Software Test Plan:

CSUN Dashboard

Prepared by:

Christian Jarmon

12th November 2022

1.0 INTRODUCTION

The product being tested, named CSUN Dashboard, is a set of search tools made for students and advisors to allow the exploration of class catalogs, professors, and majors offered by the university.

2.0 OBJECTIVES AND TASKS

2.1 Objectives

To identify deviations of functionality from its specified requirements for the functional portions of the frontend made in ReactJS and deviations of data performance and accuracy from API.

2.2 Tasks

3.0 SCOPE

General

Verify that all frontend components function as required and that API returns data expected and required for such components to function properly.

- Planner
    - Verify correct API endpoints are used.
    - Verify responses for every possible user action
        - If a user picks a class, it responds by listing it.
        - If a user removes a class, it delists it.
        - If a user picks a conflicting class, it lets the user know and prevents the addition of said conflicting class/
        - Shows the cost of selected classes after every addition or removal
- Professor Search
    - Verify correct API endpoints are used
    - Verify correct data is returned
    - Ratings
        - Verify each rating hyperlink leads to the correct rating page.
        - Verify the post endpoint is functional
- Major Page
    - Verify all majors offered by CSUN are present on page
        - Verify that each hyperlink leads to the correct page
- Ratings Page
    - Verify that the rating process works
        - User inputs rating information
        - Then when the user hits post, that rating shows up on the page of the professor was made for

4.0 TESTING STRATEGY

Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach which will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools which are used to test the designated groups of features.

The approach should be described in sufficient detail to permit the identification of the major testing tasks and the estimation of the time required to do each one.

- For the front end, there exist two layers of responsiveness, the proper function calls for every user action and the visual response that the user sees as a result of those function calls. Therefore, what has to be tested is that the proper functions are being called at all times and that they return what they are supposed to given certain inputs and situations.
    - The Cypress E2E testing framework is the best candidate for this.
        - UI/UX checks
        - Can stub and intercept network requests
            - Mainly to verify all backend API GET and POST requests are being called correctly from user's POV

- API
    - Manual granularity checks must occur to identify what the API is supposed to return according to the specification
    - Automated scripts will be written to compare the two string outputs, one manual, and one from the API.
    - Will be written in Python using libraries including but not limited to, ***urllib3***, ***json***, ***re*** (the regex library), etc.

4.1 Unit Testing

Definition:

- Due to the uniformity of the API and the UI/UX components, a high-granularity

testing structure is not required but rather a general approach. For the API, every endpoint just has to be confirmed against one set of parameters, which would then confirm the functionality of all other parameters, such as the Subjects for the Catalog or Emails for the Professor endpoints. For the front end, most interactions follow a GET request and then a Display procedure for what was returned.

Participants:

- Nima Shafie will write the scripts to verify the API
- Christian Jarmon will use Javascript and the Cypress testing framework to define and automate the UI/UX tests.

Methodology:

- API
    - Manually record the intended results of each endpoint and then compare it against a simulated GET request
    - Use Python and its libraries ***urllib3***, ***json***, ***re*** (the regex library), etc. as mentioned in 4.0 introduction
- UI/UX
    - Using the Cypress E2E testing framework to verify UI functionality
        - Verify buttons and hyperlinks function and lead to their correct destination respectively
        - Stub and intercept network requests to verify proper requests and responses are being sent and received, respectively.

4.2 System and Integration

Testing Definition:

List what is your understanding of System and Integration Testing for your project.

Exiting the development environment, a build process with be performed with NPM's build tool for ReactJS verifying the results of the separate unit tests in the build

Participants:

Who will be conducting System and Integration Testing on your project? List the individuals that will be responsible for this activity.

- David Huezo
- Micheal Balian
- Nima Shafie

Methodology:

Describe how System & Integration testing will be conducted. Who will write the test scripts for the unit testing, what would be the sequence of events of System & Integration Testing, and how will the testing activity take place?

Using the results from the UNIT tests, a build process will take place using NPM's Build tool. The same interactions and the same GET requests will occur and will be verified using the results from the UNIT tests.

4.3 Performance and Stress Testing

IF APPLICABLE, IF NOT PUT N/A

Definition:

List what is your understanding of Stress Testing for your project.

- Flood the server with a bunch of API GET requests to test response time and identify any bottlenecks

Participants:

Who will be conducting Stress Testing on your project? List the individuals that will be responsible for this activity.

Christian Jarmon

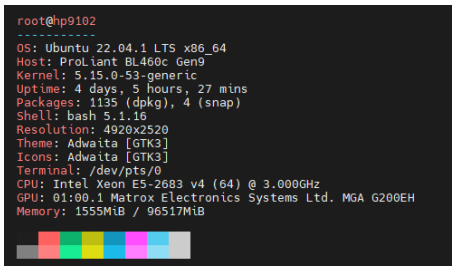
Methodology:

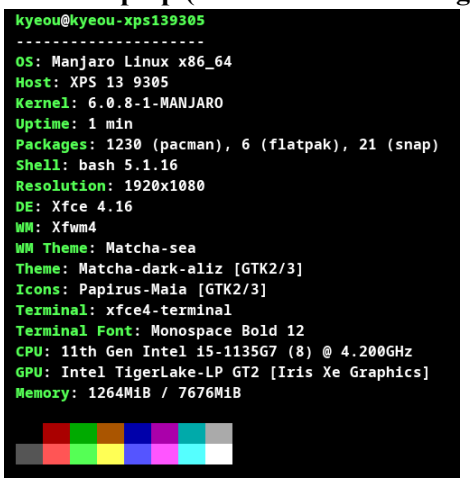
Describe how Performance & Stress testing will be conducted. Who will write the test scripts for the testing, what would be the sequence of events of Performance & Stress Testing, and how will the testing activity take place?

- A multithreaded Python script will be written to perform a GET Request on all the endpoints some set amount of times.
    - Quantity of requests will be as follows
        - 10
        - 50
        - 100
        - 500
        - 1000
- After each test run, the time it took for each request to complete client-side will be analyzed as such.
    - These times will be averaged and analyzed as such. (Example Output)
        - Test Start Time: 2022-12-03 09:13:31.654382+00:00
        - ------
        - Processes finished 500
        - Average Time: 1.207428 secs
        - Best: 0.085 secs
        - Worst: 4.263 secs
        - ------
    - The server-side logs will be recording in the same way
        - They will be separate to measure for certain bottlenecks should they happen either in Database queries, parsing, frontend parsing, or connection bottlenecks.

5.0 ENVIRONMENT REQUIREMENTS (TOOLS)

List test tools and environment in which those tools will be used (i.e. operating system, PC specs.)

- **All tests will be conducted in a linux environment with automated bash scripts running on a centralized server with only one outside machine for control metrics**
- **The tests will be conducted with the use of 2 machines.**
  - **A centralized Server**
  - 

  - **A linux laptop (to simulate the average user)**
  - 

6.0 TEST CASE DEFINITIONS

- HOME Page

Test Case ID	Test Case Steps/Expected Results
TC-1 - Can See Homepage	**Action** / **Expected Result** Visit Home Page → • Can see text, `"The CSUN course planner you've been looking for"` Click `"Get Started"` button → • URL should be `"https://csundash.kyeou.xyz/planner"`

- PLANNER Page

<table>
<tr><th>Test Case ID</th><th>Test Case Steps/Expected Results</th></tr>
<tr><td>TC-2 - Can Select Subject and Term</td><td>
<table>
<tr><th>Action</th><th>Expected Result</th></tr>
<tr><td>Click <code>"Term"</code> dropdown</td><td>• Can see list of available semesters</td></tr>
<tr><td>Click <code>"Subject"</code> dropdown</td><td>• Can see list of subject codes</td></tr>
<tr><td>Click <code>ALL</code> subjects</td><td>• For every subject, a list of available sections offered show up</td></tr>
</table>
</td></tr>
<tr><td>TC-3 - Sections Show Up When Term Selected</td><td>
<table>
<tr><th>Action</th><th>Expected Result</th></tr>
<tr><td>Click <code>"Term"</code> dropdown</td><td>• Can see list of available semesters</td></tr>
<tr><td>Click <code>"Subject"</code> dropdown</td><td>• Can see list of subject codes</td></tr>
<tr><td>Click <code>ALL</code> subjects</td><td>• For every subject, a list of available sections offered should show up
• Intercepted network requests to /classes and /schedule should match response bodies requested by Cypress</td></tr>
</table>
</td></tr>
<tr><td>TC-4 - Can Add Classes</td><td>
<table>
<tr><th>Action</th><th>Expected Result</th></tr>
<tr><td>Click <code>"Term"</code> dropdown</td><td>• Can see list of available semesters</td></tr>
<tr><td>Click <code>"Subject"</code> dropdown</td><td>• Can see list of subject codes</td></tr>
<tr><td>Click <code>COMP</code> subject</td><td>• Sections for <code>COMP</code> should show up
• Intercepted network requests to /classes and /schedule for the <code>COMP</code> subject should match response bodies requested by Cypress</td></tr>
<tr><td>Click the + button for any random class.</td><td>• Class should show up in the selected classes Section</td></tr>
</table>
</td></tr>
</table>

TC-5 - Can't Add Classes If It Conflicts

Action	Expected Result
Click `"Term"` dropdown	• Can see list of available semesters
Click `"Subject"` dropdown	• Can see list of subject codes
Click `COMP` subject	• Sections for `COMP` should show up • Intercepted network requests to /classes and /schedule for the `COMP` subject should match response bodies requested by Cypress
Click the + button for any random class.	• Class should show up in the selected classes Section
Find and try to add a class that happens at the same time as the class selected in the previous step	• Class is not added • Popup message stating `"This class conflicts with current schedule!"` shows up

TC-6 - Can Remove Classes

Action	Expected Result
Click `"Term"` dropdown	• Can see list of available semesters
Click `"Subject"` dropdown	• Can see list of subject codes
Click `COMP` subject	• Sections for `COMP` should show up • Intercepted network requests to /classes and /schedule for the `COMP` subject should match response bodies requested by Cypress
Click the + button for any random class.	• Class should show up in the selected classes Section
Click the X button for the class selected in the previous step.	• Class should no longer be in the selected classes section

TC-7 - Cost

Action	Expected Result

	Click `"Term"` dropdown	• Can see list of available semesters
	Click `"Subject"` dropdown	• Can see list of subject codes
	Click `COMP` subject	• Sections for `COMP` should show up • Intercepted network requests to /classes and /schedule for the `COMP` subject should match response bodies requested by Cypress
	Click the + button for any random class.	• Class should show up in the selected classes Section • Cost should update in the top right with amount $2326

- MAJORS Page

Test Case ID	Test Case Steps/Expected Results	
TC-8 - Can Visit All Majors	**Action**	**Expected Result**
	Visit Majors page	• URL should be, `"https://csundash.kyeou.xyz/majors"`
	Click every link on the page	• URL should match REGEX => `"/https:\/\/csundash.kyeou.xyz\/majors\/[a-z0-9]{4}/"`

- FACULTY Page

Test Case ID	Test Case Steps/Expected Results	
TC-9 - Can View Faculty Members	**Action**	**Expected Result**
	Visit Faculty page	• URL should be, `"https://csundash.kyeou.xyz/faculty"`
	Click `"Subject"` dropdown	• Can see list of subject codes
	Click `ALL` subjects	• For every subject, a list of

		professors who teach that subject should show up • Intercepted network requests to /professors should match response bodies requested by Cypress

- RATINGS Page

Test Case ID	Test Case Steps/Expected Results
TC-10 - Can View Ratings	

Action	Expected Result
Visit Faculty page	• URL should be, `"https://csundash.kyeou.xyz/faculty"`
Click `"Subject"` dropdown	• Can see list of subject codes
Click `COMP` subject	• COMP professors should show up • Intercepted network requests to /professors should match response bodies requested by Cypress
Open dropdown for any professor	• `Professor Rating` button should be visible
Click `Professor Rating`	• URL should match REGEX => `"/https:\/\/csundash.kyeou.xyz\/ratings\/[A-Z]{4}\/[[\w+.?]*]*@[[\w+.?]*]*/"`

Test Case ID	Test Case Steps/Expected Results
TC-11 - Can Add Rating	

Action	Expected Result
Visit Faculty page	• URL should be, `"https://csundash.kyeou.xyz/faculty"`
Click `"Subject"` dropdown	• Can see list of subject codes
Click `COMP` subject	• COMP professors should show up • Intercepted network requests

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>to /professors should match response bodies requested by Cypress</td></tr>
<tr><td>Open dropdown for any professor</td><td>• <code>Professor Rating</code> button should be visible</td></tr>
<tr><td>Click <code>Professor Rating</code></td><td>• URL should match REGEX => <code>"/https:\/\/csundash.kyeou.xyz\/ratings\/[A-Z]{4}\/[[\w+.?]*]*@[[\w+.?]*]*/"</code></td></tr>
<tr><td>Fill in all required fields with random data and post the rating</td><td>• The rating should show up, being the last on the page</td></tr>
</table>
</td></tr>
<tr><td>TC-12 - Can Add Rating [Negative]</td><td>
<table>
<tr><th>Action</th><th>Expected Result</th></tr>
<tr><td>Visit Faculty page</td><td>• URL should be, <code>"https://csundash.kyeou.xyz/faculty"</code></td></tr>
<tr><td>Click <code>"Subject"</code> dropdown</td><td>• Can see list of subject codes</td></tr>
<tr><td>Click <code>COMP</code> subject</td><td>• COMP professors should show up
• Intercepted network requests to /professors should match response bodies requested by Cypress</td></tr>
<tr><td>Open dropdown for any professor</td><td>• <code>Professor Rating</code> button should be visible</td></tr>
<tr><td>Click <code>Professor Rating</code></td><td>• URL should match REGEX => <code>"/https:\/\/csundash.kyeou.xyz\/ratings\/[A-Z]{4}\/[[\w+.?]*]*@[[\w+.?]*]*/"</code></td></tr>
<tr><td>Fill in all BUT ONE required fields with random data and attempt to post the rating</td><td>• Should get alert saying <code>"All Fields Are Required"</code>
• Rating should not be posted</td></tr>
</table>
</td></tr>
</table>

7.0 CONTROL PROCEDURES

Problem Reporting

- Any and all errors will be logged during runtime.
- The test scripts will have a log of successful checks and an error log of any issues during execution (i.e. connection error, I/O Errors, etc)

Change Requests

- Lead Christian Jarmon is responsible for all sign-offs to changes while David Huezo and Micheal Balian will be responsible for requesting such changes

8.0 RESOURCES/ROLES & RESPONSIBILITIES

Specify the staff members who are involved in the test project and what their roles are going to be (for example, Mary Brown (User) compiles Test Cases for Acceptance Testing). Identify groups responsible for managing, designing, preparing, executing, and resolving the test activities as well as related issues. Also, identify groups responsible for providing the test environment. These groups may include developers, testers, operations staff, testing services, etc.

- Test Design: Michael Balian, David Huezo
- Test Execution: Christian Jarmon, Nima Shafie

9.0 RISKS/ASSUMPTIONS

Identify the high-risk assumptions of the test plan. Specify contingency plans for each (for example, a delay in delivery of test items might require increased night shift scheduling to meet the delivery date).

- Testing with the assumptions that remedies to identified bottlenecks can be produced
    - Any possible issue with the system that arises should have naive solutions and optimal solutions.
        - In the case the optimal solution can’t be implemented in time, the naive solution will be deployed