REQUIREMENTS SPECIFICATION

Nov. 19, 2021

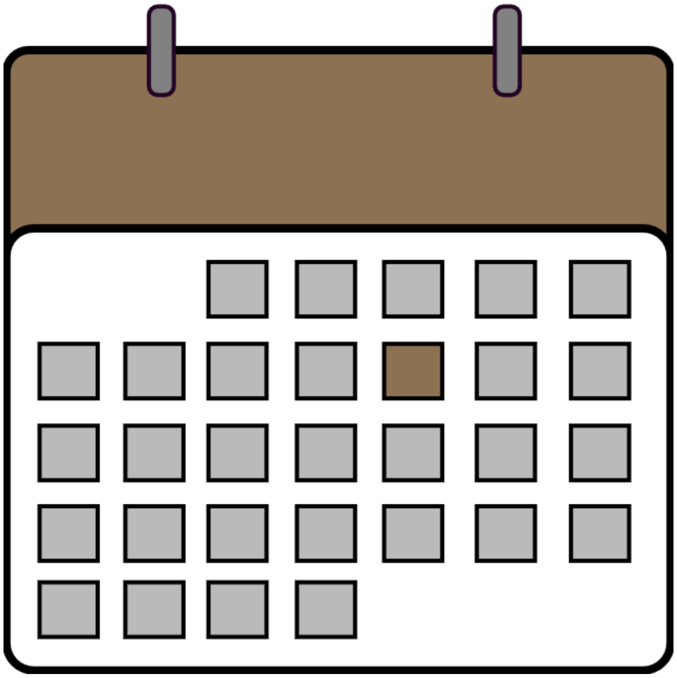
VERSION 1.2

SPONSOR: DR. VIACHESLAV FOFANOV

MENTOR: VOVA SARUTA

JOSEPH DOMABYL V ANDREW LIDDELL JUNJIAN YIN DANIEL DRAKE

Magisters



Accepted as baseline requirements for the project:

For the client: ______________________________________ Date: __________________

For the team: ______________________________________ Date: __________________

TABLE OF CONTENTS

1 - INTRODUCTION

Working with and coordinating large groups of people can be a time consuming task. Consider a scenario where you've been set to schedule potentially hundreds of different people, all with different strengths, weaknesses, and availability. You've been working on this schedule for a couple of weeks only to find that one of your team members can't make the 2pm timeslot on Friday they said they could. You now have to find them a new spot in the schedule they can attend, but you will have to factor in everyone else’s schedule before changing it. Moving one person might require that you move three more people down the line. This has the potential to become destructive for your workflow and is what university departments all around the nation deal with on a regular basis. This is a persistent issue that plagues these departments semester after semester, particularly the scheduling of teaching assistants (TA’s) and graduate teaching assistants (GTA’s) for lab positions. For the rest of this document we will be referring to both TA’s and GTA’s simply as TA’s unless specified otherwise.

The current workflow used by our client, Dr. Viacheslav Fofanov, is to use an Excel spreadsheet and manually assign these TA’s to where they need to be. Dr. Fofanov is the associate director for the School of Informatics, Computing, and Cyber Systems (SICCS) at Northern Arizona University (NAU). Specifically, Dr. Fofanov coordinates GTA’s to teach labs and TA’s to assist their graduate counterparts. Dr. Fofanov currently spends an estimated 20-30 hours creating the schedule but emphasized that this number is largely dependent on a multitude of other variables. This might include elements such as how long it takes for students to respond with their personal schedules, how long it takes to create the table depending on volume, or how long it takes to cross-reference time slots in order to ensure an accurate schedule. Perhaps one of the most important aspects of this problem is that it is persistent and has the potential to be troublesome *throughout* each semester, not just at the beginning. That is, a TA might unexpectedly leave their position and their spot would need to be filled. However, the problem goes deeper than the brief explanation above, which we will describe in further detail below.

2 - PROBLEM STATEMENT

The current process for assigning TA's is as follows; our client will first gather the information of the newly hired TA's and assess their abilities and qualifications. Depending on their experience, they are assigned to a specific lab with their scheduling constraints in mind. This means Dr. Fofanov will design the new TA schedule with all of their constraints in mind. To officially record this, the schedule is manually built using an Excel spreadsheet. Each of the TA's are manually entered and assigned to their labs to lecture and/or grade for. The figure below describes the current workflow of our client.

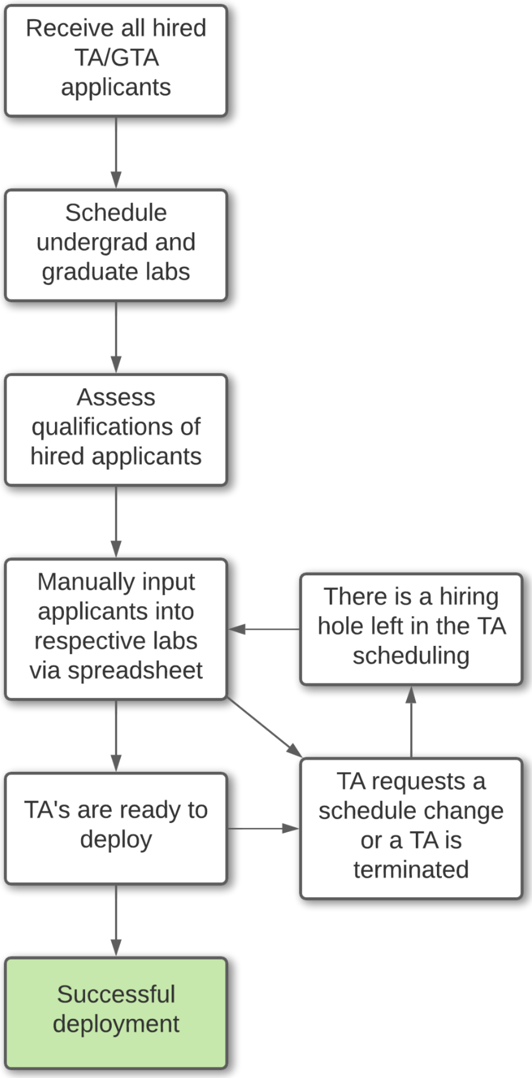

Receive all hired TA/GTA applicants
Schedule undergrad and graduate labs
Assess qualifications of hired applicants
Manually input applicants into respective labs via spreadsheet
There is a hiring hole left in the TA scheduling
TA's are ready to deploy
TA requests a schedule change or a TA is terminated
Successful deployment

Figure 1. Client Workflow

Besides the time consumption, the issues with this current process truly expose themselves once there is a change in the schedule. If a TA is terminated, has a schedule discrepancy, or wishes to switch classes, it has the potential to create a series of problems:

- Butterfly effect of several people potentially having schedule conflicts from a single change
- Constant reevaluation of TA's qualifications and schedules
- Some TA's may be forced out of a class against their will
- Time consuming manually reentering individuals into the spreadsheet
- Peer pressure between TA's for scheduling advantages

To solve the time consumption of the butterfly effect, reevaluation, and the manual input of TA's, we will design an optimization algorithm that will automatically sort the TA's depending on their qualifications and availability constraints provided. We will create an interface where the lab organizer can easily see all possible changes and implement the best schedule for all of the TA's at the click of a few buttons. From the possibilities shown to the lab organizer, they can negotiate schedules with the students as soon as possible. As a stretch goal, there will be an option to request swaps between the TA's if one wants another's time slot. The identities would be anonymized to prevent any sort of potential peer pressure situations.

3 - SOLUTION VISION

To narrow down the solution, the project will be built as a web application holding several modules. We will host the application via an AWS server, which will hold a database to store the TA information, a web framework environment to ensure the ease of operation and manipulation of the database, and an optimization algorithm to automatically sort the database when any changes are made by the front end. The team has decided upon these technologies to implement our solution:

- AWS web hosting service, Lightsail
  - This will be the machine that hosts our website, allowing the application to be accessed any time
  - Comes with security guarantees provided by Amazon
- Django Web Framework
  - Will be running the main GUI functionality for our application
  - Will display all TA information and allow for responsive scheduling changes

    - Comes with out-of-the-box security features to ensure information protection
    - Comes prepackaged with many tools to speed the development process
- Optimization Algorithm
    - Will automatically organize the TA's into their classes
    - A table of scheduling information will determine how the algorithm places the TA's
- Admission from Django forms
    - "Forms" will be a functionality automatically provided by Django
    - Will allow the TA's to quickly input their schedules, qualifications, and account information for the database

With these technologies, we will be able to build a fantastic solution for most scheduling issues. It will automate the manual input and evaluation of all TA's and their attributes. Seeing the possible schedule arrangements using the Django web app will allow for great communication between the lab organizer and the TA's in order to account for unforeseen changes in the schedule. Not to mention the stability, reliability, and ease of access to the application, The Django web application will provide numerous safety features to ensure the privacy of its users and their information. The concept of this solution is applicable to many other issues related to scheduling, and could solve the scheduling problems of institutions world-wide.

4 - PROJECT REQUIREMENTS

For this project to be successful, the team has developed a set of domain-level requirements that lay out the features of the web application. In this next section, we will be discussing all of the known requirements, then breaking them down into two subsections: functional requirements and nonfunctional requirements. Each requirement will be presented hierarchically, beginning with the domain-level requirements and breaking them down into smaller, more detailed representations. Our application will supply the following domain-level requirements:

- Security that authenticates users and is accessible by different permissions
- A GUI for lab organizers and TA's
- Organize, manage, and schedule TA's for lab placement

Based on these domain-level requirements, we can now present more detail regarding functional and nonfunctional requirements.

4.1 - Functional Requirements

The application has a set of functional requirements to operate from. This section will outline the essential functions of the application in order to be considered complete. We will be examining the list below:

- User-based authentication and permissions
    - Permissions of TA's, GTA's, and lab organizers
    - Information security
- TA scheduling optimization
    - Scheduling based on qualification
    - Lab organizer customization
    - Distinction between TA's (GTA vs. TA, grader vs. lecturer, etc.)
    - Speed
- GUI
    - For TA's and lab coordinators to interact with
    - Renders data into a visual representation
- Data management system
    - Provide data to the GUI
    - Store information regarding TA's, semesters, and lab organizers
    - Persistent state
- Account configuration and setup
    - Lab organizer account creation
    - TA account creation
    - Access to the application

All five of these functional requirements will interact with each other. For example, the optimization algorithm will couple with the Redis database in order to visually represent the data in the GUI. Now that we have defined our functional requirements, we can examine each on a deeper level.

4.1.1 - User-based Authentication and Permissions

Users of the application will first be authenticated by the system and subsequently given different permissions based on their account status. The importance of strict roles and permissions are to ensure security, confidentiality, and to ensure the equal satisfaction of all TA's that are scheduled. The application will complete most of the scheduling work for the lab organizer while also allowing the organizer to make any changes necessary to fit the current context. The TA's will not have any direct permissions in terms of schedule changes, but will be allowed to see the entire schedule. Listed below are all the requirements for user-based authentication and permissions:

- Authentication
    - Handled by the Django web framework
      The system will examine a flag set within the account details and determine whether the user is considered a lab organizer or a TA.
- Lab organizer account type
    - Considered a "superuser" by the system
    - Able to make changes to the current schedule:
        - Move TA's, regardless of qualification
        - "Swap" TA's
          Swaps refer to switching time frames between two TA's that are both able to fit with their respective time constraints. This will be used if two TA's wish to switch shifts for both of their convenience.
        - View potential swaps or schedule changes when looking at the entire schedule
    - Able to edit weights of lab qualifications (see section 4.1.2)
- TA account type
    - Allowed to view the current schedule for classes offered during the semester
    - Unable to make direct schedule edits
    - Able to request swaps by the lab organizer. This ability will be anonymized between two TA's (see section 4.3.3)
    - Will be flagged as either a TA or a GTA, as a grader or a lecturer, and whether or not they have an existing contract with NAU (see section 4.1.2)

4.1.2 - Optimization Algorithm

The application will provide a way of organizing and placing TA's into their qualified positions. In the backend of the system will be an optimization algorithm that will contextually decide which TA gets assigned to which lab. This algorithm will contain a generalized formula for scheduling TA’s using a number-based scoring system. All TA’s hired for a specific semester will be evaluated through each semester and assigned a score based on their skill qualification. The TA with the highest score for a certain lab will be assigned to that lab.

- General constraints
    - TA’s not available during a specific time period will not be assigned to that time frame
      If a TA is only a grader, thus they are not required to be physically present in a lab, their time constraints will be ignored.
    - TA’s that score below a threshold designated by the lab organizer for a specific lab will not be assigned to that lab.
- Custom constraints/criteria and weight
    - When a lab organizer creates a lab object in the system, they will be asked to edit any existing constraints.
    - Each lab will, by default, utilize the generalized optimization algorithm.
    - Constraints, criteria, and criteria weight will be editable by the lab organizer to accommodate for specific lab conditions.
        - For example, if a lab organizer is responsible for staffing a lab that requires an above-average understanding of artificial intelligence, they will have the ability to weigh that criteria accordingly.
- TA scores
    - TA's will be assigned a score throughout the algorithm that will place them in their most qualified labs
    - The score will initially start at 0 and be incremented/decremented according to the defined lab organizer criteria
      The TA with the highest score will be presented to the lab organizer as the best option. The lab organizer will be able to edit any TA assignment as they see fit (see section 4.1.1)
- Priority
    - The algorithm will first assign TA's to labs that are more difficult to staff rather than overfitting qualified TA's.

        - For instance, labs that are expected to have less qualified TA's will be assigned first such that easier-to-staff labs are not filled with overqualified TA's.
    - Lab organizers will have the option of assigning a “priority score” to an individual lab.
    - The lab with the highest priority will be staffed first.
    - The algorithm will prioritize TA's based on their contract status with NAU.
        - For instance, a TA that has an existing contract with NAU must be assigned a lab. A TA that does not have a contract does not necessarily need to be assigned a lab.
- Speed
    - The optimization algorithm will take no longer than 15 seconds to organize less than 100 TA’s for a given semester

4.1.3 - GUI

The application will feature a GUI that will allow users to visualize TA scheduling as well as employ a set of buttons that provide a series of different functions. Both user account-types will be presented with a different version of the GUI upon login. Each user will be shown pages based on the work they are trying to accomplish.

- Login GUI
    - Will act as the homepage of the application
    - Will ask the user for their email and password
    - Will allow the creation of new lab organizer accounts
- GUI for Lab Coordinator
    - Scheduling page
        - Option to swap two TA’s
        - Option to remove a TA
        - Course list overview of the selected semester
        - History of changes that have been made to the schedule
        - List of contracted TAs and uncontracted TA’s
    - Page of TA’s
        - Lists information about TA’s
            - Previously taught courses
            - Evaluation from previous semesters
        - Current status of a TA account
            - Uncreated

        If a TA has not begun the creation of their account but an email has been sent, their status will be considered “uncreated”.
        - Missing information
        - Complete
          If a TA account contains all necessary information, their account will be considered “complete”.
- GUI for TA’s
    - View current schedule
    - Update personal information
    - Update course catalog
    - Update skills and qualifications
    - Update availability
- Speed
    - Either version of the GUI will take no longer than 15 seconds to fully load after a user logs in

4.1.4 - Data Management System

The application will be using a database package that will store three primary sets of data. The first will be referred to as the “semester pool”, the second as the “TA pool”, and the third as the “lab organizer pool”. Each lab organizer will be allocated a semester pool for each semester they are responsible for managing. All pools will be accessible by the GUI in order to provide a visual representation of the data for the lab organizers and TA’s. The following are the requirements of the data management system:

- Semester pools
    - Acts as a container for individual lab objects.
    - Each lab object will represent a single lab that needs to be staffed by the lab organizer for that semester.
- TA pool
    - Acts as a container for TA objects
    - Each TA object will represent a single TA.
- Lab organizer pool
    - Acts as a container for lab organizer objects
    - Lab organizer objects will contain all information regarding a single lab organizer.
- Maintain previous semester information

  - The database will maintain lab information for up to 2 academic years, or 4 semesters, as long as the lab is not being offered
  - In the event that a lab is not being offered for longer than 2 academic years, the data will be removed from the system. If the lab is then offered after this time period, a lab organizer will be required to create a new lab object with updated data.
- Persistent data state
  - The data management system will provide a method of persistence by storing data on the local disk of the hosting machine. The reason that this is mentioned as a requirement is due to the team's decision to use the database framework, Redis. As Redis is an in-memory cache based system, data persistence must be provided as its own function.

To better visualize the data management system, the team has provided Figure 2 below.

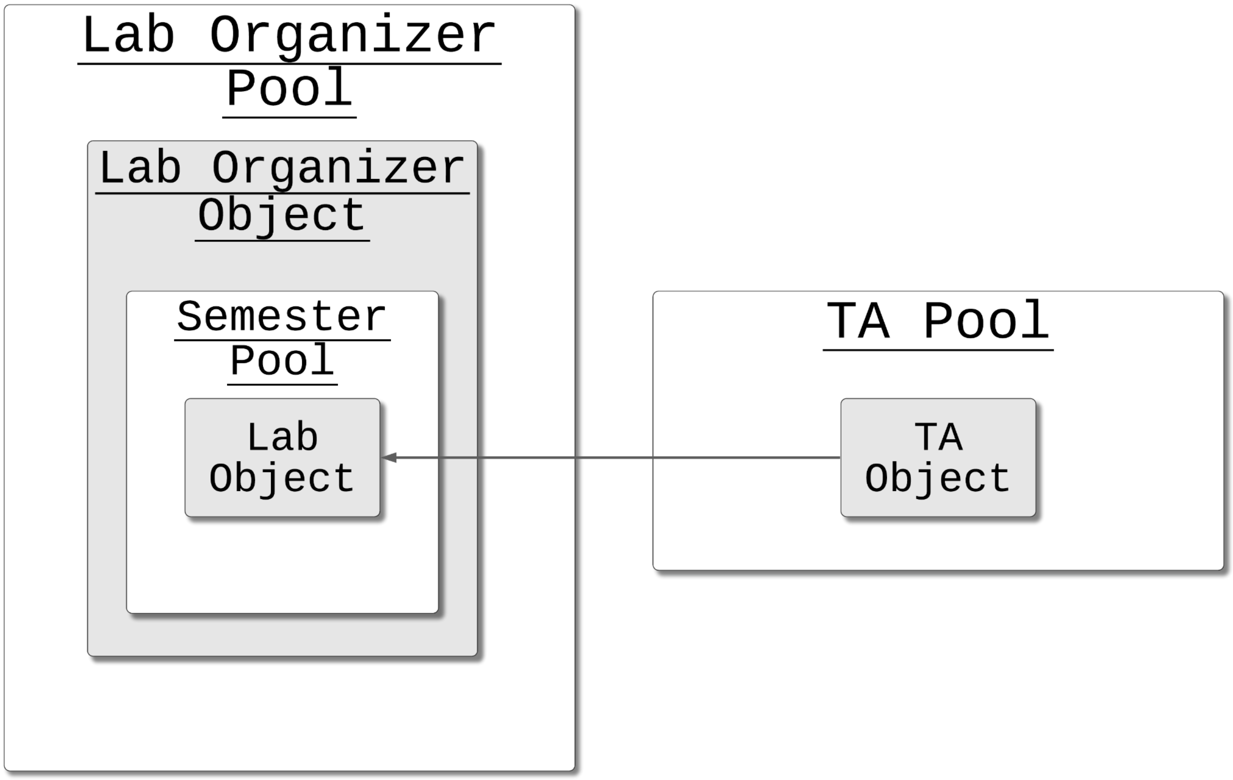


Figure 2. Data Management System

4.1.5 - Account Configuration and Setup

There will be two primary aspects of account configuration and setup. First, the lab organizer will be able to create an account for themselves to be able to moderate their labs. Second, each TA will be able to access the application in order to provide their personal data, allowing them to be scheduled within the system.

- Lab organizer account
    - Lab organizer accounts will have higher privileges than their TA counterparts
    - Lab organizers will create their own accounts directly from the login page of the application
    - Lab organizers will be required to provide a name, email, and password upon creation
    - Lab organizers will be able to edit the above information as necessary
- TA account
    - TA's will only be able to create an account at the discretion of a lab organizer
    - Lab organizers will have the ability to send an email directly to TA's, where the TA will be given a link to the application to create their account.
    - TA's will be required to provide their name, email, a password, and their student IDs. Further, they will provide information relevant to the semester they are going to be working: time availability, qualifications, and whether or not they have an existing contract with NAU.
    - TA's will be able to edit the above information as necessary
    - After a TA has an account, they will then be able to login from the homepage of the application

4.1.6 - Summary

The functional requirements are necessary to provide lab organizers the ability to schedule and manage TA's and TA's the ability to be able to view the labs they are assigned to. In summary,

- Users will be authenticated and given different permissions based on their account status
- TA's will be organized and scheduled via a score-based optimization system
- Lab organizers will be presented with a GUI to visually manage their accounts and schedule TA's. TA's will be presented with a GUI that outlines the labs they are responsible for
- Data will be managed and stored within the server that hosts the application
- Lab organizers will create their own accounts and be given the ability to establish TA accounts

4.2 - Performance Requirements

The web application will require a set of nonfunctional, or “performance” requirements in order to operate efficiently. These requirements outline how the application will perform within the entire system and are used to outline the entire process. We will be examining:

- Security
- Professional look-and-feel
- Modularity
- Sensitive data anonymization

First, we will be discussing security, which will be required to protect sensitive user information. Then, we will be looking at a professional look-and-feel. This requirement is necessary to provide an easily accessible and understandable environment for the user. Finally, we will explore modularity, which is required to ensure the extended lifespan of the application as well as allow the lab organizer more customization. Now, we can examine each requirement in greater detail, beginning with security.

4.2.1 - Security

The security of the web application is an extremely important non-functional requirement that will require a lot of attention from the developers. The list below will be our standards for the security of the web application:

- Protection against cross-site scripting
- Protection against cross-site request forgery
- Protection against SQL injection
- Protections against Clickjacking
- Protection against host-header violation
- Encryption through HTTPS

In order to ensure our web application is encrypted, we will be using “certbot” to secure the website with HTTPS. When sending out links for users to create their accounts, each will be personalized. Only one account can be created with one link at a time. If the wrong person gains access to the link, the issue will be easily fixed. The lab organizer webpage will be only accessible to them and will be only accessible by their credentials. As stated above, TA’s will only be able to read the whole schedule without making any direct changes to it.

The backend of the system will be a Redis database, which will be rendered and encapsulated by the Django web app. The only people with access to the database will be the developers, or the client will be allowed to have access to it if they so choose. When a user is created, their password will automatically be hashed and stored in the system.

4.2.2 - Look-and-Feel

The application will be used both for work and study. The team anticipates that lab organizers will be the primary users, thus spending the most amount of time on the website. The TA's will be secondary users, only using the application infrequently to check their schedules and request changes. However, both parties will be using the application and as such, the team will be providing a look-and-feel that promotes accessibility and extended periods of time using the application.

- Accessibility via look-and-feel
    - The application will have self describing GUI elements. For example, the login page will explicitly inform the user where to enter their username and password as well as feature a "login" button. Further, buttons will be labeled such that their functionality is evident without the user having to read any user manuals.
- Extended work periods
    - The main theme will be of cool colors with low saturation to promote less strain on the eyes
    - The application will be as "clutter free" as possible. By this, we mean that there will be no extraneous UI elements that offer little or no purpose. For example, there will be no sound effects attached to the website.

4.2.3 - Modularity

The application requires modularity for the purpose of extending its total lifespan. Inevitably, the team will eventually step away from the project. As such, the components of the system will be modular and serviceable by the owner of the hosting machine. The individual components will be as independent from each other as possible to allow the owner to easily replace or modify them.

- Optimization algorithm
    - The optimization algorithm will reside in it's own dedicated file on the server. Any other components that interact with it will be able to adapt to any changes made to the file. The owner will have full control over the content of the file.

- HTML files via Django
    - The HTML files will be dependent on Django to properly render them but will be written in a modular fashion such that they can be moved/directed to any other existing file on the server.
- Redis functions
    - Redis functions will be written so that they can be used wherever they are required. For example, a function that locates the specific ID of a TA will be accessible by any file designated to display that ID.

4.2.4 - Sensitive Data Anonymization

The client has requested the ability for TA's to request "swaps" between their already-assigned lab positions. This means that two TA's will be able to trade positions that they are both qualified for. The client has also requested that TA's remain anonymous from each other, which will require the team to hide identities when TA's are viewing their schedules via our application.

4.2.5 - Summary

In summary, elements of each performance requirement will either be indirectly or directly implemented via a series of existing technology, planning, and proper design. The application will be secure, protected from SQL injection, clickjacking, cross-site scripting, and more. The application will be designed to be easily understandable by a new user. The system will remain modular such that when the team moves on from the project, it will be serviceable and customizable by the owner of the hosting machine. Finally, sensitive TA data will be kept anonymous from other TA's.

4.4 - Requirements Summary

All requirements have been thoroughly planned, thought out, and a design has been decided on. While the team understands that requirements evolve as the project matures, we are confident that we will be able to remain agile and quickly adapt to unforeseen circumstances. With the design plan in place, the team will be able to implement each functional, nonfunctional, and environmental requirement at a speed that complies with the limited timeframe of the capstone experience.

5 - POTENTIAL RISKS

After or during development of the project, there will be inevitable problems the client and team will encounter. These issues could be caused by a service or user not acting as intended by the developers. In this section, the team will list out several potential mishaps, while also offering potential mitigations for each of them.

5.1 - Unreliable TA Form Information

When potential employees register an account to the web application, the lab organizer must assume that the TA's will provide correct information. This will not always be the case. They may make a mistake in their schedule constraints, say they have experience in classes that they don't, or they can give an inaccurate evaluation on their proficiency in assisting for a class. This will affect how the algorithm scores and places them in the schedule, which could create a butterfly effect for other students. The client wants large changes to the schedule to happen as close to the beginning of the semester as possible in order to avoid changing TA's that are already well established with their labs.

To mitigate the issue, the application will have to allow for the changing of constraints in a convenient fashion. The option to change specific constraints should be available when the lab organizer clicks on the specific TA. The client would also be encouraged to double check with their TA's to make sure their schedule constraints are sound. In order to mitigate incorrect proficiency information, the lab organizer should evaluate the TA's at the beginning of the semester to make sure they are well suited to assist with their class. Any further changes to constraints will affect the entire schedule, so it is encouraged that those types of changes are completed as soon as possible.

5.2 - Bad account creation

Another issue with account creation is ensuring that all accounts created are valid, and that there are no duplicates. There is a possibility that the wrong person may receive the wrong link to register. When this person registers, they will be injected into the database along with all of their personal information. If this is wrong, it could potentially clutter the database with people that aren't employed or have the wrong schedule constraints.

In order to negate this, the team has decided that only one account will be allowed creation per link sent by email. If there is a bad entry in the database, the intended recipient will likely reach out to the lab organizer to fix the issue since they will be unable to register. The bad recipient will be removed in place of the intended one. This will also completely fix the issue of potential exploits of putting multiple entries of one person into the database.

5.3 - Inefficient GUI Workflow

There are a lot of design requests for this web application, along with many visual elements to better understand the bigger picture of the scheduling process. These will be great tools for the lab organizer; however, the team is concerned that the great number of design choices may clutter the screen, and result in a GUI that may be difficult for some lab organizers to learn properly. For a stretch goal, the team wishes to allow many users to use this tool, so it should be as user-friendly as possible.

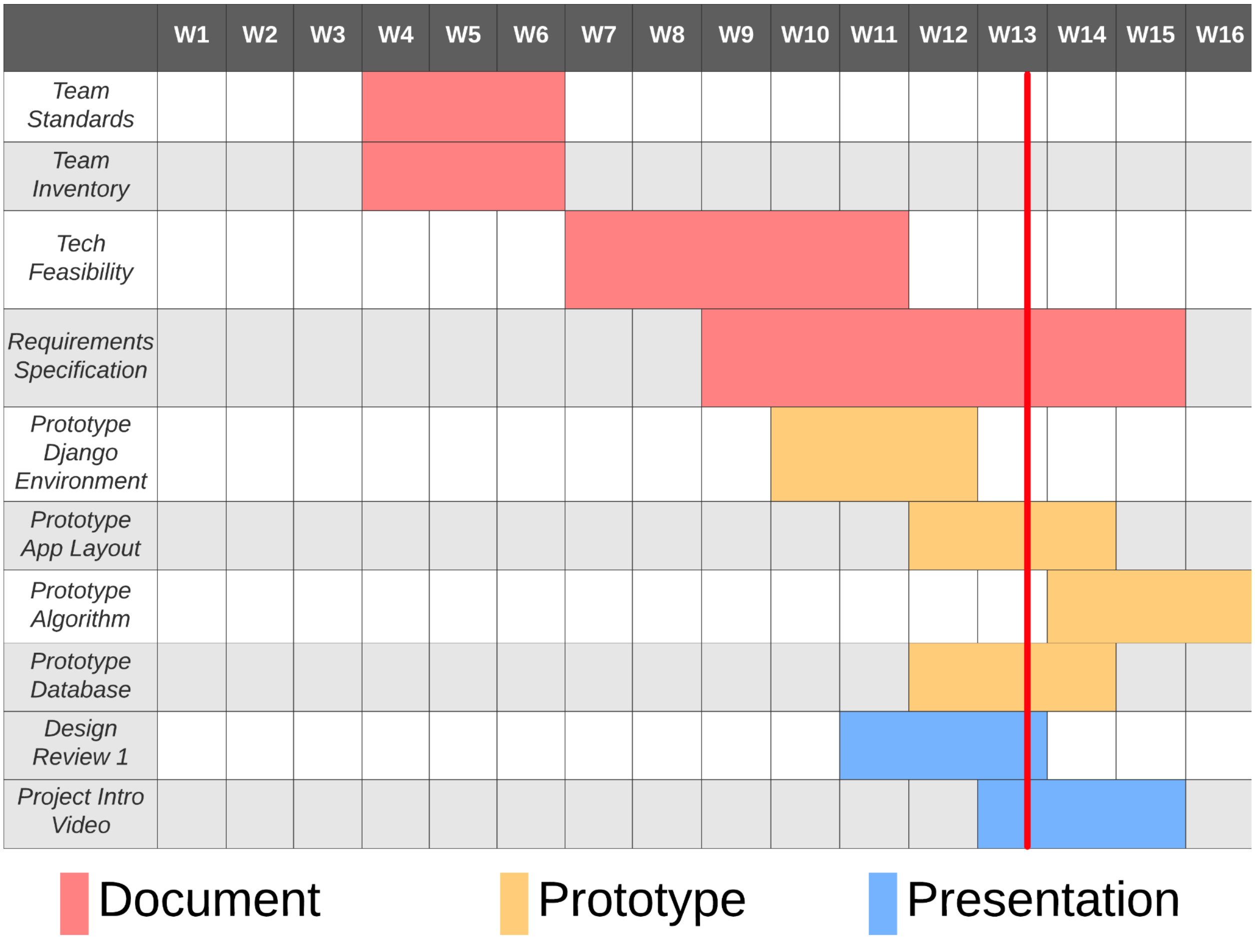
To mitigate this potential issue, a tabbed environment may be the best solution to simplify the workflow of the software. Certain aspects of the dashboard that are similar will be placed into a separate tab (Ex: the schedule table and potential changes will go in one tab, while change history and specific TA information will go in a different tab). The team is not sure that this will be necessary; however if this issue presents itself, we will have a tabbed environment prepared for deployment.

5.4 - Summary

The above mitigations will not be a silver bullet to the issues, but they will decrease the chance of significantly larger issues presenting themselves in the future for the lab organizer. Unreliable TA evaluation, bad account creation, and bad GUI workflow are all issues with medium severity, but there exist solutions that the team can assist with to ensure the satisfaction of the client.

6 - PROJECT PLAN

This section will identify the current plan and outline major milestones for the remainder of the project. Below are two Gantt charts, one that identifies the Fall 2021 semester and one that identifies the Spring 2022 semester.



	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16
Team Standards				🟥	🟥	🟥										
Team Inventory				🟥	🟥	🟥										
Tech Feasibility							🟥	🟥	🟥	🟥	🟥					
Requirements Specification									🟥	🟥	🟥	🟥	🟥	🟥	🟥	
Prototype Django Environment										🟨	🟨	🟨				
Prototype App Layout												🟨	🟨	🟨		
Prototype Algorithm														🟨	🟨	🟨
Prototype Database												🟨	🟨	🟨		
Design Review 1											🟦	🟦	🟦			
Project Intro Video													🟦	🟦	🟦	

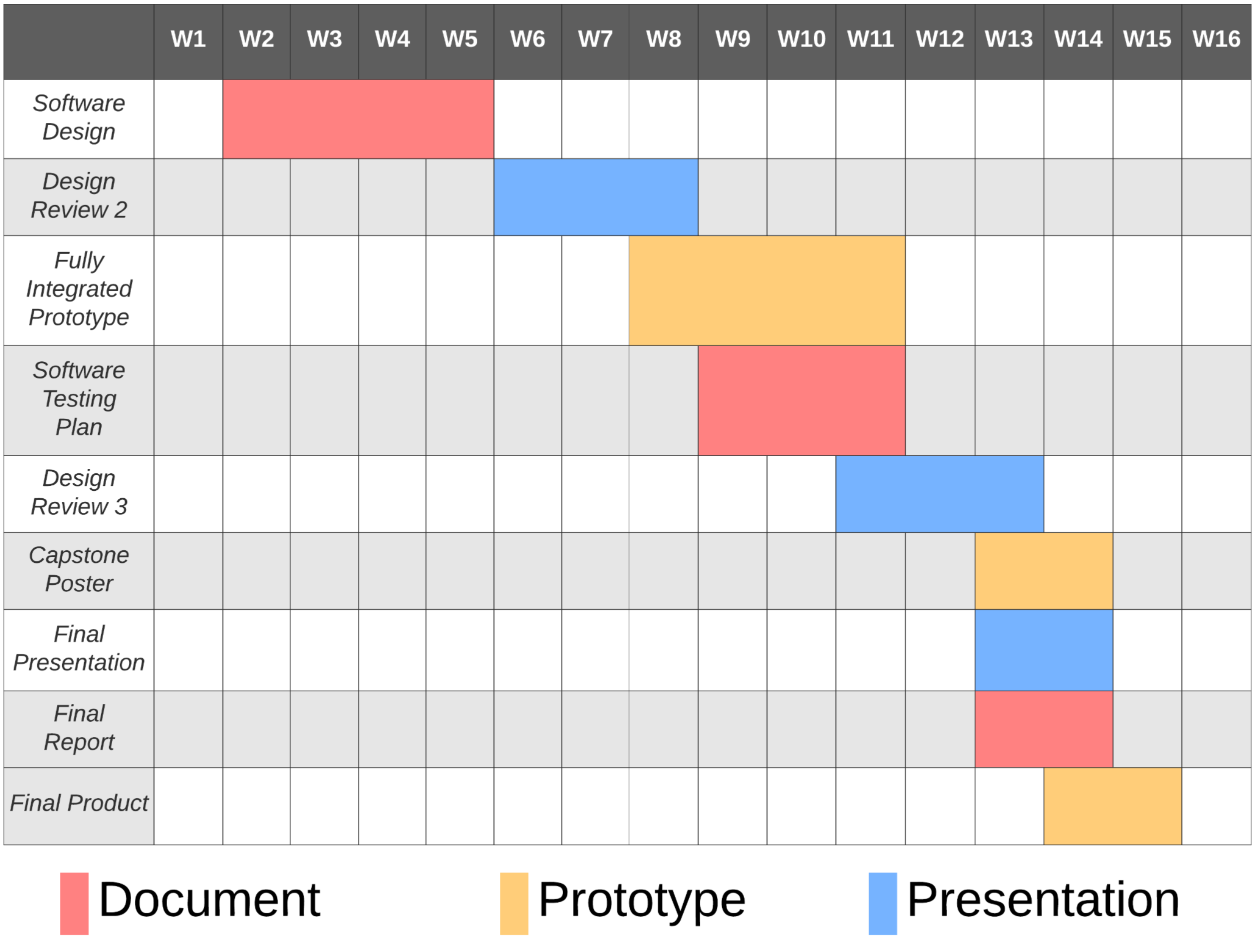
🟥 Document 🟨 Prototype 🟦 Presentation

Figure 3. Fall 2021 Semester

Looking above at Figure 2, at the time of writing this document, we are in week 13 of the semester. In previous weeks we have completed our Team Standards document, our Team Inventory document, and our Technological Feasibility document. We have begun prototyping the Django environment and completed our Project Introduction. For the rest of this semester, our major milestones will be:

- Identifying any remaining project requirements before the end of week 15
  - Involves the requirements being confirmed and accepted by the client

- Prototyping the application, ensuring that each requirement is technically feasible by the end of week 16
    - We will be establishing a connection between Django and Redis
    - We will create a database environment that is capable of reading/writing data provided by the client
    - We will create the first iteration of the optimization algorithm



	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16
Software Design		🟥	🟥	🟥	🟥											
Design Review 2						🟦	🟦	🟦								
Fully Integrated Prototype								🟨	🟨	🟨	🟨					
Software Testing Plan									🟥	🟥	🟥					
Design Review 3											🟦	🟦	🟦			
Capstone Poster													🟨	🟨		
Final Presentation													🟦	🟦		
Final Report													🟥	🟥		
Final Product														🟨	🟨	

🟥 Document 🟨 Prototype 🟦 Presentation

Figure 4. Spring 2022 Semester

Next semester, our primary focus will be programming the final application. We will periodically develop Design Review assignments as well as writing two more major documents, Software Design and the Software Testing Plan. Our major milestones for this semester will be:

- Officially hosting the application via AWS
- Fully integrating the prototype into the final product

- Completing a checklist of all requirements outlined above
- Delivering the final product

7 - CONCLUSION

In conclusion, this project is designed to aid Doctor Fofanov in the scheduling and management of TA's. The problem lies in the time consuming nature of manually scheduling large teams. Our client currently manages TA's using an Excel spreadsheet, which takes him roughly 30-40 hours per semester; however, the total time required is dependent on a number of different variables and has the potential to increase indefinitely. Our proposed solution will be creating a web application that allows users to create, edit, and automatically sort TA schedules for a given semester. Users will be presented with a visual representation of the data they have created, access the system at any time, and different configuration features to better suit their needs. By establishing the sets of requirements outlined above, the team is confident in their research and understanding of the solution vision. We are excited to be working on this project and are passionate about solving this issue not only for our client, but for other organizations with similar problems.