CSC 179

Section 01

Team: Red Shift

Team Members: Santiago Bermudez, Sabeeha Baqui, Steven Graham, Hardev Singh, Ivan Gutierrez, Eric Truong, Alex Tran

Project Name: Employee Health Dashboard

Test Report

Deliverable #3

Table of Contents:

1. Defect Report

    Functional Testing

    Non-Functional Testing

2. Summary & Analysis: (for both functional and non-functional testing)

1. Defect Report

Functional Testing

Document your team testing effort and defect report as follows:

Test case ID: 0001

Test case detail (procedure):

So, this test involves the basic function of adding an employee's data to the Vendia database from our front-end.

Expected output:

Our "Admin" should be able to type in and input data on the webpage and expect to see it register on Vendia.

Actual output:

*We got errors like the following:

- *error: The top-level-await experiment is not enabled (set experiments.topLevelAwait: true to enabled it)*
- *Module build failed (from ./node_modules/babel-loader/lib/index.js):*

Defect:

Our issue was that we had incorporated Apollo into our code, which caused issues with our attempts to add employee data as Apollo had conflicted with Vendia. We also did not fully understand the use of async functions at the time. It also turned out that we did not correctly use our API keys.

Defect Severity- see defect categories below: Severity 1

Date found: June 12th, 2022

Defect resolution - date fixed –closed date: June 16th, 2022

In order to fix this issue, we completely had to start over again from scratch, creating new unis and new project files. This time, we had to scrap the use of Apollo altogether, focusing strictly on just using Vendia's commands.

Test case ID: 0002

Test case detail (procedure):

This test involves the basic feature of viewing aggregate data for all employees on the front-end.

Expected output:

Our user should be able to select a category for which they can see the aggregate data on the webpage.

Actual output:

The user can see the categories to display on the web page and select them to see the data, but can't see anything beyond that. They would have to inspect and see the console to see the results. Any display attempts we had would give us garbage and errors.

Defect:

Our issue in this case was that we did not have a lot of knowledge when it comes to event handling, and so we would often lose time with writing code and fixing errors that pop up.

Defect Severity- see defect categories below: Severity 1

Date found: June 12th, 2022

Defect resolution - date fixed –closed date: June 22nd, 2022

As a simple remedy to this issue, we decided to use simple window pop-up notifications to help resolve our display issue. This approach simplified things compared to our attempts to create new pages and expand things on our current page.

Test case ID: 0003

Test case detail (procedure):

This test involves the basic feature of viewing filtered data for certain categories of employees on the front-end and searching for specific data.

Expected output:

Our user should be able to select a filter or search with a 'keyword' for the data they are looking for on the webpage.

Actual output:

The user can see the search bar and filtering stuff to display more specific data on the web page, but they can't see anything beyond that. They would have to inspect and see the console to see the results.

Defect:

The issue in this case was getting using Vendia's listing function with the name of an employee as opposed to the employee's ID, so the challenge was figuring out a way to get an ID from an employee's name, as there was no specific function for that.

Defect Severity- see defect categories below: Severity 1

Date found: June 17th, 2022

Defect resolution - date fixed –closed date: June 22nd, 2022

We solved this issue by first using lift to filter out by name and then used list indexing to grab the first instance of an employee if there are multiple employees with the same name. With list indexing, we were also able to use syntax to grab the ID property of an employee object.

Test case ID: 0004

Test case detail (procedure):

This test case involves looking at whether one can share data with another user, who is Using a second Vendia account. Is it possible to invite someone to participate and create a partner node and to share with that account the data without personal identification information?

Expected output:

The admin in this case should be able to share their data with a specific individual and hold back the names of employees. They should also be able to revoke access to that data.

Actual output:

Nothing, in this case.

Defect:

The issue in this case was that we did not have Vendia set up properly for the admin to be able to use mutations to restrict access to certain types of data (*in our case, the employee names). We needed to start over with a new schema using a new top level type called "x-vendia-acls".

Defect Severity- see defect categories below: Severity 2

Date found: June 15, 2022

Defect resolution - date fixed –closed date: June 30th, 2022

We fixed this issue by creating a new uni with a revised schema. We then collaborated together to test the mutations and queries needed to make data sharing work.

Non-Functional Testing

Address and summarize NFRs testing such as usability, performance and GUI improvements- more on this later on as we know more from the client …

Test case ID: 0001

Test case detail (procedure):

This test involves the issue of whether our website can accurately calculate the aggregate data for a small group of employees.

Expected output:

The website should be able to do accurate calculations for employee averages when prompted to display such data without error.

Testing Type: Reliability testing.

Defect: None found.

Defect Severity- see defect categories below: N/A.

Date found: N/A.

Defect resolution - date fixed –closed date: June 21st, 2022

Test case ID: 0002

Test case detail (procedure):

This test involves the issue of whether the user can reliably understand and use the dashboard interface without any confusion or need for too much assistance

Expected output:

The user should be able to know what to do or how to interact with the functions that are provided in the website on an intuitive level. The user should also be able to use everything with no incomplete or unnecessary functions.

Testing Type: Usability Testing

Defect: There was a sidebar in our webpage which had some links to other pages, but no real usage or function whatsoever.

Defect Severity- see defect categories below: Severity 4

Date found: June 16th, 2022

Defect resolution - date fixed –closed date: June 20th, 2022

Test case ID: 0003

Test case detail (procedure):

This test involves the issue of whether or not our program can be modified with ease to correct defects, improve performance, or add new features.

Expected output:

One should be able to add new features or functions without having to change up other program files or without having to alter too many other program files. One should also be able to make changes to a certain program file without affecting other files (*or without affecting to many other files) directly or indirectly.

Testing Type: Maintainability Testing

Defect: None found.

Defect Severity- see defect categories below: N/A.

Date found: N/A.

Defect resolution - date fixed –closed date: June 16th, 2022

2. Summary & Analysis: (for both functional and non-functional testing)

An overview of the test effort and test results analysis

Functional Testing:

The testing effort on this app involved automated testing through react. Throughout the course of testing, we came upon several defects in the application that were categorized and resolved by the fixed date specified. The outputs were fixed to match what was expected

Non-functional testing:

The non-functional testing effort on this website mainly focused on things like whether or not our user can easily understand functions of our dashboard without too much guidance, or whether our code can be modified and expanded upon without creating too many issues for the system as a whole.

Defect Severity Classification:

Severity 4 - Low defect - Does not impact the functionality- Trivial.

Severity 3 - Medium defect- issue with a specific item and a function - does not block functionality.

Severity 2- Major defect - impact a certain module and the function is not fully working

Severity 1- Critical defect- Impacts the most important features – not working and not functional as expected.