# System Requirements Document (SRD) for Mbole Pay

## 1. \*\*Introduction\*\*

### 1.1 Purpose

Mbole Pay is a web-based platform designed to digitize and automate community savings groups ("njangi" or "tontine") by leveraging smart contracts, automated payments, and transparent record-keeping. This document outlines the functional and non-functional requirements for the system.

### 1.2 Scope

Mbole Pay will:

- Enable digital management of community savings groups.

- Automate transactions, contributions, and payouts using smart contracts.

- Provide secure and transparent financial tracking.

- Support dispute resolution through anonymous voting.

- Offer an accessible, scalable, and secure platform for tontines.

### 1.3 Stakeholders

- \*\*End Users:\*\* Members of savings groups (individuals, businesses, cooperatives).

- \*\*Admins:\*\* Group leaders and financial managers.

- \*\*Developers:\*\* Engineers responsible for system implementation and maintenance.

- \*\*Regulatory Bodies:\*\* Entities governing financial transactions.

---

## 2. \*\*Functional Requirements\*\*

### 2.1 User Management

- \*\*User Registration & Authentication\*\*

- Users can register via email, phone number, or social login.

- Multi-factor authentication (MFA) for enhanced security.

- Admins can approve or reject member applications.

- \*\*Role-Based Access Control\*\*

- Members: Can contribute, view transactions, and vote on disputes.

- Admins: Can create groups, manage members, and oversee financial transactions.

- Super Admins: Platform-wide management and compliance monitoring.

### 2.2 Savings Group Management

- \*\*Create and Manage Groups\*\*

- Admins can create savings groups with customizable rules.

- Members can join/leave groups based on eligibility.

- \*\*Member Contributions & Tracking\*\*

- Automated record-keeping of all member contributions.

- Real-time tracking of total group savings and individual balances.

### 2.3 Payment and Payout Processing

- \*\*Automated Contribution Collection\*\*

- Supports mobile money, bank transfers, and card payments.

- Auto-debit for scheduled contributions via smart contracts.

- \*\*Payout Scheduler\*\*

- First-come, first-paid system based on agreed payout sequence.

- Automated notifications to recipients.

### 2.4 Dispute Resolution

- \*\*Anonymous Voting System\*\*

- Members can raise disputes and vote on resolutions.

- Smart contracts execute majority-approved decisions.

### 2.5 Reporting & Notifications

- \*\*Transaction & Contribution Reports\*\*

- Downloadable member statements.

- Visual dashboards for financial tracking.

- \*\*Real-time Notifications\*\*

- SMS/email alerts for payments, dues, disputes, and payouts.

---

## 3. \*\*Non-Functional Requirements\*\*

### 3.1 Performance

- The system should handle \*\*10,000+ concurrent users\*\*.

- Transaction processing should complete in \*\*under 3 seconds\*\*.

### 3.2 Security

- End-to-end encryption for transactions and user data.

- Smart contract audits to prevent vulnerabilities.

- Secure authentication (OAuth, JWT-based sessions).

### 3.3 Scalability

- \*\*Infrastructure as Code (IaC)\*\* to automate deployment and scaling.

- Isolated environments for each tontine group.

- Load balancing and database partitioning for high availability.

### 3.4 Usability

- Mobile-first responsive design.

- Multilingual support (English, French, Pidgin).

- User-friendly dashboards and transaction history views.

### 3.5 Compliance

- Adherence to \*\*financial regulations\*\* and \*\*data privacy laws\*\* (e.g., GDPR, local banking rules).

- Smart contract governance framework for fair execution.

---

## 4. \*\*System Architecture\*\*

### 4.1 High-Level Architecture

- \*\*Frontend:\*\* React.js (for an interactive user interface).

- \*\*Backend:\*\* Node.js with Express.js (REST API services).

- \*\*Database:\*\* PostgreSQL (transaction management) & IPFS (for immutable records).

- \*\*Blockchain:\*\* Ethereum or Binance Smart Chain (for smart contracts).

- \*\*Payment Gateway:\*\* Integrations with \*\*Flutterwave, Paystack, or MTN Mobile Money\*\*.

### 4.2 Infrastructure

- \*\*Hosting:\*\* AWS/GCP/Azure with auto-scaling.

- \*\*IaC:\*\* Terraform/Ansible to automate environment setup.

- \*\*CI/CD:\*\* GitHub Actions for automated deployments.

---

## 5. \*\*Constraints & Assumptions\*\*

### 5.1 Constraints

- Smart contract execution fees (gas fees) must be optimized.

- Mobile money integration depends on regional providers.

### 5.2 Assumptions

- Users have access to smartphones or computers.

- Tontine groups are willing to adopt digital solutions.

- Members trust the automation process.

---

## 6. \*\*Future Enhancements\*\*

- AI-based fraud detection for abnormal transactions.

- Decentralized Identity (DID) for enhanced member verification.

- Integration with local microfinance institutions.

---

## 7. \*\*Appendices\*\*

- \*\*Appendix A:\*\* API Reference

- \*\*Appendix B:\*\* Smart Contract Logic Breakdown

- \*\*Appendix C:\*\* Security Protocols

---

This document serves as the foundational blueprint for the development of Mbole Pay. Further refinements will be made based on user feedback and regulatory requirements.