Software Requirements Document (SRD)

1. Preface

Audience:
This document is intended for developers, project stakeholders, disaster management authorities, testers, and system administrators.

Version History:

Version	Date	Author	Description
1.0	2025-05-21	Jawad Ahmed, Abdul Razaq, Muhammad	Initial version of SRD for DDRD

---

2. Introduction

Purpose:
This document outlines the system and software requirements for the Dynamic Disaster Relief Database (DDRD). This platform aims to centralize real-time data for disaster relief including victim details, resources, volunteer management, and emergency response logistics.

System Overview:
The system will be developed using Django (Python) and PostgreSQL. It supports CRUD operations for disaster data and integrates with government and non-profit relief services. REST APIs will be used for external system communication.

Business Goals:

- Improve response efficiency during disasters.
- Centralize data from multiple sources.
- Provide analytics for decision-making.

---

3. Glossary

Term	Definition
CRUD	Create, Read, Update, Delete operations in a database.
API	Application Programming Interface for communication between software.
NGO	Non-Governmental Organization.
Django	A high-level Python web framework for rapid development.

Term	Definition
PostgreSQL	A powerful, open-source object-relational database system.

---

4. User Requirements Definition

Functional Requirements:

- Users can register and login securely.
- Admin can add disaster zones, victims, relief camps, and volunteers.
- NGOs can submit and track relief efforts.
- Volunteers can register and receive assignments.
- API access for other systems to fetch disaster-related data.

Non-Functional Requirements:

- Must handle concurrent users (up to 1000 at once).
- Uptime of 99.5% during active disasters.
- Role-based access control.
- Data encryption and secure login.
- Response time < 2 seconds per request.

Product & Process Standards:

- PEP8 for Python code.
- Django project standards.
- RESTful API design.
- JSON format for API responses.

---

5. System Architecture

- **Backend**: Django (Python)
- **Database**: PostgreSQL
- **Frontend**: HTML/CSS (with Django templates)
- **REST API**: Django REST Framework
- **Deployment**: Gunicorn + Nginx + PostgreSQL on Ubuntu Server

Modules:

- `Authentication`
- `DisasterZone Management`
- `Victim Records`
- `Volunteer Management`
- `NGO Coordination`
- `Admin Dashboard`

6. System Requirements Specification

Functional Requirements

ID	Description
FR1	Register/Login users securely
FR2	Admin can create disaster zones
FR3	NGOs can post relief events
FR4	Victims and resources can be added and updated
FR5	Volunteers can be assigned to zones
FR6	Reports and dashboards for analytics

Non-Functional Requirements

ID	Description
NFR1	API response time must be < 2 sec
NFR2	System must scale to support 1000 users concurrently
NFR3	All sensitive data must be encrypted
NFR4	Access should be restricted based on roles

---

7. System Models

- **Data Flow Model**:
  - User → Authentication → Role-based Dashboard → Data Interaction
- **Entity-Relationship Diagram**:
  - Entities: User, Victim, DisasterZone, NGO, Volunteer
  - Relationships: One-to-many between zones and victims, many-to-many for volunteers and zones.
- **Use Case Diagrams**:
  - Admin: Manage users, zones, resources
  - NGO: Submit relief efforts
  - Volunteer: View and accept tasks
  - Victim: View assistance status (if needed)

---

8. System Evolution

- Support for SMS notifications in future versions.

- Integration with GIS mapping tools.
- Real-time alert broadcasting using APIs.
- Mobile app extension using Django REST.

---

9. Appendices

Hardware Requirements:

- **Minimum**: 4 GB RAM, Dual-Core CPU, 20 GB disk space
- **Recommended**: 8 GB RAM, Quad-Core CPU, 50 GB SSD

Database Schema (Simplified):

- **Users** (username, password, role)
- **Victims** (name, age, status, location)
- **Zones** (zone_id, disaster_type, location)
- **NGO_Events** (event_id, ngo_id, details)
- **Volunteers** (user_id, availability)

---

10. Index

Would you like this in a PDF file?