C&I Doctoral Tracking Tool
As-Built Report

12/15/2022

Project:
C & I Doctoral Tracking Tool

Project Sponsors:
Gretchen McAllister & Michele Benedict

Faculty Member:
Michael Leverington

Team Name:
What's Up Doc

Team Members:
Adam Larson (Lead), Brandon Shaffer, and Eddie Lipan

Team Mentor:
Daniel Kramer

Table of Contents:

1.0 Introduction

Attending and graduating college constitutes some of the most formative and adversarial years of a person's life. According to a census taken by the Education Data Initiative in 2021, 3.1 million post baccalaureate students are enrolled in graduate programs at universities in the United States[1]. Generally these students are striving to learn and better themselves each day of attendance but many will likely face similar adversities, such as timeliness, deadlines and examinations, all while balancing social lives or work schedules. With all these stressors, graduate students should be able to reliably track and manage their graduate program without fear of adding additional stress.

Most undergraduate courses are implemented through Learning Management Systems (LMS) such as Blackboard Learn or Canvas, which act as course delivery and grade feedback tools. This is possible as most undergraduate courses classify as "pass/fail" or fall under some version of the standard A-F grading system. Assignments and examinations for undergraduate courses are generally quite rigid in that they pertain to lectures or chapters of an assigned reading, thus they may be outlined and planned before a semester even begins. This rigidity is absent in graduate school as graduate students perform activities that are less quantifiable, such as shadowing mentors and researching. Therefore, it is considerably harder to deliver the graduate program using a LMS.

In order to attain candidacy in the C&I Ph.D program, students must complete a minimum of 60 graduate-level course units, where the average course is 3 units. C&I graduate students must also complete professional development requirements, comprehensive written and oral exams, a qualifying research paper, get the approval and assignment of a dissertation committee, and get the submission of an approved dissertation prospectus.

The Coordinator of the Curriculum and Instruction (C&I) doctoral program at Northern Arizona University (NAU), Gretchen McAllister, Ph.D., and Administrative Services Assistant, Michele Benedict, have both experienced the shortcomings of Blackboard Learn firsthand. As a result, neither is able to track the milestones of their graduate students using a standard LMS. At present, Gretchen and Michele are having graduate students funnel all deliverables to a single computer via email which are then

[1] Hanson, Melanie. "College Enrollment Statistics [2022]: Total + by Demographic." *Education Data Initiative*. 22 Jan. 2022. <https://educationdata.org/college-enrollment-statistics#:~:text=Report%20Highlights.,students%20are%20in%20graduate%20programs.>.

held locally in folders labeled with the student's name. No further organization is performed other than storing the deliverable into the appropriate students' folder. Graduate student grading and submission inquiries are achieved by emailing Michele directly, who then browses the files locally, and provides feedback via email. The current program workflow is inefficient and creates a large bottleneck for all parties involved.

Team What's Up Doc's solution is an intuitive website application, allowing graduate students to track their progress through the graduate program up until candidacy. The website application will also allow administrators to view student progress and analyze how each student is doing within the program based on their remaining milestones. Given that some users may not have a technical background, the app would require an intuitive, visually appealing user interface making it easy to upload deliverables and track progress, all while avoiding the time-consuming correspondence with Michele. Another goal of the website application is to offload data entry duties from the administrators and onto the students themselves.

The purpose of this As-Built Report is to accurately narrate where Team What's Up Doc's current product stands in terms of, the overall development process, initial and final requirements, final architecture and implementation, testing, overall project timeline, future work to be done, and finally an appendix detailing all technologies required to work on this project.

2.0 Process Overview

Team What's Up Doc did have specific roles throughout the duration of the project. Adam Larson was the team lead, back end developer, and service connections manager. His main duties as team lead were to interface with our faculty mentor Dr. Michael Leverington and graduate mentor Daniel Kramer when need be. He would also be the main avenue of communication between Team What's Up Doc and our clients Dr. Gretchen McAlliser and Michele Benedict. As team lead, Adam would also ensure that deadlines were met and that all work was equally partitioned. In the event that there were any conflicts or production issues, Adam would address those issues with the team member, and/or team, accordingly. As back end developer and server connections manager, Adam was in charge of managing the MySQL database, hosting the server through AWS and connecting the front end and back end features via the Spring framework.

Brandon Shaffer acted as the chief editor, GitHub repository manager, and front end developer for Team What's Up Doc. As chief editor, he was in charge of formatting and editing all documents before they were to be submitted. This included adding page numbers, grammar checking, formatting for continuity, and doing a final once over on every document before submission. As the GitHub repository manager, Brandon would have to keep an eye on the repository during the development cycle and make sure that the main branch is correct and up to date. This entails thoroughly vetting all commits and pull requests as they are made so as to avoid conflicts within the codebase. Brandon would also occasionally clean up the repo by deleting any outdated or negligible files and directories from the repository. Finally, as a front end developer, Brandon was helping on all phases of the front end development project cycle. Brandon wrote code for both the team website and the project website. For the project, Brandon was responsible for coding the landing page and the administrator home page. This entailed coding the index.html landing page, the admin.html admin home page and the admin.js JavaScript function file.

Eddie Lipan acted as a vital front end developer for Team What's Up Doc. Eddie was also helping on all phases of the front end development project cycle. He would split the team website workload with Brandon. As for the project itself, Eddie was in charge of creating the student home page. This entailed coding the home.html student home page and the home.js JavaScript function file for the student home page.

As for the software development process adopted for this project, Team What's Up Doc chose not to adhere to a formal software development process such as SCRUM or Agile. Instead, Team What's Up Doc chose a more modular approach. They

chose to split the project into three overarching modules: the front end, the service connections, and the back end. We found this process to be the most intuitive and approachable from the start and allowed us to categorize ourselves before any work had begun to be distributed. In the end, as stated above, Brandon and Eddie decided to tackle the front end features as a pair, and Adam chose to tackle the service connections and back end features. We chose this direction due to each other's personal experiences and familiarities with the certain technologies that would be used during the development process. Both Brandon and Eddie already had extensive experience with HTML, CSS, and JS web programming. Adam had experience working with MySQL databases and Spring framework.

Finally, Team What's Up Doc chose to adopt GitHub and Discord as our main communication and task management tools that aided in the development process and with versioning control. GitHub was an invaluable asset in maintaining accurate versions of our project throughout its life cycle. There were a handful of times that a team member accidentally merged a pull request into the wrong branch or merged an outdated directory, and being able to refer to a recent version of the project through GitHub saved the team many headaches. Discord also proved to be a terrific communication avenue for Team What's Up Doc as we were able to specifically create a channel for any and all aspects of the project that might require discussion. Example Discord channels include: session-planning, deliverable-stuff, front-end-related, back-end-related, service-related, pull-requests, etc. These channels were used in abundance and it helped keep certain aspects of the project modular and easier to address in the future and refer back to.

3.0 Requirements

In order to acquire the requirements for this product we held a meeting with the clients. At this meeting we discussed what it was that the clients pictured as a final product, clarified specific requests, and negotiated speculative end points and stretch goals. After the meeting our team also discussed amongst ourselves the more technical aspects of the requirements. The final requirements we landed on can be divided into three categories: functional, performance, and environmental.

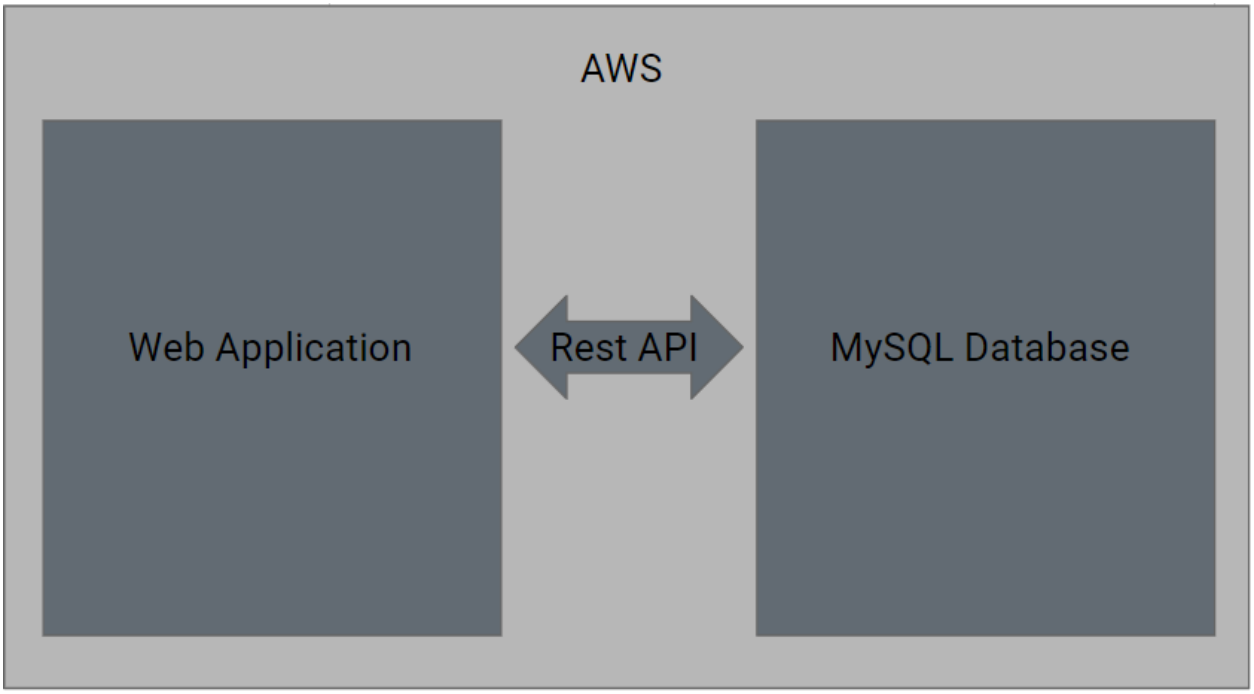
Our functional requirements, ones that specify what the product needs to accomplish, can be summarized as two levels of access separated by two separate pages with interactive GUI's. The two pages would be used to differentiate between two levels of access. Student users would be able to upload and download from the database for themselves. Faculty users would be able to access the files of all the students as well as view the overall progress of the students. The GUI's would need to be simple enough for less technically experienced users to still have functionality with the product.

When deciding whether to prioritize speed or reliability as a performance requirement, we landed on reliability. Our reason to focus on the reliability of our product was that the majority of time using the product would be inherent on factors we could not control. Since most files that are expected to be uploaded to or downloaded from the database would be relatively small, and the speeds for those actions would be dependent on the internet speeds of various users, we decided that we should focus on factors that we could control. Ensuring that buttons clicked would always respond the same way, that visual signals were accuratley reporting data, and that the correct files were always uploaded and downloaded.

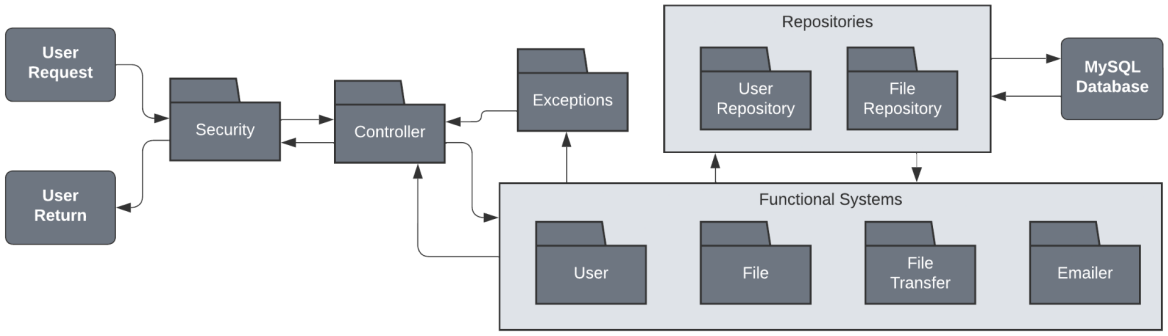
Our environmental requirements, or requirements specified from our clients and users, were more aligned towards the aesthetics of the product. Our color palette was required to follow the NAU marketing guidelines, limiting our options for color schemes. Our future users would also potentially contain people with color blindness or other conditions that can affect their ability to use our system. With these in mind we had to ensure that our GUI was designed with diversity in mind.

After determining our functional, performance, and environmental requirements, we began brainstorming our overall product design. We needed to ensure that the database was fully functional, that the server was reliably communicating between our website and the database, and that the website had to be extremely user friendly.

4.0 Architecture and Implementation



The above graphic shows the basic architectural design of our product. Both the web application's files and the database would be hosted on the same server, which for the development process was an Amazon Web Services server. We decided on using MySQL to build the database due to our past experience with the software, and Spring's Rest API to handle communication between the database and the web application. The web application serves as the GUI for the database, hiding the necessary requests and data behind a user-friendly interface.



The above graphic displays the components of the back-end of the application. A standard user request will be passed through security where the controller will then send the request to the correct functional system. The functional system then interacts with the corresponding repository in the database. The response is passed back to the

controller and back to the user. Any exceptions are caught before the controller, and an error message is passed to the controller instead.

As-built we do not have google sign-in implemented, and as an extension we do not have a security protocol in place. Due to delays in acquiring a secure server to house the database, we were unable to implement google sign-in in time to incorporate it into the rest of the application. Our aesthetic design for the front-end was constantly changing and evolving as we consulted with the clients in order to meet their mental image of the final product.

5.0 Testing

Our application's testing was split between manual and automated. The front end and database were manually tested as necessary, while the server was automated through internal mechanisms. Manual testing was limited to essentially a "change and check" on web design for proper display, format, and operation, and a quick run of standard SQL statements to ensure returns were as expected. Automated tests on the server however were far more involved. Unit and integration tests were used to varying degrees using the Spring frameworks built in systems, as well as JUnit and basic Java.

5.1 Unit testing

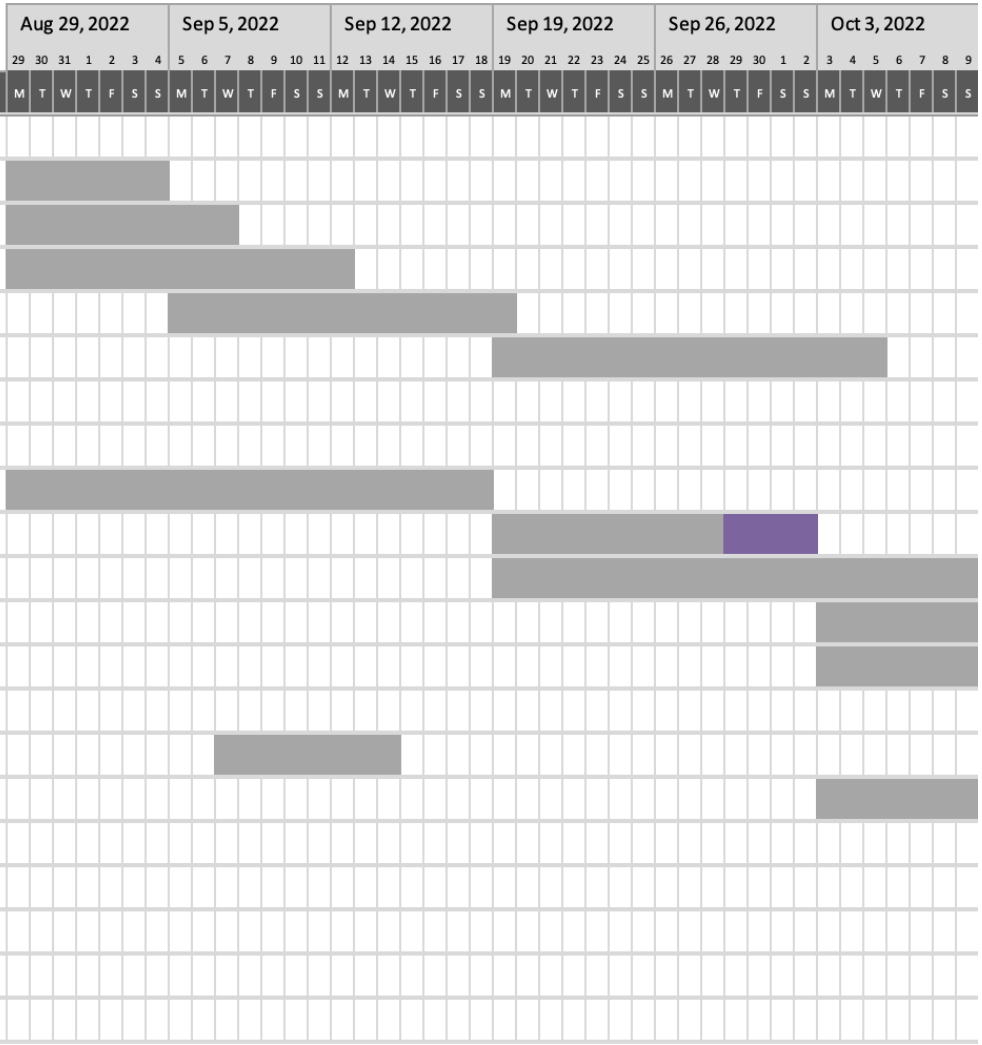
1. There were many methods to test on the server and each required setup using Spring's @Mock annotation to allow proper functionality. This annotation created a mock object that simulated an incoming request to the controller.
2. Mock objects needed to be created for each endpoint on the controller as each had their own input that varied from the others.
3. The expected output was given and the test cases compared and reported passing or failing when run individually.
4. Spring displays test cases and status on run leading belief that a generated test case log could be passed into the program at run time.
5. Fringe cases were included into each method's tests to determine if unexpected results were possible.

5.2 Integration testing

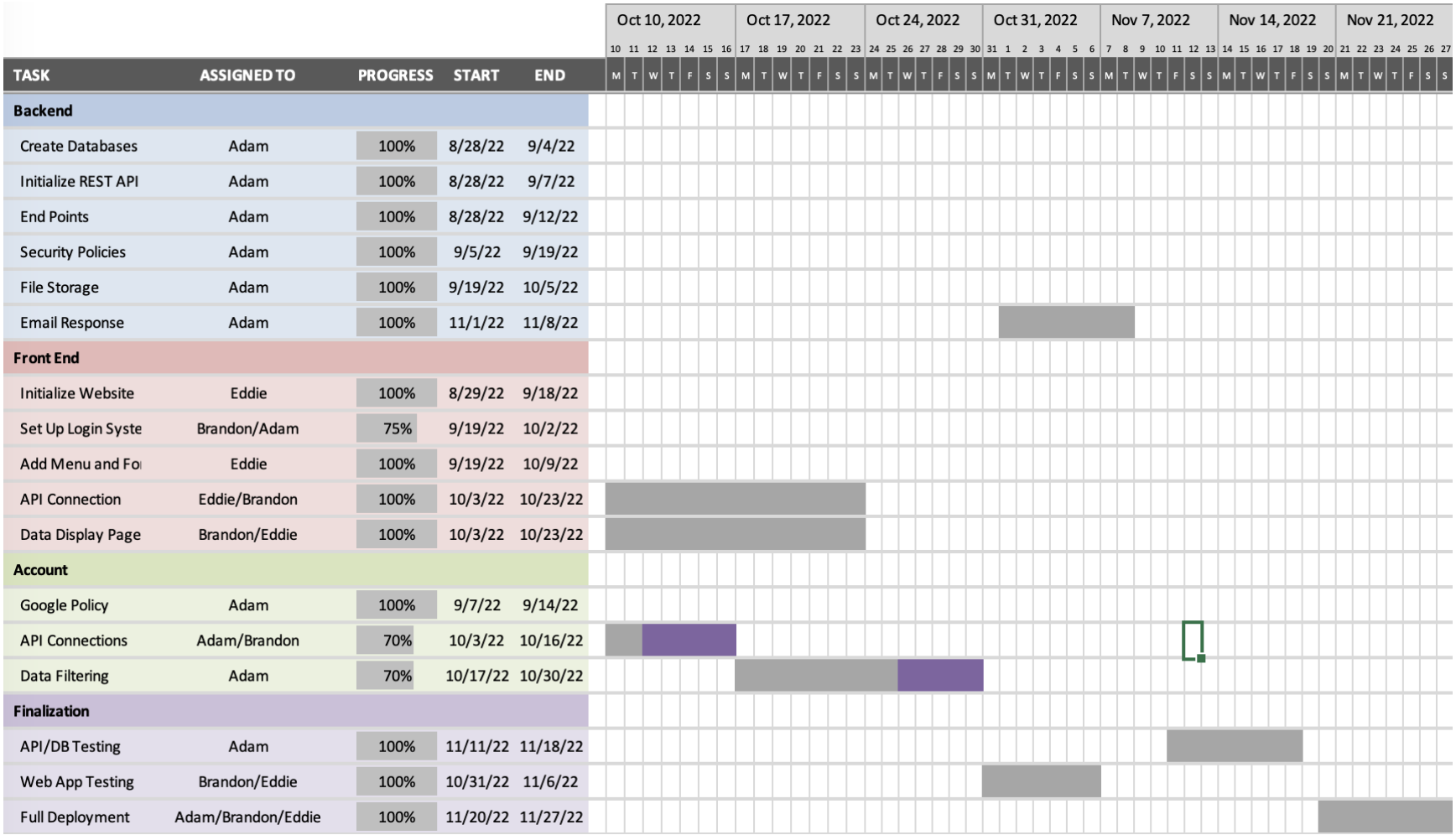
1. Two primary tests were included in this to ensure proper communication between the database and the server via the JPA repository system. This could be considered an end-to-end test as well as this module was the only point of connection within the server.
    a. A security module was also included, though its interaction was limited to incoming requests as the login system was postponed.
2. Testing of this system was almost identical to unit testing due to the Spring environment. Mock objects are passed through and returned simply via the same channel as above.
3. These objects are capable of being acted on identically to a standard request and returning the modified data to compare to the expected output.
4. Fringe cases were included into these as well, providing insight to possibly unexpected results.

Project Timeline - Fall p1

TASK	ASSIGNED TO	PROGRESS	START	END
Backend				
Create Databases	Adam	100%	8/28/22	9/4/22
Initialize REST API	Adam	100%	8/28/22	9/7/22
End Points	Adam	100%	8/28/22	9/12/22
Security Policies	Adam	100%	9/5/22	9/19/22
File Storage	Adam	100%	9/19/22	10/5/22
Email Response	Adam	100%	11/1/22	11/8/22
Front End				
Initialize Website	Eddie	100%	8/29/22	9/18/22
Set Up Login Syste	Brandon/Adam	75%	9/19/22	10/2/22
Add Menu and Fo	Eddie	100%	9/19/22	10/9/22
API Connection	Eddie/Brandon	100%	10/3/22	10/23/22
Data Display Page	Brandon/Eddie	100%	10/3/22	10/23/22
Account				
Google Policy	Adam	100%	9/7/22	9/14/22
API Connections	Adam/Brandon	70%	10/3/22	10/16/22
Data Filtering	Adam	70%	10/17/22	10/30/22
Finalization				
API/DB Testing	Adam	100%	11/11/22	11/18/22
Web App Testing	Brandon/Eddie	100%	10/31/22	11/6/22
Full Deployment	Adam/Brandon/Eddie	100%	11/20/22	11/27/22



Project Timeline - Fall p2



TASK	ASSIGNED TO	PROGRESS	START	END
Backend				
Create Databases	Adam	100%	8/28/22	9/4/22
Initialize REST API	Adam	100%	8/28/22	9/7/22
End Points	Adam	100%	8/28/22	9/12/22
Security Policies	Adam	100%	9/5/22	9/19/22
File Storage	Adam	100%	9/19/22	10/5/22
Email Response	Adam	100%	11/1/22	11/8/22
Front End				
Initialize Website	Eddie	100%	8/29/22	9/18/22
Set Up Login Syste	Brandon/Adam	75%	9/19/22	10/2/22
Add Menu and Fo	Eddie	100%	9/19/22	10/9/22
API Connection	Eddie/Brandon	100%	10/3/22	10/23/22
Data Display Page	Brandon/Eddie	100%	10/3/22	10/23/22
Account				
Google Policy	Adam	100%	9/7/22	9/14/22
API Connections	Adam/Brandon	70%	10/3/22	10/16/22
Data Filtering	Adam	70%	10/17/22	10/30/22
Finalization				
API/DB Testing	Adam	100%	11/11/22	11/18/22
Web App Testing	Brandon/Eddie	100%	10/31/22	11/6/22
Full Deployment	Adam/Brandon/Eddie	100%	11/20/22	11/27/22

6.0 Project Timeline

The previous two images show our projected timeline for the Fall semester with some dates ranging back into early summer. Due to a rather late start, many deadlines were missed and were made up later in the semester. Changes throughout the project made it rather difficult to meet deadlines as well.

The largest two categories of work were split into the Front and Back End systems. These were obvious choices as the line between them is rather distinct. Initial job assignments had Eddie doing the website, Brandon setting up the server, and Adam managing the database and setting up security systems. This changed shortly before summer to have both Eddie and Brandon working on the website with Adam doing everything behind the scenes.

There was arguably one key phase (aside from finishing it) where the work was most crucial. This was the first few weeks of the Fall semester wherein the server was up and running with partial functionality and available for the website to connect to while being formatted and created.

Notable milestones that were significant but not entire phases were related to web connections via TLS and managing CORS and CSRF permissions. These permitted the web page to function and retrieve data properly to deliver to the end user. Additionally receiving a server for use was a rather large step in and of itself. NAU had been contacted in February for assistance and nothing was able to be hosted properly until late October.

7.0 Future Work

7.1 Desired Features

Throughout the duration of the project, many last minute changes were requested that were not practical to implement with the amount of time and work left to complete. As many as possible were picked up and completed, though below are others that were desired in future iterations.

- Additional roles - Currently there is a flag for if a user is an administrator or not. A desired feature that came up near the end of development for a future iteration was a role for students, faculty/advisors, and administrators. The administrator and advisor roles would be very similar in view, though have a limited scope in granting access to new students.
- More emails - An automated email system was implemented to provide students with confirmation that their file was successfully uploaded to the system. Once working, our clients liked the idea and requested they receive a notification email when a student completes a phase so they can start processing that student's completion without needing to wait for the student to contact them.

7.2 Requested Features

Due to unforeseen circumstances, several features intended to be in the final version had to be cut from development to meet time constraints. One of these was unfortunately a very core function, though arrangements have been made to remedy this in coming weeks.

- Login system - This was a core functional requirement necessary for multiple students to be able to track their information and could not be finished. There were several factors that prevented this, and it could have perhaps been completed if these were handled differently, but it ultimately was unable to be included. The upcoming work is focused on this feature and will provide the desired use for the Spring semester.
- Accessibility - Meeting ADA requirements was a requested feature, primarily with the option of color customization for those with colorblindness or an option to have things read. Due to time constraints and additional requests that were deemed higher priority this did not make it into the final product.

Also of worthy future consideration is a more permanent host. This does not fall into the previous categories but is a situation that needs to be addressed. With the product being hosted on a temporary CS owned AWS server and NAU unwilling to make accommodations, another solution needs to be considered. This is slated to take place alongside the login system in coming weeks.

8.0 Conclusion

Successfully completing a graduate program is no small task, and it becomes even more challenging if a student has trouble tracking their progress within the graduate program. Before Team What's Up Doc was formed, the administrators of the C&I Doctoral Program were requiring all graduate students to submit deliverables via email. Once a deliverable was submitted via email, the administrator would save the student's deliverable in a local folder, named after each student, on a single local computer's desktop. All inquiries about the program had to go through this single administrator and her email. This workflow created a large bottleneck for both the administrators and the graduate students.

Team What's Up Doc aims to reimplement the data management process for graduate students enrolled in the NAU C&I doctoral program through an easily accessible website application. A College of Education website application would introduce an efficient data management mechanism for C&I graduate students and offload a tremendous amount of stress for our clients, Gretchen McAllister, Ph.D., Coordinator of the NAU C&I doctoral program and Michele Benedict, Administrative Services Assistant. Not only will this application reduce the stress of our clients, but if utilized properly, it will greatly reduce the stress associated with tracking a graduate level program and give the students valuable feedback on their progress within the program. Students will be able to upload/download/delete PDF files for each individual milestone within each phase. As files are uploaded, the taskbars will change color to reflect that a milestone has been updated. This website will also serve as a valuable analytical tool for both the students and the administrators. Students will be able to accurately analyze how they are doing within the graduate program based on their completed milestones, and administrators will be able to analyze the graduate program phase by phase, for each student. Administrators will also be able to export relevant data sets to useful formats such as CSV, Excel, and PDF for further use.

With further refactoring and refinement, Team What's Up Doc firmly believes that this C&I Doctoral Tracking tool can be an invaluable tool to the C&I Doctoral program. We also believe that with the proper refinement, this website application can be made to service all the graduate programs in need of a doctoral tracking tool here at NAU. This would entail that any graduate program can be made into its own deployable doctracker.org website with just a few text changes. Our product will save both our clients and their graduate student's loads of time and headache simply due to the independence and accountability that the product demands. Students are responsible for tracking and submitting their own work throughout the graduate program, which in turn, allows the administrators to view their progress within the program asynchronously.

Overall, Team What's Up Doc found the Capstone Experience to be a tremendous, but rewarding, challenge. We were able to accurately participate in an industry situation that closely mimicked the projects we would soon come to work on in the future. We gained valuable insight on working with a group and coding in a team setting. We got great experience working with our client on a weekly basis by providing consistent work feedback and receiving consistent feedback from them as a result. Team What's Up Doc grew immensely as developers, due to the application of core programming concepts, design concepts, and languages taught here at NAU. We also had to expand our knowledge in certain areas of the project which gave Team What's Up Doc even more on the job learning experience and exposure for future projects.

Lastly, Team What's Up Doc would like to formally thank our CS Faculty mentor Dr. Michael Leverington and mentors Tomos Prys-Jones, Anirban Chetia, and Daniel Kramer for their guidance throughout the Capstone Experience. We would also like to give thanks to our wonderful clients, Gretchen McAllister, Ph.D., Coordinator of the NAU C&I doctoral program and Michele Benedict, Administrative Services Assistant, for believing in us and providing us with this invaluable learning opportunity.

9.0 Appendix A - Development Environment

9.1 Hardware

Front End
Development occurred on a variety of Windows machines. The front-end website has minimal hardware requirements (HTML, CSS, JS are not demanding) and has been shown to run on Google Chrome, Microsoft Edge, and Mozilla Firefox.
Back End
Development took place on a 2020 M1 MacbookPro with 16GB of RAM. Physical requirements are largely irrelevant to run the server/database as they are fairly light. The largest constraint is likely the processor as it was crashing on an AWS instance due to load from the MySQL server - Professor Leverington said he added another ‘core’ for us to use, so the processor here is assumed. Previous recent course work involving Spring has been successfully run on a 2013 intel based MacBook with little difference save for initial boot time of the program (5 seconds to 9 seconds or so).

9.2 Toolchain

Front End
The front end uses html, javascript, and css for its functionality. To make requests to the database, we utilized the Fetch API from javascript. Fetch was the easiest request method to understand that we found. As far as the data tables go, they were built using DataTables and jQuery, with Bootstrap 5 to style. Lastly, Team What’s Up Doc’s front end developers utilized the Google Chrome Web Development Tool to aid them in successfully coding the web pages for this project.
Back End
Four tools were primarily used during development of the application, IntelliJ, Maven, MySQL, and Spring.io. IntelliJ was the IDE of choice as it is a higher end commercial choice than Eclipse. Within this environment only one plugin was used called JPA buddy, which assisted in determining the availability of built in commands and creating additional database calls when necessary. Spring.io was the initializer for our Spring project. This initializer provided the full project as a zip file, with chosen dependencies, Java version, project build tool, executable choice, and naming conventions. Creation of this from scratch would be very challenging and it provided a large amount of saved time. Maven was selected through this as the build tool and allowed management of the dependency path

and compiling to a JAR executable format quickly and effectively. Finally MySQL was the database of choice as it's both simple to use and familiar through other courses.

9.3 Setup

Front End

Website

1. Download Website Folder from the repository onto your host server
2. Open the project folder in your IDE of choice.
3. Check the URLs in the fetch requests on both home.js and admin.js in the scripts folder for the correct domain name linking them to the database.
4. Check the URIs used in index.html, admin.html and admin.js! If the domain or any redirect URI domain changes, they must be added to the Google Identity Services Client dashboard.

Back End

Server

1. Go to spring.io
2. Fill out the form and dependencies as desired
3. Unzip the downloaded file into the location of choice
4. Open Java IDE and top level folder of project
5. Wait for Maven dependencies to be included and the project to be built
6. Development environment is ready to be modified

Database

1. From command line or terminal (Unix based - did not use Windows) check if MySQL is present with 'mysql --version'
2. If not present use 'sudo apt install mysql-server' else skip
3. Login as root with 'mysql -u root -p'
4. Create a new database with 'create database genericDBName'
5. Switch to database and create tables
    a. 'use genericDBName'
    b. 'create table someTable ({field info});' as needed
6. Create a new user with necessary permissions for the application
    a. 'create user 'someUser'@'localhost' identified by 'somePassword';'
7. Edit new user permissions and flush
    a. 'grant {permissions} to 'someUser'@'localhost';'
    b. 'flush privileges;'
8. This new user is used in the Spring project to access and create information

9.4 Production Cycle

Front End

Updating Website

1. Using a text editor, open the file you wish to update.
    a. To edit the actual page content generally edit the .html file
    b. To edit the page format (ie color, font, size), edit the .css file
    c. To edit the dynamic functions edit the .js file.
2. After implementing desired changes, save the file. Then upload the file to the host server using sftp, WinSCP, or another similar file transfer service.
3. Check the website to see if the changes are functioning properly.

Back End

Versioning

1. Open the pom.xml
2. Version number and default executable name are near the top and can be easily changed
    a. No rebuilding necessary for pom.xml edits

Compiling

1. Save the project, open command line, and navigate to the project folder with ‘mvnw’ in it
2. In this folder use the command ‘mvn clean package’
3. The program runs through tests and compiles
4. Move into the newly created (or previous if this is not the first time) ‘target’ folder. The executable is located there
    a. Currently named ‘PhDTracker-0.0.1-SNAPSHOT.jar - this changes depending on the version number/name in the pom.xml file

Deployment

1. This is done simply with sftp or another file transfer service
2. Connect to the destination and move the file over
3. Run the program with ‘java -jar {filename}’ and wait for running message
4. Pause program with control+z
5. Send it to the background with ‘bg %1’ assuming no other applications are running
6. It’s now safe to exit the environment. The program will continue to run