Wastewater Modeling

Ayaan Sunesara
Quinlon Horndasch
Matthew Delorenzo
Grace Salau

CONCEPT OF OPERATIONS

REVISION – 1
September 17,2022

Concept of Operations
for
Wastewater Modeling

Team <15>

Approved by:

Project Leader	Date
Prof. Kalafatis	Date
T/A	Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	[09/15]	[WW team]		Draft Release
Rev.	**Date**	**Originator**	**Approvals**	**Description**
1	[10/3]	[WW team]		Revision 1 release – updated Conops

Table of Contents

Table of Contents III
List of Tables V
List of Figures VI
Figure 1. System Overview VI
1. Executive Summary 1
2. Introduction 2
2.1. Background 2
2.2. Overview 2
2.3. Referenced Documents and Standards 3
3. Operating Concept 4
3.1. Scope 4
3.2. Operational Description and Constraints 4
3.3. System Description 4
3.4. Modes of Operations 5
3.5. Users 6
3.6. Support 6
4. Scenario(s) 7
4.1. LANL Application 7
4.2. Public Application 7
4.3. Expansion of Tracked items 7
5. Analysis 7
5.1. Summary of Proposed Improvements 7
5.2. Disadvantages and Limitations 7
5.3. Alternatives 8
5.4. Impact 8
6. Functional System Requirements - Introduction 12
6.1. Purpose and Scope 12
6.2. Responsibility and Change Authority 13
7. Applicable and Reference Documents 14
7.1. Applicable Documents 14
7.2. Reference Documents 14
7.3. Order of Precedence 15
8. Requirements 16
8.1. System Definition 16
8.2. Characteristics 17
8.2.1. Functional / Performance Requirements 17
8.2.2. Software Requirements 18

8.2.3. Interface Requirements 20
Appendix A: Acronyms and Abbreviations 21
Appendix B: Definition of Terms 21
9. Interface Control Document - Overview 25
10. References and Definitions 26
10.1. References 26
10.2. Definitions 26
11. Server ↔ Data Processing Interfacing 27
12. Data Processing ↔ Predictive Algorithm Interfacing 28
13. Predictive Algorithm ↔ Flask Interfacing 29
14. Flask ↔ UI Interfacing 30
15. User Interface 30
16. Validation Plan 32
17. Execution Plan 37

List of Tables

Table 1. Reference documents

List of Figures

Figure 1. System Overview

1. Executive Summary

Los Alamos National Laboratory (LANL) has requested a browser-based application to track COVID-19 cases through wastewater testing. Currently, LANL (along with most other major government agencies) uses reported COVID-19 cases as an indicator of which areas are at risk or currently hotspots for the virus. Unfortunately, this form of tracking COVID-19 has limitations due to inconsistency of reporting, asymptomatic cases, and the long time between testing and results. However, with new strategies like wastewater research, COVID-19 data can be gathered earlier and more accurately. To take advantage of this strategy, the wastewater modeling application must allow for the input, analysis, and storage of data in the cloud, while also organizing the data for easy viewing. This will allow LANL management to make more informed decisions with regards to dealing with and preventing further spread of the virus. The application will satisfy the growing need for a more accurate testing option while creating an opportunity to counter the virus in the early stages rather than react to the aftermath.

2. Introduction

This document is an introduction to wastewater modeling, using the virus levels in a county's wastewater as a predictive measure to the correlating reported cases of the virus. In this instance, the virus in focus is SARS-CoV-2 (COVID-19) but this can be outreached to deal with other viruses, such as the flu. COVID-19 wastewater modeling will serve as a measure for LANL management to take precautionary measures for their company along with their families if the trend between COVID-19 wastewater levels correlates with the number of reported cases.

2.1. Background

COVID-19 is a virus that stopped the world in its tracks by onsetting a worldwide pandemic that had millions of people on lockdown in the United States alone until a vaccine was developed. During this period of panic researchers and scientists developed and implemented various tools to help control as well as to understand the virus at hand - wastewater modeling was one of these measures. Individuals that are infected with COVID-19 shed the virus through their feces before symptoms are present even if they are fully asymptomatic. As a result, the levels of COVID-19 can be tested through wastewater, enabling the data to be a potential predictive model as to whether cases could increase or decrease within a certain span of time.

Similar wastewater modeling websites have been created for other cities, but our application will be specific to Los Alamos, New Mexico. Through wastewater modeling we could eventually be able to predict the hotspots of COVID-19 and potentially apply this process, algorithm and data analysis to other viruses and diseases, such as the flu or monkeypox.

2.2. Overview

The Web Development app will act as a predictive model for the rise of COVID-19 cases. Once the data is analyzed, a predictive algorithm and model will be created to represent the relation of reported COVID-19 cases and the level of COVID-19 in Los Alamos's wastewater data. After, then use the recorded data to predict what the trend of COVID-19 could be for about a week into the future and use it as a preventive measure to inform individuals and Los Alamos management to take preventative measures like wearing masks or having employees work from home for an allotted amount of time. Below is a system overview that helps visualize how the system will interact and communicate, the two major systems are the frontend and backend which all have their supplemental subsystems that are necessary for the functionality of the system. Essentially data that details, Covid -19 wastewater levels as well as clinical cases, will be held in the AWS server which will communicate with data processing in order to parse through data and organize it, which will then be fed into the algorithm to create a prediction which will then be shown on the frontend GUI.

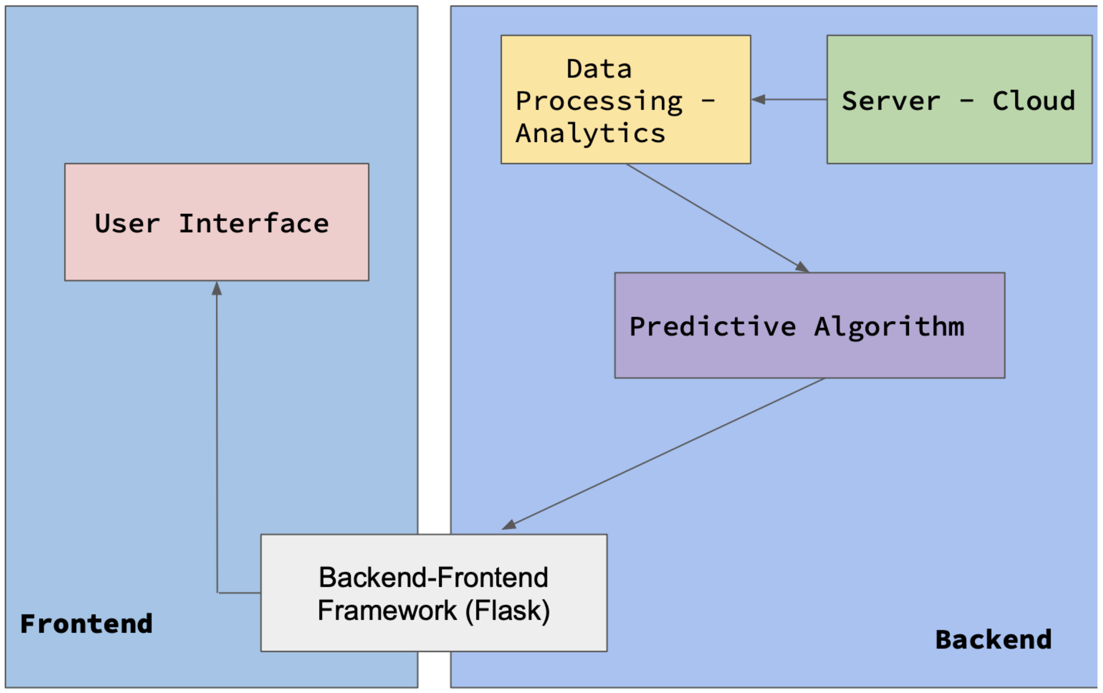


Figure 1. System Overview

2.3. Referenced Documents and Standards

Document Name	Revision/Release Date	Publisher
Heroku Manual	7.0	Salesforce Inc.
Python2.6	2.6	Python Software Foundation
AWS EC2	4.9.458	Amazon Inc.
Flask	2.2.2	Python Software Foundation
Python Library Reference	3.10.7	Python Software Foundation

Table 1. Reference documents

3. Operating Concept

3.1. Scope

The proposed project requires creating a web application. The web application will contain predicted COVID-19 cases about a week out from the current date which will be presented in a graphical format. The algorithm to calculate the number of predicted cases will be developed in the backend of the application through the utilization of wastewater data provided by the representative of a local county. The output of the algorithm will be presented in the graphical representation on the web application for LANL management to make conclusions from.

3.2. Operational Description and Constraints

This project will allow Los Alamos National Laboratory (LANL) to track the cases of COVID-19 via testing the local wastewater. This project requires developing an algorithm that creates a prediction of what the cases will look like about a week in advance. Having a week-long warning can allow the laboratory to take precautions based on the resulting trend. This allows LANL to tailor their work environment, if test cases are rising or falling, to the current threat level of COVID-19. Additionally, this project can be used on a larger scale such as tracking the COVID-19 spread between multiple counties, by testing their wastewater sources and creating a representative model for it. This project has the potential to be expanded to other viruses or diseases, such as the flu or monkeypox, through additional wastewater data testing and modeling.

3.3. System Description

Our web-application consists of two primary subsystems - backend and frontend development. The back end of the application consists of data storage, hosted through a cloud storage medium, such as Amazon Web Services (AWS). The backend will consist of the subsystems that handle the data, such as initial collection/processing, analysis, and projections that utilizes python 3.6 and various libraries.

The frontend will then take these results and present them through a user interface, enabling users to effectively view correlations and projections between the recent wastewater trends to the COVID-19 cases within their region. Wastewater data will also be able to be uploaded by certain users (managers or data analysts) if needed.

3.3.1. Backend

3.3.1.1. Data Collection and Processing

This stage is responsible for the collection and processing of wastewater data (primarily COVID-19 levels) and the corresponding COVID-19 cases from that location. Although the ultimate source of data will come from Los Alamos Labs, the initial collections will come from local wastewater systems within the state, such as Brazos County (depending on availability). The COVID-19 positive cases reported within the location will then be collected from official government sources (such as texas.gov and cdc.gov if available). The data will likely come in various forms (Excel files, tables, or string/dictionary formats), a data processing program (Python or MySQL) will be utilized to standardize it all to a single format.

3.3.1.2. Data Analysis

With all data processed uniformly, it can then be analyzed to determine potential trends between the COVID-19 levels in the wastewater and the number of reported cases. This includes plotting the COVID-19 wastewater levels over a predetermined span of time, cleaning the data by removing extreme outliers from the sample, and determining a trendline that best fits the data. The same is then done for the corresponding reported number of COVID-19 cases.

The correlation between the wastewater and the number of COVID-19 cases can be modeled and compared through determining the variances between the trendlines. Depending on the strength of the correlation, the results in the model can be used to draw conclusions based on various emergent qualities. This can include determining the accuracy of the model to the location through deviance of the wastewater data from reported cases and finding the amount of time ahead. The wastewater model can also determine trends by how left translated it appears as compared to the reported cases.

3.3.1.3. Data Projection (Trend Algorithm)

With the wastewater data trends modeled for a specific location, and then project them further into the future. This can initially be done purely by extending the determined trendline forward, such as in slopes depicting the beginning of an upward trend. This has the limitations of how accurate the trendline model is to the location (from data analysis).

However, to make these predictions more advanced than trendline extension, machine learning may be utilized to make more informed predictions. This component is reliant upon the amount of data available to be utilized for supervised learning and testing, such as previous wastewater data and the reported COVID-19 cases. This could be done through a Python/MATLAB training algorithm.

3.3.2. Front End

The front end consists of the UI in which the user will be able to effectively view and interact with the contents of the web application. This consists of taking the resulting data from the backend and converting it to an easy to view model.

This requires the web app to be hosted publicly for users to have access to the web-app, which can be done through deployment tools such as Heroku. Then, a link between the backend-data (likely Python/JavaScript files) will need to be established to the HTML of the application, which can be accomplished through a web framework like Flask.

Finally, the data on the webpage will then be formatted such that the webpage can be viewed and interacted with easily, which is done through CSS, HTML and JavaScript styling.

3.4. Modes of Operations

With the requirements presented to us by the Los Alamos Laboratory, there is only one main mode of operation. This is where the developer of the COVID-19 wastewater data can upload such data to the web application and present that community with their number of predicted cases of COVID-19 by about a week into the future. If there are any requirements that are presented to us in the future relating to the modes of operation, this section will be updated accordingly.

3.5. Users

The primary use of the COVID-19 wastewater web application will be for management at the Los Alamos Laboratory to allow them to have access to a system to make informed decisions on company protocols (COVID-19 related). Individuals who are permitted to upload additional wastewater data, will be able to do so through the web-application. Since the predicted trends will be shown using a web application, there is no training that is required for installing or using the web application. Workers at the laboratory will benefit from it because they can start taking precautionary measures if cases seem to be rising from the predicted trend to keep themselves and others safe at the laboratory and in the county.

3.6. Support

To support the user's interaction with the web application, a sort of tech support would be given by providing contact information to the user on the web application to report any issues or questions with the application. A user manual is not necessary for the user as the UI of the web application will be sufficient for most of its use cases. Any involved tasks (such as uploading additional data) will have clear directions in the application.

4. Scenario(s)

4.1. LANL Application

The primary use of the wastewater modeling application will be to provide LANL managers a tool to be knowledgeable about and prevent the spread of COVID-19 in the LANL facility. Currently, LANL (among other major institutions) must wait for long periods of time to get accurate test results of COVID-19. This prevents them from stopping the virus from spreading within the facility. However, by using, analyzing, and presenting the wastewater data, in an easy to upload and read format, managers can stay ahead of the curve rather than react after the fact. Wastewater will be continuously tested and uploaded for an accurate, fast, and steady stream of COVID-19 results.

4.2. Public Application

Much like LANL, the public and businesses also have the same issue of getting reliable and current COVID-19 data. This can create a false sense of security for many because of an area can be inaccurate to the current state. With the use of the wastewater application, this issue can be addressed through an early and representative depiction of Covid-19. With the use of the application in certain locations, the public will be able to make informed decisions on when to wear masks, stay out of highly condensed areas and practice proper social distancing.

4.3. Expansion of Tracked items

Wastewater allows for faster COVID-19 test results but is not the only thing trackable through wastewater. Much like with COVID-19, the flu, monkeypox, and even marijuana can be tested through wastewater data. These different tests could create expansion of the application or create different versions pertaining to each individual test. Much like with COVID-19 cases this can create opportunity for both awareness and prevention of each case. Not only would this be a good library for viruses' historic spread in certain areas but also creates an opportunity to influence real-time decisions.

5. Analysis

5.1. Summary of Proposed Improvements

- Depicts an accurate model of COVID-19 trends up about a week before reported cases data is updated.
- Easy to use interface to view previous data, correlations, and projections.
- Decreased reliance on individual data from official case number reports.

5.2. Disadvantages and Limitations

- Locations where wastewater data can be collected on a regular basis is limited.
- The amount of data that can be received regularly is limited.
- The precision/accuracy of the wastewater to detect COVID-19 is still being tested.
- The strength of the correlations between COVID-19 levels in the wastewater and the subsequent reported COVID-19 cases.

5.3. Alternatives

As is the case with most cities, Los Alamos is utilizing the official COVID-19 cases reported from New Mexico to analyze trends. This system results in constraints that can be addressed through the utilization of wastewater modeling:

5.3.1. Constraints

- The unreliable availability and access to data from the state, as current government policies regarding COVID-19 data may be influential factors.
- The need for access to individual data (COVID-19 case results).
- Accuracy of COVID-19 positive cases (false positives and/or unreported positive cases).
- Timing of the positive COVID-19 reported data is post-infection, making preventative measures to limit the spread less effective.

5.2.2. Benefits

- The wastewater data coming from cities rather than individual reports, resulting in a macro-view of COVID-19 spread without the reliance on data sourced directly reported from the state/government.
- The ability to obtain COVID-19 information without the need for any individual data.
- Unreported cases and other variations in reports would be better accounted for through a single wastewater measurement.
- The potential to depict COVID-19 levels before individual tests are reported, causing preventative measures to be more effective in minimizing risk.

5.4. Impact

Through enabling trends in COVID-19 cases to be effectively tracked and projected up to about a week in advance, preventative measures can be taken earlier to limit the spread of the virus, thereby benefiting the overall health of the population. The macro nature of the wastewater data also reduces the need to gather individual data, thereby providing increased privacy of individual information.

Wastewater Modeling
Matthew Delorenzo
Quinlon Horndasch
Grace Salau
Ayaan Sunesara

Functional System Requirements

REVISION – Draft
28 September 2022

FUNCTIONAL SYSTEM REQUIREMENTS FOR Wastewater Modeling

PREPARED BY:

Author	Date

APPROVED BY:

Project Leader	Date
John Lusher, P.E.	Date
T/A	Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	[09/29/22]	[Grace Salau]		Draft Release

6. Introduction

6.1. Purpose and Scope

This document provides information on the subsystem requirements regarding the performance, functionality, software, and communication requirements for the system to be fully functional. Wastewater modeling is broken into two major system groups which are the front end and the backend. The front-end system involves a graphic user interface that communicates with the backend to show the predictive trends stemming from the acquired wastewater and clinical cases in the LANL area. This shows positivity rates (predicted covid trends) for the upcoming week in a graph as well as other graphics. The backend system consists of a predictive algorithm subsystem, data processing and analytics, as well as a server. The predictive algorithm is machine learning created to take in the data that's been processed and analyzed and predict the trends for the following week. The server supplied by AWS is where the data will be held for it to communicate with the algorithm as well as the frontend.

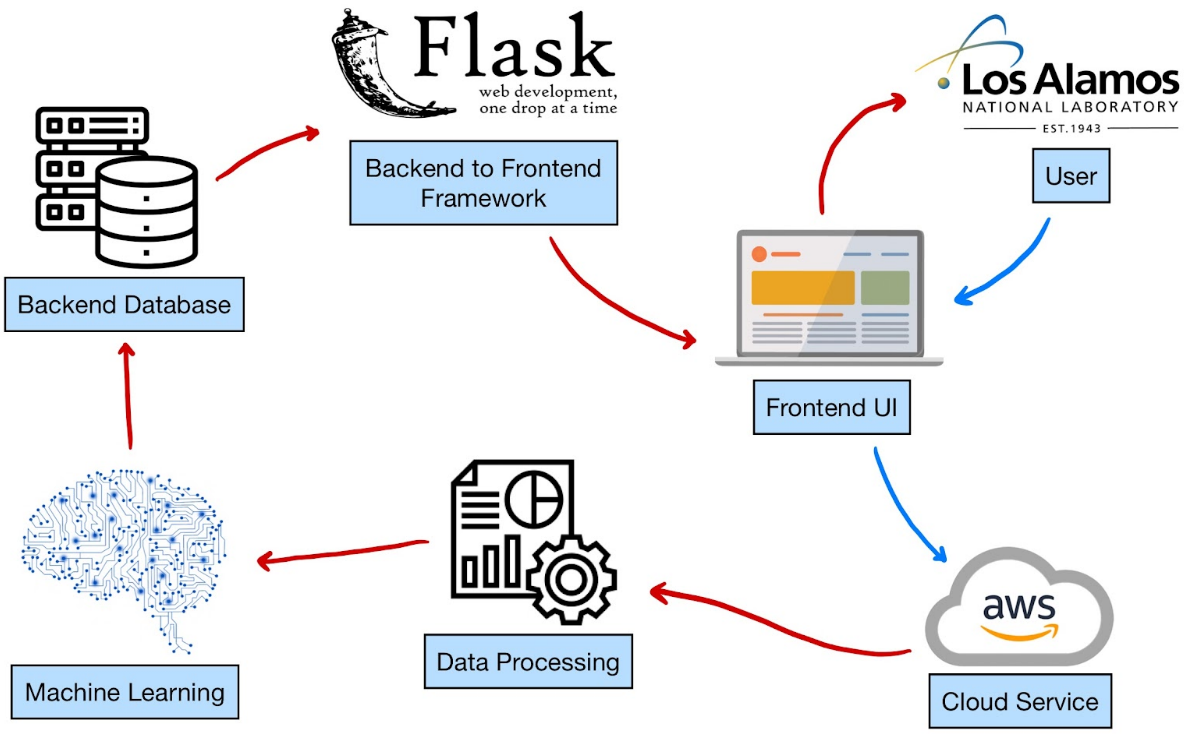

Flask
web development, one drop at a time
Backend to Frontend Framework
Los Alamos NATIONAL LABORATORY EST.1943
User
Backend Database
Frontend UI
aws
Cloud Service
Data Processing
Machine Learning

Figure 2. Functional System Diagram of Wastewater Modeling Web Application

6.2. Responsibility and Change Authority

The team leader, Quinlon Horndasch, is responsible for approving and verifying requirements of the project are met. Said requirements can only be passed with the approval of the team leader and Professor Stavros Kalafatis and prior discussion with the team and LANL sponsors.

Subsystem	**Responsibility**
Frontend User Interface	Quinlon Horndasch
Predictive Algorithm (Machine Learning)	Grace Salau
Backend Data Connection to Frontend (Flask)	Matthew DeLorenzo
Data Processing and Analytics	Ayaan Sunesara
AWS Server and Data Organization	Matthew DeLorenzo and Ayaan Sunesara

Table 1. Subsystem Leads

7. Applicable and Reference Documents

7.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Name	Revision/Release Date	Publisher
Python Library Reference	3.10.7	Python Software Foundation

Table 2. Applicable Documents

7.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Name	Revision/Release Date	Publisher
Heroku Manual	7.0	Salesforce Inc.
Python2.6	2.6	Python Software Foundation
AWS EC2	4.9.458	Amazon Inc.
Flask	2.2.2	Python Software Foundation

Table 3. Reference Documents

7.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings, or other documents that are invoked as "applicable" in this specification are incorporated as cited. All documents that are referred to within an applicable report are for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

8. Requirements

8.1. System Definition

Wastewater Modeling is composed of two major core subsystems, the backend system, and the frontend system. The backend consists of three subsystems that control the data, beginning with the collection and organization of data in a storage server supplied by AWS. This is then cleaned by the data processing subsystem which enables the predictive algorithm to synthesize what the upcoming positivity rate of COVID-19 will be based on the processed and analyzed data. The resulting data is then passed to the frontend through the Flask framework. The frontend side consists of the user interface web development subsystem, which uses a GUI to present the data and predictive trends (from backend system) of COVID-19 based on the levels/copies of SARS-CoV-2 in the wastewater as well as the clinical cases in the area. Figure 2 presents a block diagram of the overview and flow of the system, to show how each subsystem communicates and flows to present the finished product.

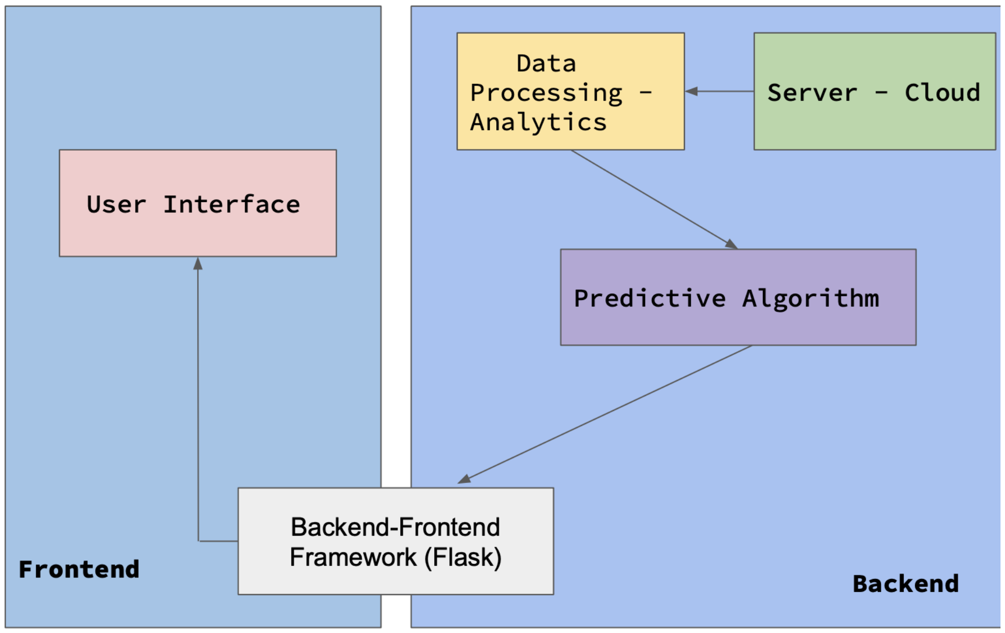


Figure 3. Block Diagram of System

AWS Server: The AWS server will be used to store the excel data automatically when it is uploaded onto the website. There are frameworks in AWS which processes the excel data that can then be used in Python for calculations. Once the data is processed, Flask can be utilized to grab any data or graphs stored in the AWS server to be posted as an output in real time.

Data Processing/Analytics: Processing of the data is composed of reading the wastewater numbers from the excel sheet that is uploaded onto the frontend of the web application to calculate the amount of positive COVID-19 cases. Once the wastewater data is parsed and the calculations are stored to be used for the predictive algorithm, a trendline graph of COVID-19 cases will be generated.

Predictive Algorithm: The predictive algorithm consists of using machine learning to get accurate predictions of the positivity rates and COVID-19 trends based on a plethora of parameters. Some of these parameters include but are not limited to population size and density of the region, levels/copies of the SARS-CoV-2 virus in the wastewater and the number of clinical cases in the area. The combination of these parameters gives a solid basis for creating a neural network or combination of deep machine learning algorithms to give an accurate prediction. The creation of this algorithm will be created using python and accessing libraries such as TensorFlow, NumPy and matplotlib.

Backend Framework to UI: The resulting prediction data structure generated from the algorithm can then be accessed and passed to the frontend of the web application. This is done through utilizing a Flask microframework. This provides tools such as the Jinja template library that enables the web-app to render specified backend data from a Python program into the HTML, which is then displayed within the user's browser. Flask can also be utilized in retrieving user input data from the UI if needed, such as HTML forms, to communicate back to the backend subsystems.

User Interface: The user interface consists of the GUI that interacts with the user to see the graphed trends/predictions of the wastewater's COVID-19 levels. It will also include interactive tabs, consisting of a function where the user (primarily LANL management) can upload data which then communicates with the backend system and displays the predictive trend for the data uploaded.

8.2. Characteristics

8.2.1. Functional / Performance Requirements

8.2.1.1. Predictive Algorithm

The predictive algorithm predicts COVID-19 positivity rate about a week into the future with 85% accuracy to prove the created algorithm is valid.

Rationale: Having a predictive algorithm within the range of 85% depicts a realistic outcome as compared to around >70% accurate where the algorithm is considered ideal rather than realistic.

8.2.1.2. Data processing Execution

The total time necessary to parse and clean the data introduced by the server should not exceed 60 seconds.

Rationale: Giving 60 seconds as the time frame to parse and clean the data is reasonable due to the runtime for looping through data which can take some time depending on the size of the dataset. The code needs to traverse the data set and remove unnecessary data which can be optimized in the future.

8.2.1.3. File Upload

The frontend user interface will have a functional tab which provides the user with a file upload option. This system must work 100% of the time and allow for at least 1000 data points.

Rationale: The file upload function should work 100% of the time because it is the primary way the web application will receive information. The file that is being uploaded must also take in at least 1000 data points without failure so that LANL can properly populate the web application with wastewater data.

8.2.1.4. Data Storage

The AWS Server will be able to hold all data necessary for the predictive algorithm to generate a prediction. This system will allow 16 GB of persistent storage that can be accessed at any point by the algorithm.

Rationale: As Heroku web applications have limited persistent storage, there must be a separate area where the raw data is stored necessary for the predictive algorithm to run correctly. 16 GB is chosen as an initial estimate for the upper limits of storage needed to train a machine learning program. Heroku is also compatible with AWS storage systems, allowing easier connections to be made.

8.2.2. Software Requirements

8.2.2.1. Software - Cloud Server

The AWS server will be utilized with Python as the primary programming language to ensure compatibility with backend subsystems.

Rationale: Python is chosen as the server interaction language due to it being easily readable, a simple application development tool, and to minimize the interfacing complications between subsystems.

8.2.2.2. Programming language - Predictive Algorithm

The programming language used to create the predictive algorithm will be python 3.6 as well as TensorFlow for any machine learning requirements.

Rationale: Due to the flexibility and frameworks that Python has; it makes it easier to develop code. Additionally, there are large amounts of documentation related to machine learning in Python which will be a great source of information.

8.2.2.3. Programming Language - Frontend User Interface

The programming language used to create the website and build the frontend graphical user interface will be HTML and will be hosted through Heroku.

Rationale: Heroku allows deployment of websites via GitHub without any expenses. The website will be programmed in HTML as it is the standard language to build a website and it works well with CSS (HTML styling language).

8.2.2.4. Software - Backend Framework

The web microframework chosen to integrate the backend data into the HTML of the UI will be Flask. Flask uses its tools such as Jinja2 templating system to complete the backend to frontend connection.

Rationale: Flask was chosen to be the web-app backend framework due to the data processing and predictive algorithm subsystems utilizing Python programs - Flask is ideal for lightweight and extensible Python based web-applications.

8.2.3. Interface Requirements

8.2.3.1. File Inputs

The input of the data file must be in a .csv file format. There will also be a file size limitation which will be decided in the future once there is proof of what file size the data processing takes is too long to execute.

Rationale: Having the file input be restricted to a .csv format allows for a singular method of accessing the data. Additionally, a file size being too big will cause the code in the backend to run for an extended period, making the web application inefficient.

8.2.3.1.1 Operating System

The web application can be run through devices on Linux, Windows, or MacOS. However, the application is optimized for desktop/laptop devices, so mobile devices are not recommended to interact with the website.

Rationale: This web application is not limited to a specific operating system, as it is publicly hosted on a Heroku cloud platform. However, the website is not designed for mobile devices, which makes the functionality less effective for mobile users.

Appendix A: Acronyms and Abbreviations

Below is a list of common acronyms and abbreviations used in this project.

AWS	Amazon Web Services
ANN	Artificial Neural Network
CSS	Cascading Style Sheets
CSV	Comma Delimited Value
DL	Deep Learning
GUI	Graphic User Interface
LANL	Los Alamos National Laboratory
OS	Operating System

Appendix B: Definition of Terms

Frontend: The part of the system that interacts with the user, usually contains a graphical user interface.
Backend: The part of the system that is not accessed by the user, usually a database that stores and manipulates data.
Neural Network: A subset of machine learning that models the human brain by being composed of layers, to give accurate models and analysis.
Flask: Backend framework used in web development to interact with the user interface (Connects backend to frontend).
Jinja: Library within Flask framework enabling templates to implement Python data into HTML.
TensorFlow: Machine learning library that helps coders to create dataflow graphs by using numerical computation and large-scale machine learning.
NumPy: Python library used for working with data in arrays.
Matplotlib: A visualization library in python used to create graphs and charts.
HTML: Hypertext Markup Language which is responsible for frontend development.

Wastewater Modeling

Matthew Delorenzo
Quinlon Horndasch
Grace Salau
Ayaan Sunesara

Interface Control Document

REVISION – Draft
29 September 2022

INTERFACE CONTROL DOCUMENT FOR Wastewater Modeling

PREPARED BY:

Author	Date

APPROVED BY:

Project Leader	Date

John Lusher II, P.E.	Date

T/A	Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	[09/29/2022]	Grace Salau		Draft Release

9. Overview

The Interface Control Document will provide an overview of the communications and interfaces between each software subsystem. The overall web application is structured around data analysis in the backend. First the AWS server must first collect and clean the data which is then accessed by the predictive algorithm (machine learning) for analysis and then projects its results to the frontend Heroku web application through a Flask integration. The user can then interact with the system through the GUI.

10. References and Definitions

10.1. References

Refer to section 2.2 of the Functional System Requirements (FSR) document.

10.2. Definitions

Below is a list of common acronyms and abbreviations used in this project.

AWS	Amazon Web Services
ANN	Artificial Neural Network
CSS	Cascading Style Sheets
CSV	Comma Delimited Value
DL	Deep Learning
GUI	Graphic User Interface
LANL	Los Alamos National Laboratory
OS	Operating System

Frontend: The part of the system that interacts with the user, usually contains a graphical user interface.
Backend: The part of the system that is not accessed by the user, usually a database that stores and manipulates data.
Neural Network: A subset of machine learning that models the human brain by being composed of layers, to give accurate models and analysis.
Flask: Backend framework used in web development to interact with the user interface (Connects backend to frontend).
TensorFlow: Machine learning library that helps coders to create dataflow graphs by using numerical computation and large-scale machine learning.
NumPy: Python library used for working with data in arrays.
Matplotlib: A visualization library in python used to create graphs and charts.
HTML: Hypertext Markup Language which is responsible for frontend development.

11. Server ↔ Data Processing Interfacing

With the uploaded data properly stored as an excel file in the AWS server, calculations are done, and AWS Quicksight can then process the data and create the desired charts. This method allows Heroku, the host of the application, to connect to AWS Quicksight which limits the amount of data transfer that is being done. Furthermore, if there are obstacles in transferring data from Heroku to AWS Quicksight, downloading the data as a parsed CSV from the AWS server and calculating externally using Python is an alternative option. This method has extra steps which may take more time but will reduce errors in data processing. Once the data is calculated and processed, a graph can then be created and exported onto the frontend of the application and hosted back on Heroku. One thing to note is that the graph will also need to be stored on AWS server to be updated for the future so the graph data will need to be exported back to AWS.

12. Data Processing ↔ Predictive Algorithm Interfacing

With the data properly parsed, cleaned, organized, and separated into training and learning sets and the necessary parameters for the algorithm to read and separate the data is set, the data is delegated into its respective layers. The data will then be imported into the algorithm to use as a CSV file. Within the written code of the algorithm a matrix of variables for the independent data will be created by pulling data supplied from the server, as well as generating the dependent variable and then encoding categorical data. The main code for the predictive algorithm will then use the data and the created datasets are output as an accurate model of the calculated predicted trend.

13. Predictive Algorithm ↔ Flask Interfacing

With the predictive algorithm finished, a resulting data structure (such as a dictionary) consisting of the date and the COVID-19 levels (known and predicted), will be generated. As this result is generated within a Python file, the Flask microframework will be integrated into it. This framework contains tools such as Jinja's “ render_template() ”, which will be utilized to pass the data in as a parameter. Flask also involves the utilization of the “ app.route() ” to specify the URL of the website that handles the associated logic. With these aspects of the microframework integrated, the data can then be utilized in HTML to provide the user with an interface containing the data that was just processed

14. Flask ↔ UI Interfacing

The HTML code of the UI now needs to display the resulting data generated by the predictive algorithm. This process is done through Flask templating. With the data structure containing the prediction results passed in through a "render_template()", the HTML code of the UI can then utilize this template to extract the data. This can be done through Jinja's templating engine, which enables Python code to be written and passes information from the backend to the HTML file. This is often done through double curly brackets "{{}}" to return the passed-in variable's value. Jinja also enables a loop to be created, allowing for multiple values of a single variable (like an array/dictionary) to be an output. With this, the raw data can be extracted for the UI to depict or graph as needed.

15. User Interface

7.1. Title Block

The Title Block will include the title of the web application (Wastewater Modeling), team number (Team #15), names of the members, and tabs (described in section 7.2.). This title block will be uniform throughout the entirety of the web application. That means that regardless of what tab the user is currently in, the title block will remain. This allows the user to navigate throughout the web application without having to go back to the homepage every time a new tab is needed.

7.2. Functional Tabs

The user interface will include functional tabs to create an organized environment for the user. These tabs will include but are not limited to: Home, File Upload, Graphs and Data, History, GitHub, and TAMU. Each of these tabs will allow the user to easily navigate through the web application.

7.2.1. Home

The user interface will include a home tab which will direct the user back to the homepage of the web application. The homepage will include the basic title block (discussed in section 7.1.), project description and images related to the project.

7.2.2. File Upload

The user interface will include a File Upload tab which will allow the user to upload wastewater data (in CSV format) to simultaneously train the predictive algorithm while also providing further analysis of uploaded data. This tab will include a file upload button, preview of uploaded data, explanation on how to use the function and explanation on what happens next (what tab to check for completed data analysis).

7.2.3. Graphs and Data

The user interface will include a Graphs and Data tab which will provide the user with the graphs associated with the uploaded data post calculations and predictive algorithm. The tab will include the current (most recently uploaded data) COVID-19 levels in a graphical format along with the associated prediction from the machine learning element, uploaded raw data and recommendation of action (from CDC) based on the prediction.

7.2.4. History

The user interface will include a history tab which will allow the user to view past data sets, graphs, and predictions (from the predictive algorithm). This will allow the user to compare the current trend to the predictive trend and confirm data from a previous upload.

7.2.5. GitHub

The user interface will include a GitHub tab which will bring the user directly to the Team #15 Wastewater Modeling GitHub to look at the code used to create the web application.

7.2.6. TAMU

The user interface will include a TAMU tab which will bring the user directly to the Texas A&M University website to get more insight into the University that the project is being completed in.

Validation Plan

Validation Plan					
Subsystem	Test Name	Success Criteria	Methodology	Status	Responsible Engineers
Predictive Algorithm	Testing Functionality of Predictive algorithm	Runs without error, takes in csv file data of various sizes, and outputs sample graph	Testing corner cases as well as fundamental cases in the algorithm with a data set/data that's reliable	In progress	Grace
	Getting a 30% accuracy on predictive algorithm	Outputted predictive trend has an accuracy of 30% when tying in all the necessary parameters with wastewater data / clinical cases	Using a dataset that has been deemed as reliable and running the algorithm and validating the trend against what the trend was	Untested	Grace
	Getting a 50% accuracy on predictive algorithm	Outputted predictive trend has an accuracy of 50% when tying in all the necessary parameters	Using a dataset that I have deemed as reliable and running the algorithm and validating the trend against what the trend was	Untested	Grace
	Getting an 85% accuracy on predictive algorithm	Outputted predictive trend has an accuracy of 85% when	Using a dataset and inputting more training and testing data and running the	Untested	Grace

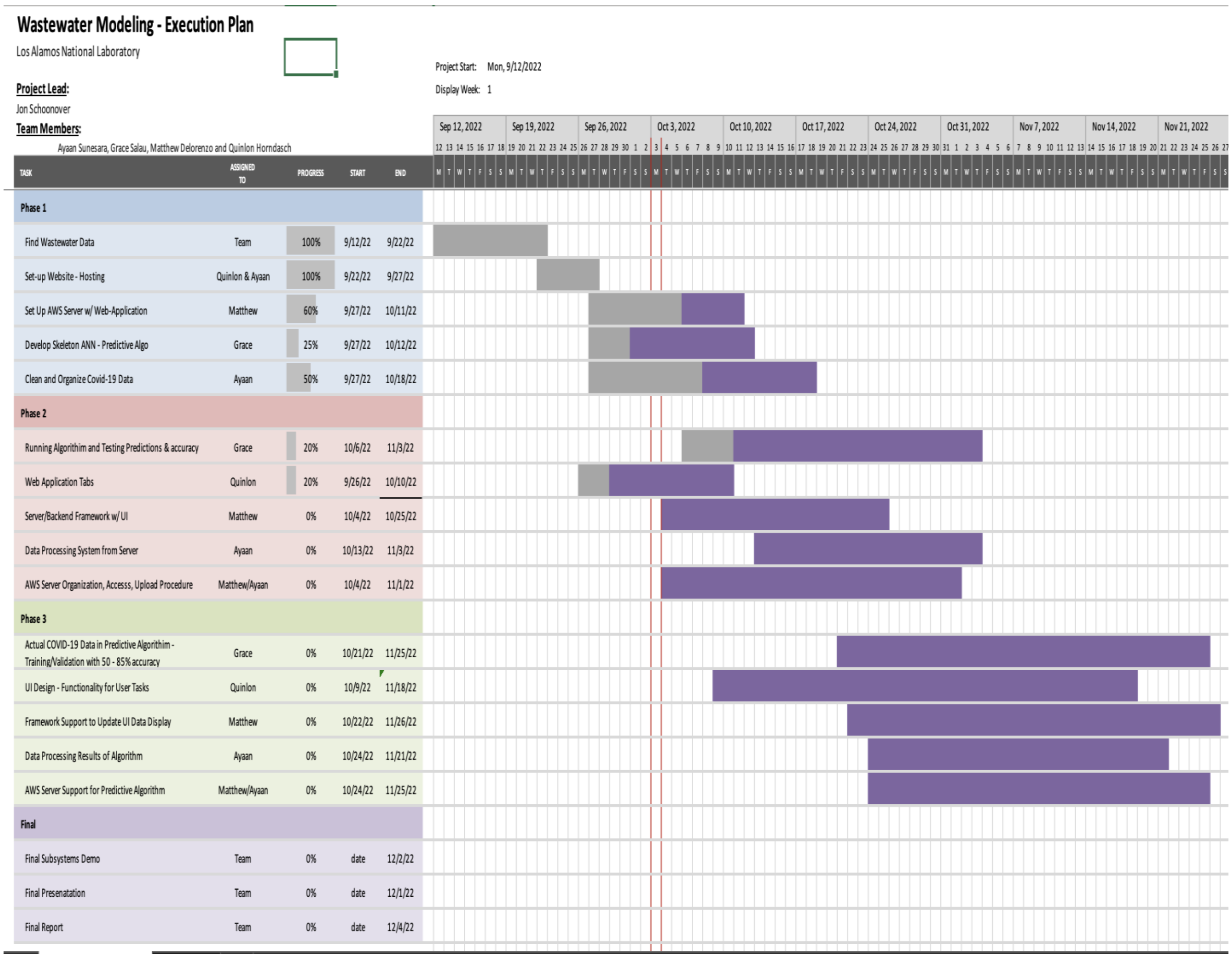
		tying in all the necessary parameters	algorithm and comparing the outputted trend against what happened		
User Interface	Testing functionality of tabs in the web application	Each tab correctly directs user to the specified part of the website (ie. Home tab brings user to the homepage)	Open the web application and click on each of the tabs to ensure the tabs correctly direct the user	Untested	Quinlon
	File upload function uploads data to AWS server	The user uploads the data (CSV) and the data gets properly uploaded onto the AWS server	Using the file upload button, a dataset will be uploaded. Next the AWS server will be checked to ensure the data was stored correctly	Untested	Quinlon
	Graphs properly show the uploaded data's trend and predictive algorithms prediction	The uploaded data is properly shown visually through graphs along with the predictive trend within the web application	Open the web application, click on the upload files tab and upload a dataset that has a known trend then click on the graphs and data tab to ensure the correct data is presented	Untested	Quinlon

	History tab shows past uploads, graphs, and datasets	When history tab is clicked and loaded, the previously uploaded datasets and associated graphs are visible	Open the web application, click on the upload files tab and upload multiple datasets then click on the history tab to ensure that there are multiple entries, and that the data is viewable	Untested	Quinlon
Data Processing	Downloading Data from Example Files to Analyze	Able to grab data from the excel sheets and create an array out of it from the server	Use an example data file to represent the wastewater data LANL will give to figure out if AWS Quicksight can read data values from excel sheet	Untested	Ayaan
	Parse Useful Data and Save as New File	Remove unnecessary data from the array created to have the data and covid values only	Be able to parse the data grabbed from the excel sheets and be able to distinguish required data to save separately. (Only covid and date data can stay)	Untested	Ayaan
	Create Graphs and Save it on AWS Server	Plot the trend on a graph and export it back to AWS server to use	Using matplotlib library in Python to plot the x and y values, creating a graph showing a trendline	Untested	Ayaan

	Output Graphs on Front End of Web Application	The web application updates the graph page when a new csv file is uploaded	Connect the graph from AWS server to the Flask framework to show the most recent covid case graph to the user. There should be a change when a new csv file is uploaded	Untested	Ayaan
	Able to connect the web app backend data to the frontend.	A variable declared in a Python file can be displayed on the UI of the web-app.	Utilize Flask templating to pass the variable from the Python file into a predefined portion of HTML code - test if correct value is then displayed	In progress	Matthew
Backend-Frontend Integration	Data values from the server can be displayed.	A variable derived from the AWS Server can be displayed on the UI of the web-app	Find the variable stored in AWS from the server, utilize Flask templating and routing in the file to pass the variable to a predefined HTML code portion	In progress	Matthew
	Able to push large data constructs (graphs/projections) to UI.	A larger chunk of data (such as an array variable for a graph) can be displayed to the UI	Utilize Flask to take the data structure through templating/routing, and loop through the values to display them in the HTML side	untested	Matthew

	Can refresh presented data with an updated version when specified.	When a button is pushed on the UI, the backend can push the requested data to the frontend	When a button is pushed, triggers backend data processing to update, Flask utilized as before to push new data to the UI.	untested	Matthew
	Can transfer user input data from the UI to the server.	User can input data in the UI that is able to be input in the data server.	Use Flask/HTML to grab the UI data, place it in the Python code, then deposit it into the server data repository correctly.	untested	Matthew

Execution Plan



Wastewater Modeling - Execution Plan

Los Alamos National Laboratory

Project Lead:

Jon Schoonover

Team Members:

Ayaan Sunesara, Grace Salau, Matthew Delorenzo and Quinlon Horndasch

Project Start: Mon, 9/12/2022

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Phase 1				
Find Wastewater Data	Team	100%	9/12/22	9/22/22
Set-up Website - Hosting	Quinlon & Ayaan	100%	9/22/22	9/27/22
Set Up AWS Server w/ Web-Application	Matthew	60%	9/27/22	10/11/22
Develop Skeleton ANN - Predictive Algo	Grace	25%	9/27/22	10/12/22
Clean and Organize Covid-19 Data	Ayaan	50%	9/27/22	10/18/22
Phase 2				
Running Algorithim and Testing Predictions & accuracy	Grace	20%	10/6/22	11/3/22
Web Application Tabs	Quinlon	20%	9/26/22	10/10/22
Server/Backend Framework w/ UI	Matthew	0%	10/4/22	10/25/22
Data Processing System from Server	Ayaan	0%	10/13/22	11/3/22
AWS Server Organization, Accesss, Upload Procedure	Matthew/Ayaan	0%	10/4/22	11/1/22
Phase 3				
Actual COVID-19 Data in Predictive Algorithim - Training/Validation with 50 - 85% accuracy	Grace	0%	10/21/22	11/25/22
UI Design - Functionality for User Tasks	Quinlon	0%	10/9/22	11/18/22
Framework Support to Update UI Data Display	Matthew	0%	10/22/22	11/26/22
Data Processing Results of Algorithm	Ayaan	0%	10/24/22	11/21/22
AWS Server Support for Predictive Algorithm	Matthew/Ayaan	0%	10/24/22	11/25/22
Final				
Final Subsystems Demo	Team	0%	date	12/2/22
Final Presenatation	Team	0%	date	12/1/22
Final Report	Team	0%	date	12/4/22