5/9/2018

Final Report

Team U.I. Fit – Version 1.0

Team Members:
Charles Chatwin
Matthew Burns
Tanner Brelje
Joshua Gutman

Sponsor: Dr. Abolfazl Razi
Mentor: Dr. Abolfazl Razi

TABLE OF CONTENTS

INTRODUCTION

The field of molecular biology is an ever advancing science that requires tedious experimentation and research. In order to generate concrete research results, many molecular biologists must place precious time and resources into large-scale experiments. These experiments are used to show the process of biochemical molecular interaction, or in layman's terms, the result of the reaction between two or more molecules. In order to aid this meticulous process, Northern Arizona University graduate Brandon Thomas, in tandem with an experienced team of graduate students and molecular biologists, created the program BioNetFit. BioNetFit is a command line tool that was created to provide a fast and easy method to simulate complex molecular bonds. These simulations allow researchers to later run the tests in a real environment in a way that gives them a degree of confidence in the expected interaction.

Built on top of BioNetGen and NFSim, BioNetFit allows researchers to simulate a single experiment many times with a range of parameters. The results of each of these simulations is compared to an experimental results file that the researchers upload. Because the combinations of different parameters scale non-linearly, various methods are used to select the best combinations of parameters, including genetic algorithms, simulated annealing, and a few others. After the simulation has been run many times (usually thousands), BioNetFit creates a final output file that shows information about the molecules involved in the reaction.

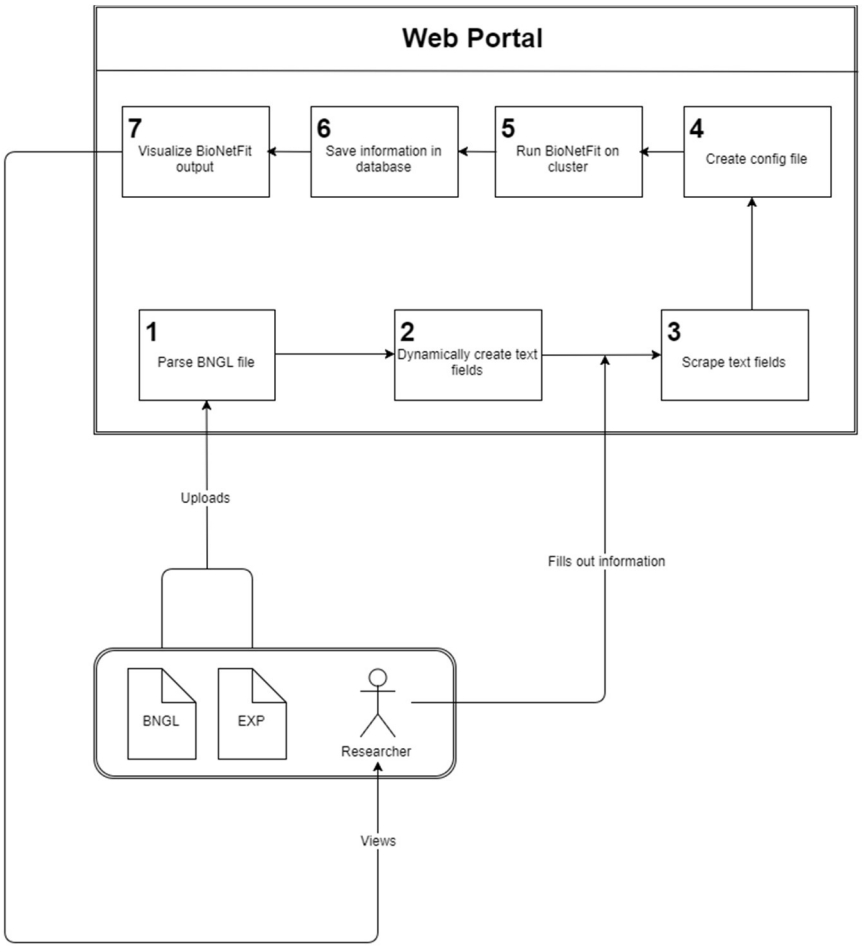
While the program is incredibly useful, its implementation is not user friendly. It exists as a unix-only, command line tool. Researchers who are not technically proficient will struggle at getting the program to run and will attempt to find other avenues when faced with having to spend time learning a new system to run their tests. Additionally, the results currently output by the program are difficult to digest, and do not lend themselves easily to analysis for data collection by researchers. Without much labeling or clear direction in the results, researchers will have to extract results from a command line output, which as previously discussed, is problematic for many scientists. Lastly, the program cannot run large scale experiments in a short amount of time. This lessens the practicality of the program as the experiment becomes more and more grand in scale. In order to make BioNetFit the stellar application it can be, these issues need to be addressed. If done correctly, the program could see widespread use in molecular biology labs all over the world.

Created in order to tackle these challenges, Team U.I. Fit, composed of Charles Chatwin, Matthew Burns, Tanner Brelje and Josh Gutman, will develop software solutions in order to turn BioNetFit into the powerful program it can be. Led by client and mentor Dr. Abolfazl Razi, our team will aim to:

- Create an attractive and simple web portal for BioNetFit.
- Visualize the results of BioNetFit into easily digestible data for researchers.
- Implement parallelization in order to run large experiments on a computer cluster.

Firstly, our team will develop a simple and effective Web 2.0 Graphical User Interface that will house the BioNetFit software. This web page will be easily accessible and easily usable by any researcher who wishes to run tests using the BioNetFit software. In the web portal, the user can either create their own BNGL file from

scratch, or upload one that was either previously downloaded, or was saved remotely on the website itself. From here, the user can run the file either locally on their own system or, in the case of a large file, run the file on the computing cluster Monsoon, available on the Northern Arizona University campus. From there, the program will output a configuration file that will be visualized to the user using graphs and charts. Finally, the user can either save and visualize the outputs on the website, or download them to their own local machine. The BNGL files can then be tweaked if wanted, and the experiment can be run again.



In this document, we will overview the entirety of the software development process. We will begin with providing an overview to the process, in order to help readers gain a depth of understanding of the project's purpose. We will then move on to discuss the requirements that the software product, and architecture and design of the solution. Additionally, we will delve into the implementation of the software, the technology we used to complete the development process, and the testing methods used in order to assure the software worked efficiently. Lastly, we will overview the timeline that the project underwent, and the future works that could sprout from the product that we have delivered.

PROCESS OVERVIEW

To begin the development process, we had a fairly clear, albeit informal idea of how we wanted the software to be constructed. After working with our client, we had laid out a blueprint as to what we wanted our website to look like and began building. Once we had a stable foundation for the software, we then began adding features as needed to the software. Because our client had many different ideas that he wanted implemented within the software, these changes proved to be sporadic, yet effective. Overall, while we may not have had an official development plan, the development itself progressed without much of a hitch. This process worked fairly well for us, as this was a difficult semester for many of our team members. With each teammate engulfed in a mixture of difficult classes, multiple jobs, and personal issues, having a less structured model for development helped a lot. Each member of the team was able to use their free time to make improvements as needed, and the software took shape in small portions over time, rather than in large chunks. At the end of the day the software was implemented well, so we are fairly satisfied with our process.

Below, you can view the members that make up Team U.I. Fit, and the original jobs and titles that were assigned to each of the members:

Charles Chatwin
Team Leader, in charge of communication, web development, and requirements acquisition.

Joshua Gutman
Lead Programmer, in charge of release management, and visualization creation.

Matthew Burns
Web designer, in charge of generating the user interface, and other web based functionalities.

Tanner Brelje
Documentation designer, in charge of editing and writing documents, as well as database design.

Code Management:

In order to manage the code base, we used the popular online coding repository site GitHub. In the initial phases of the project, we created a repository to store the testing and proof of concept code that we had written in the early stages of development. In the second semester, we created a new repository to hold the code base for the website itself. We used this repository to ensure that everyone was testing and developing using an up to date version of the website.

The repository for the project can be viewed at https://github.com/JoshGutman/BioNetWeb

REQUIREMENTS

The project requirements, which specify the features of the project, as well as their methods of implementation, are detailed in this section. They will be discussed first at a high level, and then will progressively increase in detail as we continue throughout the paper. In order to generate these requirements, we held multiple meetings with our client in order to gain an understanding of what was required of the web portal. Eventually, we agreed on a list of requirements that were expected of the minimal viable product. The domain-level requirements, which lay out the features an end-user can expect in the final product, are as follows:

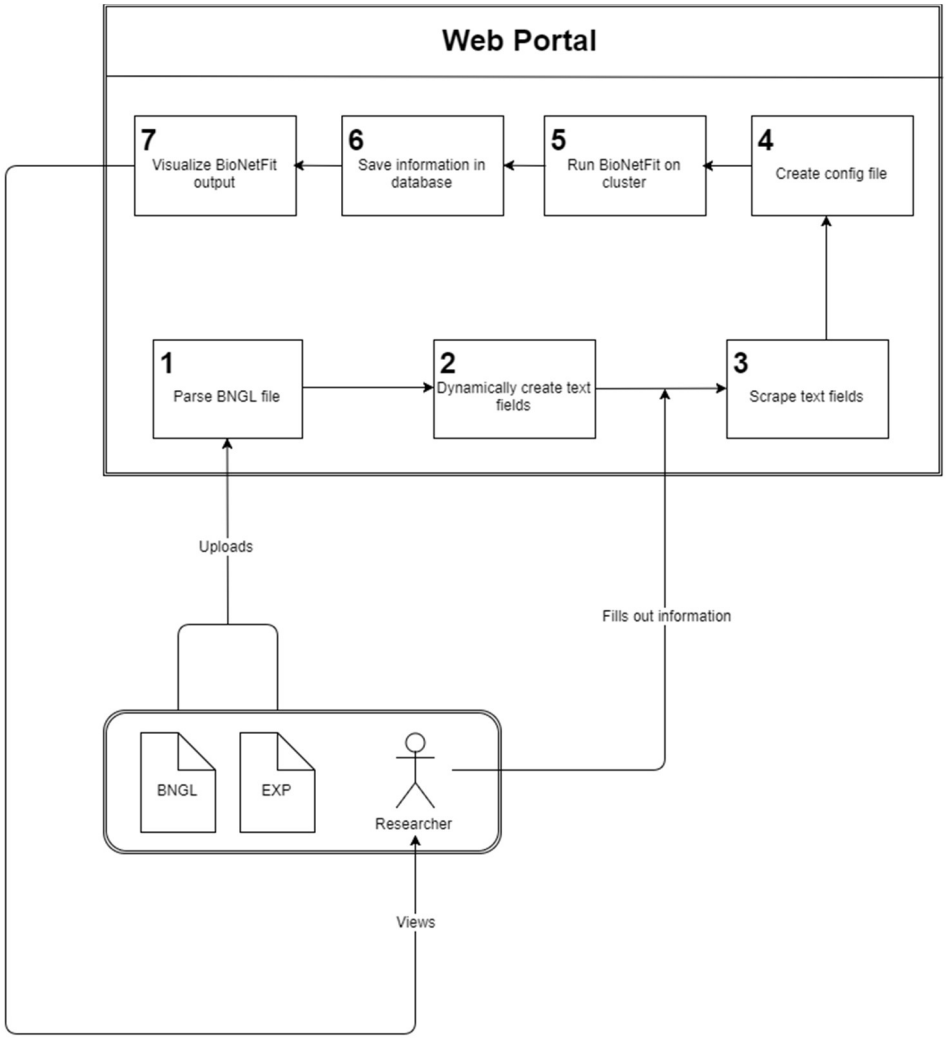
1. Generate config files from BNGL files
    - The web portal must be able to generate config files via and easy to use to form, and run experiments based on both the uploaded .bngl and .exp files, as well as the generated config file.
2. Run BioNetFit on a cluster or server
    - The software must be able to make contact with a cluster computing system, preferably the NAU Monsoon cluster, in order to run large scale assignments in a reasonable timeframe. At the very least, a proof of concept for cluster connection must be shown.
3. Visualize the output of BioNetFit
    - Create 3 different types of visualizations for the outputs of BioNetFit. These visualizations will include best fit diagrams, average permutation diagrams, and fit-value for best and average observable diagrams.
4. Save all information related to BioNetFit run
    - Users of the software must be able to create a user account that tracks the projects completed by the user. Additionally, the program must feature databasing that is capable of holding all of a user's projects.
5. Download any and all files from a BioNetFit run
    - All files generated by the web portal, whether that be generated configuration files, or files that are generated as a result of BioNetFit, must be able to be downloader to the user's computer.

The first three requirements can be seen as the main workflow of our project from the user's perspective, while the last two are extra features related to the database that will make the user's experience with our project more efficient and convenient.

ARCHITECHTURE

In this section, we will review the overall architecture of the software solution. This will detail how the web portal is to be use efficiently, including diagrams and walkthroughs that will detail the overall process of using the software. We will delve into possible issues that may arise when using the solution, and explore how these were handled within the architecture. In the next section, we will detail how the architecture was implemented, including the certain software and programming languages that were involved in the creation of the web portal.

Let us first observe our diagram that was seen in the introduction of the paper. This diagram outlines the intention of the software, and how the overall user experience of BioNetFit is improved as a result:



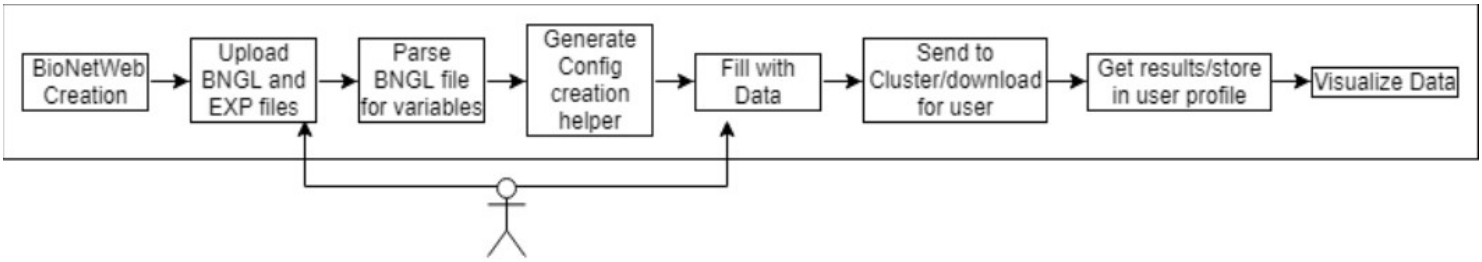
Workflow:

0. User uploads a bngl and exp file to the webserver.
1. The web portal then parses the provided files, extracting the given parameters for the experiment.
2. The web portal generates text fields to hold additional parameters that will be filled out by the user.
3. The web portal scrapes the text fields filled out by the user.
4. The web portal then generates a config file based on the parameters provided by the user.

Note: At this point in the architecture, the user is given the option to download the generated config file, rather than run an experiment. In the case that a download is requested, the file arrives in the users default downloads folder. From there, the user can halt the process, or continue to phase 5 at run the experiment.

5. The experiment is run on the NAU Monsoon cluster, or an equivalent solution. For the majority of the development process, experiments were run via a virtual environment that simulated a cluster computing system.

6. The output of the project is saved within the database to be accessed the by the user. The user can choose to download the created files to their computer, or keep the files attached to their account via the online database solution.



7. The user can then view visualizations for the results of the BioNetFit experiment. The visualizations help to condense the data into easy to decipher graphs. These graphs include visualization of average and best fit data for the permutations and generations that are created by the software.

At this point, the user has successfully created the necessary files, and has run BioNetFit using the web portal. Above, you can see a more condensed visualization of the overall architecture of the design. The simplicity of the web portal was intentioned to remove the user from as much of the BioNetFit process as possible by black boxing the process. The architecture successfully does this, as the user only has to provide the required BioNetFit files, and fill in the required parameters for the experiment. This is a vast improvement from original setup for BioNetFit, as we have bypassed the use of the command line and the made the process incredibly simple for the user to follow. We will now delve into the technology that was used to create each section of the software solution, and how it correlates to the architecture of the software.

IMPLEMENTATION

In this section, we will delve into the technologies that made each section of the software solution possible. We will provide a quick summary of each of the technologies, how they were implemented, and how they relate to architecture of the design of the web portal. We will begin by overviewing the web portal, and the technologies that went into its creation. We will then walk through the requirements of the software and delve into the software that made each of the steps of the architecture possible.

WEB PORTAL

Technologies:

- **Python Programming Language**

  Python serves as a basis for the entirety of the project, as many of the different facets of the project make use of specific python modules and packages.

- **Django**

  Django is a framework for python that allows for the easy development of and hosting of HTML files. We used this module to incorporate the easy design of HTML, with the complex modules that would connect to the computing cluster and visualizations of results.

- **HTML**

  HTML is one most basic programming languages that is used to generate websites. It is an easy to understand language, and is dynamically hosted via the Django framework.

Architectural Use:

The architecture finds its basis heavily in the Python programming language. From the beginning, our plan was to include python wherever possible, in order to ensure uniformity of the overall design. We will see this as we delve into the specific modules used.

Django serves as the overarching “glue” that keeps all of the individual parts of the solution together. Because Django is a module of python, it was easy to incorporate items such as database connectivity and connection to the cluster via pythonic SSH capabilities. Overall this was an excellent technological choice, as it allows the website to not only be updated quickly, but allows it to remain on the cutting edge of technology.

Much of the framework of the web portal relies on two things; 1. A simple and fluid design to be perceived by the user, and 2. easy dynamic alteration of the web pages. HTML offers a simple solution to both of these requirements.

REQUIREMENT 1: CONFIG FILE GENERATION

Technologies:

- **Javascript**

  Javascript is used to generate text fields and scrape information that is provided by the user. This information can then be turned into a certain file type, and can be either downloaded or saved to the online database.

Architectural Use:

Our use of Javascript within the project was not the original intention, however there is no module that python provides in order to quick scrape and generate files from text fields. Because of this, Javascript became an easy alternative, as it works well with HTML and generates files at a quick pace.

REQUIREMENT 2: CLUSTER COMPUTING

Technologies:

- **SSH via Paramiko**

  Paramiko allows for the simple inclusion of SSH commands into the overall structure of the website. This can be used to access the cluster computing center, and send experiments to the nodes available on the server.

- **Monsoon Virtual Enviroment**

  The Monsoon virtual environment provides a simulation of the real NAU computing cluster, however it only includes two nodes of processing power.

Architectural Use:

The use of the NAU Monsoon cluster never panned out as well as we had hoped. Due to potential security risks, the software was barred from making use of the full cluster offered by NAU. Thankfully, we were granted use of a virtual environment that mimics Monsoon, as detailed below.

Access to the NAU computing cluster Monsoon is very limited, however to provide a proof of concept, we were given access to a virtual environment that simulated a cluster computing environment.

REQUIREMENT 3: VISUALIZATIONS

Technologies:

- **D3**

  D3 is a module of Javasccipt that allows for easy construction of visualizations, mainly charts and graphs. The module shares many similarities to MatLab in its ability to module project data over time.

Architectural Use:

D3 proved to be an excellent solution to create visualizations. For each of the three types, we were able to easily parse and feed data from each of the files into the program to construct visualizations. Additionally, due to our previous use of Javascript in the program, the module integrates with the file creation mentioned earlier.

REQUIREMENT 4/5: SAVING & DOWNLOADING

Technologies:

- **MongoDB via PyMongo**

  PyMongo is a Python module that allows for communication between a python framework and MongoDB. Because MongoDB is a non-sql database, information can be stored freely, allowing for the storage of various text file types.

- **Django**

  The Django framework provides the ability to download files that have been generated by the both the Javascript config creation, as well as the experimentation results.

Architectural Use:

MongoDB was an excellent choice in order to hold the various file types that are generated by the system. Because we have so many different types that must be stored within the database, having a non-sql database removes any restrictions on the information stored. Each file is save in a set based on projects, and the projects are each attached to a given username.

Django proves to be an excellent framework for the project in its entirety. Using Django, we can implement buttons within the code to easily download the files generated by the website. This functionally can span throughout the entire website, allowing for the complete download of project files as well.

CONCLUSION

Compared to the original plan of the software, there were many changes that had to occur over time. At first, we wanted to implement Python in as many facets of the software as possible, in order to increase the ease of modification of the product. Over time however, we had realized that not everything we wanted to accomplish with the software would be feasible by only using Python. Therefore we changed our ideology, and implemented many different Javascript modules in order to accurately develop a working piece of software. Below you can view a table of our original ideas for the software, as well as the final choices that we eventually built in to the software.

Requirement	Fulfilled By	Discarded Alternatives
Dynamic HTML Generation	Django	PHP
Scraping Text Fields	Javascript	Django
Creating Files	Javascript	Django
SSH from server to server	Paramiko	-
Database	MongoDB / PyMongo	MySql
Visualizing Data	D3	MatPlotLib, Plotly

TESTING

Testing this software proved to be a difficult task. The software that we have developed is not a program that is meant for use by the general public. Because of this, it was hard to conduct solid user based testing, as many of the people who would be available to test the software would likely be unable to understand what the solution was producing. In order to accurately test the software in this format, we would have needed to address molecular biologists from the real world, in order to get their insight on the helpfulness of the software. We did however run multiple simulations with the client, who has a firm understanding of the topic. Each of these tests proved to be beneficial, and the resulting software was one that our client could make use of and share with his peers.

As for the testing of the actual software, we were successfully able to implement tests for each of the major portions of the solution. Below is a table outlining a general idea of each of the tests that we conducted on the software. Each of these tests will be explained in greater depth below the chart.

Subject	What Needs Testing	Testing Method
Web Portal	Needs to be simple and accessible	Tested 25 different files with varying parameters.
Database	Needs to store information and limit use	Stored 50 large projects within the database.
Visualizations	Needs to be reactive and robust	Generated multiple visualizations with correct and incorrect data
Web Services	Needs to have a fast response with the virtual cluster	Successfully downloaded all output files and project archives from the virtual cluster.

Web Portal
Testing of the web portal was done by attempting to create 25 different files with differing parameters, in order to assure that files were generated correctly.

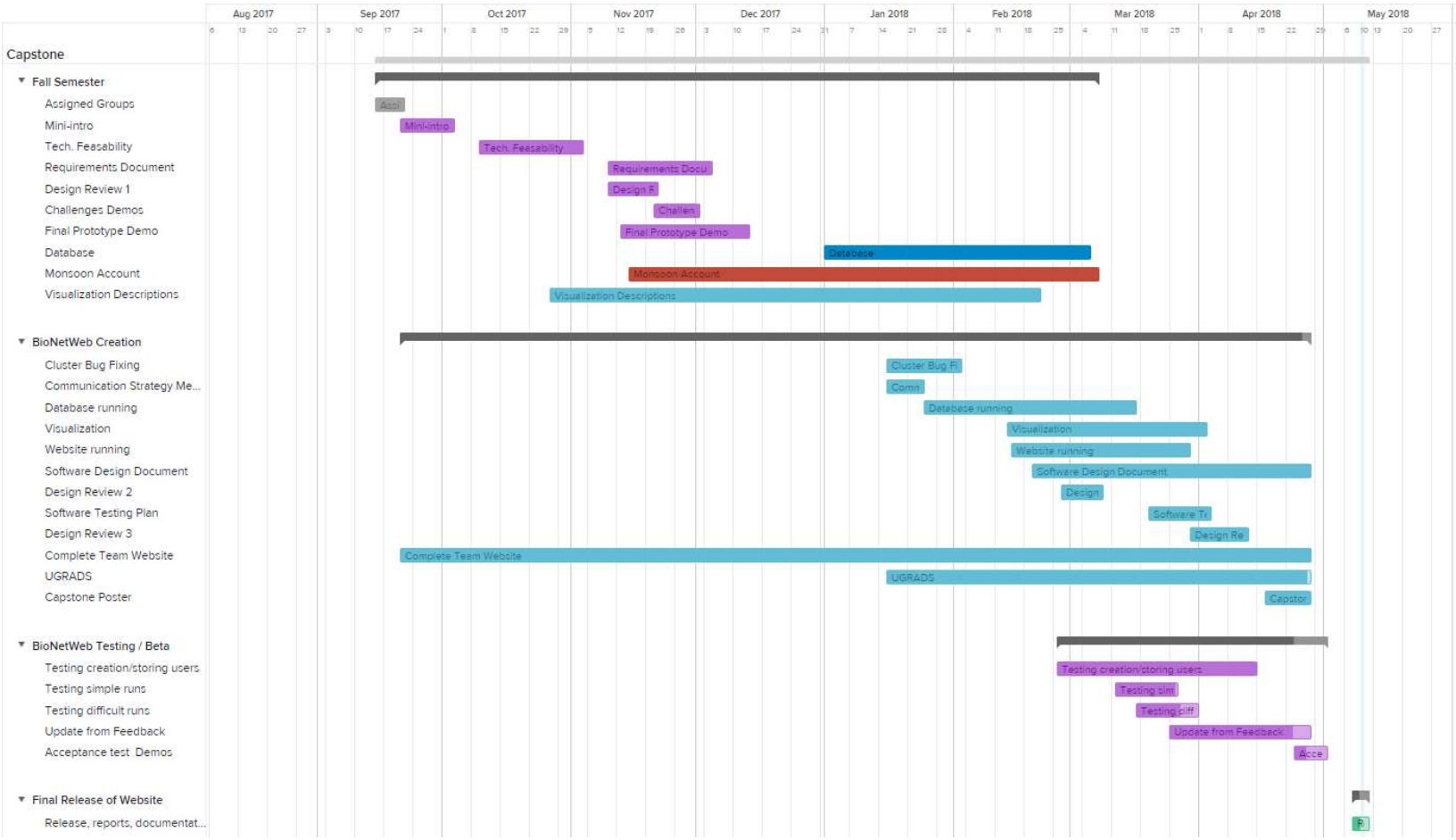
Database
In order to assure the database could handle storage of project files, we tested the storage of 50 differing projects at once. Each one was successfully stored and recovered from the database.

Visualizations
Visualizations were tested often throughout the development process, always generating the correct graphs for a given set of viable data.

Clustering
Clustering remained a challenge to test, as issues that were out of our hands caused the virtual environment to crash at random points. We were, however, able to establish a stable connection point at all times when the server was available.

PROJECT TIMELINE

Now that you understand the product that we have delivered, let us go over the timeline and see how we stuck to our original plan. Below you can see our Gantt chart that was consulted throughout the duration of the project:



We understand that this is a dense chart, so we will highlight some of the most important points. Overall, we were able to create a minimal viable product by the expected time within the chart. All testing and debugging was completed before the final week of school, and all documentation for the UGRADS symposium was done and presented on time. As of writing this, the team is currently finishing the final documents for the Capstone curriculum, as well as preparing for the final acceptance test for the software solution.

FUTURE WORK

Future improvements of the software solution will likely fall within 2 categories: security and visualizations. As mentioned earlier, we were unable to make use of the actual NAU Monsoon cluster due to security issues that may arise within the software. This was not a huge surprise, as there is a history of Capstone projects causing the Monsoon cluster to be hacked and used for malicious purposes. Given the needed time and hosting platforms, the security of the site can be massively improved, in order to assure that these types of exploits would not occur within the system. If that was accomplished, experiments could be run at a breakneck speed, allowing projects that may take hours to only take a matter of minutes. As for the visualizations, we believe that we have made a lot of progress in the vein of helping to visualize the output of BioNetFit. Additional work however would lead to more in depth visualizations of the data, possibly generating charts alongside the graphs that we have developed. Perhaps consulting professionals in the field will help to shed light on what other statistics would be helpful for users of the software.

CONCLUSION

Overall, Team U.I. Fit is very proud of the work that they were able to create for Dr. Razi and the creators of BioNetFit. We have successfully addressed each of the requirements given by the original requisition, and have expanded on features that have been requested later into the development cycle by our client. The resulting software will be able to help potential users of the BioNetFit software, who are wary of the complications that may come from attempting to use the command line tools, to use this groundbreaking software in order to test molecular combinations. This software allows a user to upload the files required to run BioNetFit in a simple and easy to understand manner. The system can then run experiments using a computing cluster or similar system in order to generate experiment results quickly and succinctly. Lastly, the solution allows for a user to view the visualization of results via graphs that help make sense of the original results of BioNetFit. Due to the changes we were able to make to BioNetFit, we can see this software garnering a wide use among those within the industry of molecular biology. Additionally, further improvements to the software may result in mass success of the BioNetFit solution. This would be incredibly impactful for both Team U.I. Fit, Dr. Razi, and the NAU graduates who assisted in both the creation and advancement of the software.

APPENDIX A: DEVELOPMENT ENVIROMENT

Hardware

This software solution was developed using a wide array of machines with different specs. These ranged from an Omen high performance gaming laptop, to a simple Chromebook that was dual booted with Gallium OS, a Linux distribution. Needless to say, we **do not believe** that there is a minimum level of computing power required to run the software solution. The software solution was also developed using both **Windows** and **Linux** operating systems. During the implementation process, we did not encounter any issues that would cause friction between these two operating systems. We did not at any time use a Mac operating system in order to run or edit the code base, however we do not envision any issues with using this operating system in order to edit the code. We will note however that this appendix features setup information for only Windows and Linux distributions.

Toolchain

In order to address the tools that factored into generating the code for the solution, we have divided the technology used into three categories: IDEs, required technologies, and cluster technologies.

- **IDEs** – In order to develop the code for the project, the members of the team used a wide variety of different IDEs. These included the IDLE Python IDE, Text Editor, Notepad++, Atom, PyCharm, and Visual Code Studio. We would recommend that the user install their IDE of choice, as there did not occur a situation where programming on different IDEs caused differences between code within the repository.

- **Required** – In order to fully test the code, we recommend that the user download the Python programming language, as well as the PyMongo, Paramiko, and Django python packages. Additionally, we recommend having MongoDB installed on the testing device.

- **Clustering** – We are aware that not everyone who may edit or test this code in the future will have access to a clustering component. If you are conducting work at NAU, please contact the current heads of the Monsoon computing cluster for information on gaining access to a virtual environment.

Setup

In order to set up an environment for development/testing on your own device, please consult the following instructions:

Code Retrieval

1. If you currently have Git installed on your device, please skip to step 3. If not, proceed to step 2.
2. Install Git on your device by visiting https://git-scm.com/ and following the installation process.
3. Set up a specific folder within your device to hold the code you will be downloading. Name this folder accordingly so you will not forget which folder it is.
4. Using Git, clone the BioNetWeb repository into your created directory from https://github.com/JoshGutman/BioNetWeb.git. For assistance with using Git, please visit https://git-scm.com/docs.

Python Setup

1. If you are using a Windows distribution, you can install Python using the manager provided at https://www.python.org/downloads/. We recommend using Python 3 for the most up to date software.
2. If you are using a Linux distribution, you can install Python using the terminal command “sudo apt-get install python3”.
3. To run the software, you must have the following Python packages: Django, Paramiko, and PyMongo. In order to install these packages, please consult Python documentation, this step will differ depending on the device being used, and not just the operating system.

MongoDB

1. To install MongoDB on your local machine, we highly recommend consulting this informative video guide: https://www.youtube.com/watch?v=pWbMrx5rVBE
2. More information on MongoDB can be found at https://www.mongodb.com/.

Production

Here we will outline the required steps for running the software, as well as making edits to the software in the future.

Running the Program

1. Run MongoDB, using the steps provided in the video referenced in the MongoDB section of setup.

2. Using the command line, navigate to the storage location for the BioNetWeb code. From here, navigate into the BioNetWeb folder. You should see a file that reads manage.py.

3. Depending on the operating system, run the command python manage.py runserver. For information on how to run this command on your OS, please visit the python documentation.

4. Using a web browser of choice, visit localhost:8080 in order to view the website.

Editing the Program

Note: The manage.py runserver does not have to be turned off in order to make edits to the code. Any edits will be dynamically changed live as they are saved to the device.

1. If using an IDE, open the folder containing the entire code repository within the IDE for a full view of the project.

2. If using a text editor, navigate to the given file you want to edit within the repository.

3. After making changes to a file, be sure to save the changes in order to assure they will appear on the local version of the server.

4. If you believe you have made a change that constitutes a new version of the software be created, please message the owner of the GitHub repository for questions about making major changes to the online code base.

5. We strongly recommend that the user studies the Django framework before editing, as there are many intricacies within the module that are too complex to be listed here. In short however, we advise that any changes made to the software be tracked, in order to allow for bug fixing if needed.