Mondu - Coding Challenge

Thanks for applying to Mondu. The goal of this code challenge is to help us identify what you consider production-ready code. You should consider this code ready for final review with your colleague, i.e. this would be the last step before deploying to production.

The Task

We recommend that you read through all of the instructions in this document before starting :)

We would like you to create an API that manipulates the basic operations of a Bank.

Please, implement only the following API endpoints:

- Create an account
- Deposit money on a specific account
- Withdraw money from a specific account
- Get balance of a specific account

The stack

Good engineering is about using the right tool for the right job, and constantly learning about them. Therefore, feel free to mention in your README how much experience you have with the technical stack you choose, we will take note of that when reviewing your challenge.

Feel free to pick one of these languages for the coding challenge:

- Ruby
- Go
- Python

You are also free to use any web framework. If you choose to use a framework that results in boilerplate code in the repository, please detail in your README which code was written by you (as opposed to generated code).

Requirements

- It is up to you to decide what fields need to be stored and what parameters are required for each API endpoint.
- All API responses must be JSON
- Provide Docker environment to build and run the application locally
    - Optionally, deploy to the cloud (AWS or Heroku) and include deployment script
- Provide a README.md file containing:
    - Description
    - Installing (setup) and testing instructions
    - API documentation(endpoints, parameters and responses for each endpoint)

Evaluation

The aspects of your code we will assess include:

- **Architecture**: how clean is the separation of concerns?
- **Code quality**: is the code simple, easy to understand, and maintainable? Is the coding style consistent with the language's guidelines? Is it consistent throughout the codebase?
- **Testing**: how thorough are the automated tests? Will they be difficult to change if the requirements of the application were to change? Are there any unit or any integration tests?
- **Logical errors:** does the code have implementation mistakes? We will assess only written code, not missing functionality.
- **Technical choices**: do choices of libraries, databases, architecture etc. seem appropriate for the chosen application?

Bonus point (those items are optional):

- **Completeness**: does the application do what was asked? If there is anything missing, does the README explain why it is missing?
- **Clarity**: does the README clearly and concisely explain the problem and solution? Are technical trade offs explained?
- **Security**: are there any obvious vulnerabilities?

Thanks again!:)