SOFTWARE REQUIREMENT SPECIFICATION (SRS)

For

CharityNexus

V 1.0

Table of Contents

1. Introduction

1.1 Purpose of the Document

The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive overview of the requirements and specifications for the development of CharityNexus, a decentralized charity donation platform. This document serves as a guideline for the design, implementation, and testing phases of the project.

1.2 Scope of the Project

CharityNexus aims to create a secure and transparent platform for charitable donations, leveraging blockchain technology and smart contracts to facilitate seamless transactions between donors and verified charitable organizations. The platform will enable users to browse, select, and contribute to various charitable causes while ensuring accountability and transparency in donation tracking.

1.3 Definitions, Acronyms, and Abbreviations

- **CharityNexus**: The decentralized charity donation platform being developed.
- **SRS**: Software Requirements Specification.
- **DApp**: Decentralized Application.
- **UI**: User Interface.
- **API**: Application Programming Interface.
- **Blockchain**: A distributed and immutable ledger technology.
- **Smart Contracts**: Self-executing contracts with predefined rules on the blockchain.

2. Overall Description

2.1 Product Perspective

CharityNexus is a standalone decentralized application (DApp) designed to operate on blockchain technology. It interacts with blockchain networks for transaction processing, smart contract execution, and data storage. The platform integrates with external APIs for real-time data feeds and verification services, enhancing its functionality and reliability.

2.2 User Classes and Characteristics

2.2.1 Donors

- **Characteristics**: Individuals or organizations interested in making charitable donations.
- **Actions**: Browse charities, make donations, track donation history.
- **Requirements**: User authentication, donation tracking, payment integration.

2.2.2 Charities

- **Characteristics**: Verified charitable organizations or initiatives.
- **Actions**: Create charity listings, receive donations, provide updates.
- **Requirements**: Charity verification, project tracking, communication tools.

2.2.3 Administrators

- **Characteristics**: Platform administrators responsible for managing users and content.
- **Actions**: Verify charities, manage user accounts, monitor platform activity.
- **Requirements**: Admin dashboard, user management tools, content moderation.

2.3 Operating Environment

The CharityNexus platform operates in a distributed environment, leveraging blockchain networks for transaction validation and data integrity. It is accessible via web browsers and mobile devices, providing a seamless user experience across multiple platforms.

2.4 Design and Implementation Constraints

2.4.1 Blockchain Integration

- **Constraint**: Dependence on blockchain network for transaction processing and data storage.
- **Mitigation**: Utilize scalable and reliable blockchain solutions with low transaction costs.

2.4.2 Regulatory Compliance

- **Constraint**: Compliance with legal and regulatory requirements for charitable donations.
- **Mitigation**: Implement KYC (Know Your Customer) processes, adhere to data protection regulations.

2.5 Assumptions and Dependencies

- **Assumption**: Users have basic knowledge of blockchain technology and cryptocurrency transactions.
- **Dependency**: Integration with external APIs for real-time data updates and verification services.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Authentication

- **Description**: Users must register and log in to access platform features.
- **Inputs**: Username, email, password.
- **Outputs**: User session token.
- **Actions**: Sign up, login, logout.

3.1.2 Charity Listing

- **Description**: Charities can create listings detailing their missions, projects, and funding needs.
- **Inputs**: Charity details, project information, funding goals.
- **Outputs**: Charity listing page.
- **Actions**: Create listing, edit listing, deactivate listing.

3.1.3 Donation Processing

- **Description**: Donors can make donations to specific charities using cryptocurrencies or fiat currencies.
- **Inputs**: Donation amount, currency type, charity selection.
- **Outputs**: Donation confirmation.
- **Actions**: Make donation, view donation history.

3.1.4 Donation Tracking

- **Description**: Users can track their donations and view real-time updates on how funds are utilized by charities.
- **Inputs**: Donation ID, charity details.
- **Outputs**: Donation status, impact reports.
- **Actions**: Track donation, view impact reports.

3.1.5 Smart Contract Integration

- **Description**: Smart contracts handle donation transactions, ensuring transparency and security.

- **Inputs**: Contract parameters, transaction data.
- **Outputs**: Transaction confirmation, contract execution logs.
- **Actions**: Execute smart contract functions.

3.2 Non-functional Requirements

3.2.1 Performance

- **Requirement**: The platform should handle concurrent user interactions and transaction processing efficiently.
- **Measurement**: Response time, throughput.

3.2.2 Security

- **Requirement**: Data encryption, secure authentication, and authorization mechanisms.
- **Measurement**: Compliance with security standards (e.g., SSL/TLS protocols).

3.2.3 Usability

- **Requirement**: Intuitive user interfaces, clear navigation, and accessibility features.
- **Measurement**: User feedback, usability testing results.

3.2.4 Scalability

- **Requirement**: Ability to scale the platform to accommodate growing user base and transaction volumes.
- **Measurement**: Scalability testing, load balancing strategies.

3.2.5 Availability

- **Requirement**: High availability and uptime for continuous platform access.
- **Measurement**: Monitoring tools, uptime metrics.

3.3 External Interface Requirements

3.3.1 User Interface

- **Requirement**: Responsive web design, mobile compatibility, intuitive UI elements.
- **Interfaces**: Web browsers, mobile devices.

3.3.2 Payment Integration

- **Requirement**: Integration with payment gateways for cryptocurrency and fiat currency transactions.
- **Interfaces**: Payment APIs (e.g., PayPal, Stripe), blockchain wallets.

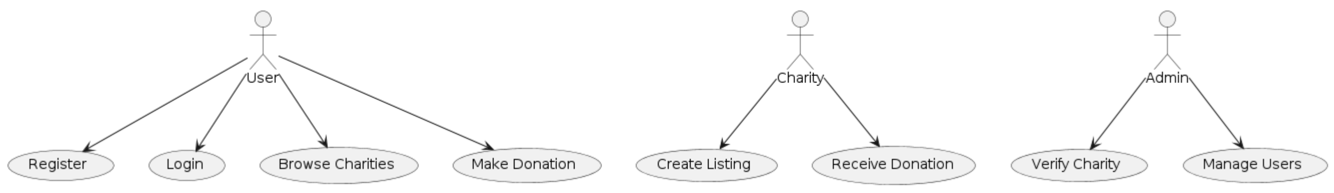
3.3.3 Blockchain Integration

- **Requirement**: Integration with blockchain networks for smart contract execution and data storage.
- **Interfaces**: Blockchain APIs (e.g., Ethereum, Binance Smart Chain).

4. System Models

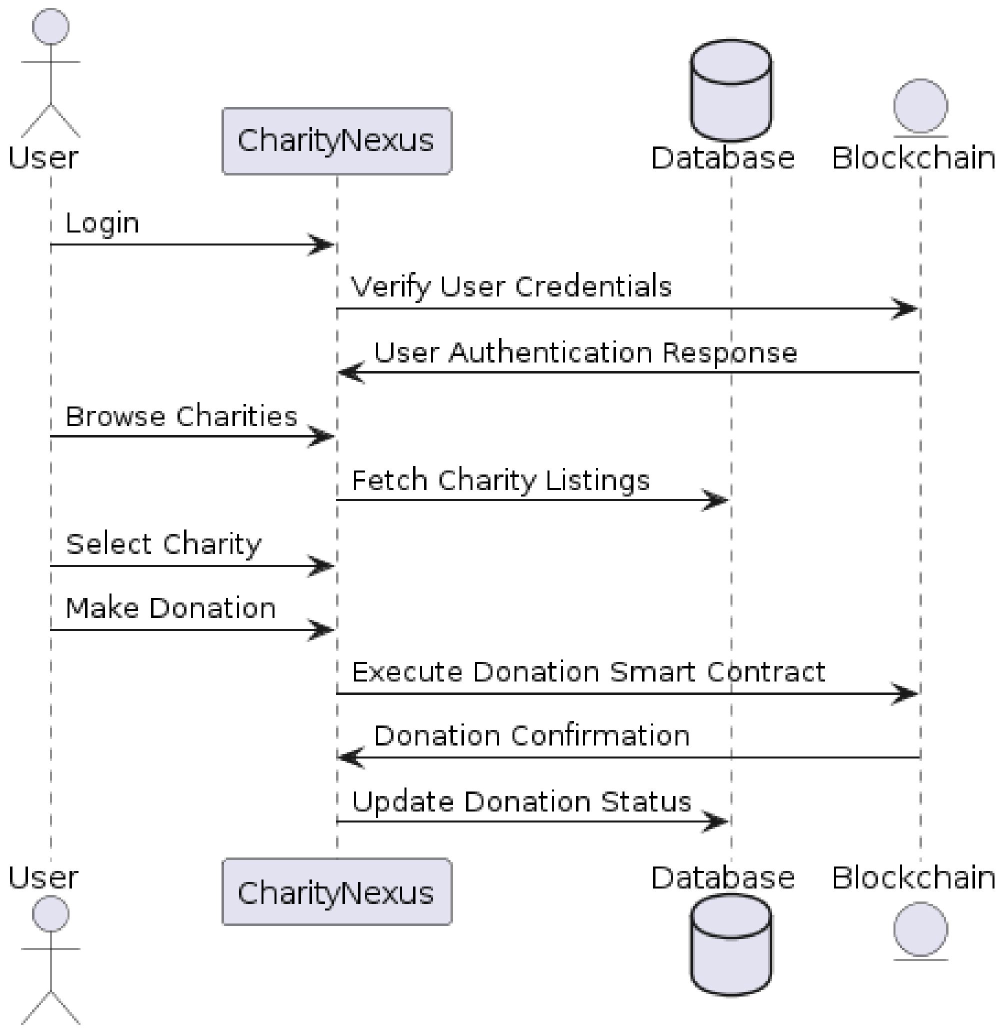
4.1 Use Case Diagram

The use case diagram illustrates the interactions between actors and the CharityNexus system. Here is a simplified representation:



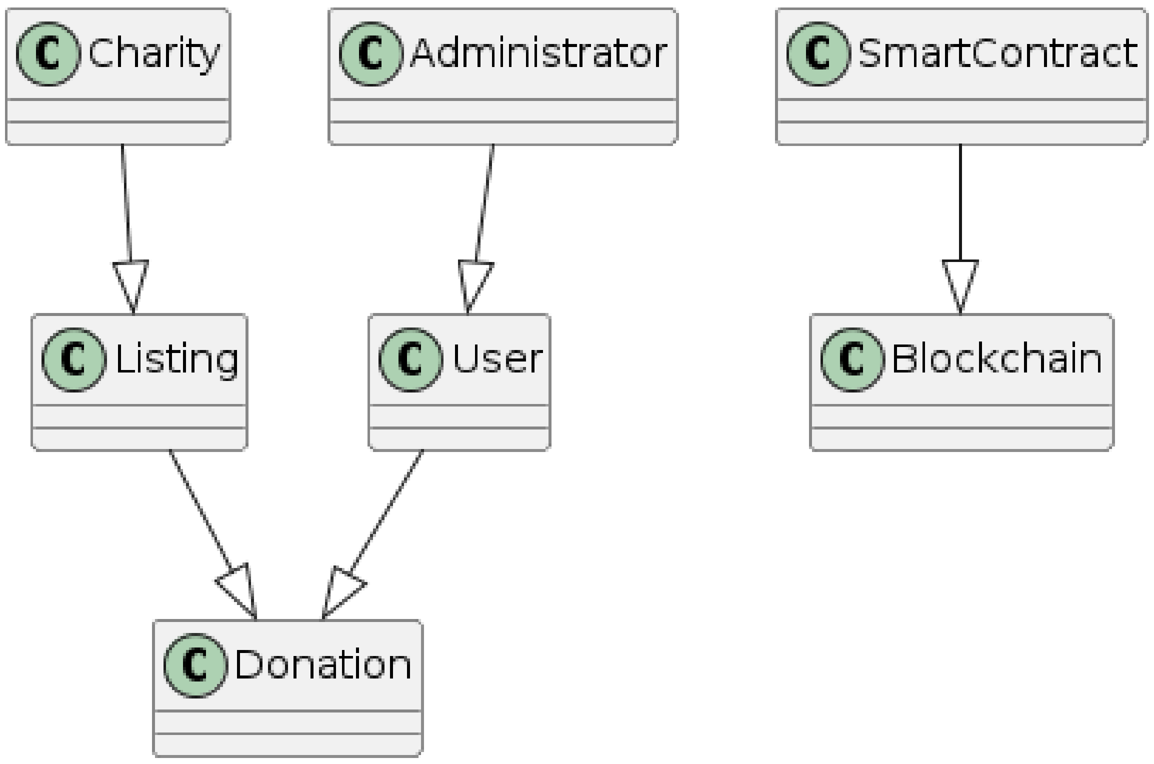
4.2 Sequence Diagram

The sequence diagram shows the flow of interactions between actors and system components for a donation transaction:



4.3 Class Diagram

The class diagram depicts the key classes and their relationships within the CharityNexus system:



6. Appendices

6.1 Glossary

- **CharityNexus**: The decentralized charity donation platform.
- **SRS**: Software Requirements Specification.
- **DApp**: Decentralized Application.
- **UI**: User Interface.
- **API**: Application Programming Interface.
- **Blockchain**: A distributed and immutable ledger technology.
- **Smart Contracts**: Self-executing contracts with predefined rules on the blockchain.

6.2 References

- Blockchain Technology Overview
- Smart Contracts Explained
- Charity Commission for England and Wales