12/4/2017

Requirements Documentation

Team U.I. Fit – Version 1.1

Team Members:
Charles Chatwin
Matthew Burns
Tanner Brelje
Joshua Gutman

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

Accepted as baseline requirements for the project:

For the client: _____ Date _____

For the team: _____ Date _____

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. The program has unequivocally proved its usefulness to the molecular biology community, as the program has begun to see use within the professional setting of places such as Los Alamos National Laboratory in New Mexico. However, as with the commercialization of many other kinds of software, BioNetFit, while proving useful, is also proving to be a challenge for many researchers to fully utilize. In its current state, BioNetFit exists as a unix-only, command line tool. While this may be acceptable for users within the fields of engineering and computer science, this may cause unneeded hurdles for scientists with little knowledge of higher intensity computer workings. As a result, 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. Other issues have also become apparent in the software, including the lack of visualizations for the outputs of the software, as well as the extensive amount of time needed to process large scale experiments with the program. Unfortunately, many researchers cannot fully harness the power of BioNetFit due to these issues, even though the code of the program can work wonders for the industry.

# time	Ltot	freeL	Rtot	Rdim	RLbonds	pR
0.000000000000e+00	1.264649400000e+05	1.264649400000e+05	9.999999999811e+03	1.236278721052e+03	0.000000000000e+00	4.249866570215e+03
5.000000000000e+00	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132599569e+03	9.999907706411e+03	4.340349248364e+03
1.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595529e+03	9.999907706411e+03	4.428741136799e+03
1.500000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595521e+03	9.999907706411e+03	4.512769052866e+03
2.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.592648507003e+03
2.500000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.668584277603e+03
3.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.740771064712e+03
3.500000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.809393955909e+03
4.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.874628912888e+03
4.500000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.936643187073e+03
5.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	4.995595771003e+03
5.500000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	5.051637828924e+03
6.000000000000e+01	1.264649400014e+05	1.164650322950e+05	1.000000000121e+04	2.572132595515e+03	9.999907706411e+03	5.104913066161e+03

Figure 1: *Example output of BioNetFit in its current state.*

In order to create a permanent fix to these issues, Dr. Abolfazl Razi, in partnership with Dr. Richard Posner and Dr. William Hilavaecek, has contracted Team U.I. Fit, composed of Charles Chatwin, Matthew Burns, Tanner Brelje and Josh Gutman, to create software solutions in order to increase the overall usability of BioNetFit. In order to accomplish this task, team U.I. Fit has conducted bi-weekly meetings with Dr. Razi in order to pinpoint the exact issues that are plaguing the system. These issues, known as requirements, will lay the groundwork for the project going forward, as well as create goals that the software will aim to achieve by the end of the implementation phase. After analyzing the issues of BioNetFit, Team U.I. Fit in correspondence with Dr. Razi have identified the three main requirements for a feasible software solution. These requirements may contain lesser, sub-requirements within them, but each of these categories represents a vital section of the software solution that will vastly increase BioNetFit's usability amongst researchers.

The Main Requirements of a BioNetFit software solution are as follows:

- Create a GUI that increase heavily the usability of BioNetFit.
- Visualize results and outputs of BioNetFit to increase ease-of-interpretation.
- Increase the speed of processing for large scale experiments via implementation on a computing cluster.

Throughout the remainder of this paper, each of these requirements, as well as the sub-requirements that factor in to each major requirement. We will first discuss the problem and solution to BioNetFit's issues in greater detail. From there, we will address both the functional and environmental requirements of the software solution. To close, we will discuss the risks and challenges associated with the project, the development plan for the foreseeable future, and closing comments on the software solution. Overall, if planned, designed, and implemented correctly, this software solution will drastically change the use of BioNetFit, and could allow for global use of software in both academia as well as professional settings. We at Team U.I. Fit have become impassioned in our pursuit to create an amazing software that can be used by researchers around the globe. We firmly believe that our solution will not only transform BioNetFit into the successful software it can be, but will continue to elevate BioNetFit long after our involvement with the program has concluded.

PROBLEM STATEMENT

The field of bio-molecular engineering is one full of nuance and experimentation. In order to process molecular combinations, researchers must spend hours or even days manually creating and running thousands of different experiments. These experiments are intensive both in labor and in cost, however it is a mandatory process that helps the field advance. A frustrating aspect of these experiments however, is the possibility of miscalculations and human errors. These can ruin an experiment and render the process useless, wasting all of the valuable resources mentioned previously. In order to counter this, programs have been created in order to simulate these larger experiments. BioNetFit and NFSim are some such programs, and are the basis for the project that will be created as a result of this documentation.

BioNetFit allows a researcher to calculate the expected results molecular experiments without having to expend any resources and very little time. These processes create experimental results files that can be interpreted and analyzed in order to aid in the creation of a real time experiment. After the program generates this file, as well as a BioNetGen Language (BNGL) file, the system will then produce a config file which is used by BioNetFit needs to run the simulation. After creating the config file, the researcher will access the command line from a linux machine, an operating system (OS) only 2.98% of desktop users have experience with, and will run BioNetFit from here. Then the system will slowly go through the thousands of iterations of molecular bindings in order to generate a text column result for the user to interpret.

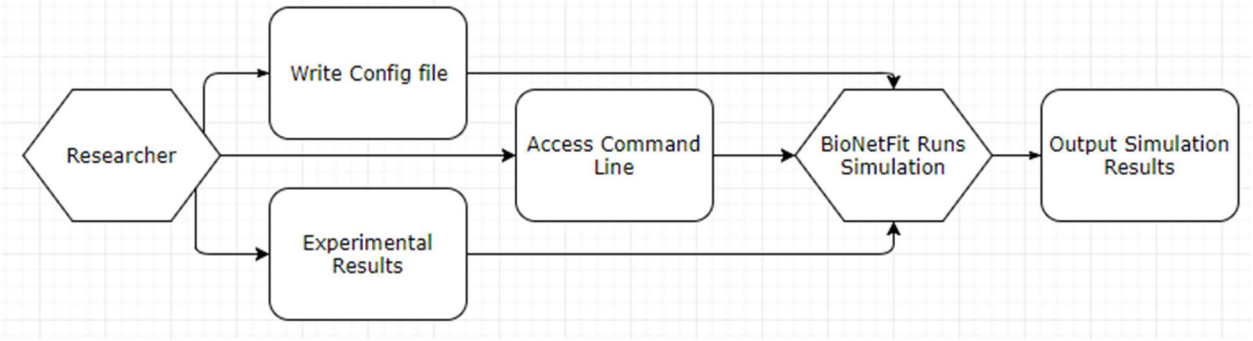


Figure 2: *BioNetFit process flowchart*

Observing the defined workflow, the problems with BioNetFit in its current iteration can be identified as one of either usability or speed.

- The current OS, Linux, forces the user to adapt to a new work environment and transfer all data to the new operating system.
- The generation of a config file is complex, as some values affect what the program requires in order to run correctly.

- Without a dedicated IDE for the file writing, accuracy in dependent values as well as general syntax mistakes take time to find and fix.
- Interacting with the command line, while not too complicated creates a mistake point that increases the likelihood that the user will have to restart the process or damage part of their system.
- Even with high amounts of allocated resources, the process will still take hours, days, or weeks in order to run large batch simulations.
- The programs output exists of columns of numbers defined in the config file, with no easy way to interpret the data.

SOLUTION VISION

In order to address the issues presented by BioNetFit, Team U.I. Fit plans to implement an online **Graphical User interface (GUI)**, built using Web 2.0 design fundamentals featuring output **visualization** and **parallelization** on the NAU Monsoon cluster. This will allow a researcher on any desktop and any OS to run BioNetFit simulations from anywhere in the world, receive clear and concise visualized data, and run large scale experiments in an adequate timeframe.

In order to flesh out the needs of researchers and for our system, our website will have the following features:

- The GUI will take in the BNGL and experimental result input files, assist the user in entering their parameter values, and then generate a config file.
- Run BioNetFit on a cluster using the inputted and generated files.
- Take the BioNetFit output and visualize it in some manner so it can be more easily understood.
- Allow a researcher to save their previous files in a "logbook" held on our database, and re-run these previous simulations with modified/changed data.
- In order to solve the long wait times for results, our website will be accessing a version of BioNetFit that will be put onto the NAU monsoon computing cluster in order to provide results in a timely manner.

These systems tackle all of the major problems pointed out in the previous section. The solution improves both usability and speed, as well as getting rid of crucial areas more prone to user error. We considered simply putting the program onto an optimized computer or server, however that merely solves the universal OS problem without touching the issue of speed. We settled on the solution outlined above as it was the one that most increased positive change in the way researchers use BioNetFit, as well as having little to no cons in terms of performance issues. Since the only data that we will be dealing with is user login information, user log books, config files, results files, privately held Monsoon cluster credentials and admin privileges, we have a fairly simple structure allowing for a streamlined and efficient system architecture. In fast moving fields such as medicine and chemistry, having the ability to visualize and run your experiments before spending time and money on chemicals and lab time is invaluable. Additionally, increasing the ease of use for researchers will increase the viability of the program as a research tool, and will promote the advancement of the field.

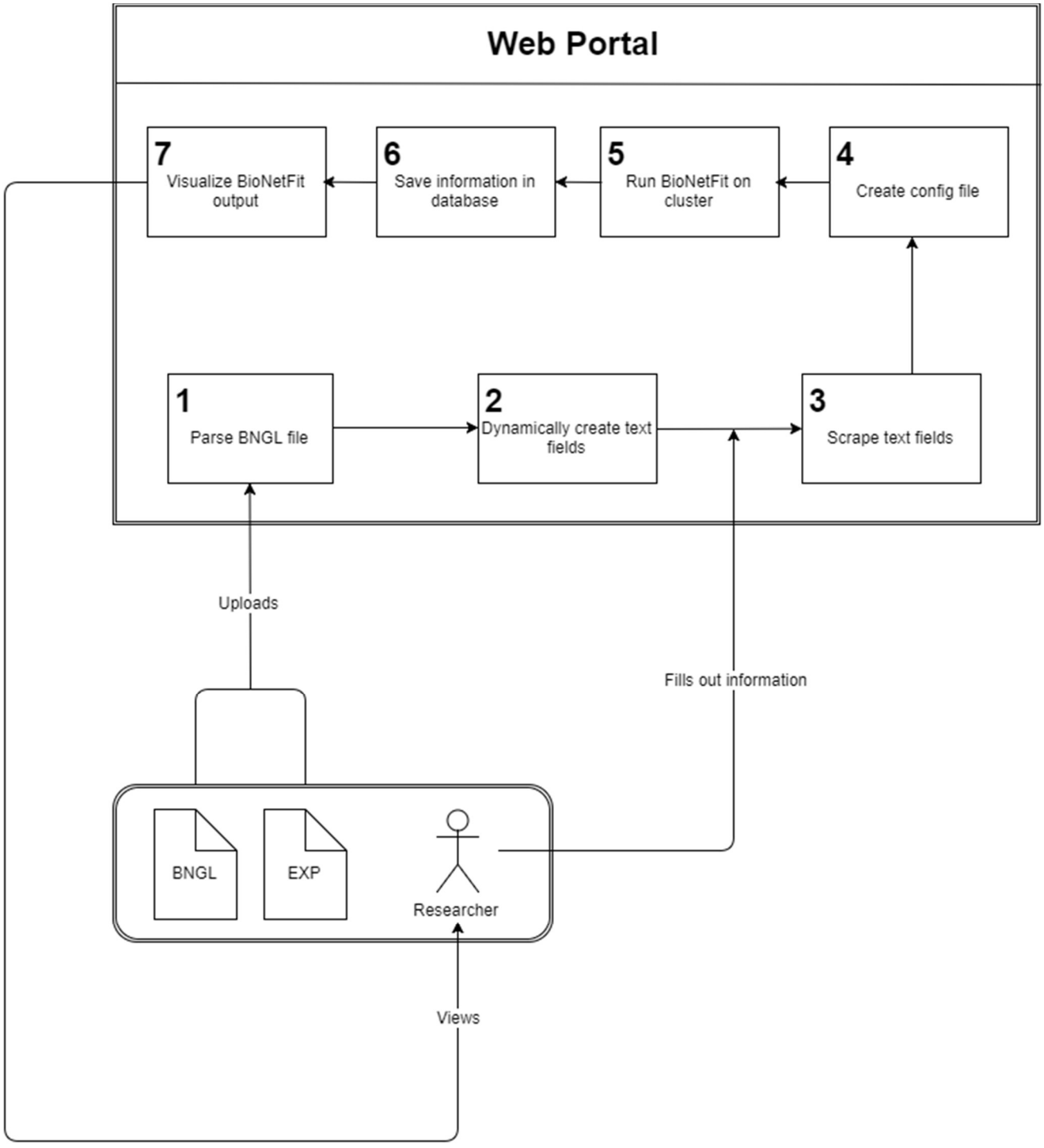


Figure 3: *Predicted workflow of software solution for BioNetFit*

PROJECT 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.

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
2. Run BioNetFit on a cluster or server
3. Visualize the output of BioNetFit
4. Save all information related to BioNetFit run
5. Repeat past experiments with identical or altered inputs
6. Download any and all files from a BioNetFit run

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

FUNCTIONAL REQUIREMENTS

Each of the domain-level requirements can be split into several functional requirements, which specify more detailed implementation-related requirements. The domain-level requirements are numbered, and each of the stemming functional requirements is bulleted below.

1. Generate .config files from BNGL files
    - Uploading files
    - Parsing uploaded files
    - Dynamic HTML generation
    - Scraping text fields
    - File creation

2. Run BioNetFit on a cluster or server
    - Establish SSH connections
    - Send files over SSH connection
    - Run commands remotely using SSH connection

3. Visualize the output of BioNetFit
    - Read output file(s)
    - Plot one or many variables in output file(s)
    - Display the plot to the user

4. Save all information related to BioNetFit run
    - Create a database
    - Create user accounts in the database
    - Store files in a database

5. Repeat past experiments with identical or altered inputs
    - Retrieve files from database
    - Remotely editing files

6. Download any and all files from a BioNetFit run
    - Retrieve files from a database
    - Download files from a database

CREATING THE CONFIGURATION FILE

UPLOADING FILES

The user must be able to upload both a BNGL file and an experimental results file in order to ultimately create a configuration file. While uploading a file can be done in Javascript, we are using the web-development platform *Django*, a third-party Python module, for many aspects of our project. Django can handle, among other things, receiving uploaded files from the user, which will eliminate the need to convert and prepare the file data into a readable form for Django in the future. Therefore, uploading files will be handled by Django.

PARSING UPLOADED FILES

Once the BNGL and experimental results files are uploaded, the BNGL file must be parsed to determine exactly which parameters will be variable in the BioNetFit run. This step will determine what information needs to go into the configuration file. Because Django will read in the file, and because it is the simplest and most straightforward method, we will be using vanilla Python to parse the BNGL file. As mentioned above, this will eliminate the need to translate the data into any other forms, as everything up to this point is being done in Python. Using Python, we can also make the code that parses the BNGL file highly modular and simple, making it easy to account for any new terminology or features added to BNGL in the future.

DYNAMIC HTML GENERATION

After the user has uploaded the BNGL and experimental results file, and the BNGL file has been parsed for all relevant information, the web-portal must dynamically create a new webpage where the user can enter values or ranges of values for each variable parameter. Because Django can handle the creation of text fields, and because the variable parameters were already determined using Python, Django will be used to dynamically generate HTML.

SCRAPING TEXT FIELDS

After the text fields have been generated, and the user has entered all relevant information, the text fields must be read in order to get the information. Django can also handle this aspect of the project, and will be used to scrape the text fields.

FILE CREATION

The final step in the creation of the configuration file is, unsurprisingly, generating the configuration file itself. At this point in the workflow, the program will already have all the required information, so the only thing left to do is to output and save it to a file. This will be done in vanilla Python, in order to preserve the unified development environment, and because it is the simplest and most straightforward way to accomplish this.

RUNNING BIONETFIT ON A CLUSTER OR SERVER

ESTABLISHING SSH CONNECTIONS

One of our long-term goals is to get a special account on NAU's High Performance Computing Cluster *Monsoon* that will be used exclusively to run BioNetFit. Monsoon accounts are accesses remotely via SSH connections. In order to keep our project as simple as possible, and to maintain a unified development environment, we are going to using the third-party Python module *Paramiko*. Paramiko has already been tested, and an SSH connection was successfully established between a team member's laptop and their personal Monsoon account.

SENDING FILES OVER SSH

Before BioNetFit is run on Monsoon, the experimental results file, BNGL file, and configuration file must be transferred to Monsoon. Although Paramiko's main function is to establish SSH connections, it also has a function that fully encapsulates sending files. Sending a file is as easy as passing in its local address and the destination to Paramiko. Therefore, Paramiko will also be used to send all the necessary files to Monsoon.

RUNNING COMMANDS REMOTELY

After an SSH connection has been established, and the input files for BioNetFit are transferred to Monsoon, BioNetFit itself must be run. Using the SSH connection established by Paramiko, commands can be run on Monsoon by simply passing the desired command as a string to a function in Paramiko. Because the locations of the input files will vary, the command string to run BioNetFit will have to be dynamically generated. This can be accomplished in vanilla Python, as long as the locations of the input files are kept track of. In conclusion, Paramiko will be used to establish SSH connections, transfer files, and run BioNetFit on Monsoon. Vanilla Python will be used to keep track of file locations and dynamically generate the command to run BioNetFit.

VISUALIZING THE BIONETFIT OUTPUT

READING BIONETFIT OUTPUT FILES

BioNetFit output files are called gdat files, and consist of a 2D matrix, where each column is the value of a variable, and each row is a time that that value occurred at. Reading the output files will be done with vanilla Python. Each variable will be stored in its own list, with each element in the list being a value of that variable at time X. Reading the files with Python will make the visualization process simpler, more modular, and more unified. Any changes to the format of gdat files in future versions of BioNetFit could be accounted for by making changes to the function that reads in and interprets them. Additionally, different methods of visualization can be added by adding different functions that read and interpret different data. For example, the output from each generation of the BioNetFit run (rather than the final output) can be read in, and then plotted against the experimental results file to allow the user to determine if BioNetFit is truly choosing the best parameter combinations.

PLOTTING ONE OR MANY VARIABLES

Plotting and visualizations will be done with the third-party Python module *MatPlotLib*. It's simple to use and maintain, and keeps a unified development environment. Creating a plot can be done by simply passing in each list of variables and a list of the times at which those variables occur to a function in MatPlotLib. This will create a graph with all variables plotted against the time. However, the user may want to toggle certain variables on or off. Because of this, instead of creating a single graph with all variables, we will create a separate graph with for each variable, and save it as a transparent PNG. Then, using Django, we will create a checkbox for each variable to allow the user to show or hide them. Each variable that is shown will have its graph overlaid on top of all other graphs (which is made possible by saving the graphs as transparent PNGs). Any hidden variables will have their graphs deleted. MatPlotLib has functions to save graphs as transparent PNGs, and Django has functions to create check boxes, and show/delete images.

DISPLAYING THE PLOT TO THE USER

The visualization will be displayed on our website when the user selects the experiment. Because BioNetFit can take days or sometimes weeks to complete, the visualizations will not be available right away. However, when the simulation is complete, the user will be able to select the experiment and the visualizations will be displayed. This feature relies on our implementation of a database, which will be discussed later in this document.

SAVING INFORMATION RELATED TO BIONETFIT

CREATING A DATABASE

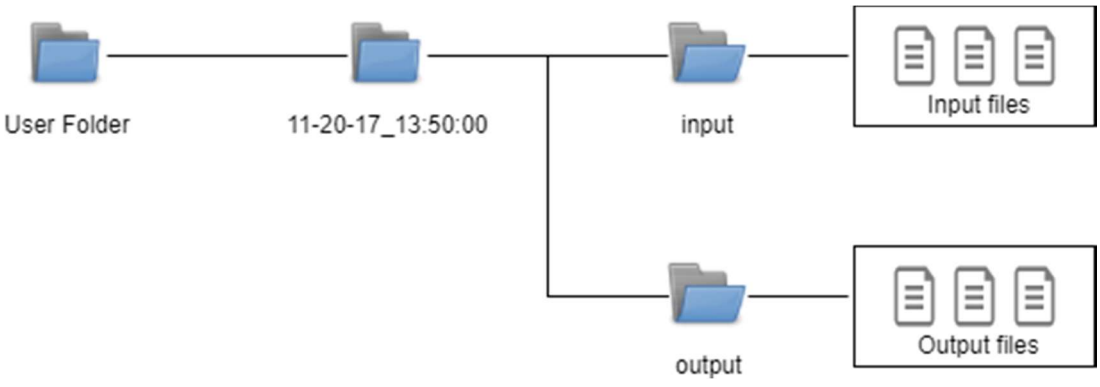
In order to save information in an organized and maintainable way, a database must be used. We have settled on MongoDB for many reasons. It's open-source, free to use, scalable, maintainable, and can be interfaced with through Django. The fact that Django can interact with MongoDB again keeps a unified development environment, and eliminated the need to convert data to a different format.

CREATING USER ACCOUNTS

If a user wants to save their results, they must make an account. The account creation page can be read and sent to MongoDB through Django. It will consist of an email address field, a password field, and a password confirmation field. To minimize the damage in the event of a data leak, No other personal information can be entered. After the user creates their account, they can login to save the input and output files of a BioNetFit run, view those files (and visualizations of those files if BioNetFit has finished running), and edit input files to rerun BioNetFit.

STORING FILES IN THE DATABASE

MongoDB has a built-in distributed file system called GridFS. When a user uploads files, a new directory will be created within the user's account directory. The directory will be have a default name of the time and date of the upload. Within that directory, there will be two subdirectories: *input* and *output*. The input directory will store all input files. The output directory will store all output files. Below is a diagram of the directory structure:



REPEATING PAST EXPERIMENTS

RETRIEVING FILES FROM THE DATABASE

Sometimes, after BioNetFit has finished running, the user will realize that they used incorrect or suboptimal parameter ranges, and that they have to run it again. In this event, rather than reupload and recreate all the BioNetFit input files, it would be much more convenient to just edit the existing files. In order to do this, we will have to retrieve the files from MongoDB. However, if the user needs to make more than small changes, or already has an edited file on their local machine, it would be easier to just upload a new file. In this case, the file will still be retrieved from the database, but there will be an option for the user to upload a file instead of edit the old one.

REMOTELY EDITING FILES

Rather than delivering the files directly to the user, we will provide the contents of the file in a editable text field via Django, allowing the user to edit the document directly in their browser. When they make and confirm the changes, a new project directory will be made in the database, and the other input files from the edited file's directory will be copied to the new directory. If the file is a BNGL file, the program will parse the new file and dynamically generate text fields in preparation for the creation of a new configuration file, just as if the user had uploaded a BNGL file. If the file is an experimental results file, the user can either edit the file or choose to upload a new one.

DOWNLOADING FILES FROM BIONETFIT

RETRIEVING FILES FROM THE DATABASE

Similar to when a user is repeating past experiments, files need to be retrieved from the database in order for the user to download them. Rather than making a whole new interface and process for downloading the files, it would be easier and more convenient for the user if there was simply a "Download" button in the editing interface.

DOWNLOADING THE FILES

As stated previously, Django can be used to retrieve the files from MongoDB and then serve them to the user.

ENVIROMENTAL REQUIREMENTS

BioNetFit as a program exists as a command line accessible Linux application. This means that everything we do must take as much advantage of this setup as possible, as well as facilitating access with the other external systems. Aside from this, the client did not enforce any other necessary specifications, so the rest was decided by what integrated the best with the cluster, the GUI and our database.

NAU Monsoon requires its users to be on NAU internet services to gain access to it. In order for the site to be fully realized it would either have to be hosted on a remote web service that would have a VPN system set up to allow it to log in, or it would need to be held on an NAU web hosting service. The simpler and smoother solution is to have it hosted on NAUs' website, so we will either obtain a domain specific for the product, or it will be hosted on our client's web pages.

As we are developing a GUI, we are using a combination of HTML and CSS to create the layout for the site itself. Additionally, Python was chosen as our method of generating dynamic content on our website to enter the config file parameters in. This was shown as easier than doing it in PHP and Javascript, and lends itself to other uses in our system. Putting BioNetFit on the cluster allows the GUI to take full advantage of any web development options possible as long as they could communicate with the cluster. As outlined above, we settled on using the python module Paramiko to communicate with the cluster, which natively sends and receives files from the system for the user.

In order to store user login and simulation data, as well as our python generation scripts, a database is required. MongoDB, an open source database, provides easy integration via Python. This aides our system tremendously, as much of our software is using or integrates well with Python. Our system will likely rely majorly on Python, creating a dependency that needs to be worked around.

POTENTIAL RISKS

In using this collection of technologies to create a more cohesive portal for researchers to run BioNetFit, we create a number of risks in our system that could harm the security of our user and system data.

Our project has two main high priority risks. The first being the instances where the system would be unable to access Monsoon, and the second is insecure storage and usage of Monsoon login details. The project's reliance on the NAU Monsoon computing cluster to run BioNetFit runs the risk of putting users out of a fast way to run simulations if something goes wrong. With the use of it being so integral to our project's design, it is our top priority to ensure the stability of this connection. While we are obtaining permission for our system to be able to remotely access the account, there is a risk of updates to monsoon requiring the system to be redone. Additionally, if the cluster undergoes maintenance and connection is lost, the user cannot run their program. To mitigate this, we are allowing the user to download the config files to their own system. This way, users are still able to run the simulation on their own system, albeit slower. After maintaining connection, database storage of the Monsoon Login details, as well as secure access to the Monsoon system will become a top priority. Since the cluster is not under our control specifically, we need to make sure that we aren't creating security concerns for the cluster, especially those that would get our access to it revoked.

Of lesser concern, we need to both secure a domain to host the BioNetFit GUI as well as preventing the database from leaking personal user data. Our current plan is to have our website hosted using NAUs web services. Amazon's web hosting service exists as a backup option, as well as providing a testing platform for our prototypes. Users' personal information is limited to login name, password and any simulation files that they create. This minimizes the risk of personal information being leaked, and provides the user with security about their identity on our service.

Thankfully, risks to user data is kept to a minimum with the current architecture. Once an accurate config generation is complete, the user gets access to that file and can check it manually for errors in parameters. After that, the file is sent to BioNetFit and the results are generated and sent back. The main risk here is that the system loses the user's "location" and is unable to send the results to them. Thankfully with the systems python architecture this is a minimized risk, but if it fails, the user loses out on valuable time. However, since the files will be automatically saved to their profile, they are able to access and run it again easily. This is a fairly easily minimized risk that will allow our GUI to stay ahead of competitors. Since ones that exist like NFSim are hardware specific applications, they are limited ours is a flexible and universal system that already has a leg up on other resources for researchers.

PROJECT PLAN

Over the course of August through November, the project has taken extensive steps in production and is rapidly approaching the implementation phase. At the time of writing, late November, 2017, the the specifics for the project have not only been ironed out with the client, but have also been researched extensively by Team U.I. Fit. As discussed earlier in the document, solutions for each of the client's problem have been theorized and even tested. This bodes well for the project in the remaining month, as the prototyping phase will be easily transitioned to because of these breakthroughs.

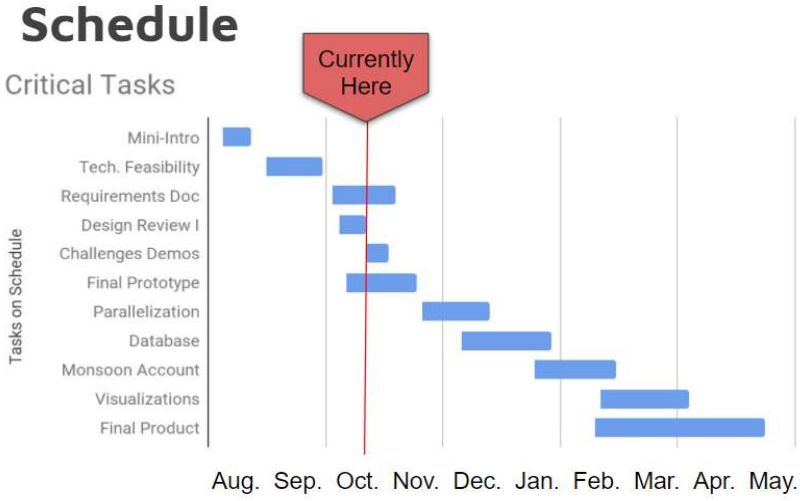
Currently, team U.I. Fit has achieved the following milestones regarding the project:

- Team Standards and Team Inventory Documentation
- Introductory Presentation on Solutions to the client's problems
- Technological Feasibility Documentation

Each milestone was completed within the allotted timeframe dictated by the capstone course, and the group has not fallen behind on any of the current tasks administered by the course instructor. Apart from the current document, the team must complete the following before the current semester ends in mid-December:

- Technical Challenges Demonstration
- Prototyping

Looking forward, these tasks will be easily manageable by the team, and will lead to a smooth transition into the implementation phase of the project when the next semester begins. For a comprehensive look into the current status of the project, as well as the past and future milestones for the project, please observe the Gantt chart below for a comprehensive timeline of events.



CONCLUSION

In conclusion, we have addressed the multiple issues that are currently plaguing BioNetFit, as well as the solutions that we have planned in order to address them. From there, we have delved into the structural requirements for the software solution, both functional and environmental. Structurally we have identified six main requirements that we must implement:

- Config file creation
- Cluster computing
- Visualization
- File storage and saving
- Experiment replication
- Downloading.

Additionally, we have discussed the issues that may arise when attempting to implement the software solution on the given environment. To conclude, we discussed the future plans of the project, as well as the milestones that have currently been surpassed.

Overall, the project appears to be feasible and the requirements, while possibly difficult to implement at times, should be achieved within the allotted time given by the capstone curriculum. Over the next six months, Team U.I Fit will begin work applying the information given within this document into a physical software solution. This will involve many tests, trials, and different implementation strategies in order to achieve the goals laid forth by this paper. The end result will give BioNetFit the optimization it truly deserves, and will help the software take steps to a global use in labs throughout the world.