ALLAN SHARAD | 665782

IQBAL SHARIFF | 665356

MyChama

Connecting Savings Groups to Smart Financial Tools

School of Science and Technology, United States International University-Africa

APT3065 Mid-Term Project

System Design

1. Introduction

1.1 Purpose of the Document

This System Design Document (SDD) provides a comprehensive overview of the technical architecture, system components, data flows, and interface designs for the MyChama platform. It serves as a bridge between the requirements gathered during the planning phase and the actual implementation. The document outlines the proposed solution's structure, behaviors, and interactions to ensure a shared understanding among team members and stakeholders. Its primary goal is to guide developers, testers, and system architects during development and maintenance of the application.

1.2 Scope of the System

MyChama is a web and mobile application designed to help informal savings groups (commonly referred to as chamas in Kenya) manage their financial and organizational activities. The system will allow members to register, contribute funds, request and approve loans, track repayment schedules, participate in meetings, vote on group decisions, and communicate efficiently through announcements and polls. The system will automate manual processes such as recordkeeping, contribution tracking, and meeting coordination, ensuring transparency, accountability, and ease of use for all group members regardless of technical expertise.

1.3 Intended Audience

This document is intended for all project stakeholders who are involved in the design, development, deployment, and validation of the MyChama system. This includes but is not limited to software developers, UI/UX designers, project managers, testers, product owners, and financial or technical stakeholders. Each section is tailored to address different roles, offering both technical depth and conceptual clarity to support informed decision-making and collaborative development.

1.4 Glossary

- **Chama** – A Swahili term referring to a self-organized savings or investment group.
- **Loan Repayment** – The act of paying back borrowed funds according to agreed terms.
- **Contribution** – A periodic amount paid by a chama member into the group fund.
- **Poll** – A voting mechanism used within the system to make group decisions.
- **API** – Application Programming Interface; allows different parts of the system to communicate.
- **M-Pesa** – A mobile money transfer service widely used in Kenya for digital payments.

2. System Overview and Design Philosophy

2.1 High-Level Description

The MyChama system is designed as a cross-platform application that enables savings groups (chamas) to digitize their financial and administrative operations. It is built on a modular client-server architecture that separates the frontend, backend, and service layers for maintainability and scalability.

The frontend will be developed using React for web and React Native (or Flutter) for mobile applications, offering a responsive and consistent user experience across devices. The backend is built with Node.js and Express.js, providing RESTful APIs to handle authentication, user roles, contributions, loans, announcements, and meetings. Data is stored in MongoDB, a flexible NoSQL database suited to evolving schemas. Notifications and real-time updates are managed via Firebase Cloud Messaging (FCM) or OneSignal, and mobile payments are integrated using M-Pesa APIs. The entire system is deployed on a cloud platform (e.g., Render, Railway, or AWS), with CI/CD pipelines managed via GitHub Actions for automated testing and deployment.

This high-level architecture ensures that the system is loosely coupled, scalable, and able to adapt to growing user needs.

2.2 Design Principles

i. Modularity

The system is divided into well-defined modules, each responsible for a specific domain (e.g., finance, communication, meetings). This allows for parallel development, easier maintenance, and reuse of components.

ii. Reusability

UI components, backend services, and business logic are designed to be reusable across different parts of the application. Shared libraries and hooks are implemented to reduce duplication and improve consistency.

iii. Scalability

The system supports horizontal scaling through stateless APIs and microservice-ready structures. MongoDB's document-based model allows the application to handle high volumes of transactions and data growth over time without performance degradation.

iv. Performance

Caching strategies, asynchronous operations, and efficient API routing ensure fast response times and minimal load on the server. The use of lazy loading and efficient state management in the frontend further optimizes user experience.

v. Security

User authentication is secured using JWT (JSON Web Tokens) with role-based access control (RBAC) enforced throughout the API. Sensitive information such as payment data and credentials are stored securely and environment variables are used to manage secrets. GitHub branch protection rules, secret scanning, and 2FA are enforced to secure the codebase.

Together, these principles ensure that MyChama is reliable, maintainable, secure, and capable of supporting the digital transformation of informal savings groups in Kenya and beyond.

3. Architectural Design

3.1 Architecture Diagram

The architecture of MyChama is based on a modular client-server model, designed to support future microservices expansion within the backend. The following component diagram provides a high-level view of the system's major functional units and their interactions.

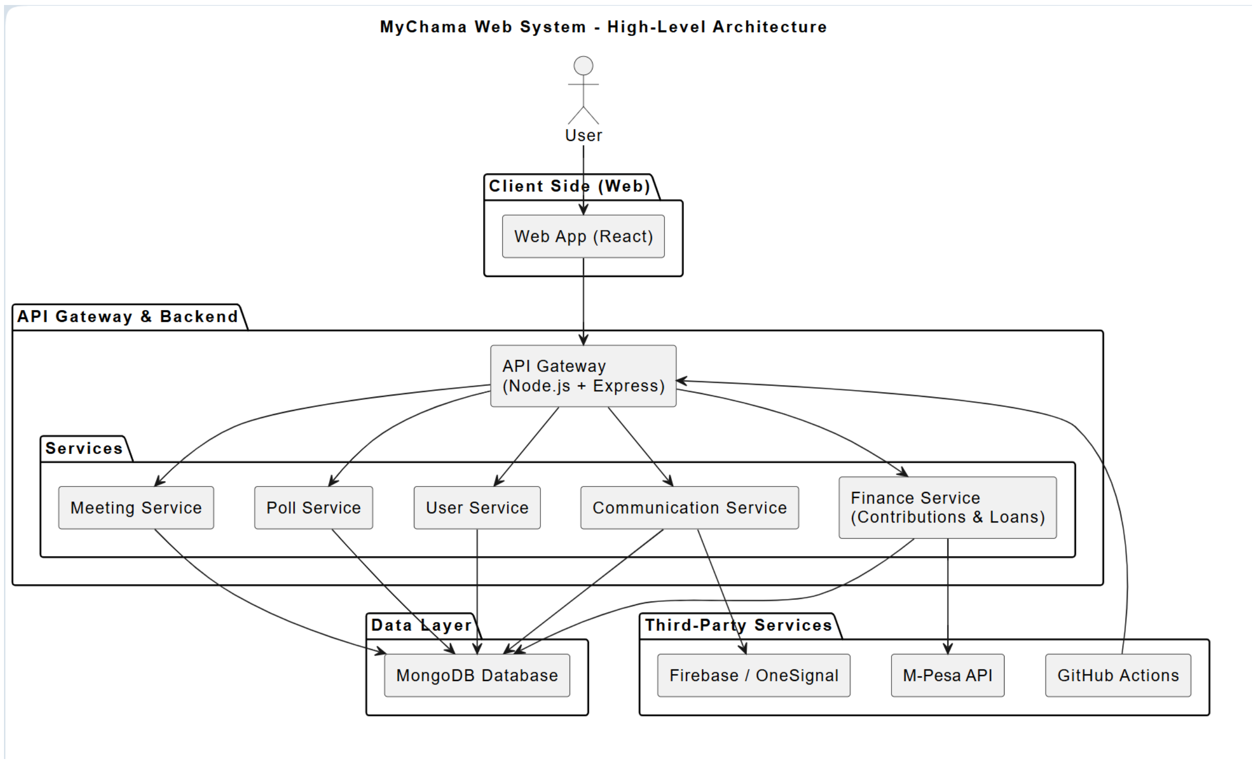


Figure 1: Component Diagram

The component diagram above illustrates the high-level architectural structure of the MyChama web application. It is based on a modular, layered architecture using a client-server model. The system is composed of a React-based web frontend, which communicates with a centralized API

Gateway built using Node.js and Express. This API gateway routes requests to specialized backend services (components) including:

- User Service – manages authentication, roles, and user data.
- Finance Service – handles contributions, loan requests, approvals, and repayments.
- Meeting Service – schedules group meetings and manages attendance.
- Poll Service – enables member voting and decision-making through polls.
- Communication Service – supports announcements and push notifications.

All services interact with a shared MongoDB database, which stores structured records for users, financial transactions, meetings, and messages. The system integrates with third-party services such as M-Pesa API for mobile payments, and Firebase/OneSignal for sending real-time notifications. Deployment automation and CI/CD pipelines are managed using GitHub Actions. This architecture ensures a clean separation of concerns, promotes scalability, and supports secure and efficient web-only operation of the MyChama platform.

3.2 Architecture Style

MyChama adopts a layered architecture built on a client-server model, promoting separation of concerns and modular development. At the presentation layer, the system features a responsive web interface developed using React, allowing users to interact with various system features through a modern and intuitive UI. The application layer consists of a centralized API gateway built with Node.js and Express, responsible for routing client requests and enforcing authentication and authorization. The service layer is composed of modular backend services that encapsulate the core business logic, including user management, finance operations (contributions and loans), meetings, communication, and polling. These services interact with the data layer, which uses MongoDB as the primary database for storing user data, financial records, meeting schedules, and communication logs. Lastly, the external integration layer connects the platform to third-party services such as M-Pesa for mobile payments and Firebase/OneSignal for delivering real-time notifications. This structured architectural style ensures that the system remains scalable, maintainable, and adaptable to future enhancements or distributed deployments.

3.3 Component Description

i. Frontend

- Built with React (Web) and React Native.
- Handles user interactions, form submissions, and data visualization (e.g., graphs, dashboards).
- Communicates with the backend via HTTPS and JWT-based authentication.

ii. API Gateway (Node.js + Express)

- Acts as the single-entry point for all API calls.
- Routes requests to the appropriate microservice.
- Manages authentication tokens, request validation, and role-based access control.

iii. User Service

- Handles user registration, login, profile updates, and role assignments (e.g., Chairperson, Member).
- Integrates with authentication logic (JWT issuance and verification).

iv. Finance Service

- Manages contributions, loan requests, approvals, and repayments.
- Tracks transaction history and emits notifications upon events.
- Integrates with M-Pesa APIs for payment processing.

v. Communication Service

- Enables group announcements and direct member notifications.
- Connects to Firebase Cloud Messaging (FCM) or OneSignal to deliver push notifications.

vi. Meeting Service

- Manages scheduling of meetings and stores agendas, minutes, and attendance.
- Sends calendar invites and reminders through the notification service.

vii. Poll Service

- Allows group members to create and participate in votes on group matters.
- Supports anonymous and timed polls.

viii. Database (MongoDB)

- NoSQL database storing user records, contributions, loan data, meetings, polls, and messages.
- Uses schema validation and indexing to optimize query performance.

ix. Third-Party Integrations

- M-Pesa: For sending and receiving mobile money payments.
- Firebase/OneSignal: For real-time push notifications.
- GitHub Actions: For CI/CD automation and secret management.

3.4 Interaction Flow Summary

When a user accesses the MyChama platform, they begin by logging in through the web frontend interface. After submitting login credentials, the frontend sends an authenticated request to the API Gateway. The gateway is responsible for validating the user's token and determining the appropriate backend service to handle the request. For example, financial actions are routed to the Finance Service, while user updates go to the User Service.

Once the request reaches the designated backend service, the business logic is executed. This may involve reading from or writing to the MongoDB database, performing calculations, or validating input data. If additional communication is needed, such as confirming a contribution or loan approval, the service may trigger the Notification Service to send real-time updates using Firebase Cloud Messaging or OneSignal.

After the backend processes the request and responds, the frontend updates the user interface based on the result. This could include displaying updated dashboards, status messages, or confirmation alerts. This asynchronous and service-driven interaction model ensures a responsive user experience while maintaining system modularity and scalability.

4. Detailed Design

This section provides a breakdown of the main modules of the MyChama web application. Each module description outlines its responsibilities, data flow, rules, and interactions with other components of the system. These modules map directly to the backend services defined in the architecture.

4.1 Module Descriptions

4.1.1 User Management Module

Functionality

Handles user registration, login, authentication, profile updates, and role assignment (e.g., Member, Chairperson, Treasurer).

Inputs

- Registration data (name, email/phone, password)
- Login credentials
- Profile updates

- Role selection or assignment

Outputs

- JWT authentication token
- Profile information
- Success/error messages

Business Rules

- Each user must register with a unique phone number or email.
- Only admins can assign roles beyond "Member."
- All data must be validated (e.g., strong password, required fields).

Data Elements Involved

- user_id, full_name, email, phone_number, password_hash, role, date_joined

4.1.2. Contributions Module

Functionality

Allows members to view, make, and track financial contributions to the chama fund.

Inputs

- Contribution amount
- Payment method (e.g., M-Pesa)
- Chama ID
- User ID

Outputs

- Confirmation message
- Contribution history
- Updated total balance

Business Rules

- Contributions can only be made by registered members.
- The system must verify successful payment before recording the transaction.
- Duplicate contributions within a defined timeframe must be flagged.

Data Elements Involved

- contribution_id, user_id, chama_id, amount, payment_method, contribution_date, status

4.1.3. Loan Management Module

Functionality

Enables members to request loans, while chairpersons and treasurers review, approve, and track loan status and repayments.

Inputs

- Loan amount
- Purpose
- User ID
- Loan status updates (e.g., approved, rejected)

Outputs

- Approval/rejection status
- Repayment schedule
- Notifications to stakeholders

Business Rules

- A member must meet eligibility requirements (e.g., consistent contributor, no outstanding loans).
- Loan approval requires both Chairperson and Treasurer confirmation.

- Interest and repayment rules must be configurable.

Data Elements Involved

- loan_id, user_id, amount, status, approved_by, repayment_schedule, issue_date

4.1.4 Meeting Management Module

Functionality

Allows group leaders to schedule meetings, track attendance, and manage agendas.

Inputs

- Meeting title
- Date and time
- Agenda notes
- Participant list

Outputs

- Calendar view
- Reminders
- Meeting records

Business Rules

- Only authorized roles (Chairperson/Secretary) can schedule meetings.
- Users must be notified at least 24 hours in advance.
- Attendance tracking is optional but supported.

Data Elements Involved

- meeting_id, chama_id, date, time, agenda, organizer_id, attendance_status

4.1.5 Communication & Notification Module

Functionality

Facilitates group-wide announcements, reminders, and real-time alerts using Firebase or OneSignal.

Inputs

- Message content
- Target recipients
- Message type (announcement, alert, reminder)

Outputs

- Notification messages
- Delivery status

Business Rules

- Only Chairperson and Treasurer can post announcements.
- Reminders are sent based on scheduled meeting or loan events.
- Notifications must not reveal sensitive financial data.

Data Elements Involved

- message_id, sender_id, message_type, timestamp, delivery_status

4.1.6 Polling Module

Functionality

Allows group members to vote on decisions through time-limited polls.

Inputs

- Poll question
- List of options
- Voting deadline

Outputs

- Aggregated vote results
- Poll status

Business Rules

- One vote per member per poll
- Polls close automatically at the specified deadline
- Poll creators cannot vote after publishing

Data Elements Involved

- poll_id, creator_id, question, options[], start_time, end_time, votes[]

4.2 Interface Design

4.2.1 User Interface

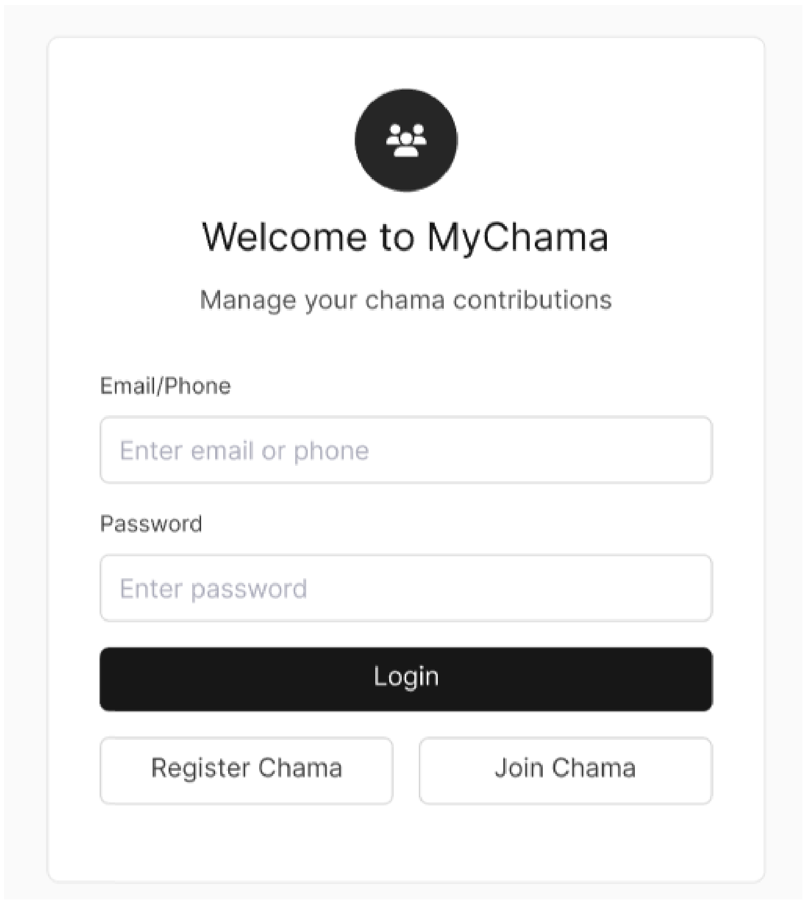
The user interface (UI) of MyChama has been designed using Figma, a collaborative design tool that enables high-fidelity wireframing and prototyping. The Figma UI focuses on delivering a clean, intuitive, and responsive experience optimized for web use. The design incorporates minimalistic layouts, Swahili-English hybrid labels, and consistent visual hierarchies for both desktop and mobile browser views.

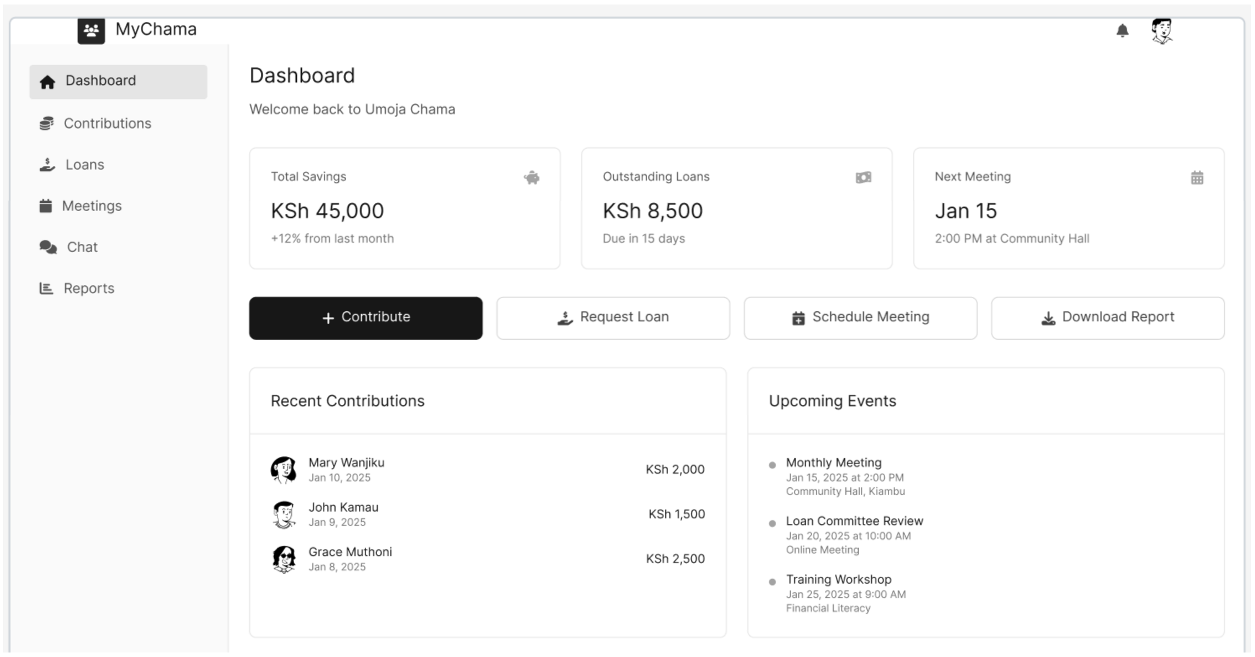
The primary screens designed in Figma include:

- Login and registration pages with authentication input fields and role selection.
- A dashboard that summarizes key metrics such as total savings, active loans, and upcoming meetings.
- Contribution and loan interface that allow users to input amounts, view history, and submit requests.

- A Meetings calendar where users can view, schedule, and receive reminders for upcoming events.
- Communication modules such as announcements, group chat, and polls for collective decision-making.
- A reports page that presents contribution summaries and loan trends using visual charts and downloadable files.

Each Figma screen is structured using a modular grid system and reusable components to maintain visual consistency across the application. The designs also account for accessibility, with clear buttons, readable fonts, and logical navigation flows.





MyChama

Dashboard

Contributions

Loans

Meetings

Chat

Reports

Dashboard

Welcome back to Umoja Chama

Total Savings	Outstanding Loans	Next Meeting
KSh 45,000	KSh 8,500	Jan 15
+12% from last month	Due in 15 days	2:00 PM at Community Hall

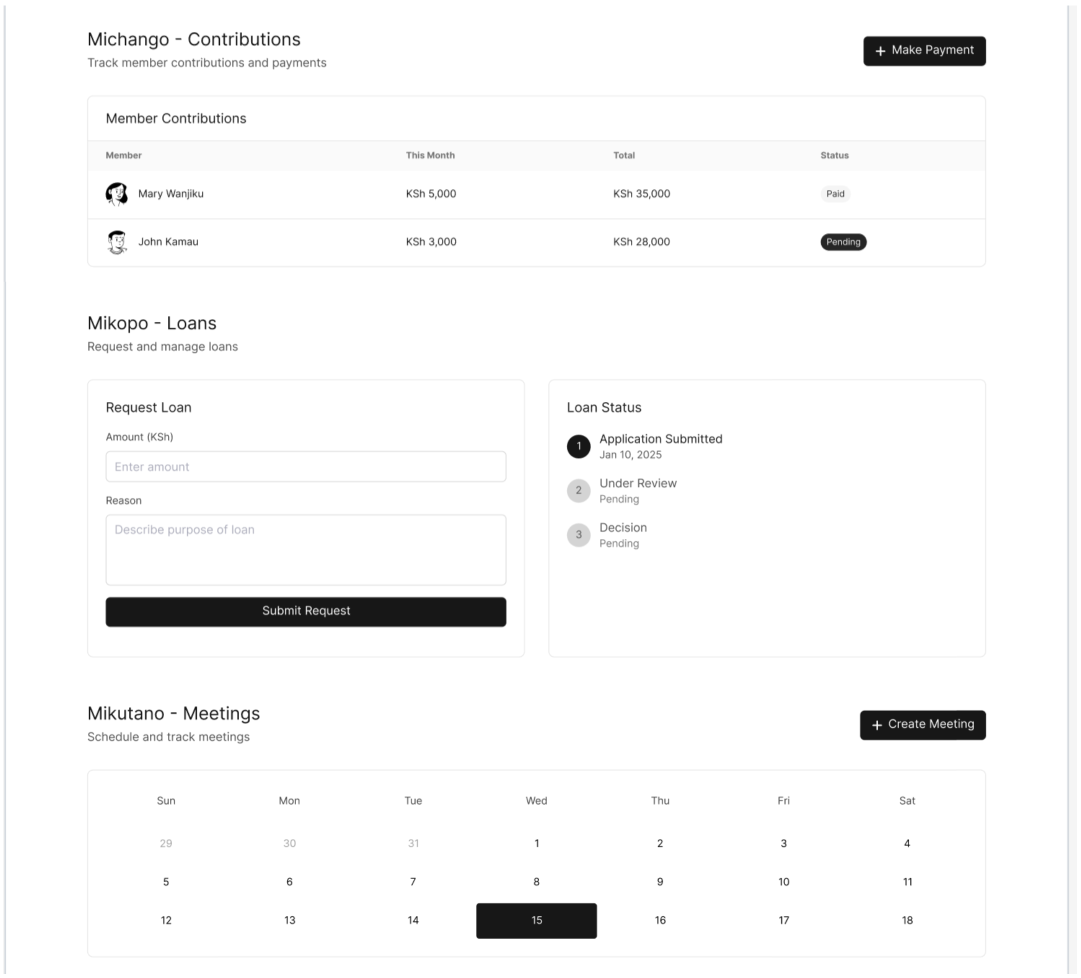
+ Contribute | Request Loan | Schedule Meeting | Download Report

Recent Contributions

Mary Wanjiku Jan 10, 2025	KSh 2,000
John Kamau Jan 9, 2025	KSh 1,500
Grace Muthoni Jan 8, 2025	KSh 2,500

Upcoming Events

- Monthly Meeting
  Jan 15, 2025 at 2:00 PM
  Community Hall, Kiambu
- Loan Committee Review
  Jan 20, 2025 at 10:00 AM
  Online Meeting
- Training Workshop
  Jan 25, 2025 at 9:00 AM
  Financial Literacy



Michango - Contributions

Track member contributions and payments

+ Make Payment

Member Contributions

Member	This Month	Total	Status
Mary Wanjiku	KSh 5,000	KSh 35,000	Paid
John Kamau	KSh 3,000	KSh 28,000	Pending

Mikopo - Loans

Request and manage loans

Request Loan

Amount (KSh)

Enter amount

Reason

Describe purpose of loan

Submit Request

Loan Status

1 Application Submitted
Jan 10, 2025

2 Under Review
Pending

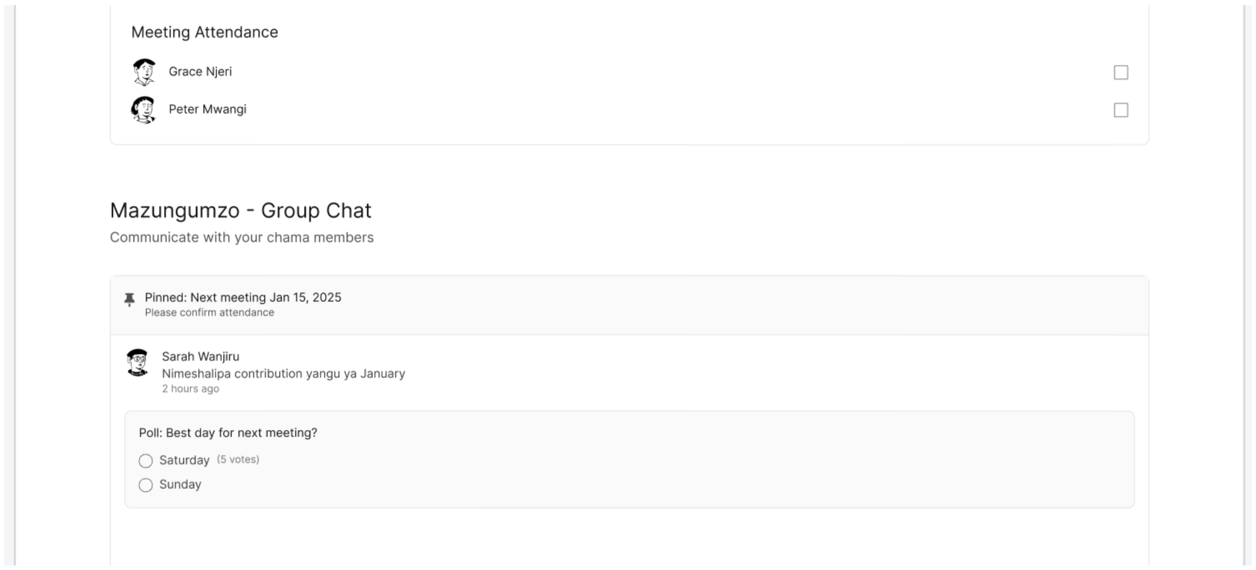
3 Decision
Pending

Mikutano - Meetings

Schedule and track meetings

+ Create Meeting

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18


Meeting Attendance

Grace Njeri ☐

Peter Mwangi ☐

Mazungumzo - Group Chat

Communicate with your chama members

Pinned: Next meeting Jan 15, 2025

Please confirm attendance

Sarah Wanjiru

Nimeshalipa contribution yangu ya January

2 hours ago

Poll: Best day for next meeting?

◯ Saturday (5 votes)

◯ Sunday

4.2.2 Navigation Flow (Web Application)

1. Login / Registration Page

→ Upon successful login, redirect to Dashboard.

2. Dashboard

→ Access to:

▪ My Contributions

▪ Loan Requests

▪ Meetings & Events

▪ Announcements / Polls

▪ Reports

▪ Settings / Logout

3. Contributions

→ View history → Click "Contribute" → Enter amount → Submit.

4. Loans

→ View loan status → Click "Request Loan" → Submit request → Wait for approval.

5. Meetings

→ View calendar → Click meeting for details → RSVP or add notes.

6. Announcements / Polls

→ View list → Read / vote → View results or post new if authorized.

4.2.3 API Interface Specifications

Below are example API endpoints used in the MyChama system:

i. User Login

Endpoint URL: POST /api/auth/login

HTTP method: POST

Request body:

json

```
{
  "email": "user@example.com",
  "password": "securePassword123"
}
```

Response:

json

```
{
  "token": "jwt-token-string",
  "user": {
    "id": "u123",
    "name": "Jane Doe",
    "role": "Member"
  }
}
```

Status codes

- 200 OK – Login successful

- 401 Unauthorized – Invalid credentials

ii. Make Contribution

Endpoint URL: POST /api/contributions

HTTP method: POST

Request body:

json

```
{
  "userId": "u123",
  "amount": 500,
  "paymentMethod": "M-Pesa"
}
```

Response:

json

```
{
  "message": "Contribution recorded successfully",
  "transactionId": "c987"
}
```

Status Codes:

- 201 Created
- 400 Bad Request – Missing fields

iii. Request Loan

Endpoint URL: POST /api/loans/request

HTTP method: POST

Request body:

json

```json
{
  "userId": "u123",
  "amount": 2000,
  "purpose": "Emergency fund"
}
```

Response:

json

```json
{
  "status": "pending",
  "loanId": "l789",
  "message": "Loan request submitted for review"
}
```

Status Codes:

- 202 Accepted
- 403 Forbidden – Not eligible

iv. Get Upcoming Meetings

Endpoint URL: GET /api/meetings/upcoming

HTTP Method: GET

Response:

json

```json
[
  {
    "meetingId": "m001",
    "title": "Monthly Planning",
    "date": "2025-08-10T14:00:00Z",
    "location": "Chama Hall"
  }
]
```

Status Codes:

- 200 OK
- 404 Not Found – No meetings scheduled