San Francisco State University

SW Engineering CSC 648/848

Milestone 4

Section 04 Team 05

Project title

Priority Hub (Code: PH)

Team members

- Luis Aguilar (Team Lead)
- Tyler Tam (Scrum Master)
- Trevor Eichler (GitHub Master)
- Andy Li (Front Lead)
- Lam Tran (Back Lead)

Revision History

Version	Date	Author	Comments
1.0	Friday, November 10th, 2023	Team 05	Draft version
1.1	Wednesday, November 29th, 2023	Team 05	Final version

November 29th, 2023

Table of contents

Content and structure for Milestone 4 document for review by institutors

1. QA Testing

1.1. Unit Testing

1.1.1. Selected P1 Features

ID	Category	Feature name	Description
1	Authentication	User should be able to Sign In	• User should be able to enter its credentials (email and password) and access the application.
2	Task Management	User should be able to Create a Task	• User can create a Task by filling the **Title**, **Due Date**, and **Category** fields.
3	Task Management	User should be able to Complete a Task	• User can click a **Complete** button to set the task as completed. • After setting the task as completed, it won't be shown anymore in the list of tasks. It will only be accessible using the **History of tasks** button.
4	Task Management	User should be able to Delete a Task	• User can click a **Delete** button to delete a specific task. • After deleting the task, it won't be shown anymore in the list of tasks. It will only be accessible using the **History of tasks** button.
5	Task Management	User should be able to track their time when working on a Task	• When accessing a task, the user can record it's time (stopwatch functionality) when working on that task. • They can record their time multiple times per task.

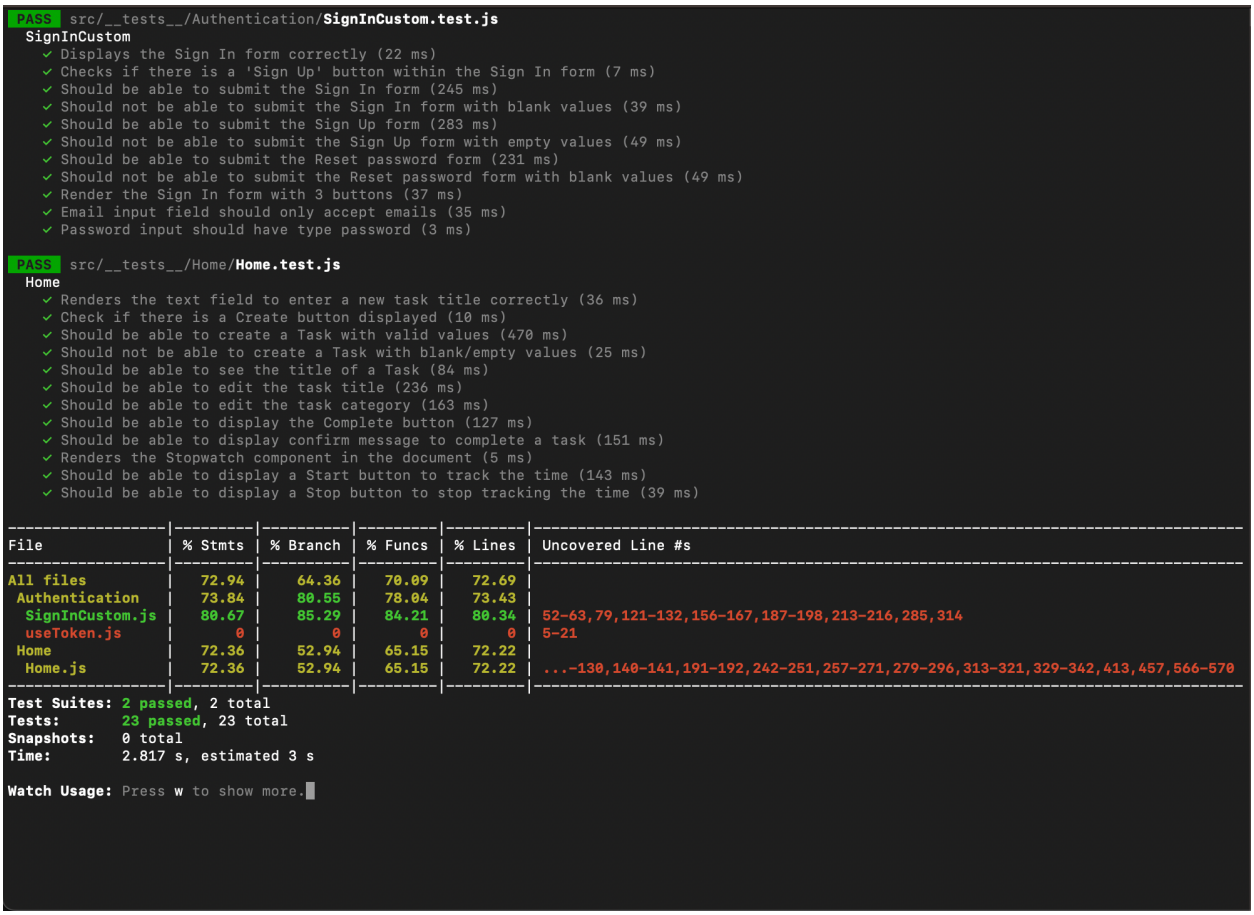
1.1.2. GitHub URL of test cases directory

Test Cases directory (testing and development branches):

- https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team05/tree/testing
- https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team05/tree/development

1.1.3. Description of functional and statement coverage

1.1.3.1. Frontend coverage



All our 5 selected P1 features code are in the following files:

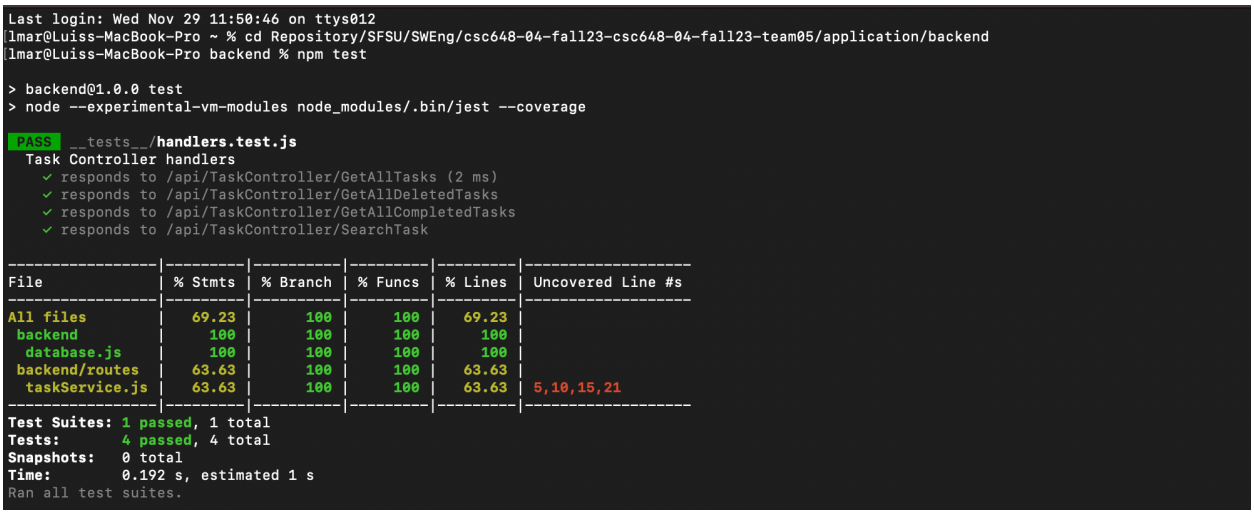
- SignInCustom.js
- Home.js

Therefore, we created the following files for the unit tests:

- SignInCustom.test.js
- Home.test.js

We developed tests for 5 P1 selected features in the **SignInCustom.test.js** and **Home.test.js** test files. Some of the tests verifies if the React components render correctly, if the Sign In form loads correctly, if the Sign In form allows blank values, if the correct buttons are displayed, etc.

1.1.3.2. Backend coverage



Our features are in the following files:

- task.js
- taskService.js

Our tests were developed in the following files:

- handlers.test.js

We developed unit tests to verify if the backend methods for task management were working properly. For example, we test if the function that in charge of returning the list of tasks, returns a list of objects. We mocked data to test these task management functions.

1.2. Integration Test

1.2.1. Selected P1 Features

#	Category	Feature name	Description
1	Authentication	User should be able to Sign In	• User should be able to enter its credentials (email and password) and access the application.
2	Authentication	User should be able to Sign Up	• User can sign up by filling their **first name**, **last name**, **email,** and **password**.
3	Task Management	User should be able to Create a Task	• User can create a Task by filling the **Title**, **Due Date**, and **Category** fields.
4	Task Management	User should be able to Edit Task	• User can select a task from the list, click in the **Edit** button, and edit any field from the form.
5	Task Management	User should be able to Delete Task	• User can select a task from the list, click the **Delete** button, application will confirm the action, and the task will be removed from the main list of tasks, and only be visible when accessing the **History of tasks** button.

6	Task Management	User should be able to Complete a Task	• User can click a **Complete** button to mark the task as completed. • After marking it as completed, it won't be shown anymore in the list of tasks. It will only be accessible using the **History of tasks** button.
7	Task Management	User should be able to Search Tasks	• User can search through the list of tasks by typing any string value.
8	Task Management	User should be able to track their time when working on a Task	• When accessing a task, the user can record it's time (stopwatch functionality) when working on that task. • They can record their time multiple times per task.
9	End of the Day Summary	User should be able to view an **End of the day** summary for all their tasks.	• In the **Summary** screen (option in sidebar), the user can select a date and see their overall progress of their tasks within that selected date.
10	Dashboard	User should be able to filter the task's charts using two date fields	• In the **Dashboard** screen (option in sidebar), the user can select a range between two dates and load the data within four different charts to provide different metrics.

1.2.2. Test Cases

Test cases are in the excel file named "***Priority Hub Integration Testing.xlsx***"

This file has been submitted to the master and development branches:

- https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team05/tree/master
- https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team05/tree/development

1.2.3. Results of test cases

The process of the Integration testing was as follows:

1. The assigned QA Tester performs the testing following the steps of its assigned test cases.
2. If a step of a test case is not working as expected, its result is set to "fail", and a bug/issue is created using the GitHub Issues tool.
3. Each issue created is assigned to the developer in charge of the development of the feature where the bug was found.
4. The assigned developer must fix the issue with a commit and a pull request into the development branch. Later, the code needs to be deployed into our online website for the QA Tester to re-test it.
5. After re-testing, if the bug is solved, the issue is closed in GitHub Issues.

List of issues (GitHub Issues) registered after Integration testing:

ID	Feature #	Test Case	Issue #	Issue description	Pull request #	Pull request (solution) description
1	4	PH-TC-013	#9	Due date is not working properly when editing a task	#13	• **Cause**: The code was not considering the change of time zone in our current area (California). • **Solution**: We turned the dates into UTC and subtract the days according to our current time zone.
2	5	PH-TC-017	#11	The Dashboard option was not being displayed in the sidebar of the application	#13	• **Cause**: The option was not deployed in the online website • **Solution**: The code was deployed into the website web server.
3	8	PH-TC-024	#12	The Start button of the Activity Tracker section, within each task, was disabled.	#13	• **Cause**: The functional button code was not deployed into the online website. • **Solution**: The functional code was deployed into the website's web server.

All issues have been solved and the sanitized code is in the **development** branch.

1.2.4. Bug Tracker System

GitHub Issues: https://github.com/CSC-648-SFSU/csc648-04-fall23-csc648-04-fall23-team05/issues

1.2.5. Description of test coverage

In total, the team tested 10 P1 features with 31 test cases.
From the 10 P1 features, the team found 3 issues/bugs:

- 1 bug in Feature 4 [User should be able to Edit Task]

- 1 bug in Feature 5 [User should be able to Delete Task]
- 1 bug in Feature 8 [User should be able to track their time when working on a Task]

In the initial testing, the application passed 28 test cases, except for the 3 test cases mentioned in the table in Results of test cases

From the 3 bugs registered, 2 bugs were related to some new code still pending to be deployed into our online website. The remaining bug was due to the code was not considering the time zone change.

After the bugs were registered in **GitHub Issues**, the team solved them by performing a new pull request to the **development** branch. Then, after more re-testing, issues were solved and closed.

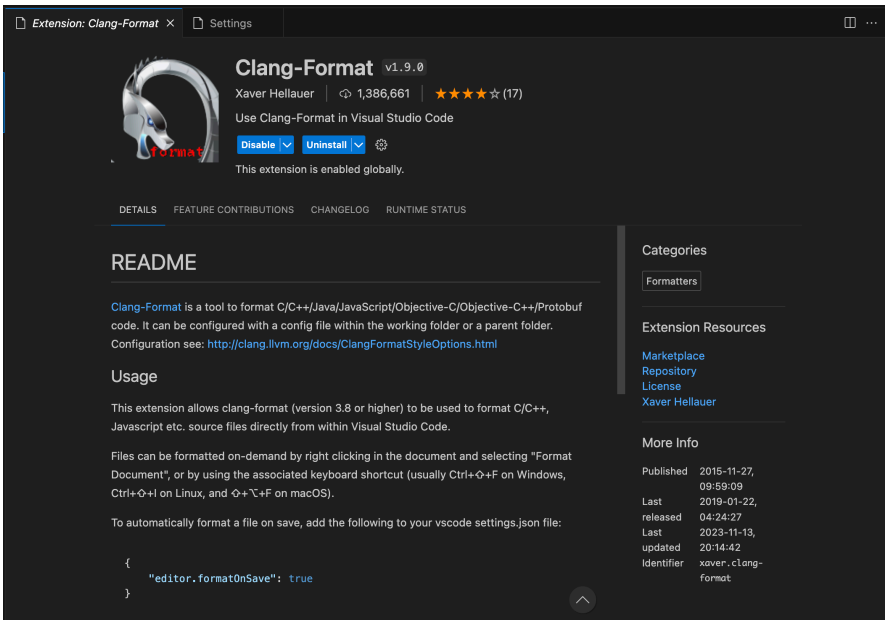
2. Coding Practices

2.1. Coding Style

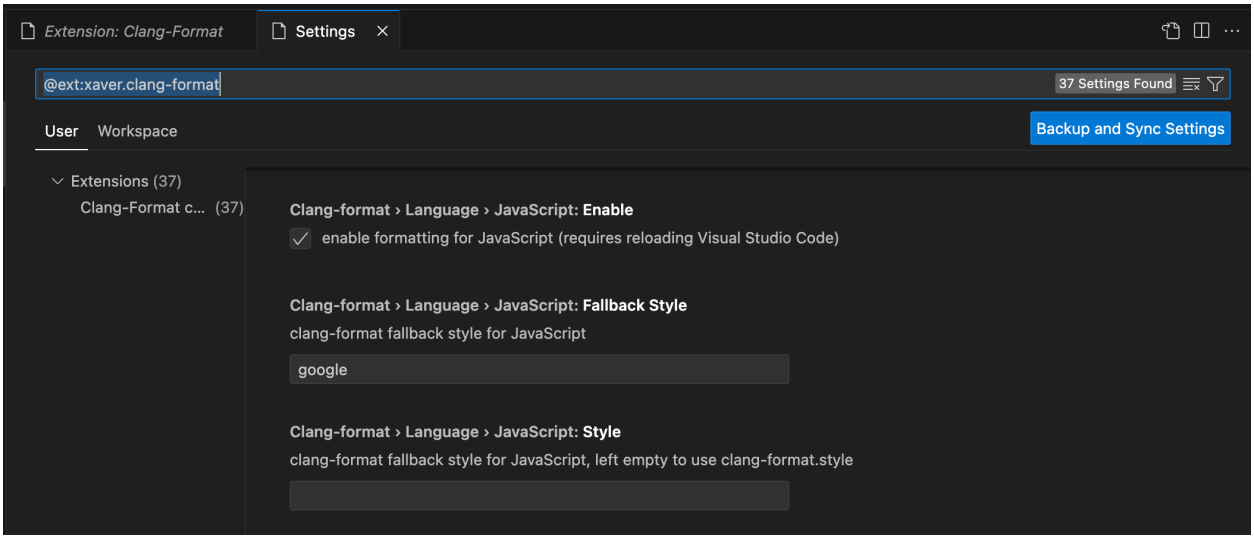
Our team chose the **Google JavaScript Style Guide**[1] for our application coding style.
Since our application uses **JavaScript** as the main programming language (React for Frontend and Express + NodeJS for Backend) we decided to use the Google style. This ensures a common coding style across all our application.

To enforce the style the team uses the **Clang-Format**[2] extension in **VSCode** (default IDE). Therefore, when a team member needs to submit his changes to our **GitHub** repository, they first run the extension's formatting function. This way our **JavaScript** code is formatted accordingly to the Google style.

[1] https://google.github.io/styleguide/jsguide.html
[2] https://clang.llvm.org/docs/ClangFormat.html



Also, to use the extension we had to configure it in **VSCode**. We enabled the **JavaScript** formatting option and set **Google** as the coding style used for the formatting function.



Source files related to 5 P1 features that shows coding style:

ID	Category	Feature name	Source files
1	Authentication	User should be able to Sign In	• (frontend) SignInCustom.js • (frontend) useToken.js
2	Authentication	User should be able to Sign Up	• (frontend) SignInCustom.js • (frontend) useToken.js
3	Task Management	User should be able to Complete a Task	• (frontend) Home.js • (backend) task.js

			• (backend) taskService.js
4	End of the Day Summary	User should be able to view an **End of the day** summary for all their tasks.	• (frontend) SummaryMain.js • (frontend) SummaryTasksCompleted.js • (frontend) SummaryTimeSpent.js • (frontend) SummaryTimeSpentTable.js • (backend) summary.js
5	Dashboard	User should be able to filter the task's charts using two date fields.	• (frontend) DashboardMain.js • (frontend) DashboardTaskByCategory.js • (frontend) DashboardTaskByDate.js • (frontend) DashboardTaskByStatus.js • (frontend) DashboardTaskByTimeSpent.js • (backend) dashboard.js