LAB 2:

Test-Driven Development (TDD) & Behavior-Driven Development (BDD)

OBJECTIVE:

The objective of this lab was to understand the concepts of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) using Node.js.

TOOLS AND TECHNOLOGIES:

Node.js -JavaScript runtime for development

Mocha + Chai - For TDD (unit testing)

Jasmine - For BDD (behavior testing)

VS Code / Terminal -Development and execution environment

THEORY:

Test-Driven Developmen(TDD) is a practice where a test is conducted at first(it fails).Then, write a minimum code to make it pass. Refactor the code while keeping the tests green.

Behaviour driven Development(BDD) is similar to TDD but emphasizes behavior and readability.It describes scenarios from the user's perspective using natural language and tests them with tools like jasmine or cucumber.

OBSERVATION:

A. TDD using Mocha & Chai

1. Initialize Node project and install packages:

npm init -y

```
npm install mocha chai --save-dev
```

2. calculator.js (implementation file):

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

3. test/calculator.test.js (test file):

```
const { expect } = require('chai');
const { add } = require('../calculator');
describe('Calculator - TDD', () => {
it('should return 5 for 2 + 3', () => {
expect(add(2, 3)).to.equal(5);
});
});
```

4. Run the test:

```
npx mocha
```

B. BDD using Jasmine

1. Install Jasmine and initialize:

```
npm install --save-dev jasmine
npx jasmine init
```

2. calculator.js (same as before):

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

3. spec/calculatorSpec.js:

```
const add = require('../calculator');
describe('Calculator - BDD', () => {
it('adds 2 and 3 to return 5', () => {
expect(add(2, 3)).toBe(5);
});
});
```

4. Run Jasmine tests:

```
npx jasmine
```

CONCLUSION:

It taught me how to use TDD to build code incrementally and how BDD is used to write tests that fulfils users expectation.

Name: Dipesh Thapa

Roll No: 21