**Client Requirement Document for**

**Bunyan**

**Version**: 1.0

**Date**: 5th of November 2024

**Prepared** **By**: Nour Ayman, Rawan Ahmed Mohamed, Mohamed Mahmoud Hussein Ahmed Raslan

1. **Project Overview** 
   1. **Purpose:**

To develop a comprehensive platform for managing donations, volunteering opportunities, events, and organizational services using scalable and maintainable software design patterns. Create an app that connects donors with various charitable causes (e.g., medical help supplies, poverty alleviation, education, disaster relief). It allows users to browse through various campaigns and make donations easily.

* 1. **Background:**

The client is a nonprofit organization aiming to streamline operations such as managing donations, assigning volunteering tasks, tracking performance, and engaging donors through various rewards and events.

* 1. **Project Goals:**
* Enhance donor and volunteer engagement.
* Ensure scalability, maintainability, and security through well implemented design patterns.
* Provide analytics and feedback mechanisms to measure program impact.

1. **Scope of Work** 
   1. **Inclusions:**

* User authentication and role-based access.
* Donation management with multiple methods and categories.
* Volunteering opportunities.
* Admin dashboards for CRUD operations and analytics. Where each admin has their own set of responsibilities.
* Event management.
* Reward mechanisms for users.
  1. **Exclusions:**
* Third-party integrations.
* Custom API development for external applications unless specified.
  1. **Assumptions:**
* Users will have internet access.
* The application will be accessed using a mobile device with either an android or iOS platform.
* Payment gateway integrations will use widely supported APIs.

1. **Functional Requirements** 
   1. **User Stories/Use Cases:**

* As a doner, I want to donate items or money using various methods.
* As a doner, I want to see fundraising goals and be able to contribute towards them.
* As a doner, I want to receive notifications about fundraising goals, and events.
* As a doner, I want to sometimes be able to set up recurring monetary donations.
* As a doner, I want to set up automated payments and chose the start and end date of those payments.
* As a doner, I want to get a notification every time one of the automated payments is made.
* As a volunteer, I want to be able to offer help.
* As a volunteer, I want to be able to sign up for volunteering opportunities.
* As a volunteer, I want to receive notifications about volunteering opportunities.
* As a beneficiary, I want to be able to request help.
* As a beneficiary, I want to receive donations.
* As a beneficiary, I want to receive notifications about events.
* As an admin, I want to oversee operations.
* As a payment admin, I want to pre-approve cash donations.
* As a payment admin, I want to view all active and inactive recurring donations.
* As a payment admin, I want to modify or cancel recurring donations.
* As an event admin, I want to organize events and set notifications to be sent out about them.
* As a donations amin, I want to make sure the donation process is completed.
* As a user, I want to provide feedback on the platform.
* As a user, I want to be able to make suggestions for improvements.
  1. **Core Functionality:**
* Authentication: Role based authentication for users, volunteers, and admins.
* Donation Categories: Options for monetary and itemized donations.
* Fundraising Goals: Visibility and tracking of fundraising targets.
* Volunteering: Profile creation, skill specification, and task assignments.
* Events: Notifications, RSVP, and check ins for users.
* Fail-safe mechanisms: If a recurring payment fails, retry attempts should be made up to three times before marking the donation as failed.
* Search and filter: robust search and filter capabilities to ease the use of the application.
  1. **System Requirements:** Realtime analytics dashboard for donations, volunteering hours, and event participation.
  2. **User Interface Requirements:**
* Intuitive UI with branding aligned to the organization’s identity.
* Accessibility standards compliance (WCAG).
  1. **Data Handling:**
* Secure storage of user profiles, data, donation history, and volunteer hours.
* Maintaining tax compliant receipts for purchases made by the organization.

1. **Nonfunctional Requirements** 
   1. **Performance:**

* Support up to 10,000 concurrent users.
* Response times under 2 seconds for critical actions.
  1. **Reliability:** Uptime of 99.9%.
  2. **Security:** Data encryption, secure payment processing, and GDPR compliance.
  3. **Maintainability:** Modular code with detailed documentation and use of design patterns (e.g., MVC, Singleton).
  4. **Usability:** User-friendly interfaces for both end users and admins.
  5. **Scheduling logic:** to handle recurring donations.
  6. **Scalability:**
* The system should support a large number of recurring donation schedules without performance degradation.
* The system should have a database that can handle a large number of users.

1. **Target Audience and User Roles** 
   1. **User Profiles:**

* Donors: Individuals or organizations contributing money or items.
* Volunteers: Individuals offering skills and availability.
* Beneficiaries: Refugees or other program participants.
  1. **Roles and Permissions:**
* Payment Admin: Manages payment methods and receipts.
* Donation Admin: Oversees donation tracking.
* Event Admin: Oversees events organizations.

1. **Technical Requirements** 
   1. **Platform Compatibility: M**obile platforms (iOS, Android).
   2. **Integration Needs:**

* Firebase authentication.
* Payment gateway integration.
  1. **Technology Stack Preferences:**
* Flutter for cross platform development.
* Firebase for backend services.

1. **Project Timeline** 
   1. **Milestones and Phases:**

* Requirements Gathering (Week 1,2).
* Design and Prototyping (Week 3-7).
* Development Phase 1: Core Features (Week 8).
* Development Phase 2: Analytics and Rewards (Week 9-12).
* Testing and QA (Week 13-14).
* Deployment (Week 15).
  1. **Deliverable Schedule:**
* Prototype: Week 8.
* Beta Version: Week 13.

1. **Budget Constraints** 
   1. **Cost Estimate:** Estimated budget: application hosting, and storage capacities.
   2. **Payment Terms:** Milestone based payment schedule.
2. **Acceptance Criteria** 
   1. **Success Metrics:**

* 90% of test cases pass during QA.
* User feedback scores of 4/5 or higher in UAT.
  1. **Testing Requirements:**
* Unit testing for individual components.
* Integration testing for external APIs.

1. **Appendices** 
   1. **Glossary:**

* **CRUD:** Create, Read, Update, Delete.
* **GDPR:** General Data Protection Regulation.
  1. **Reference Material:**

***Design patterns mapped to functionalities:***

* **Strategy:** Authentication, payment methods, user types.
* **Singleton:** Current app user, Firebase authentication.
* **Decorator:** Tax calculation on receipts, text styling.
* **Facade:** Admin dashboards, Analytics dashboard.
* **Observer:** Notifications, Event Updates.
* **Factory:** User Profile Creation, Event Creation
* **MVC:** Main App Architecture.
* **Command:** Request Handlers, Admin Tasks.
* **Adapter:** Data Transformation, Third-Party Integration
* **Template Method:** Payment Handling, Event Notifications.
* **Iterator:** List Traversals, User Profiles
* **Proxy:** Permission Handler, Resource Access
* **State:** Tracking Status