## Mini Project Report

On

## “Disaster Management System”

Submitted To

**NMIMS**

**Mukesh Patel School of Technology Management & Engineering, Shirpur**

****

Submitted By

1. Gaurav Nitin Patil (70022400783) (B345)
2. Tejas Rajput (70022400645) (B340)
3. Om Borse (70022300554) (B341)

Under The Guidance of

**Prof. Sachin Bhandari**

**Department of Computer Engineering**

# YEAR 2024-25

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.N.** | **Title** | **Page No.** |
|  | Abstract | 3 |
| 2. | Introduction | 3 |
| 3. | Objectives | 3 |
| 4. | Project details | 3 |
| 5. | Snapshot | 4 |
| 6. | ER Diagram | 6 |
| 7. | Advantages | 6 |
| 8. | Disadvantages | 7 |
| 9. | Conclusions | 7 |
| 10. | References | 7 |

1. **Abstract**

The Disaster Management System (DMS) is a database-driven application designed to streamline disaster response operations. Built using Python (Tkinter for GUI) and Microsoft SQL Server, the system provides an efficient way to manage victims, resources, volunteers, shelters, and emergency calls. It supports CRUD (Create, Read, Update, Delete) operations, predefined analytical queries, and custom SQL execution for real-time decision-making. The system ensures centralized data management, improving coordination among disaster response teams and optimizing resource allocation during emergencies.

**2. INTRODUCTION:**

Natural disasters such as earthquakes, floods, and hurricanes require rapid and well-coordinated responses. Traditional disaster management often suffers from disorganized data, delayed communication, and inefficient resource tracking. This project addresses these challenges by developing a centralized database system that integrates all critical aspects of disaster management.

The system allows:

* Real-time tracking of victims, shelters, and medical needs
* Efficient resource allocation (food, water, medical supplies)
* Automated reporting for quick decision-making
* Historical data analysis for future preparedness

**3. Objectives**

The primary objectives of this project are:

1. To develop a user-friendly GUI for disaster management operations.
2. To establish a centralized SQL database storing disaster-related data.
3. To implement CRUD operations for managing victims, volunteers, shelters, and resources.
4. To provide predefined analytical queries for quick insights (e.g., injured victims, resource shortages).
5. To allow custom SQL queries for flexible data retrieval.
6. To ensure scalability for handling large-scale disasters.

**4. Project Details**

3.1 Technologies Used

* Frontend: Python (Tkinter)
* Backend: Microsoft SQL Server (pyODBC)
* Key Features:
  + 15+ database tables (Victims, Shelters, Resources, Volunteers, etc.)
  + 30+ predefined queries (aggregations, joins, filters)
  + Custom SQL execution for advanced users
  + Data modification (Add, Edit, Delete records)

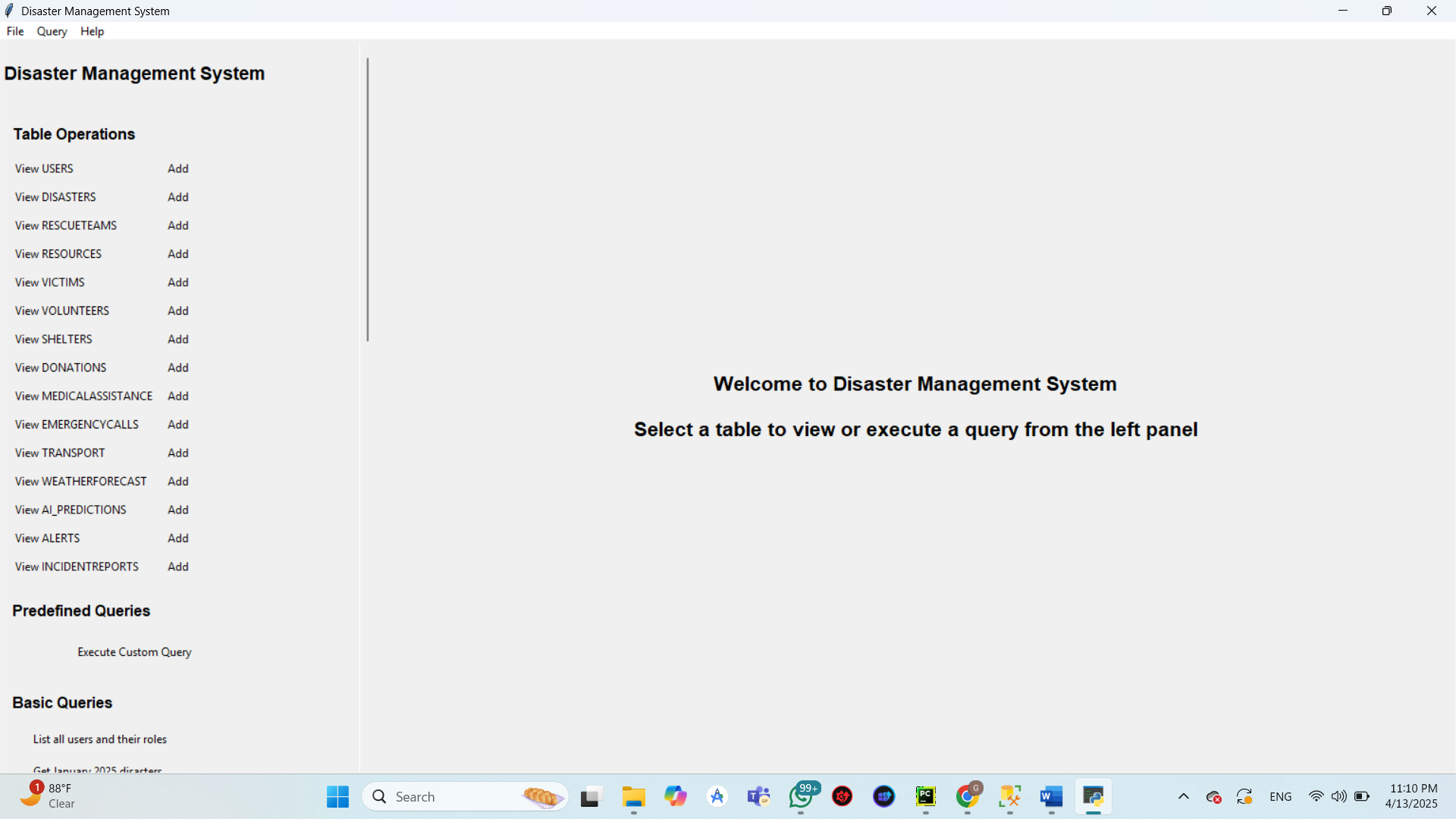
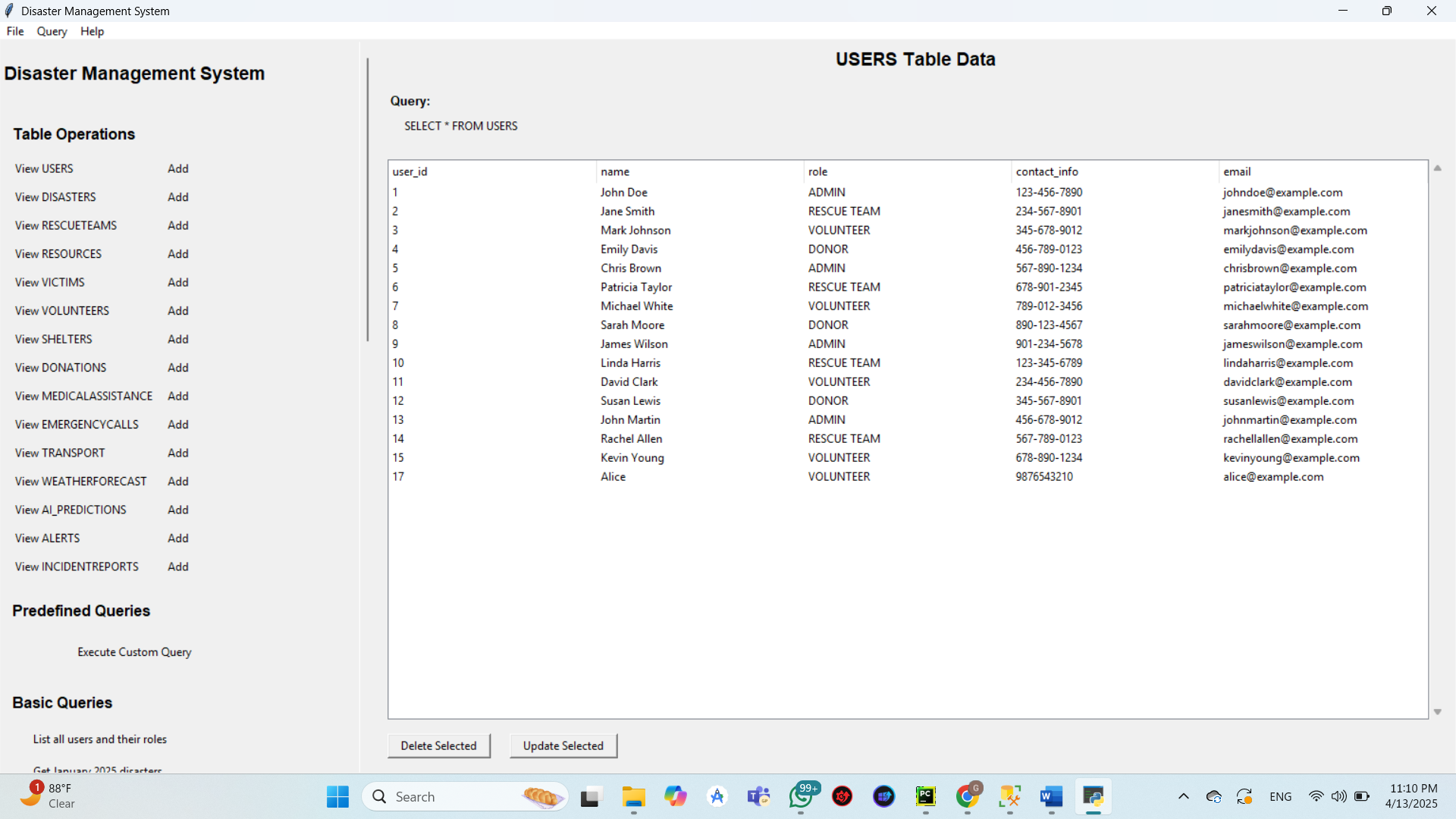
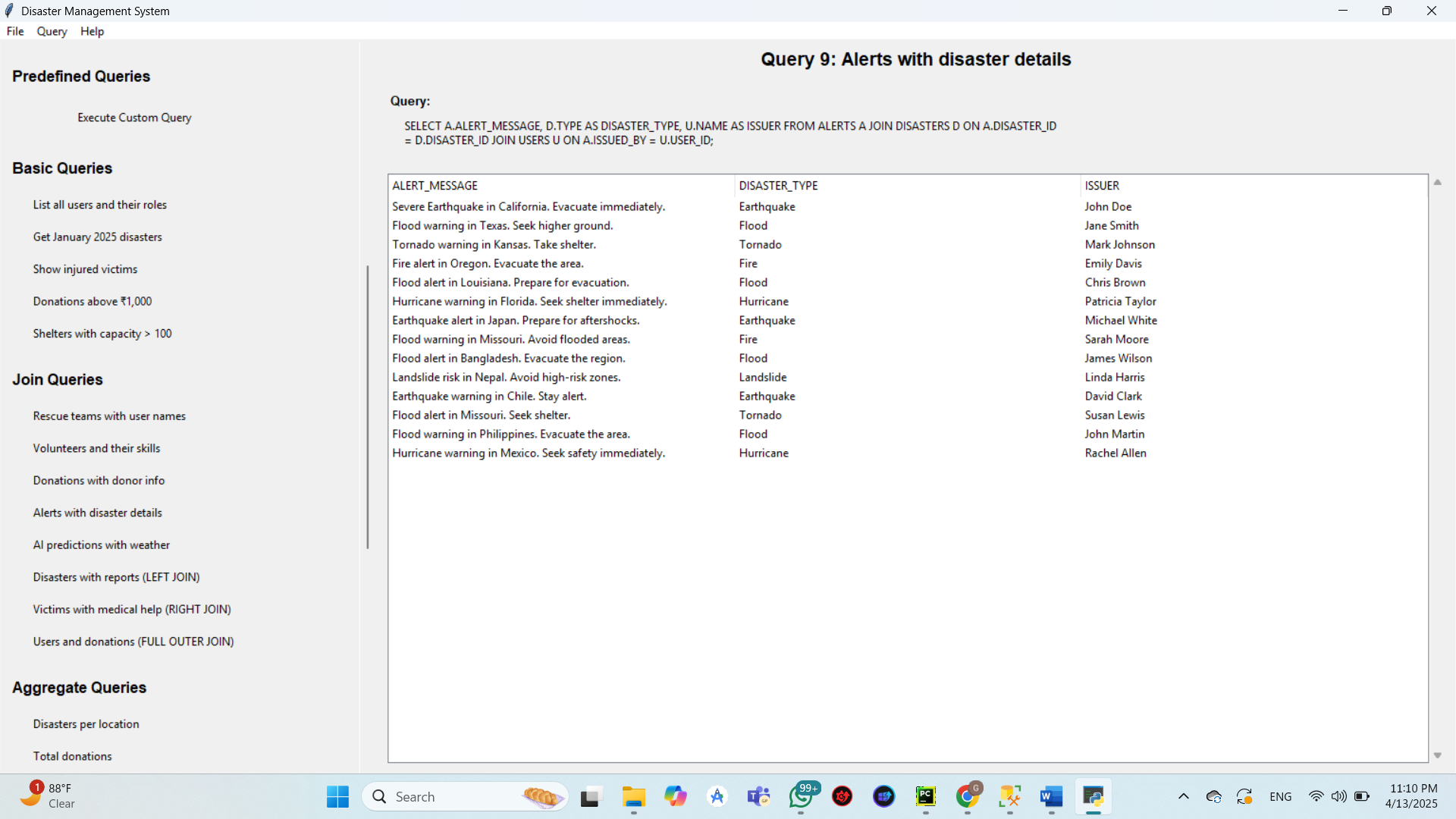
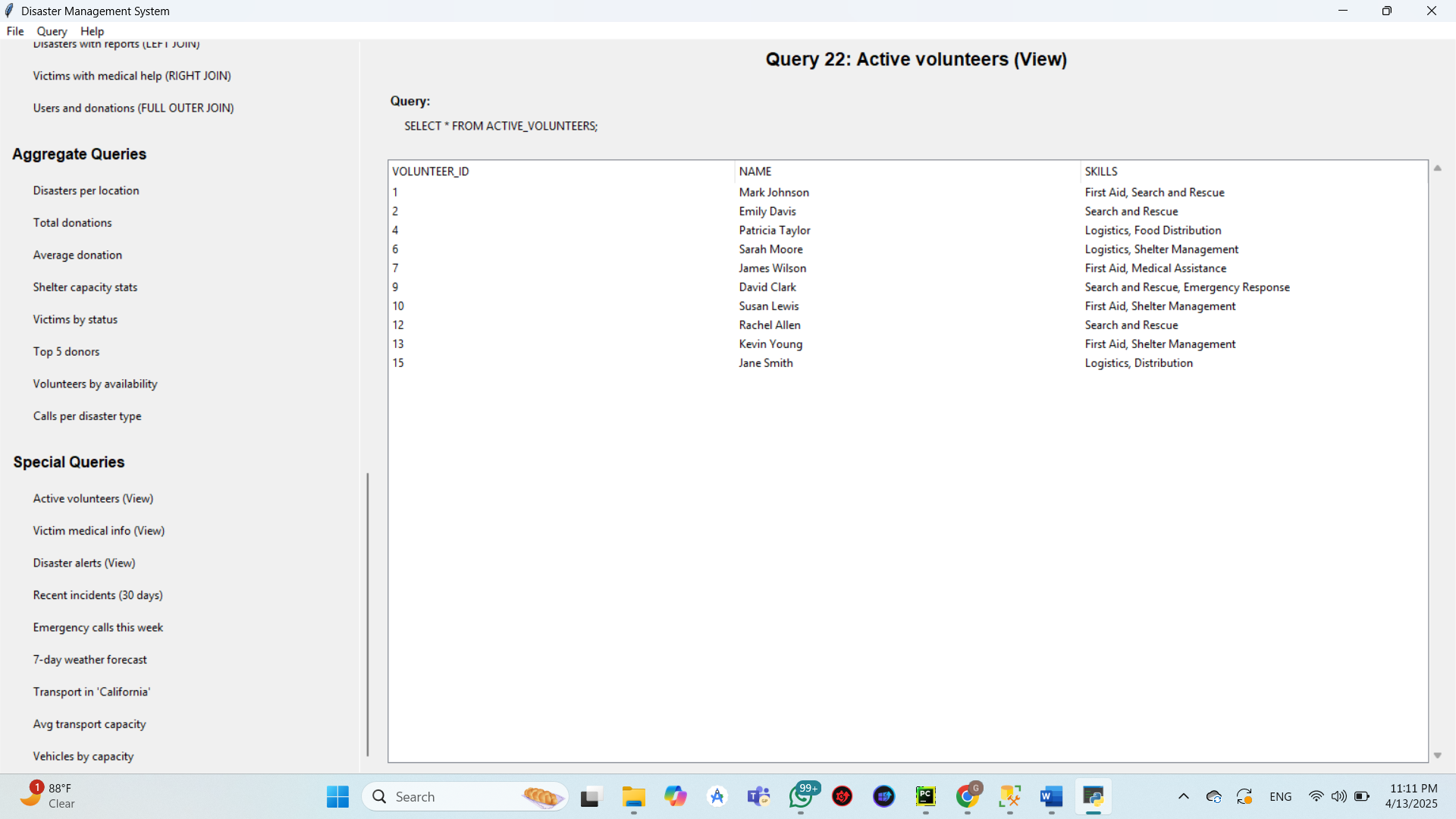
3.2 System Modules

1. User Management – Tracks responders, volunteers, and administrators.
2. Disaster Tracking – Records disaster types, locations, and severity.
3. Victim Management – Stores victim details (status, medical needs).
4. Resource Allocation – Manages food, water, and medical supplies.
5. Shelter Management – Tracks shelter locations and capacities.
6. Emergency Call Logs – Records distress calls for rapid response.
7. Analytics & Reports – Generates insights on disaster impact.

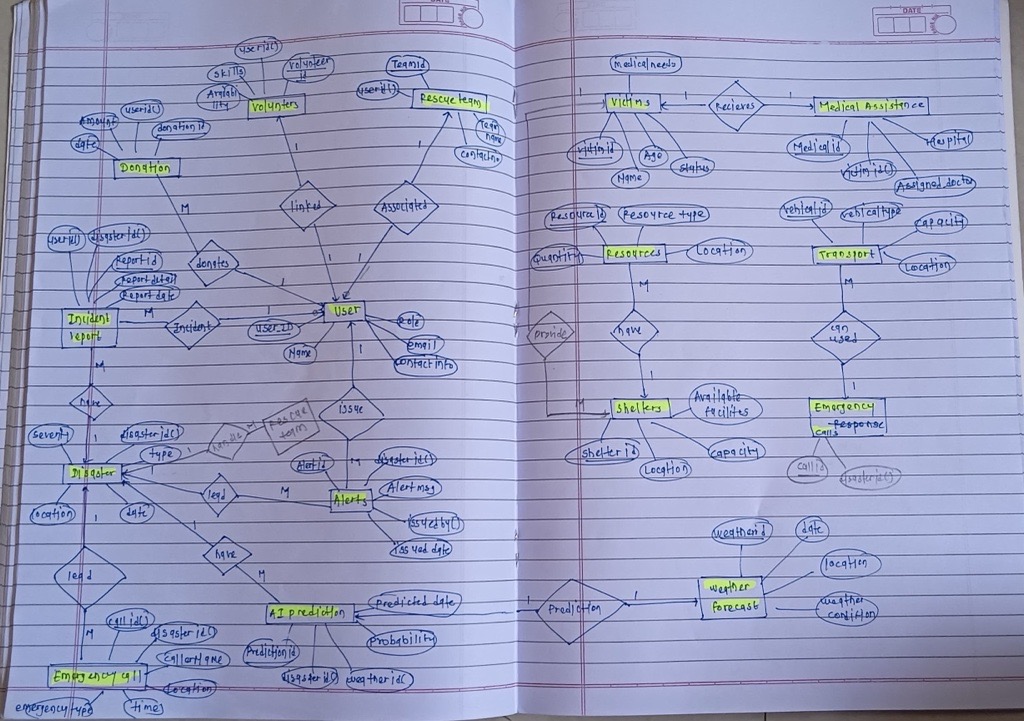
3.3 Database Schema

The database consists of interconnected tables, including:

* USERS (Responders, Admins)
* DISASTERS (Type, Location, Date)
* VICTIMS (Name, Status, Medical Needs)
* RESOURCES (Type, Quantity, Location)
* SHELTERS (Capacity, Facilities)
* EMERGENCYCALLS (Caller, Location, Emergency Type)
  1. **Snapshot**

* 1. **ER Diagram**



* 1. **Advantages**
* Centralized Data – Eliminates silos, ensuring all teams access the same information.
* Faster Response – Quick victim tracking and resource allocation.
* Data-Driven Decisions – Built-in analytics improve efficiency.
* Scalability – Handles small incidents to large-scale disasters.
* User-Friendly – Intuitive GUI for non-technical users.
* Custom Query Support – Advanced users can run complex SQL.
  1. **Disadvantages**
* Dependent on SQL Server – Requires a database server setup.
* No Real-Time Sync – Lacks cloud-based live updates.
* Limited GIS Integration – No mapping for disaster zones.
* No Mobile Access – Desktop-only application.
* Basic UI – Tkinter has limited modern design capabilities.
  1. **Conclusion**

The Disaster Management System (DMS) successfully addresses critical challenges in disaster response by providing a structured, database-driven approach. It improves coordination, reduces response times, and enhances resource management.

Future Enhancements:

* Mobile & Web Access for field responders
* Real-Time Alerts via SMS/Email
* GIS Integration for disaster mapping
* Machine Learning for predictive analytics
  1. **Reference**
* Python Software Foundation. (2023). Tkinter Documentation. <https://docs.python.org/3/library/tkinter.html>
* Microsoft. (2023). SQL Server Documentation. <https://docs.microsoft.com/en-us/sql/>
* PyODBC. (2023). Python ODBC Database Connectivity. <https://github.com/mkleehammer/pyodbc>