**CS673 Software Engineering** 

**Team 3 - Health and Wellness Tracker**

**Software Test Document**

| **Team Member** | **Role(s)** | **Signature** | **Date** |
| --- | --- | --- | --- |
| Amanda Yee | QA Leader | *Amanda Yee* | 09/22/2024 |
| Yu Luo | Design & Implementation Leader | *Yu Luo* | 09/22/2024 |
| Kenny Light | Requirements Leader | *Kenny Light* | 9/23/2024 |
| Chris Ceravolo | Team Leader | *Chris Ceravolo* | 9/23/2024 |
| Zihao Qian | Security | *zihaoqian* | 9/23/2024 |
| Edward Lee | Configuration | *Edward Lee* | 9/23/2024 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Team 3 | 09/23/2024 | All sections |
| 2 | Amanda Yee | 10/07/2024 | - Updated Testing Summary, Automated Testing Report, and Testing Metrics to reflect Iteration 2  - Added Manage Profile test to Manual Testing Report |
| 2 | Chris Ceravolo | 10/07/2024 | Update manual test of charts |
| 2 | Eddie Lee | 10/07/2024 | Add Authentication test to Manual Testing Report |
| 3 | Amanda Yee | 10/13/2024 | Updated Automated Testing Report, Testing Metrics, and Testing Summary |
| 3 | Zihao Qian | 10/14/2024 | Update Manage Goal Testing report of frontend and backend. |
| 3 | Chris Ceravolo | 10/14/2024 | Updated manual test of charts |

[Testing Summary](#_heading=h.gjdgxs)

[Manual Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Summary

*In this section, you will summarize what was tested, who is involved in testing, testing techniques used, and testing result. You may have the following tests*

* + *Unit Testing*
  + *Integration testing*
  + *System Testing*
  + *Acceptance Testing*
  + *Regression Testing*

Testing is an important part of building our web application to ensure that it functions correctly and aligns with our expectations, while also helping us to identify any potential issues before integrating new features.

We have implemented unit tests to test individual components of our application in isolation to ensure they are working as intended. We continued to build out this testing suite during development and all team members were involved. We have various unit tests for the front end as well as back end connection. They all currently pass.

We also conducted integration testing during development to ensure that the front end (client-side application) and back end (external database) are communicate properly with each other. We manually verified that the data submitted is correctly saved in the database, and that the retrieved data matches our expectations. Any team member who works on features that connect the front and back end performed this integration testing and currently the integration testing passes.

Our team has set up Docker which helps to streamline the system testing process. The containerized environment ensures that system testing is consistent and replicable across different stages of development. Before each release, the QA leader will conduct a final system test through Docker to check that the application is functioning as intended. Currently, this system test passes.

# Manual Testing Report

*In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.*

*Here is a sample template that can be used for each test case. For system tests or acceptance tests, you may also include some screenshots.*

* Test case ID, name
* New or old:
* Test items: (what do you test )
* Test priority (high/medium/low)
* Dependencies (to other test case/requirement if any):
* Preconditions: (if any)
* input data:
* Test steps:
* Postconditions:
* Expected output:
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:

(You can use an additional spreadsheet for this section as well)

## Test: Create and View Users

* Test items: (what do you test ): **CreateUser.js, ViewUser.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **N/A**
* Preconditions: (if any): **Server and frontend are running**
* Test steps: **Navigate to “Create Users” page in web browser, enter user data and hit submit, navigate to “View Users” page and refresh**
* Postconditions: **A list of user data appears on the page, including a row for the user that has just been created**
* Pass or Fail: **Pass**
* Bug id/link: <https://seprojects-cs673olf24team3.atlassian.net/browse/HW-27>
* Additional notes: **This test is to be run by admin and not users. This test ensures that the frontend and server are communicating properly.**

## Test: Create Goals

* Test items: (what do you test): **CreateGoal.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **No**
* Preconditions: (if any): **Server and front end are running**
* Test steps: **Navigate to “Create Goal” page in application, select health metric from dropdown, input target value and click “Submit” button.**
* Postconditions: **A new document in the “goals” collection on MongoDB should be created with an autogenerated goalId.**
* Pass or Fail: **Pass**
* Bug id/link: (this should link to your github issue id): **https://seprojects-cs673olf24team3.atlassian.net/browse/HW-97**
* Additional notes: **This test ensures that the front end and server are communicating properly, whereby the user input information is saved to the external DB.**

## Test: Create Daily Health Data Form Submission

* Test items: **DailyData.js**
* Test priority: **High**
* Dependencies (to other test case/requirement if any): **N/A**
* Preconditions: (if any)
  + **The Daily Data Form must be fully rendered with all MUI components loading correctly**
  + **Mulish font should be applied globally**
  + **All form input fields should be visible and editable**
* Test steps:
  + **Render Form: Navigate to the Daily Data form page to ensure all fields (Weight, step count, sleep, water, exercise) are rendered properly**
* Postconditions: **All input fields should be filled out with valid data.**
* Pass or Fail: **Pass**
* Bug id/link: <https://seprojects-cs673olf24team3.atlassian.net/browse/HW-105>
* Additional notes:
  + **Ensure that the input fields reflect the proper styles as per the design: rounded corners, background color, and correct font (Mulish).**
  + **Verify that the dropdown fields are accessible and maintain their values after selection.**
  + **Check that focus states and form validation (if applicable) behave as expected.**

## Test: Docker Configuration

* Test items: **docker-compose-dev.yml, client/Dockerfile.dev, server/Dockerfile.dev**
* Test priority: High
* Preconditions: **All dependencies are present in package.json, Docker images have been built**
* Test steps: **Run “docker-compose -f docker-compose-dev.yml up”**
* Expected output: **Automatic redirect to localhost:3000, where the application functions just as when run locally using “npm start” in two terminals**
* Pass or Fail: **Pass**
* Bug ID/link: <https://seprojects-cs673olf24team3.atlassian.net/browse/HW-114>, <https://seprojects-cs673olf24team3.atlassian.net/browse/HW-159>
* Additional notes:
  + **Since Docker was initially set up with a single Dockerfile in the code directory, we needed to reconfigure after having both a server and client that needed a Dockerfile for each of them**
  + **Also tested with “curl** [**http://localhost:5000**](http://localhost:5000)**” to validate the server could be reached, expecting a response code: 200 with Content: “Server is running…”**

## Test: Login Page

* Test items: (what do you test): **Login.js, App.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **N/A**
* Preconditions: (if any): **Server and frontend are running**
* Test steps: **Navigate to “Login” page in web browser, input wrong password**
* Postconditions:  **Pop up ‘check your password and email’ notification**
* Pass or Fail: **Pass**
* Bug id/link:[**https://seprojects-cs673olf24team3.atlassian.net/browse/HW-26**](https://seprojects-cs673olf24team3.atlassian.net/browse/HW-26)
* Additional notes: **This test is to be run by admin and not users. This test ensures that the login system and password input are communicating properly and the front end flags an incorrectly input password.**

## Test: Logout Button

* Test items: (what do you test): **Logout.js, Logout.test.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **N/A**
* Preconditions: (if any): **Server and frontend are running**
* Test steps: **Navigate to “Login” page in web browser after click the button**
* Postconditions: Pass or Fail: **Pass**
* Bug id/link: [**https://seprojects-cs673olf24team3.atlassian.net/browse/HW-178**](https://seprojects-cs673olf24team3.atlassian.net/browse/HW-178)
* Additional notes: **This test is to be run by admin and not users. This test ensures that the logout button could make sure to redirect to the login page and all the information of users will be invisible.**

## Test: Authentication

* Test items: (what do you test): **utils/authenticate.js, Login.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **N/A**
* Preconditions: (if any): **Server and frontend are running, user is logged out**
* Test steps: **Navigate to any page aside from “Login”. Expect to be redirected to the Login page. Log in with correct credentials, and navigate to another page again.**
* Postconditions:  **Redirect to Login page when logged out, access other pages when logged in**
* Pass or Fail: **Pass**
* Bug id/link: [**https://seprojects-cs673olf24team3.atlassian.net/browse/HW-163**](https://seprojects-cs673olf24team3.atlassian.net/browse/HW-163)
* Additional notes: **Validate that your JWT is stored in your local storage with Inspect. Under “Application”, there should be an “authToken” in “Local storage”**

## Test: Manage Profile

* Test items: (what do you test): **ManageProfile.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **No**
* Preconditions: (if any): **Server and front end are running and user is logged in**
* Test steps: **Navigate to “Manage Profile” page, update any personal information and click “Update Profile” button.**
* Postconditions: **The “users” collection on MongoDB is appropriately updated, specifically for the logged-in user ID and no other user IDs are impacted.**
* Pass or Fail: **Pass**
* Bug id/link: [**https://seprojects-cs673olf24team3.atlassian.net/browse/HW-27**](https://seprojects-cs673olf24team3.atlassian.net/browse/HW-27)
* Additional notes: **This test ensures that the front end and server are communicating properly, whereby the user input information is saved to the external DB.**

## Test: Manage Goal

* Test items: (what do you test): **CreateGoal.js, validateGoalForm.js,goalController.test.js**
* Test priority (high/medium/low): **High**
* Dependencies (to other test case/requirement if any): **No**
* Preconditions: (if any): **Server and front end are running and user is logged in**
* Test steps: **Navigate to the “Manage Goal” page, update any training goals and click the “Update Goal” button.**
* Postconditions: **The “goals” collection on MongoDB is appropriately updated, and**

**The goals will be saved in the front end, making sure users could check that everytime after they login in.**

* Pass or Fail: **Pass**
* Bug id/link: **https://seprojects-cs673olf24team3.atlassian.net/browse/HW-180**
* Additional notes:**This test updated in Iteration3.**

## Test: Charts on home page

* Test items: (what do you test): **Home.js**
* Test priority (high/medium/low): **Medium**
* Dependencies (to other test case/requirement if any): **No**
* Preconditions: (if any): **Server and front end are running in a local environment and user is logged in, OR app is deployed via Heroku and user is viewing live site**
* Test steps: **Enter data for the past one to seven days.** **Navigate to “Home” page.**
* Postconditions: **Five charts are visible with the latest data that you entered.**
* Pass or Fail: **Pass**
* Bug id/link: [**https://seprojects-cs673olf24team3.atlassian.net/browse/HW-54**](https://seprojects-cs673olf24team3.atlassian.net/browse/HW-54)
* Additional notes: **This test can be done iteratively, adding data, deleting data, to ensure things are working as expected.**

# Automated Testing Report

*Describe briefly the automated testing you have done, including where the test code resides in your code repository, what test frameworks are used, and the screen shots or generated testing report.*

We are using the Jest Testing Framework for automated testing.

The test code resides in two locations in our repository, separated by the client side and server side. Each of these folders contain multiple test files, organized by the specific functionalities being tested.

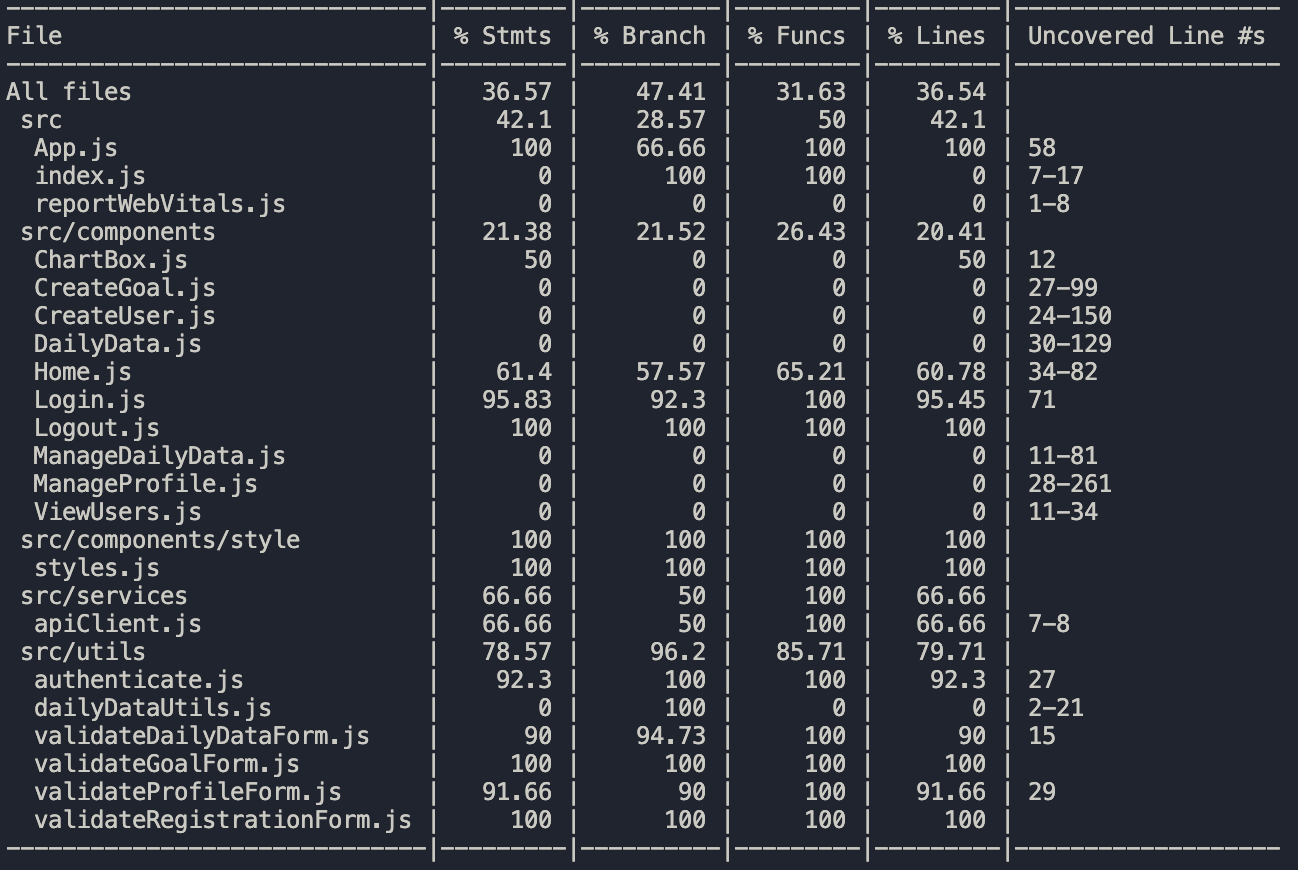
**Client side testing folder: /code/client/src/\_\_tests\_\_**

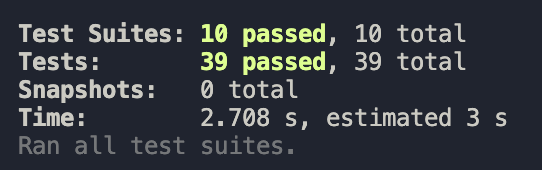
This folder contains 10 testing suites with a total of 39 tests. It tests form

validation for each of our pages, page rendering, chart rendering, user authentication

and connection to the server.

*Client side testing report:*

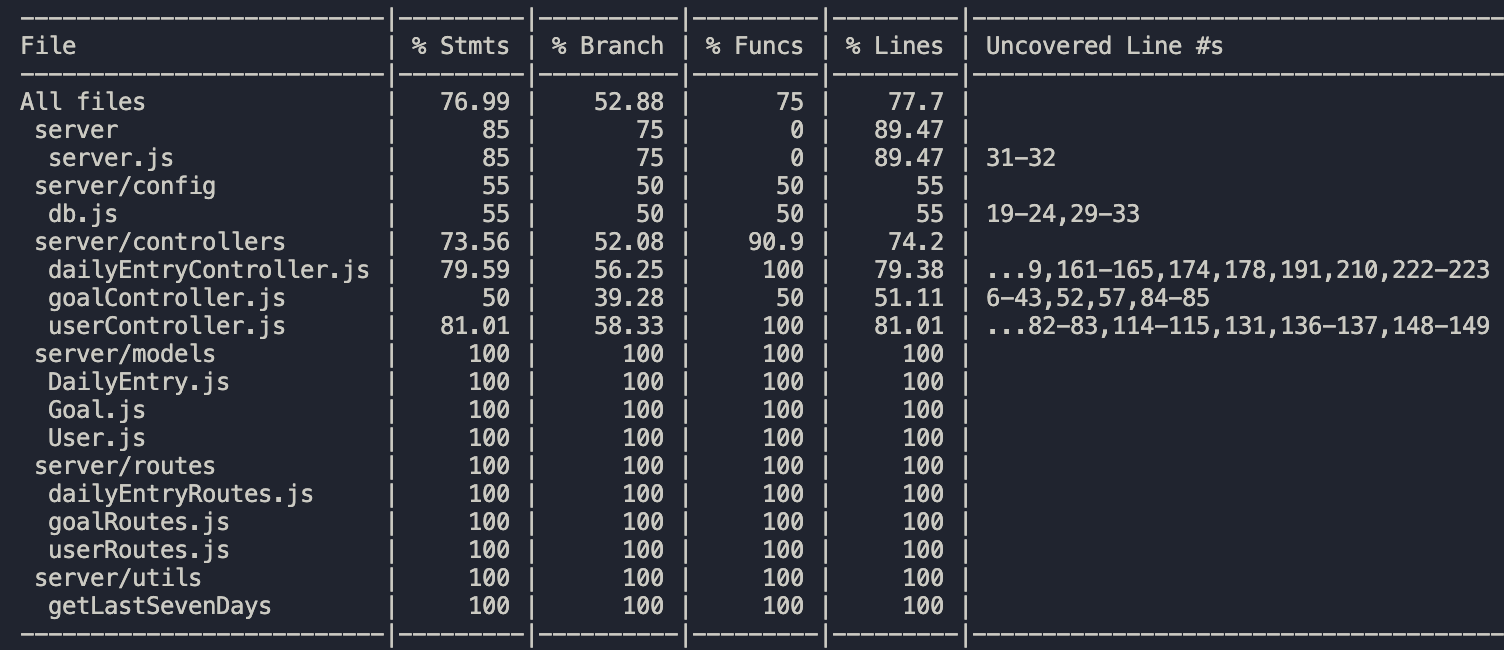


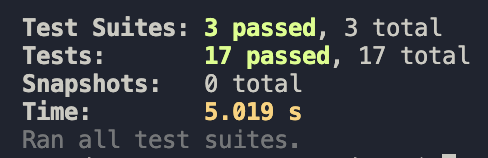


**Server side testing folder: /code/server/\_\_tests\_\_**

This folder contains 3 testing suites with a total of 17 tests. These test the user and goal endpoints, ensuring that users and goals are created successfully and that users can be retrieved. It also tests the login functionality is working as expected and the correct user is returned based on who logs in.

*Server side testing report:*





Additional tests will be created as we develop our application.

# Testing Metrics

*In this section, you shall report any metrics used for the evaluation, e.g. # of test cases, test coverage, defects rate, etc.*

* + Test Case Count: 56
  + Test Pass Rate: 100%
  + Test Coverage for Statements (server): 76.99%
  + Test Coverage for Statements (client): 36.57%

# References

* + Jest Framework: <https://jestjs.io/>

# Glossary

N/A