Software Requirements Specification (SRS) Document for Sadaqa App

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for the Sadaqa App, a platform designed to facilitate charitable contributions from the Muslim community in Tanzania. The app connects donors directly with mosques, orphanages, and care homes, ensuring that donations are delivered securely and transparently. This document outlines the functional and non-functional requirements, system architecture, and other key aspects of the system.

1.2 Scope

The Sadaqa App is a community-focused platform that enables users to donate to mosques, orphanages, and care homes. The system shall provide features for user registration, donation management, charity center profiles, and admin oversight. The app shall be available on both mobile and web platforms, with a focus on ease of use, security, and transparency.

1.3 Definitions, Acronyms, and Abbreviations

- Sadaqa: Voluntary charity in Islam.
- Zakat: Obligatory charity in Islam.
- Admin: System administrator responsible for managing the platform.
- Charity Center: Organizations such as mosques, orphanages, and care homes that receive donations.
- User: Individuals who donate through the platform.
- Guest Donor: A donor who makes a donation without creating an account.
- Registered Donor: A donor who creates an account to access additional features.
- Charity Center Manager: A user who manages a charity center's profile and campaigns.
- Charity Center Staff: A user who assists in managing a charity center's activities.
- System Admin: A user who manages the entire platform.
- Support Staff: A user who assists with platform support.

1.4 References

- Islamic principles on charity.
- GDPR (General Data Protection Regulation).
- Tanzanian financial regulations.

1.5 Overview

This document is organized into sections that describe the system's functional and non-functional requirements, system architecture, and other key aspects. The document provides a detailed description of the system's features, user interactions, and technical requirements.

2. Overall Description

2.1 Product Perspective

The Sadaqa App is a standalone system that interacts with users, charity centers, and payment gateways. The system shall integrate with external payment processors (e.g., Amana Bank, mobile money) to facilitate secure transactions.

2.2 Product Functions

The Sadaqa App shall provide the following key functions:

- Guest Donations: Allow donors to make one-time donations without creating an account.
- Optional Account Creation: Prompt donors to create an account after making a guest donation to access additional features.
- User Registration and Authentication: Allow users to register and log in to access personalized features.
- Donation Management: Support one-time and recurring donations.
- Charity Center Profile Management: Allow charity centers to register, create profiles, and post campaigns.
- Admin Oversight: Provide tools for admins to manage users, charity centers, and platform settings.
- Reporting and Analytics: Generate reports on donation trends and user activity.

2.3 User Characteristics

- Guest Donors: Individuals who make one-time donations without creating an account.
- Registered Donors: Individuals who create an account to access additional features.
- Charity Center Managers: Individuals who manage a charity center’s profile and campaigns.
- Charity Center Staff: Individuals who assist in managing a charity center’s activities.
- System Admins: Individuals who manage the entire platform.
- Support Staff: Individuals who assist with platform support.

2.4 Constraints

- The system shall comply with Islamic principles on charity.
- The system shall comply with Tanzanian financial regulations.

- The system shall ensure data security and privacy in accordance with GDPR.

2.5 Assumptions and Dependencies

- Users shall have access to a smartphone or computer with internet connectivity.
- Charity centers shall provide accurate and up-to-date information for their profiles.
- Payment gateways shall be available and functional at all times.

3. Specific Requirements

3.1 Functional Requirements

- 3.1.1 Guest Donations
- The system shall allow donors to make one-time donations without creating an account.
- The system shall collect the following information from guest donors:
    - Name (optional for anonymous donations).
    - Email (for receipts and communication).
    - Payment details.
- The system shall generate a digital receipt for each guest donation and send it to the donor's email address.
- The system shall allow guest donors to optionally create an account after completing their donation.

3.1.2 Optional Account Creation

- After a guest donation, the system shall prompt the donor to create an account to access additional features.
- The system shall pre-fill the registration form with the donor's email (if provided during the donation).
- The system shall link the guest donation to the newly created account if the donor chooses to register.
- The system shall provide incentives for account creation, such as:
    - Saving payment information for faster donations.
    - Setting up recurring donations.
    - Viewing donation history and receipts.

3.1.3 User Registration and Authentication

- The system shall allow users to register using their email address or phone number.
- The system shall require users to verify their email/phone number via OTP (One-Time Password).

- The system shall allow users to log in using their credentials or via social media (e.g., Google, Facebook).
- The system shall provide a password recovery option via email or SMS.

3.1.4 Donation Management

- The system shall allow users to browse a list of charity centers by category or location.
- The system shall allow users to search for specific charity centers using a search bar.
- The system shall allow users to view detailed profiles of charity centers, including their mission, needs, and ongoing campaigns.
- The system shall allow users to select a charity center and choose to donate a specific amount.
- The system shall provide users with a summary of their donation before confirming the payment.
- The system shall generate a digital receipt for each donation, which can be downloaded or emailed.

3.1.5 Recurring Donations

- The system shall allow registered donors to set up recurring donations (daily, weekly, monthly) to their preferred charity centers.
- The system shall allow users to manage or cancel recurring donations from their profile.

3.1.6 Donation History

- The system shall allow registered donors to view a history of all their past donations, including the date, amount, and recipient charity center.
- The system shall allow users to filter donation history by date, amount, or charity center.
- The system shall allow users to download receipts for past donations.

3.1.7 Charity Center Management

- The system shall allow charity center managers to register by providing necessary details (name, address, mission, etc.).
- The system shall require charity centers to be verified by the admin before they can receive donations.
- The system shall allow charity center managers to create and manage their profiles, including uploading photos, videos, and descriptions of their work.
- The system shall allow charity center managers to update their needs and campaigns (e.g., "We need blankets for winter").
- The system shall allow charity center managers to post specific needs or campaigns (e.g., "Help us build a new mosque").

- The system shall allow charity center managers to set fundraising goals and track progress in real-time.
- The system shall allow charity center managers to view all donations received, including donor details (if permitted by the donor).
- The system shall allow charity center managers to generate reports on donation trends and export data for accounting purposes.
- The system shall allow charity center managers to send thank-you messages or updates to donors who have supported them.

3.1.8 Admin Features

- The system shall allow system admins to view, edit, or delete user accounts.
- The system shall allow system admins to reset user passwords if requested.
- The system shall allow system admins to verify and approve charity centers before they are listed on the platform.
- The system shall allow system admins to suspend or remove charity centers that violate platform policies.
- The system shall allow system admins to generate reports on overall donation trends, including total donations, most supported charity centers, and donor demographics.
- The system shall allow system admins to export data for financial or operational analysis.
- The system shall allow system admins to review and approve posts, campaigns, and updates from charity centers.
- The system shall allow system admins to remove inappropriate content or suspend accounts if necessary.
- The system shall allow system admins to monitor platform activity, including user logins, donations, and charity center updates.
- The system shall allow system admins to set up alerts for suspicious activity or technical issues.
- The system shall allow admins to set up alerts for suspicious activity or technical issues.

3.2 Non-Functional Requirements

3.2.1 Performance

- The system shall load within 3 seconds on average, even with high user traffic.
- Payment transactions shall be processed within 5 seconds.

3.2.2 Security

- The system shall encrypt all user data and payment information using SSL/TLS.
- The system shall comply with GDPR and other data protection regulations.
- The system shall implement multi-factor authentication (MFA) for administrative access.
- The system shall support Role-based Access Control (RBAC) to restrict unauthorized system access.

3.2.3 Usability

- The system shall provide a user-friendly interface for administrators and security teams.
- The system shall offer customizable dashboards and reports.
- The system shall include comprehensive documentation, user guides, and contact for support.

3.2.4 Maintainability

- The system shall provide clear error messages and logging for troubleshooting.
- The system shall allow for easy updates and maintenance without significant downtime.

3.2.5 Interoperability

- The system shall operate seamlessly across different Linux distributions.
- The system shall support integration with various security and monitoring tools.

3.2.6 Reliability

- The system shall ensure data integrity and consistency across all components.
- The system shall recover automatically from failures within 5 minutes.

4. System Architecture

4.1 Architectural Overview

The Sadaqa App follows a 3-tier architecture, which separates the system into three main layers: Presentation Layer, Application Layer, and Data Layer. This architecture ensures modularity, scalability, and ease of maintenance. Each layer has distinct responsibilities and interacts with the other layers through well-defined interfaces.

4.2 Presentation Layer (Frontend)

The Presentation Layer is responsible for the user interface (UI) and user experience (UX). It handles all interactions between the user and the system. The frontend will be developed using Flutter for mobile applications and React.js for the web platform.

4.2.1 Components of the Presentation Layer

1. *User Interface (UI) Components:*
    - Dashboard: Displays key metrics such as total donations, active charity centers, and recent activities.
    - Charity Center Profiles: Shows detailed information about each charity center, including their mission, needs, and ongoing campaigns.
    - Donation Interface: Allows users to select a charity center, enter the donation amount, and choose a payment method.
    - Admin Panel: Provides admins with tools to manage users, charity centers, and generate reports.
2. *User Interaction Handling:*
    - The frontend will handle user inputs, such as form submissions, button clicks, and navigation between screens.
    - It will also display notifications, alerts, and error messages to the user.
3. *Integration with Backend:*
    - The frontend will communicate with the backend via RESTful APIs to fetch data (e.g., charity center details, donation history) and send user inputs (e.g., donation requests, profile updates).
4. *Responsive Design:*
    - The UI will be designed to work seamlessly across different devices (mobile, tablet, desktop) and screen sizes.

4.3 Application Layer (Backend)

The Application Layer is the core of the system, handling business logic, data processing, and communication between the frontend and the database. The backend will be developed using Node.js with Express.js for building RESTful APIs.

4.3.1 Components of the Application Layer

1. User Management:
    - Handles user registration, authentication, and profile management.

- Implements role-based access control (RBAC) to restrict access to certain features based on user roles (e.g., donor, charity center, admin).

2. Donation Management:
   - Processes donation requests, including one-time and recurring donations.
   - Integrates with payment gateways (e.g., Amana Bank, mobile money) to securely process transactions.
   - Generates digital receipts for donations and sends them to users via email or in-app notifications.
3. Charity Center Management:
   - Allows charity centers to register, create profiles, and post campaigns.
   - Verifies charity centers before they can receive donations.
   - Tracks donation progress for ongoing campaigns and updates charity center profiles accordingly.
4. Admin Features:
   - Provides tools for admins to manage users, charity centers, and platform settings.
   - Generates reports on donation trends, user activity, and charity center performance.
   - Monitors platform activity and sets up alerts for suspicious behavior or technical issues.
5. Notification System:
   - Sends push notifications and emails to users for donation confirmations, recurring donation reminders, and updates from charity centers.
   - Notifies admins about critical issues, such as failed transactions or security breaches.
6. API Endpoints:
   - The backend will expose RESTful APIs for the frontend to interact with. Key endpoints include:
     - User registration and login.
     - Donation processing.
     - Charity center profile management.
     - Report generation.

4.4 Data Layer (Database)

The Data Layer is responsible for storing and managing all system data. The Sadaqa App will use MySQL as the primary database management system (DBMS). The database will store user information, charity center details, donation records, and campaign data.

4.4.1 Components of the Data Layer

1. Users Table:
    - Stores information about registered users, including their name, email, phone number, and password hash.
    - Tracks user roles (e.g., donor, charity center, admin) and permissions.
2. CharityCenters Table:
    - Stores information about registered charity centers, including their name, address, mission, and verification status.
    - Tracks ongoing campaigns and donation progress.
3. Donations Table:
    - Records all donations made by users, including the donation amount, date, and payment method.
    - Links donations to specific users and charity centers.
4. Campaigns Table:
    - Stores details about charity center campaigns, including the campaign title, description, goal amount, and current amount raised.
    - Tracks the start and end dates of each campaign.
5. Reports Table:
    - Stores generated reports on donation trends, user activity, and charity center performance.
    - Allows admins to access historical data for analysis and auditing.
6. Security and Backup:
    - The database will implement encryption for sensitive data (e.g., user passwords, payment information).
    - Regular backups will be performed to ensure data integrity and availability in case of failures.

4.5 Interaction Between Layers

1. Frontend-Backend Interaction:
    - The frontend sends HTTP requests to the backend APIs to fetch data or perform actions (e.g., making a donation, updating a profile).

- The backend processes these requests, performs the necessary business logic, and returns the appropriate response (e.g., donation confirmation, charity center details).

2. Backend-Database Interaction:
   - The backend interacts with the database to retrieve or store data (e.g., user information, donation records).
   - Database queries are optimized for performance and security to prevent issues such as SQL injection.
3. Third-Party Integrations:
   - The backend integrates with external payment gateways (e.g., Amana Bank, mobile money) to process donations.
   - It also integrates with email and push notification services to send alerts and updates to users and admins.

4.6 Deployment Architecture

The Sadaqa App will be deployed using a cloud-based infrastructure to ensure scalability, reliability, and high availability. Key components of the deployment architecture include:

1. Web Server:
   - Hosts the frontend application (Flutter for mobile, React.js for web).
   - Serves static assets (e.g., HTML, CSS, JavaScript) to users.
2. Application Server:
   - Hosts the backend application (Node.js with Express.js).
   - Handles API requests from the frontend and communicates with the database.
3. Database Server:
   - Hosts the MySQL database.
   - Stores all system data and ensures data integrity and security.
4. Load Balancer:
   - Distributes incoming traffic across multiple application servers to ensure high availability and performance.
5. Cloud Storage:
   - Stores media files (e.g., images, videos) uploaded by charity centers.
   - Ensures fast and reliable access to these files.
6. Monitoring and Logging:
   - Monitors system performance and logs errors or suspicious activity.
   - Provides real-time alerts to admins in case of issues.

4.7 Security Architecture

1. Data Encryption:
    - All sensitive data (e.g., user passwords, payment information) will be encrypted using SSL/TLS during transmission.
    - Data at rest (e.g., in the database) will be encrypted using AES-256 encryption.
2. Authentication and Authorization:
    - Users will be authenticated using email/phone number and password.
    - Admins will use multi-factor authentication (MFA) for added security.
    - Role-based access control (RBAC) will restrict access to certain features based on user roles.
3. Regular Security Audits:
    - The system will undergo regular security audits to identify and fix vulnerabilities.
    - Penetration testing will be performed to ensure the system is secure against attacks.

4.8 Scalability and Performance

1. Horizontal Scaling:
    - The system will be designed to scale horizontally by adding more application servers as user traffic increases.
    - A load balancer will distribute traffic evenly across servers.
2. Caching:
    - Frequently accessed data (e.g., charity center profiles, donation history) will be cached using Redis to reduce database load and improve performance.
3. Database Optimization:
    - Database queries will be optimized for performance using indexing and query optimization techniques.
    - Database replication will be used to ensure high availability and fault tolerance.

12 WEEKS TIMELINE

Week 1: Project Setup and Planning

Day 1:

- **Task**: Kickoff meeting and project overview.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Discuss the SRS document, clarify requirements, and assign roles.

Day 2:

- **Task**: Set up version control (Git/GitHub) and project management tools (Jira).
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Create repositories and organize tasks for tracking progress.

Day 3:

- **Task**: Set up development environments (React.js, Node.js, MySQL).
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Install necessary software and tools for all team members.

Day 4:

- **Task**: Finalize the database schema and API design.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Design the database tables and RESTful API endpoints based on the SRS.

Day 5:

- **Task**: Purchase and configure the domain name.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Buy a domain name and set up DNS settings.

Day 6:

- **Task**: Create a detailed task breakdown and assign responsibilities.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Break down the project into smaller tasks and assign them to team members.

Day 7:

- **Task**: Set up Google Cloud Platform (GCP) for hosting.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Create a GCP account and set up the initial cloud infrastructure.

Week 2: Frontend Development (User Registration and Login)

Day 8:

- **Task**: Create the user registration page UI.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the registration form using React.js.

Day 9:

- **Task**: Create the user login page UI.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the login form using React.js.

Day 10:

- **Task**: Implement form validation for registration and login pages.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Add validation for email, password, and other fields.

Day 11:

- **Task**: Connect the registration page to the backend API.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for user registration and connect it to the frontend.

Day 12:

- **Task**: Connect the login page to the backend API.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for user login and connect it to the frontend.

Day 13:

- **Task**: Implement error handling for registration and login.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Display error messages for invalid inputs or failed API calls.

Day 14:

- **Task**: Test the registration and login functionality.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure the registration and login features work as expected.

Week 3: Frontend Development (Donation Interface and Charity Center Profiles)

Day 15:

- **Task**: Create the donation interface UI.

- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the donation form using React.js.

Day 16:

- **Task**: Create the charity center profile page UI.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the profile page for charity centers.

Day 17:

- **Task**: Implement the donation summary page.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Create a page that shows a summary of the donation before confirmation.

Day 18:

- **Task**: Connect the donation interface to the backend API.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for processing donations and connect it to the frontend.

Day 19:

- **Task**: Implement the charity center profile backend logic.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for fetching charity center details.

Day 20:

- **Task**: Test the donation interface and charity center profiles.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure the donation and profile features work as expected.

Day 21:

- **Task**: Fix bugs and optimize the donation interface.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Week 4: Frontend Development (Admin Dashboard and Charity Center Management)

Day 22:

- **Task**: Create the admin dashboard UI.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the admin dashboard using React.js.

Day 23:

- **Task**: Create the charity center management interface UI.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Design and implement the interface for charity center managers to update profiles and campaigns.

Day 24:

- **Task**: Implement the admin dashboard backend logic.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for fetching admin-related data.

Day 25:

- **Task**: Implement the charity center management backend logic.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend endpoint for updating charity center profiles and campaigns.

Day 26:

- **Task**: Test the admin dashboard and charity center management features.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure the admin and management features work as expected.

Day 27:

- **Task**: Fix bugs and optimize the admin dashboard.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Day 28:

- **Task**: Review and finalize the frontend development.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Ensure all frontend features are complete and functional.

Week 5: Backend Development (User Authentication and Registration)

Day 29:

- **Task**: Implement user authentication using JWT (JSON Web Tokens).
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up JWT for secure user authentication.

Day 30:

- **Task**: Implement password hashing for user registration.

- **Assigned To**: Developer 2 (D2).
- **Explanation**: Use bcrypt to hash passwords before storing them in the database.

Day 31:

- **Task**: Implement email verification for user registration.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Send a verification email to users after registration.

Day 32:

- **Task**: Test user authentication and registration.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure authentication and registration work as expected.

Day 33:

- **Task**: Fix bugs in user authentication and registration.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Day 34:

- **Task**: Implement password recovery functionality.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Allow users to reset their passwords via email.

Day 35:

- **Task**: Test password recovery functionality.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure password recovery works as expected.

Week 6: Backend Development (Donation Processing and Recurring Donations)

Day 36:

- **Task**: Implement one-time donation processing.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend logic for processing one-time donations.

Day 37:

- **Task**: Implement recurring donation functionality.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Set up the backend logic for handling recurring donations (daily, weekly, monthly).

Day 38:

- **Task**: Test one-time and recurring donation functionality.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure donations are processed correctly.

Day 39:

- **Task**: Fix bugs in donation processing.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Day 40:

- **Task**: Implement donation history for users.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Allow users to view their past donations.

Day 41:

- **Task**: Test donation history functionality.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure donation history is displayed correctly.

Day 42:

- **Task**: Fix bugs in donation history functionality.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Week 7: Backend Development (Charity Center Management and Reporting)

Day 43:

- **Task**: Implement charity center registration and verification.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Allow charity centers to register and be verified by admins.

Day 44:

- **Task**: Implement charity center profile updates.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Allow charity centers to update their profiles and post campaigns.

Day 45:

- **Task**: Test charity center registration and profile updates.
- Assigned To: Tester (T).

- **Explanation**: Perform manual testing to ensure charity center features work as expected.

Day 46:

- **Task**: Fix bugs in charity center management.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Day 47:

- **Task**: Implement donation reporting for charity centers.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Allow charity centers to view donation trends and generate reports.

Day 48:

- **Task**: Test donation reporting functionality.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure reports are generated correctly.

Day 49:

- **Task**: Fix bugs in donation reporting.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Week 8-9: Payment Gateway Integration (M-Pesa)

Day 50:

- **Task**: Set up a business account with Vodacom Tanzania for M-Pesa.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Register for the M-Pesa API and obtain necessary credentials.

Day 51:

- **Task**: Integrate the M-Pesa API for mobile money payments.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Develop backend logic for processing payments via M-Pesa.

Day 52:

- **Task**: Test M-Pesa payment integration.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure payments are processed correctly.

Day 53:

- **Task**: Fix bugs in M-Pesa payment integration.
- **Assigned To**: Developer 2 (D2).

- **Explanation**: Address any issues found during testing.

Day 54:

- **Task**: Implement payment confirmation and receipts.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Send payment confirmations and receipts to users via email or SMS.

Day 55:

- **Task**: Test payment confirmation and receipts.
- Assigned To: Tester (T).
- **Explanation**: Perform manual testing to ensure confirmations and receipts are sent correctly.

Day 56:

- **Task**: Fix bugs in payment confirmation and receipts.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Week 10-11: Testing and Debugging

Day 57:

- **Task**: Perform end-to-end testing of all features.
- Assigned To: Tester (T).
- **Explanation**: Test the entire app to ensure all features work together seamlessly.

Day 58:

- **Task**: Fix bugs found during end-to-end testing.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Address any issues found during testing.

Day 59:

- **Task**: Perform performance testing.
- Assigned To: Tester (T).
- **Explanation**: Test the app's performance under high traffic and optimize as needed.

Day 60:

- **Task**: Fix performance issues.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Optimize the app's performance based on test results.

Day 61:

- **Task**: Perform security testing.
- Assigned To: Tester (T).

- **Explanation**: Test the app for vulnerabilities and ensure data security.

Day 62:

- **Task**: Fix security issues.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Address any vulnerabilities found during testing.

Day 63:

- **Task**: Finalize testing and prepare for deployment.
- Assigned To: Tester (T).
- **Explanation**: Ensure all tests are complete and the app is ready for deployment.

Week 12: Deployment and Post-Launch

Day 64:

- **Task**: Deploy the web app on Google Cloud Platform (GCP).
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Host the app on GCP and configure the domain name.

Day 65:

- **Task**: Perform final testing on the live environment.
- Assigned To: Tester (T).
- **Explanation**: Ensure everything works as expected after deployment.

Day 66:

- **Task**: Monitor the app and address any post-launch issues.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Fix any issues reported by users and optimize the app further.

Day 67:

- **Task**: Set up monitoring and logging tools.
- **Assigned To**: Developer 2 (D2).
- **Explanation**: Monitor the app's performance and log errors for future improvements.

Day 68:

- **Task**: Prepare user documentation and guides.
- **Assigned To**: Developer 1 (D1).
- **Explanation**: Create user-friendly guides for donors, charity centers, and admins.

Day 69:

- **Task**: Conduct a post-launch review meeting.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Review the project's success and identify areas for improvement.

Day 70:

- **Task**: Plan for future updates and features.
- **Assigned To**: Developer 1 (D1) and Developer 2 (D2).
- **Explanation**: Discuss and prioritize features for future updates (e.g., mobile app using Flutter).