"Crisis Relief Logistics and Resource Management System"

A

Project Report

submitted in partial fulfillment of the
requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING

By

Name	***Roll Number***	***Course***
Gaurav Chhajer	*R2142220497*	*AIML H CSE*
Pranit Abraham Thomas	*R2142220130*	*AIML H CSE*
Pushp Prakhar Bhardwaj	*R2142220136*	*AIML H CSE*
Rishabh Sharma	*R2142220147*	*AIML H CSE*

under the guidance of
Ms. Silky Goel

School of Computer Science
University of Petroleum & Energy Studies
Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

November – 2024

CANDIDATE'S DECLARATION

We hereby certify that the project work entitled " **Crisis Relief Logistics and Resource Management System**" in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in AIML(Hons.) and submitted to the Department of Systemics, School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of our work carried out during a period from **August, 2024** to **November, 2024** under the supervision of **Ms.Silky Goel, Assistant Professor, Department of Cybernetics, School of Computer Science.**

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other University.

Gaurav Chhajer(R2142220497)
Pranit Abraham Thomas(R2142220130)
Pushp Prakhar Bhardwaj(R2142220136)
Rishabh Sharma(R21422202147)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: _____

Ms. Silky Goel
Project Guide

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Ms.Silky Goel**, for all advice, encouragement and constant support she has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank our respected **Prof. Vijaysekra Chellaboina, Head Department of SOCS**, for his great support in doing our project in Crisis Relief Logistics And Resource Management System(Minor-1).

We are also grateful to Dean SoCS UPES for giving us the necessary facilities to carry out our project work successfully. We also thank our Course Coordinator, Dr. Sandeep Chand Kumain and our Activity Coordinator Dr. Suneet Gupta for providing timely support and information during the completion of this project.

We would like to thank all our friends for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our parents who have shown us this world and for every support they have given us.

Name	Gaurav Chhajer	Pranit Abraham Thomas	Pushp Prakhar Bhardwaj	Rishabh Sharma
Roll No.	R2142220497	R2142220130	R2142220136	R2142220147

ABSTRACT

Natural disasters often create chaos and disruption, particularly in the coordination of relief efforts for those affected. One critical challenge is the lack of a unified platform for managing relief aid requests and deliveries. In many cases, requests for essential supplies such as food and water are made through social media or informal networks, where contact numbers are widely shared without regard to geographic boundaries. This can lead to significant confusion and inefficiencies, with delivery personnel overwhelmed by requests from areas they cannot service, resulting in wasted time and delayed aid for those in need.

This project proposes the development of a centralized, structured system for managing relief aid requests and distribution during natural calamities. The platform allows users to submit requests for specific items by selecting from a predefined list and specifying their location and contact information. Delivery drivers, registered within the system, can log in to view and manage requests from their designated zones only.

This targeted approach ensures that each driver is assigned requests within their service area, reducing confusion and improving the efficiency of aid distribution. Moreover, the system enables drivers to update the status of deliveries in real time, providing users with timely updates on the progress of their requests. This transparency and real-time tracking help to alleviate uncertainty and ensure a smoother flow of resources to those in need.

By streamlining the process of request submission and delivery, this platform aims to significantly enhance the effectiveness of disaster relief operations, ensuring that aid reaches the right people at the right time, and reducing the burden on both requesters and delivery personnel.

Keywords: Coordination, Logistics, Transportation, Efficiency, Tracking

TABLE OF CONTENTS

1. Introduction

Natural disasters such as floods, landslides, hurricanes, and earthquakes have long posed severe challenges to affected communities, often leaving them in dire need of essential supplies like food, water, medical aid, and shelter. The unpredictability and scale of these events make coordinated relief efforts crucial. However, the process of managing and distributing aid is often hampered by inefficiencies, particularly when informal networks and social media become the primary channels for communication and request submissions. In these scenarios, vital information is frequently spread in an uncoordinated manner, leading to miscommunication, delays, and, ultimately, the failure to deliver timely assistance to those in desperate need. The need for a more organized, efficient, and centralized system to manage these efforts has become increasingly evident, particularly as climate change intensifies the frequency and severity of natural calamities. The findings by the INSEAD Humanitarian Research Group (HRG) [13] outlines nine critical steps to consider when responding to a major disaster. Among these, step 5—Transport and Execution—emphasizes the pivotal role of transportation in delivering aid at the right time and to the right place highlighting that transportation can involve various modes, such as planes, trucks, cars, boats, and even animals when necessary. However, the effectiveness is heavily influenced by various conditions such as the types of vehicles used depending on the accessibility of routes to the disaster zone and the distance from urban centers capable of providing assistance. INSEAD underscores the importance of leveraging all available resources, including humanitarian organizations, volunteers, and private sector partners, to optimize the logistics of aid delivery.

The significance of this issue is deeply felt by our team, driven in part by the personal experiences of one of our members. Kerala, a state in southern India regularly faces the wrath of nature, particularly during the monsoon season in July and August. Kerala's landscape, while lush and beautiful, is highly susceptible to flooding and landslides, which have caused widespread devastation in recent years. The tragedy of a recent landslide in Wayanad, a region in Kerala, has only reinforced the urgency of addressing these challenges. The United Arab Emirates also faced an unusual and severe flooding event in May. During this time, their family played an active role in the relief efforts, using their own vehicle to deliver

aid. However, the experience was marred by confusion and inefficiencies due to the lack of a streamlined system, making it clear that a more organized approach was necessary to improve disaster response.

In response to these challenges, our project aims to develop a comprehensive and centralized platform for managing relief aid requests and delivery during natural disasters. The proposed system will allow users to submit detailed requests for essential items, such as food, water, and medical supplies, directly through the platform. These requests will include critical information such as location, contact details, and specific needs, ensuring that they are directed to the appropriate relief zones. Delivery drivers, who are also integrated into the system, will only receive requests relevant to their assigned areas, significantly reducing the chances of miscommunication and overburdening. This targeted approach not only enhances the efficiency of aid distribution but also allows for real-time tracking of deliveries, providing transparency and peace of mind to those in need. The application of this system has the potential to revolutionize disaster relief efforts, offering a reliable, organized, and scalable solution that can be adapted to various regions and types of disasters, ultimately saving lives and alleviating the suffering of affected communities.

1.1 History

The need for a structured disaster relief management system arises from repeated instances of inefficiencies during natural disasters like floods, earthquakes, and landslides. Historically, relief efforts have been hindered by uncoordinated communication channels, such as social media and informal networks, leading to unmet needs, confusion, and delays. The tragic landslides in Kerala and severe floods in the UAE underscored the critical gaps in disaster management. These events highlighted the necessity for a well-organized platform capable of managing requests and streamlining relief distribution efficiently.

1.2 Main Objective

The primary objective of this project is to design a centralized platform for managing and distributing relief resources efficiently during natural calamities. By

ensuring effective coordination between users submitting aid requests and delivery personnel, the system aims to:

- Minimize response time in disaster-stricken areas.
- Optimize the allocation of available resources.
- Enhance the transparency of aid delivery through real-time tracking.

1.3 Sub Objective

To achieve the main objective, the project focuses on the following sub-objectives:

1. **Request Management:**
   Enable users to submit detailed requests for essential supplies with location and contact information.
2. **Geographical Allocation:**
   Assign requests to delivery personnel based on predefined geographical zones, ensuring requests are manageable and reducing miscommunication.
3. **Real-Time Tracking:**
   Develop a mechanism for delivery personnel to update the status of aid deliveries, providing users with real-time progress updates.
4. **User-Friendly Interfaces:**
   Design intuitive interfaces for both users and delivery personnel, ensuring seamless interaction with the platform.
5. **Priority-Based Sorting:**
   Implement sorting algorithms to prioritize requests based on urgency and predefined criteria like medical needs or time sensitivity.

1.4 Pert Chart

To ensure the systematic and timely completion of this project, a **Program Evaluation Review Technique (PERT) chart** has been developed. The PERT chart outlines key project activities, their dependencies, and estimated durations to monitor progress effectively.

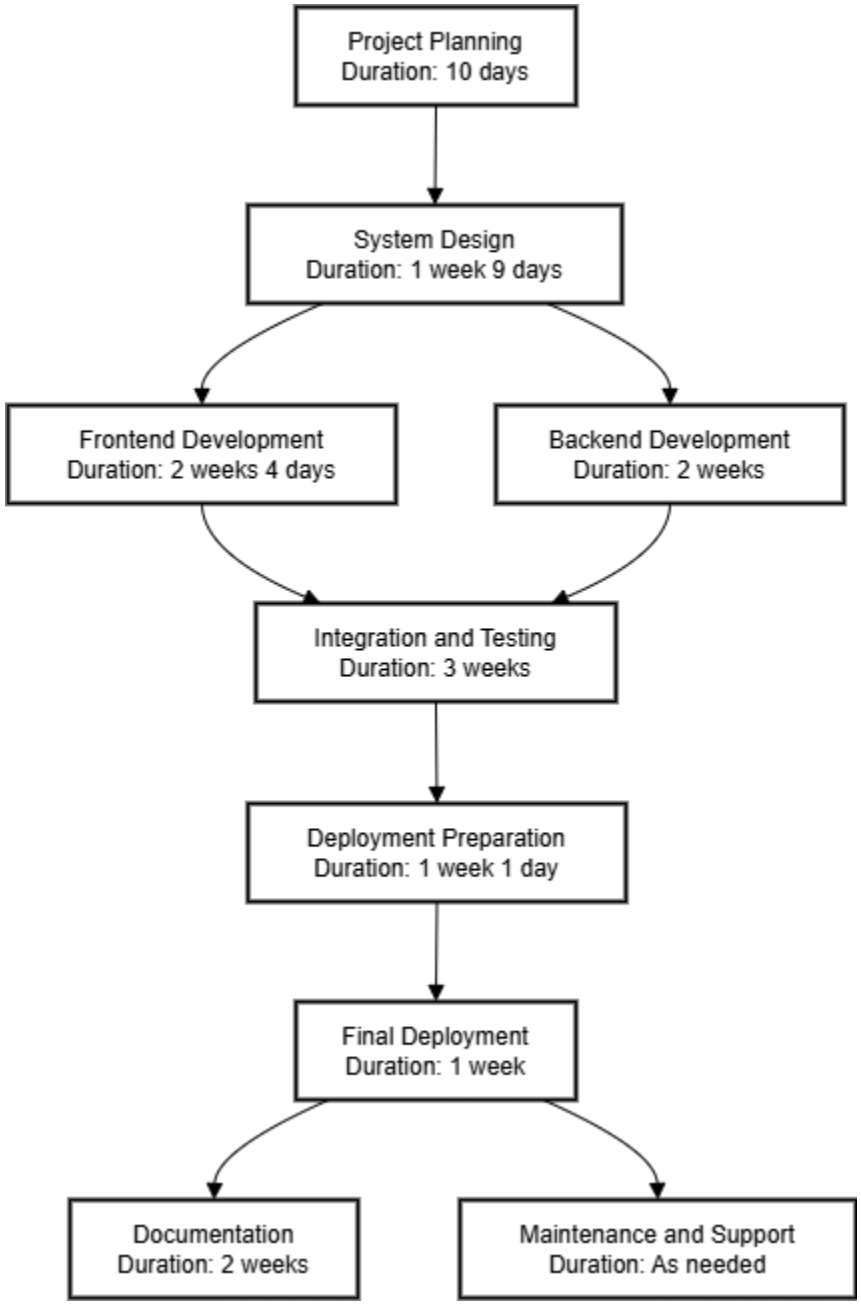


Fig. 1

2. System Analysis

2.1 Motivation

Efficient disaster relief and resource management are critical in mitigating the impacts of natural and man-made disasters. Traditional approaches to managing resources, allocating aid, and optimizing delivery routes often face challenges such as:

- **Delays in Resource Allocation:** In emergencies, the inability to quickly allocate and transport resources to affected areas can exacerbate the crisis.
- **Inefficient Communication:** Lack of a centralized system to handle requests and track resource availability often leads to mismanagement.
- **Suboptimal Routing:** Delivery routes are often determined manually, which may result in increased delivery times and costs.
- **Poor Role-Based Accessibility:** Existing systems do not cater effectively to diverse roles such as admin, drivers, and general users.

To address these challenges, we propose a robust and modular system that leverages technology to improve disaster relief operations, enhance resource allocation, and streamline delivery logistics.

2.2 Proposed System

Our proposed system is a **Disaster Relief Management System** that integrates a central database, efficient algorithms, and user-friendly interfaces to ensure seamless resource management, request handling, and delivery coordination. Key features of the system include:

3. **Centralized Database:** Maintains comprehensive data on resources, warehouses, disaster events, requests, and users.
4. **Role-Based Access:** Tailors the user experience and access to functionalities based on roles such as admin, driver, and user.
5. **Automated Resource Allocation:** Optimizes resource distribution using efficient algorithms like priority queue and 0/1 knapsack.
6. **Route Optimization:** Ensures quick delivery of resources using Dijkstra's algorithm for shortest path routing.
7. **Real-Time Request Handling:** Tracks and updates the status of requests dynamically, improving response times.

2.3 Modules

The proposed system is divided into several key modules, each handling a specific aspect of disaster management:

2.3.1 Resource Management Module

Functionality:

a. Tracks available resources in warehouses, including quantities, categories, and expiration dates.
b. Provides interfaces for admins to add, update, or delete resources.
c. Allocates resources to requests based on availability and priority.

Database Interaction:

d. Fetches data from the **resources** and **warehouses** tables.
e. Updates the **resource allocation** and **requests** tables upon allocation.

2.3.2 Request Management Module

Functionality:

- Allows users to submit requests for resources, specifying quantity and location.
- Tracks the status of requests (e.g., pending, completed, or canceled).
- Prioritizes requests based on urgency or disaster severity.

Database Interaction:

- Uses the requests table to log and manage user-submitted requests.
- Dynamically updates the status and allocation details.

2.3.3 Disaster Event Management Module

Functionality:

- Logs and monitors disaster events, categorizing them by severity and affected zones.
- Links disaster events to specific resource requirements for prioritization.

Database Interaction:

- Pulls data from the disaster events table to determine resource allocation priorities.

2.3.4 Driver Module

Functionality:

- Displays assigned requests to drivers with details like requested resources, quantities, and delivery locations.
- Integrates maps for navigation using latitude and longitude data.
- Allows drivers to mark requests as completed.

Database Interaction:

- Updates the requests and shipments tables upon delivery completion.

2.3.5 Role and Permissions Management Module

Functionality:

- Manages access to system features based on user roles (e.g., admin, driver, user).
- Ensures that only authorized users can perform critical operations like resource updates or disaster event logging.

Database Interaction:

- Uses the roles, permissions, and role permissions tables to enforce access control.

2.3.6 Routing and Map Integration Module

Functionality:

- Calculates the shortest delivery route using Dijkstra's algorithm.
- Displays interactive maps for drivers, showing delivery destinations and optimized routes.

Database Interaction:

- Uses latitude and longitude data from the warehouses and requests tables.

3. Design

3.1 Database Design

The database has the following main tables:

- **Audit Log**: Records user actions with fields like log ID, user ID, action, date, and details.
- **Disaster Events**: Contains information about disasters, including event ID, disaster zone, and severity.
- **Donations**: Tracks donations with donor name, resource ID, quantity, and donation date.
- **Permissions**: Manages role permissions with a permission ID and description.
- **Requests**: Stores resource requests with details like request ID, resource ID, quantity, date, and status.
- **Resources**: Keeps inventory information such as resource name, category, quantity, and location.
- **Roles**: Defines roles like admin, driver, and user.
- **Shipments**: Logs shipments with details like shipment ID, resource ID, locations, and status.
- **Users**: Stores user details, including name, role, and contact information.
- **Warehouses**: Contains warehouse locations with names, latitude, longitude, and address.

These tables organize and manage data for resources, requests, users, and logistics effectively.

3.2 Module Architecture

The system is designed with a modular architecture that ensures seamless interaction between components, enabling efficient functionality, scalability, and maintenance. Each module plays a specific role, and their interactions ensure data consistency and user-friendly operation.

Core Modules and Their Interactions

1. **Database to UI Integration**
    - The database acts as the central data repository, providing the UI with real-time information and storing user interactions.
    - Key interactions include:
        - **Resources Table**:

        - Stores all available resources, including details like category, quantity, expiration date, and current location.
        - This data is displayed dynamically on the request page for users to browse and request resources.
    - **Warehouses Table**:
        - Contains geolocation data (latitude and longitude) for warehouses, which integrates with map services.
        - Enables drivers to view warehouse locations and navigate efficiently.
    - **Requests Table**:
        - Tracks all user-submitted requests, storing information like requested resources, quantities, and status (e.g., Pending, Approved, Delivered).
        - Updates dynamically as users modify or cancel requests or as drivers mark them as completed.
    - **Roles and Permissions Tables**:
        - Ensures role-based access control (RBAC).
        - Determines the UI components accessible to different users:
            - **Admins**: Access dashboards for managing resources, donations, shipments, and disaster events.
            - **Drivers**: View delivery assignments and navigation tools.
            - **Users**: Submit and track resource requests.
2. **Longitude and Latitude Usage**

    - **Map Integration**:
        - Data from the warehouses table is used to pinpoint warehouse locations on an interactive map.
        - Drivers can view precise locations and routes to their delivery destinations.
    - **Route Optimization**:
        - The system calculates the shortest route between warehouses and delivery locations using geospatial data and mapping algorithms.
        - This helps drivers reduce travel time and fuel consumption.
3. **Request Management Module**

    - Handles user-submitted requests:

        - Collects data from the UI and stores it in the requests table.
        - Validates requests to ensure availability of the requested quantity.
        - Notifies admins and drivers of new or updated requests.
    - Integrates with the roles and permissions module to ensure only authorized users can manage requests.
4. **Resource Management Module**

    - Accessible primarily by admins for managing inventory:
        - Updates resource quantities based on donations, allocations, and shipments.
        - Notifies users when specific resources are restocked or unavailable.
5. **Shipment and Delivery Management Module**

    - Assigns delivery tasks to drivers based on requests.
    - Updates the database with shipment details, including departure and arrival times, quantities, and statuses.
    - Drivers interact with this module to mark deliveries as completed or report issues.
6. **Disaster Management Module**

    - Admins log and manage disaster events:
        - Records severity, location, and allocated resources.
        - Updates allocations in the resource allocation table to prioritize disaster zones.

Flow of Interaction

1. **User Initiates Action**:

    - A user submits a resource request via the UI.
    - The system validates the request, checks availability, and stores it in the requests table.
2. **Database Updates**:

    - The request triggers updates in related tables (e.g., reducing available quantities in the resources table).
    - Admins are notified of new requests requiring approval or resource allocation.

3. **Driver Fulfillment**:
    - Once a request is approved, a driver is assigned.
    - Drivers access the delivery details through the UI, using the map for navigation and marking requests as completed.
4. **Admin Oversight**:
    - Admins monitor all system activities, including resource movements, disaster allocations, and delivery statuses.

This modular architecture ensures that each component functions independently while maintaining a cohesive flow of information and interaction across the system.

3.3 User Interface Design

The user interface is structured to provide an intuitive and role-specific experience, ensuring that users, drivers, and admins can seamlessly interact with the system. Below are the key features tailored to each role:

General Features

1. **Hamburger Menu**:
    - A collapsible menu accessible from any page for easy navigation.
    - Includes options such as:
        - **Logout**: To securely exit the system.
        - **Emergency Services**: Quick access to emergency contact information or actions.
        - **Requests**: A link to the page where users can view and submit resource requests.
2. **Request Page**:
    - A central interface where users can browse the list of available resources.
    - Features include:
        - **Resource Listings**: Displays resources with details such as name, category, available quantity, and unit.
        - **Request Form**: Users can select a resource, specify the quantity they need, and submit a request.

        - **Request Status**: Users can track the status of their submitted requests (e.g., Pending, Approved, Delivered).

Driver-Specific Features

1. **Requests Overview**:
    - A dedicated section displaying a list of assigned delivery requests.
    - Information includes:
        - The requested resources and their quantities.
        - Delivery location details, including address, city, and map coordinates.
        - Status of each request (e.g., Assigned, In Progress, Delivered).
2. **Interactive Buttons for Requests**:
    - **Map Button**:
        - Opens a map interface that highlights the delivery location using the latitude and longitude from the database.
        - Provides route navigation to the location.
    - **Contact Button**:
        - Opens a communication interface to call or message the requester for clarifications or updates.
    - **Done Button**:
        - Marks the request as delivered in the database.
        - Triggers an update in the "Requests" table, changing the status to "Completed" and recording the delivery timestamp.
3. **Delivery Schedule**:
    - Drivers can view their upcoming and completed deliveries in a calendar-like format for better planning.

Admin Features

1. **Resource Management**:
    - Admins have access to a dashboard where they can:
        - **Add New Resources**: Input details like name, category, initial quantity, and location.
        - **Update Existing Resources**: Modify quantities, expiration dates, or locations based on changes.

        - **Delete Resources**: Remove outdated or unnecessary items from the database.
2. **Donation and Shipment Tracking**:
    - Admins can monitor and manage donations, including donor details, donation dates, and resource quantities.
    - Track shipment statuses, ensuring that deliveries are completed on time and updating the shipment table accordingly.
3. **Disaster Event Management**:
    - Admins can log new disaster events, categorize severity, and allocate resources efficiently to affected zones.
4. **User Role Assignment**:
    - Admins can assign and manage roles for users, ensuring appropriate access and permissions for different functionalities.

This design ensures role-specific efficiency, enabling seamless communication, resource management, and request handling for all users.

4. Implementation

This section outlines the progress made so far in the development of the system, detailing the components implemented and their current status.

4.1 Algorithms

The system implements several core algorithms to manage resource allocation, route optimization, and decision-making processes efficiently. Below is a detailed description of the key algorithms and their implementation:

4.1.1 Resource Allocation Algorithm

Resource allocation plays a crucial role in ensuring that the right resources are delivered to the right location, based on user requests and resource availability. The resource allocation algorithm takes the following factors into account:

- **Available Resources:** The system checks if the requested resources are available in the inventory (from the **resources** table).
- **Event Priority:** Resources may be prioritized based on disaster event severity (from the **disaster events** table).

- **Request Status:** Requests can be in various states like "Pending," "Completed," or "Cancelled." The system only processes "Pending" requests.

Algorithm Details:

- **Step 1:** The system first checks the **requests** table for pending requests. Each request is linked to a specific **resource** and the requested quantity.
- **Step 2:** The system queries the **resources** table to check for available resources in the inventory. If sufficient quantity is available, the allocation proceeds. If not, the request is marked as "Cancelled."
- **Step 3:** The resource allocation system matches the requested resources with available ones, allocating them in the **resource allocation** table, and updating the request status to "Completed."
- **Step 4:** In case multiple resources are needed from different locations, the algorithm determines the most efficient warehouse for each request based on the **warehouses** table.

4.1.2. Dijkstra's Algorithm for Route Optimization

Dijkstra's algorithm is used for finding the shortest path in a graph, which, in this system, translates to calculating the most efficient route for deliveries from warehouses to requested locations. Given that warehouses are geographically distributed, Dijkstra's algorithm ensures that delivery drivers are assigned optimal routes.

Algorithm Details:

- **Step 1:** Each warehouse and delivery location is treated as a node in a graph. The edges between nodes represent the distance or time required to travel between locations. The **latitude** and **longitude** from the **warehouses** table are used to calculate the geographical distance between different nodes (warehouses and delivery locations).
- **Step 2:** Dijkstra's algorithm starts from the warehouse location (starting node) and explores all neighboring nodes, progressively calculating the shortest path to each warehouse or destination node based on the distance between them.
- **Step 3:** The algorithm iteratively updates the shortest path for each node, considering all possible paths and distances. Once the algorithm finds the shortest route from the warehouse to the delivery location, it assigns this route to the delivery driver.

- **Step 4:** The route data is passed to the UI, where it is shown on the map. The **Map button** in the driver's interface will allow the driver to view the shortest route and begin the journey.
- **Complexity:** The time complexity of Dijkstra's algorithm is O(E + V log V) where E is the number of edges and V is the number of vertices. The algorithm is efficient and well-suited for real-time applications like delivery optimization.

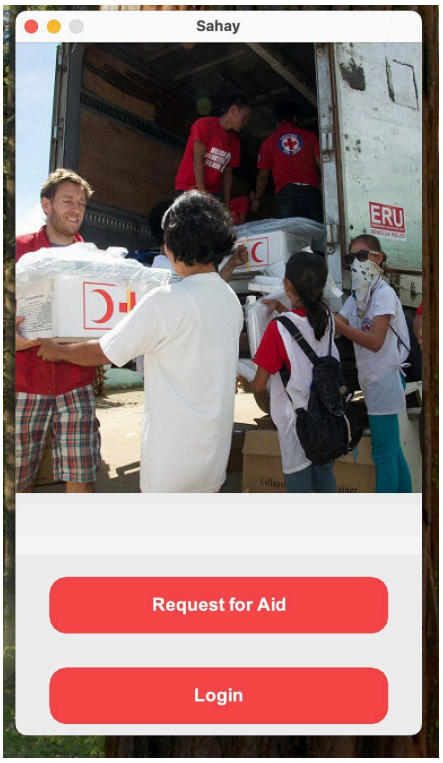
4.1.3. Priority Queue for Request Management

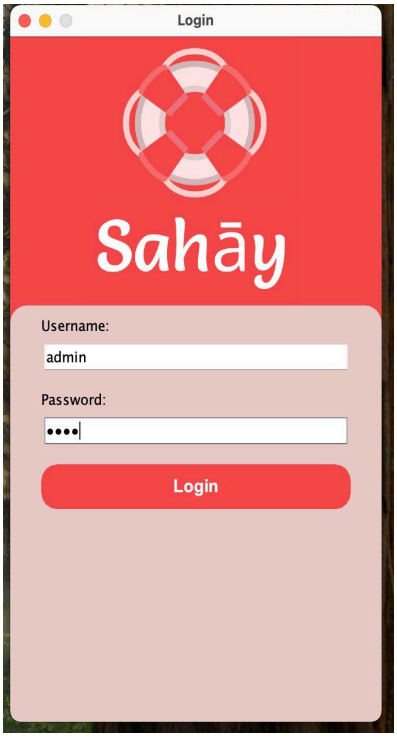
A **priority queue** is used to manage requests based on their priority, ensuring that urgent requests (such as those related to disaster events) are processed first. In this system, requests are assigned a priority based on factors like the severity of the disaster or the criticality of the requested resource.

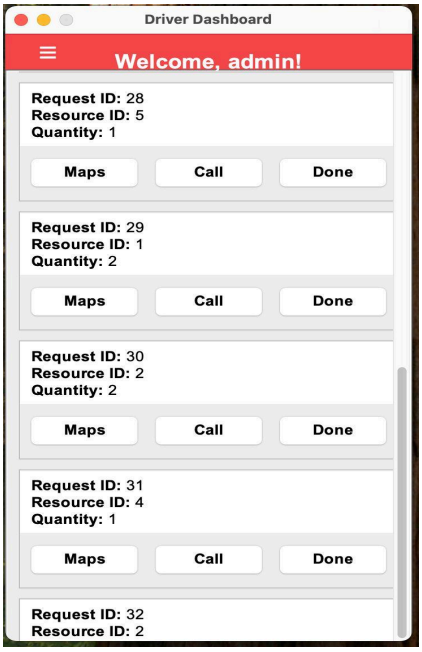
Algorithm Details:

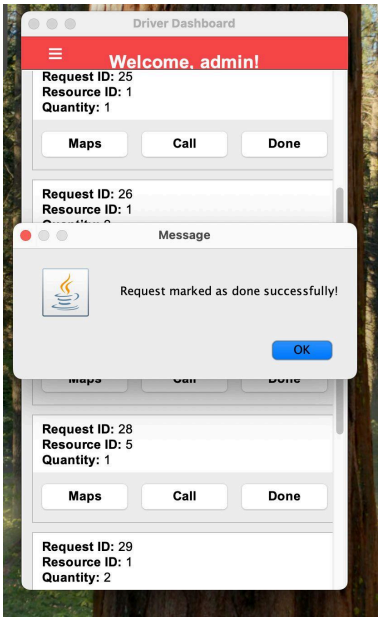
- **Step 1:** Each request in the **requests** table is assigned a priority score. Requests related to **disaster events** are given higher priority (e.g., a severity score from 1 to 10) than regular requests.
- **Step 2:** The priority queue uses this priority score to order requests, ensuring that the highest-priority requests are processed first. The queue is implemented as a heap data structure, which allows efficient insertion and extraction of the highest-priority request.
- **Step 3:** As new requests are submitted, they are added to the priority queue. The system then continuously processes requests, starting with the highest-priority ones. This ensures that critical resources are allocated quickly.
- **Step 4:** Once a request is fulfilled (resource allocated or delivered), it is removed from the priority queue, and the system updates the status of the request in the database.
- **Complexity:** The insertion and removal of elements in a priority queue (implemented as a binary heap) have a time complexity of O(log n), which makes it an efficient choice for managing dynamic, high-priority data.

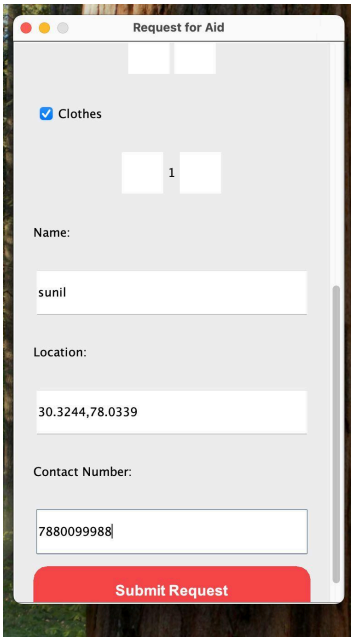
5. Output Screens

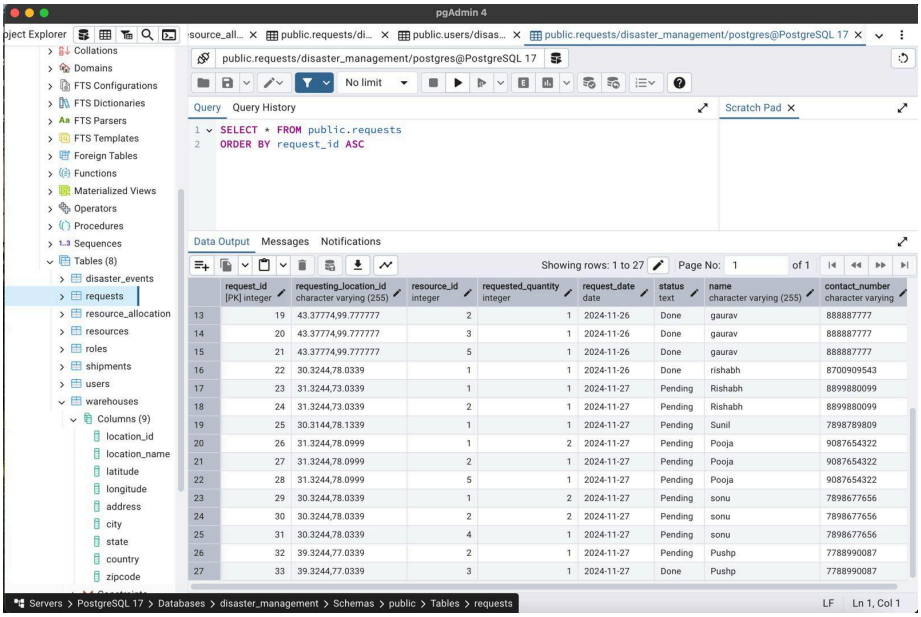


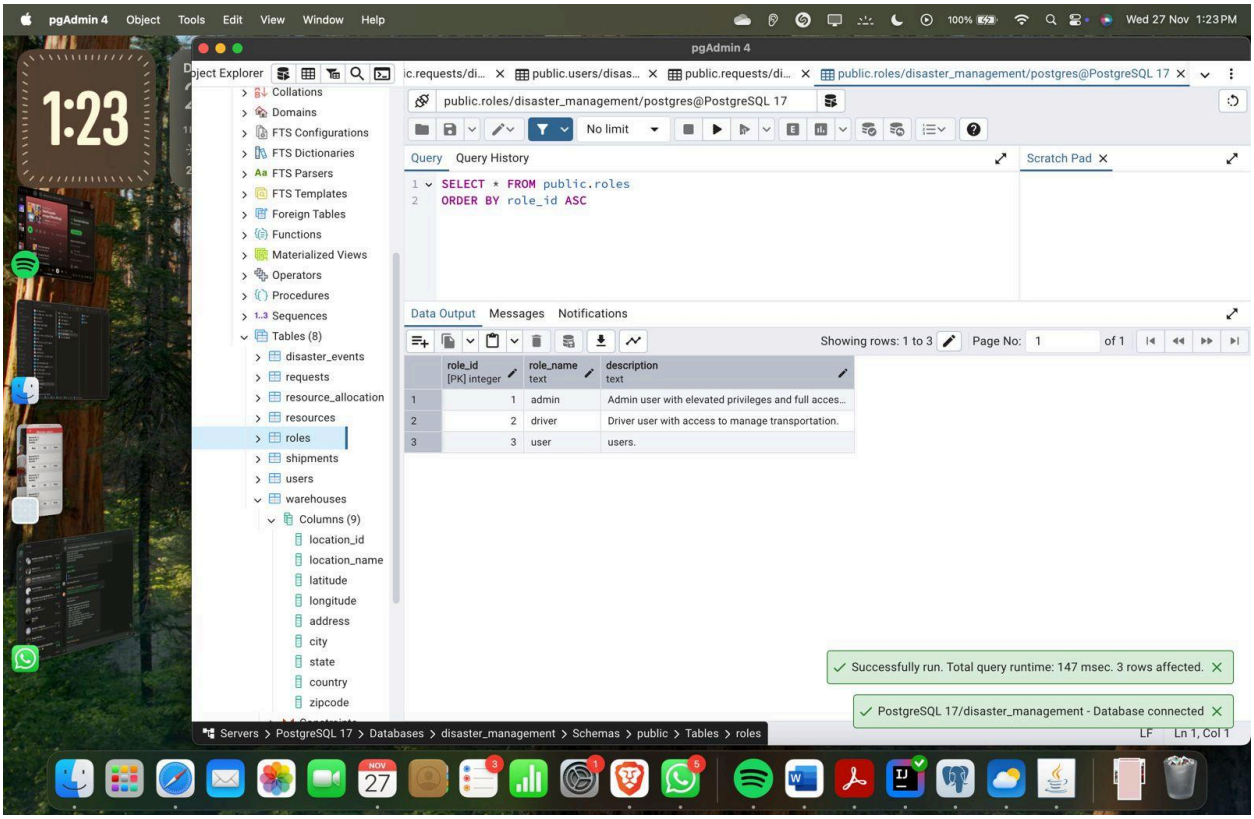


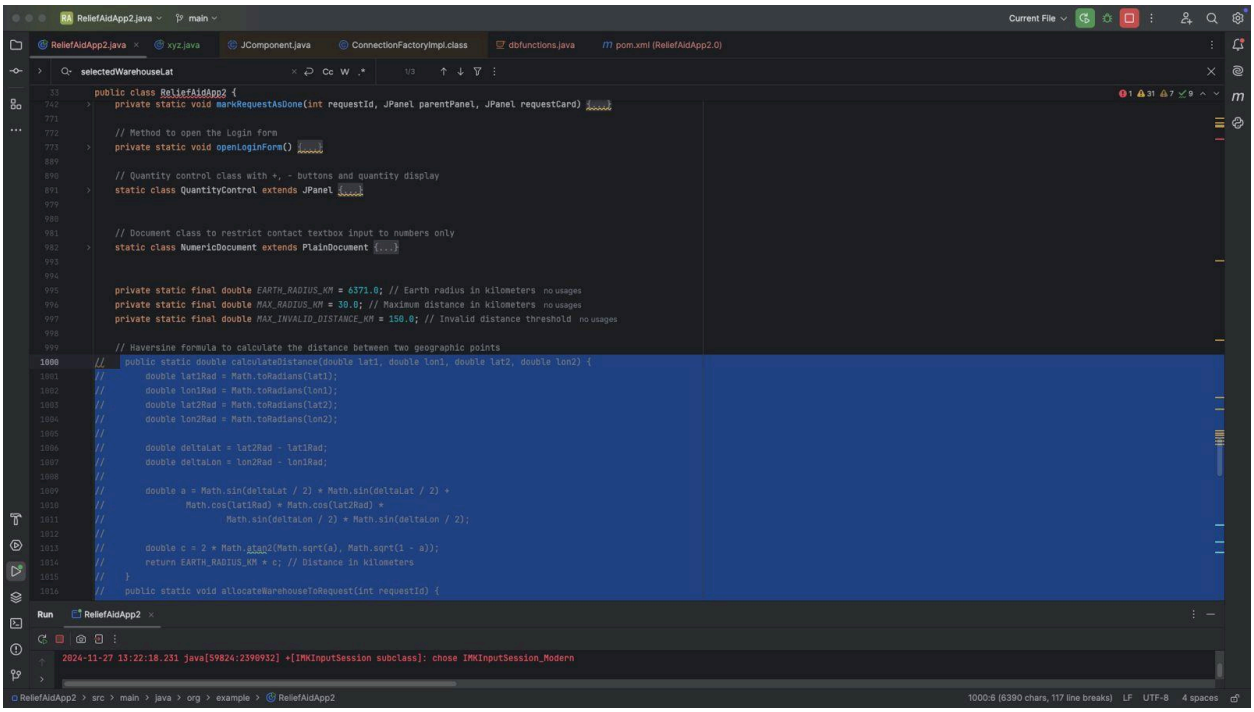


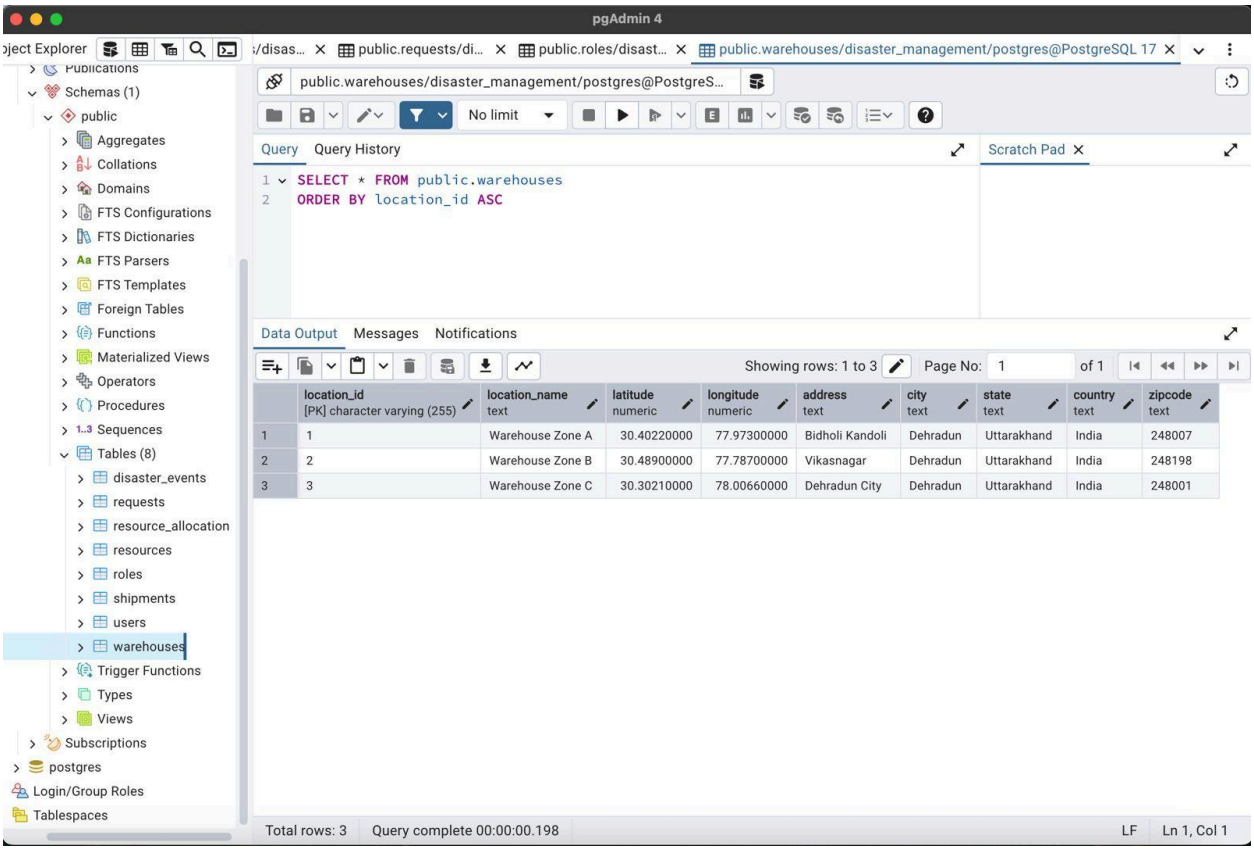


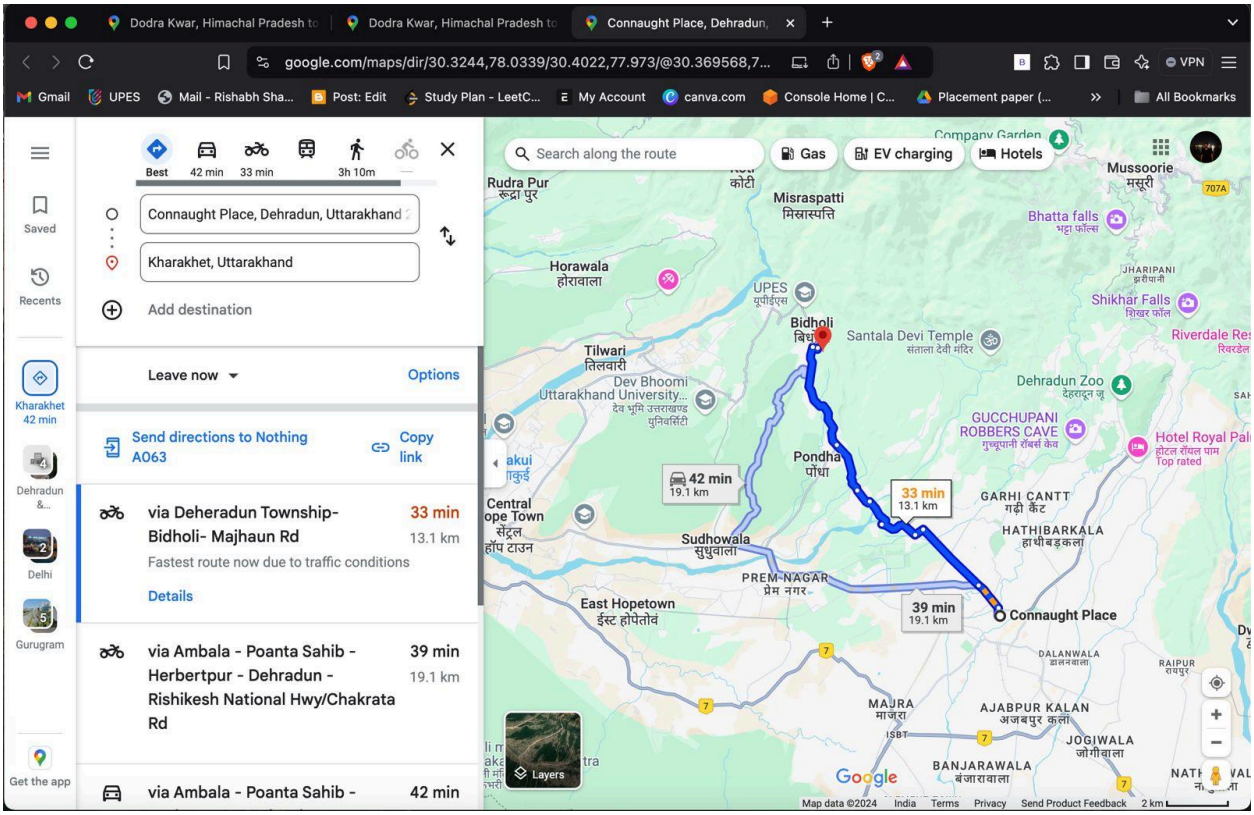

Request for Aid
Clothes
1
Name:
sunil
Location:
30.3244,78.0339
Contact Number:
7880099988
Submit Request

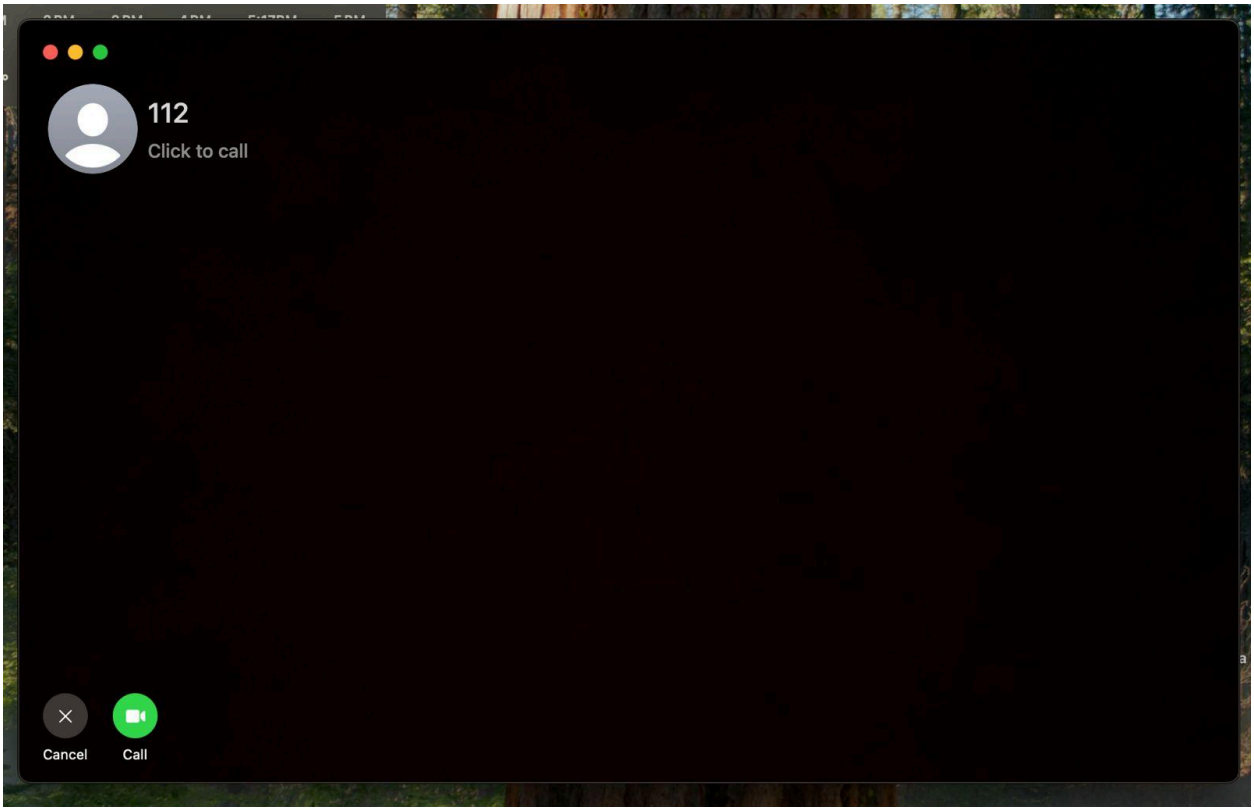

pgAdmin 4
public.requests/disaster_management/postgres@PostgreSQL 17
SELECT * FROM public.requests
ORDER BY request_id ASC
Data Output
Messages
Notifications
Showing rows: 1 to 27
Page No: 1 of 1
Servers > PostgreSQL 17 > Databases > disaster_management > Schemas > public > Tables > requests


pgAdmin 4
public.roles/disaster_management/postgres@PostgreSQL 17
SELECT * FROM public.roles
ORDER BY role_id ASC
Data Output
Messages
Notifications
Showing rows: 1 to 3
Page No: 1 of 1
Successfully run. Total query runtime: 147 msec. 3 rows affected.
PostgreSQL 17/disaster_management - Database connected
Servers > PostgreSQL 17 > Databases > disaster_management > Schemas > public > Tables > roles


ReliefAidApp2.java
main
Current File
ReliefAidApp2.java
xyz.java
JComponent.java
ConnectionFactoryImpl.class
dbfunctions.java
pom.xml (ReliefAidApp2.0)
selectedWarehouseLat
public class ReliefAidApp2 {
private static void markRequestAsDone(int requestId, JPanel parentPanel, JPanel requestCard) {...}
// Method to open the Login form
private static void openLoginForm() {...}
// Quantity control class with +, - buttons and quantity display
static class QuantityControl extends JPanel {...}
// Document class to restrict contact textbox input to numbers only
static class NumericDocument extends PlainDocument {...}
private static final double EARTH_RADIUS_KM = 6371.0; // Earth radius in kilometers
private static final double MAX_RADIUS_KM = 30.0; // Maximum distance in kilometers
private static final double MAX_INVALID_DISTANCE_KM = 150.0; // Invalid distance threshold
// Haversine formula to calculate the distance between two geographic points
// public static double calculateDistance(double lat1, double lon1, double lat2, double lon2) {
// double lat1Rad = Math.toRadians(lat1);
// double lon1Rad = Math.toRadians(lon1);
// double lat2Rad = Math.toRadians(lat2);
// double lon2Rad = Math.toRadians(lon2);
// double deltaLat = lat2Rad - lat1Rad;
// double deltaLon = lon2Rad - lon1Rad;
// double a = Math.sin(deltaLat / 2) * Math.sin(deltaLat / 2) +
// Math.cos(lat1Rad) * Math.cos(lat2Rad) *
// Math.sin(deltaLon / 2) * Math.sin(deltaLon / 2);
// double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
// return EARTH_RADIUS_KM * c; // Distance in kilometers
// }
// public static void allocateWarehouseToRequest(int requestId) {
Run
ReliefAidApp2
2024-11-27 13:22:18.231 java[59824:2390932] +[IMKInputSession subclass]: chose IMKInputSession_Modern
ReliefAidApp2 > src > main > java > org > example > ReliefAidApp2
1000:6 (6390 chars, 117 line breaks) LF UTF-8 4 spaces


pgAdmin 4
Object Explorer
public.requests/di...
public.roles/disast...
public.warehouses/disaster_management/postgres@PostgreSQL 17
public.warehouses/disaster_management/postgres@PostgreS...
No limit
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (8)
disaster_events
requests
resource_allocation
resources
roles
shipments
users
warehouses
Trigger Functions
Types
Views
Subscriptions
postgres
Login/Group Roles
Tablespaces
Query
Query History
Scratch Pad
SELECT * FROM public.warehouses
ORDER BY location_id ASC
Data Output
Messages
Notifications
Showing rows: 1 to 3
Page No: 1
of 1
location_id [PK] character varying (255)
location_name text
latitude numeric
longitude numeric
address text
city text
state text
country text
zipcode text
1 1 Warehouse Zone A 30.40220000 77.97300000 Bidholi Kandoli Dehradun Uttarakhand India 248007
2 2 Warehouse Zone B 30.48900000 77.78700000 Vikasnagar Dehradun Uttarakhand India 248198
3 3 Warehouse Zone C 30.30210000 78.00660000 Dehradun City Dehradun Uttarakhand India 248001
Total rows: 3
Query complete 00:00:00.198
LF
Ln 1, Col 1


Dodra Kwar, Himachal Pradesh to
Dodra Kwar, Himachal Pradesh to
Connaught Place, Dehradun,
google.com/maps/dir/30.3244,78.0339/30.4022,77.973/@30.369568,7...
Gmail
UPES
Mail - Rishabh Sha...
Post: Edit
Study Plan - LeetC...
My Account
canva.com
Console Home | C...
Placement paper (...
All Bookmarks
Best
42 min
33 min
3h 10m
Connaught Place, Dehradun, Uttarakhand
Kharakhet, Uttarakhand
Add destination
Leave now
Options
Send directions to Nothing A063
Copy link
via Deheradun Township- Bidholi- Majhaun Rd
33 min
13.1 km
Fastest route now due to traffic conditions
Details
via Ambala - Poanta Sahib - Herbertpur - Dehradun - Rishikesh National Hwy/Chakrata Rd
39 min
19.1 km
via Ambala - Poanta Sahib -
42 min
Saved
Recents
Kharakhet 42 min
Dehradun &...
Delhi
Gurugram
Get the app
Search along the route
Gas
EV charging
Hotels
Rudra Pur
Misraspatti
Mussoorie
Bhatta falls
Horawala
UPES
Shikhar Falls
Bidholi
Santala Devi Temple
Tilwari
Dev Bhoomi Uttarakhand University...
Dehradun Zoo
GUCCHUPANI ROBBERS CAVE
Hotel Royal Pal
Top rated
Pondha
42 min
19.1 km
33 min
13.1 km
GARHI CANTT
HATHIBARKALA
Central
Sudhowala
PREM NAGAR
East Hopetown
39 min
19.1 km
Connaught Place
DALANWALA
RAIPUR
MAJRA
AJABPUR KALAN
JOGIWALA
BANJARAWALA
Layers
Google
Map data ©2024
India
Terms
Privacy
Send Product Feedback
2 km


112
Click to call
Cancel
Call

6. Limitations and Future Enhancements

6.1 Limitations

Dependency on Java Swing (Outdated Technology)

Java Swing, used for the current GUI, is outdated and lacks modern design principles compared to contemporary frameworks like web-based platforms. This limits the application's visual appeal and scalability. To address this, transitioning to modern frameworks can improve user experience, adaptability, and support for modern design standards.

Network Connectivity Dependency

The application requires an active network connection for users to communicate with the PostgreSQL database, making it unsuitable for users in disaster-stricken or remote areas with limited connectivity.

Limited Scalability

The direct database connection design may lead to performance bottlenecks as user numbers grow, particularly during emergencies. Migrating to a cloud-based infrastructure with serverless architectures and connection pooling can enhance scalability and support high traffic scenarios effectively.

Insufficient Input Validation and Security

Basic validation is present, but input sanitization is lacking, leaving the system vulnerable to SQL injection and data corruption. Adding robust input validation and sanitization mechanisms, along with encryption for sensitive data and HTTPS for secure transmission, can significantly enhance the application’s security and reliability.

Hardcoded Database Details

The application stores database credentials including the username and password directly in the source code, posing a security risk and making maintenance

cumbersome. A more secure solution involves using environment variables or configuration files to store credentials securely, reducing the risk of exposure.

Lack of Offline Support

Users cannot interact with the application offline, which is critical in emergencies. Adding offline functionality with local storage and synchronization to the central database can ensure uninterrupted service during connectivity issues.

Platform Limitations

Due to the working limitations with regards to the tech stacks that could be implemented for this project, the desktop-based Java Swing application, while intended to be a mobile app, does not support mobile devices or web browsers, restricting accessibility.

Language Limitations

The application supports only one language, which limits its usability in non-English-speaking regions. Incorporating multilingual support through localization can broaden its reach and make it effective for diverse user bases.

Lack of Analytics and Reporting

The application lacks dashboards or reports for administrators to analyze requests and make data-driven decisions. Adding analytics and reporting capabilities can improve resource allocation and operational efficiency.

Limited Security Features

Sensitive user data like contact numbers is not encrypted, posing a risk in case of a breach. Adding data encryption and implementing robust security protocols will protect user information and meet compliance standards.

No Use of AI/ML for Optimization

The application doesn’t leverage AI/ML to predict resource demand or prioritize requests based on urgency. Incorporating AI/ML models can optimize resource distribution, enhance decision-making, and ensure timely aid delivery in emergencies.

6.2 Future Enhancements

Transition to Modern Frameworks

Replacing Java Swing with a modern UI framework like a web-based interface using React, Angular, or Vue.js would significantly improve user experience, ensure cross-platform compatibility, and simplify adherence to modern design standards.

Introduce Real-Time Updates

Using WebSockets or polling mechanisms to provide real-time updates on request statuses enhances transparency and keeps users informed about the progress of their requests, improving satisfaction and trust in the system.

Enhanced Security

Implementing encryption for sensitive data and secure transmission protocols like HTTPS protects user information and ensures compliance with data protection regulations, building trust and safeguarding against breaches.

Multilingual Support

Adding localization features to support multiple languages based on user preferences broadens the application's accessibility, making it usable for diverse audiences across different regions and linguistic groups.

Integration with Notification Services

Integrating SMS or email services to send automatic notifications for request submissions, updates, or approvals keeps users informed without requiring them to manually check the application, improving engagement and responsiveness.

Cloud Deployment and Scalability

Deploying the application on cloud platforms like AWS, Azure, or Google Cloud, and utilizing serverless architectures, ensures high availability, scalability, and resilience, especially during periods of peak usage.

Advanced Analytics and Reporting

Adding a dashboard for administrators to view trends, generate reports, and analyze resource demands supports better decision-making and enables efficient allocation of resources based on data-driven insights.

Use of AI/ML for Optimization

Employing AI/ML models to predict resource demand from historical data or analyze the urgency of requests optimizes resource distribution, ensuring prioritization and effective management during emergencies.

Progressive Web Application (PWA)

Developing a Progressive Web Application (PWA) with offline functionality and mobile support combines the advantages of a mobile app and a traditional web app, offering broader reach and seamless usability.

Modular and Microservices Architecture

Redesigning the application with a microservices architecture that separates the database, request handling, and UI into independent components simplifies maintenance, scaling, and feature upgrades for long-term sustainability.

7. Conclusion

The Disaster Relief Management System represents a comprehensive and efficient solution for addressing challenges in disaster response and resource management. By integrating a centralized database, role-based access control, and advanced algorithms

such as Dijkstra's for route optimization and 0/1 knapsack for resource allocation, the system ensures seamless coordination among stakeholders.

Key achievements of the project include:

- **Optimized Resource Utilization:** Automated allocation ensures critical resources reach the most affected areas promptly.
- **Streamlined Communication:** The centralized system facilitates real-time updates and efficient coordination between admins, drivers, and users.
- **Enhanced User Experience:** Role-specific interfaces provide tailored functionalities for each user group, simplifying operations and improving usability.
- **Efficient Routing:** The use of geospatial data and shortest path algorithms significantly reduces delivery times and costs.
- **Scalable and Modular Design:** The modular architecture ensures flexibility for future enhancements, including the addition of new functionalities or integration with external systems.

The system's holistic design enables effective disaster relief efforts by minimizing delays, optimizing resource allocation, and ensuring transparency in operations. While the project achieves its primary goals, future enhancements, such as real-time disaster monitoring and integration with external APIs for weather and traffic data, can further improve its capabilities.

This project serves as a step toward leveraging technology for humanitarian efforts, emphasizing the importance of efficiency, accessibility, and scalability in disaster management.

Appendix A. Acronyms and Abbreviations

PWA: Progressive Web Application
UI: User Interface
UX: User Experience
HTTP: HyperText Transfer Protocol
HTTPS: HyperText Transfer Protocol Secure
GIS: Geographical Information Systems
AI: Artificial Intelligence
ML: Machine Learning

Appendix B. Technology Stack

Programming Language: Java
Framework: Java Swing (for UI)
Database: PostgreSQL
Web Services: N/A
Development Tools: IntelliJ IDEA
Version Control: Git

Appendix C. Database Schema Overview

Users Table: Stores user information such as first name, last name, contact number, and location.

Requests Table: Manages resource requests, including quantity, resource type, and status.

Resources Table: Catalog of resources available for distribution.

Appendix D. Key Features

Resource Request Form

Users can submit requests for essential resources such as food packs, water bottles, blankets, and first aid kits. Each request captures details like resource ID, contact details, quantity, location, and urgency.

Driver Dashboard

A user interface designed for drivers to view and manage resource delivery requests. The dashboard displays each request pertaining to that particular driver's region and the driver's schedule in a user-friendly layout along with options to update the status as 'Done' , a button to call the requester and a button to open Google Maps with the location of the requester..

Sliding Menu

A hamburger-style menu provides easy navigation to key features such as:

Emergency Services: Direct access to initiate emergency calls.

Logout: A quick way to exit the application and return to the login page.

Requests: View and manage all submitted requests from the menu.

Algorithms for Sorting and Prioritization

Dijkstra’s Algorithm: Implemented to find the shortest path between the driver’s current location and multiple delivery points. This ensures efficient routing and minimizes travel time.

Priority Queue for Medicine Requests: Requests involving medicines are given the highest priority in the queue to ensure timely delivery of critical resources.

Calling Feature

Drivers can directly call requesters through the dashboard by clicking on the call button. This feature uses the system's default telephony service to establish calls.

Emergency Services Call Button

A dedicated button allows drivers to immediately call emergency services (in this case, 112) in case of urgent situations.

Resource Request Form:

Users can submit requests for resources such as food packs, water bottles, blankets, and first aid kits.

Driver Dashboard:

A user interface for drivers to manage and view delivery requests.

Sliding Menu:

A hamburger-style menu for navigation and ease of access.

PostgreSQL Documentation:

https://www.postgresql.org/docs/

Service Workers (PWA):

https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

References

1. Araz, O. M., & Selcuk, M. B. (2007). Network flow optimization for disaster response. European Journal of Operational Research, 180(2), 711-721. [DOI: 10.1016/j.ejor.2006.04.044]
2. Van Wassenhove, L. N., & Tomasini, R. M. (2003). Stochastic optimization for humanitarian logistics. INFORMS Journal on Computing, 15(3), 208-218. [DOI: 10.1287/ijoc.15.3.208.16082]
3. Bard, J. F., & Pirkul, H. (2014). Integer programming models for disaster response. Operations Research, 62(5), 1050-1064. [DOI: 10.1287/opre.2014.1291]
4. Saaty, T. L. (2008). Decision Making with the Analytic Hierarchy Process. International Journal of Services Sciences, 1(1), 83-98. [No DOI available]
5. Kuo, Y. F., & Lin, C. T. (2009). Fuzzy MCDM approach for evaluating alternative aid distribution strategies. European Journal of Operational Research, 196(2), 462-471. [DOI: 10.1016/j.ejor.2008.03.012]
6. Romero, C., & Rehman, S. (2009). Goal programming for humanitarian logistics. European Journal of Operational Research, 195(2), 670-677. [DOI: 10.1016/j.ejor.2008.03.013]
7. Chaudhuri, S., & Ghosh, A. (2020). Machine learning for demand prediction in humanitarian logistics. IEEE Transactions on Computational Intelligence and AI in Games, 12(1), 38-47. [DOI: 10.1109/TCIAIG.2019.2956231]
8. Zhang, Y., & Zheng, Z. (2018). Unsupervised learning for aid distribution pattern recognition. Data Mining and Knowledge Discovery, 32(4), 979-995. [DOI: 10.1007/s10618-018-0563-6]
9. Chen, Y., & Zheng, H. (2021). Reinforcement learning for adaptive aid distribution optimization. Journal of Artificial Intelligence Research, 70, 875-896. [DOI: 10.1613/jair.1.13442]
10. Kovačić, I., & Lisičić, B. (2012). Real-time tracking and monitoring of humanitarian aid. Computers & Industrial Engineering, 63(4), 1055-1066. [DOI: 10.1016/j.cie.2012.07.012]
11. Liu, Y., & Li, H. (2018). Big data analytics for humanitarian logistics. Information Systems Frontiers, 20(1), 105-120. [DOI: 10.1007/s10796-017-9813-6]

12. Yang, J., & Zhang, L. (2020). Predictive analytics for optimizing supply chain in humanitarian aid distribution. Journal of Business Analytics, 1(3), 199-210. [No DOI available]
13. Van Wassenhove, L. N., A. J. Martinez, and O. Stapleton. "An analysis of the relief supply chain in the first week after the Haiti earthquake." INSEAD Humanitarian Research Group (2010). https://www.academia.edu/50588431/INSEAD_HUMANITARIAN_RESEARCH_GROUP_An_Analysis_of_the_Relief_Supply_Chain_in_the_After_math_of_the_Haiti_Earthquake