MeshFund Schema Modeling & Auth Flow

# 1. Database Schema Modeling (MVP)

Use PostgreSQL with Prisma or Drizzle for modeling. These are the key tables and their core fields:

## User

* - id (UUID, primary key)
* - email (string, unique)
* - passwordHash (string)
* - fullName (string)
* - phoneNumber (optional, string)
* - createdAt (timestamp)
* - updatedAt (timestamp)

## Group

* - id (UUID, primary key)
* - name (string)
* - adminId (foreign key to User)
* - contributionAmount (decimal)
* - frequency (enum: weekly, monthly, etc.)
* - payoutOrder (json or array of user IDs)
* - isLocked (boolean)
* - currentCycle (int)
* - maxMembers (int)
* - createdAt (timestamp)

## GroupMember

* - id (UUID, primary key)
* - groupId (foreign key to Group)
* - userId (foreign key to User)
* - joinedAt (timestamp)
* - isActive (boolean)
* - reputationScore (int)

## Contribution

* - id (UUID, primary key)
* - groupId (foreign key)
* - userId (foreign key)
* - amount (decimal)
* - cycleNumber (int)
* - status (enum: paid, missed, late)
* - paidAt (timestamp)

## Payout

* - id (UUID, primary key)
* - groupId (foreign key)
* - userId (foreign key, recipient)
* - cycleNumber (int)
* - amount (decimal)
* - status (enum: pending, completed)
* - paidAt (timestamp)

# 2. Authentication Flow

The authentication system should use secure best practices, powered by JWT and password hashing.

Signup/Login/Logout Flow:

* - User signs up with email + password (phone optional for now)
* - Password is hashed using bcrypt before storage
* - User receives a success response with a JWT (signed with secret key)
* - Client stores JWT securely (AsyncStorage) and attaches it to API requests
* - Middleware on backend verifies JWT for all protected endpoints
* - Login route checks credentials, re-issues a new token if valid
* - Logout simply deletes the token client-side (stateless auth)
* - Add token expiration (e.g., 7 days), and optionally implement refresh token logic later