**CS673 Software Engineering** 

**Team X - Project Name**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Bharat Gogineni | Team leader | *Bharat Gogineni* | 5/28/2022 |
| Ruiqi Chang | Requirement Leader | *Ruiqi Chang* | 5/28/2022 |
| Chinmay Bhelke | Design and Implementation Leader | *Chinmay Bhelke* | 5/15/2022 |
| Benoît Clemenceau | QA Leader | *Benoît Clemenceau* | 5/15/2022 |
| Taina Conde | Configuration Leader | *Taina Conde* | 5/30/2022 |
| Chinmay Bhelke | Security Leader | *Chinmay Bhelke* | 5/28/2022 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.0** | **Benoît Clemenceau** | **5/31** | **Initial version** |
| **2.0** | **Benoît Clemenceau** | **6/14** | **Manual tests and automated Testing reports** |

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

[Manuel 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

We are using manual tests as well as automated tests. Each team member tested what they developed before committing, but we will need automatic testing for the next iteration. For that, the Jest testing framework has been installed on the front-end and QA Leader and team members started writing unit tests.

# 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)

* Logging In
* Test items: Auth0 flow to log in to the app using email or an external provider
* Test priority (high/medium/low): high
* Dependencies (to other test case/requirement if any):
* input data: user email / external provider account
* Test steps:
  + Click on Login / Signup button
  + Either:
    - Enter email address and password
    - Click Continue button
  + Or:
    - Click Continue with Google button
    - Click to choose Google account
* Expected output: User gets redirected to /home and is now logged in
* Actual output: User gets redirected to /home and is now logged in
* Pass or Fail: PASS
* Adding a job to track
* Test items: Adding a job user story
* Test priority (high/medium/low): high
* Dependencies (to other test case/requirement if any): Logging In
* Preconditions: Logged in
* input data: Company, Job title, Job description, Status
* Test steps:
  + Click on Add Job button
  + Fill out Company, Job title, Job description, and Status fields
  + Click on Save Job button
* Expected output: New collapsible Job card appears on screen
* Actual output: New collapsible Job card appears on screen
* Pass or Fail: PASS

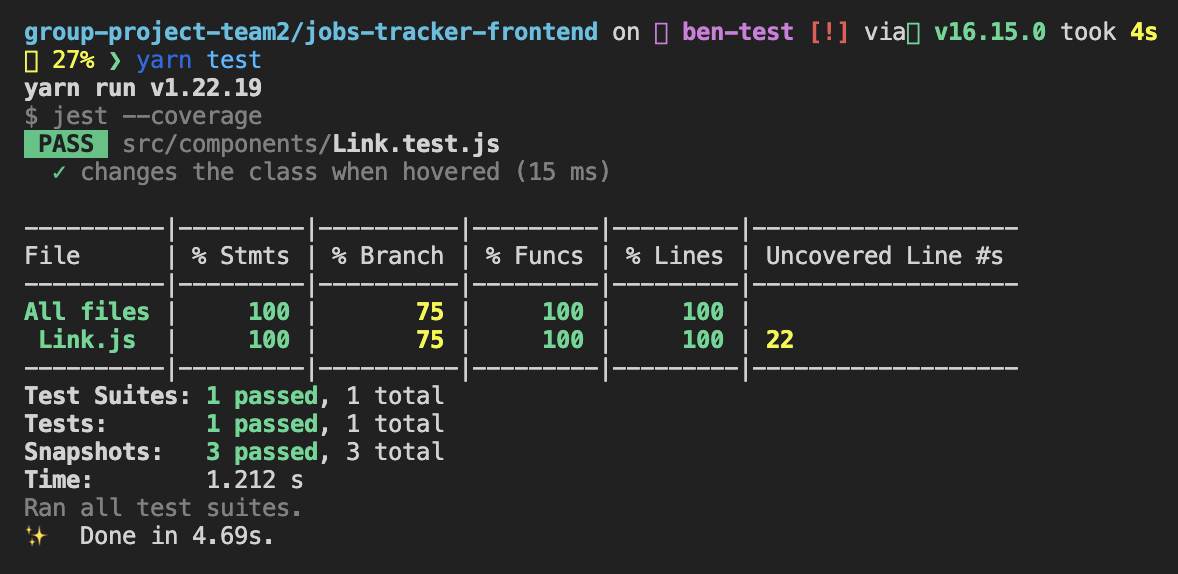
Same thing for basically every user story we are implementing.

# 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.

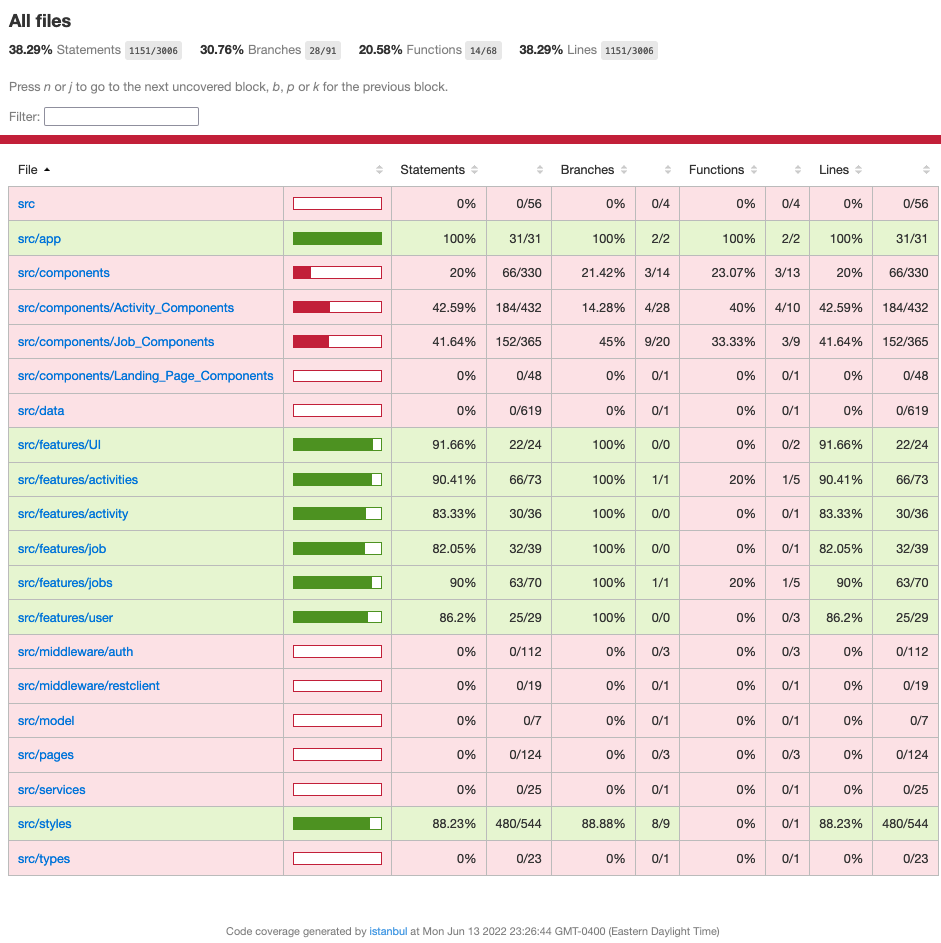
For the front-end, each component file has an associated Component.test.js file for writing unit tests.

When running npm test or yarn test, the Jest framework is being used to run all the tests and produces a report of the output of the tests as well as test coverage. This report looks like this in the CLI:



And can also be accessed via a HTML webpage being generated each time (group-project-team2/jobs-tracker-frontend/coverage/lcov-report/index.html):

**06/13 prod-branch test report:**



# Testing Metrics

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

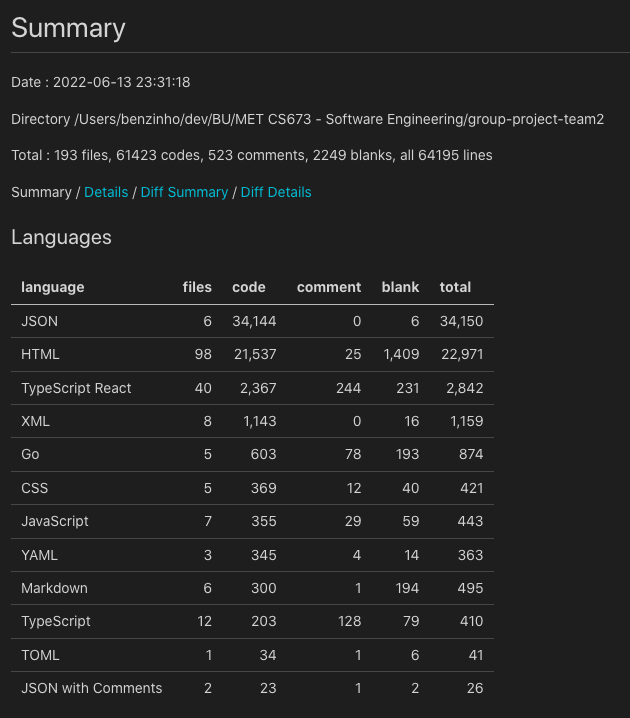
The metrics we defined in the SPPP regarding the testing strategy are: **Number of tests**, **Test Coverage**, **Percentage of tests passed**, **Total test duration**.

The Jest testing framework provides us with all these metrics and divides the Test Coverage one into 4 different metrics: Function coverage, Statement coverage, Branch coverage, and Line coverage. These metrics can be defined as follows:

* Function coverage: Has each function (or subroutine) in the program been called?
* Statement coverage: Has each statement in the program been executed?
* Branch coverage: Has each branch (also called DD-path) of each control structure (such as in if and case statements) been executed? For example, given an if statement, have both the true and false branches been executed? Another way of saying this is, has every edge in the program been executed?
* Line coverage: has each executable line in the source file been executed?

**06/13 prod-branch metrics:**

LOC: 2,367



Coverage: see upper section

# References

<https://krishankantsinghal.medium.com/how-to-read-test-coverage-report-generated-using-jest-c2d1cb70da8b>

# Glossary