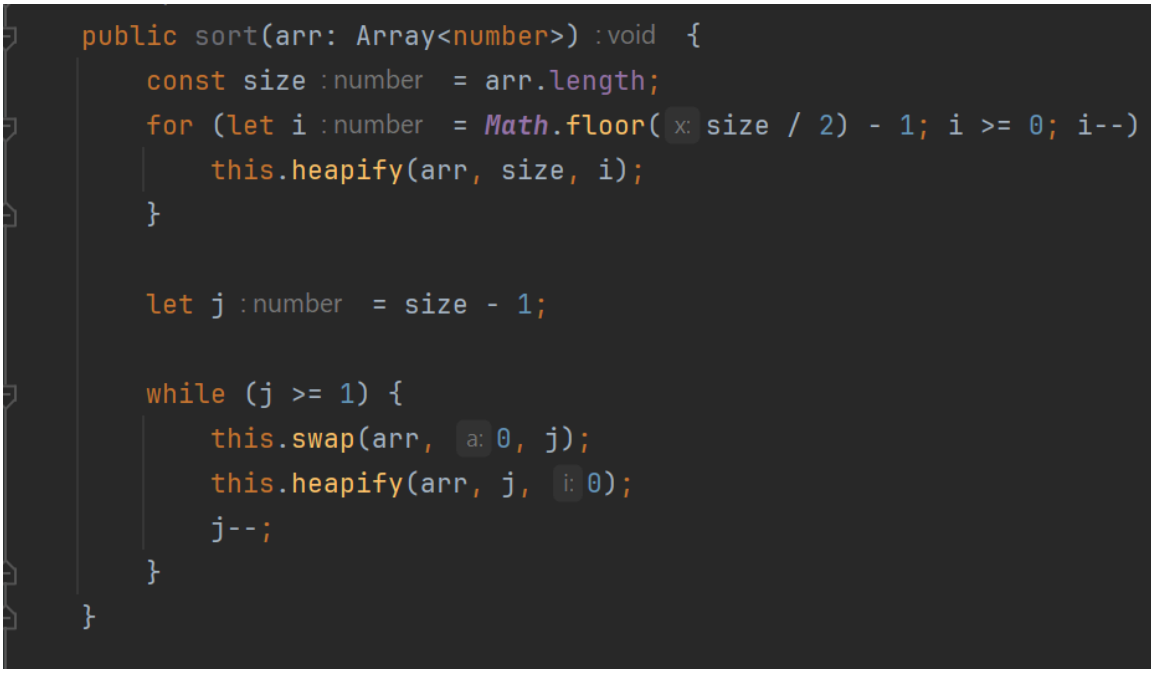
Timothy Mendez Feliciano
CSE 565: Software Verification, Validation and Testing
Dr. Injung Kim
September 1, 2024

Software Unit Testing Framework Project - Project Report

I. Task 1 - Create Heapsort Algorithm
   A. I created a class that contains 3 functions (Sort, Heapify, Swap)

1. 

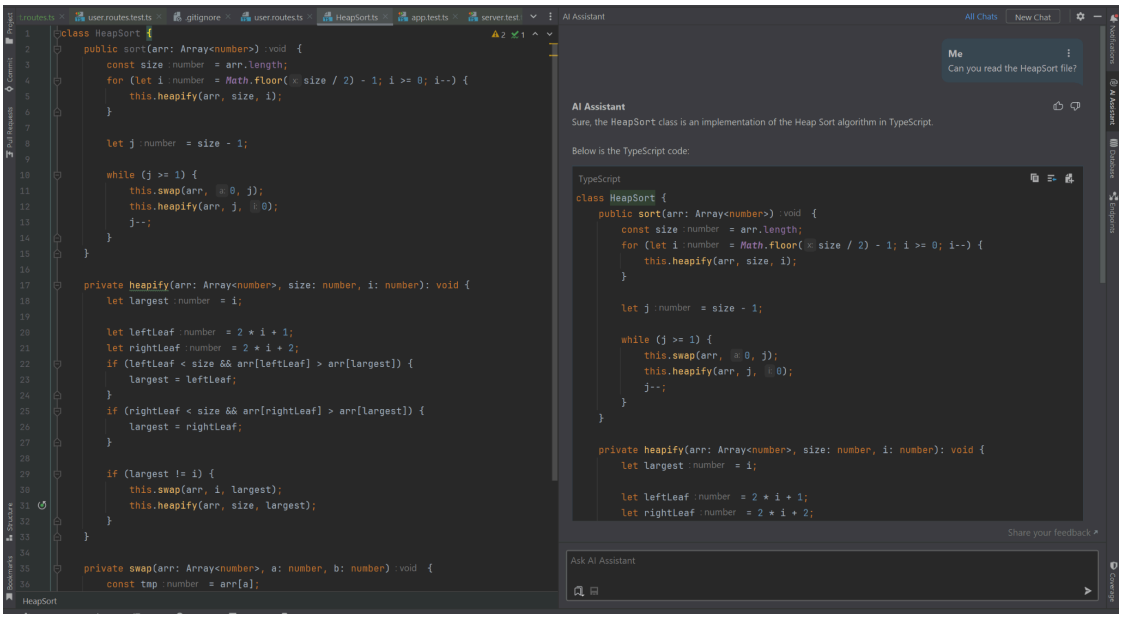
```
public sort(arr: Array<number>) :void  {
    const size :number  = arr.length;
    for (let i :number  = Math.floor( x: size / 2) - 1; i >= 0; i--)
        this.heapify(arr, size, i);
    }

    let j :number  = size - 1;

    while (j >= 1) {
        this.swap(arr,  a: 0, j);
        this.heapify(arr, j,  i: 0);
        j--;
    }
}
```

2.
```
private heapify(arr: Array<number>, size: number, i: number): v
    let largest : number  = i;

    let leftLeaf : number  = 2 * i + 1;
    let rightLeaf : number  = 2 * i + 2;
    if (leftLeaf < size && arr[leftLeaf] > arr[largest]) {
        largest = leftLeaf;
    }
    if (rightLeaf < size && arr[rightLeaf] > arr[largest]) {
        largest = rightLeaf;
    }

    if (largest != i) {
        this.swap(arr, i, largest);
        this.heapify(arr, size, largest);
    }
}
```
3.
```
private swap(arr: Array<number>, a: number, b: number) : voi
    const tmp : number  = arr[a];
    arr[a] = arr[b];
    arr[b] = tmp;
}
```

II. Task 2

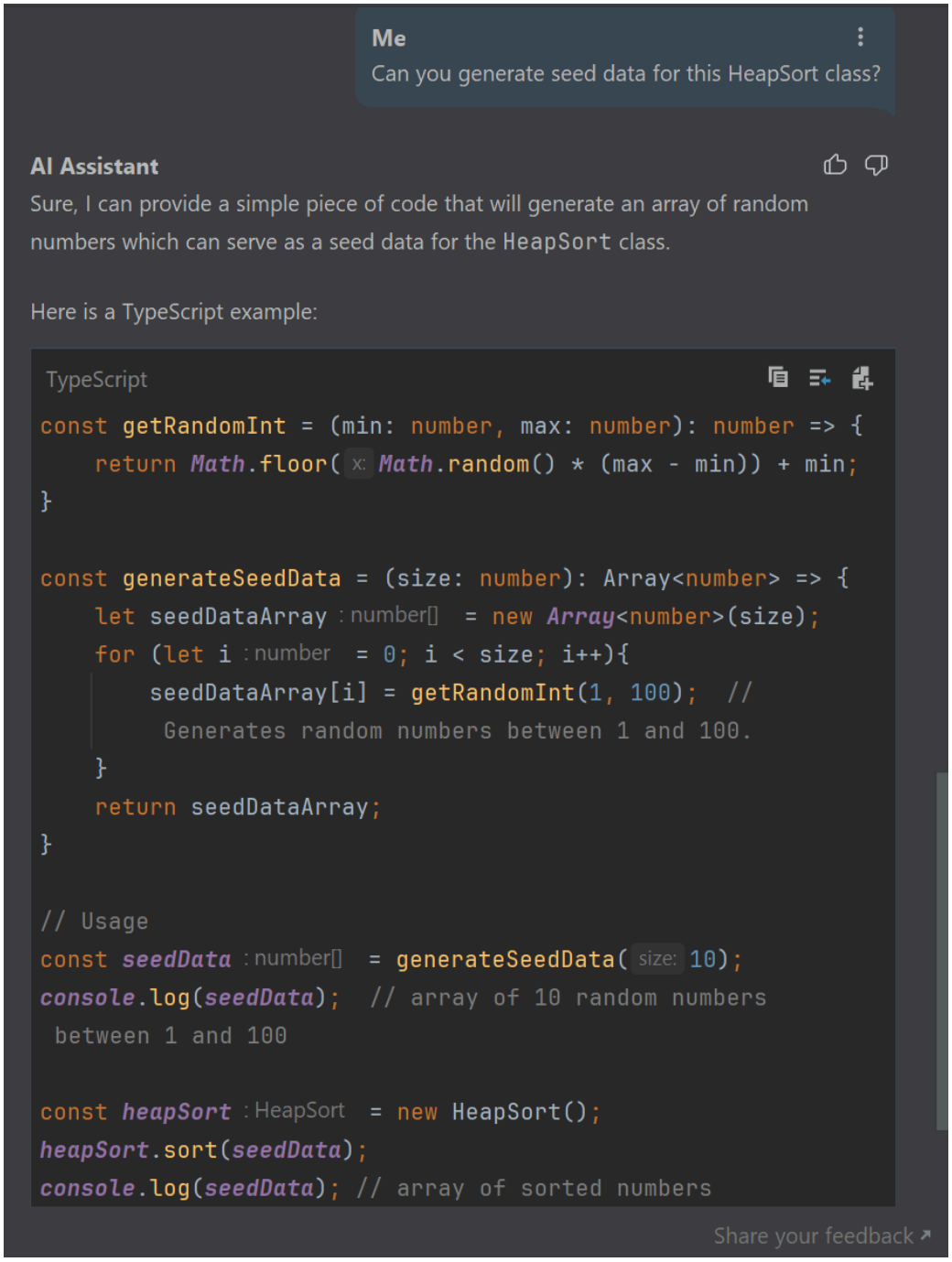
A. Research and Identify a Unit Testing Framework to test code created.

1. Given that I created this project with Node.js and TypeScript, as an Express API.
2. Given this configuration I have decided to utilize Jest and Supertest.
    a) Jest Documentation:
        (1) https://archive.jestjs.io/docs/en/22.x/getting-started.html
    b) Supertest Documentation:
        (1) https://ladjs.github.io/superagent/
3. Jest allows us to describe test stories, create small unit tests, and utilize expect statements to verify the result of our functions.
4. Supertest mocks the calls of the api. Providing us access to lower level HTTP api such as get and post requests.

B. Research and use a generative AI tool to create unit-level test cases using a unit-testing framework.
    1. After research I have decided to use Jetbrains AI given that it can supposedly exist within my repository, and is context aware of the files in my project. It is also built on top of OpenAI and internal models.
    2. To prove its context aware abilities. I have asked it to read the HeapSort file created and it replied correctly.
    3. 

        a) The right hand side is my interaction with JetbrainsAI.
    4. Before creating the Unit Tests I have asked the JetbrainsAI to create seed data for testing the heapsort.

a)



5. I have modified the file above to facilitate automated testing.

6.

```
import {writeFile} from "fs";
import path from "path";

const getRandomInt = (min: number, max: number): number => {
    return Math.floor( x: Math.random() * (max - min)) + min;
}

const generateSeedData = (size: number): Array<number> => {
    let seedDataArray : number[]  = new Array<number>(size);
    for (let i : number  = 0; i < size; i++) {
        seedDataArray[i] = getRandomInt(1, 100);  // Generates random
    }
    return seedDataArray;
}

// Usage
const seedData : number[]  = generateSeedData( size: 10);
const seedPath : string  = path.join(__dirname,
    '..',
    'seed',
    'seedData.ts'
)

writeFile(
    seedPath,
    data: `export const seedData: number[] = [${seedData}];`,
    callback: err : NodeJS.ErrnoException | null  => {
        console.error(err?.message)
    }
)
```

    a) Now the seed can be generated with every run and written to a file accessible by the test.
7. JetbrainsAI created a Unit Test from my prompt.

8.
```typescript
import app from "../../src/app";
import request from "supertest";
import { seedData } from "../../seed/seedData";
import HeapSort from "../../src/HeapSort";

describe( name: "HeapSort routes", fn: () : void => {
    test( name: "HeapSort route should correctly sort seedData", fn: async () : Promise<void> => {
        const originalSeedData : any[] = [...seedData]; // create a copy of seedData for compari
        const sortedSeedData : any[] = [...seedData].sort( compareFn: (a, b) => a - b);

        const res = await request(app).post("/heapSort").send({ data: seedData });

        expect(res.body).toEqual(sortedSeedData); // expect the sorted result
        expect(res.body).not.toEqual(originalSeedData); // expect the result to not equal the
         seedData
    });
});
```

C. Identify and Report the prompt used to generate the test cases.
   1. Can you read the HeapSort file?
      a) Was excellent at reading the file.
   2. Can you generate seed data for this HeapSort class?
      a) Was excellent at generating the seed data for HeapSort algorithm. However, it did immediately use the algorithm. I had to modify it to save the data to a specific file.
   3. Can you take a look at the user.routes.test.ts file?
      a) JetbrainsAI read the files excellently.
   4. Using Jest, and Supertest, can you create a Unit Test for the heapSort route? Please read from the seedData file to import the seedData const and use for the unit test. In the test, expect the result from HeapSort.sort() to not equal seedData. And expect it to be sorted correctly.
      a) In general, the test looks mostly good.
      b) It correctly summons seedData. And correctly creates a deep copy of the originalSeedData and utilizes it when comparing the HeapSort result in the expect file.
      c) It also correctly verifies that the result and original value are different.
      d) It correctly sends the seedData to the correct route in the express API. Despite the route not having been created at the moment. So, it correctly assumed the route.
      e) However, Jetbrains did add the HeapSort as in import when it is not needed. And, it also imported the file from the incorrect route. It seemed to utilize the root of the project as the base of the file search. And did not consider the file to be used inside the __ tests __ folder.

III. Task 3 - Execute test cases generated by the AI tools and provide output.

```
Terminal:  Local ×  +
PS C:\Users\timot\Webstorm\software-unit-testing-project> npm run test

> software-unit-testing-project@1.0.0 test
> ts-node ./src/generateSeedData.ts && jest --coverage

undefined
 PASS  __tests__/server.test.ts
 FAIL  __tests__/routes/heapSort.routes.test.ts
  ● Test suite failed to run

    __tests__/routes/heapSort.routes.test.ts:4:22 - error TS2306: File 'C:/Users/timot/Webstorm/software-unit-testing-project/src/HeapSort.ts' is not a module.

    4 import HeapSort from "../../src/HeapSort";
                           ~~~~~~~~~~~~~~~~~~~~

 PASS  __tests__/routes/user.routes.test.ts
 PASS  __tests__/app.test.ts
---------------------|---------|----------|---------|---------|-------------------
File                 | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
---------------------|---------|----------|---------|---------|-------------------
All files            |   79.16 |      100 |   66.66 |   79.16 |
 src                 |     100 |      100 |     100 |     100 |
  app.ts             |     100 |      100 |     100 |     100 |
 src/routes          |   66.66 |      100 |      50 |   66.66 |
  heapSort.routes.ts |   44.44 |      100 |       0 |   44.44 | 7-13
  user.routes.ts     |     100 |      100 |     100 |     100 |
---------------------|---------|----------|---------|---------|-------------------

Test Suites: 1 failed, 3 passed, 4 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        1.892 s, estimated 2 s
Ran all test suites.
PS C:\Users\timot\Webstorm\software-unit-testing-project>
```

A.

B. As you can see, the file produced an error by not being able to find the correct import during test. Only 44% of the lines were tested.

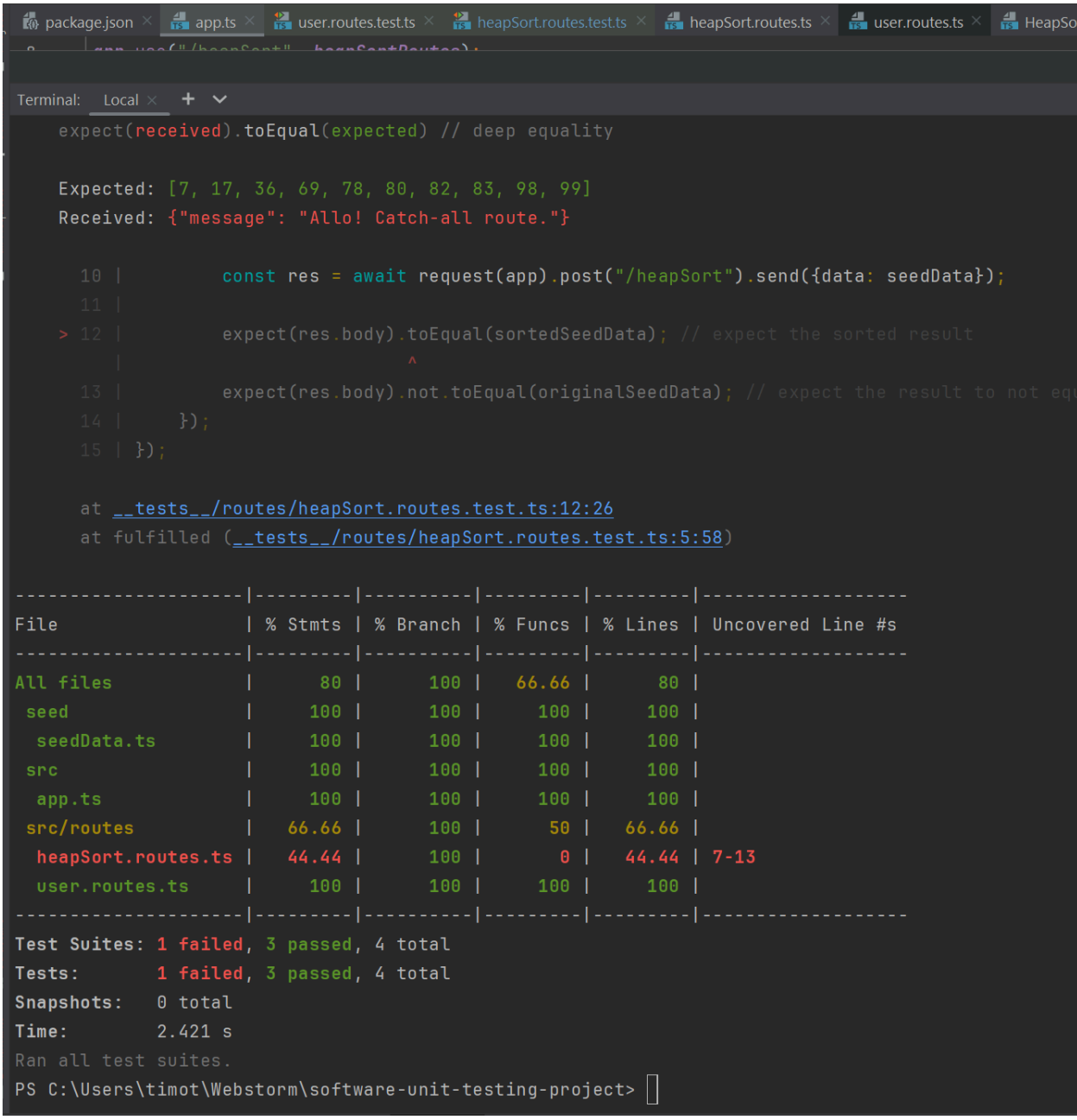
C. As a result, the testing was not able to be completed.

IV. Task 4

A. The test case for heapSort.routes.ts is not valid. Since the api call itself could not be tested.

B. The test case developed by JetbrainsAI was not correct enough and not sufficient for the test.

C. We can first, update them by removing the incorrect import.



D.

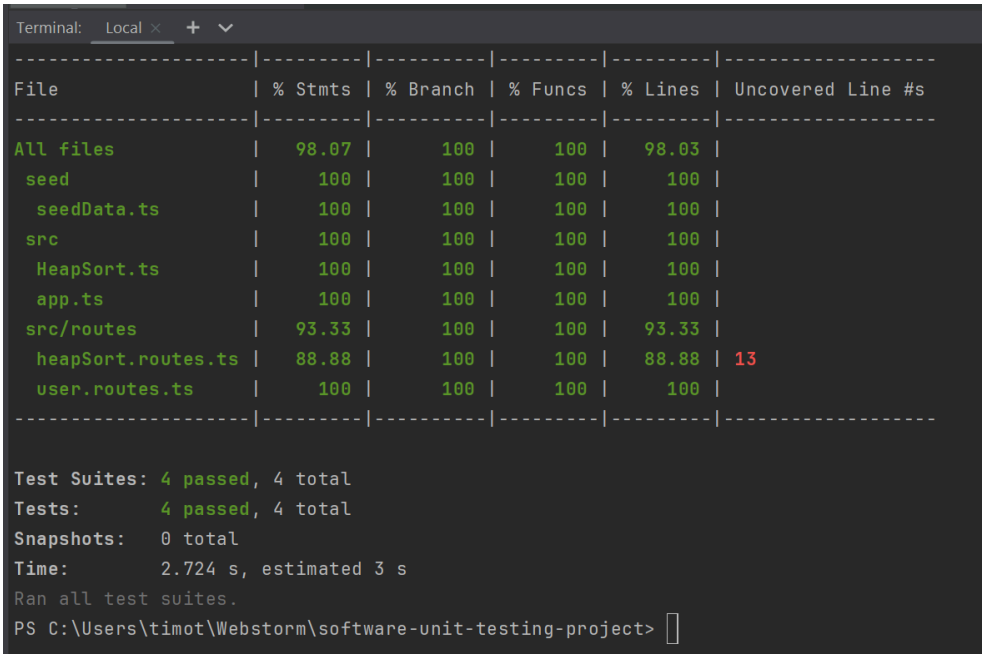
E. We still obtained an incomplete test case. The first issue here Jetbrains AI made the assumption that the route was a *post* route. Given that JetbrainsAI is supposed to be context aware of the code within the project it should've realized the route was set as *get*.

F. However, the correct way of preparing this call was to create a post route. So, I modified the route.

G. The second issue here was that *.send({data:seedData})* was an incorrect assumption from the JetbrainsAI.

H. By, simply sending the seedData in the .send function as *.send(seedData)*, we get a successful test ran.

I.



J. From the output, the only line missing to be tested was line 13. This line is a console.error function from the catch segment of a try-catch.

V. Task 5

A. Make an assessment of the generative AI.

1. Effectiveness:
    a) Jetbrains AI is a remarkable tool. It is great at finding files within the projects, and it is great at creating seed files.
    b) Jetbrains AI is also really good a creating tests cases based on previous examples within the project.
    c) The test case created by Jetbrains AI was able to follow instructions from the prompt. Clearly creating 2 different expect statements based on what I wrote.
    d) Jetbrains AI was also very good at using the libraries such as supertest and jest.
2. Final Assessment:
    a) I believe Jetbrains AI to be excellent at speeding up development of the test cases.
    b) It quickly created a script to generate seed data for the HeapSort class to facilitate testing.
    c) It quickly created a test that was largely correct. Only two changes needed to be made to improve the test.
    d) Jetbrains AI made assumptions about the code structure and file placements that needed to be amended.
    e) Jetbrains AI still requires the user to have knowledge of the API, project structure, and libraries used in case the AI makes incorrect assumptions.