# Fund Transfer Application (FundTransferApp)

This Go-based application provides a lightweight API for internal fund transfers between user accounts using HTTP endpoints for submitting transaction details and querying account balances. It supports new account creation with initial balance (zero balance is permitted), account balance inquiry, and fund transfer between accounts. Postgres database is used to store account state and transactions. Go version 1.24.5 and Postgres version 17.5.3 is used for this application.

## ✨ Features

* RESTful API endpoints for account management and transactions
* PostgreSQL-backed account and transaction ledger
* Optimistic concurrency control using last\_updated timestamps
* Structured JSON responses with custom status and error codes
* Automatic retry mechanism for concurrency conflicts

## ⚙️ API Endpoints

### 1. Create Account

**Endpoint**: POST /accounts

**Request Body:**

{  
 "account\_id": 123,  
 "initial\_balance": 100.50  
}

**Success Response:**

{  
 "status": "success",  
 "code": 2001,  
 "message": "Account created",  
 "data": {  
 "account\_id": 123,  
 "initial\_balance": 100.5  
 }  
}

### 

### 2. Get Account Details

**Endpoint**: GET /accounts/{account\_id}

**Success Response:**

{  
 "status": "success",  
 "code": 2002,  
 "message": "Account retrieved",  
 "data": {  
 "account\_id": 123,  
 "balance": 100.5  
 }  
}

### 3. Transfer Funds

**Endpoint**: POST /transactions

**Request Body:**

{  
 "source\_account\_id": 123,  
 "destination\_account\_id": 456,  
 "amount": 25.75  
}

**Success Response:**

{  
 "status": "success",  
 "code": 2003,  
 "message": "Transfer successful",  
 "data": {  
 "source\_account\_id": 123,  
 "destination\_account\_id": 456,  
 "amount": 25.75  
 }  
}

## 

## 📊 Assumptions

* All accounts use the same currency (e.g., USD)
* No authentication or authorization required
* Floating point amounts are acceptable for this prototype

## 📖 Request/Response Codes

| Code | Meaning |
| --- | --- |
| 2001 | Account created |
| 2002 | Account retrieved |
| 2003 | Transfer successful |
| 1001 | Method not allowed |
| 1002 | Invalid request payload |
| 1003 | Account already exists |
| 1005 | General DB error |
| 1015 | Insufficient funds |
| 1016 | Concurrency error on debit |
| 1018 | Concurrency error on credit |

## 🚀 Setup & Run Instructions

### 🐘 PostgreSQL Setup

# 1. Install PostgreSQL  
sudo apt update  
sudo apt install postgresql postgresql-contrib  
  
# 2. Start PostgreSQL and login using postgres account and “postgres” password  
sudo service postgresql start  
sudo -u postgres psql  
  
# 3. Create DB and table  
CREATE DATABASE bank;  
\c bank  
  
CREATE TABLE accounts (  
 id INT PRIMARY KEY,  
 balance NUMERIC NOT NULL,  
 last\_updated TIMESTAMP NOT NULL  
);  
  
CREATE TABLE transactions (  
 id SERIAL PRIMARY KEY,  
 from\_account INT,  
 to\_account INT,  
 amount NUMERIC NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

### 🧰 Go Setup

# 1. Install Go  
# Visit https://golang.org/doc/install and follow the instructions for your OS  
  
# 2. Clone the project  
git clone <your-github-url>  
cd <project-folder>

# 4. Initialize the Go module  
Set the environment variables to point to go installation path like GOPATH, GOROOT, GOMODCACHE  
  
# 5. Initialize the Go module  
go mod init FundTransferApp  
  
# 6. Install dependencies  
go get github.com/lib/pq  
  
# 7. Run the server  
go run FundTransferApp.go

Server will start at: http://localhost:8081

## 🌐 Testing With cURL or Postman

### Create Account

curl -X POST http://localhost:8081/accounts \  
 -H "Content-Type: application/json" \  
 -d '{"account\_id":101,"initial\_balance":500}'

### Get Account

curl http://localhost:8081/accounts/101

### Transfer

curl -X POST http://localhost:8081/transactions \  
 -H "Content-Type: application/json" \  
 -d '{"source\_account\_id":101,"destination\_account\_id":102,"amount":50}'

## ✏️ License

This project is free to use under the MIT license.