Backend Path

Learning Resources

- Golang Tutorial : Go Full Course
- Golang Doc

1. Project Setup and Basic Structure

1.1 Project Initialization

- Set up a Go module with proper package structure
- Implement dependency management using Go modules
- Create a configuration system using environment variables
- Set up a logging framework (e.g., zerolog, zap, or slog)
- Implement graceful shutdown handling

1.2 Database Design and Setup

- Design database schema with proper relationships and indices
- Implement database migrations system

Table Name	Columns
users	id, username, email, password_hash, role, created_at, updated_at
transactions	id, from_user_id, to_user_id, amount, type, status, created_at
balances	user_id, amount, last_updated_at

Table Name	Columns
audit_logs	id, entity_type, entity_id, action, details, created_at

2. Core Implementation

2.1 Domain Models and Interfaces

- Implement domain models using Go structs:
  - User struct with methods for validation
  - Transaction struct with state management
  - Balance struct with thread-safe operations
- Define interfaces for services and repositories
- Implement JSON marshaling/unmarshaling for all models

2.2 Concurrent Processing System

- Create a worker pool for processing transactions
- Implement a transaction queue using channels
- Use sync.RWMutex for thread-safe balance updates
- Implement atomic counters for transaction statistics
- Create a concurrent task processor for batch operations

2.3 Core Services

- **UserService**:
  - User registration with password hashing
  - User authentication
  - Role-based authorization
- **TransactionService**:
  - Credit/debit operations
  - Transfer between accounts
  - Transaction rollback mechanism
- **BalanceService**:
  - Thread-safe balance updates
  - Historical balance tracking
  - Balance calculation optimization

3. API Implementation

3.1 HTTP Server Setup

- Implement a custom router with middleware support
- Set up CORS and security headers
- Implement rate limiting
- Add request logging and tracking

3.2 API Endpoints

- **Authentication Endpoints:**
  - `POST /api/v1/auth/register`
  - `POST /api/v1/auth/login`
  - `POST /api/v1/auth/refresh`
- **User Management Endpoints:**
  - `GET /api/v1/users`
  - `GET /api/v1/users/{id}`
  - `PUT /api/v1/users/{id}`
  - `DELETE /api/v1/users/{id}`
- **Transaction Endpoints:**
  - `POST /api/v1/transactions/credit`
  - `POST /api/v1/transactions/debit`
  - `POST /api/v1/transactions/transfer`
  - `GET /api/v1/transactions/history`
  - `GET /api/v1/transactions/{id}`
- **Balance Endpoints:**
  - `GET /api/v1/balances/current`
  - `GET /api/v1/balances/historical`
  - `GET /api/v1/balances/at-time`

3.3 Middleware Implementation

- Authentication middleware
- Role-based authorization middleware
- Request validation middleware
- Error handling middleware
- Performance monitoring middleware

4. Deployment and DevOps

4.1 Docker Setup

- Create multi-stage Dockerfile
- Implement docker-compose with:
    - Application service
    - Database service
    - Redis for caching
    - Monitoring services

4.2 Monitoring and Observability

- Implement Prometheus metrics
- Set up Grafana dashboards
- Add distributed tracing
- Implement structured logging

5. Extra Features

- **Implement Event Sourcing:**
    - Store all state changes as events
    - Rebuild state from event stream
    - Implement event replay functionality
- **Add Caching Layer:**
    - Implement Redis caching
    - Cache invalidation strategies
    - Cache warm-up mechanisms
- **Implement Advanced Features:**
    - Scheduled transactions
    - Batch transaction processing
    - Transaction limits and rules
    - Multi-currency support
- **High Availability Setup:**
    - Database replication
    - Load balancing
    - Circuit breaker implementation
    - Fallback mechanisms

6: Dockerize

- Dockerize your project after watching the tutorial.
- You need to create Dockerfile for your project.
- Your need to use docker-compose.