**Submersible Probe Control System - Requirement Understanding Document**

**Objective**

To develop a RESTful microservice that enables the remote control of a submersible probe on a 2D ocean floor grid. The system interprets movement and rotation commands, avoids obstacles, and tracks visited coordinates.

**Functional Requirements**

**1. Grid Initialization**

* A 2D grid represents the ocean floor.
* Coordinates are identified by integer pairs (x, y).
* The grid has fixed boundaries (e.g., width x height).
* Obstacles can be placed at specific coordinates.

**2. Probe Initialization**

* The probe starts at a given coordinate (x, y) with a facing direction.
* Directions supported: NORTH, SOUTH, EAST, WEST.

**3. Movement Commands**

* Commands are a sequence of characters sent to the API.
* Supported commands:
  + F: Move forward one unit in the current direction.
  + B: Move backward one unit opposite to the current direction.
  + L: Rotate 90° to the left.
  + R: Rotate 90° to the right.

**4. Constraints**

* The probe must not move out of the defined grid boundaries.
* The probe must not move into any obstacle.

**5. Tracking**

* Maintain a list of all coordinates visited by the probe during a session.
* Expose an API endpoint to retrieve this history.

**Non-Functional Requirements**

* Must be implemented as a REST API using Java and Spring Boot.
* Must follow TDD principles with proper unit tests.
* Must integrate with MS SQL to persist probe movement logs.
* Must use Log4j for logging.
* Must include global exception handling.
* Code should be clean, modular, and well-documented.
* Must have a Git commit history that reflects the TDD cycle and feature development steps.

**Assumptions**

* Only one probe is in operation per request/session.
* Obstacles and grid dimensions are initialized once at application start (static for now).
* No concurrency or multi-threaded behavior is required at this stage.