**Software Requirements Specification (SRS) for NGO Volunteer Management System**

# Table of Contents

1. **Introduction**
   * Purpose
   * Scope
   * Definitions, Acronyms, and Abbreviations
   * References
2. **Overall Description**
   * Product Perspective
   * Product Functions
   * User Characteristics
   * Constraints
   * Assumptions and Dependencies
3. **System Requirements**
   * Functional Requirements
   * Non-functional Requirements
4. **System Design Details**
   * System Architecture
   * Database Schema
   * User Interface Design Requirements
5. **External Interfaces**
   * User Interfaces
   * APIs
   * Third-party Tools and Libraries
6. **Performance Requirements**
7. **Security Requirements**
8. **Appendices**

# 1. Introduction

**1.1 Purpose**

This document provides a detailed overview of the requirements for the NGO Volunteer Management System. The platform will enable NGOs to manage their volunteering programs effectively while offering volunteers a way to find and engage with meaningful projects.

**1.2 Scope**

The platform will be accessible via web browsers, providing functionality for user authentication, project management, volunteer applications, activity logging, and communication between NGOs and volunteers. The system will utilize the MERN stack for development and hosting through free-tier solutions.

**1.3 Definitions, Acronyms, and Abbreviations**

* **NGO**: Non-Governmental Organization
* **MERN**: MongoDB, Express.js, React, Node.js
* **JWT**: JSON Web Tokens (used for authentication)

**1.4 References**

* MongoDB Documentation
* React Documentation
* Node.js and Express.js APIs

# 2. Overall Description

**2.1 Product Perspective**

The platform will serve as an intermediary between NGOs and volunteers, eliminating manual processes like email chains and paper-based logs. It will provide real-time tracking, reporting, and communication.

**2.2 Product Functions**

1. Role-based authentication for NGOs and volunteers.
2. Project creation and management by NGOs.
3. Volunteer browsing and applying for projects.
4. Activity logging by volunteers.
5. Real-time communication between users.
6. Rating and feedback for completed projects.

**2.3 User Characteristics**

1. **NGO Representatives**: Non-technical users responsible for managing volunteer projects. Requires a simple and intuitive interface.
2. **Volunteers**: Diverse users ranging from students to working professionals, comfortable with basic web interactions.

**2.4 Constraints**

1. Hosted on free-tier platforms (Netlify, MongoDB Atlas, Render).
2. Limited storage and computational resources.
3. Mobile responsiveness to support varying devices.

**2.5 Assumptions and Dependencies**

1. Users will have stable internet connections.
2. Browsers will support modern JavaScript features.
3. Hosting services will maintain uptime.

# 3. System Requirements

**3.1 Functional Requirements**

**Authentication:**

* User registration with role specification (NGO or Volunteer).
* Secure login with JWT-based authentication.
* Password recovery via email.

**Project Management:**

* NGOs can create, edit, and delete projects.
* Projects include title, description, skills required, location, start/end dates.

**Volunteer Applications:**

* Volunteers can browse and apply for projects.
* NGOs can accept/reject applications.

**Activity Logging:**

* Volunteers log hours and provide descriptions of completed tasks.
* NGOs can approve/reject activity logs.

**Communication:**

* Real-time messaging between NGOs and volunteers.
* Notifications for application status changes.

**Reporting:**

* NGOs can view/download activity summaries.
* Volunteers can view their contribution history.

**3.2 Non-functional Requirements**

1. **Performance**: The system should support up to 1,000 concurrent users on free-tier hosting.
2. **Scalability**: Easily extendable to premium hosting plans.
3. **Security**: Use HTTPS for data transmission and encrypt passwords.
4. **Usability**: Minimal learning curve; intuitive navigation.
5. **Availability**: 99% uptime on free hosting plans.

# 4. System Design Details

**4.1 System Architecture**

1. **Frontend**: React with Tailwind CSS for responsive design.
2. **Backend**: Express.js, with REST APIs.
3. **Database**: MongoDB Atlas for document-oriented data storage.
4. **Hosting**:
   * **Frontend**: Netlify or Vercel.
   * **Backend**: Render (Node.js hosting).

**4.2 Database Schema**

**Users Collection:**

{

name: String,

email: String,

password: String,

role: String, // 'admin', 'volunteer'

location: String

}

**Projects Collection:**

{

title: String,

description: String,

skillsRequired: [String],

startDate: Date,

endDate: Date,

adminId: ObjectId, // Reference to Users collection

}

**Applications Collection:**

{

projectId: ObjectId,

volunteerId: ObjectId,

status: String, // 'applied', 'approved', 'rejected'

}

**Activity Logs Collection:**

{

projectId: ObjectId,

volunteerId: ObjectId,

hoursLogged: Number,

description: String

}

**4.3 User Interface Design Requirements**

1. Simple login and registration pages.
2. Dashboard for project management (NGOs).
3. Browsing interface for projects (Volunteers).

# 5. External Interfaces

**5.1 User Interfaces**

* Accessible via modern browsers (Chrome, Firefox, Edge).

**5.2 APIs**

* RESTful APIs for:
  + Authentication
  + CRUD operations on projects
  + Application management
  + Activity logging

**5.3 Third-party Tools and Libraries**

1. **React Router** for frontend routing.
2. **Socket.IO** for real-time chat.
3. **MongoDB Atlas** for database.
4. **Tailwind CSS** for styling.

**6. Performance Requirements**

1. Average response time < 500ms under normal load.
2. System should handle 100 simultaneous application submissions.

**7. Security Requirements**

1. Use bcrypt for password encryption.
2. Secure API endpoints using middleware.
3. Regular monitoring of database for vulnerabilities.

**8. Appendices**

**Appendix A: Acronyms and Glossary**

* NGO: Non-Governmental Organization
* MERN: MongoDB, Express.js, React, Node.js

**Appendix B: References**

* MongoDB Atlas documentation
* React official documentation
* Express.js middleware guide