

**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 Detailed Design Specifications (SDD)

|  |  |  |
| --- | --- | --- |
| 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](#_Toc64633650)

[Introduction 3](#_Toc64633651)

[Context & Goals 4](#_Toc64633652)

[Team’s Evaluation 4](#_Toc64633653)

[Proposed Solution 4](#_Toc64633654)

[Client Involvement 5](#_Toc64633655)

[Search Process 5](#_Toc64633656)

[Table of Requirements 5](#_Toc64633657)

[Table 1. Graphs 6](#_Toc64633658)

[Table 2. Graph Features 6](#_Toc64633659)

[Table 3. Webpage Features 6](#_Toc64633660)

[Table 4. Front-end Framework Learning Curve 7](#_Toc64633665)

[Table 5. Back-end Framework Learning Curve 7](#_Toc64633666)

[Table 6. Database Framework Learning Curve 7](#_Toc64633667)

[Table 7. Software Components 8](#_Toc64633668)

[Security 8](#_Toc64633669)

[Software Architecture 9](#_Toc64633670)

[Detailed Software Architecture 9](#_Toc64633671)

[Perspectives 9](#_Toc64633672)

[Software Components 10](#_Toc64633673)

[Software Connectors 11](#_Toc64633674)

# Introduction

In the following parts we will discuss the team’s evaluation on how we chose the reusable code for this project. Then, we will explain the client’s involvement in the decisions of the team’s evaluation with a brief explanation of the search process for every framework. After that, three (3) tables were made to specify the weight, complexity, and priority of every user story. Then, another three (3) tables to demonstrate how we ranked the frameworks by their learning curve, adaptability and included resources. After this, a diagram of the software’s architecture was made to simplify its complexity, and finally, a complete explanation in detail of every layer in the system.

# Context & Goals

* The outcome aimed for in this document, will be to detail the design and structure as simple and clear as possible. First, the ideal procedure would be to rate every user story with some specific characteristics, like weight and prioritization for example. After that, it would be explained the use of the chosen frameworks, why specific versions, and the applications for each one within the project. Then move unto the detail and specifications of the system’s architecture, this would be accomplished by understanding how the front-end will be working and addressing different tasks, as well as the back-end, and characterizing the database’s involvement with the back-end. Finally, an explanation of some additional connections to the project would be necessary for the desired outcome.

# Team’s Evaluation

* The selection of the reusable framework was mainly made by the learning curve of each software for each member. This evaluation consisted of selecting three different frameworks for each phase of the application (frontend, backend, and database) to be compared with and determine which framework was going to be implemented. A research was made in order to identify which framework would be better suited for our webpage.

# Proposed Solution

* The proposed systems of architecture in the meetings were reduced to the most efficient ones for this project, and these systems were the Layered Architecture and the Model-View-Controller. Basically, the layered architecture consists of a client-server connection in which the processing and data management are separated, this way changes can be made in one layer without affecting the others. On the other hand, the model-vie-controller pattern consists of dividing the program’s logic into those three logical components (model, view, controller), separating the internal representations of data, which also ends up being layer-like. But at the end, the team chose the layered architecture, as it applied and resembled more the idea that we had in mind.

# Client Involvement

* In this specific area, the client Danilo Perez has not been involved as much as the Professor Othoniel Rodriguez, which will be the acting client in this document. The client has suggested a couple of alternatives for re-usable software like libraries (Cucumber, jQuery, ReactJS) and frameworks (Bootstrap, AngularJS, WinJS), which have helped greatly when it came to making a decision in what we were going to use to develop the website. Not only that but when we bring ideas, the client helps us by constructively criticizing and sometimes suggesting other ways of producing the same results while being more efficient. We also received ideas on how to structure the architecture of the system, like using a database for the data from the APIs.

# Search Process

* There were three frameworks chosen (one per phase).
  + The three frameworks presented for the front end were React JS, Django, and Angular JS. In this case the React JS was selected because it was the framework with less learning curve for all the members.
  + The three presented for the back end were Node JS, PHP, and Django. Node JS was the framework selected for the website since it presented the lesser learning curve for all members.
  + While the three selected for the database were Firebase DB, MongoDB, and AWS. Firebase DB was the database selected for the website since it presented the lesser learning curve for all members.

# Table of Requirements

In the following tables, we will describe the User Stories’ weight, complexity, and priority for each one of the User Stories.

Weight: Based on how much time it would take a team member to complete them.

Complexity: Based on the difficulty level to develop and implement them.

Priority: Based on which User Stories are the ones that need to be completed first.

## Table 1. Graphs

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Weight | Complexity | Priority |
| As a user I want to view graphically a summary of the deaths, cases, and vaccinated people so that I can review the data | High | 6 | 95 |
| As a user I want to search total cases for a time period so that I can review the data | High | 4 | 85 |
| As a user I want to search municipality hospitalization data for a time period so that I can review the data | High | 4 | 75 |
| As a user I want to search municipality testing data for a time period so that I can review the data | High | 4 | 65 |
| As a user I want to search total deaths for a time period so that I can review the data | Medium | 2 | 55 |
| As a user I want to view a comparative graphic by age ranges and the total deaths by age group | Medium | 4 | 45 |
| As a user I want to view a comparative graphic by age ranges and the total positive cases by age group | Medium | 4 | 35 |
| 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 | Medium | 2 | 25 |
| As a user I want to view total vaccinated people by a time period so that I can review the data | Low | 2 | 15 |

Note: Weight: Low-Medium-High; Complexity: 1-10; Priority: 1-100

## Table 2. Graph Features

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Weight | Complexity | Priority |
| As a user I want to be able to download/embed a given graphic to another website so that I can share the data | High | 10 | 95 |
| As a user I want to change the selected graphic from xy scattered to a pie chart so that it can be visualized differently | High | 8 | 80 |
| As a user I want to change the zoom of a graphic so that a specific part can be seen more clearly | Medium | 4 | 65 |
| As a user I want to share the current state of a graphic so that I can notify friends and family | Medium | 2 | 50 |
| As a user I want to change the current graphic through a button so that I can view a different graph | Medium | 6 | 35 |
| 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 | Low | 2 | 20 |
| As a user I want to hover over a graphic so that I can see a specific data point | Low | 2 | 5 |

Note: Weight: Low-Medium-High; Complexity: 1-10; Priority: 1-100

## Table 3. Webpage Features

| Requirement | Weight | Complexity | Priority |
| --- | --- | --- | --- |
| 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 | High | 6 | 100 |
| As a user I want to verify the references of the data showed so that I can also verify it | Medium | 2 | 75 |
| As a user I want to know how frequently the graphics are being updated so that I know how accurate the data is currently | Low | 2 | 50 |
| As a user I want to report an error in the webpage so that it can be fixed | Low | 2 | 25 |

Note: Weight: Low-Medium-High; Complexity: 1-10; Priority: 1-100

# Ranking of Alternatives On deciding which framework will be used for the project. The team decided to compare each framework and rate them from one (1) to ten (10) on three (3) categories:

# Learning Curve: This is based on the understanding of each member of the team in relation to the framework that is being evaluated.

# Adaptability: This category was used to evaluate the ability to adjust each framework to our own needs and preferences.

# Resources: In this category the team did a research for each framework to determine which framework provides the best resourceful for our project.

## Table 4. Front-end Framework Learning Curve

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Front-end Framework | Learning Curve | Adaptability | Resources | Total |
| ReactJS v17.0.1 | 10 | 10 | 8 | **28** |
| AngularJS v1.8.2 | 8 | 8 | 6 | **22** |
| Django v3.0.12 | 2 | 4 | 8 | **14** |

## Table 5. Back-end Framework Learning Curve

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Back-end Framework | Learning Curve | Adaptability | Resource | Total |
| NodeJS v15.6.0 | 10 | 10 | 10 | **30** |
| Django v3.0.12 | 2 | 2 | 8 | **12** |
| PHP v8.0 | 2 | 4 | 4 | **10** |

## Table 6. Database Framework Learning Curve

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Database Framework | Learning Curve | Adaptability | Resource | Total |
| Firebase v26.5.0 | 10 | 10 | 10 | **30** |
| MongoDB v4.0 | 6 | 8 | 6 | **20** |
| AWS v2.1.24 | 2 | 8 | 8 | **18** |

## Table 7. Software Components

|  |  |
| --- | --- |
| Component | Usage |
| GitHub | As a team repository for the entire project. |
| Docker | As a container transfer of the project as an instance. |
| AWS (EC2) | As a virtual computer to run the server. |
| Apache | As a forwarder of the traffic of the IP. |

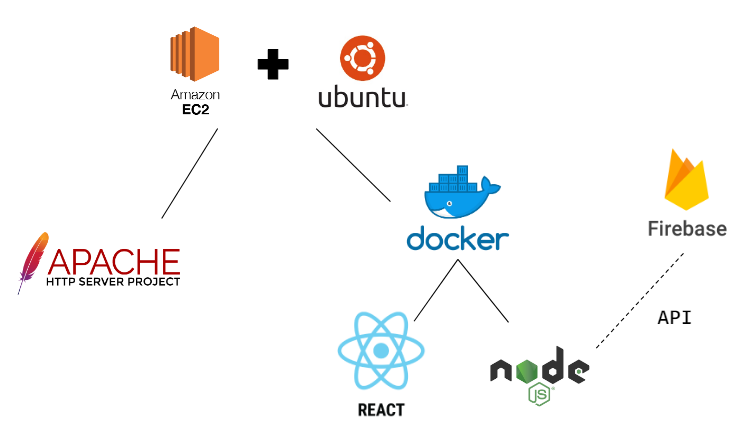
# Security

In order to protect the server and the webpage, each framework will search once every two (2) hours for new updates available. In case that an update was found, this will trigger an alert in the system that a new update was found. Once one of the developers get that alert, the team will proceed to make a research of the new update to evaluate if all the components the current framework has are compatible with the new update that was launched.

That research will conclude with a final discussion between the developers to conclude if the updates are suitable at this time for implementation on the project. If it is determined that the update is not suitable for the framework, the update will be rescheduled, until a new one comes out, or at least a patch. Then another research would take place and another discussion for no later than thirty (30) days.

# Software Architecture

* The system will consist of three (3) layers, the Presentation Layer (React), the Logic Layer (NodeJS) and a Data Access Layer (NodeJS), with some other Software Components which will be explained in the Detailed section.



# Detailed Software Architecture

Provide more than one perspective of the Software Architecture being adopted, detailing not only Software Components and their interconnection through Software Connectors, but also scenarios for deployment of software modules or alternatives.

# Perspectives

* Layered Architecture: With this type of pattern, we will have the Presentation Layer first, which will be where the UI (User Interface) components will be held. The UI components will be used in only one page, which will be the Home Page. Then, the Presentation Layer will be communicating to the Logic Layer to receive the data to be displayed in the graphics. In the Logic Layer it will be determined how the data can be created, stored, and changed, but validating everything, that way we can keep communicating constantly with the Data Access Layer to send and receive data for it to be used or saved for later. Finally, in the Data Access Layer it will be provided a simplified access to the API’s data stored in our database.
* Model-View-Controller Architecture: Another way of looking at the system, would be with this type of pattern, in which we will have the View, which will hold the UI components that will be used as Controllers through the user’s interactions. These Controllers will be communicating to the Model, letting it know what kind of data needs to be altered. Then, in the Model, it will be determined how the data will be created, stored, and changed, validating everything; but also having simplified access to the API’s data, stored in our database, this way we can communicate back to the View to alter the data for the user.

# Software Components

* Front-End: This is the phase of the software that will be presented to the user, where it will be able to interact with the website. The customization of the webpage will be in this phase along with the structure, as it will be presented to the user. Here the user will find a summary of statistics, all the graphs available and information about the graphs like where the data came from and when was the last time the graphs were updated. The user, for example, can select the graph to be displayed, interact with those graphs, and create a hyperlink to share or use in another place. This phase will be constantly communicating with the back-end to receive all the information that it needs to show. In the development part, the front-end will be in charge of handling some important tasks. We’ll start by creating the React app to start coding the project, and tackling the header, container, and footer components. After that, we will make three components, the summary, references and report components. But then moving to work on the colorblind button component and the pie chart, bar graph, comparation graph and line graph components. Finally, we have to consume the data from the database from within the components to present into the graphs.
* Back-End: In this phase all the logical functions (calculations and modifications of data) and other connections of the webpage will be coded, which only the developers will have access to. This phase will contain all the connections the webpage will need such as server connection, user connection and database connection. Because here we will be connecting to the database, this is the phase where we filter the data for the graphs to be utilized into them, so the parsing will be happening here. This is also the only phase that will be communicating with the other phases at the same time since it will need to pass the data it receives from the database to the front-end. In the development part, we will start by setting up the NodeJS to start working in the rest of the back-end and also configuring and linking the Firebase’s Cloud Firestore, the database that we will be using. Then move on to work with the API methods, the HTTP requests to the database and also filter the data from the database as mentioned before. Finally, we will be setting up a function to update the data from the APIs in a daily basis so that we do not have to do it manually, maing the page more self-sufficient.
* Database: This phase will have a copy of all the data extracted from the APIs. Firebase Cloud Firestore will be the database used for this phase. Aside the data from the APIs, this phase will also contain the connection to the back-end phase in order to send the data requested.

# Software Connectors

* GitHub: This platform will be used as a repository to save the code and implement/update the main project. This repository will be keeping a copy of all the versions for future reference so nothing would be really lost. Every member will have access to the project, but it would be required to never work on the master branch. To implement changes into the code, a branch will have to be made to test and run that piece of code. When that branch is tested and confirmed that it truly runs, then it will be pushed to the master branch to apply the changes.
* Docker: After having the project relatively ready, we will start to use this platform where we will be able to package the project into a container to upload into an AWS instance to be able to be hosted.
* AWS: With Amazon Web Services, we can utilize their cloud computing platform, specifically EC2, to make use of a virtual computer to run as our server. We would then move unto installing dockers first into the virtual machine and then place our package inside.
* Apache: With this open-source web server software, we will redirect the traffic to the IP of our AWS virtual machine to the port 80 for http. From here we can utilize some useful tools like for example data processing.