LAB 2

Title: Introduction to Test-Driven Development (TDD) and Behavior-Driven Development (BDD)

Objective: To understand and apply the principles of **Test-Driven Development (TDD)** and **Behavior-Driven Development (BDD)** in a **Node.js** environment using **Jest**.

Theory: **Test-Driven Development (TDD)** is a software development technique where developers write tests **before writing the actual code**. It follows the **Red-Green-Refactor** cycle:

- **Red:** Write a test that fails (because the code isn't written yet).
- **Green:** Write the minimum code needed to make the test pass.
- **Refactor:** Improve the code without changing its behavior, while keeping all tests passing.

This approach ensures that each part of the application is tested from the beginning, resulting in fewer bugs and better design.

Behavior-Driven Development (BDD) is a natural extension of TDD. It emphasizes:

- Describing **expected behaviors** instead of low-level code details.
- Using **human-readable language** for tests to encourage collaboration between developers, testers, and non-technical stakeholders.

Popular BDD frameworks include **Jest** (for JavaScript/Node.js), **Cucumber**, and **Mocha** with **Chai**.

Code: Let's say we want to test a simple function that adds two numbers:

1. Create `math.js`

```
function add(a, b){
    return a + b;
}
module.exports = add;
```

2. Create **math.test.js**

```
const add = require('./math');
test('adds 2 + 3 to equal 5', () => {
  expect(add(2, 3)).toBe(5);
});
```

3. Run the test

```
npm test
```

Conclusion: In this lab, we explored writing tests before implementing code (TDD) and focusing on behavior-driven tests (BDD). Using **Jest**, we created unit tests that confirmed our functions worked as expected. This development method improves code quality, helps identify bugs early, and encourages better collaboration. Practicing TDD and BDD leads to more maintainable and reliable software.