CHAPTER III

METHOD

Research Design

This research project will employ a Software Development Life Cycle (SDLC) model called Agile. The most basic SDLC model adopted is Waterfall for software and web development (Chandra, 2015; Kumar Pal, 2018). However, adopting the classical waterfall model in a real-world web application development project is impractical since it is idealistic and challenging to implement (Kumar Pal, 2018). Moreover, the sequential nature of the Waterfall SDLC made it unsuitable for this project. That is why the project development methodology will adopt another SDLC model called Agile Model. The said framework is different from the expected linear sequential life cycle of the Waterfall Model.

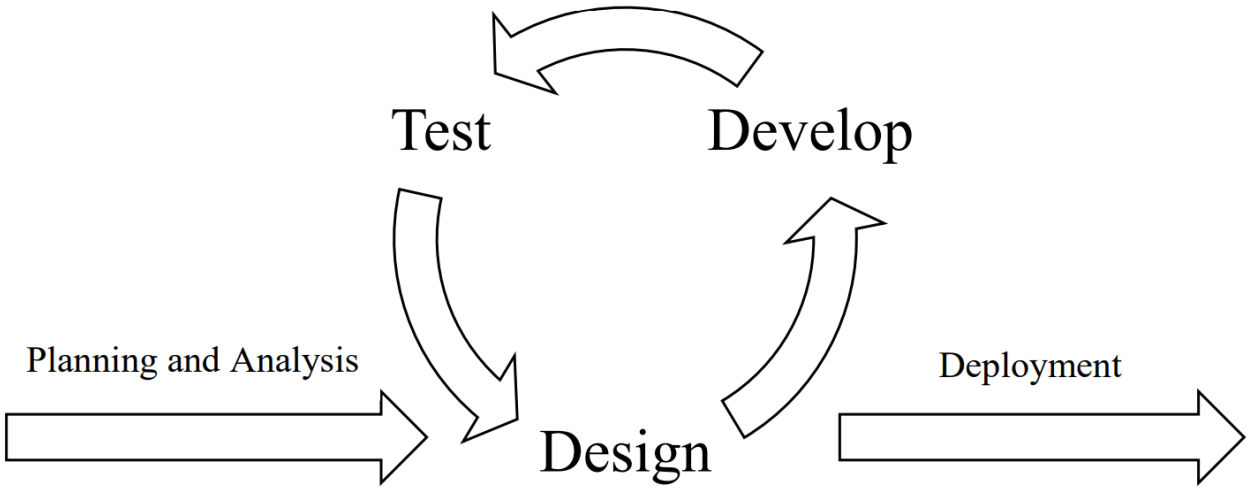

Figure 2. Agile Software Development Life Cycle Model

The primary purpose of the Agile Software Development model is to facilitate quick project completion adaptively. The salient nature of Agile SLDC will allow the researcher to adapt

to the unexpected circumstances in the development process due to its iterative and incremental nature (Figure 1). In other words, the researcher can make it up as the project goes along with the Agile Model. Whereas the Waterfall SLDC model, the researcher will structure everything before starting the project. However, with no adaptability due to its linear sequential flow, any erroneous prospects and consequences will be disregarded and not be rectified (Chandra, 2015). That is why the researcher will adopt the Agile Model since it is the most suitable SLDC model that allows the researcher to employ the advantages such as adaptability, efficiency, flexibility, incremental and continuous iteration, the high success rate with less time requirement, risk-reduction, and the elimination of cost (Dixit et al., 2020). Thus, the envisioned project stages will adopt the Agile Model.

Locale of the Study

The study will be conducted in the City of Digos, a capital of 8002 Davao Del Sur, Davao Region XI, Mindanao, Philippines. Furthermore, the locale of the deployment of the study will also be located in the City of Digos. The locale was purposefully chosen since the research problem as well as the research gap was uncovered in the said locale. Moreover, the target users for the deployment of COVID Pulse are the Digoseños. The word “Digoseños” is a plural form of a demonym word of the citizens and residents who live in the City of Digos located in Davao Del Sur, Philippines, called Digoseño. Additionally, the COVID Pulse project is also aimed to inscribe the Cor Jesu College, Inc. Institution.

Materials and Methods

The protocol defined for the study will be strictly adhered unto the Agile Software Development Life Cycle Model. This section will include the stages of the adopted model: Planning and Analysis, Design, Develop, Test, and Development.

Planning and Analysis Stage

This planning and analysis stage is the pre-development phase. The researcher will emphasize the procedures that assess the requirements of the COVID pulse web application development to satisfy the project objectives. One of the things that are initially done in this stage will be the Feasibility Analysis, wherein all the relevant factors such as the Technical, Economic, Legal, and Scheduling (TELOS) are considered (McLeod, 2021). Also, this stage consists of the consideration of the potentially conflicting needs and consequences in the development process. It also allows the researcher to document, assess, verify, and maintain the protocol of the development procedure.

Partial Feasibility Analysis. For the partial Feasibility Analysis in the pre-development phase, this project requires an overhaul of the web application and responsive web design for the development and setting up of the project. At present, the technical, operational, and scheduling feasibility will only be considered anchoring from the Technical, Economic, Legal, and Scheduling (TELOS) framework (McLeod, 2021). The economic, human, legal factor feasibility will be excluded from consideration since it is still equivocal or unnecessary considering the project is a straightforward web application. Furthermore, the project does not require any budget because most of the resources, apart from the physical tools and human resources needed, such as electricity and computer, are accessible and open-source. The researcher has partially determined the technical resources and applicability to the COVID Pulse development requirements.

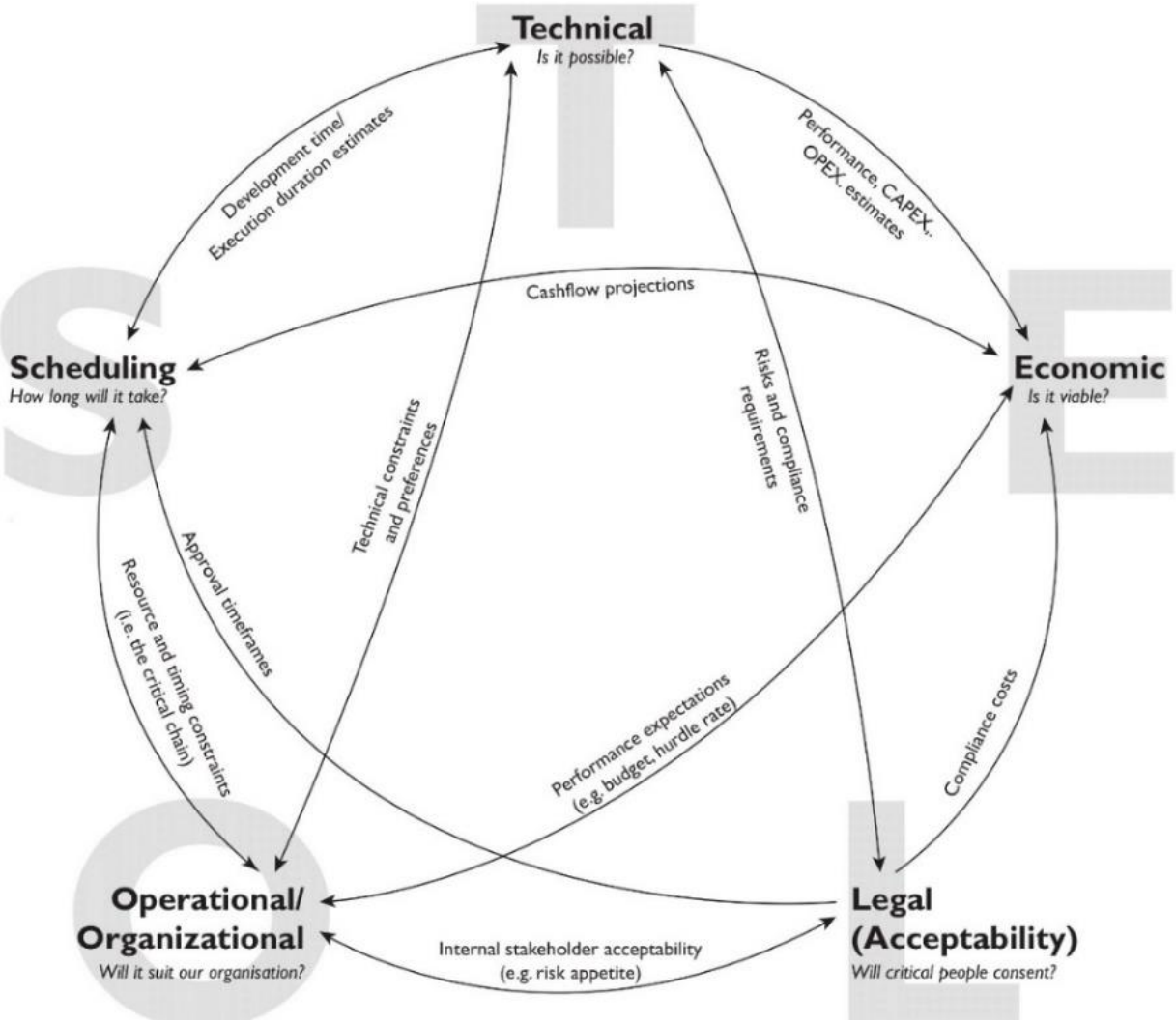

Figure 3. Visualizing the TELOS Framework by McLeod, 2021

In terms of the hardware requirements, besides two computers and an internet connection, the basic hardware requirements for the COVID Pulse project can be needless since the researcher will utilize the Google Firebase service to host the COVID pulse web application. However, the COVID Pulse project will still use another computer to test the web application. Hence, the researcher the hardware requirements to set up a server can be optional.

Additionally, the proposed project is technically possible and is an existing concept. The technology needed for the development is also available, and most of the tools are accessible and open-source. The researcher has a substantial background for some of the necessary technical requirements and has earned a Responsive Web Design certificate (Larson, 2021) for the knowledge and skills consideration. However, in some unforeseen circumstances where the requirements were beyond the researcher's aptitude, the researcher can adopt an alternative to fulfill the specific requirement. Thus, considering all of these, it is technically possible to develop the proposed project. From all the areas of TELOS, the project will be heavily scoped on the technical feasibility. Specifically, the following are the foreseen bare minimum software that is essential: Integrated Development Environment (IDE), Prototyping Tool and Graphics Editor, and Web Browser

Development Stage

The researcher has defined the development phases in every part of the COVID Pulse web application. Every stages involves the development of the specific feature of COVID pulse through the implementation and coding of the designed project. In other words, this is the primary stage in the realization of the COVID Pulse web application design and translating it into a source code. Each module that will be designed in the designing stage by the researcher will be implemented and coded. After the development stage, the researcher will test the module functionality and determine whether it is appropriately working through end-to-end (E2E) testing. Each development phase will focus on the three segments: Frontend, API (Middleware), and Backend.

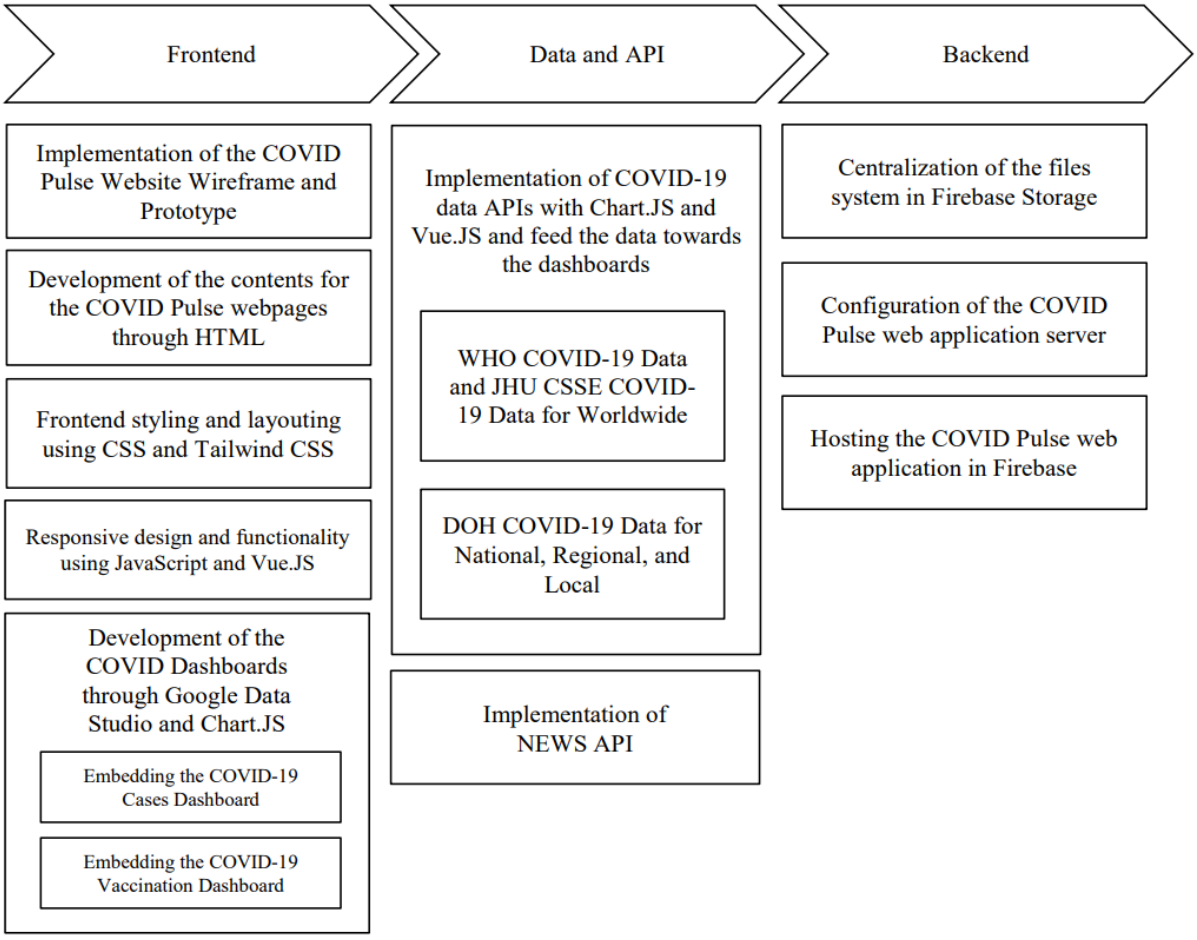

Figure 8. Simplification of the Envisioned Development Phases

System Flow

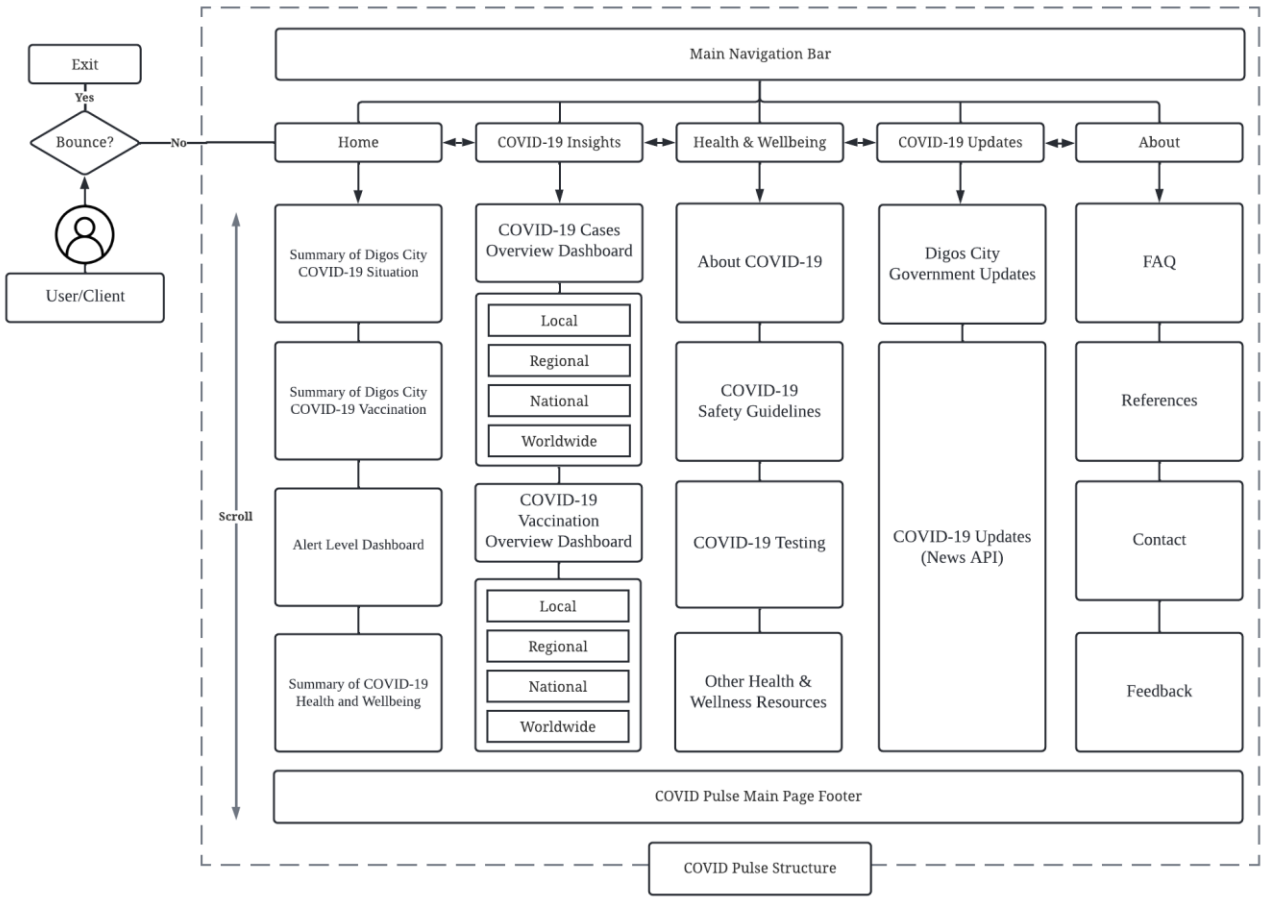

Figure 4. COVID Pulse Website Application Sitemap Structure

Home - The home webpage will act as the website's beginning point. The default page loads when the target users, such as the Digoseños, visit the COVID Pulse website.

COVID-19 Insights - This webpage will contain the main objective of this project. The elaborated COVID-19 dashboard that visualizes the COVID-19 Cases and Vaccination per segment will be embedded in the said webpage.

Health & Wellbeing - This webpage will contain information about the SARS-CoV-2 virus, COVID-19 safety guidelines, COVID-19 testing, and other health-related resources.

COVID-19 Updates - This webpage will contain the essential updates from the Digos City government, such as the Alert Level ordinance, and will also contain the COVID-19 related news articles for the Digoseños to be constantly updated and informed.

About - The purpose of this web page is to inform the web application visitors about the COVID Pulse's details and the web application's critical operations.

System Design

The designing stage is crucial for the development of the COVID Pulse. In this stage, the researcher will identify and describe the web application's features, operation, and specification to establish the intended objectives. The system designing of the COVID pulse will consist of various design considerations and concepts. Additionally, it conceptualizes and offers good visual and descriptive prospects about the web application and its system aspects to allow the final version to be consistent with design structures as described initially in the proposed system architecture models. Hence, this stage is necessary since it will allow the researcher to implement and code the devised and analyzed prospects in the Planning and Analysis phase through a programming language. Lastly, the web application system design of COVID Pulse will be divided into three segments: Frontend, Backend, and APIs.

Frontend Prototyping. The initial but most crucial phase of the design stage of the development process is prototyping the COVID Pulse web application. The researcher will be able to ideate the reference for the end product, which allows the development process to be convenient and reduce the cost and time as it provides a comprehensive high-level reference and overview of the final output. Furthermore, the prototyping phase will allow the researcher to make quick necessary modifications and be flexible with the User Interface and User Experience

design. Through wire framing, the prototyping will be done through Low-Fidelity and Hi-Fidelity prototypes (Figures 4 and 5). Although sketching is often part of the prototyping procedure, it was not included since it is deemed unnecessary.

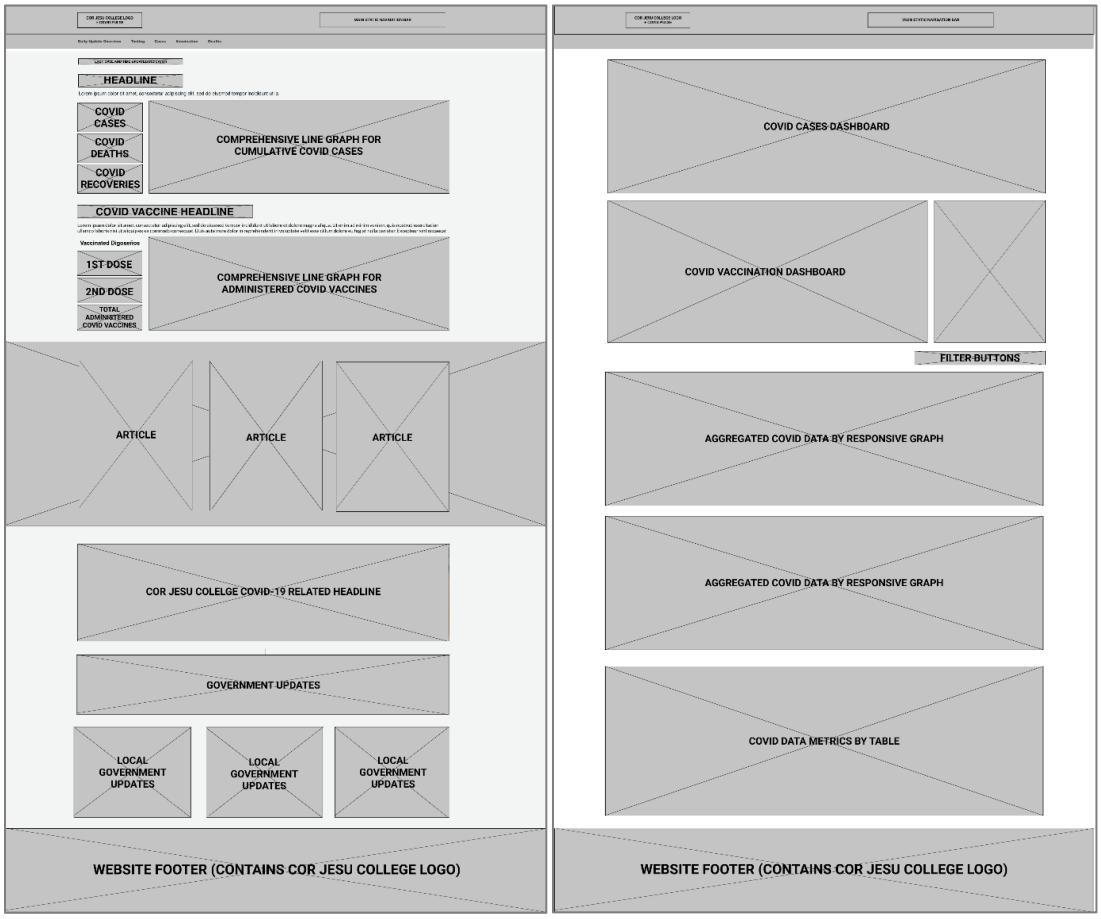

Figure 5. Low-Fidelity Sample of Provisional Wireframe: COVID Pulse Home [First Section] and COVID-19 Insights Webpages [Second Section]

Low-fidelity (Lo-Fi) prototyping is essential for the researcher to quickly conceptualize the COVID pulse's design features. However, the appearance design will be disregarded in low-fidelity prototyping (Figure 4). This type of prototyping technique will allow the researcher to convert the high-level design concept of the web application into a testable functional prototype and know each purpose of the components. Then, the low-fidelity prototype will be anchored to

the next prototyping stage, which is the high-fidelity prototyping technique. See Figure 6 for a hi-fidelity sample of COVID Pulse.

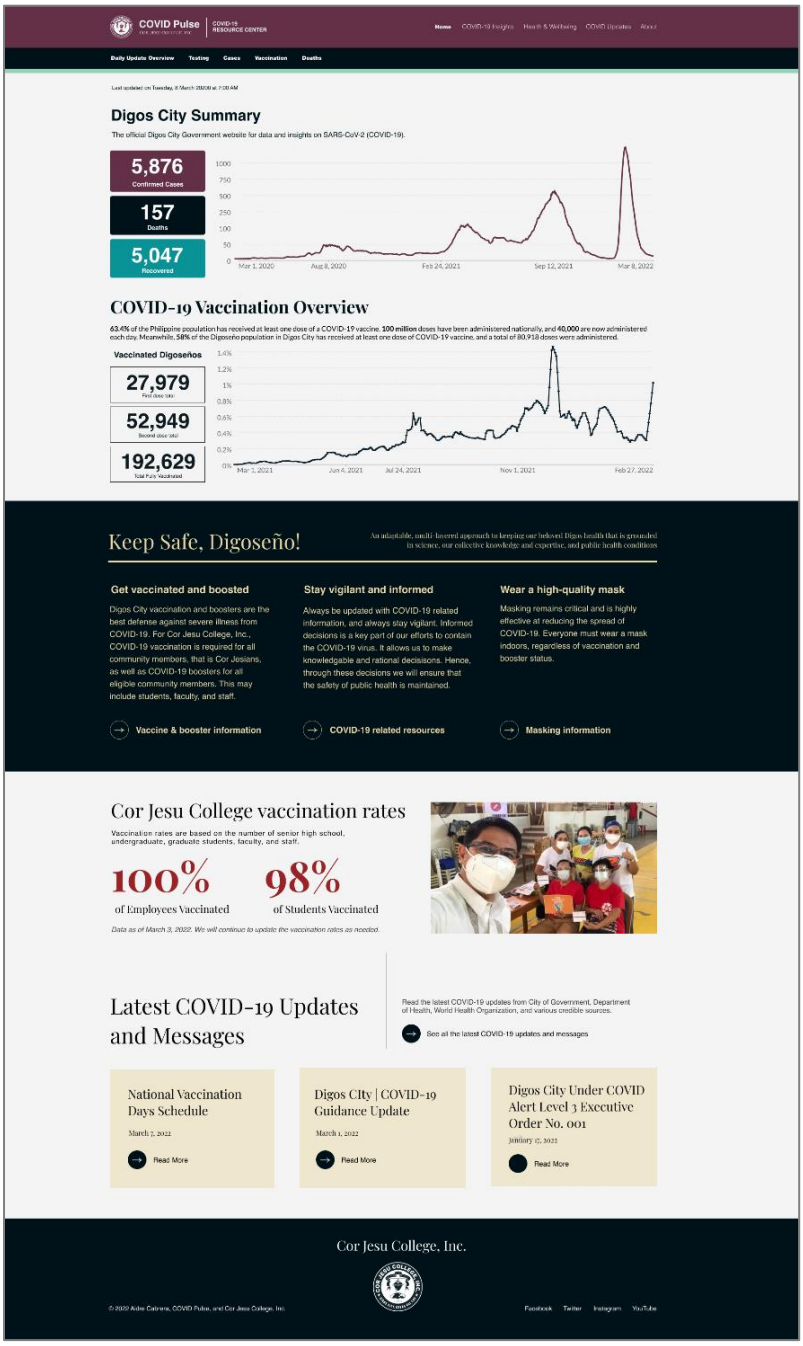

Figure 7. Provisional Sample of the COVID Pulse Home Page (1920 x 3224)

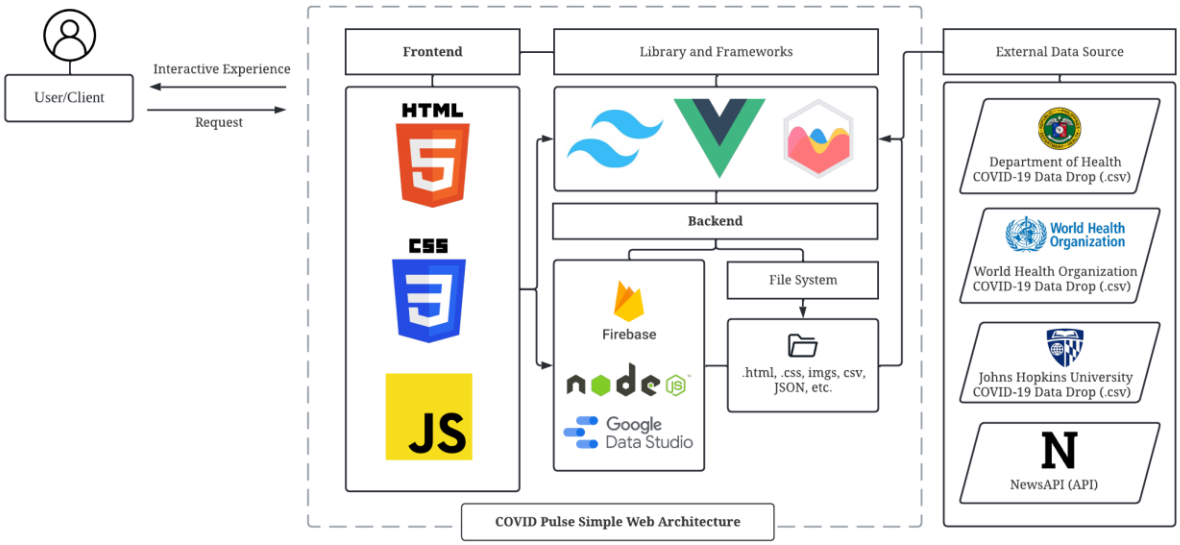

Figure 9. Envisioned COVID Pulse designed web architecture, which consists of Frontend, Backend, and APIs

Frontend and Backend Layer. The outer layer of the web application project that the Digoseños (Users) see and interact with is the Frontend, also known as the client-side. Specifically, it is the visual elements such as the User Interface (UI) and User Experience (UX) designing of COVID Pulse. The backend layer is scoped on the server-side of the web application, in which the primary purpose is to make sure everything of the web application is functional. Also, it is the part where the clients of the COVID Pulse will not interact and cannot be interacted with users.

The frontend layer will consist of the languages that are the fundamental pillars for Web Development: HTML, CSS, and JavaScript. The researcher will adopt JavaScript and other frameworks and libraries for the backend. Additionally, the researcher will implement frameworks and libraries such as Tailwind CSS, Vue.js, and Chart.js during the COVID Pulse web application development.

Hypertext Markup Language (HTML). It is the fundamental building block of the webpage since it allows the researcher to define the structure and modules of web content. Therefore, HTML is crucial for the COVID Pulse web application since it will contain the websites' basic text and hypertext contents. However, HTML is always a tandem with CSS.

Cascading Style Sheets (CSS). CSS will always coincide with HTML for the researcher to style and specify how the presentation of the User Interface will look (i.e., colors, fonts, and layout) and feel and how the primary contents of the COVID Pulse web application are presented to Digoseños.

JavaScript. This programming language is a dynamic client-side scripting that will allow the researcher to make the COVID Pulse web application include more functionality, responsiveness, and dynamic features. Although sufficient, developing the frontend design with Vanilla JavaScript and Vanilla CSS, plain JavaScript or CSS without any libraries or frameworks, is arduous, time-consuming, inefficient, and risky (Delaney, 2021). For instance, for Vanilla CSS, the semantic class names make maintaining it laborious and time-consuming (Hovhannisyan, 2021). This decision will also abide by one of the primary system design principles that will be adopted in the system design. That is, it "should not reinvent the wheel" (Davis, 1995). Hence, it is necessary to implement libraries and frameworks.

Tailwind. It is a utility-first framework of CSS that is parceled with classes, enabling faster development of the frontend layer. Other than the time-saving procedure of Tailwind during the development process, it also provides other benefits such as symmetrical layouts, high productivity, and efficiency of the development of the COVID Pulse web application.

Vue.js. It is a progressive JavaScript framework used to build web interfaces and one-page applications. In other words, it will allow the researcher to save time during the development since it is a progressive JavaScript framework that allows the process to be smooth and easy with a shallow learning curve. Furthermore, it is chosen since it is a suitable lightweight, flexible, modular, and highly performant framework.

Chart.js. It is an open-source JavaScript data visualization library that will be adopted. Unlike the other leading data visualization library such as D3.js, Chart.js is straightforward, requires less effort, and sufficed the bare minimum requirement of generating data graphics to develop the COVID-19 dashboard. Furthermore, it will also be paired with Google Data Studio as a complementary for converting the COVID-19 data into reports.

Data and APIs

The data dashboard will acquire the COVID-19 aggregated epidemiological data from various APIs in terms of Local, Regional, National, and Worldwide. Specifically, the data source will be from the following repositories:

JHU CSSE COVID-19 Data. It is a COVID-19 data repository collected, provided, and operated by the Center for Systems Science and Engineering from Johns Hopkins University. It is publicly available for everyone to be accessed from the GitHub JHU CSSE repository.

WHO COVID-19 Data. It is the official COVID-19 data source aggregated by the World Health Organization and is distributed by comma-separated values (CSV) files.

DOH COVID-19 Data. The official COVID-19 data source aggregated by the Department of Health can be accessed through the DOH Data Drop.

Software Configuration

The dependencies and devDependencies that will be implemented in the COVID Pulse project can be found in the `package.json` file found in the repository. It contains the metadata relevant to the COVID Pulse project repository. It will be used for managing the COVID Pulse project's dependencies, devDependencies, scripts, and version. Note that there are some packages such as `day.js and axios` that were installed during the pre-development and configuration of the project requirements. The following is the tentative configuration of the project:

```
{
  "name": "coronaviruspulse-app",
  "version": "0.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview --port 5050",
    "test:e2e": "start-server-and-test preview http://127.0.0.1:5050/
'cypress open'",
    "test:e2e:ci": "start-server-and-test preview
http://127.0.0.1:5050/ 'cypress run'",
    "test:unit": "cypress open-ct",
    "test:unit:ci": "cypress run-ct --quiet --reporter spec"
  },
  "dependencies": {
    "axios": "^0.26.1",
    "bootstrap": "^5.1.3",
    "chart.js": "^3.7.1",
    "dayjs": "^1.11.1",
```

```
    "jquery": "^3.6.0",
    "newsapi": "^2.4.1",
    "smooth-scrollbar": "^8.7.4",
    "title-case": "^3.0.3",
    "vue": "^3.2.33",
    "vue-chartjs": "^4.0.7",
    "vue-number-animation": "^1.1.2",
    "vue-router": "^4.0.14"
  },
  "devDependencies": {
    "@cypress/vite-dev-server": "^2.2.2",
    "@cypress/vue": "^3.1.1",
    "@tailwindcss/typography": "^0.5.2",
    "@vitejs/plugin-vue": "^2.3.1",
    "autoprefixer": "^10.4.5",
    "cypress": "^9.6.0",
    "postcss": "^8.4.12",
    "start-server-and-test": "^1.14.0",
    "tailwindcss": "^3.0.24",
    "vite": "^2.9.6"
  }
}
```

Node.js Installation – As of writing, the Node.js version that will be used in the development of the COVID Pulse will be version 16.14.2. The installation of Node.js and Node Package Manager is straightforward. The Windows Installer (.msi) node-v[version-here]-x64.msi was downloaded from the official Node.js website.

1. Downloading the Node.js Installer for Windows

2. Running the Node.js installation

3. Installation will include Chocolatey

4. Restarting the System

npm. It is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. Note that the Node.js installation procedure comes with the Node Package Manager (npm). However, for the COVID Pulse project, the researcher further installed the alternative of npm, which is pnpm.

The package manager such as pnpm is essential in the setup and development process of the COVID Pulse project since it allows the researcher to install, update, and remove the packages that is involved during the development process. pnpm will be mainly used since it has major advantages compared to the default package manager of Node.js.

Installation of pnpm via npm:

- cd ./aidrecabrera/cjc-coronavirus-pulse
- npm install -g pnpm

Vue.js. The main JavaScript framework that will be utilized to build the COVID Pulse frontend user interface. Vue.js is anchored in the standard HTML, CSS, and Javascript while allowing the resesarcher to develop the project while integrating the declarative and reactive nature of the framework. Furthermore, it is a simple framework unlike Angular.js and React.js while allowing the COVID Project to progress and scale up overtime. Therefore, it is a progressive and save time during the development since it is a progressive JavaScript framework that allows the

process to be smooth and easy with a shallow learning curve. Furthermore, it is chosen since it is a suitable lightweight, flexible, modular, and highly performant framework.

Vue.js installation via pnpm:

- `cd ./aidrecabrera/cjc-coronavirus-pulse`
- `pnpm init vue@latest`
- `(The command will execute create-vue)`

Configuration of COVID Pulse project through Vue project scaffolding tool:

✓ Project name: …

- `cjc-coronavirus-pulse`

✓ Add TypeScript? …

- `No`

✓ Add JSX Support? …

- `No`

✓ Add Vue Router for Single Page Application development? …

- `Yes`

✓ Add Pinia for state management? …

- `No`

✓ Add Vitest for Unit testing? …

- `No`

✓ Add Cypress for both Unit and End-to-End testing? …

- `Yes`

✓ Add ESLint for code quality? …

- `Yes`

✓ Add Prettier for code formatting? …

- Yes

```
Scaffolding project in ./< cjc-coronavirus-pulse>...
Done.
```

- cd ./cjc-coronavirus-pulse

Tailwind. It is a utility-first framework of CSS that is parceled with classes, enabling faster development of the frontend layer. Other than the time-saving procedure of Tailwind during the development process, it also provides other benefits such as symmetrical layouts, high productivity, and efficiency of the development of the COVID Pulse web application.

Tailwind CSS installation via pnpm:

- cd ./aidrecabrera/cjc-coronavirus-pulse
- pnpm install -D tailwindcss
- pnpx tailwindcss init

Chart.js. It is an open-source JavaScript data visualization library that will be adopted. Unlike the other leading data visualization library such as D3.js, Chart.js is straightforward, requires less effort, and sufficed the bare minimum requirement of generating data graphics to develop the COVID-19 dashboard. Furthermore, it will also be paired with Google Data Studio as a complementary for converting the COVID-19 data into reports.

Chart.js and vue-chartjs wrapper installation via pnpm:

- cd ./aidrecabrera/cjc-coronavirus-pulse
- npm install vue-chartjs chart.js

Integration of Chart.js to Vue integration:

- `import Chart from 'chart.js/auto'`

Firebase. It is a platform develop by Google that provides development services. The researcher will mainly utilize Firebase services such as Firebase Hosting and to build and monitor the COVID Pulse web application.

Installation of Firebase via pnpm:

```
$ pnpm install firebase-tools
```

```
$ pnpm install firebase
```

Configuration of Firebase for COVID Pulse project:

```
$ firebase login
```

```
$ firebase init
```

Deployment of COVID Pulse project in Firebase Hosting service:

```
$ firebase deploy
```

Cypress. It is a frontend end-to-end testing tool. This framework will be used for the testing stage of the COVID Pulse application. Cypress was picked as the testing tool for the project since it has major advantages when it comes to simplicity and acquiring quick, consistent, and reliable testing process. Lastly, it allows the researcher to execute automated web application debugging and testing of COVID Pulse project.

Installation of Cypress through pnpm:

- `cd ./aidrecabrera/cjc-coronavirus-pulse`
- `pnpm install cypress --save-dev`
- `pnpm i -D @cypress/webpack-dev-server @cypress/vue`

List of installed dependencies and devDependencies:

```
coronaviruspulse-app@0.0.0 T:\coronaviruspulse\coronaviruspulse-app
      ├── @cypress/vite-dev-server@2.2.2 ->
      .\node_modules\.pnpm\@cypress+vite-dev-
      server@2.2.2_vite@2.9.6\node_modules\@cypress\vite-dev-server
      ├── @cypress/vue@3.1.1 ->
      .\node_modules\.pnpm\@cypress+vue@3.1.1_cypress@9.6.0+vue@3.2.33\node_m
      odules\@cypress\vue
      ├── @firebase/analytics-types@0.7.0 extraneous ->
      .\node_modules\.pnpm\@firebase+analytics-
      types@0.7.0\node_modules\@firebase\analytics-types
      ├── @firebase/app-check-interop-types@0.1.0 extraneous ->
      .\node_modules\.pnpm\@firebase+app-check-interop-
      types@0.1.0\node_modules\@firebase\app-check-interop-types
      ├── @firebase/app-check-types@0.4.0 extraneous ->
      .\node_modules\.pnpm\@firebase+app-check-
      types@0.4.0\node_modules\@firebase\app-check-types
      ├── @firebase/app-types@0.7.0 extraneous ->
      .\node_modules\.pnpm\@firebase+app-
      types@0.7.0\node_modules\@firebase\app-types
      ├── @firebase/auth-interop-types@0.1.6 extraneous ->
      .\node_modules\.pnpm\@firebase+auth-interop-
      types@0.1.6_a64185e619fe35b9275fc3c76f6268db\node_modules\@firebase\aut
      h-interop-types
      ├── @firebase/auth-types@0.11.0 extraneous ->
      .\node_modules\.pnpm\@firebase+auth-
      types@0.11.0_a64185e619fe35b9275fc3c76f6268db\node_modules\@firebase\au
      th-types
      ├── @firebase/database-types@0.9.7 extraneous ->
```

.\node_modules\.pnpm\@firebase+database-types@0.9.7\node_modules\@firebase\database-types
├── @firebase/firestore-types@2.5.0 extraneous -> .\node_modules\.pnpm\@firebase+firestore-types@2.5.0_a64185e619fe35b9275fc3c76f6268db\node_modules\@firebase\firestore-types
├── @firebase/functions-types@0.5.0 extraneous -> .\node_modules\.pnpm\@firebase+functions-types@0.5.0\node_modules\@firebase\functions-types
├── @firebase/messaging-interop-types@0.1.0 extraneous -> .\node_modules\.pnpm\@firebase+messaging-interop-types@0.1.0\node_modules\@firebase\messaging-interop-types
├── @firebase/performance-types@0.1.0 extraneous -> .\node_modules\.pnpm\@firebase+performance-types@0.1.0\node_modules\@firebase\performance-types
├── @firebase/remote-config-types@0.2.0 extraneous -> .\node_modules\.pnpm\@firebase+remote-config-types@0.2.0\node_modules\@firebase\remote-config-types
├── @firebase/storage-types@0.6.0 extraneous -> .\node_modules\.pnpm\@firebase+storage-types@0.6.0_a64185e619fe35b9275fc3c76f6268db\node_modules\@firebase\storage-types
├── @tailwindcss/typography@0.5.2 -> .\node_modules\.pnpm\@tailwindcss+typography@0.5.2_tailwindcss@3.0.24\node_modules\@tailwindcss\typography
├── @types/duplexify@3.6.1 extraneous -> .\node_modules\.pnpm\@types+duplexify@3.6.1\node_modules\@types\duplexify
├── @types/long@4.0.2 extraneous ->

```
.\node_modules\.pnpm\@types+long@4.0.2\node_modules\@types\long
├── @types/sinonjs__fake-timers@8.1.1 extraneous ->
.\node_modules\.pnpm\@types+sinonjs__fake-
timers@8.1.1\node_modules\@types\sinonjs__fake-timers
├── @types/sizzle@2.3.3 extraneous ->
.\node_modules\.pnpm\@types+sizzle@2.3.3\node_modules\@types\sizzle
├── @types/yauzl@2.10.0 extraneous ->
.\node_modules\.pnpm\@types+yauzl@2.10.0\node_modules\@types\yauzl
├── @vitejs/plugin-vue@2.3.1 -> .\node_modules\.pnpm\@vitejs+plugin-
vue@2.3.1_vite@2.9.6+vue@3.2.33\node_modules\@vitejs\plugin-vue
├── ast-types@0.13.4 extraneous -> .\node_modules\.pnpm\ast-
types@0.13.4\node_modules\ast-types
├── autoprefixer@10.4.5 ->
.\node_modules\.pnpm\autoprefixer@10.4.5_postcss@8.4.12\node_modules\au
toprefixer
├── axios@0.26.1 ->
.\node_modules\.pnpm\axios@0.26.1\node_modules\axios
├── bootstrap@5.1.3 ->
.\node_modules\.pnpm\bootstrap@5.1.3\node_modules\bootstrap
├── chart.js@3.7.1 ->
.\node_modules\.pnpm\chart.js@3.7.1\node_modules\chart.js
├── check-more-types@2.24.0 extraneous -> .\node_modules\.pnpm\check-
more-types@2.24.0\node_modules\check-more-types
├── cypress@9.6.0 ->
.\node_modules\.pnpm\cypress@9.6.0\node_modules\cypress
├── dayjs@1.11.1 ->
.\node_modules\.pnpm\dayjs@1.11.1\node_modules\dayjs
├── firebase-tools@10.7.2 -> .\node_modules\.pnpm\firebase-
tools@10.7.2\node_modules\firebase-tools
```

```
├── firebase@9.7.0 ->
.\node_modules\.pnpm\firebase@9.7.0\node_modules\firebase
├── jquery@3.6.0 ->
.\node_modules\.pnpm\jquery@3.6.0\node_modules\jquery
├── lodash._objecttypes@2.4.1 extraneous ->
.\node_modules\.pnpm\lodash._objecttypes@2.4.1\node_modules\lodash._obj
ecttypes
├── newsapi@2.4.1 ->
.\node_modules\.pnpm\newsapi@2.4.1\node_modules\newsapi
├── postcss@8.4.12 ->
.\node_modules\.pnpm\postcss@8.4.12\node_modules\postcss
├── smooth-scrollbar@8.7.4 -> .\node_modules\.pnpm\smooth-
scrollbar@8.7.4\node_modules\smooth-scrollbar
├── start-server-and-test@1.14.0 -> .\node_modules\.pnpm\start-server-
and-test@1.14.0\node_modules\start-server-and-test
├── tailwindcss@3.0.24 ->
.\node_modules\.pnpm\tailwindcss@3.0.24\node_modules\tailwindcss
├── title-case@3.0.3 -> .\node_modules\.pnpm\title-
case@3.0.3\node_modules\title-case
├── vite@2.9.6 -> .\node_modules\.pnpm\vite@2.9.6\node_modules\vite
├── vue-chartjs@4.0.7 -> .\node_modules\.pnpm\vue-
chartjs@4.0.7_chart.js@3.7.1+vue@3.2.33\node_modules\vue-chartjs
├── vue-number-animation@1.1.2 -> .\node_modules\.pnpm\vue-number-
animation@1.1.2\node_modules\vue-number-animation
├── vue-router@4.0.14 -> .\node_modules\.pnpm\vue-
router@4.0.14_vue@3.2.33\node_modules\vue-router
└── vue@3.2.33 -> .\node_modules\.pnpm\vue@3.2.33\node_modules\vue
```

dependencies:

```
axios 0.26.1                dayjs 1.11.1                jquery 3.6.0
title-case 3.0.3            vue-number-animation 1.1.2
bootstrap 5.1.3             firebase 9.7.0              newsapi 2.4.1
vue 3.2.33                  vue-router 4.0.14
chart.js 3.7.1              firebase-tools 10.7.2       vue-chartjs 4.0.7
```

devDependencies:

```
@cypress/vite-dev-server 2.2.2     @tailwindcss/typography 0.5.2
cypress 9.6.0                      tailwindcss 3.0.24
@cypress/vue 3.1.1                 @vitejs/plugin-vue 2.3.1
postcss 8.4.12                     vite 2.9.6
@cypress/webpack-dev-server 1.8.4  autoprefixer 10.4.5
start-server-and-test 1.14.0
```

Hardware Specification for Development:

The hardware specification for the computer that will be used in the development of COVID Pulse web application are of the following:

Computer Hardware Specification for Development:

- Processor: AMD Ryzen 3 4300GE with Radeon Graphics 3.50 GHz
- System Type: 64-bit Operating System, x64-based processor
- Installed Memory (RAM): 16.0 GB (2x8 DDR4-3200MHz)

Computer Hardware Specification for Testing:

- Processor: Intel® Core(TM) i5-4460 CPU with Intel® HD Graphics 4600 3.20GHz
- System Type: 64-bit Operating System, x64-based processor
- Installed Memory (RAM): 4.0 GB (1x4 DDR3-1600MHz)

Software Specification for Development:

The software specification for the computer that will be used during the development of COVID Pulse web application are of the following:

- Windows 10 Professional Edition
- Visual Studio Code (Version 1.66.2, Latest release as of Writing)
- Visual Studio Code Extensions (For development speed):
    - Volar (Version 0.34.11)
    - Tailwind CSS IntelliSense (Version 0.8.3)
    - Auto Close Tag (Version 0.5.14)
    - Auto Rename Tag (Version 0.1.10)
    - ESLint (Version 2.2.2)
    - HTML CSS Support (Version 1.11.0)
- Mozilla Firefox (For CSS-oriented development and testing)
- Microsoft Edge (For main browser for local preview and testing)
    - JSON Formatter Extension
- Adobe Photoshop 2021
- Figma (Latest version as of writing)
- Firebase Hosting Service (Spark plan):
    - Downloads (Bandwidth): 10GB/month
    - Storage: 10GB

Web Application Testing and Simulation

The testing stage is part of the development process. The project will adopt a type of functional test called End-to-End (E2E) testing, specifically an automated Horizontal E2E. This type of testing involves testing the entire software, or in this case, website web application, from start to end and will coincide with the user flow (Hamilton, 2019). Its primary goal is to test the developed project to validate if everything from all the integrated units is behaving as expected (Figure 4). E2E testing is mainly done from the aspect of the end-user by simulating actual real-world user experience and verifying the entire system.

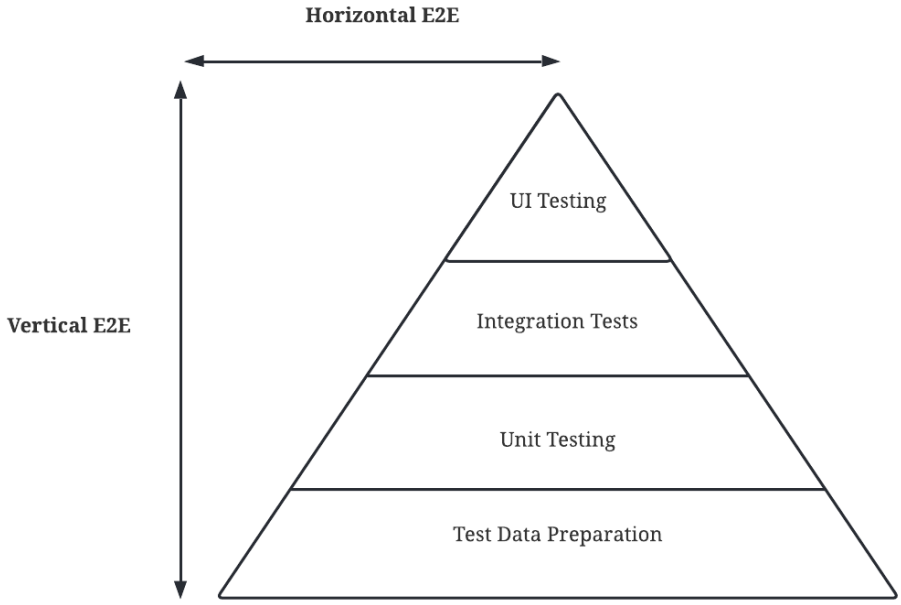

Figure 10. E2E Test Automation Horizontal & Vertical Scale

The vertical E2E test breaks down the COVID Pulse structure into various segments. Each segment ssssof the web application will be individually tested and analyzed in a hierarchical order (Figure 4). So, unlike the vertical testing, the testing phase of the project COVID Pulse web application does not involve breaking down the codes into individual units before testing it. There are many benefits of E2E testing, such as wide testing coverage, ensuring system consistency,

reducing time and cost, and detecting errors and bugs during the development phase (Rajkumar, 2022). E2E testing will also allow the researcher to verify the web application flow, which will alleviate the project's potential risks. Therefore, the researcher will adopt the Horizontal E2E testing during the iteration and incrementation of the development process (Figure 9) since it is an efficient, suitable, and reliable testing methodology.

Data Analysis

Content