VirusWatch

Software Design Document v1.0

Kevyn Sisante
Anas Albedaiwi
Colton Barboro
Bryan Stahman
Ziang Zhou

Project Sponsor: Dr. Crystal Hepp
Project Mentor: Volodymyr Saruta

February 5, 2021

Table of Contents

1. Introduction

Since COVID-19 was first detected in the United States, more than 100 million people have been confirmed to be infected with the virus and 2.9 million people have died as of February 5, 2021. In a high-density residential environment, the spread and speed of COVID-19 will increase significantly, leading to population outbreaks. The terrain is densely populated and people have considerable disadvantages in maintaining social distance. Fortunately, state officials have been more strict with COVID policies including mandating masks in most public areas and not allowing social gatherings of 10 or more people.

Our task is to design a web application for our client, Crystal Hepp, to detect virus content in wastewater and analyze virus data. Crystal Hepp is an infectious disease expert and currently works at the Institute of Pathogenic Microbiology at Northern Arizona University. Crystal and her team work to track COVID-19 by collecting wastewater samples from different regions in Coconino County.

Our team will design and develop an application that will become a hub that can be used to analyze virus data and create visualizations for users to read the data more easily. Currently, the collection of wastewater is still primarily done near Flagstaff, but future collection areas will be expanded in hopes that we can get a bigger picture of the pandemic within the entire county. We believe creating this application can increase efficiency in analyzing COVID-19 and can have a major impact on current research strategies.

2. Implementation Overview

Since the virus can be detected in wastewater, collecting and testing of wastewater samples can be used to measure the extent of an outbreak in an area. Policy makers are struggling to create alternative methods of identifying potential hotspots so as to be able to predict the spread of the disease. As previously mentioned, our goal is to create a secure web application that can be used by stakeholders to store, view, and analyze their data. We have outlined our application to have three major components; a library of data analysis tools and scripts, a GUI, and a MySQL database.

2.1 Tools and Technologies

To implement our secure web application solution, the following tools and technologies listed below will be used:

- Django
    - A full stack web application framework that is able to easily communicate with backend functions and the database.
- MySQL
    - A relational database that will house all of our user data. MySQL is the best option here because it is compatible with many operating systems such as Linux, Unix, and Windows. It also keeps user passwords encrypted, effectively increasing account security.
- R/Python
    - These programming languages, along with preprogrammed data science packages, will aid to create the tools and scripts necessary for the stakeholders to perform their data analysis.

3. Architectural Overview

The web application that will be built to satisfy the requirements laid before the team will utilize the multitude of frameworks and libraries with the technologies chosen for building.

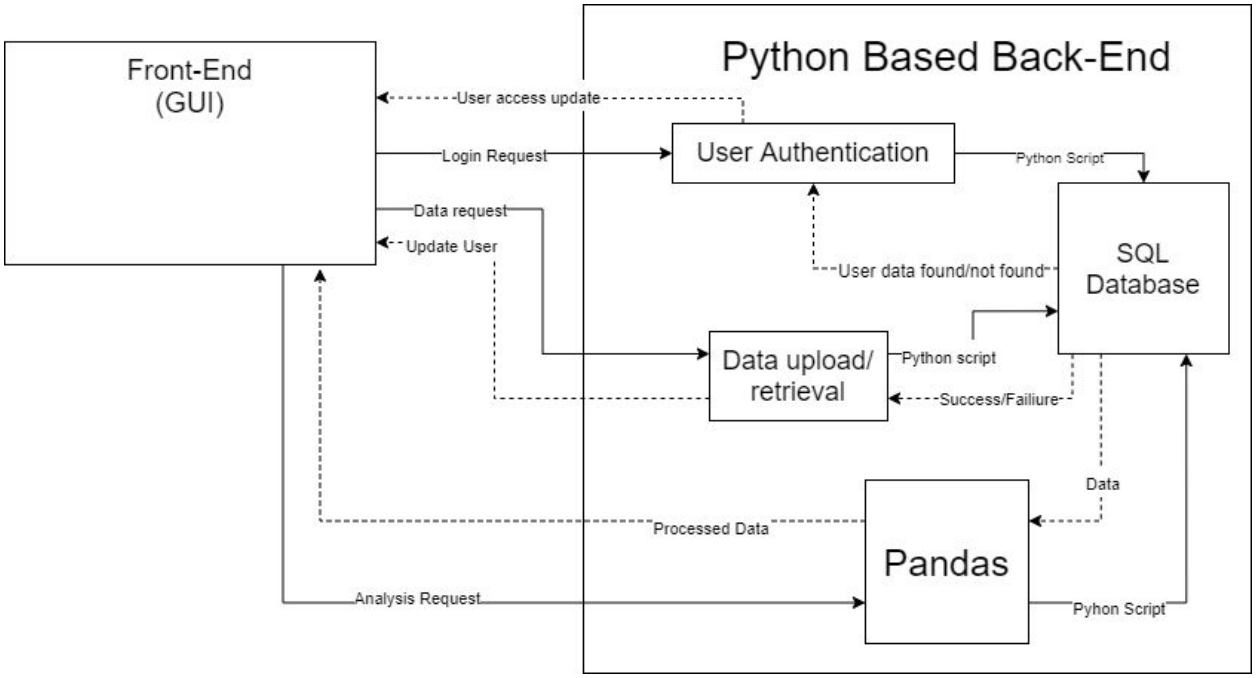

Figure 3.1 Application Architecture Diagram

Front-End (GUI)

The front end GUI will be the primary method in which the user interacts with the web application. The GUI will be responsible for a multitude of functions that the user can engage with. Such functions include:

- The ability to upload/download data files according to the privileges afforded to the user.
- Engage in user account creation and login functionality for the user to access the application.
- The ability to request analysis on data according to the privileges afforded to the user.
- Observe visualized analysis of data processed by the application.

The GUI will essentially be the window that allows the user to see the results of the application.

Back-End

The back end modules will be the main modules that work to make the application function as intended. The Django framework that makes up the back end is a Python based framework that consists of a multitude of libraries which will assist in the functionality of the site.

- User Authentication

    - Django's built in user authentication functionality will be used to implement roles based permissions in the web application.
- Data upload and download
    - The ability to upload and download files from the database is supported by Django and can be accessed according to the permissions afforded to the user based on their account role.
- Pandas
    - Pandas is the primary library used to support data analysis scripting using Python, which manipulates data taken from the database and encapsulates it as an Object type to be as mutable as the programmer needs it to be.
- Database
    - The SQL Database will store all of the data that's needed for the application to function as intended. The Database will be responsible for for storing user information such as user email addresses and passwords, as well as user privilege information and data afforded to the user based on the privileges assigned to them.

4. Module and Interface Descriptions

In this section, we will take a closer look at each of the modules and interfaces previously mentioned. These components include data analysis tools, a MySQL database, and a front-end interface. Each component has its own key role regarding the functionality of the project and

must be properly integrated with the other systems in order to achieve an efficient, fully functioning application. Here, we will explain why each component is necessary and how they will be used in the overall project.

4.1 Data analyzation

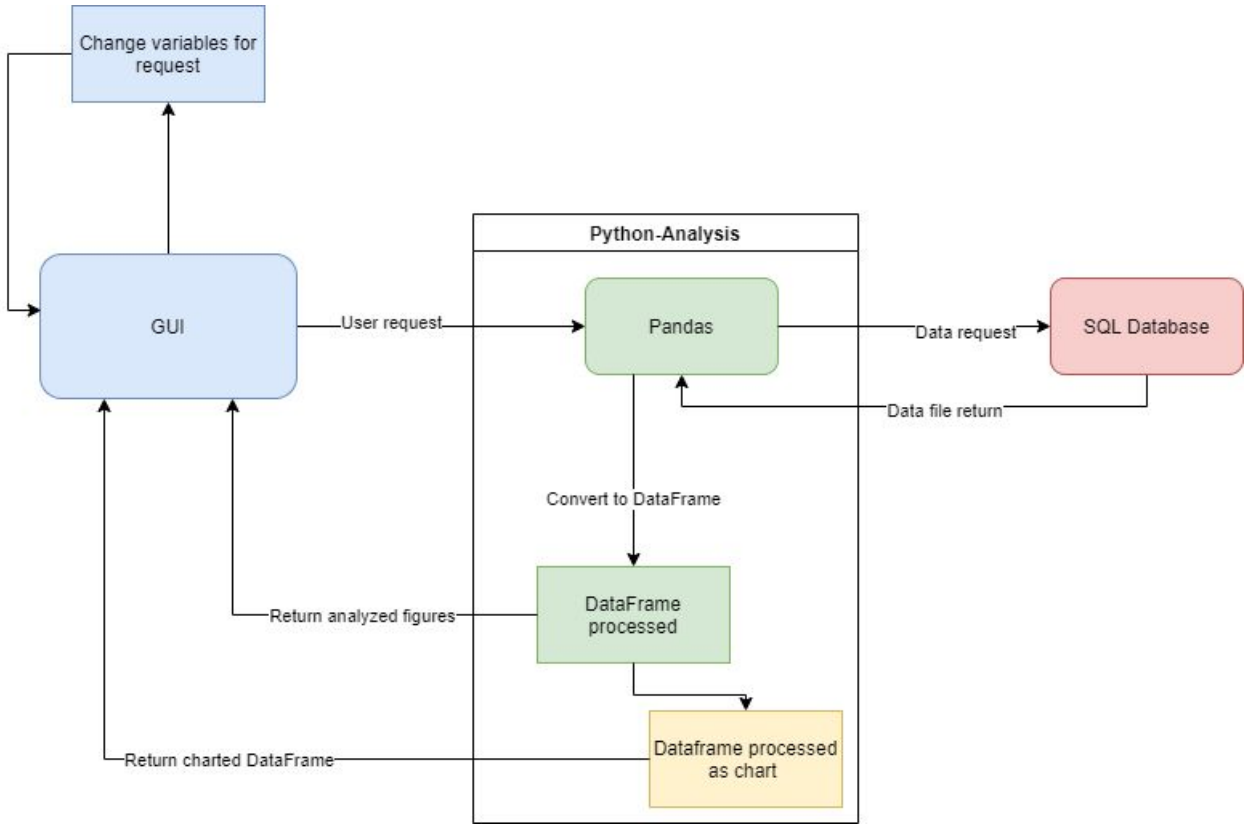

Figure 4.1.1 Data Analysis Interaction Diagram

The Python based data analysis tools existing in the back-end module will be used to process requests from the user interface for data the user wants analyzed and shown. The user will use the front end to choose the variables of the data they want processed and it will be forwarded as a request to the back end pandas module, where it will send a request to the database for the table(s) of requested data for analysis, before returning it to the Pandas module for processing. After processing, both the processed data and the graphed data will be returned to the user interface for the user to interact with as desired.

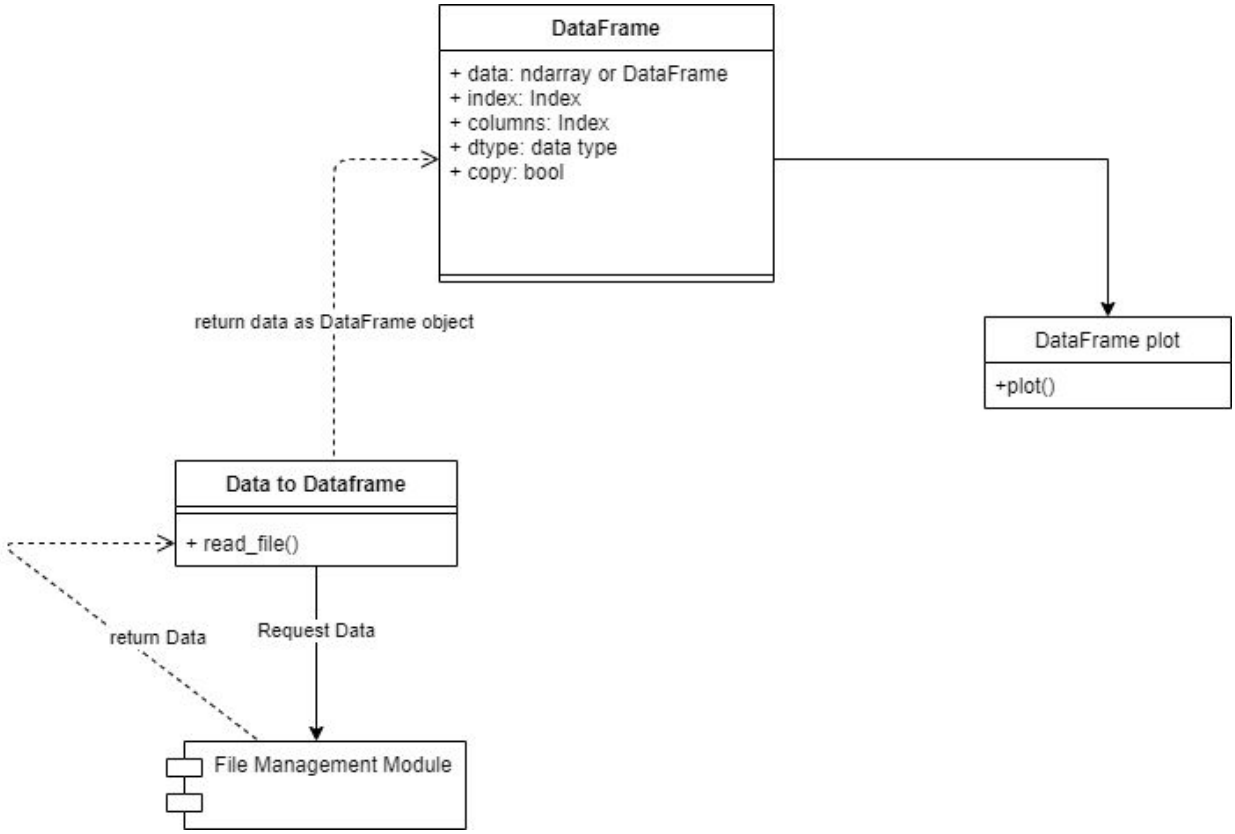

Figure 4.1.2 Data Analysis UML Diagram

The primary objects being handled by the analysis tools are DataFrame objects, which are used to encapsulate the data taken in from sources like Excel files and SQL tables and put them into a mutable object that can be manipulated as needed. The main method called after the data is requested is read_file(), which converts the data table in the database into a mutable data frame so it's usable by the rest of Pandas tools. The other main method utilizing the DataFrame object is plot(), which will take the dataframe and transform it into a chart that will be returned to the GUI for observation.

4.2 Database

To store user data in the back-end, we will be using a structure of entities as depicted in figure 4.2.1. In this Enhanced-Entity-Relationship diagram, we can see the different relationships between the tables in our database.

The auth_user table is responsible for storing data related to a user account. User accounts will be organized under groups, which will represent different organizations within the Northern Arizona area. To prevent unauthorized access, a member of a particular group will only be able to view and upload data associated with their group. The auth_group table defines the specific groups, and users are assigned groups in the auth_user_groups table. Specific permissions are defined within the auth_permission table. For special cases, accounts can be given permissions not related to any particular group using the auth_user_user_permissions table.

When a user uploads a file, a row in the user_file table will be added that contains the user's id, file name, time uploaded, and group id. This information can be used to locate the directory in which the file is located on the server, thus creating a link between the file system and the database.

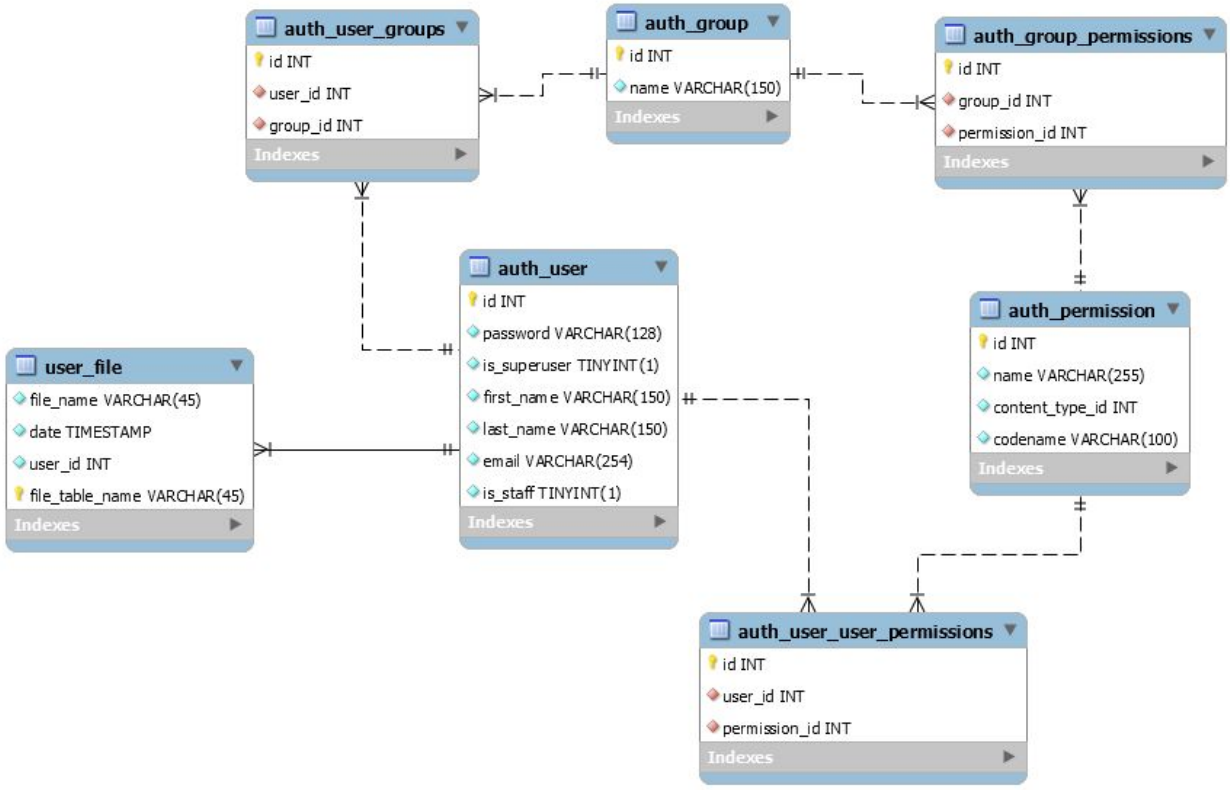

Figure 4.2.1 Database Enhanced-Entity-Relationship Diagram

4.3 Front-end Interface

The front-end interface will be the user's primary means of viewing, accessing, and analyzing the hosted data. It will also be responsible for registering users and authenticating proper credentials stored in the database. The front-end must be designed with the end user in mind, meaning that the following must be taken into consideration:

- Ease of use
- Readability
- Elegance and Attractiveness

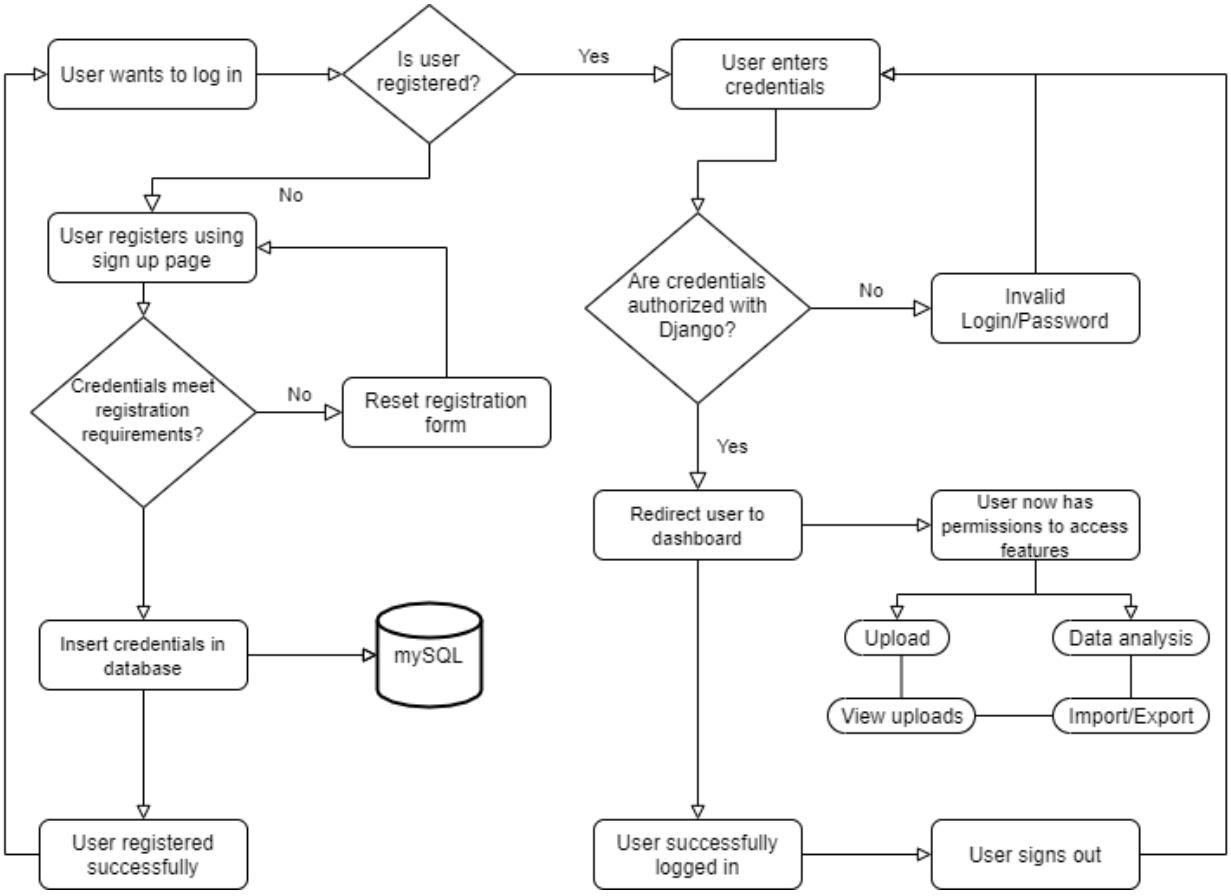

Figure 4.3.1 Flowchart of user registration and login functionality

The front-end interface will provide an area to house the data analysis tools and an area to view graphical analytic results as well as allow the user to communicate with the database more easily. Using Django's built-in User Authentication libraries, we can ensure that each account and the data within it is secure. Figure 4.3.1 shows the general flow control of front-end functionality. The front-end interface will contain the following pages:

- Landing Page
    - This page will introduce new users to the web application. It will provide a brief description of our project and explain the functionality.
- Registration Page
    - This page will be used to register new users. It will ask a user to create a username, email address, and password that will be checked with preprogrammed

constraints that will ensure an account's security. Once verified, the information is then passed into our database where the account will be stored.

- Login Page
    - This page will be used to log in users already registered in the database. Once the credentials are entered, it will be passed to Django's built-in authentication system where it verifies that the account is within the database. When the account has been authenticated, the user will be redirected to their personalized dashboard.
- Dashboard
    - The dashboard is where the user will find all the tools needed for data analysis. This page will be user defined in a sense that the user will be only able to access features within their allowed permissions. This means that a user can only perform data analysis on files they themselves have uploaded.

4.4 File management

The File Management module is responsible for uploading, organizing, storing, downloading, and deleting the excel files within the application. Uploaded excel files will be stored locally on the server and organized within directories corresponding to the group the user belongs to. Additionally, the metadata related to each file will be stored in the database. This data can be used to recreate the path to the file location on the server.

As depicted in figure 4.4.1, the File Management module's functions utilize the Database class to manage connections and execute SQL queries with the database. The Download class is responsible for retrieving files stored in the local directory for the purpose of data analysis or if the user requests to download a file. The Delete class will delete a given file from both the local file system and its record in the database. The Upload class will insert a file under a specified directory and add a reference to it in the database.

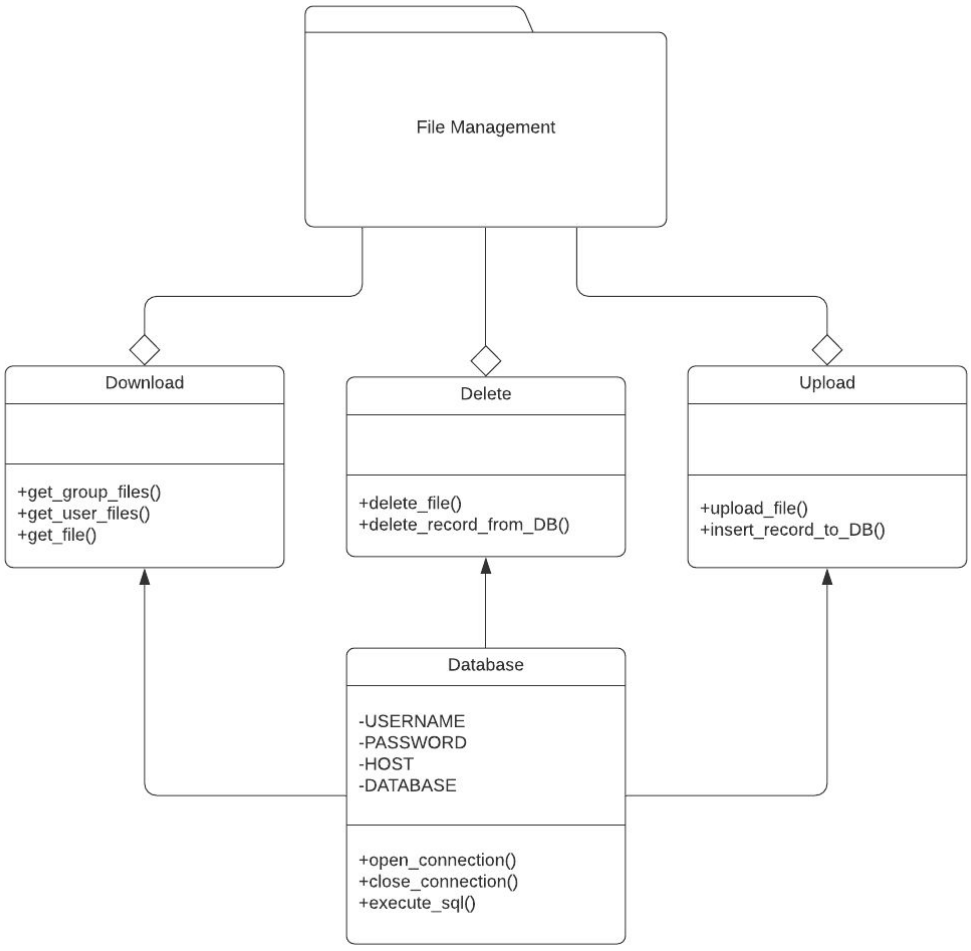

Figure 4.4.1 File Management Module

4.5 Notifications

When a file is uploaded within a specific group, members of that group that have opted in for notifications will receive an email notifying them of the new upload. Django provides a built-in module for sending emails, making this functionality easy to implement within the application. As shown in figure 4.5.1, when a file is successfully uploaded to the server, upload_file() will call send_mail() with the necessary parameters.

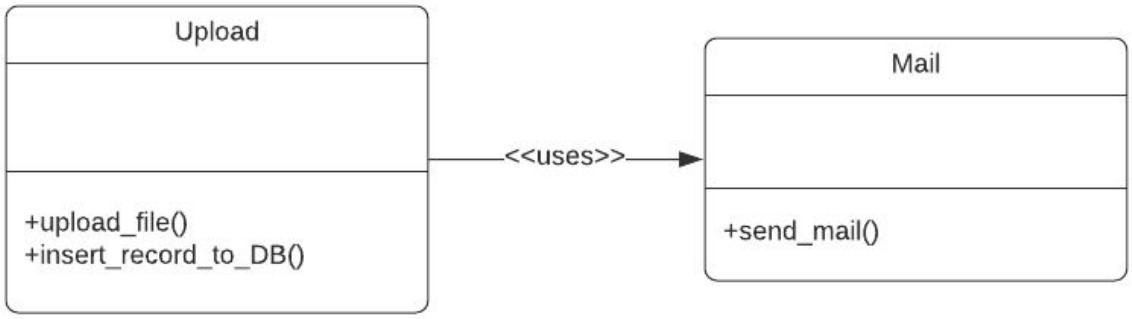

Figure 4.5.1 Notifications Module

5. Implementation Plan

In order to implement the modules and interfaces mentioned in the previous section, our team has decided to break up the project amongst the team members into the following sections: Front-end/GUI, Accounts/User Roles, Data Analyzation, Database/Security, and Search/Organization functions. For a visualization of our Implementation Plan, please refer to Figure 5.2.

The development phase of our project began when the team began to implement the different modules presented in the demo last semester. Since each module was created separately, we have been working on these sections at the same time to ensure all modules correctly interact every step of the way. As each component is created and tested, it is reviewed by a designated code reviewer before it is pushed into our team repository.

Our team's end goal is to deliver a functioning Alpha version of our web application that contains all features mentioned in the MVP. The Alpha version will be a fully integrated system ready to be used for analysis. Our team plans to complete this version by the end of March. To achieve our plans in a timely manner, certain tasks were assigned to certain team members (Table 5.1).

Front-end/GUI	Accounts/User Roles	Data Analyzation	Database/Security	Search/Organization Functions
Kevyn Sisante & Anas Albedaiwi	Kevyn Sisante	Colton Barboro	Bryan Stahman	Bryan Stahman & Ziang Zhou

Table 5.1 Distribution of Labor

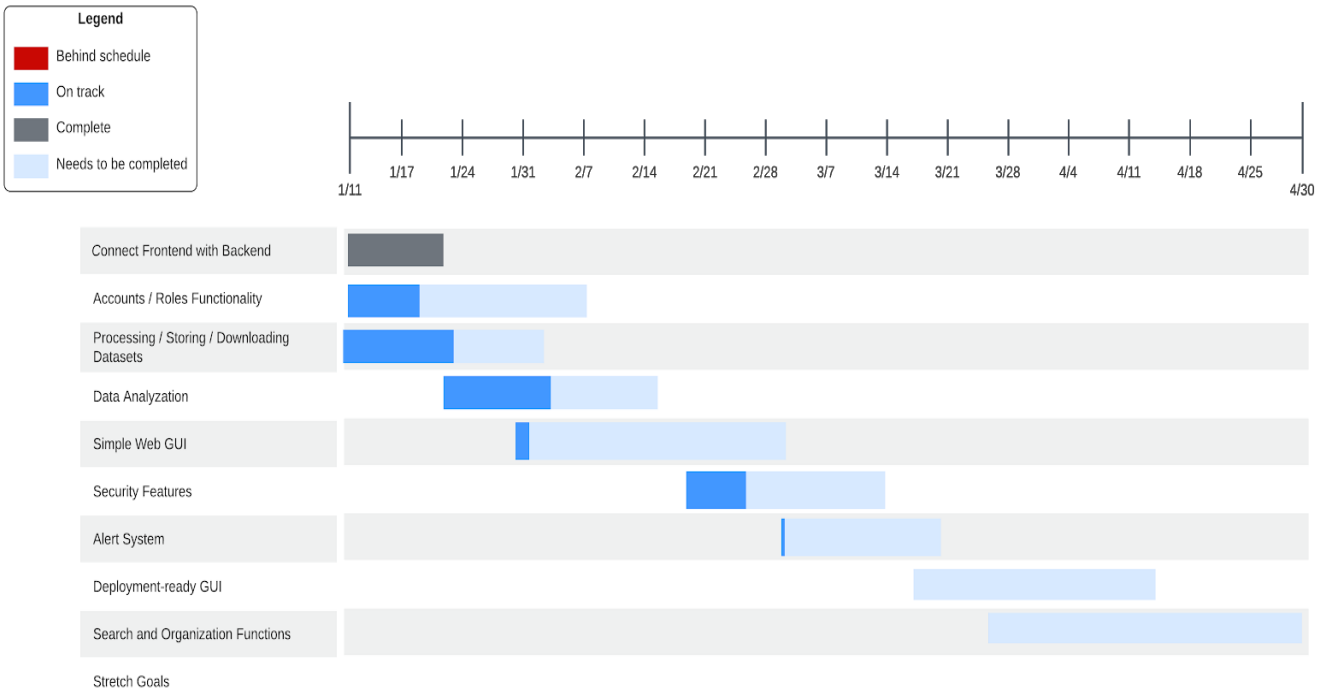

Figure 5.2 Updated Gantt Chart for the Spring ‘21 semester

6. Conclusion

Prior to being assigned to this project, most data collection out of Dr. Hepp's lab was performed manually, meaning that all data had to be collected, inputted into a file, and analyzed by hand. This impacted their collection of data as it was too complicated, the variety of data was not easy to handle, and the data storage was too difficult.

Based on the above issues, the team had decided to develop a secure website application program that is long-term, large-scale, automatically collects and sorts the virus content data in wastewater, and uploads the results to the database. This will enable users to compare data and analysis of the number of cases involved in any location in order to take appropriate action.

The development team will lay out a secure web application according to three categories: function (what operations can be implemented on the web page), performance (ensure that the function meets the requirements) and environmental requirements (ensure that the web page is safe). One of the main functions of the application is to receive data files uploaded to the web server and process the data, such as case count, % positive, etc. After the data is analyzed, Python's data analyzation system will be used to push the drawn data to the user. Currently Virus Watch web application will be designed to allow two roles, the administrator and the stakeholder. As the application grows larger in scale, more roles and permissions can be added accordingly. When VirusWatch is completed, it would be ideal if the project makes a large contribution to society.