Group Assignment: Testing Assignment

By

Mr. WARIS DAMKHAM 6388014

Miss PATTANAN KORKIATTRAKOOL 6388022

Submitted to

Lect. Dr. Wudhichart Sawangphol

A Report Submitted in Partial Fulfillment of

the Requirements for

ITCS371 Introduction to Software Engineering

Faculty of Information and Communication Technology

Mahidol University

2021

Table of Contents

Task 1

Test Case ID: #1

Test Title: Checking that no student's age is equal to 60 in the database

Description: Check the database to ensure that no student has reached the age of 60.

Test type: Unit Testing

Test priority (Low/Med/High): Low

Test Designed by: Waris Damkham and Pattanan Korkiattrakool

Test Designed date: 15/11/2022

Test Executed by: Waris Damkham and Pattanan Korkiattrakool

Test Executed date: 15/11/2022

Pre-condition: The user needs to connect to "http://localhost:3000".

Step	Test Steps	Test Data	Expected result	Actual result	Status (Pass/Fail)	Note
1	Navigate to Database	-	No student has reached the age of 60.	No student has reached the age of 60.	Pass	-

Code to test

```
test("Test Case#1: Checking that no student's age is equal to 60 in
the database",async() =>{
        const posts = await
axios.get("http://localhost:3000/students").then((res)=>res.data);
        const msg1 = posts.data[0].STU_AGE;
        const msg2 = posts.data[1].STU_AGE;
        const msg3 = posts.data[2].STU_AGE;
        const msg4 = posts.data[3].STU_AGE;
        const msg5 = posts.data[4].STU_AGE;
        var ans = [msg1,msg2,msg3,msg4,msg5];
        expect(ans).not.toEqual("60");
    });
```

Testing Output

```
 Unit testing
   √ Test Case#1: Checking that no student's age is equal to 60 in the database (1 ms)
   √ Test Case#2: Checking that the database table is not empty
```

Test Case ID: #2

Test Title: Checking that the database table is not empty

Description: Test that the student's database is not empty.

Test type: Unit Testing

Test priority (Low/Med/High): Low

Test Designed by: Waris Damkham and Pattanan Korkiattrakool

Test Designed date: 15/11/2022

Test Executed by: Waris Damkham and Pattanan Korkiattrakool

Test Executed date: 15/11/2022

Pre-condition: The user needs to connect to "http://localhost:3000".

Step	Test Steps	Test Data	Expected result	Actual result	Status (Pass/Fail)	Note
1	Navigate to Database	-	Database is not empty	Database is not empty	Pass	-

Code to test

```
test("Test Case#2: Checking that the database table is not
empty",async() =>{
        const posts = await
axios.get("http://localhost:3000/students").then((res)=>res.data);
        expect(posts.data).not.toBeNull();
    });
```

Testing Output

```
Unit testing
  √ Test Case#1: Checking that no student's age is equal to 60 in the database (1 ms)
  √ Test Case#2: Checking that the database table is not empty
```

Task 2

Test Case ID: #3

Test Title: Get all students' information in the database via /students

Description: Test the StudentService API to retrieve all students' information in the database.

Test type: Integration Testing

Test priority (Low/Med/High): Med

Test Designed by: Waris Damkham and Pattanan Korkiattrakool

Test Designed date: 15/11/2022

Test Executed by: Waris Damkham and Pattanan Korkiattrakool

Test Executed date: 15/11/2022

Pre-condition: -

Step	Test Steps	Test Data	Expected result	Actual result	Status (Pass/Fail)	Note
1	Navigate to Student Information Page	-	The user will see all	The user will see all	Pass	-
2	Click on Select all button		student's information displayed.	student's information displayed.		

Code to test

```
test("Test Case#3: Get all students' information in the database via
/students", async () =>{
    const res = await request(app).get("/students");
    expect(res.body.data).toEqual([
    {
        STU_ID: 1,
        STU_FNAME: "Andrew",
        STU_LNAME: "Black",
        STU_AGE: 25,
    },
    {
        STU_ID: 2,
        STU_FNAME: "Alexandra",
        STU_LNAME: "Brown",
        STU_AGE: 25,
    },
    {
        STU_ID: 3,
        STU_FNAME: "Amanda",
        STU_LNAME: "Davidson",
        STU_AGE: 25,
    },
    {
        STU_ID: 4,
        STU_FNAME: "Benjamin",
        STU_LNAME: "Duncan",
        STU_AGE: 25,
    },
    {
        STU_ID: 5,
        STU_FNAME: "Christopher",
        STU_LNAME: "Ellison",
        STU_AGE: 25,
    },
    ]);
});
```

Testing output

```
Integration testing
  √ Test Case#3: Get all students' information in the database via /students (10 ms)
  √ Test Case#4: Get the information of the last student in the database via /student/:id (7 ms)
```

Test Case ID: #4

Test Title: Get the information of the last student in the database via /student/:id

Description: Test the StudentService API to retrieve the last students' information in the database.

Test type: Integration Testing

Test priority (Low/Med/High): Med

Test Designed by: Waris Damkham and Pattanan Korkiattrakool

Test Designed date: 15/11/2022

Test Executed by: Waris Damkham and Pattanan Korkiattrakool

Test Executed date: 15/11/2022

Pre-condition: A valid student ID must be known by the user.

Step	**Test Steps**	**Test Data**	**Expected result**	**Actual result**	**Status (Pass/Fail)**	**Note**
1	Navigate to Student Information Page	Student ID	The user will see the last	The user will see the last	Pass	-
2	Fill in the student ID box with student ID number 5.		student's information displayed.	student's information displayed.		
3	Click on Select button					

Code to test

```
test("Test Case#4: Get the information of the last student in the
database via /student/:id", async () =>{
    const req = await request(app).get('/student/5');
    expect(req.body.data).toEqual(
    {
        STU_ID: 5,
        STU_FNAME: "Christopher",
        STU_LNAME: "Ellison",
        STU_AGE: 25,
    });
});
```

Testing output

```
Integration testing
  √ Test Case#3: Get all students' information in the database via /students (10 ms)
  √ Test Case#4: Get the information of the last student in the database via /student/:id (7 ms)
```

Task 3

Test Case ID: #5

Test Title: Test the web service by inserting a new student and retrieving his information from the database.

Description: Test the StudentService API with a web service by inserting a new student and retrieving information in the database.

Test type: System Testing

Test priority (Low/Med/High): High

Test Designed by: Waris Damkham and Pattanan Korkiattrakool

Test Designed date: 15/11/2022

Test Executed by: Waris Damkham and Pattanan Korkiattrakool

Test Executed date: 15/11/2022

Pre-condition: The user enters a new student ID, first name, last name, and age into the student information page, and the user needs to connect to "http://localhost:3000".

Step	**Test Steps**	**Test Data**	**Expected result**	**Actual result**	**Status (Pass/Fail)**	**Note**
1	Navigate to Student Information Page	New student ID, first name,	The user will see the new	The user will see the new	Pass	-
2	Click on Student ID field	last name, and age	student's information displayed.	student's information displayed.		
3	Type”6388014”					

4	Click on Student Firstname field					
5	Type”Waris”					
6	Click on Student Lastname field					
7	Type”Damkham”					
8	Click on Student Age field					
9	Type”21”					
10	Click on Insert button					
11	Click on Student ID field					
12	Type”6388014”					
13	Click on Select button					

Code to test

```
test("Test Case#5: Test the web service by inserting a new student and
retrieving his information from the database.", async () =>{
    const browser = await puppeteer.launch({
        headless: false,
        slowMo: 80,
        args: ["--window-size=1920,1080"],
        executablePath:
      "C:/Program Files (x86)/Microsoft/Edge/Application/msedge.exe",
    });
    const page = await browser.newPage();
    await page.goto("http://localhost:3100/");
    await page.click("input#STU_ID");
    await page.type("input#STU_ID", "6388014");
    await page.click("input#STU_FNAME");
    await page.type("input#STU_FNAME", "Waris");
    await page.click("input#STU_LNAME");
    await page.type("input#STU_LNAME", "Damkham");
    await page.click("input#STU_AGE");
    await page.type("input#STU_AGE", "21");
    page.on("dialog", async dialog => {
        await dialog.accept();
    })
    await page.click("input#insert");
    await page.click("input#STU_ID");
    await page.type("input#STU_ID", "6388014");
    await page.click("input#select");
    const studentObject = await page.evaluate(() => {
    return {
        firstName: document.getElementById("STU_FNAME").value,
        lastName: document.getElementById("STU_LNAME").value,
        age: document.getElementById("STU_AGE").value,
    };
    });
    expect(studentObject).toEqual({
        firstName: "Waris",
        lastName: "Damkham",
        age: "21",
    });
},20000)
```

Testing output

```
System testing
  √ Test Case#5: Test the web service by inserting a new student and retrieving his information from the database. (624 ms)
```