Low Level Design

1. **Database Schema Design**
    a. **accounts**
        i. Primary key using UUID
        ii. Columns for name, type (enum)
        iii. Timestamps for created_at and updated_at
    b. **transactions**
        i. Primary key using UUID
        ii. Columns for reference_no, date, narration
        iii. Timestamps for created_at and updated_at
    c. **entries**
        i. Primary key using UUID
        ii. Foreign keys to transactions and accounts
        iii. Columns for amount, type (enum)
        iv. Timestamps for created_at and updated_at
2. **Project Structure**
    a. **Module Structure**
        i. Account Module
            1. Controller, Service, Entity, DTOs, Tests
        ii. Transaction Module
            1. Controller, Service, Entity, DTOs, Tests
        iii. Entry Module
            1. Service, Entity, DTOs
    b. **Shared Components**
        i. **Filters**: Exception handlers
        ii. **Guards**: Authorization
        iii. **Interceptors**: Response transformers
        iv. **Pipes**: Validation
        v. **Utils**: Helper functions
3. **API Endpoints Design**
    a. **Account Endpoints**
        i. **POST /accounts**: Create a new account
        ii. **GET /accounts**: List all accounts
        iii. **GET /accounts/:id**: Get account by ID
        iv. **GET /accounts/:id/balance**: Get account balance
    b. **Transaction Endpoints**
        i. **POST /transactions**: Create a transaction with entries
        ii. **GET /transactions**: List transactions with optional filters
            1. Query parameters: accountId, startDate, endDate
        iii. **GET /transactions/:id**: Get transaction by ID

4. **Data Transfer Objects (DTOs)**
    a. **Account DTOs**
        i. **CreateAccountDto**: name, type
        ii. **AccountResponseDto**: id, name, type, timestamps
        iii. **AccountBalanceDto**: id, name, balance
    b. **Transaction DTOs**
        i. **CreateTransactionDto**: reference_no, narration, entries (array)
        ii. **CreateEntryDto**: account_Id, amount, type, transaction_id
        iii. **TransactionResponseDto**: id, reference_no, date, narration, entries, timestamps
5. **Service Layer Components**
    a. **Account Service**
        i. create(): Create a new account
        ii. findAll(): Get all accounts
        iii. findOne(): Get account by ID
        iv. getBalance(): Calculate account balance
    b. **Transaction Service**
        i. create(): Create transaction with validation for balanced entries
        ii. findAll(): Get transactions with filtering
        iii. findOne(): Get transaction by ID
6. **Validations**
    a. Ensure entries balance (total debits equal total credits)
    b. Convert amount to lowest denomination (e.g.cents) for calculations/saving to avoid floating-point errors
7. **Development and Deployment**
    a. **Development Environment**
        i. Docker Compose for local development
        ii. Environment variables for configuration
    b. **Deployment**
        i. Docker image
        ii. Database migrations
8. **Implementation Plan**
    a. Set up project structure and database
    b. Implement entity models and repositories
    c. Develop core business logic in services
    d. Build API controllers and validation
    e. Add error handling and documentation
    f. Write tests
    g. Create Docker configuration