**USER MANUAL: DISASTER MANAGEMENT SYSTEM**

The Disaster Management System (DMS) is a desktop application that provides a structured platform to manage data related to natural and man-made disasters. Built with Python (Tkinter) and SQL Server, this system allows users to manage records, run queries, and visualize disaster-related information in real-time.

* **System Requirements**
* Software Requirements
* Python 3.x
* Microsoft SQL Server
* Python Libraries:
  + tkinter
  + pyodbc
* SQL Server Management Studio (optional for DB management)
* Hardware Requirements
* Processor: Intel Core i3 or above
* RAM: Minimum 4 GB
* Disk Space: Minimum 500 MB free
* Login/Startup
* **Run the Python script: python dms\_main.py**
* **The GUI window opens showing the main menu with options for:**
  + View Table Data
  + Add New Data
  + Execute Predefined Queries
  + Run Custom SQL Query
* **Navigation Guide**

1. View Table Data

* Select this option to display data from any of the available tables.
* Tables include:
  + DISASTERS, AFFECTEDAREAS, VICTIMS, RELIEF, VOLUNTEERS, SHELTERS, etc.
* Data is shown using a Treeview table format.

Steps:

1. Click on “View Table Data”
2. Select the table from the dropdown
3. Click “Show Data”

2. Add New Data

* Allows you to insert new records into tables.

Steps:

1. Click on “Add New Data”
2. Select the desired table
3. Enter values in the fields displayed
4. Click “Submit”

3. Execute Predefined Queries

* Run preset queries involving JOINs, GROUP BY, aggregates, views, etc.

Examples:

* List all victims in a given disaster
* Find shelters with capacity > 100
* Average relief amount by disaster type

Steps:

1. Click “Execute Predefined Queries”
2. Choose the query from the list
3. Click “Run Query”
4. View results in the Treeview

4. Run Custom SQL Query

* Advanced users can write and run custom SQL statements directly.

Steps:

1. Click “Run Custom SQL Query”
2. Enter SQL query (e.g., SELECT \* FROM VICTIMS)
3. Click “Execute”
4. Results will appear in the Treeview

⚠️ Note: Use valid SQL syntax and ensure your query does not harm the database (e.g., avoid DELETE without WHERE clause).

* **Table Descriptions**

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Purpose** | **Attribute** |
| **Users** | Stores admin, rescue teams, volunteers, and donors. | USER\_ID (PK), NAME, ROLE, CONTACT\_INFO, EMAIL |
| **Disasters** | Logs disaster events with type, severity, location, and date. | DISASTER\_ID (PK), TYPE, SEVERITY, LOCATION, DATE |
| **RescueTeams** | Assigns teams to disaster sites with contact details. | TEAM\_ID (PK), USER\_ID (FK), TEAM\_NAME, CONTACT\_NUMBER |
| **Resources** | Tracks availability of food, medical aid, shelters, and equipment. | RESOURCE\_ID (PK), RESOURCE\_TYPE, QUANTITY, LOCATION |
| **Victims** | Stores details of affected individuals, medical needs, and status. | VICTIM\_ID (PK), NAME, AGE, STATUS, MEDICAL\_NEEDS |
| **Volunteers** | Database of volunteers with skills, location, and availability. | VOLUNTEER\_ID (PK), USER\_ID (FK), SKILLS, AVAILABILITY |
| **Shelters** | Maintains shelter locations, capacity, and available facilities. | SHELTER\_ID (PK), LOCATION, CAPACITY, AVAILABLE\_FACILITIES |
| **Donations** | Records donations, donor details, and fund allocation. | DONATION\_ID (PK), USER\_ID (FK), AMOUNT, DATE |
| **MedicalAssistance** | Manages injured individuals, assigned doctors, and hospitals. | MEDICAL\_ID (PK), VICTIM\_ID (FK), ASSIGNED\_DOCTOR, HOSPITAL |
| **EmergencyCalls** | Stores distress calls with time, location, and emergency type. | CALL\_ID (PK), DISASTER\_ID (FK), CALLER\_NAME, LOCATION, EMERGENCY\_TYPE, TIME |
| **Transport** | Tracks available vehicles for rescue and resource transport. | VEHICLE\_ID (PK), VEHICLE\_TYPE, CAPACITY, LOCATION |
| **WeatherForecast** | Stores weather data to predict and prevent disasters. | WEATHER\_ID (PK), DATE, LOCATION, WEATHER\_CONDITION |
| **AI\_Predictions** | Analyzes historical data for disaster probability in an area. | PREDICTION\_ID (PK), DISASTER\_ID (FK), WEATHER\_ID (FK), PROBABILITY, PREDICTED\_DATE |
| **Alerts** | Logs alerts sent to public and officials during emergencies. | ALERT\_ID (PK), DISASTER\_ID (FK), ALERT\_MESSAGE, ISSUED\_BY (FK), ISSUED\_DATE |
| **IncidentReports** | Stores reports filed by officials and volunteers on the ground. | REPORT\_ID (PK), DISASTER\_ID (FK), USER\_ID (FK), REPORT\_DETAILS, REPORT\_DATE |

* **Error Handling**
* Invalid Input: Prompts the user to enter correct data types
* Database Errors: Displays SQL error messages in a popup
* Empty Queries: Custom query window shows warning if no input is given
* **Best Practices**
* Always back up your SQL Server database.
* Verify data types and primary key values while inserting new records.
* Keep disaster names and IDs consistent for joins and queries.
* Use custom query feature cautiously.
* **Tips**
* Use predefined queries for reports.
* You can add SQL views in SSMS and access them through custom queries.
* To update or delete data, SQL must be run via the custom query section.