MeshFund: Expo + Backend Project Structure & Setup Guide

**VISION AND APP DESCRIPTIONS**

# 1. Core Concept

MeshFund is a decentralized, peer-governed financial platform that enables communities — local or global — to create and manage their own savings systems, peer lending networks, and investment circles. It empowers trusted groups to pool, circulate, and withdraw money from anywhere in the world, removing the barriers of geography, bureaucracy, and central control.

# 2. Mission Statement

To make finance borderless, communal, and human — by giving people everywhere the power to build and manage their own money systems.

# 3. Unique Value Proposition (UVP)

What sets MeshFund apart globally:

|  |  |
| --- | --- |
| Feature | Global Value |
| Cross-Border Financial Groups | Diaspora families, international student circles, and remote teams can save and invest together, no matter where they are. |
| Anywhere Access, Anytime Use | Members can join from different countries and retrieve money in their local currency via linked wallets or payment partners. |
| Peer-Led Governance | Each group sets its own rules — contribution amounts, payout schedules, penalties — with no external interference. |
| Borderless Withdrawals | Integrated with payment APIs and MoMo gateways to allow funds to be cashed out locally, even across currencies. |
| Trust Without Borders | Built-in transparency, dispute resolution, and reputation systems make trust possible across continents. |
| No Bank Dependency | Enables access for the underbanked, mobile-first users, and users in countries with weak financial infrastructure. |

# 4. Problems MeshFund Solves

- Global communities lack financial tools for collaboration. Most group savings apps are either local or built for formal institutions.

- Traditional finance doesn’t work for distributed families or collectives. There’s no safe way to manage money together across countries without relying on costly remittance tools or banks.

- Trust is fragile without transparency. Informal money groups often collapse due to disputes, mismanagement, or lack of enforcement.

# 5. The Future MeshFund Envisions

- A Kenyan nurse in Italy, a developer in Accra, and a mother in Tamale forming a shared savings group with instant visibility and fair payout logic.

- A diaspora collective funding a solar project in their hometown, tracking every contribution and expenditure transparently.

- An online creative community pooling investment funds and supporting members without needing bank accounts.

# 6. Strategic Positioning

Primary Use Cases:

- Diaspora savings and family contribution circles

- Rotational savings groups across borders

- Peer-managed investment groups

- Informal credit unions in underserved regions

# 1. Project Structure (Monorepo - React Native + API)

Recommended folder layout for managing both frontend (Expo) and backend in a single workspace:

/meshfund  
 /apps  
 /mobile → Expo React Native app  
 /api → Node.js backend (Express or tRPC)  
 /packages  
 /ui → Shared UI components (buttons, inputs)  
 /lib → Shared utilities and constants  
 /db → Prisma/Drizzle schema & migrations  
 /docs → Vision, roadmap, flows, etc.  
.env → Environment variables  
pnpm-workspace.yaml or package.json (for monorepo)

**Mobile App (Expo):**

Auth screens

Group creation + join flow

Group dashboard

Contributions screen

Payout tracking screen

Notification screen

Navigation + UX polish

**Backend:**

Auth endpoints (signup, login)

Group management logic

Contribution and payout logic

User verification + validation

Push notification scheduler (cron)

# 2. Expo Setup Instructions (Frontend)

* - Install Node.js and pnpm (recommended)
* - Run: npx create-expo-app apps/mobile --template
* - Enable TypeScript support
* - Install TailwindCSS via nativewind and configure babel + tailwind.config.js
* - Set up routing with react-navigation
* - Create base screens: Splash, Auth, Dashboard, Group, Contribution, Payout
* - Create reusable UI components in packages/ui
* - Use AsyncStorage for temporary state
* - Test live using Expo Go on Android and iPhone
* - Build production APK/IPA with EAS: eas build --platform android / ios

# 3. Backend Setup (Node.js + Express or tRPC)

* - Create apps/api folder and initialize with pnpm init or npm init
* - Use Express (or tRPC) with TypeScript for API routes
* - Connect to Neon or Supabase PostgreSQL database
* - Use Prisma or DrizzleORM to define schema
* - Set up folders: /routes, /controllers, /services, /models
* - Create .env for DATABASE\_URL and JWT\_SECRET
* - Use Zod or Joi for request validation
* - Secure routes using JWT tokens and middleware
* - Host locally (dev) or deploy via Railway/Render for production

# 4. Backend Feature Responsibilities

## Backend Design

* - Define modular API structure with TypeScript
* - Database tables: Users, Groups, Contributions, Payouts
* - Secure API endpoints with JWT middleware

## Frontend Design

* - Create mobile-first React Native layouts
* - Implement clean onboarding → dashboard → group flow
* - Use Zustand or Context API for state management

## Implement Auth

* - API: /signup /login routes (POST)
* - Hash passwords using bcrypt
* - Return JWT upon login
* - Frontend: Form validation, store token securely, auto login

## Mesh Group Logic

* - Create group endpoint: name, rules, target amount, members
* - Join group endpoint with verification
* - Lock group when full and track active cycle
* - Admin sets payout order or system assigns based on rules

## Contribution Tracking

* - Track payment status per cycle per member
* - Mark late, paid, missed contributions
* - Allow retries before deadline with limits

## Payout System

* - Auto-trigger payout when all members pay
* - Select user in queue for payout (by order or votes)
* - Log payout details: amount, time, recipient, status

## Notifications

* - In-app alerts: payment reminder, payout success
* - Optional push notifications using Expo Notifications API
* - Server-side cron to check for due contributions and send alerts