

**Polytechnic University of Puerto Rico**

**Department of Computer Engineering and Computer Science**

**COE 4002-39: Capstone I / CS 4002-49: Computer Science Project I**

**WI-20 / Prof. Othoniel Rodriguez**

**Covid-19 Data Visualization & Analysis:**

Software Test Design (STD)

Version 2.0

|  |  |  |
| --- | --- | --- |
| Name | ID | Program |
| Jose M. Lopez Torres | #72583 | CS |
| Elias J. Marrero Ruiz | #97850 | CS |
| Ernesto Feliciano Muniz | #96861 | COE |
| Diego Canals Reyes | #80877 | COE |

# Table of Contents

[Table of Contents 2](#_Toc73662276)

[Introduction 4](#_Toc73662277)

[Behavioral Driven Development-Planification 5](#_Toc73662278)

[Gherkin Syntax 5](#_Toc73662279)

[TESTS 6](#_Toc73662280)

[Test-Driven Development 6](#_Toc73662281)

[Unit Test 6](#_Toc73662282)

[Jest 7](#_Toc73662283)

[Enzyme 7](#_Toc73662284)

[Testing 7](#_Toc73662285)

[Logs 8](#_Toc73662286)

[Regression Tests 8](#_Toc73662287)

[Agile Regression 8](#_Toc73662288)

[When 8](#_Toc73662289)

[Automation 9](#_Toc73662290)

[Cucumber 9](#_Toc73662291)

[Automating with Cucumber 9](#_Toc73662292)

[User Acceptance Tests (UAT) 10](#_Toc73662293)

[When and How to UAT ? 10](#_Toc73662294)

[User Acceptance Test of Web-Application 10](#_Toc73662295)

[Table 1: Detailed User Acceptance Tests 11](#_Toc73662296)

[Client Involvement 14](#_Toc73662297)

[Full Automation : Selenium IDE 14](#_Toc73662298)

[Justification for Automation Tools 15](#_Toc73662299)

[Software Quality Assurance Plan 16](#_Toc73662300)

[Prevention and Detection 16](#_Toc73662301)

[Documentation 16](#_Toc73662302)

[Code Reviews 17](#_Toc73662303)

[Templates 17](#_Toc73662304)

[Code Review Template for Development Team 18](#_Toc73662305)

[Template for User Acceptance Test Documentation 19](#_Toc73662306)

[Conclusion 20](#_Toc73662307)

[References 21](#_Toc73662308)

# Introduction

The following document will present all the information regarding the design of the tests to the web application. The Behavior Driven Development used in the project will be explained in detail as well the benefits of using it in the planification. All the different types between the development team and the client will be explained and given a specific time to happen. The tests that are performed will follow the test-driven development that will also be detailed in other sections of the document. The details of how each test is going to be performed , the criteria for the approve of the user acceptance test and the frameworks will also be explained.

# Behavioral Driven Development-Planification

The team development will follow the technique of Behavior-Driven Development(BDD) to enhance the use of Agile methodology. This is a strategy guided to develop accordingly to the behavior of the web application. The team selected this technique in order to establish a better communication with the client. BDD also compliments the use of the user stories that are going to be tested following Agile. This type of development will bring many advantages to the process including:

* The development team can respond quickly to the feedback given by the client.
* Improve the ability to deliver the web page application in the designated time.
* Be able to produce documentation that is automatically verified with the webpage behavior.
* Quick collaboration for resolving a problem between both entities.
* Go back quickly to a previous change because it will work with only small changes at one time.

To be able to harvest all these advantages the team will follow the best practices of BDD. These practices will help to set a standard of work that can be applied in each element of the webpage. The practices will be as follow:

* Each one of the user stories will be discussed by the development team and the client . The discussion will cover examples of functionality in order to reach a mutual agreement of what is expected of each one of them.
* The user stories will follow the basic structure of plain language of give-when-then.
* All the examples mentioned will be documented using the Gherkin Syntax and then discussed them with the client.
* Then these examples will be tested to see the behavior. This will be made by following an automated way of testing.

# Gherkin Syntax

The documentation of the examples that are going to be converted to the language of Gherkin . By using the rules and syntax of the Gherkin language the documentation can be easily understood and discussed with the client. Moreover, this language includes a set of keywords that will be used for the user stories that describe the desired behavior of the examples. The keywords will also give a standardized structured to be used during the development of the user acceptance tests. Furthermore , this language is the component key for creating automated tests in the Cucumber framework.

# TESTS

This section will describe the different tests that are going to be performed for the Software Test Design. These tests will use the Gherkin Syntax and will use the Test-Driven Development. Also, all the require tools, procedures and frameworks will be thoroughly discussed.

# Test-Driven Development

The Test-driven development (TDD) is the approach selected for following the Agile methodology. In which the test cases will be develop to each specific user story for validation. The development team will use this approach by writing single unit tests that describe a very detailed behavior of the code. Then the tests will follow the following criteria:

* Create the code for the designated test.
* Run the designated test.
* Verify is the test doesn’t affect other tests and if it passes.
* In case the test fails , fixed the code and test again.
* Repeat the process each time a test is added.

The TDD will prevent that bugs in the code from early stages are passed on to late dates in the development. This will help the development team and the client by saving resources and capital. Moreover, as the development progresses this approach will assure that all the code is tested before is presented the application and that meets all the requirements asked.

# Unit Test

The success of the TDD approach will be based in the unit tests. These tests are test uniquely design for a very specific part of the code. By isolating parts of the code, it will facilitate to test if the functionality is working correctly. Some examples that will be tested are functions ,methods and objects inside the code. Since the development team selected React.js and Node.js as the frameworks for developing the code, the unit tests will be developed in Selenium IDE.

## Jest

This framework will allow easy access to the results of the test and to the assertions. It will generate the details of the test performed in the console and is able to render content. This makes easier for accessing the log or information about all the unit test performed.

## Enzyme

Enzyme can provide a mechanism that is used for mounting and traversing the React.js. Moreover, it will permit the development team to assert , traverse and manipulate any component of the unit test. It also permits to wrap objects in the code and print them in the console.

## Testing

The development team will run the unit tests using Jest by following this sample procedure:

1. The first step will be to create inside the ***src*** folder a file ***index.test.js*.**
2. Then select the desired function or part of the code to be tested.
3. Then the test will be running to verity if the results are correct.
4. Create a new file called ***testconfig.json***
5. In this file code will be added for the frameworks to be used.
6. The “testMatch” path inside the json file will be changed to ***index.test.js***
7. Then add in the ***package.json*** the word “jest” in the test field. This will tell Node how to run the unit test.
8. The last part will be to run the test in the command line by using the command ***jest testSample -config=./testconfig.json***

In case that the development team needs more information about why a unit test is failing the framework of Enzyme is going to be used. This will allow to wrap objects in the console and add more functionality to the test. The reason behind is that Enzyme can be used to edit further the unit test because it permits to assert, manipulate, and traverse. This will be used to break down unit test in different parts such as methods , child components and root components that will help to identify any problem.

## Logs

The logs generated by the unit test that used the frameworks of Enzyme and Jest will be stored in Logstash . This is a platform that will help the team to use the information and keep track of all the test performed with their test results.

# Regression Tests

The regression tests will be performed by the development team each time a change is implemented in the code. This will be made for assuring that the existing features or capabilities or the code are not affected. Since the approach of Agile and TDD are used this will ensure that each new feature is working correctly, and the team can continue with the respective goals of the sprint. This will prevent to stuck in problems that could develop if the regression tests are not executed. Moreover , these types of test minimize the occurrence of bugs and works effectively to possible continues requests from the client.

## Agile Regression

Adopting the regression test into agile will follow the following methodology by the team:

* + First the team will select the tests cases for execution.
  + Estimate the regression test execution time
  + Identify which test are needed to be automated
  + Design a strategy for improvement in case of re-design

## When

The regression tests are going to be executed at the end of each sprint , before the start of the new sprint. It will be tested the functionality of the new test cases added and all the core functionalities of the webpage code.

## Automation

The regression tests are going to automated using a framework , this will help the team to save time and resources. The client can also be involved with the current status of the development and how well the team is performing. For automating the regression tests the team is going to be using Cucumber.

# Cucumber

This framework will automate the regression tests as well follow the BDD approach of the webpage development. Cucumber is a testing library that allows to write the test following the recurring structure of given-when-then of BDD. This is very useful because in the “given” the team can declare the preconditions discussed with the client for the creation of the user story. Then establish the actions that are going to happened in the “when” and the expected results in “then”. Then these user stories are going to be transformed using the Gherkin syntax before they are used in Cucumber. At the end the set of conditions and pre-conditions in Gherkin will be called scenarios. Moreover , the team will be using the version of Cucumber.js because the webpage development will be using JavaScript.

## Automating with Cucumber

The user stories that are transformed to gherkin will follow the following process that will automate the process. The automation will occur by creating scripts that will be re-used for testing in Cucumber. These scripts are written as .feature , and this extension is able to store the test for execution. The detailed steps of how the development team will automate the tests are presented in the following sample procedure:

* The scenario of the sample user story will be written in the ***.feature*** file and then executed in Cucumber.js.
* In the feature file the scenario will be written and all the conditions using the gherkin syntax.
* Then the feature file will be saved as ***sample.feature***
* Then the code will be executed and parsed by Cucumber.js by using the command ***./node\_module/.bin/cucumber-js./feature/sample.feature***.
* Then step definitions for the “when” and “then” will be added to a new created file named sampleSteps.js
* Since the webpage application will use a browser the Selenium framework is required as a driver for testing all the functionality in it.
* Test that can be made through selenium include behavior of the browser, finding elements and testing buttons.
* Then the team will load the web application . This will generate a Cucumber.js data table that will be available in sample.feature.
* After loading the application, the test is going to be executed again and the results will be printed in the console.

# User Acceptance Tests (UAT)

The development team will use UAT for the validation of the web application by the client. This testing will permit the client to approve the work performed in the software in this case the web application.

## When and How to UAT ?

The tests are going to be performed at the end of the development. The client must verify all the elements that all the requirements are met. The development team will record all the results of the UAT in a document. The UAT are be constructed in the same frameworks that were used to create the regression tests; Cucumber.js and Selenium. This will allow the part for storing the results to be standardized and maintained accessible for the client and the development team.

## User Acceptance Test of Web-Application

The team will present a detailed explanation of what determines the success criteria for each User Story UAT. The tests will specifically be conducted using the Selenium Library with JavaScript and will be added to Cucumber for test automation. All the UATs will be conducted during Regression tests each time the code in the master branch is changed. The UAT Pass Criteria column will list the steps taken with Selenium to verify the success criteria are met. This section will only describe how we will use Selenium to complete the tests and not the purpose of the tests. For a description of the test’s purpose please see the UAT Section of the SRS.

## Table 1: Detailed User Acceptance Tests

| User Story | UAT Pass Criteria |
| --- | --- |
| As a user I want to view a summary of daily cases, tests per day, deaths by day and hospitalizations so that I can review the data | * From menu, select summary. * Search for summary element in html. * If element exists, test success. |
| As a user I want to view graphically a summary of the deaths, cases and vaccinated people so that I can review the data | * From menu, select graph summary. * Search for graph summary element in html. * If element exists, test success. |
| As a user I want to have a view of 4 plots of each given country in the same page. | * Verify the number of plots displayed * Verify the number of windows * Verify table headers match, if headers match test success. |
| As a user I want to search municipality hospitalization data for a time period so that I can review the data | * From menu, select municipality hospitalization data. * Search for municipality hospitalization data element in html. * Verify table headers match, if headers match test success. |
| As a user I want to change the selected graphic from xy scattered to a pie chart so that it can be visualized differently | * On selected graphic, search and click for the change chart button element. * Validate the graphic element id in html changed, if changed test success. |
| As a user I want to change the zoom of a graphic so that a specific part can be seen more clearly | * On selected graphic, search and click for the change zoom button element. * Verify that the zoom text element on the chart increased. * Repeat test with decreased button. * If both tests pass, test success. |
| As a user I want to view a comparative graphic by age ranges and the total positive cases by age group | * From menu, select view cases by age range. * Verify graphic title element id, if id matches expected text, test success. |
| As a user I want to view a comparative graphic by age ranges and the total deaths by age group | * From menu, select view deaths by age range. * Verify graphic title element id, if id matches expected text, test success. |
| As a user I want to report an error in the webpage so that it can be fixed | * Search and click on report an error button element. * Fill report with dummy data. * Click on submit report button element. * Search for “Report submitted successfully” text element, it exists test success. |
| As a user I want to verify the references of the data showed so that I can also verify it | * Search and click on provided link to Client API. * Verify that redirection succeeded. |
| As a user I want to enable a colorblind mode through a button of a graphic so that I can see different colors on the graph | * Search and toggle color blind button element. * Search page elements to verify the ColorBlind css class is in use. * If true, test success. |
| As a user I want to share the current state of a graphic so that I can notify friends and family | * Search and click for the share button element on a graphic. * Select where graphic will be shared. * Verify that success element displays on the web application to determine success. |
| As a user I want to change the current graphic through a button so that I can view a different graph | * On a graphic, select the change graph button. * Verify that graph id and headers are different from previous values. * If true, test success. |
| As a user I want click on the option of graphs in the navbar and go to the graphs section. | * Search for html element that contains the navbar with the sections. * If element exists, test success. |
| As a user I want to search total deaths for a time period so that I can review the data | * From menu, select total deaths by time period element in html. * Verify table id and headers. * If above match, test success. |
| As a user I want to search total cases for a time period so that I can review the data | * From menu, select total cases by time period element in html. * Verify table id and headers. * If above match, test success. |
| As a user I want to view different age ranges of the cases in a plot so that it can be visualized the data better | * From menu, select graph cases by age element. * Verify plot displays different age line elements. * If above match, test success. |
| As a user I want to hover over a graphic so that I can see a specific data point | * On graph element, hover mouse over any point in the graph. * Search for the element id corresponding to graph hover element. * If element exists and contains text, test success. |
| As a user I want to be able to download/embed a given graphic to another website so that I can share the data | * On graph, search and click download/embed button element. * If embed button element is clicked, verify that embed snippet element is presented. * If download button element is clicked, verify that browser downloaded png file. * If both tests pass, test success. |

# Client Involvement

The client will be present before implementing in the sprint planning of each Sprint. In these meetings the client will describe the requirements that are needed for the application and the expected outputs. The client will also participate in the Sprint reviews at the end of each sprint in which it will verify the results of the regression tests and unit tests. In the presentation of these results the client will give direct feedback about how the requirements are been achieved and possible new minor requests. Moreover, the client will provide the data required by the web application by means of a proprietary API. Furthermore , the client will be able to test all the requirements and functionalities at the end of the development by testing the web application performance in the UAT.

# Full Automation : Selenium IDE

The development team after careful planning decided to give the automation of the tests a step further. The selection of the Selenium IDE will use the cucumber logic and implemented in the creation of scripts that run each one of the user acceptance tests without the need of a manual test. This scripts will be made by testing each of the functions that were expected in the user acceptance test , regression tests and units’ tests.

# Justification for Automation Tools

The tools that were used in the design of the user Acceptance test where the Gherkin syntax and Cucumber. These tools work directly with the methodology of Agile making that is used for the creation of the software. By using these tools combined with Agile some advantages are acquired like the following:

* Reduce the costs of the operation by managing all resources at optimal performance.
* Save time and be able to work under a schedule withing stipulated due dates.
* Be able to collaborate directly with the department of finances or business.
* The cucumber software is able to generate automatized test when a change is made in the code.
* The learning curve will be higher as team members already program using JavaScript.
* Two of the team members have worked directly with React.js and Node.js bringing their experience to the team.

These advantages where key in the decision of using the designated software, frameworks, and programming languages for all the tests that are going to be designed. Nonetheless, the team members that aren’t familiar with the frameworks agree to take a crash course that will help them develop the necessary skills for the project.

# Software Quality Assurance Plan

## Prevention and Detection

The development team will work following the Scrum Agile methodology in order to minimize errors or bugs by complying to these directions:

* + Each Software Developer on the team will be responsible for providing Unit Tests on the Backlog Items they worked on during a given week/Sprint.
  + The Unit Tests will serve the dual purpose of providing agile solutions to test and debug application code as well provide documentation with proper comments on how a given function works.
  + At the end of each development week, developers must complete individual code reviews on their branch features and begin constructing UATs for pertinent features.
  + At the end of a Sprint the Development team will convene to add available UATs to Cucumber for Automated Regression Testing.
  + Include in the individual code reviews the authored featured code and comments of the functions
  + Provide commented versions of the Unit tests and the reasoning behind.
  + Verify merge conflicts before creating a Pull request.
  + Perform at least 12 individual code reviews , 4 team code reviews during the trans course of the 4 sprints.

This will also improve the results of the development team because bugs can be detected in early stages of the development. Moreover , the involvement of the client in the test will assure that the web application does not stray for the established requirements.

## Documentation

Another standard for maintaining quality of the project will be that the development team keep logs and record of all test performed using LogStash. By documenting using and standardized framework the results will be consistent and better for understanding with the client. Furthermore, a printed documented for validating the User Acceptance Tests will be presented to the client for keep all the approvals, denials or feedback documented in paper.

## Code Reviews

The development team will also host individual code reviews to improve the quality of the code before adding it to the main project. It will also help to write efficient code without the minimum quantity of errors or bugs.

## Templates

The following section will present the templates that will be used for the code reviews and the user acceptance tests.

## Code Review Template for Development Team

**Covid-19 Data Visualization**

**Code Reviews**

**Polytechnic University of Puerto Rico**

**Team Member:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ **Date:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Element Tested: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Sprint: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Checklist**

**Functions/Methods/Execution Pass Fail**

Applications Loaded Correctly

Commented Correctly

Requirements Implemented

No Logic Errors

Clean Code Practices

Adopted Lightweight Code

No Efficiency Defects

Input Validation

**Comments :**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Template for User Acceptance Test Documentation

**Covid-19 Data Visualization**

**User Acceptance Tests**

**Polytechnic University of Puerto Rico & Department of Health of Puerto Rico**

**Client:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **UAT:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Date:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Team:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID # of UAT** | **Pass** | **Fail** | **Feedback** |
| **Sample Test 1** |  |  |  |
| **Sample Test 2** |  |  |  |
| **Sample Test 3** |  |  |  |
| **Sample Test 4** |  |  |  |

**Special Requests:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Client Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Date:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Conclusion

The development team will continue to work in the test following the BDD , TDD approaches without striving off Agile methodology. This will help the team to meet all the client specifications and comply with the sprints. Moreover, the end result of the web application project will be at optimal performance at the end of the last sprint.

# References

E. says: “Jest and Enzyme: Unit testing in React in 2021,” *LogRocket Blog*, 28-Dec-2020. [Online]. Available: https://blog.logrocket.com/jest-and-enzyme-unit-testing-in-react-in-2021/. [Accessed: 15-Feb-2021].

“Quality assurance in agile-scrum environment.” [Online]. Available: https://www.todaysoftmag.com/article/1355/quality-assurance-in-agile-scrum-environment. [Accessed: 9-Feb-2021].

*A Guide to Regression Testing and Automation via Selenium*. [Online]. Available: https://testsigma.com/regression-testing/selenium-regression-testing. [Accessed: 7-Feb-2021].

*A Guide to Making Regression Testing Agile Compatible*. [Online]. Available: https://testsigma.com/regression-testing/regression-testing-in-agile. [Accessed: 6-Feb-2021].

“What is Test Driven Development (TDD)? Tutorial with Example,” *Guru99*. [Online]. Available: https://www.guru99.com/test-driven-development.html#:~:text=Test%20Driven%20Development%20(TDD)%20is,what%20the%20code%20will%20do.&text=TDD%20instructs%20developers%20to%20write,an%20automated%20test%20has%20failed. [Accessed: 1-Feb-2021].

“Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE,” *Guru99*. [Online]. Available: https://www.guru99.com/unit-testing-guide.html#5. [Accessed: 29-Jan-2021].