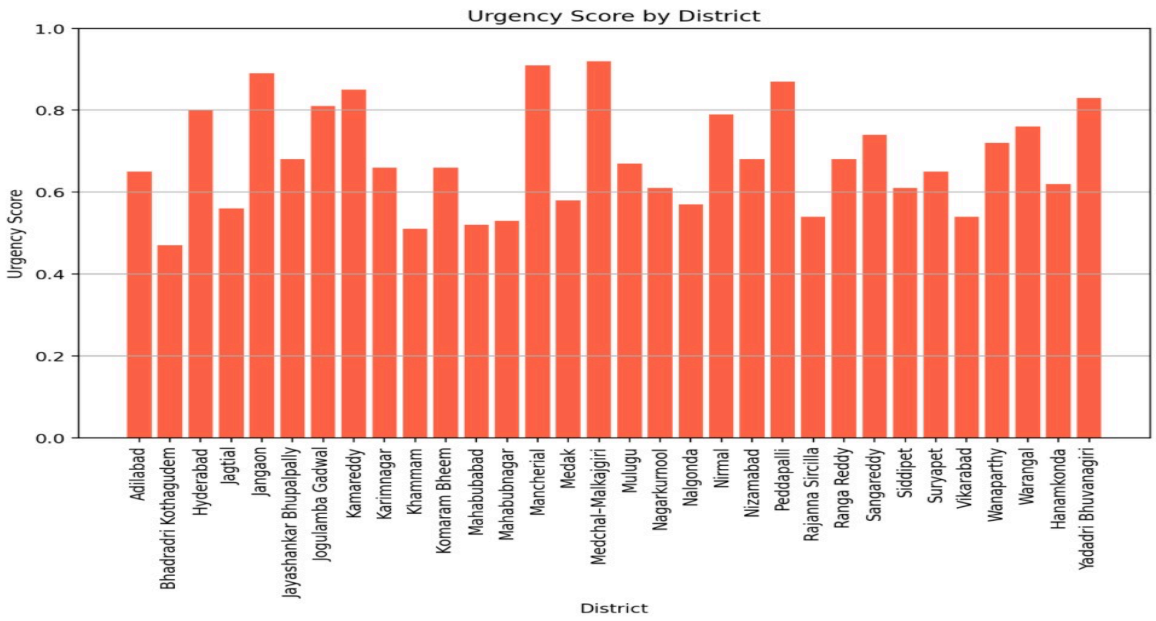


Fig. 6.9 Urgency Score bargraph

Fig. 6.9 showcases a bar chart representing the Urgency Score by District for the selected state. Each bar indicates the urgency score for a particular district, calculated based on weather severity, disruption level, flood impact, and road access status.

	District	Urgency Score	Population	Food Needed	Shelter Needed	Medical Needed
0	Adilabad	0.65	5,409	70	35	52
1	Bhadradri Kothagudem	0.47	16,132	151	75	113
2	Hyderabad	0.8	19,408	310	155	232
3	Jagtial	0.56	9,644	108	54	81
4	Jangaon	0.89	10,100	179	89	134
5	Jayashankar Bhupalpally	0.68	17,729	241	120	180
6	Jogulamba Gadwal	0.81	7,909	128	64	96
7	Kamareddy	0.85	5,294	89	44	67
8	Karimnagar	0.66	9,467	124	62	93
9	Khammam	0.51	18,051	184	92	138
10	Komaram Bheem	0.66	13,302	175	87	131
11	Mahabubabad	0.52	11,049	114	57	86
12	Mahabubnagar	0.53	13,654	144	72	108
13	Mancherial	0.91	6,349	115	57	86

Fig. 6.10 Resource prioritisation table

Fig. 6.10 illustrates the resource prioritization output generated by the system based on the urgency score and population of each district. It calculates and displays the estimated number of food units, shelter units, and medical kits required for every region.

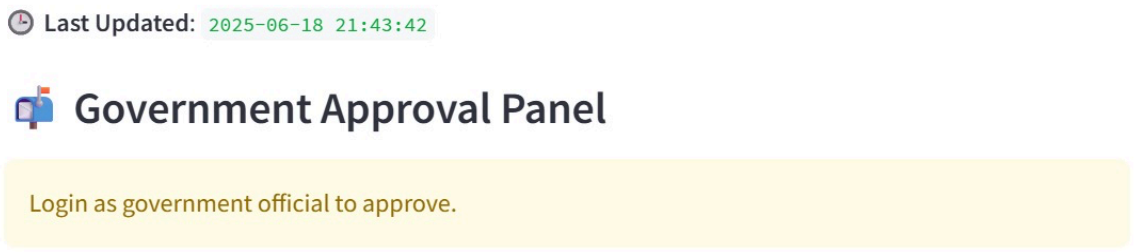


Fig. 6.11 Government approval panel

Fig. 6.11 displays a secure interface prompting authorized personnel to log in before approving aid requests. This ensures that only verified officials can access and approve submitted NGO requests, maintaining integrity and control over critical decision-making in aid distribution.

Fig. 6.12 Filter page

Fig. 6.12 This panel allows government officials to define total available resources for food, medical kits, and shelter units. Additionally, it includes an Urgency Filter dropdown that helps filter districts based on urgency levels such as “All,” “Urgent Support Required,” or “Immediate Deployment,” ensuring more targeted and effective allocation of aid supplies.

	District	Latitude	Longitude	Weather Severity	Disruption	Flood	Road Access	Urgency Score	AI Recommendation	Population
0	Mumbai	19.076	72.8777	0.01	0.77	7	Medium	0.45	Monitor Situation	7,964
1	Pune	18.5204	73.8567	0.17	0.92	9	Blocked	0.78	Immediate Deployment	16,241
2	Nagpur	21.1458	79.0882	0.83	0.78	7	Blocked	0.82	Immediate Deployment	5,202
3	Nashik	20.0059	73.7769	0.64	0.82	8	Blocked	0.82	Immediate Deployment	14,584
4	Aurangabad	19.8762	75.3433	0.91	0.81	8	Blocked	0.87	Immediate Deployment	14,979
5	Thane	19.2183	72.9781	0.8	0.8	8	Blocked	0.84	Immediate Deployment	18,294
6	Solapur	17.6599	75.9064	0.29	0.97	9	Low	0.63	Urgent Support Required	17,831
7	Amravati	20.9374	77.7796	0.04	0.8	8	High	0.49	Monitor Situation	17,983
8	Kolhapur	16.705	74.2433	0.88	0.85	8	Low	0.68	Urgent Support Required	7,997
9	Latur	18.4088	76.5604	0.19	0.85	8	Low	0.54	Urgent Support Required	6,502

Fig. 6.13 Urgency Analysis of Maharashtra

Fig. 6.13 The table displays dynamically generated data for selected districts in Maharashtra, showing key parameters like weather severity**,** flood level**,** road accessibility**,** and urgency scores**.**

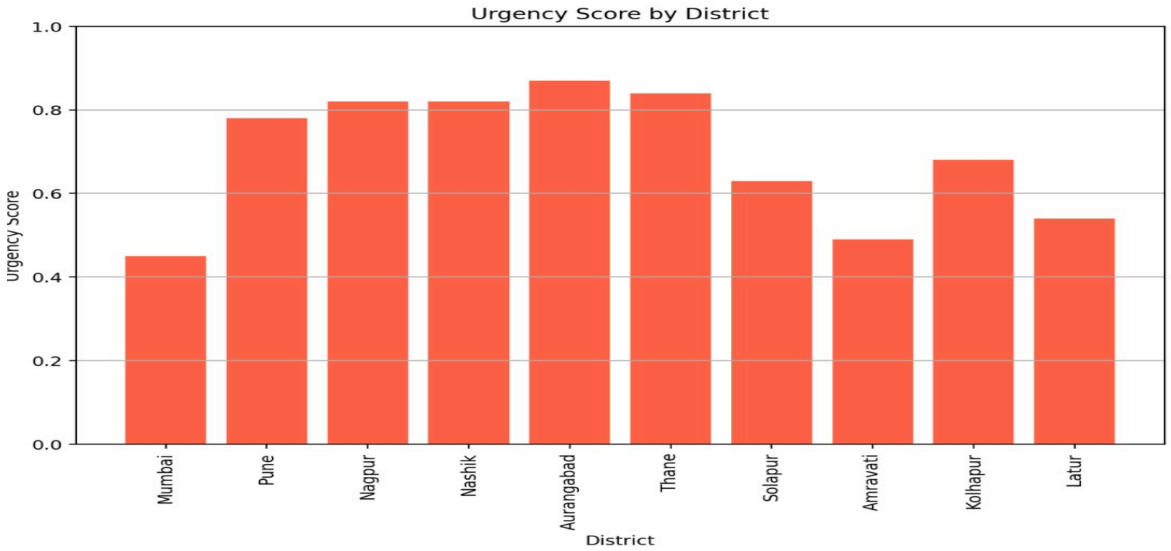


Fig. 6.14 Bar graph

Fig. 6.14. The bar graph visualizes Urgency Scores across various districts in Maharashtra. Higher urgency scores (e.g., Aurangabad, Thane) indicate critical need for aid deployment, while lower scores (e.g., Mumbai, Amravati) suggest lesser immediate risk.

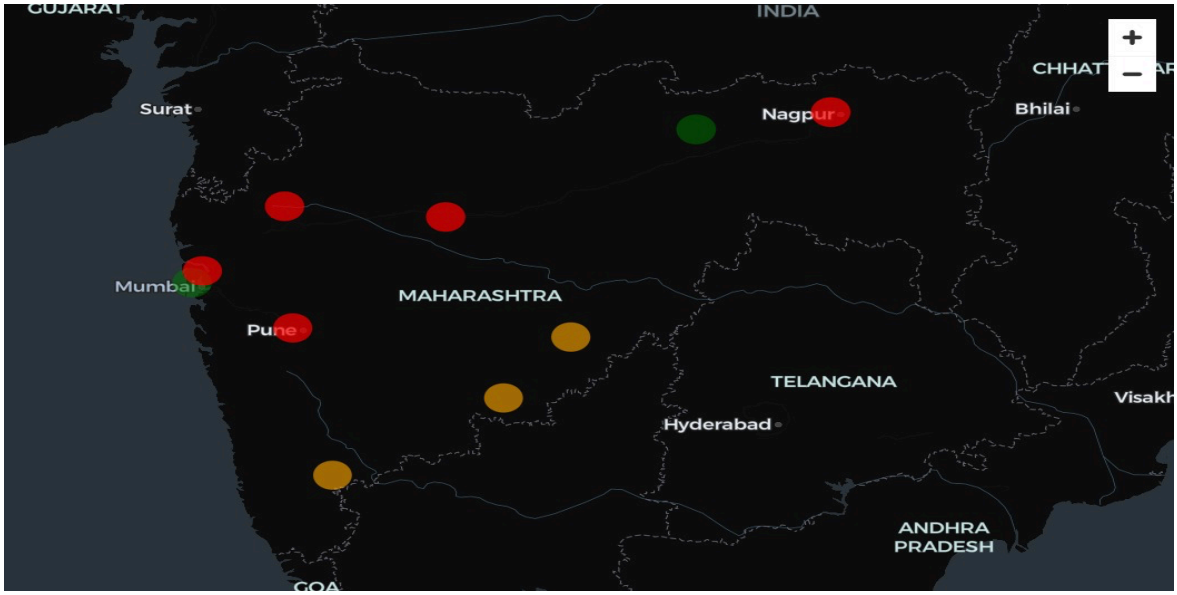


Fig. 6.15 Maharashtra map

Fig. 6.15. The district-level map of Maharashtra shows color-coded urgency markers based on real-time AI analysis. Red circles indicate Immediate Deployment zones, orange for Urgent Support Required, and green for regions under Monitor Situation.

Resource Prioritization

	District	Urgency Score	Population	Food Needed	Shelter Needed	Medical Needed
0	Mumbai	0.45	7,964	71	35	53
1	Pune	0.78	16,241	253	126	190
2	Nagpur	0.82	5,202	85	42	63
3	Nashik	0.82	14,584	239	119	179
4	Aurangabad	0.87	14,979	260	130	195
5	Thane	0.84	18,294	307	153	230
6	Solapur	0.63	17,831	224	112	168
7	Amravati	0.49	17,983	176	88	132
8	Kolhapur	0.68	7,997	108	54	81
9	Latur	0.54	6,502	70	35	52

Fig. 6.16 Prioritisation table

Fig. 6.16. This Resource Prioritization table shows the AI-calculated estimates of food, shelter, and medical aid needed across Maharashtra districts. The urgency score, derived using a weighted rule-based algorithm, dynamically adjusts resource quantities based on severity, disruption, and population.

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

The AI-Based Humanitarian Aid Distribution System effectively showcases the potential of data-driven crisis response. By leveraging district-level data such as weather severity, disruption, flood levels, and road accessibility, the system calculates urgency scores to prioritize resource allocation. It empowers government officials with a streamlined dashboard featuring choropleth maps, urgency heatmaps, and AI recommendations, ensuring real-time visibility and smarter decision-making.

Key features such as dynamic resource need estimation, NGO aid request submission, and a government approval panel enable end-to-end aid coordination with transparency. The system's integration with geospatial visualization tools further enhances situational awareness. Overall, it is an intuitive, scalable, and impactful solution that demonstrates how AI can revolutionize humanitarian logistics—minimizing delays, maximizing resource efficiency, and ultimately saving lives in crisis-affected areas.

7.2 FUTURE ENHANCEMENTS

Although the current version of the AI-Based Humanitarian Aid Distribution System is functional and effective in urgency detection, visualization, and resource prioritization, several enhancements can further improve its scalability, impact, and real-time responsiveness:

- **Granular Area Analysis**: Extend analysis capabilities beyond districts to include mandals or village-level granularity for hyper-localized aid assessment and distribution.
- **Mobile-Responsive Interface**: Develop a lightweight mobile version or native Android/iOS app to allow field agents and officials to access dashboards and approve requests remotely.

- **Advanced Predictive Models**: Incorporate more complex AI models (e.g., Gradient Boosting, LSTM) for better accuracy in forecasting future crises and resource demands based on historical and real-time data.
- **Real-Time Data Integration**: Enable dynamic ingestion of live feeds from IoT sensors, weather APIs, and satellite imagery for continuous monitoring and automated updates.
- **Multilingual Support**: Add support for multiple Indian languages to ensure inclusivity for NGO workers and officials across different regions.
- **Blockchain for Transparency**: Introduce blockchain-based tracking of aid flow and approvals to ensure transparency, prevent corruption, and build trust with stakeholders.

REFERENCES

[1] M. C. Ramesh et al., 2022. *"Deep Reinforcement Learning for Dynamic Resource Allocation in Humanitarian Crises,"* IEEE Access, vol. 10, pp. 32101–32112, 2022. doi: 10.1109/ACCESS.2022.3202113.

[2] Y. Zhang et al., 2023. *"Multi-Agent Systems for Coordinated Humanitarian Logistics,"* arXiv preprint, 2023. doi: 2302.05643

[3] R. Banerjee et al., 2021. *"AI-Driven Geospatial Analysis for Disaster Response using Satellite Imagery,"* Springer Humanitarian Tech Journal, 2021.

[4] A. Mohan et al., 2023. *"Forecasting Crisis Hotspots with Machine Learning: Reducing Aid Wastage,"* Nature Human Behaviour, vol. 7, no. 4, pp. 532–546, 2023. doi: 10.1038/s41562-023-01521-7

[5] M. Ali et al., 2023. *"Emergency Supply Chain Optimization using Federated Learning,"* ACM Transactions on Internet Technology (TOIT), vol. 23, no. 1, pp. 1–24, 2023. doi: 10.1145/3559447

[6] S. Wang et al., 2020. *"Real-time Disaster Response with Crowd-Sourced Geospatial Intelligence,"* International Journal of Disaster Risk Reduction, vol. 49, 101738. doi: 10.1016/j.ijdrr.2020.101738

[7] D. Kumar et al., 2022. *"Big Data-Enabled Relief Prioritization during Flood Emergencies,"* Elsevier Procedia Computer Science, vol. 199, pp. 1254–1263, 2022. doi: 10.1016/j.procs.2022.01.155