Pandemic Processing

Requirements Document

(Version 1.3)

December 10th, 2018

Project: Epidemiological Modeling Portal

Team Members:

Anthony Schroeder

Joseph Eppinger

Tanner Massahos

Sponsor:
Dr. Joseph Mihaljevic

Mentors:
Dr. Eck Doerry and Jun Rao

Accepted as baseline requirements for the project:

_____	_____	_____	_____
Team Lead Signature	Date	Client Signature	Date

Table of Contents

1 INTRODUCTION

Infectious disease has been a problem throughout all human history. For example: The Plague of Justinian in 541 which lasted over 200 years and resulted in an estimated 25-50 million deaths, and the Black Death during 1347-1350, which killed ~60% of the European population. Compare this to a modern epidemic: The West African Ebola Outbreak, which resulted in the deaths of 11,310 individuals. It is clear that there has been a dramatic change in the number of casualties due to infectious disease when compared to plagues of the past, and one reason for this is the development of the science of epidemiology.

Simply put, epidemiology is the study of infectious disease and how it spreads within a defined population. Epidemiology assists with lessening the numbers of deaths due to infectious disease via models. Models are predictive mathematical formulas that show how disease can spread through a community. Given a set of initial conditions, a model can generate predictions of key outcome variables, such as how the fraction of the population that is infected changes overtime. These models ultimately are an attempt at predicting how a disease will affect a population given what epidemiologists know about it.

These models can differ in style such as deterministic mathematical models or complex spatially-explicit stochastic models. Epidemiological modelling allows:

1. Scientists to create models of infectious disease given a certain set of assumptions and data
2. Based on these models, scientists can predict optimal vaccination times and other preventative measures

Epidemiologists also apply various methodologies and base their models on varying assumptions, meaning there could be a handful of differing models for one infectious disease all of which could generate different optimal vaccination times and predictive preventative measures.

Many of these scientists are creating these models in parallel, and must discuss them in order to determine which is the most accurate. This deliberation component is key in that it ensures the optimal model is chosen for a given situation. The best model could be any one of the models proposed, or even a new hybrid created from the discussion.

This is where the major problem arises, according to our client Dr. Joseph Mihaljevic, an epidemiologist himself and assistant professor at NAU SICCS. There is a lack of an efficient system where epidemiologist from all around the world can

critique and discuss one another's models in order to receive an optimized model that generates the best predictions as quickly as possible, in turn saving the largest amount of lives.

Our solution to alleviate this problem is the Epidemic Observation Network or EON for short. EON will be a secure, fast, and user friendly web application where epidemiologists from all around the globe can share and discuss their models.

2 PROBLEM STATEMENT

The modelling of infectious disease is a rather well-defined process. Each time an epidemiologist wants to model the outbreak of infectious disease, they step through the following process:

1. **Collect data via data sets or researched data:**
   The epidemiologist gathers data about the population and how the infection spreads from a mathematical standpoint.
2. **Model data using C, R, or Python:**
   Using this data, the epidemiologist crafts a model using a series of mathematical equations which models the infectious disease. This model is then implemented as code so that it can be rerun quickly with varying parameters.
3. **Develop the best model through discussion:**
   Next, they confer with other epidemiologists on what the best model looks like. This discussion produces a model that should most closely approximate the real-world spread of the disease.
4. **Build predictions based on the model:**
   The epidemiologist then predicts the best possible course of action for minimizing the damage done by the disease, and proposes preventative measures to the authorities who can act accordingly.

It is not that this current workflow is at its core broken, it is simply that it could be further optimized. In the world of epidemiology, there is a lack of an efficient system where epidemiologists from all around the world can critique and discuss one another's models to arrive at an optimized model that generates the best predictive measures. Some specific problems with this workflow include:

- Epidemiologists must wait until full publication to view models of other epidemiologists they are not personally acquainted with.
- Deliberation is difficult when no public space for it exists. Currently, epidemiologists can only communicate through emails and GitHub comments.

- Communities of epidemiologists are isolated clusters, as in they only tend to communicate with those that they already know well, which reduces the extent to which deliberation can produce new ideas.
- Current discussion and models are not extremely public or easily accessible. An epidemiologist must publish or actively send their data to authorities before action can be taken.
- Difficult to communicate results with managers and public health officials.

By streamlining this key step in the workflow, EON will provide an invaluable missing resource for epidemic management. Specifically, it will streamline comparing and discussing models to come to a faster consensus. EON will allow epidemiologists to quickly and efficiently generate models, compare and discuss models, and arrive at an optimal model so they can create preventative measures to combat disease outbreaks.

3 SOLUTION VISION

We have been working closely with Dr. Joseph Mihaljevic throughout our developmental process in order to develop a compelling vision for a robust and user friendly web application which further optimizes the general workflow of an epidemiologist and potentially saves millions of lives. EON will allow epidemiologists to:

- **Share models with the community**
    - Method: EON will be a place where epidemiologists can post their models' code for others to view and discuss.
    - Purpose: This will allow for quick critiques and harbor discussion. These critiques and discussions will lead to an optimized model which will generate the best predictive measures.
- **Decide how their models appear to viewers**
    - Method: EON will allow epidemiologists to decide how public their models are. For example, a scientist might want to keep their code private, but the final graphical output is fine for viewing.
    - Purpose: Some scientists don't want to share their models or data due to publication purposes so we want the user to have the option to make a post public or private.

- **Interact with and provide feedback on other user's models**
    - Method: EON will provide a discussion forum for epidemiological models. Additionally, users will be able to modify and re-upload each other's code.
    - Purpose: This builds a sense of community and allows users to modify assumptions and visualize how it changes the output.
- **Discuss future models without a fully developed model existing**
    - Method: EON's forum will not just be for models which already exist. Users can create discussions without any working prototype developed.
    - Purpose: This allows users to simply post an in-progress model for discussion, meaning no code is required upon upload.
- **Edit and share the code used for models in a GitHub type of fashion**
    - Method: EON will support version history and cloning of the model code.
    - Purpose: This will allow users to view and edit the code used to generate a model and create new models based on old ones for further critiquing and discussion.

With universal access via any browser, a web application provides an easy and intuitive way for these epidemiologists to connect with one another and speed up and enhance the overall workflow. The only core requirements to use this application is a working Internet connection and a web browser. The next few points of discussion are roughly what we plan on providing to the client alongside further clarification about the features and architecture of EON.

What data does EON generate and how is it used?

The system will request that the user develops model code in such a way that it ultimately outputs a single CSV for usage by the graphical interface. From this CSV, a visual will be generated and displayed to the user. The CSV, user model code, and visual will all be stored accordingly as the user model code will need to be referenced when a new user attempts to make a modification.

EON System Architecture

The system architecture of EON is critical when it comes to generating a secure, fast, and user friendly web application where epidemiologists from all around the globe can share and discuss their models. Ultimately the technologies we have heavily researched and selected will influence how we implement our requirements.

The model in Figure 3.1 is how our components will interact with each other. We chose to utilize the Django web framework as from research it has proven to be extremely flexible and robust. Django will tie both the front end and back end together creating a cohesive environment for both users and developers. To store all the user data such as code, models, comments, and user accounts, we have selected PostgreSQL as database back end due to its built-in security, speed, scalability, and reliability. Lastly, we have selected Amazon Elastic Compute Cloud (EC2) as our hosting solution. EC2 is known to be secure, resizable, and portable.

The typical workflow of a user uploading a model begins with a request to the Django web application, where the user uploads model code and configuration information. After the user submits this form, the model code and configuration data is sent to the EC2 server where it is tested. If successfully ran, a CSV is outputted and important data pertaining to the model is sent to the Django web app to be shown to the user and properly stored within the PostgreSQL database.

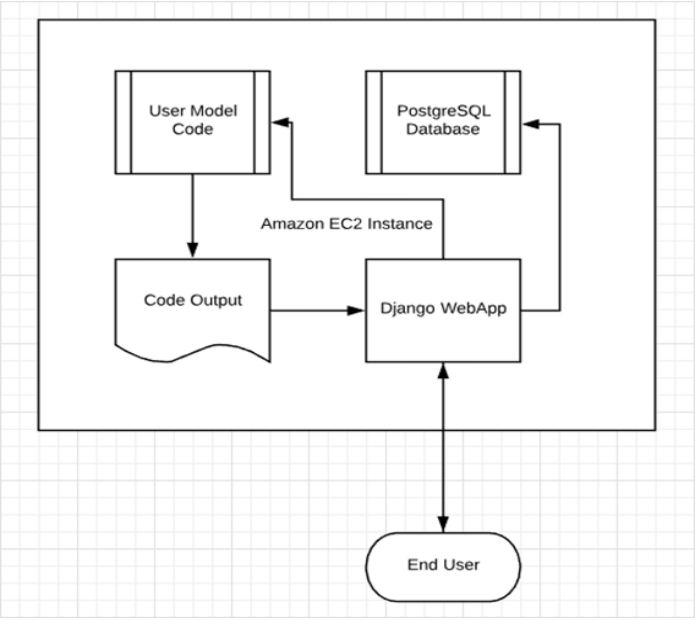

Figure 3.1 A simple model of our system

The next section outlines specific requirements pertaining to functional, performance, and environmental specifications.

4 PROJECT REQUIREMENTS

The application will require a number of specific features in order to function. As such, these requirements have been distilled into several tiers. Each level of requirements is derived from the level above, and the environmental constraints are identified based on the domain requirements.

Each level is identified by a heading where 'DR#' stands for domain requirement, 'FR#' stands for functional requirement, 'PR#' stands for performance requirement, and 'EC#' stands for environmental constraint.

Domain-level Requirements

These are requirements which specify EON most broadly. These are the highest level of requirements and features which must be implemented for a working application. These have been obtained and developed from numerous meetings with Dr. Mihaljevic.

- **DR1 Maintain User Accounts:**
  User data must be stored securely in order to verify specific user models are protected and they have the ability to share with who they would like.

- **DR2 Custom Model Storage and Running:**
  The web application will store compiled code for future runs with modified parameters that are specific to the configuration form the user filled out when posting the model.

- **DR3 Code and Project Version Control:**
  Users can create open and private projects. Model owners can decide who can access and perform version control operations on their models.

- **DR4 Multiple and Future Language Support:**
  The application must support multiple languages, since researchers may use a variety of them. Modularity is essential for future scalability and support of

new languages. The base languages which must be supported are Python and C.

- **DR5 Modifiable Graphing Layout:**
  Users will be able to control how their final output appears to other users. The owner of a model will decide what type of graph and how the data is mapped to the graph upon model creation. Other users can then modify and adjust the parameters of the model to see new output.

- **DR6 Discussion Forum:**
  Registered users can discuss their models using a forum system. Additionally, they can discuss future models or ask questions of other epidemiologists using this forum.

4.1 FUNCTIONAL REQUIREMENTS

These functional requirements specify the exact features which must be included. These were developed by breaking down the above domain requirements into smaller components. The functional requirements will be broken into high level categories, and further elaborated on through low level requirements.

DR1.FR1 Secure User Accounts

- Users accounts will require a username, password and email for creation.
    - A verification email will be sent to the user before their account can post models and comment in discussion forums.
- Users can decide to privatize any part of their profile. Privatized portions are not visible to other users.
- Users can stay logged in so long as they have cookies enabled.
    - The application will store session information for the user.
    - Cookies expire after two weeks.

***DR1.FR1.1* Store User Information Securely**

- User information will be stored in the PostgreSQL database.
- Passwords will be encrypted for additional security.

DR1.FR1.2 Account Information Specifics

- Passwords must be secure and alphanumeric with standard symbols (!@#$%^&*).
    - Passwords must have one lowercase letter, one uppercase letter, and one number.
- Usernames will have a ten-character maximum.
- Biographies will have a 2000-character maximum.

DR1.FR2 Modifiable Account Information

- Users accounts will be modifiable and the ability to store other user information in their profile which is not required during creation.
    - At minimum, the user can assign themselves an institution, profile picture, contact information, and a short biography.
        - These values can be changed in the future by the user, but each change requires a password resubmission.
    - Other information may be stored, but is not required in a minimally viable version of this web application.

DR1.FR3 Administrative Privileges

- Certain user accounts will be able to moderate forums and models in order to keep discussion civil and appropriate.

DR1.FR3.1 Admins and Moderators

- User roles must be implemented to identify users as administrators and/or moderators.

DR2.FR1 Model Uploading

- Users will be able to upload their model code to the web application.
- The models uploaded will be stored in a flat file which is mapped to within the database.
- Users can edit or remove their models at any time they desire.

DR2.FR1.1 User Determined Model Parameters

- Users decide which parameters of the model code are modifiable through the graphical view.
- Some parameters are static and unchangeable.

DR2.FR2 Model Running

- Users can run their own model code and, depending on permissions and visibility, other users can modify the model and view the graphical output.
- Users can change the parameters for running a model at the owner's discretion. Ex: They may be able to change the starting population, but not the starting percentage infected.

DR2.FR2.1 Model Output

- Models output will be temporarily stored in a CSV which is interpreted by a graphing engine.
- Default model output will be cached for quicker loading of the model page.

DR2.FR3 Model Navigation

- Users can search the models stored in the web application by:
  - Popularity
  - Recently uploaded
  - Disease type
  - Epidemic ID
  - Time frame
  - Searching by keywords

DR2.FR3.1 Model Keywords

- Models can be tagged with certain keywords which are relevant to that model.
- Users will be able to define what these keywords are.

DR3.FR1 Model Management

- Models can be made private by their respective owners in order to restrict public access.
    - Owners can allow specific other users to view or modify private repositories.
- Model owners identify how users can interact with their project by deciding who can:
    - View
    - Comment
    - Create notifications
    - Modify code

***DR3.FR1.1* User Groups**

- Model owners can specify different groups with different permissions with regards to their models. They can then add any users they want to these groups.

DR3.FR2 Model Version Control

- Model owners and specified users will be able to perform Git-like operations such as:
    - Changing the code and re-submitting the same model with a history of the code.
    - Downloading the code.
    - Cloning the code into a new model.

***DR3.FR2.1* Change Comments**

- Each time a user changes the model, they must post a comment describing their changes.

***DR3.FR2.2* Documentation**

- Models can have associated documentation which is visible at the owner's discretion.

DR4.FR1 Multiple Language Support

- The web application will support models written in:
    - C
    - Python
- While less important, integrating the following languages is helpful, but not necessary:
    - R
    - C++

***DR4.FR1.1* Library Support**

- The application must identify which libraries are required for running certain models and install those libraries as necessary.

DR4.FR2 Modularized Language Support

- It should be easy to extend support to future languages by simply installing the required files and making appropriate connections to the application.

DR5.FR1 User Chosen Graphical View

- The model owner decides how the graphical representation of the model appears. The chosen representation will change based on the model; whichever the owner decides is most relevant will be chosen. These options are:
    - Line graph
    - Scatterplot
    - Geographical map (Not required, but possible if there is time during development)
- The model owner can decide how many graphs the data is output to, and each will appear in a different tab.

***DR5.FR1.1* Graph Specifications**

- The axes for the graph are defined by the model owner.
- A line of best fit can be determined by the model owner.

- The user will also be allowed to layer on static data, not generated by the model.
    - Restricted to lines and points

DR5.FR2 Dynamic Graphs

- The graph should update dynamically when the user submits their initial input values for the model.
- The input ranges are specified by the model owner.

DR6.FR1 Forum Structure

- Moderators and administrators define topics for the forum.

DR6.FR2 User Posts

- Users can create new posts under any of the topics for the forum.
- Users can comment under any post in the forum.
- Users can insert mathematical equations into their posts.

***DR6.FR2.1* Post Formatting**

- Posts will be formatted using a simple markup language.

***DR6.FR2.2* Equation Formatting**

- The application will include an equation editor which will allow users to craft equations for their posts.
- Users should be able to post equations containing:
    - Sums, integrals, and derivatives
    - Standard arithmetic operations (addition, subtraction, multiplication, division, powers, logarithms)

***DR6.FR2.3* Markup Language Specifics**

- Forum markup language may specify text:
    - Size: Ranging from ten point to seventy-two-point font
    - Font: Verdana, Times New Roman and Arial

- Text styles: Bold, italics, and underline
- Left-aligned, right-aligned or centered text
- Bullets and numbering

DR6.FR3 Model Post Linking

- Each model will have an associated model discussion thread.
- The model owner decides which topic the model forum post is placed under.

4.2 PERFORMANCE REQUIREMENTS

The following performance requirements outline the quantitative components for the previous functional requirements. Each of these are related to the functional requirements which fall under the same domain level requirements. These were determined based on the realistic explanation of our client and our best judgement.

DR1.PR1 Fast Account Creation

- Account creation should take ≤ sixty seconds for 90% of users.

DR2.PR1 First Code Run Speeds

- User code will run in ≤ ten seconds during the first run.
- Running user code in sequential runs should take no longer than seven seconds.

DR3.PR1 Git Operations

- Cloning a repository should take ≤ five seconds.
- Committing changes to a repository should take ≤ seven seconds.

DR4.PR1 Adding New Libraries

- Adding a new library should not add more than five seconds to the initial runtime of the code.

DR5.PR1 Graph Loading Times

- Displaying the default graph should be done in two seconds.
- Displaying user models should take no longer than eight seconds.
- Current models in progress should have an indication to let the user know the request is being ran.

4.3 ENVIRONMENTAL CONSTRAINTS

The following constraints outline the pre-existing constraints that the application must adhere to. These were identified based on conversations with Dr. Mihaljevic about the existing flow for modeling, and on consequences of the medium chosen for the application.

EC1 Multiple Programming Languages Supported

- The application must support the myriad of languages which epidemiologist's user for their models, and support future languages.

EC2 Uniformity of Uploaded Code and CSV Output

- Models are written in single languages and only output CSV files.

EC3 Fully Supported Browser

- The three most popularly used web browsers must minimally supported:
    - Chrome
    - Firefox
    - Internet Explorer

5 POTENTIAL RISKS

There are some risks associated with developing this application. These are a threat to the well-being and performance of our solution. These risks range from low to high severity and likelihood. We expect high likelihood risks to occur often, medium to occur infrequently, and low likelihood to occur almost never. High severity risks are application threatening, medium severity threats threaten the usage of the application, and low severity threats affect user experience. We have taken measures, or plan to take measures, to mitigate the effect of these risks.

Risks Overview:

Risk	Severity	Likelihood
Individuals uploading/downloading malicious code	High	Low
User's programming language not supported	Medium	Medium
Server overloaded with running models	Medium	Low
New users may find the platform challenging to use	High	Medium
Users wish to maintain IP rights to code	Medium/High	Medium
Singular point of reference	High	High

Risk Mitigation:

1. **Individuals Uploading/Downloading Malicious Code**

   The ability to upload code and run local simulations on a server gives users the potential to upload malicious code that would perform undesired operations, such as mine cryptocurrency or install malicious software. We plan to build a black box to contain/validate any malicious code, along with the capability to report code for malicious intent.

   As we expect a smaller user base, the chance for someone to attempt to abuse the system is low, but not improbable.

2. **User's Programming Language Not Supported**

   Within the ever-growing field of data science, different programming languages may become popular in the future. We plan to implement a modular design to allow for future languages to be supported.

The chance someone has a preferred language of preference is medium. There are many languages out there, but this application will aim to support the most popular ones.

3. **Server Overloaded with Running Models**

   As the user base grows, the demand for models to be run increases. We need some way to reduce the overhead demand while running these simulations.

   By creating a table to store different pre-generated results for each model, repeated function calls will be reduced that may otherwise tax the server. A stretch goal is to include a cache in the browser that records similar results for requests by a user to speed up modeling responsiveness.

   The chance for the server to be overloaded with simulation requests is high as users might tab through simulation models several times, sending several dummy requests creating an artificial demand on the server.

4. **New Users May Find the Platform Challenging to Use**

   As technical demos and websites give a lot of capability to users of a platform, the first few posts might be difficult for some users. To get users more comfortable with the forum, we will include tutorial videos and discussion posts explaining how to get started. This will help new users understand the different components and resources on the platform.

   User retention is an important component as users need to feel comfortable using the platform. This means we need to explain how to use the platform, and accept suggestions from users who have ideas about making the platform more user friendly.

5. **Users Wish to Maintain IP Rights to Code**

   Some epidemiologists might wish to keep their code and implementations secret to avoid plagiarism. Users will have the option to keep different aspects of a project private until being published by an organization.

   If users still wish to avoid uploading code, they can upload a CSV files in place of code to protect personal work, and use the graphing system to show the capabilities without risking their code being stolen or plagiarized.

   Groups might also wish to keep project information within a set of contributors.

   This is a medium/high severity, because some people working on projects wish to keep their work private from others until they are finished.

6. Singular Point of Reference

Throughout this project, we have only surveyed our client and might not have a full understanding of what epidemiologists want from an online modeling platform.

We plan to interview and have user testing for more suggested requirements and capabilities.

This is highly important, because with different perspectives and opinions of EON, we can have increasingly better insight to common issues and requirements. With an increase in usability, we can generate a better product by providing the tools needed by a larger quantity of epidemiologists.

6 PROJECT PLAN

We have segmented our project into different milestones to indicate the progress and the section of the project that we are currently working on. The most important components of the project must be prioritized first. Additionally, the application should be ready for user testing early so that there is plenty of time to see how users handle the application and adapt it as it is built.

Small tasks and general topics will be delegated between group members.

Figure 6.1 is a schedule for our big picture milestones:

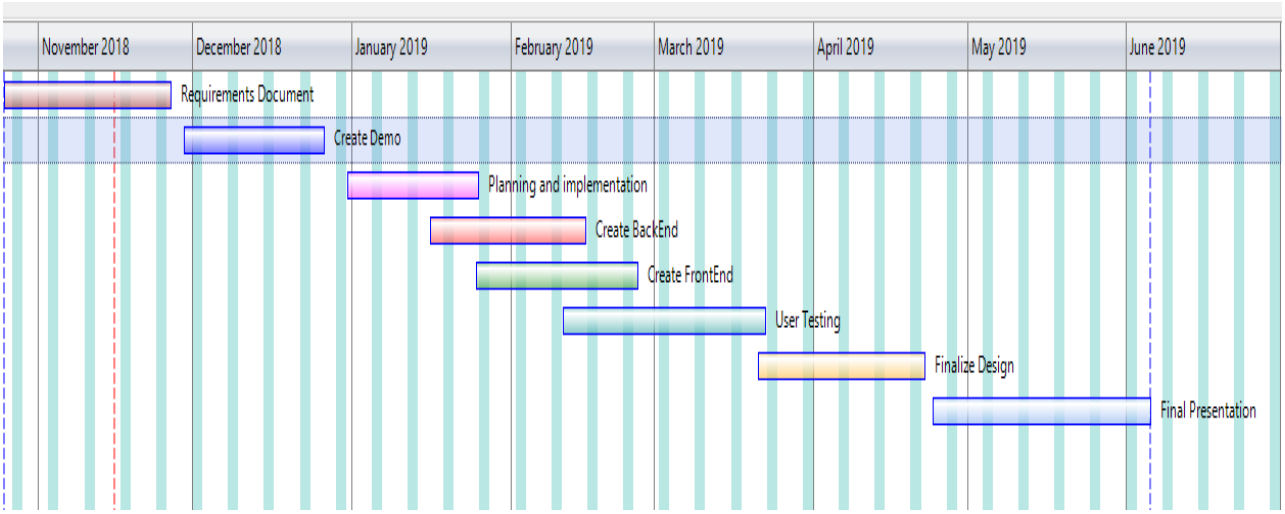

Figure 6.1 Predicted project timeline

As shown in Figure 6.1, the following schedule is planned around several major development phases.

Milestones:

1. Create a demo to show that the most important challenges are being addressed:
   Nov. 2018 – Dec. 2018

    - Create a minimum viable product (MVP) that will allow us to show a proof of concept, demonstrating that users can upload code, and generate an interactive graph for others to use.

2. Planning and implementation for remainder of the product:
   Dec. 2018 – Jan. 2019

    - Plan the actual implementation and workflow of the developing the application and discuss design decisions, including but not limited to database, page structure, and any potential development issues.

3. Create the back end for the application:
   Jan. 2019 – Feb. 2019

    - The application must be able to handle given user models, and process them.
        - Version control implementation for model uploading and modification.
        - Database for storing the following information:
            - Forum topics, threads, and comments
            - Model data
            - User profile information
        - Black box implementation for validating user model code.

4. Create an appealing front end:
   Jan. 2019 – Feb. 2019

    - Create the user interface that will utilize the back end.

- Create user account and management system for forum posts and model uploading.

5. Test the application to ensure it fulfills our requirements:
   Feb. 2019 – Mar. 2019
   - Have a small group of epidemiologists use the platform and answer surveys about the web application.
   - Verify that requirements are satisfied and provide a usable experience.

6. Finalize design of product and prepare for delivery:
   Mar. 2019 – Apr. 2019
   - Commit any changes that has not been fulfilled and if time remains accomplish stretch goals.
   - Used for extra requested features and lagged schedules.

7. Final presentation and completion of project:
   Apr. 2019 – May 2019
   - Finalize product and deliver it to the client.

7 CONCLUSION

Infectious diseases are a major health challenge and affect modern day society. Hundreds of thousands die every year from preventable diseases. The use of data-driven disease modeling in modern epidemiology can dramatically reduce the number of lives lost. A model must be created of a particular disease in order to properly plan, and epidemiologists construct these models so that they can determine the optimal preventative measures to be taken. The problem is that in creating these models, these scientists have a difficult time exchanging information before publication. Thus, EON will bridge this gap in the model development process by giving epidemiologists a public place for them to present and review their ideas. EON will become a public repository for epidemiological models, and a forum through which those models can be critiqued and refined.

This document has outlined the specific requirements necessary for EON so that it can adequately solve the issues which plague the epidemiological model constructing process. These requirements have been gathered by meeting with Dr. Joseph Mihaljevic and discovering what the problems are with the workflow.

EON is integral in an age where information spreads quickly and diseases move even faster. EON will be the solution that can revolutionize epidemiology and could be applied on an immense scale and generate a large future impact. It has the potential to save millions of lives around the world.

8 GLOSSARY

Amazon EC2 - Amazon's creative computing cloud: An elastic computing environment which can host websites and web applications.

C - A popular low-level compiled language with many powerful tools

C++ - An extension of C which implements object oriented design philosophies.

Django - A web framework built using Python and typical web technologies used for a variety of different web applications.

GitHub - A web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features.

IP - Intellectual Property: Works which are protected from plagiarism in the United States.

Plotly - An open source graphing library for python.

PostgreSQL - An ACID compliant database management system. ACID compliant means malformed queries are not submitted, ensuring data security.

Python - A popular multipurpose interpreted programming language.

R - An interpreted programming language popularly used in data science.