Software Design Document

02/12/21

Team Name - Efficient Tester

Project Sponsor: Dr. Tara Furstenau

Team Mentor: Tomos Prys-Jones

Team Members

Tyler Conger

Miguel Villareal

Xiaobai Li

Bailey Mauss

Yihao Lu

Table of Contents

1. Introduction

In the United States, Laboratories in the United States routinely conduct many tests for infectious diseases, and at present there is the added pressure of testing for COVID-19. In particular, during the novel Coronavirus epidemic, laboratories in the United States were testing at least 600,000 samples a day. Today we are testing more samples at present than this time last year, prior to the pandemic. While current testing effectively detects positive cases of COVID-19, it can be wasteful with regards to reagents, laboratory consumables, and time.

With the ongoing COVID-19 outbreak and the reopening of public places, people are returning to work and school. We are coming into contact with more people. This increased contact could drastically increase the number of people who need testing. It has revealed a huge challenge to our ability to detect pathogens. Traditional testing is effective at detecting pathogens, but if the prevalence of infection is low in a population, many reagents and consumables are wasted. As the majority of tests will be used on healthy individuals. In these situations, it can be more effective to group samples into a single test tube and look to see whether any of the individuals are infected. If the pooled test is negative, then the researcher has saved n – 1 tests (where n is the number of individual tests that were grouped together). If the pooled sample is positive, this group is subdivided into smaller pools and the process is repeated. This method was developed by Robert Dorfman and is used by our sponsor Tara Frustenau and her research group, who help laboratory-based testers to improve efficiency.

For rapid testing, our clients have developed their own protocols for group testing on a 96-well plate. The positive samples can be quickly detected in two or three steps. First, we create six pools with 16 test samples in each pool. After passing the first test, we select the pool with a positive population test, divide each pool into two pools on average, and conduct the second group test. In each step the number of samples that are pooled together decreases, with individual samples being tested in the final step. An example of this three-step process would be: a) dividing the 96 well plate into six pools of 16 test samples, b) If a pool comes back positive after the first test, the pool is divided into two pools of 8 samples, c) Finally, if the second test comes back positive, the individual samples for each positive pool are tested individually. While the example above uses pools of 16 samples per test tube, followed by 4 and then 1, the optimal pool size may vary depending on the expected infection rate. After the second population test, samples from each positive pool are tested one by one. Group testing method is known and effective at detecting infected individuals, we believe that improvements could be made to the clarity/timing of the process with a workflow assistant application. The app would help technicians decide how to subdivide the 96-well plate, keep track of positive tests and assist in recording results.

Our solution is to develop a mobile application that runs across android and IOS with the main function of helping researchers design an optimal pool and track samples during testing. It will run on a smartphone or tablet carried by a researcher or lab technician. In the application, firstly researchers will get a graphical image of a 96 or 384 well plate. Then the application would assist in determining the number of samples to the pool together at each stage(based on the estimated positive infection rate). And we will use the database to store the test results and update the protocol in real-time. We will also build a database to store technician/researcher lab notes as well. Finally, we will develop a boot user interface (GUI) to visualize the protocol, monitor protocol execution, and review the results.

The key functional requirements of the application are：

- Users can directly click on the well plate image to conduct experiments
- All of the user's progress can be completely and securely saved
- Each experiment can only be edited by the same user, and other users can only be 'view-only' unless authenticated

The key performance requirements of the application are：

- The application response time must be within 0.2-2.5 seconds
- A user interface that allows users to interact smoothly with an application
- The application must be secure and safe to use

The key environmental requirements of the application are：

- browsers that support HTML5
- Internet connection
- Devices which can support system version above IOS11 and Android7

2. Implementation Overview

Our solution vision for our clients' problem is to create a web and mobile application that will be used by our clients and their colleagues in their lab to interact with a well plate image and create experiments and protocols that will be used on a daily basis. The overall approach that we are taking to help solve this problem is creating an application that will use a multiple number of frameworks (As seen in Figure 1 below) such as an Ionic framework for the front end GUI and will implement a Django backend/DjangoREST framework to communicate with our pillow python interactive image and our database that will be connected to through MongoDB which will store our information such as users, experiments, protocols, etc. We chose to use an Ionic front end because most of our team has experience with HTML and CSS and we feel that the learning curve for Ionic based on that experience would be shallow. Django and DjangoREST were chosen because our client uses Python scripts that will give us the data needed to run our interactive table. Django is a framework that uses Python so we felt that it was an easy decision since integrating our clients' scripts into the backend will be much easier and will not have us translating those scripts into a different programming language. In the creation of our interactive image we chose to go with the pillow python library for easy to use image creation and since it will integrate well with our django backend. For our database, we selected MongoDB because the types of data that we are storing does not necessarily follow an object oriented design which is used in many other database frameworks. It is also more suitable with our backend framework(Django) and is cheaper as well. There is plenty of documentation of these frameworks working together and also documentation on how to link these frameworks so we felt that the process of doing so would be something that can be done in a fairly short amount of time. Our team chose to use these frameworks because we all have experience with the languages used in all of them and felt it would give us more implementation time on our development time frame. We have demonstrated our understanding of what each framework could do in our technology demonstration at the end of our last academic semester and as a team we are making progress day by day, gaining more knowledge of what these frameworks are possible of and how we can use them to our advantage. As a team we feel that now in the process of implementing these technologies, the right decisions were made going forward with the technologies we chose and will continue to make more progress in completing our solution to our clients problem.

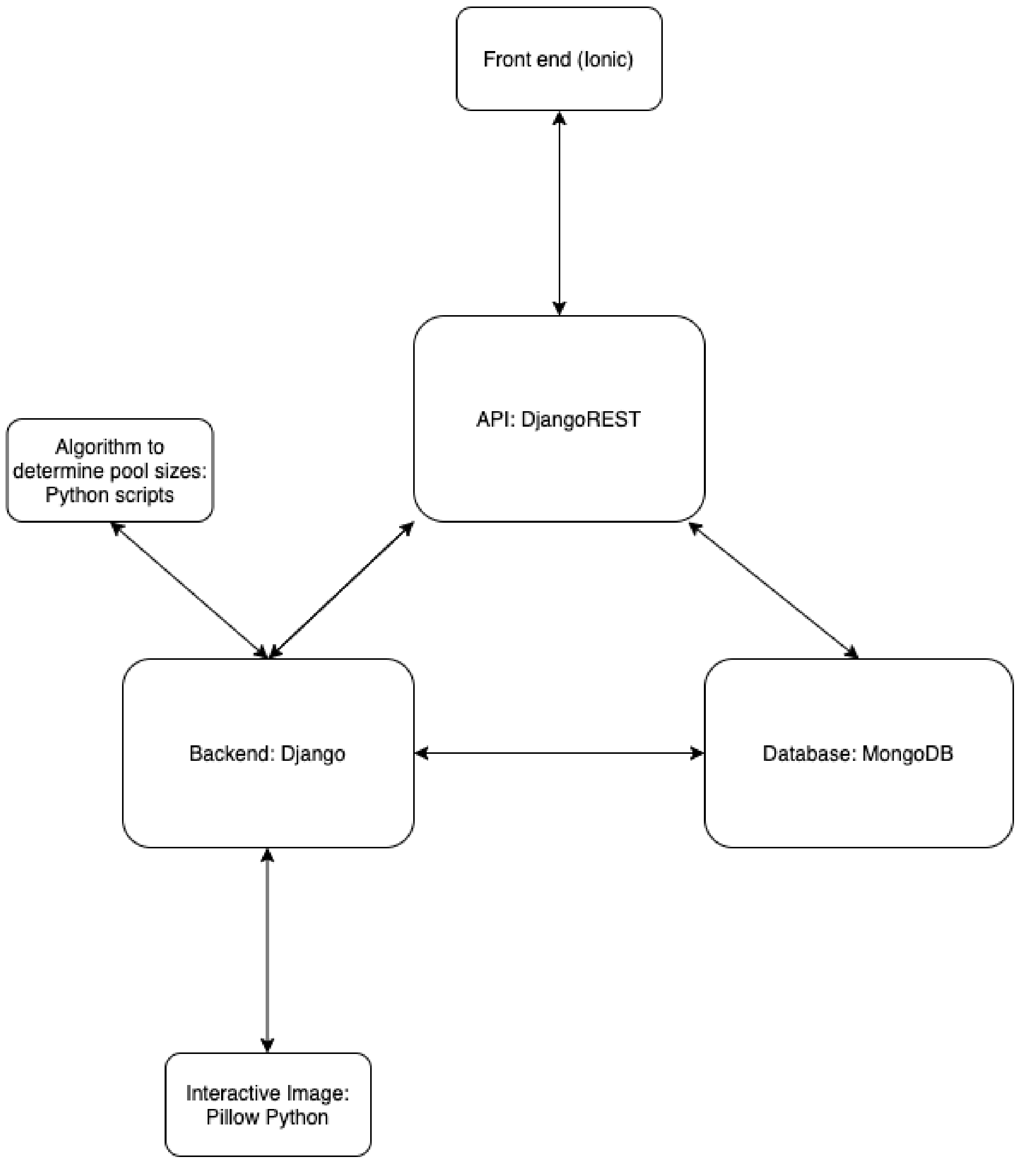
Front end (Ionic)
API: DjangoREST
Algorithm to determine pool sizes: Python scripts
Backend: Django
Database: MongoDB
Interactive Image: Pillow Python

Figure 1. This figures shows the interaction between all frameworks being used within our application and how they will go about communicating with each other.

3. Architectural Overview

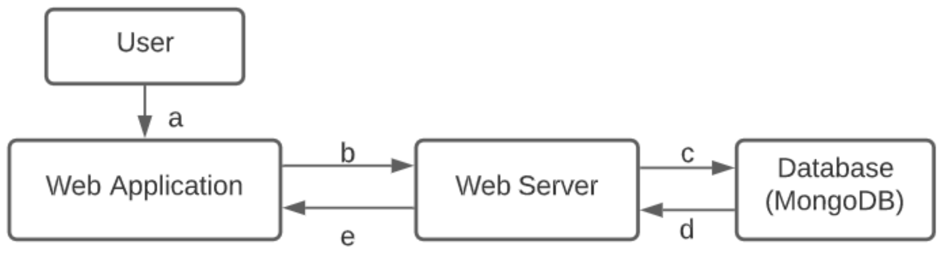

Figure 2. Showcases the flow of operations between each component. Each different interaction is labed, (a) represents the user’s interaction with the web application, (b) represents the web application sending data to the web server such as user login information, (c) represents the data being looked up and stored in the database, (d) represents the return of data from the database, (e) represents the application retrieving the data from the web server

3.1 Key responsibilities and features of each component

Our product consists of three main components (Figure 2), the web application which the user interacts with, the web server that supports the web browser and the database which stores all datas. The web browser forms the front-end of our product and will mainly be written in Django. The user stores the data through the web page to MongoDB (Figure2). The main body of the website is the same with the homepage on application which are protocols, experiments, groups, and interaction images.

The web application is the central controller of the software and acts as a middleman, passing information from the browser to the database and back again. Because of the website service, users can easily login, register and so on. And the website service is also convenient for us to conduct all kinds of testing in the design stage. The application is another part of the front end of the system. It is convenient to the user in the client operation and the application connects to the database through the server.

The database is used to collect user information and store data. The database needs to connect to both the Web server and the application to keep the information synchronized. The Web server then passes the data to the Web browser. The database requires a higher level of security to ensure that users' information is protected and secure. It also requires some scalability because the database needs to be in constant use and data may be added or changed over time. In addition, portability is required, because when users change, their servers may need to be redeployed on other systems.

3.2 Communication mechanisms and control flows of the architecture

From Figure 2, we have 5 main communication mechanisms and flows.

3.2.1 User - Web application (a in Figure2)

Users interact with the web browser and application by uploading their data and notes.

3.2.2 Web app - Web server (b in Figure2)

This step sends the login or registration information entered by the user to the server so that it can be processed accordingly and read work progress, such as test progress.

3.2.3 Server - Database (c in Figure2)

After the data and records are received, the server's job is to pass them to the database. The database then takes care of storing according to the data models.

3.2.4 Database - Server (d in Figure2)

After the database receives the data, it stores the data respectively, and returns the data to the server according to the user's needs, and the server returns the data to the Web application.

3.2.5 Server - Web application (e in Figure2)

The Web server helps implement the login system and the Web application gets data from the Web server.

3.3 The influences from architectural styles

Because our architectural style is layered, the whole project works individually. The failure of one part will not affect the whole project. Especially during development, many parts of a project can work together, such as the login system and front and back end. Their development process does not interfere with each other. The dependency between the various parts is low and joint problems are not easy to occur. When something goes wrong in one part we just deal with the immediate problem, not the other parts. Because of this layered design, we can focus more on standardizing the program. The most important point is that when we want to call some part of the logic, it does not take much time to pass an architecture.

4. Module and Interface Description

4.1 The Login System/Interaction

In the login system, we have a register page, login page, and forgot password page. If you are entering our application for the first time, you need to go to the registration page to create an account, and then use the created account to log in on the login interface, and then enter our application. Below are UML diagrams(Figure 3) depicting the different components of Login System and the classes they are composed of.

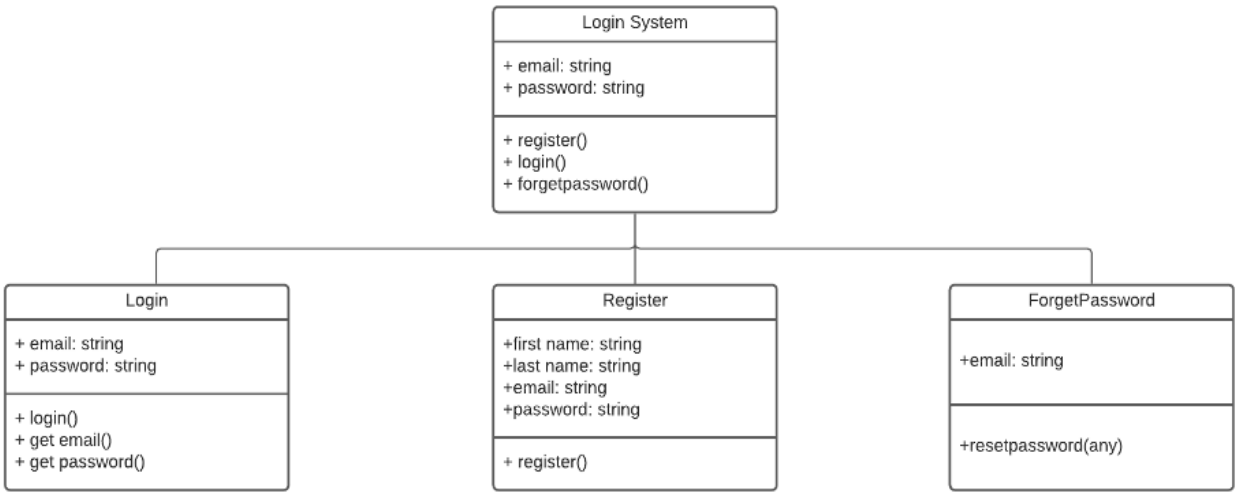

Figure 3. Showcases the interaction between the login system and its various functionality that will allow a user to login, reset a password, and register an account.

UML descriptions:

- The following explains the fields that login system will have:

  Email: An email is used for account login

  Password: A password is necessary for logging into the application and password needs to be six characters or more

  First Name and Last Name: They are used to complete personal profile during registration

- The following explains the function that different pages will have:

  In the registration page, users need to fill in their first name, last name, email, and create a password of no less than 6 characters to complete the account registration. Then the user can return to the homepage to log in.

  In the login page, the user needs to enter the registered account password to log in. Forgot Password option listed on login page.

  In the forgot password interface, the user needs to enter the email address to get the link, and the link takes you to the page to change the password information.

  If you enter the landing page for the first time, you will have a Register Option/Create new user option, and a Login Option/Use already created user option.

4.2 The Front End WEB/APP

This module will consist of the use of our Ionic framework which will be used for our frontend GUI. This framework will help us create an easy to use interface for our clients lab colleagues in which they will be able to record their experiments and protocols as well as interact with the well plate image itself and take notes on certain portions of their work. This will fit well with other components of our application (as seen in figure 4 below) because it will communicate with our Django backend which will be receiving requests from the frontend based on user inputs. These inputs could range anywhere from new user and login credentials which were discussed above as well as receiving input on new groups, experiments, protocols, notes, etc. The ease of use will be important because these lab users already have to record a lot of data as it is and having a user interface with a steep learning curve is something that we did not want them to have to endure. The image below shows how our frontend will be laid out and what components are necessary to have. These sub-pages were necessary because without them the frontend would be massively cluttered and would not be easy to navigate.

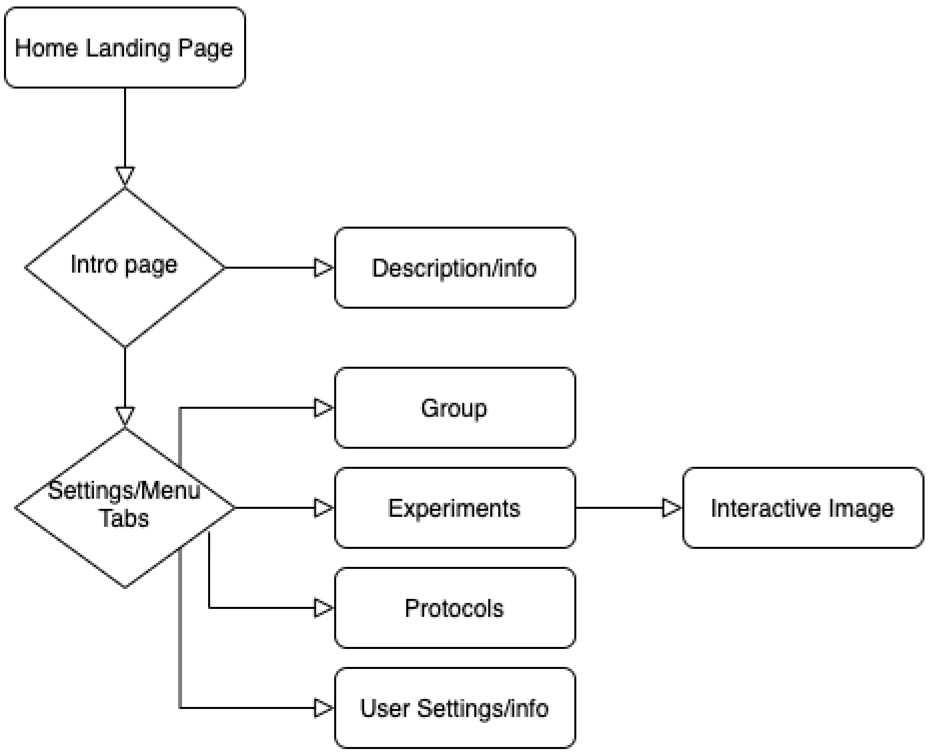

Figure 4. This figure demonstrates the layout that our frontend GUI will have and the process that the users would go through in handling the application.

The services that this component will provide are as follows. After the user has made an account and logged in, as discussed in the previous section, the application will take you to the home landing page (As seen in Figure 4 above). This landing page will consist of a description of the lab and what the application entails as well as info on contacting leads in the lab and the lab itself. This page will also have the ability of taking you to a menu portion of the application. The menu portion will consist of 4 tabs: protocols, experiments, groups, and user/application settings. The protocols sub-component will have a section where you can create a new protocol for yourself or a group in which you will have to input the plate type, number of samples, suspected positivity rate, etc , as well as look at protocols that you have already created. Another sub-component will be the experiments page, where you will be able to create a new experiment where you will have to input what protocols are being used and you will be taken to your interactive image, which will be based off of an existing protocol. Viewing an already completed experiment, or an open in-progress experiment that the user may have already been working on with its information saved. The next sub-component will be the groups page, where you will be able to either create a new group, in which you can add other users, or you can select a group that you have created or been added to. This is an important feature as many of the lab colleagues work together on certain protocols and experiments, so being able to share the work that has been done is a crucial feature. The next sub-component of the front end will be the interactive page which can be accessed after you have created or are in process of creating an experiment from the experiments page, in which users will be able to interact with a well-plate image similar to that of one used within the lab to show and record data taken in the lab. Lastly, the final sub-component that the frontend GUI will consist of is our user and application settings page. This is where the user will be able to view or edit their account information such as name, email, and password, as well as be able to change some application settings on their side of the application. All of the sub-components listed and described above are all necessary to ensure that our client and users will have all that they need in their work environment and will ensure that the application is ready for deployment when that time comes. These sub-components may be different in functionality, but when it comes to the frontend, they all play a vital role in giving the user the tools they will need when in the lab environment and will give them an easier way of recording and sharing information taken while doing their job.

4.3 The Back-end Framework / Database

4.3.1 Back-end Framework

The back-end framework is responsible for defining data models for the database. It is expected to be the definitive source of information on what kinds of data the database is storing, what information those data models contain, and how they interact with each other. The data models that the back-end stores will be as follows (and is demonstrated in figure 5 below, along with the relationship between data models):

- Users
  - Name (First, Last)
  - ID
  - Email
  - Password
  - Organization
- Lab Group
  - Name
  - ID
  - List of members

    - Authentication token-- to invite users to join group
    - Admin
    - Protocols
- Protocols
    - Plate type
    - Number of samples
        - Number of plates
    - Expected positive rate
    - Protocol name
    - Creator UserID
    - Date created
    - Date last used
    - Number and ID of experiments used in
    - Active/inactive
    - Lab group
- Experiments
    - Protocol used
    - Associated images-- well plate image including pools, positive and negative groups
    - Active/completed
    - Notes page
    - Step number within overall experiment
    - Plaintext representation of data, including all of the above data and individual case results

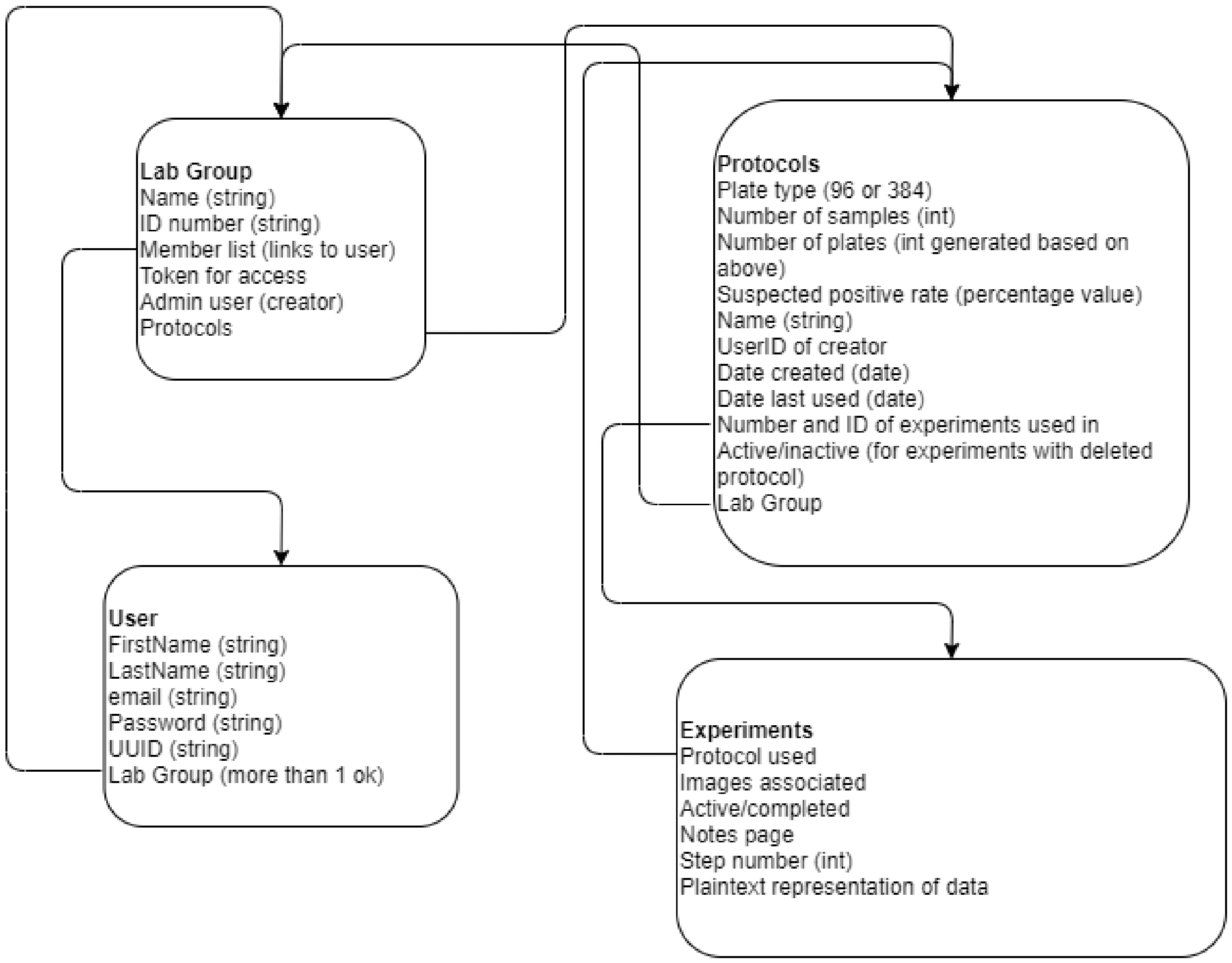

Figure 5. The data models and their relationships to each other.

By defining these data models, the back-end makes clear to the database and REST API what is being transmitted both from the user and from the Python scripts used to perform calculations.

4.3.2 Database

The database, which in this project is a MongoDB cluster, is responsible for storage and retrieval of the data generated by both the user's interaction with the front-end and the Python script. Through the REST API, it can respond to requests for data from the front-end and serve that data, or receive data from the other components of the program and store them for later use. This component is very important to the useability and value of the application, as the ability to store and retrieve data for later use is necessary to do any meaningful lab work.

4.3.3 REST API

The REST, or Representational State Transfer, API is responsible for managing and facilitating communication between the back-end, the database, and the front-end. It both accepts and sends information to and from each of these components so that they can communicate. This component is crucial to this application as it enables communication between all other parts of the application.

4.4 The Well Plate

The associated well plate image editor will be the module that interacts with the well-plate image, which will be implemented through python classes. Doing various edits in real time to update and create the well-plate as the user interacts with it. As can be seen in the below figure (Figure 6) this module is passed information regarding the newly created experiment from the backend Django code. As can be seen in Figure 1 as well, these python scripts will have main interaction with Django code that will support the page holding the well-plate image. This Django page will be linked to the frontend user interface and rest of the application. The data this module needs to properly operate are based on the information provided by the user when creating a protocol and experiment. Data included from the protocol is the type of plate, the number of samples, the name of the protocol, number of plates total, and suspected positivity rate. This information will inform the python scripts of the form of the well plate image that is to be created and displayed to the user. This is done through the createPlate method which will have interaction with the samplePooling.csv file. This is a comma separated value sheet that contains all possible group sizes based on number of samples and the suspected positivity rate. This file will require it to be read each time a new well plate is generated to properly generate each individual well plate image based on the groupings associated with that number of samples and positivity rate. This file was created through modification of the original script that was used to create pooled groups by our project sponsor. The modifications that were made increased the range and totality of the sheet to include more scenarios. Once this data is acquired the groupings will be drawn on the image for showcase to the application user. Associated well plate information will also be generated along with the original plate. This includes using methods such as generateHeader which will be used to create a header based on the protocol name, also the generateTestTubes will create the first grouping of test tubes along the bottom of the well plate image.

After the complete image is formatted and drawn, including both header, test tubes, and well plate it will be passed back to the Django back-end code for display on the page. As the user interacts with the well-plate by touching the individual wells, as this occurs the Django back-end code will use the updateImage function to continually update the image based on the user's interaction. Each updated image will be passed back to the Django back-end code and then displayed to the user on this page.

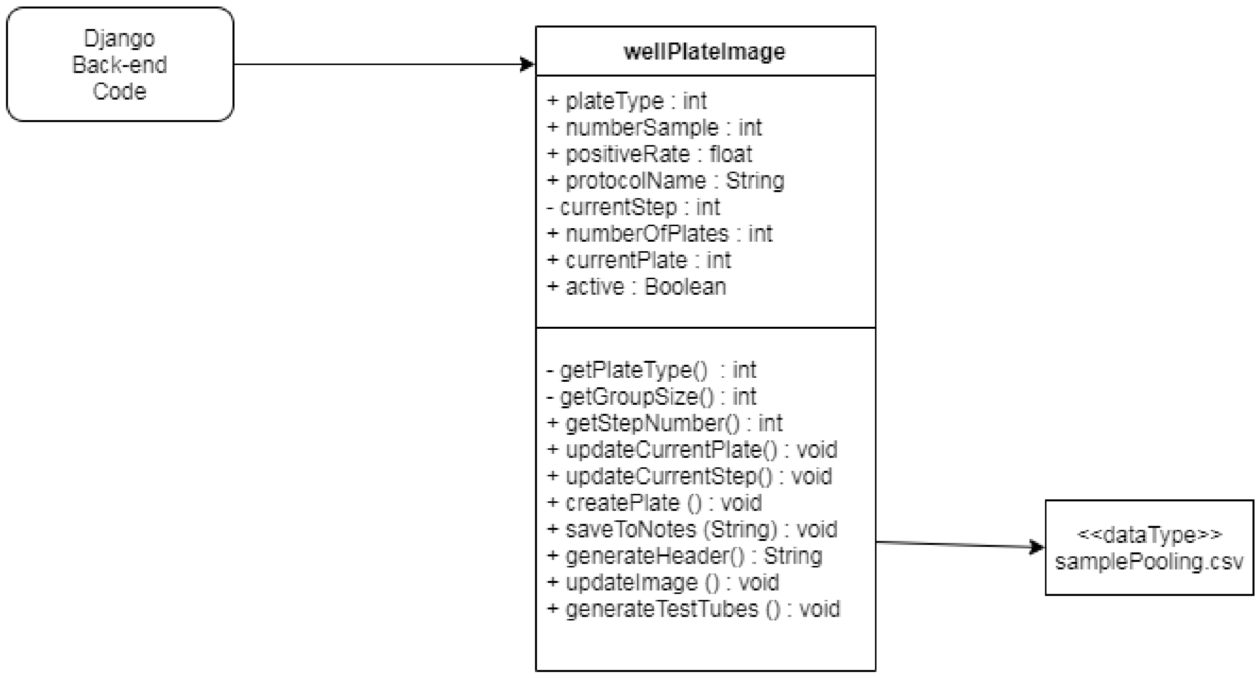

Figure 6. Showcases the interaction between the wellPlateImage scripts and the Django back-end code, as well as interaction with the modified spreadsheet of potential cases

5. Implementation Plan

As a team, we have developed an implementation plan, and we will implement the plan for the rest of the semester. Our plan is divided into four parts: Login System/Interaction (Xiaobai & Yihao), Front End WEB/App (Miguel), Back-end Framework/Database (Bailey) and Well Plate(Tyler). Throughout the project development phase, we have maintained close contact with the clients so that we can continuously deepen our understanding of project requirements and ensure the quality of our projects. For the design and development of these different parts, we have carried out a division of labor within the group.

Our plan is to complete our first generation of products before the first week of March, and then start our product testing plan. Therefore, the person in charge of each part has estimated the completion of their own module content and the time allocation. When all modules are completed and merged, we will have a full prototype tech demo at the end of this stage to show our products.

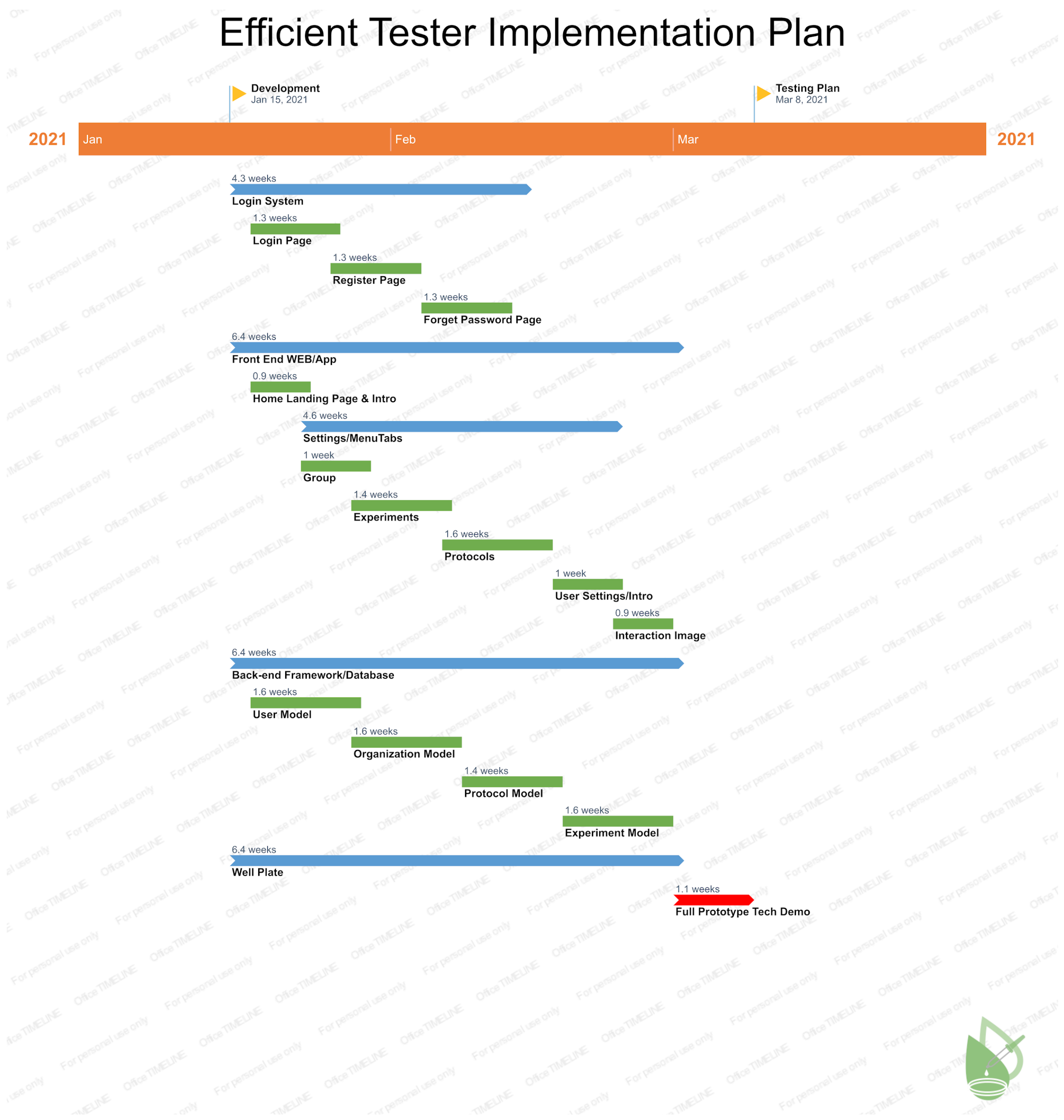

Figure 8. A visual representation of our current progress and our continued plan and timeline for development throughout the semester.

6. Conclusion

The COVID-19 virus has highlighted the need for more efficient grouped testing techniques. Both Dr. Fofanov and Dr. Furstenau have been able to identify that the multiple steps in the process often leads time-pressured lab users to make errors. Our proposed solution of a mobile, tablet, and web application to help walk the users through the protocol with the aim of reducing these human errors. Reducing burden on lab workers by making grouped testing more clearly identifiable and easily visualized is a key portion of our project that will allow lab workers to more easily complete their required tasks. This is the reason why Dr. Furstenau and Dr. Fofanov have tasked us with the creation of an application to solve these problems and bring a solution to the lab workspace.

As we have outlined, our application will have a few major areas of focus. The first being the front end, which will be developed through Ionic, this is the main web application that users will be able to interact with and it is what users will see when using the application. The first page a user will be directed to is the home landing page, which holds information about logging in or finding a lost password and setting up an account. Once a user makes it past this page they will be directed to the intro page and will be able to access the settings tab that allows direction to the rest of the application including protocols, experiments, group and user settings, as can all be seen in Figure 4. The main ionic User Interface (UI) front-end will have interactions with the Rest application programming interface (API) that will be used to communicate with the back-end of our application. This will be the main linkage to get requests and calls, operations of code, from the front end and interact with them correctly to process user actions appropriately. These two pieces of software will allow for us to easily display and transmit data between different pieces of the application. Meaning we can quickly gather data from the user about their well plate and the Django backend code and the database through MongoDB. This interaction is key in accessing the data stored in the database such as the user information, group information, protocol and experiments. The Django backend will allow us to more easily interact with the python code, which generates and translates user interactions into information that can be used to update and change the main well plate image that is displayed to the user. The MongoDB will consist of the main data storage and handling, everything from the different protocols and experiments that are created by the users to individual user and lab group information, this includes the results of each experiment with images and wells containing positive cases.

Overall, this document shows the main interactions between different pieces of software, and what type of interactions they will be. It also outlines the classes and necessary methods that will be used. This is useful as it will give us a good foundation and guide towards programing our final version of the application, as well as making sure all the pieces link together. This information came from both our weekly meetings with our project sponsor and within our team, as well as the information from our requirements document which showcased many of the necessary components for completion.

As we move forward, we may have minor speed bumps, but we believe that our way of trying to complete tasks in parallel, as can be seen in Figure 8, will help us to progress and finish this project in a timely manner. We plan to continue progression with development up to and begin testing on the second of March. Our current to date progress includes addition of the four different models; protocol, user, group and experiment, we will be using for data storage in our database. We have also linked our front end Ionic with the rest API and the database, mongoDB, as well as the Django framework. We have also begun the process of creating all the python scripts that will be used for generating and updating the well plate image.