Banking System API Documentation – Christian Carreon

Overview: A REST API allowing users to manage their bank accounts. It was made clear that grading would be focused on coding style, organization, testability and test coverage. Given this information, assumptions and simplifications were made to the code base.

- Assumptions/Simplifications:
  - Security features were left out such as JWT implementation, etc.
  - In the BankingSystemController class, a type of logger would be used to log the exceptions in the try catch blocks.
  - Made models as simple as possible, i.e., the user model does not have a DOB, first name, last name, etc.

Source Files: https://github.com/ccarreon/BankingSystem_API

Running the application with Docker:

- Run "docker-compose up" in the root directory of the project.
- The project is also in a repository on the docker hub. Located here:
  - https://hub.docker.com/r/ccarreon7/bankingsystem_api

Running/building the application (Assuming .NET CLI available):

- Use "dotnet build" in the root directory to build the application.
- Use "dotnet run" in the root directory to run the application.

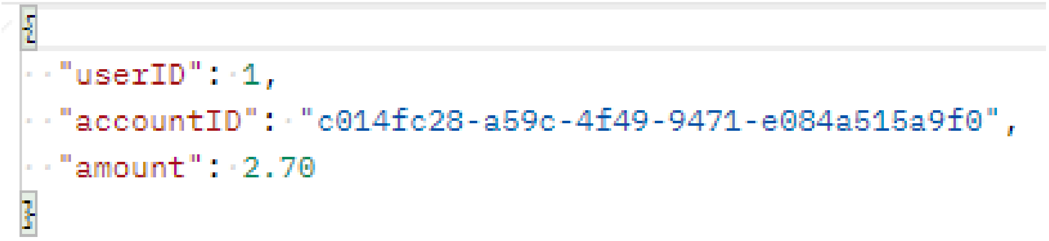
*****PLEASE READ*****

A detailed description of the entire API can be found in the **swagger.json** file in the root directory. "Withdraw" and "Deposit" API calls require a Transaction JSON object in the body (more details can be found in the swagger.json file). There is also a diagram of the API, schemas, and an example Transaction JSON object below:

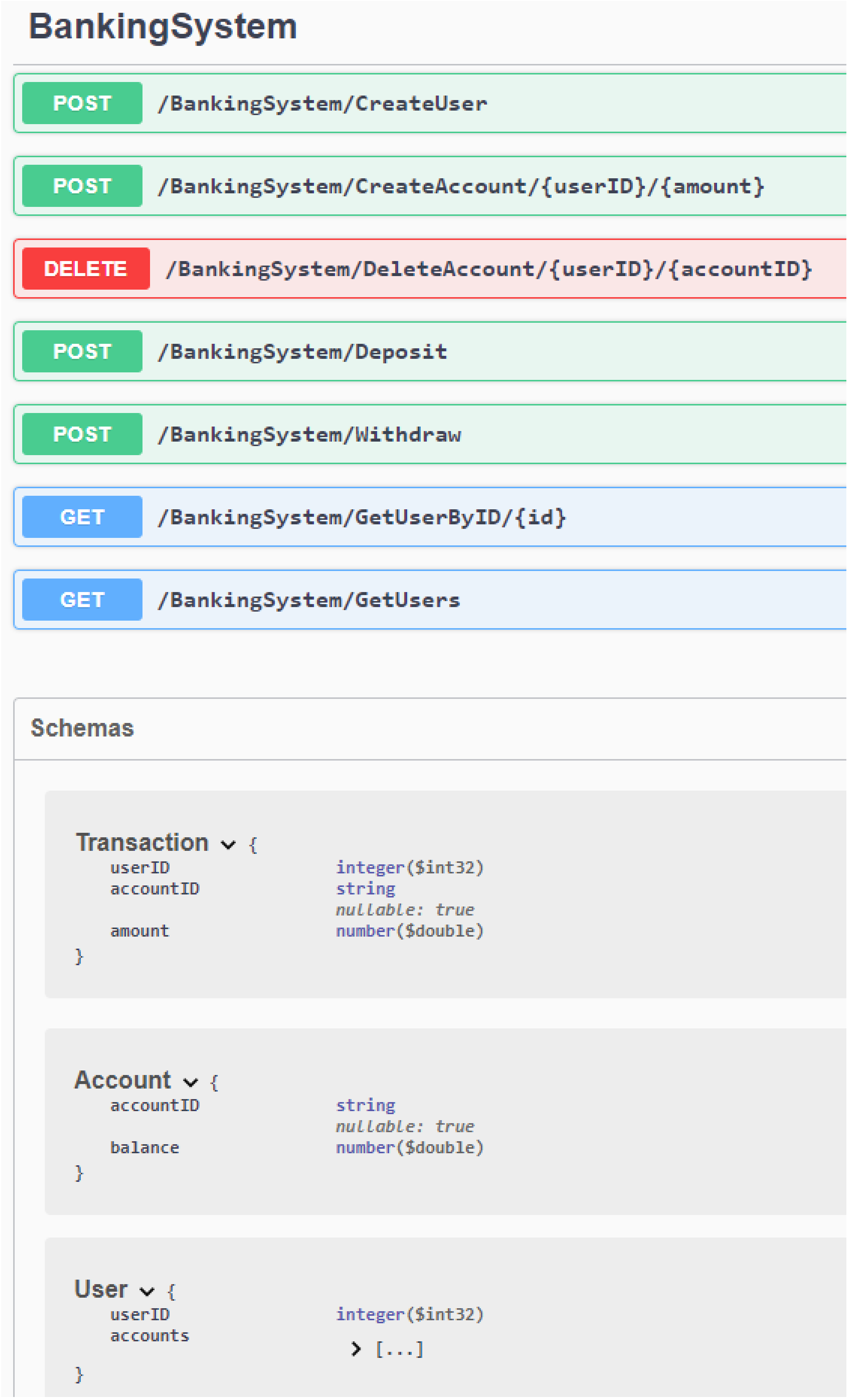
Withdraw Call example:

http://localhost:8080/BankingSystem/Withdraw

Transaction example:



```
{
  "userID": 1,
  "accountID": "c014fc28-a59c-4f49-9471-e084a515a9f0",
  "amount": 2.70
}
```



BankingSystem

POST /BankingSystem/CreateUser

POST /BankingSystem/CreateAccount/{userID}/{amount}

DELETE /BankingSystem/DeleteAccount/{userID}/{accountID}

POST /BankingSystem/Deposit

POST /BankingSystem/Withdraw

GET /BankingSystem/GetUserByID/{id}

GET /BankingSystem/GetUsers

Schemas

```
Transaction ⌄ {
    userID                  integer($int32)
    accountID               string
                            nullable: true
    amount                  number($double)
}

Account ⌄ {
    accountID               string
                            nullable: true
    balance                 number($double)
}

User ⌄ {
    userID                  integer($int32)
    accounts                 > [...]
}
```