Find ALF's Spaceship Project Documentation

Ridoy Rahman

May 14, 2024

1 Introduction

This document describes the step-by-step process of developing the `findSpaceship` function using Test-Driven Development (TDD). The goal is to create a function that identifies the coordinates of ALF's spaceship on a given map.

2 Project Setup

The project setup involves creating the necessary directories and files, initializing a Git repository, and setting up the test environment using Jasmine.

2.1 Directory Structure

- `src/` - Contains the source code.
- `spec/` - Contains the test specifications.
- `docs/` - Contains the project documentation.

2.2 Initializing Git Repository

```
git init findALF
cd findALF
mkdir src spec docs
touch README.md .gitignore
echo "node_modules/" >> .gitignore
```

3 TODO List for `findALF`

3.1 Functions

- `findSpaceship(map)` - Should return the coordinates of the spaceship or "Spaceship lost forever." if the spaceship is not present.

3.2 Test Cases

1. Should return the correct coordinates when the spaceship is present.
2. Should return "Spaceship lost forever." when the spaceship is not present.
3. Should handle an empty map.
4. Should handle invalid input (e.g., non-string input).

4 Test-Driven Development Steps

In this section, we detail the steps followed in the TDD process for developing the `findSpaceship` function.

4.1 Writing Initial Test Cases

The first step in TDD is to write the initial test cases. Below is the Jasmine test specification for the `findSpaceship` function:

```
// spec/findSpaceship_spec.js
const FindSpaceship = require('../src/findSpaceship');

describe("Find Spaceship", function () {
  var searchmap;

  beforeEach(function () {
    searchmap = new FindSpaceship();
  });

  it("should return correct coordinates when spaceship is present",
      function () {
    const map =
      "..........\n" +
      "..........\n" +
      "..........\n" +
      ".......X..\n" +
      "..........\n" +
      "..........";
    const result = searchmap.searchmap(map);
    expect(result).toEqual([7, 2]);
  });

  it('should return "Spaceship lost forever." when spaceship is not present
      ', function () {
    const map =
      "..........\n" +
      "..........\n" +
      "..........\n" +
      "..........\n" +
      "..........\n" +
      "..........";
    const result = searchmap.searchmap(map);
    expect(result).toEqual("Spaceship lost forever.");
  });

  it('should return "Spaceship lost forever." for an empty map', function
      () {
    const map = "";
    const result = searchmap.searchmap(map);
    expect(result).toEqual("Spaceship lost forever.");
  });

  it('should return "Spaceship lost forever." for invalid input', function
      () {
    const map = 12345;
    const result = searchmap.searchmap(map);
    expect(result).toEqual("Spaceship lost forever.");
  });
});
```

Listing 1: Initial Test Cases for findSpaceship

4.2 Creating the Initial Function

The initial function is created with the structure needed to pass the test cases. Below is the implementation of the `FindSpaceship` function:

```
// src/findSpaceship.js
function FindSpaceship() {}

FindSpaceship.prototype.searchmap = function (map) {
  if (!map || typeof map !== 'string' || map.trim() === "") return "
     Spaceship lost forever.";

  const rows = map.split('\n').reverse();
  for (let y = 0; y < rows.length; y++) {
    const x = rows[y].indexOf('X');
    if (x !== -1) {
      return [x, y];
    }
  }
  return "Spaceship lost forever.";
};

module.exports = FindSpaceship;
```

Listing 2: Initial Implementation of FindSpaceship

5 Conclusion

By following the TDD approach, we ensure that our code is reliable and meets the specified requirements. The process starts with writing tests, followed by implementing the minimal code necessary to pass the tests, and then refining the code.