Technical Assessment Backend Engineer

Problem Description

You are part of the team that explores the Moon by sending remotely controlled vehicles to its surface. Develop an API that translates the commands sent from Earth to instructions that are understood by the robot.

Approach

You should tackle this problem as you would any real world requirement that would be shipped as part of a real product. You should showcase how you work and the way you decompose a problem into smaller pieces.

Requirements

- The robot exposes an HTTP API using FastAPI.
- The robot should keep track of the commands it receives and its current position in a database. You can use PostgreSQL. Design the schema as you wish.
- You are free to choose to use any third-party library
- Use an online Git repository or send us a ZIP file that includes the local .git folder.
    - Code and commit messages should be treated as you would on a real-world task.
- Code quality, architecture and testing is important, especially considering you will have no direct access to the robot once on the Moon.
    - Try to use TDD when developing your solution and provide tests
- Provide a README.md with instructions on how to test the application.

Part I

When the robot touches the Moon, it is initialized with its current coordinates (X, Y) and the direction (NORTH, SOUTH, WEST, EAST) it is facing.
E.g.

```python
START_POSITION = (4, 2)
START_DIRECTION = "WEST"
# bonus points if you read it from the environment
```

Add an endpoint that returns the current coordinates and direction as an JSON object.

The robot receives an input string which contains multiple commands. The robot must interpret the individual commands and execute them. The valid commands are:

- F → Move forward one unit in the direction the robot is facing
- B → Move backwards in the direction the robot is facing
- L → Rotate left by 90 degrees
- R → Rotate right by 90 degrees

Add an endpoint that accepts a command string and returns the new position after execution.

```python
example_command = "FLFFFRFLB"
# Once the full command string is executed the robot sends back its coordinates
```

Part II

We had to abort previous missions due to obstacles damaging the robot.
The known obstacles are stored in the robot memory as a set of coordinates in the following format:

```python
obstacles = {(1,4), (3,5), (7,4)}
```

When the robot enters a coordinate with an obstacle, it stops at the coordinate immediately before. The robot reports its position, direction and that it stopped due to an obstacle.