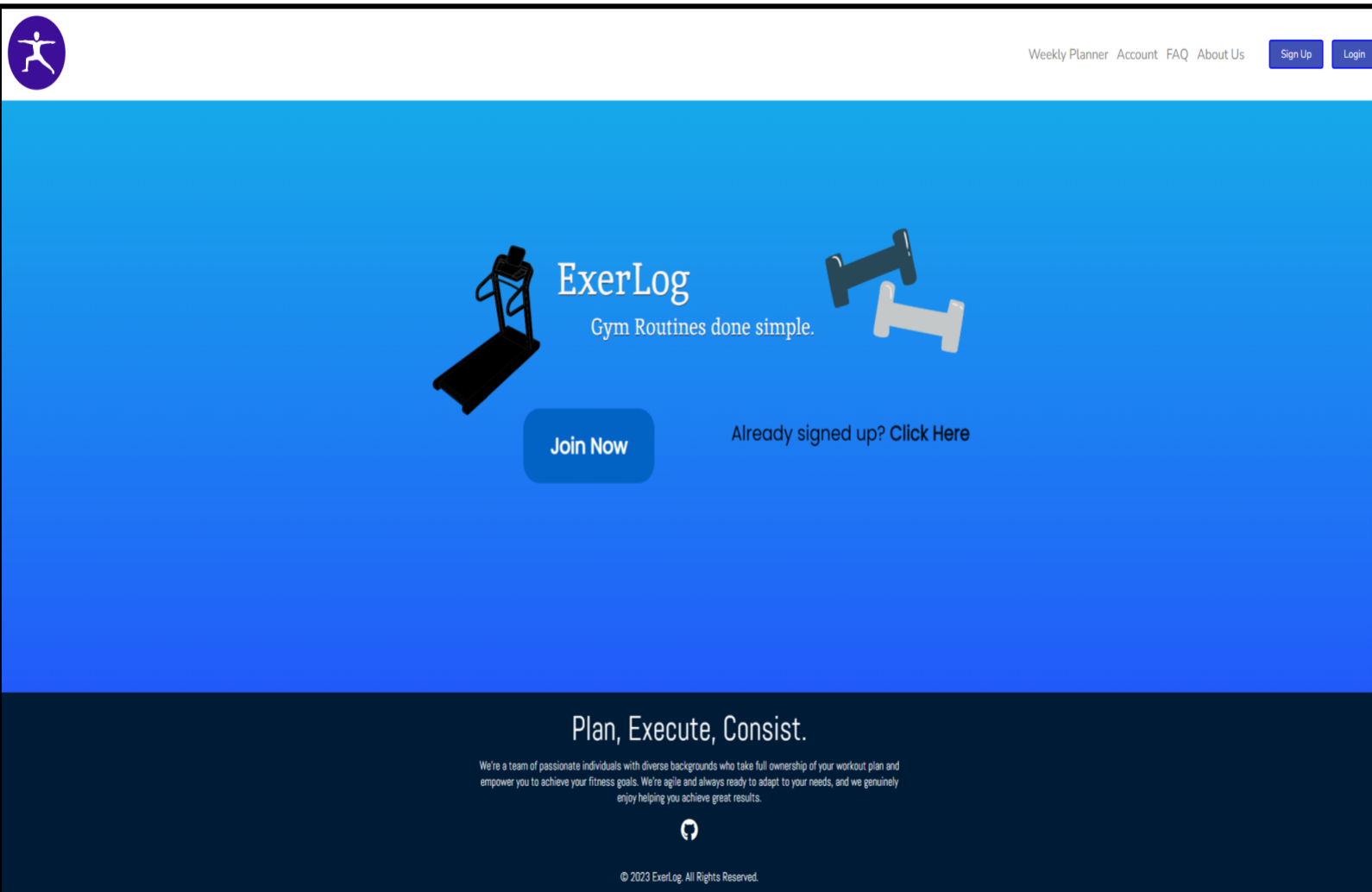


EXERLOG : GYM ROUTINES MADE SIMPLE

CSS 496 Capstone Portfolio



Created By: Jericho Timbol and Brandon Tran

Designed for the Gymgoer

Influenced by Professional Web-Developers and Fitness Experts

05/30/2023

TABLE OF CONTENTS

PART 1 – CAPSTONE REFLECTION OF LEARNING	4
CAPSTONE INTRODUCTION	4
WHAT’S THE BIG IDEA?	4
FITNESS NERD.....	4
PLANNING THE EXECUTION	5
TECHNICAL ROADMAP	5
TWO IS BETTER THAN ONE.....	6
CAPSTONE DELIVERABLES	6
LEARNING ON THE FLY	6
SPLITTING THE WORKLOAD	7
LEARNING DJANGO AND BACK-END CONCEPTS	7
COLLOQUIUM PREPARATION.....	7
PORTFOLIO WORK.....	8
POSTER WORK AND COLLOQUIUM	8
PANEL DISCUSSION PREP	8
PART II – STATUS UPDATES	9
REFLECTION OF LEARNING	9
SCHEDULING.....	9
ANALYSIS	9
CHALLENGES.....	9
WEEKLY REPORTS (6)	10
PART III – MEETINGS	16
REFLECTION OF LEARNING	16
EXPERT AND STAKEHOLDERS	16
INSTRUCTOR AND LIBRARIAN.....	16
FEEDBACK AND PLANS.....	16
MEETING REPORTS.....	17
EXPERT (2).....	17
STAKEHOLDER (2)	20
INSTRUCTORS (3)	22
LIBRARIAN (1).....	25
PART IV – CAREER PREPARATION EVENTS.....	26

REFLECTION OF LEARNING	26
CAREER FAIR AND WORKSHOP	26
EVENT REPORTS.....	27
 PART V – CAPSTONE ARTIFACTS	 30
REFLECTION OF LEARNING	30
INTENTIONAL DOCUMENTATION.....	30
TESTING OUTPUTS.....	30
PLANNING AND PREPRATION	31
POSTGRESSQL DATABASE INITIALIZATION.....	31
DJANGO BACK-END FRAMEWORK INITIALIZATION	33
EXERCISE API FUNCTIONS	36
IMPLEMENTING EXERCISE MODEL-BASED VIEWS GET	36
IMPLEMENTING EXERCISE FILTER.....	38
USERS API FUNCTIONS.....	40
IMPLEMENTING REGISTER/LOGIN/LOGOUT FUNCTIONALITY.....	40
ROUTINE AND WORKOUT API CRUD IMPLEMENTATION	44
IMPLEMENTING ROUTINE AND WORKOUT FUNCTIONALITY	44
TESTING USER ROUTINE BUILDING PROCCES	48
ROUTINE CREATION AND VIEWING.....	48
ADDING WORKOUT TO ROUTINE AND VIEWING ROUTINES	50
SWAGGER API DOCUMENTATION	52
 PART VI – ANNOTATED BIBLIOGRAPHY.....	 53
API AND WEB DEVELOPMENT RESOURCES	53
DATABASE RESOURCES.....	53
PYTHON AND DJANGO/DJANGO REST FRAMEWORK RESOURCES	53
JS, REACT AND FRONT-END RESOURCES	55

PART 1 – CAPSTONE REFLECTION OF LEARNING

CAPSTONE INTRODUCTION

Approaching the end of my undergraduate degree at University of Washington Bothell, I'm required to complete an independent capstone project in order to demonstrate the applications of my learnings. Within my Applied Computing Major with a focus in Data Analytics, the project possibilities are endless. Combining technological skills as well as skills from my secondary study were utilized to research and solve a business problem by developing a project related solution. Based on the idea of a quarter long project I knew I wanted to build some sort of application or software. The project wasn't the only artifact required for this journey. A plethora of different requirements were needed during this process. Meetings had to be held with experts, stakeholders, the librarian and instructors. Towards the end of the quarter, our cohort had to prepare to present our projects in a colloquium as well as hold a panel discussion and review on our projects. There were a lot of moving parts and tasks that needed to get done during this journey. This was the first time I would be working on an ongoing project that lasted for more than a few weeks. This felt similar to what it would be like working on an ongoing development at a workplace. Especially the aspect of documentation and communication with involved parties. Regardless, each piece of my capstone experience has allowed me the opportunity to learn and grow in new ways.

WHAT'S THE BIG IDEA?

The basis and nature of this capstone was seeing a complex idea come into fruition as a solution to a problem. At first this seemed daunting. At the beginning, my mindset went to the idea that I needed to make a ground breaking original project that everyone would be proud of. As I ruminated over the impact and problem it was going to solve, I made a realization that I was maybe putting too much weight on that notion. Looking back, I am glad that me and Brandon followed through with this idea as it has been enjoyable to work through.

FITNESS NERD

As a computer science student, I enjoy coding and building things so I knew I wanted to make some sort of application or program. I opened up my scope and started leaning on things I enjoyed and looking for problems and solutions in those areas. Web applications have always been of interest to me even though I have little to no experience. With my increased proficiency in Python, I knew I could use Django or Flask in order to build a website. If databases were involved my focus in data science would allow me to work with manipulating a

back-end database. On top of this I have also been an avid fitness enthusiast for a large majority of my life. As I was getting my daily workout in, my mind was running laps of its own trying to generate project ideas. That's when it hit me, I love working out and have been an active fitness enthusiast for the greater portion of my life. A web-application involving fitness would definitely be something I would enjoy building. One gripe I have noticed between me and gym goers alike, is finding variety in workouts, or honestly simply just choosing workouts to do. That is when I proposed the idea of a gym routine planner that uses a database to allow people to create custom workouts.

PLANNING THE EXECUTION

Throughout this class we were asked to extensively, plan and document our progress and goals. At the beginning the amount of coursework associated with documentation, reporting and reflecting seemed extremely heavy to me. Not only that but I somewhat wrote it off as busy work. Low and behold, keeping organized with planning, reflecting, and documenting was critical to my success and development during this capstone process. In my mind I often organized my workloads and activities in two parts. One regarding technical and project related endeavors, and the other revolving around staying on top of the other capstone deliverables.

TECHNICAL ROADMAP

With the concept and idea coming into fruition, it was time to work out the logistics. As stated before, my lack of web-development experience played a huge role in planning the development of this project. With a secondary discipline in data analytics and data science, working with Python and databases is a must in the field. That being said, implementing and designing a PostgreSQL database and coding the back-end connections as well as interactions with Python was my main interest. At the beginning I had no idea what it would entail to make a back-end. Nevertheless, how this would work with client-side services. This was when I knew I would need assistance in this project. Luckily one of my classmates, Brandon, was also interested in that idea. With his experience with JavaScript and front-end technologies, we both saw the opportunity to develop this full-stack gym website.

TWO IS BETTER THAN ONE

When talking about my project idea with my peers, I am lucky that one of my classmates/friend Brandon was interested in the initial idea. After some discussion we were bouncing ideas back and forth of what our gym website might be able to do. I am grateful that I was able to find someone interested in the idea. Nevertheless, have experience on the components I am not familiar with (the front-end). This solidifies the idea for me that it is important to not try to do everything yourself and communicating ideas to others may bring opportunity to integrate new or existing skillsets that you would not be able to take on yourself. Me and Brandon were excited to develop this together and having someone making the same development efforts as me motivated me more.

CAPSTONE DELIVERABLES

Looking back on the process, I am glad I paid attention to the artifacts required for numerous deliverables related to the class. For items such as status reports, meeting reports, portfolio work, and poster work, I knew I would need to present screenshots of my progress and document important ideas in my work. Keeping that in mind, I took a plethora of screenshots of my work week to week no matter if I was going to use that artifact or not. The collection of these materials has made my work-flow for these items far easier. I could have planned the completion of necessary items with more priority. Often times, I would put off capstone deliverable plans in order to make progress on the actual project. This shortcoming caused me to be somewhat untimely in submitting extra work. It has also put more pressure towards the end of the quarter by increasing the number of items I need to do before the colloquium. My intended completion dates for these capstone deliverables on my weekly planner and contract did not reflect the challenges I may have faced on a given week or time frame. While everything wasn't line up as envisioned, creating these plans oriented my actions in the correct way as I still had the feeling of items on a schedule that needed to be done.

LEARNING ON THE FLY

One of the biggest takeaways I have cemented in my brain during this process is the idea that you never stop learning and improving. The process of developing a full-stack website was exciting. However, the challenges associated with the diverse range of technological combinations was a factor that me and Brandon underestimated. The main challenges came with our unfamiliarity with the full-stack process. In essence I basically learned everything I implemented in the Django framework from scratch. I didn't account for the time I would spend doing so, however the knowledge and skills I gained was priceless.

SPLITTING THE WORKLOAD

The nature of web-applications often involves a front-end (client), and a back-end(server). Looking back on our development efforts, me and Brandon started getting after our portion of the implementation head-on. Our separate implementations required different knowledge sets and could be coded a million different ways. During the process, me and Brandon were extremely tunnelled in on jumping through the hoops technically in each of our respective assignments. As two developers learning as we go, we were trying to implement our functionalities within our own frame of mind. My teammate and I should have communicated much more than we did, especially in the start. The focus we put in our own code functionality and learnings affected us in a negative way towards the end of the class. Come to find out that there are extensive concepts that need to be used when connection our chosen combination of Django and React.

LEARNING DJANGO AND BACK-END CONCEPTS

The world of API's is a division of programming I have not dabbled in or researched. My knowledge was so limited that the professor asked if I was coding the API with my response being "I don't know, I think so". The journey of going zero to hero in this completely different form of programming has inspired me and motivated me in a plethora of ways. Learning a framework or a language is one thing, but learning an entire back-end structure and configuration is a whole different beast. The concern was I would need both to create both a functional back-end as well as an explainable and digestible one. I didn't know that for the first 3-4 weeks as I was just working on Django and database implementation rather than API routing and connection details. While I could've planned better, the long hours spent looking at all resources back-end and Django has allowed me to attain a completely new skillset I can add to my tool belt.

COLLOQUIUM PREPARATION

Towards the final weeks of the quarter, workflow changes as our class approaches the colloquium and panel discussion events. With the expectation that projects are starting to be finalized and polished, our class shifts focus to key capstone deliverables such as the portfolio and the poster. Along with this, our class is starting to practice and think about how we will be running the panel discussion for our peers.

PORTFOLIO WORK

The Capstone Portfolio and the Capstone Poster are two invaluable items necessary for the completion of this class. As stated before, we have amassed a significant amount of documentation. From meeting reports to status logs, to career preparation events. This portfolio acts as a place in which we not only connect, but organize all of our works and documents into a chronological, format that paints a picture of the achievements and insights we have made during this class. It truly felt that way as I collected and reflected on all my works. I feel a sense of pride and motivation as I look at everything overall. Crafting this document alone has made me realize improvements I could've made in terms of my work efforts in this class. New ideas revolving around planning, anticipation, and even technological approaches are just now being realized. I am excited to grow with the insights I have gained in looking at my work as a whole.

POSTER WORK AND COLLOQUIUM

Creating the capstone poster has revealed to me the importance of understanding what you are doing whilst developing an application or solution for a user. Making this poster requires you to basically give an elevator pitch about what the past 10 weeks has looked like in terms of the progress made as well as the applicability of our solution. Developing this poster has urged me to think about the most important pieces of my abundant work. Not only that but planning on how I am going to communicate those artifacts on the poster to others is pivotal. While posters are mostly visual, crafting and choosing information for this poster has jogged my mind into thinking about the importance of being able to talk about the work you produced.

PANEL DISCUSSION PREP

Presenting your work alongside peers is the final step in this capstone process. Giving a quick presentation (similar to the poster colloquium) is needed in order to show others what you have achieved and solved with your body of work. Understanding the contexts of these works is important. Not only that but supporting each other is key as we are holding the panel between ourselves. Deliverables in this area required us to dig deep to find questions that explain the "so what" aspect of completing this project in its respective medium. As a class community, we eagerly reflect and continuously work to grow as we approach the colloquium.

PART II – STATUS UPDATES

REFLECTION OF LEARNING

SCHEDULING

In our weekly reports we were asked to keep track of and document every specific class item, project item and miscellaneous piece of work we put time into. Hours were recorded on each item as well as weekly totals and cumulative quarter totals. This part of the reports took me the most time. It made me think about the work I was doing in individual pieces rather than a big concept. Doing this helped me gain clarity as I documented my project work in more digestible parts. This in turn allowed me to begin thinking about my planned work more particularly. I would say scheduling was most beneficial in thinking about the week ahead in terms of understanding and estimating the time certain tasks take.

ANALYSIS

One of the biggest benefits I found in constructing these reports was the meta-analysis done on the report. Being able to list and describe key points in the week are pivotal ideas that go beyond completing assignment requirements. In a work setting informing management about your activities is a must. Working with a team and a community means holding yourself accountable to satisfy the responsibilities you have to yourself and the team. By being able to list significant accomplishments and describing the work done that week in my words allows people reading the report to understand how those accomplishments came about. Along with the process of the accomplishments understanding implications regarding that week's activities is vital. By doing so, I felt like I was constantly jogging my memory on how my week shapes out and the quality of my work as I talk about the intention of it.

CHALLENGES

In this facet of my capstone work, I felt like I faced many challenges. Specifically, I had difficulty staying on top of my long and growing list of technical components that needed to be done during the project. While the scheduling aspect helped, in many cases timely estimates were wrong. This was partly due to the fact that I had no scope on how long (or short) it would take for me to learn a completely new concept and integrate it into the project. Adjusting times and pushing previous work week items to the next week for continued progress was a common theme. Readjusting is not a bad thing at all, however with better initial planning, I feel as if I would have a better gauge on project related timing matters when creating my reports.

WEEKLY REPORTS (6)

Week 2**Individual Weekly Report for Jericho Timbol****ExerLog 04/10/2023****Accomplishments**

- ☐ Created PostgreSQL database of Kaggle gym exercise dataset
- ☐ Created a backend environment setup using Django in Python
- ☐ Created a working database connection between database and exercises table
- ☐ Started design document for user interactions and endpoints when interacting with a database.
- ☐ Class meeting Tues/Thurs
- ☐ Completed Instructor Meeting Report #1
- ☐ Completed Research Consult Form and scheduled a meeting with the Librarian

Weekly Activities

Activity / Task / Work	Hours	Status
Class Meeting/ Instructor Meeting #1 4/4	2	Complete
Capstone Contract Work	1	In-Progress
Instructor Meeting Report #1	.5	Complete
Researching Backend Framework and DB to use	2	Complete
PostgreSQL Database creation and server setup	1	Complete
Research PostgreSQL database access and Django basics	2	Complete
Configure and test Django database connection and retrieval	1	Complete
Class Meeting 4/6	2	Complete
Meeting with Teammate to discuss design and connections	1.5	Complete
Weekly Report	1	Complete
Weekly Total	14	
Previous Weekly Cumulative Total (Carry Over)		
Current Cumulative Total	14	

Note Weekly Total and Cumulative Total fields are formulas. You will need to select and "Update Field" to update values.

Plans for Next Week

Activity / Task / Work	Est Hours
Sort and clean Exercises table in SQL	1
Remove and clean Django project and upload to GitHub	2
Code User and Routines model and test connection	3
Code and test interaction endpoint 1- View all exercises	4
Instructor Meeting #2	.5
Instructor Meeting Report #2	.5
Capstone Contract Completion	1

Week 3

Individual Weekly Report for Jericho Timbol

ExerLog

4/15/23

Accomplishments

- ☐ Uploaded all of the API & Backend onto GitHub in a clean manner
- ☐ Sorted the Exercise DB and started an AWS RDS server to host the DB during production
- ☐ 1st Endpoint Interaction Implemented GET all exercises Fully functional
- ☐ 2nd endpoint available for get exercise by ID
- ☐ Gained a vast amount of knowledge about API's and Django Rest Framework
- ☐ Brushed up on version control and learned how to host a DB through AWS
- ☐ Finished and Revising Capstone Contract

Weekly Activities

Activity / Task / Work	Hours	Status
Instructor Meeting 2	.5	Complete
Instructor Meeting Report 2	.5	Complete
Capstone Contract/Plan	2	In-Progress
Exercise DB Clean	.5	Complete
Learn and start AWS RDS Database Hosting	2	Complete
API Research on Django Rest Framework	3	In-Progress
Django API implementation Branch Started	2	In-Progress
Upload Project to Github	1	Complete
Django Routing and URL Research	2	In-Progress
Setup Basic URL Overview	1.5	Complete
Weekly Total	15	
Previous Weekly Cumulative Total (Carry Over)	14	
Current Cumulative Total	29	

Plans for Next Week

Activity / Task / Work	Est Hours
Copy my local DB setup to AWS DB	1
Continue ViewSet API research for further interaction implementations	4
Implement DB models and views for user creation and storage.	3
Implement API endpoint #3 for Exercise based on name, equipment, and or body part	2
Meet with Brandon about creating an optimized Routine DB table for retrieval on the calendar front-end	1
Create draft for user creating a routine	3

Week 5**Individual Weekly Report for Jericho Timbol****ExerLog****5/1/23****Accomplishment**

- ☐ Interviewed Stakeholder and Expert Plus Report
- ☐ Show and Tell 1
- ☐ Front-End, Cloud DB Server, and Back-End Connected
- ☐ Register, Login, Logout, Logoutall Authentication API implemented
- ☐ Created FilterSet for Exercises view: name, muscle_group, equipment

Weekly Activities

Activity / Task / Work	Hours	Status
Stakeholder and Expert Questionnaire Creation	1	Complete
Expert Meeting and Stakeholder Meeting	2	Complete
Expert/Stakeholder Report	1	Complete
Show and Tell 1 Meeting and Report	2.5	Complete
Transfer Local DB changes to AWS	1	Complete
Connect Back-end to front-end and DB	2	Complete
Finish login register and logout API	3	Complete
Django Routing and URL Path Cleanup	1	Complete
Routine, and Workout creation model created	1	Complete
Routine CRUD API implementation	2	In-Progress
React Js Research	1	In-Progress
Weekly Status Report	1	Complete
Weekly Total	18.5	
Previous Weekly Cumulative Total (Carry Over)	44	
Current Cumulative Total	62.5	

Plans for Next Week

Activity / Task / Work	Est Hours
Research best viewset and API for routine creation	2
Change filter for muscle group to choose filters and improve filter user experience perspective	3
Make sure relations in PostgreSQL local db are correct	1
Meet with Brandon to help implement page for exercise view and functioning front-end login logout	5
Once routine api function is optimized, update AWS DB with model changes and test	2
Career Prep 2 In Class + Report	2.5
Weekly Status report 6	1

Week 7**Individual Weekly Report for Jericho Timbol****ExerLog****5/15/23****Accomplishment**

- ☐ Expert Meeting 1 & Report
- ☐ Show and Tell 2
- ☐ Core Functionalities Started/Finished
 - Create Routine Create/Add Workout as a Token based authenticated User
- ☐ Swagger API View Implemented
- ☐ Heavy PostGresSQL Work
 - Cleaning DB table id's, make sure it was functioning on AWS cloud database.
- ☐ Heavy GitHub updating, fixing and merging

Weekly Activities

Activity / Task / Work	Hours	Status
Expert Meeting	1	Complete
Expert Report	.5	Complete
Show and Tell 2 Meeting and Report	2	Complete
Created and Implemented User Routine model CRUD	4	Complete
Created and Implemented User Workout model CRUD	4	Complete
Implement Nested Workout/Exercise Serializer	6	Complete
Swagger API Implementation	2	Complete
Routine, and Workout creation model created	1	Complete
Weekly Status Report	1	Complete
GitHub Update and Clean	1	Complete
Start Capstone Portfolio	2	In-Progress
Weekly Total	25.5	
Previous Weekly Cumulative Total (Carry Over)	62.5	
Current Cumulative Total	88	

Plans for Next Week

Activity / Task / Work	Est Hours
Research Axios as a way to connect backend to react	3
Capstone Portfolio Class 5/16	2
Meeting With Instructor 3 + Report	1
Create Poster Draft	3
Work On Capstone Portfolio	4
Implement backend API connection for Login	3
Meet with Brandon to pair code	1
	2
Weekly Total est	17

Week 8**Individual Weekly Report for Jericho Timbol****ExerLog****5/22/23****Accomplishment**

- ☐ Substantial Portfolio Work
- ☐ Poster Draft
- ☐ Front End Work and Merge
- ☐ Stakeholder Meeting 2

Weekly Activities

Activity / Task / Work	Hours	Status
Stakeholder Meeting 2	1.5	Complete
Stakeholder Report	.5	Complete
Instructor Meeting 3	.5	Complete
Meeting Report	.5	Complete
Capstone Portfolio Work	2	In-Progress
Draft Poster PowerPoint	2	In-Progress
Weekly Status Report	1	Complete
Research React API Consumption	4	Complete
Implement Login Register API connection	2	In-Progress
Create List of To-Do Items to finish project	1	Complete
Weekly Total	15	
Previous Weekly Cumulative Total (Carry Over)	88	
Current Cumulative Total	103	

Plans for Next Week

Activity / Task / Work	Est Hours
Finalize Poster	3
Finalize Capstone Portfolio Draft 2 (All Items done)	3
Poster Class and Peer Review	3
Colloquium Panel Questions	.5
Finish all Implementation/Close to All	2
Career Event 3	1
Stakeholder meeting 3	1
Expert meeting 2	1
Meet with Brandon to pair code clean and deploy website	1
Weekly Total est	15.5

Week 9**Individual Weekly Report for Jericho Timbol****ExerLog****5/28/23****Accomplishment**

- ☐ Extensive Portfolio Work
- ☐ Poster Draft 2 Almost Done
- ☐ Front End Work Register Done and Login almost done
- ☐ Extensive React Research
- ☐ Colloquium Panel Questions and Preparation

Weekly Activities

Activity / Task / Work	Hours	Status
Class Poster Review and Panel Work	2	Complete
Poster Peer Review	.5	Complete
Completed Portfolio Reflection for all Sections	5	Complete
Completed Portfolio Annotated Bib	1	Complete
Worked on Draft 2 for Poster	2	In-Progress
Weekly Status Report 9	1	Complete
Finished Registration frontend	1	Complete
Working on Login frontend	6	In-Progress
Career Event 3 Report and Handshake Exercise	1	Complete
Weekly Total	20	
Previous Weekly Cumulative Total (Carry Over)	103	
Current Cumulative Total	123	

Plans for Next Week

Activity / Task / Work	Est Hours
Polish Poster for Second Submission	1
Capstone Portfolio Draft 2 Completion and Editing	2
Tuesday Portfolio Class	2
Portfolio Peer Reviews	.5
Go Over Panel Questions	1
Prepare for Questionnaire role in Panel	1
Another Career Event	1
Expert Meeting 2/Report	1.5
Finish as much Front-End functionality as possible	7
Weekly Total est	17

PART III – MEETINGS

REFLECTION OF LEARNING

EXPERT AND STAKEHOLDERS

Meeting with experts and stakeholders allowed me to interact directly with individuals directly related or affected by my project. Looking back, my expert Archie was immensely helpful in developing a strong foundational API. His immense knowledge in web development solutions allowed him to structure the back-end in a digestible and clean way.

Conducting Interviews with stakeholders provided me with an outlet to abstract the ideas of the project. At the end of the day, we are looking to develop an IT solution that solves a business problem. Meeting with those who are directly affected by the solution of ExerLog provided a great deal of suggestion in improving our project. By asking those who would benefit from the app their thoughts from a non-technical but user standpoint allowed for vast improvement while developing the backend. Both forms of interactions served to provide varying perspectives in which different solutions or changes can be discussed.

INSTRUCTOR AND LIBRARIAN

Meetings with the instructor and librarian were both incredible resources that turned my questions into actionable ideas or welcomed critiques. Instructor meetings in particular helped me with concepts and work management ideas related to both capstone work as well as project endeavors. The versatile exchange of conversation in these areas helped me stay on track and trust the process.

Meeting with the UWB librarian was beneficial in framing our research question and solution set in a better space. Similar to stakeholder meetings, Je pointed us to bountiful technical coding journals such as O'Reilly in order to develop a better web app for fitness routines.

FEEDBACK AND PLANS

Feedback I received from these meetings were a huge factor in orienting my actions as well as get past road-blocks. Action items and planning for next week were huge factors in motivating me to stay on top of the project. Meetings with others were insightful and offered the suggestions of additional things. Every week there seemed to be another great idea or functionality I wanted to implement. Keeping feedback in mind is vital when talking to others about your project. At a point I was tunneled in on the implementation I had in mind, keeping the perspective of other parties in regard helped me improve pieces I may have not caught.

MEETING REPORTS

A collection of meeting reports filed after meeting with numerous individuals regarding the ExerLog project throughout the course of this quarter. At the end of the quarter, I also was required to conduct an extensive final code review in order to ensure the use of best technical practices based on the project.

EXPERT (3)

Meeting Report 1

Meeting Type (BOLD the meeting phrase)	Instructor / Expert / Stakeholder / Code Review / Librarian
Meeting Date	Tues. May 9, 2023
Attendees	Jericho Timbol – Archie Timbol
Project Title	ExerLog
Purpose	WebDevelopment Expert Consultation

Summary of Meeting

During this initial meeting, I met with my father who was a former web developer. He has been in IT his entire career and now works at Microsoft as a Senior Cybersecurity Engineer. With his extensive knowledge in computer science and specifically web-development, I planned on showing him my API progress so far. During the meeting, I showed him the Django API Root View and took him through all the functions used in my intended project (Artifact 1). After I showed him the ERD setup of my database since he has experience using SQL in the past. All the keys checked out on his part with certain tweaks on routine and workout tables. After showing him the ERD chart and the database structure, I showed him my intended form of authentication. I was showing him my way of passing tokens through postman; however, he showed me a better tool to display and document my API swagger. Lastly, I showed him my token-based authentication system. With security being his strong suit, he gave me ideas to use Knox with Django for authentication instead of the default method.

Action Items

List specific action items that you need to take because of the meeting:

- ☐ Change DRF Root API view to Swagger API Documentation View
- ☐ Switch from Django basic authentication to Knox for multi-token authentication. (Multiple devices).
- ☐ Set Permissions on views
- ☐ On Swagger learn how to pass API key Token
- ☐ Add foreign Key from workout to user and foreign key from routine to user

Reflection

What I brought to the meeting was an overall skeletal structure of my basic implementation. As I am finishing up and fixing bugs on my basic implementation routes, I am confused as to where I would concentrate my efforts. This outside perspective from a professional web developer standpoint helped me considerably in ways to tweak and improve the back-end slightly while garnering significant functionality improvements for the program as a whole. Archie's insight on token authentication was extremely helpful in the way that it helped me understand how headers and tokens are saved and refreshed in a program. It was funny because he was just picking things out as he saw when I was scrolling through code that helped

Meeting Report 2

Meeting Type (BOLD the meeting phrase)	Instructor / Expert / Stakeholder / Code Review / Librarian
Meeting Date	5/30/2023
Attendees	Jericho Timbol, Archie Timbol
Project Title	ExerLog
Purpose	Front-End Help, and Finalization Strategies.

Summary of Meeting

During the meeting, I showed Archie the progress I have made in terms of documenting the API as well as the connections I have made to the front-end. He saw my struggles in the super specific problem related to token storage on the front end. I gave him a demo of the Swagger API showing that all authentication components returned translatable data in JSON format. With the backend functioning properly, it was a problem with the token and user transition to the front-end. At this point he asked me if I implemented the backend with authentication at first. With my answer being no, he asked me why I was trying to retrieve token authentication without first being able to serve the basic functionality. He showed me the connections I had made in my database between the user and the routines and showed me ways to correlate them without authentication.

Action Items

- ☐ Rework authentication later.
 - ☐ Remove the front-end permission barriers
 - ☐ Allow user creation in a basic manner
- ☐ Implement routine creation and exercise functionality first
- ☐ Create a test page in order to display/print data that is retrieved
- ☐ Add some back-end API code to retrieve and create routines and workouts based on the

Reflection

In particular I learned that I need to take breathe and focus on important things rather than trying to get user login and authentication working at a high degree. Ideas such as local storage, logs, mapping, and history could be simplified or removed completely. I learned an immense amount of the basic steps of fetching API's. In addition to the database connections and query sets being handled in the backend, the program could be "hardened" later as my dad put in technical terms.

Comments / Issues / Notes / Other

None at the moment

CODE REVIEW

Code Review Meeting Report

Meeting Type (BOLD the meeting phrase)	Instructor / Expert / Stakeholder / Code Review / Librarian
Meeting Date	06/03/2023
Attendees	Jericho Timbol, Archie Timbol
Project Title	ExerLog
Purpose	Backend Code Review

Summary of Meeting

During this meeting, me and Archie went through an extensive code review of all the back-end code I've written. In order to assess the quality of code he, he cross-checked my strategies with common coding and web development best-practices he has developed over the years. We started with going through the entire back-end API process in Swagger. I showed him the process of registering, logging in, creating a routine, creating workouts for routines and viewing all aspects of those routines and workouts. Subsequently I showed him what logging out looks like and how it will not allow you to access authenticated materials.

After showing him the functionality of the back end, and portions of the front-end authentication and the troubles I was going through, I took him through a detailed walk through of how the back-end functions. I walked him through files regarding routing, endpoints/views, serializers, database models and all the pieces of code I wrote. After walking through it with him and explaining how the implementation works, he took some time to observe the code closer and asked me various questions regarding why I did what I did along with making suggestions in certain areas.

Action Items

List specific action items that you need to take because of the meeting:

- ☐ Security
 - In the settings file there was an exposed secret key that I must remove or relocate
 - ☐ Never keep secrets in code
 - ☐ Best practice: keep secret keys in secure variables or store them in vault
 - Incorrect error messages should not indicate a matching user in the system. I need to change error response on authentication validate function to "Incorrect Credentials".
- ☐ Organization
 - Make all python quotations uniform. Either single quotes or double quotes.
 - Group registration of models and declarations grouped together in files
 - ☐ Need to change admin registration page for models

STAKEHOLDER (2)

Stakeholder Report

Type (BOLD the meeting phrase)	Interviews / Surveys
Interaction Date(s)	04/26/23
Attendees	Triscia Alavarez
Project Title	ExerLog
Purpose	Stakeholder Meeting 1 Initial Website Idea Presentation and Functionality Planning

Summary of Interaction

During the interaction me and Triscia, an avid gym enthusiast, went over the design and functionality of my capstone project. I gave her an overview and described to her the intention and purpose of the uses this routine building. I also prepared questions regarding her current processes when creating gym routines. We discussed many aspects of what she would look for if she were to use software to curate her routines. I gained quite a bit of insight on key functionalities as well as improvements.

Action Items

List specific action items that you need to take because of the interaction:

- ☐ Connect my backend implementation to a template to provide a better demonstration of functionality to next stakeholders.
- ☐ Finish out my 8 interactions ASAP to continue addressing stakeholder feedback for additional implementation
- ☐ Expand user attributes to fitness profile
- ☐ Multi-token authentication for cross-device usage

Reflection

I learned that Triscia currently does not have a consistent way to add, track and create workouts. The functionality and concept I presented was definitely of interest to her. She recommended instead of a simple user login and profile that it include stats that you could possibly update such as weight or goals. Not only that but she was mentioning an aspect of calorie counting and maybe checking if you did the routine for accountability. This stakeholder interview with her had me excited. It made me realize that I can implement these base functionalities pretty quickly and work on a multitude of other functionalities that have been recommended during discussion.

Comments / Issues / Notes / Other

N/a

Stakeholder Report 2

Type (BOLD the meeting phrase)	Interviews / Surveys
Interaction Date(s)	05/16/23
Attendees	Triscia Alavarez
Project Title	ExerLog
Purpose	Stakeholder Meeting 2 Design/Implementation Survey and API Demo

Summary of Interaction

During the interaction me and Triscia, started out by going through the Google Forms I provided her. This form noted ideas about what the stakeholder would want to see as well as their thoughts on the structure of the project at the current point of time. That being said I showed her the figma design as of right now. Additionally, I ran her through the demo of creating a routine and workout as a user from an API standpoint.

Action Items

- ☐ Front end work with Brandon
- ☐ Logo name considerations
- ☐ Visual Goal for progress weight increase overtime
 - Chart displaying goals and such of weight increase
- ☐ Add weight category to workout exercise.
- ☐ Find more captivating fitness pictures
- ☐ Present a connected design and functionality

Reflection

After conducting this google form interview, I learned that the API functionality and documentation could use some improvement. That being said moving away from the DRF root and presenting the back-end functionality with the token-based authentication present. Front end reconsiderations should be made, but that is currently not the priority. We must focus on completing implementations and get the MVP (minimum viable product) before addressing minute concerns. A connected front-end and back-end is the goal to demo.

Comments / Issues / Notes / Other

N/a

INSTRUCTORS (3)

Meeting Report

Meeting Type (BOLD the meeting phrase)	Instructor / Expert / Stakeholder / Code Review / Librarian
Meeting Date	04/04/2023
Attendees	Jericho, Brandon, TA, Professor Anderson
Project Title	ExerLog
Purpose	Initial Contract Meeting

Summary of Meeting

- ☐ Metrics and evaluation
- ☐ Personal goals
- ☐ Schedule Same column schedule different
- ☐ In the architecture look to color code
- ☐ Experts should be involved in the certain aspect we are working on
- ☐ Survey of metric stakeholders and involvement

Action Items

- ☐ Start specifying and honing in on specific experts in the field
 - Looking to meet with possible health and fitness professionals
 - Developers with experience in full stack development
- ☐ Identify weekly checkpoints and goals for a more specific tracking on the progress of the project. This is however dynamic and fluid in nature.

Reflection

Workload wise we devised a better plan in splitting up work and making progress in the many components of our full-stack project. Not only this but the meeting with Professor Anderson allowed us to think about stakeholders and possible experts in the area of web dev and health/fitness professionals. This meeting has also prompted increased communication between me and my teammate as we support each other in the creation of the project. Thinking about survey questions and how we would build our application based on the metric it is judged on.

Comments / Issues / Notes / Other

Just to make sure we are up to date on our work each time we meet and our scope is still within the length of the quarter.

Meeting Report

	Sonal Yadav

Summary of Meeting

- -
- -
- -
- -

Action Items

- -
 - Review/feedback especially at API's
-

Reflection**Comments / Issues / Notes / Other**

Meeting Report

	Sonal Yadav

Summary of Meeting

- -
- -
- -
- -

Action Items

- -
 - Doesn't have to be perfect but should show intentions of actio
-
-
-
-
-

LIBRARIAN (1)

Meeting Report

Meeting Type (BOLD the meeting phrase)	Instructor / Expert / Stakeholder / Code Review / Librarian
Meeting Date	04/17/2023
Attendees	Jericho, Brandon, Librarian Je Salvador
Project Title	ExerLog
Purpose	Second Instructor Meeting

Summary of Meeting

- ☐ Discussed the context of our project and functionality
 - We gave Je the background of our code implementation to help generate a more focused search on key pieces of research and sources we can use for our project.
- ☐ Looked through recommended sources
 - Looked through sources relating to technological implementation (Oreily)
 - Looked through many other sources related to fitness research. Especially those relating to exercise goals rather than nutrition.

Action Items

- ☐ As I continue to implement the rest of the API endpoints, I plan to use the O'Reilly source provided by the librarian to assist me with any technological endeavors.
- ☐ Create a document taking notes on research aggregated for the best practices when it comes to routine creation.
- ☐ Possibly plan suggestion flow charts for routine template functionality in the back end.

Reflection

Je really helped me and Brandon in terms of talking about how we can implement research into our website. The amazing .gov sources we looked over together were extremely reliable and helpful. Notably the coding resource Je provided us will be pivotal in development efforts as both me and Brandon are working with technologies and tools, we don't have much experience in.

Comments / Issues / Notes / Other

None at the moment

PART IV – CAREER PREPARATION EVENTS

REFLECTION OF LEARNING

Throughout this process, we were asked to participate and reflect on numerous careers advancing opportunities that are offered at UWB. These range from job and internship fairs with real companies, to updating and increasing the value of the provided Handshake profile and portal. I found enormous value in completing these activities.

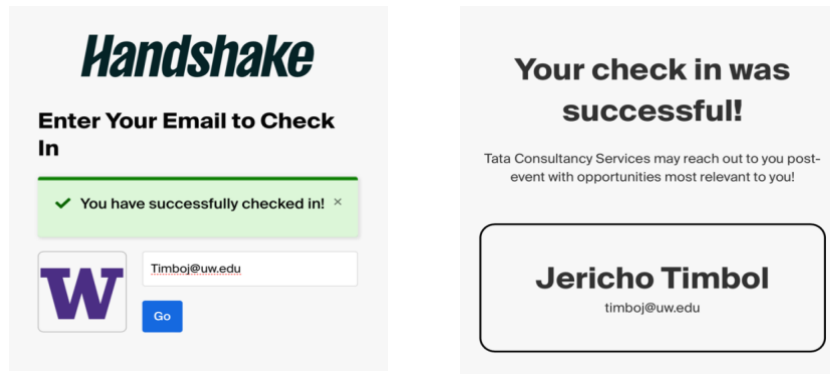
CAREER FAIR AND WORKSHOP

The career fair and workshop were two of the most impactful preparation activities I took part in. This was due to the fact that they were in person in nature and involved real results. The first event I attended was the job and internship fair on April 18th. At this event I made connections with many professionals and had amazing discussions about opportunities and what these companies were working towards. It gave me a sense of professionalism as well as a taste of reality as I start searching for my first job. I made connections with many great people and took action based on this event. Notably, I spoke with representatives from IHME, CrowdStrike, and Tata Consultancy. With these companies I made connections with the recruiters and even submitted my application for consideration. I immediately went home and made the networking connections with the recruiters and reaching out for emails. I applied for a few jobs but have yet to hear back from any. Regardless I recognized how important it was to connect with people. This was also apparent during our workshop in class about contracting work, recruitment as well as LinkedIn Profile Strategies. Applying for job listings with no rhyme or reason online is known to be a futile effort. Jessica from Brook Source held an insightful workshop on strategies to get past initial technological barriers as well as look more appealing to any type of recruiter. This workshop gave me a keener sense on how the job application process looks on the other end as well as give me a better idea of what kind of type and obligations certain jobs hold as I browse online.

EVENT REPORTS

Career Event 1 Reflection: Spring Job & Internship Fair: 4/18/23

Proof of Attendance

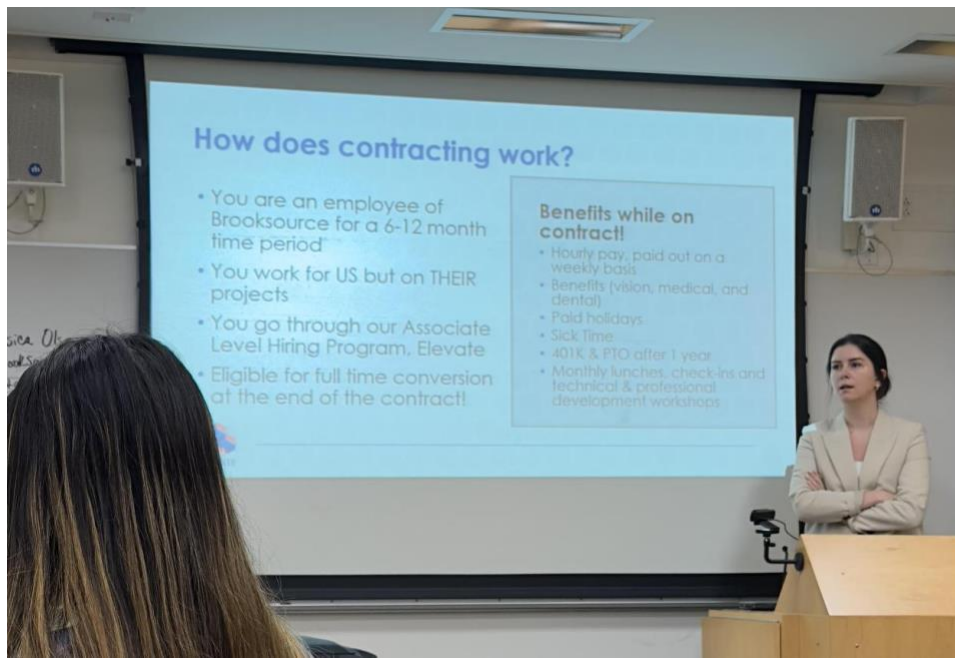


Event Description and My Interactions

UW's network and provide an opportunity to get ahead of the curve when it comes to networking

Career Event 2 Reflection: In-Class Workshop: 5/04/23

Proof of Attendance



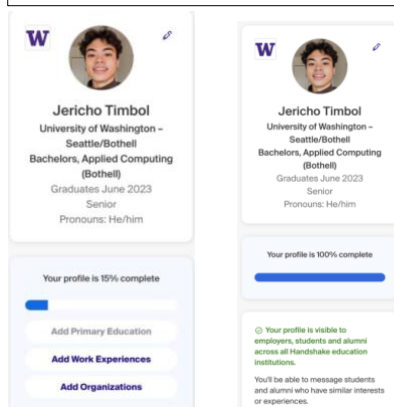
Event Description and My Interactions

Career Preparation 3 Reflection: Handshake Profile Update

Jericho Timbol
05/27/23

Proof of Work

Before and After Profile Completion



About Me Section Update

My Journey

Hello ! As I wrap up my last year at UWB I am eager to start a career in the field of Data Analytics. In our growing world of data and internet of things, I have always been curious about the amount of information, resources and pieces of knowledge you can find digitally. I look to use my curiosity to help businesses generate data driven solutions.

Organization Update

Organizations & Extracurriculars



Camp Providence Leader

Volunteering
Jul 2017 - Aug 2017 (2 months)

Camp Prov is an integrated day camp for children with special needs and their siblings, 2-12 years of age. I was assigned a group of kids to assist with their camp experience for 4 days!

Project Update

Projects



ExerLog

Mar 2023 - Jun 2023

A Django/PostgreSQL/React website allowing authenticated users to create personal workout routines from an expansive database of exercises.

Technologies:
Django/Python
React/JS
PostMan
PostgreSQL
AWS RDS



UtilizationRequirementKPIAnalysis

An assessment on how differing Utilization Requirements between two experimental groups affects business outcomes. Visualized and analyzed through various key performance indicators within the dataset. Performed additional analysis on how project startup coach types affect these KPI's between the groups.

Technologies:

Work Experience Update

Work & Volunteer Experience



Azul Restaurant & Lounge

Bartender
Jan 2019 - Present (4 years, 5 months)

I have been working for the local and wildly successful Blue Food Groups Inc. since I was 16. Throughout the years, I have amassed a phenomenal work ethic at this busy high-volume restaurant. I am able to take on various roles of professional responsibility in a fast paced environment. Everyday I must constantly communicate with my team, management, and customers to solve a wide context of business problems. This has honed me into an efficient verbal communicator, and has also given me an eye for generating positive solutions.



Providence Health & Services

Lab Assistant
Jul 2017 - Aug 2017 (2 months)

I assisted the hospital lab faculty with the distribution of specimens and their normal laboratory processes. In addition to that, this opportunity allowed me to participate in "Camp Prov".

PART V – CAPSTONE ARTIFACTS

REFLECTION OF LEARNING

This course has engrained a concept that I have been familiar with throughout my education at University of Washington Bothell. The importance of documentation has become apparent whilst working on a big project. Changes in project direction are bound to happen in a client service model. Well documented technical endeavors will help you communicate progress and ideas with teammates, stakeholders, and those with non-technical backgrounds.

INTENTIONAL DOCUMENTATION

Choosing the right artifacts was a process I didn't think would be cognitively taxing. However, choosing artifacts was harder than expected. With a large body of work, you can show milestones and progress in an infinite variety of ways. Knowing the important parts of the code and technology during the project required me to be intentional with my selections. In turn this triggered a domino effect of learning. I realized I had to understand what the most important pieces of back-end development consisted of before I could choose significant landmarks in the implementation. This process was valuable when done in tandem with my version control documentation. I have not had a huge amount of experience with GitHub projects spanning more than a month. Collecting artifacts while making my commits helped me visualize the tasks I was completing.

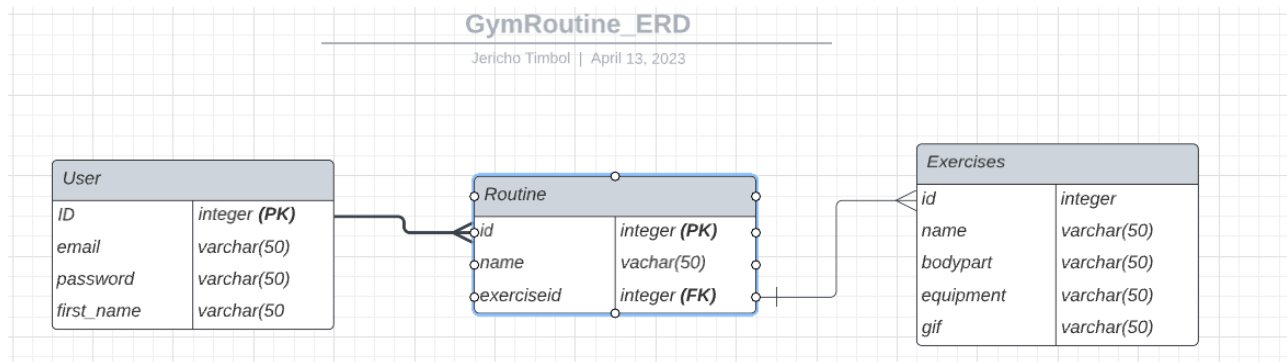
TESTING OUTPUTS

Gathering and collecting artifacts for my capstone work had a huge impact on how I checked items off my plan and to-do list. Because of the solutions produced by code, outputs of the back-end implementation had to be demonstrated. This immensely benefited the structure of my back-end code as I was constantly testing outputs in order to demonstrate expected and clean outcomes for each API call. Documenting the expected outputs also allowed me to consider various inputs and the outcomes these wrong inputs have. The integrity of my code was greatly improved in this regard.

PLANNING AND PREPRATION


POSTGRESSQL DATABASE INITIALIZATION

1. Initial ERD For PostgreSQL DB



Serving as an exercise planner with a user routine-model, I decided to use a relational database in order to serve data to the clients. With my familiarity in PostgreSQL and database design, I knew the first step was to create an entity relation diagram. An ERD is a flowchart like diagram that shows how objects in a system relate to each other. This was essential to planning the coding aspect of the project.

2. Kaggle Data Set Source



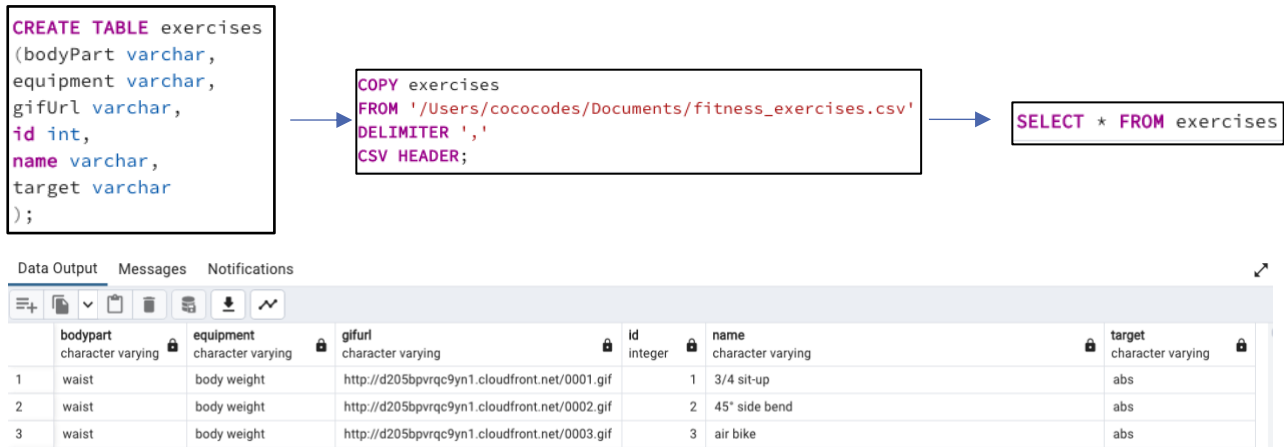
Fitness Exercises

This dataset contains 1300 exercises with exercise data and animations.

bodyPart	equipment	gifUrl	id	name	target
Bodypart that is targeted by the exercise	Equipment needed for the exercise	URL of an animated GIF describing the exercise	ID	Name of the exercise	Targeted muscle

With the database scheme plotted out, it was now time to integrate a collection of workouts for users to choose from. I found an amazing database with a significant selection of workouts in addition to attributes for each exercise. With the scope of the project involving the selection of exercises, this was an essential part of the database. The extra attributes are vital to allowing users to filter, browse and even learn how to perform different exercises

3. SQL Creating and Storing dataset into PostgreSQL



To create and manage my database I chose to utilize the tool PgAdmin, a PostgreSQL graphical user interface. The commands above show the SQL CRUD statements I used in order to transfer the large Kaggle CSV into my local PostgreSQL DB.

4. AWS RDS Setup

Databases

Group resources

Modify

Actions

Restore from S3

Create database

Filter by databases

<

1

>

<div><div></div></div> DB identifier	<div><div></div></div> Role	<div><div></div></div> Engine	<div><div></div></div> Region & AZ	<div><div></div></div> Size	<div><div></div></div> Status	<div><div></div></div> Actions
<div><div></div>database-1</div>	Instance	PostgreSQL	us-west-2b	db.t3.micro	<div><div></div>Modifying</div>	<div><div></div>2 Actions</div>

With my local database setup with the initial exercise table, I began to consider early how this would work in a production setting. I needed a way to store my database so me and my partner can both work with the current rendition. Additionally, we needed users to be able to access the database as well. After some research, I decided to upload my database to Amazon Web Service's Relational Database Services. This allows full management of the database running on Amazon's server. This shows the creation and running instance of the database on the cloud server.

DJANGO BACK-END FRAMEWORK INITIALIZATION

1. Back-End Functionality Planning

#	Interaction	Expected Result
1	User must register an account or login	A user must login or create an account with correct parameters in order to see other navbar functions on the website.
2	User opens Exercise catalog page	A database query aggregates all of the exercises with their name, body part, equipment requirement and gif link.
3	User filters Exercise list for a specific name, body part, and or equipment	A database query aggregates all of the exercises with their filtered attributes and the UI presents this in a table to the user.
4	User opens routine page	A database query retrieves all the created routines for the user
5	User can create routines	A user can use a form to create a new routine with a name. (Optional) User must select a template for the type of routine they want to create.
7	User can add exercises to the routine	An add exercise button will allow a user to filter through exercises and choose which ones to add to the routine which is then committed.
8	User can logout	A user logs out removing all options from sidebar and securely exiting the environment.

Implementing the back-end requires certain functionalities and endpoints that returns a response to client requests. With a REST API implementation in mind, deciding the endpoints I would serve was essential in coding the back-end and the connection between both technology stacks.

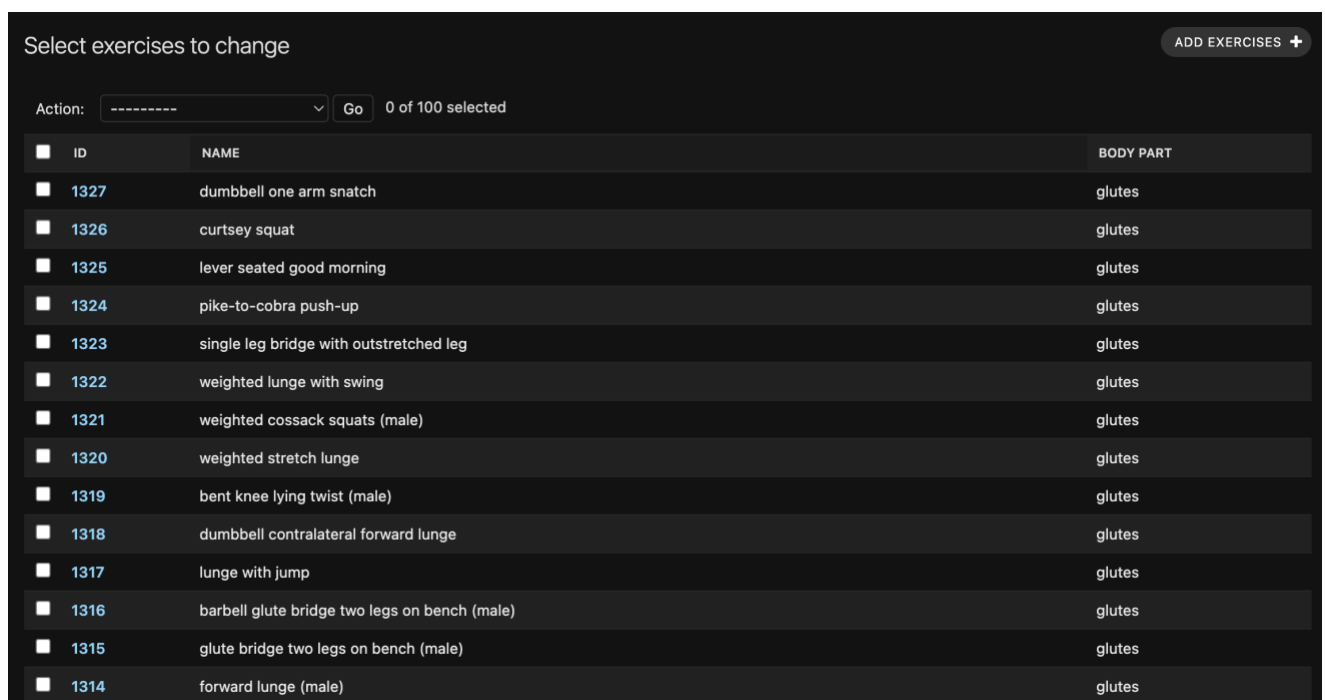
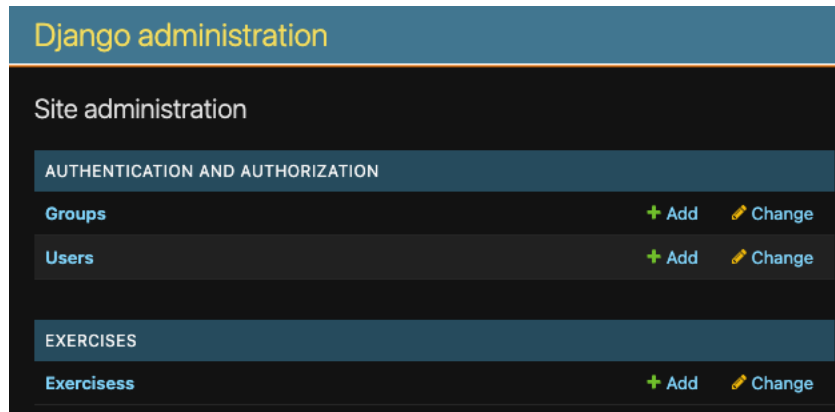
2. Database Model Creation and Connection

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'gymroutinedb',
        'USER': 'cococodes',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

```
1  from django.db import models
2
3  # Create your models here.
4  class Exercises(models.Model):
5      bodypart = models.CharField(db_column = 'bodypart',blank=True, null=True)
6      equipment = models.CharField(db_column = 'equipment',blank=True, null=True)
7      gifurl = models.CharField(db_column = 'gifurl',blank=True, null=True)
8      eid = models.IntegerField(db_column = 'id',blank=True, null=False, primary_key= True)
9      name = models.CharField(db_column = 'name',blank=True, null=True)
10     target = models.CharField(db_column = 'target',blank=True, null=True)
11
12     class Meta:
13         managed = False
14         db_table = 'exercises'
15     def __str__(self):
16         return self.name
```

Django requires the creation of objects called “models” that serves as access points for your database. Similar to tables in the database, they contain all the attributes reflecting any changes back to the database. Configurations must be made in the settings in order to access the PostgreSQL database.

3. Database Retrieval Test



Retrieving the database was done through the amazingly convenient admin page included in the Django framework. This allows you to register your database and models to an accessible administer page. Here you can view things such as users and the models you have migrated. The above artifact depicts the successful retrieval and view of the exercises residing within the database.

EXERCISE API FUNCTIONS

IMPLEMENTING EXERCISE MODEL-BASED VIEWS GET

1. Creating Exercise Model Serializer

```
class ExercisesSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Exercises  
        fields = ('id', 'name', 'muscle_group', 'equipment', 'gif_url')
```

In order to view data from the back-end, Django requires the implementation of serializers. These serializers allow data sent “through the wire” to be converted into formats readable to front-end interactions and client-side operations and viewability.

2. Implementing Django Rest Framework Model View Set

```
class ExercisesViewSet(viewsets.ModelViewSet):  
    queryset = Exercises.objects.all()  
    serializer_class = ExercisesSerializer  
  
    def retrieve(self, request, pk=None):  
        exercise = get_object_or_404(self.queryset, pk=pk)  
        serializer = ExercisesSerializer(exercise)  
        return Response(serializer.data)
```

Using Django’s ViewSets Class, I was able to severely simplify the API creation process. This class allows you to implement create, update, read, and delete (CRUD) operations for related views. Using the ModelViewSet, this meant grouping views and operations for the exercise model. I made some changes to the class including overriding the “GET” method. A new retrieve function was written in order to serve status responses and serialize the data.

3. Setting HTTP Routes

```
#The REST Framework router will make sure our requests end up at the right resource dynamically.
# If we add or delete items from the database, the URLs will update to match
router = routers.DefaultRouter()
router.register(r'exercises', views.ExercisesViewSet)

# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
]
```

The artifact above depicts the process of routing a Django view to a specified URL. In order to add the URL of a ModelViewSet I decided to use a router in order to route all the related endpoint operations the class provided. This includes the main retrieve functionality I was testing for.

4. Testing Exercise GET Function

Exercises List

OPTIONS

GET

GET /exercises/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
[
  {
    "id": 1,
    "name": "3/4 sit-up",
    "body_part": "waist",
    "equipment": "body weight",
    "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/0001.gif"
  },
  {
    "id": 2,
    "name": "45° side bend",
    "body_part": "waist",
    "equipment": "body weight",
    "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/0002.gif"
  },
  {
    "id": 3,
```

By using Django's default router, we are provided a way to test http functions and URL's. The artifact above shows the result of using the /exercises/ get route in order to retrieve the list of all the exercises with their key datapoints.

5. Testing Exercise ID Based GET

Api Root / Exercises List / Exercises Instance

Exercises Instance

DELETE OPTIONS GET

GET /exercises/1/

```

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 1,
  "name": "3/4 sit-up",
  "body_part": "waist",
  "equipment": "body weight",
  "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/0001.gif"
}

```

Raw data HTML form

Id	<input type="text" value="1"/>
Name	<input type="text" value="3/4 sit-up"/>
Body part	<input type="text" value="waist"/>
Equipment	<input type="text" value="body weight"/>
Gif url	<input type="text" value="http://d205bpvrqc9yn1.cloudfront.net/0001.gif"/>

PUT

Selecting exercises for a routine will require the extraction and viewing of a single exercise. That being said, the artifact shows the result of implementing a detail based retrieve through the URL path including the ID of an exercise.

IMPLEMENTING EXERCISE FILTER

1. Filter Set Implementation Class

```

class ExerciseFilter(filters.FilterSet):
    name = filters.CharFilter(lookup_expr = 'istartswith')
    # body_part = filters.CharFilter(lookup_expr = 'istartswith')
    muscle_group = filters.CharFilter(lookup_expr = 'istartswith')
    equipment = filters.CharFilter(lookup_expr = 'istartswith')

```

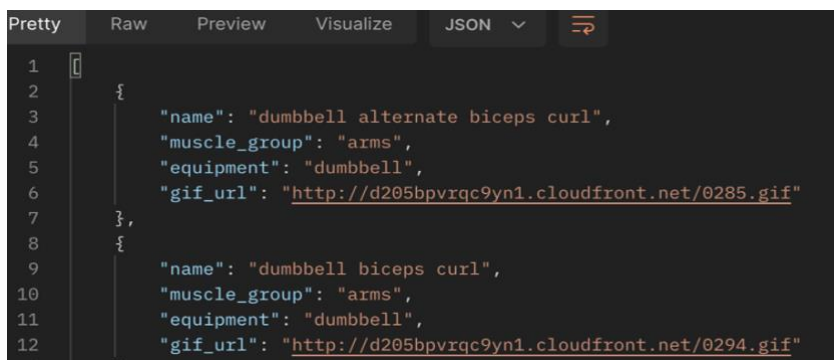
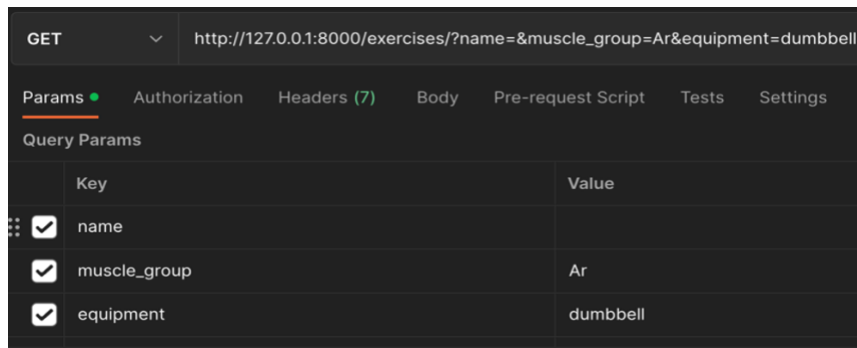
```

filterset_class = ExerciseFilter

```

Filtering the exercises addressed one of the key functionalities that allows users to view and discover exercises in a more dynamic manner. By utilizing the muscle_group, equipment, and name attributes of the model, I implemented a filterable search view. This allows users to get a dynamic filter that follows “starts with” ordering.

2. Testing Filter Set in Post



Using Postman, I tested many variations of the filter set. Variations I tested included a combination but not limited to partial field entries, empty field entries, multiple field entries, and non-existent field entries which functioned as expected during Postman testing

USERS API FUNCTIONS

IMPLEMENTING REGISTER/LOGIN/LOGOUT FUNCTIONALITY

1. User Actions Endpoints

```
#Non-routed url Paths
path('register/', RegisterAPI.as_view(), name = 'register'),
path('login/', LoginAPI.as_view(), name='login'),
path('logout/', Knox_views.LogoutView.as_view(), name='logout'),
path('logoutall/', Knox_views.LogoutAllView.as_view(), name='logoutall'),
```

The above shows the URL routes coded to connect to the numerous authorizations related to user functions. Logout and Logout All are already bundled within the Knox module.

2. User, Login and Logout Serializers

```
5
6 # User Serializer
7 class UserSerializer(serializers.ModelSerializer):
8     class Meta:
9         model = User
10        fields = ('id', 'username', 'email')
11
12 class LoginUserSerializer(serializers.Serializer):
13     username = serializers.CharField()
14     password = serializers.CharField()
15
16     def validate(self, data):
17         user = authenticate(**data)
18         if user and user.is_active:
19             return user
20         raise serializers.ValidationError("Invalid Details.")
21 # Register Serializer
22 class RegisterSerializer(serializers.ModelSerializer):
23     class Meta:
24         model = User
25         fields = ('id', 'username', 'email', 'password')
26         extra_kwargs = {'password': {'write_only': True}}
27
28     def create(self, validated_data):
29         user = User.objects.create_user(validated_data['username'], validated_data['email'], validated_data['password'])
30
31         return user
```

The code above depicts my implementation of the serializers required to return data regarding the user. User serializers returned user information. The Login serializer provided a means to serialize username and password inputs as well as validating the information as an existing and active user. The Register serializer makes sure to provide a means to create an account with the

necessary fields. Setting the password to write-only added security. The custom create writable serializer allows the creation of a user object with the validated input data in the form.

3. Login and Register API View

```
class LoginAPI(generics.GenericAPIView):
    serializer_class = LoginUserSerializer

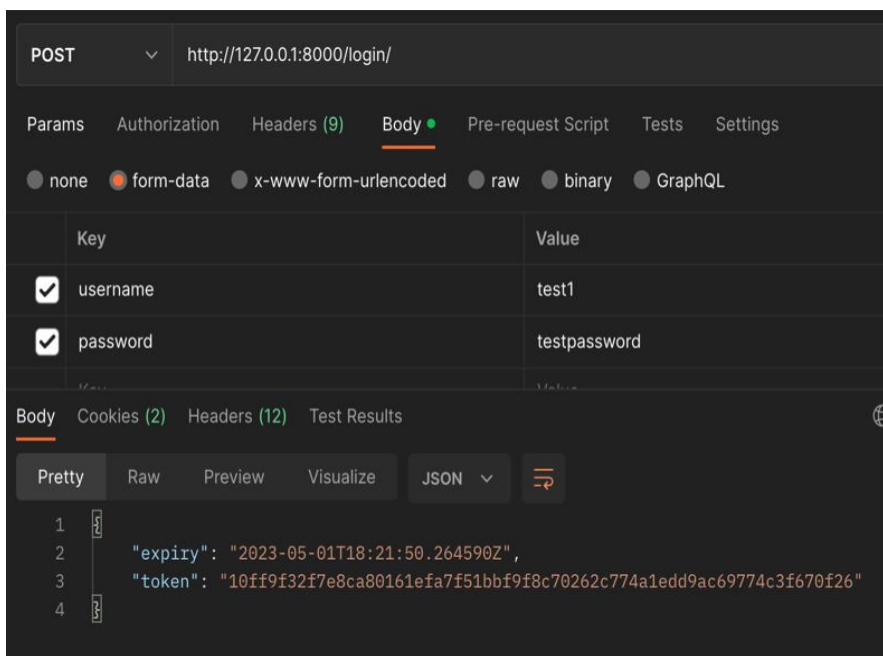
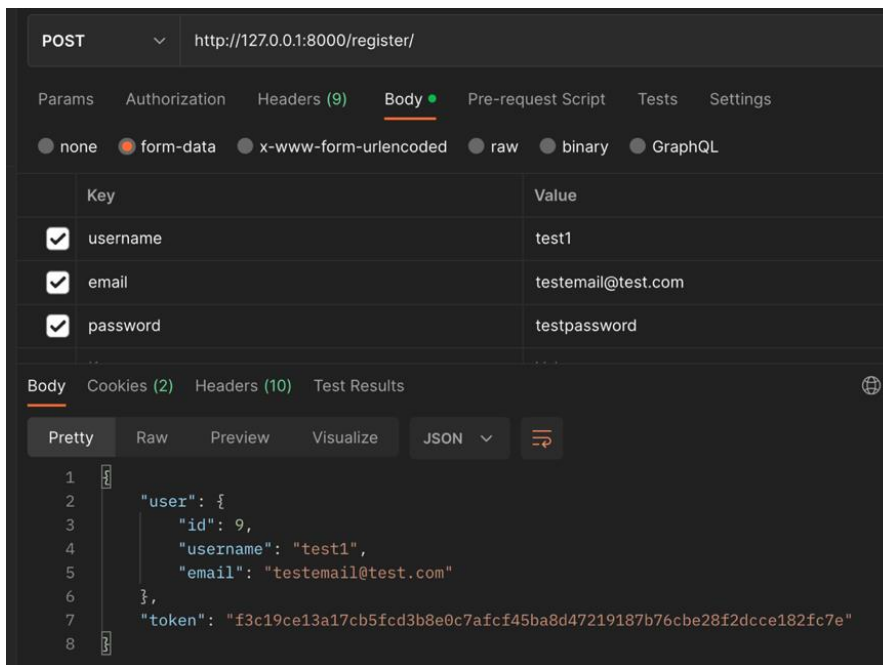
    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.validated_data
        return Response({
            "user": UserSerializer(user, context=self.get_serializer_context()).data,
            "token": AuthToken.objects.create(user)[1]
        })

# Register API
class RegisterAPI(generics.GenericAPIView):
    serializer_class = RegisterSerializer
    permission_classes = [permissions.AllowAny]

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        user = serializer.save()
        return Response({
            "user": UserSerializer(user, context=self.get_serializer_context()).data,
            "token": AuthToken.objects.create(user)[1]
        })
```

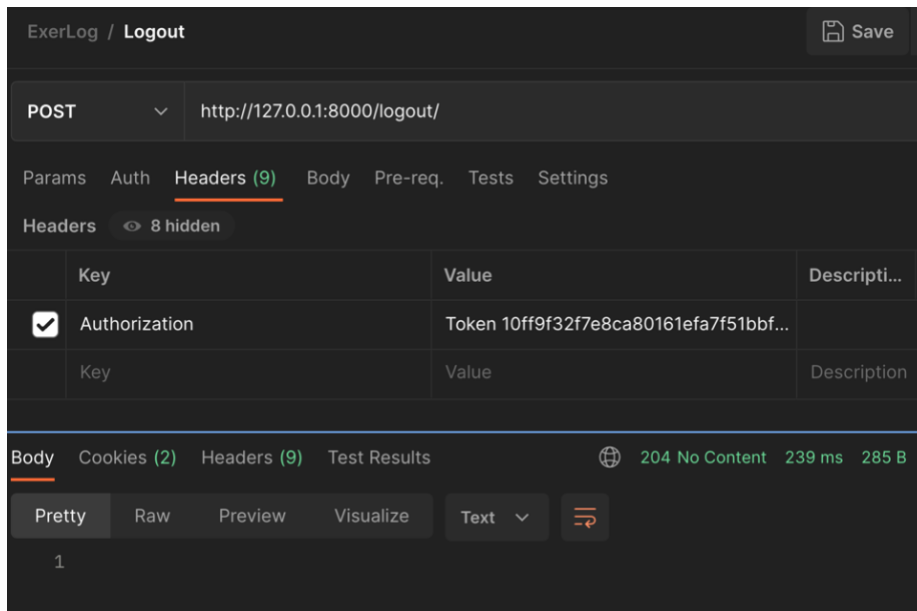
For User related views, I needed to create registration functionality as well as login views. With a token-based authorization scheme in mind, tokens were necessary for both views. LoginAPI takes validated user data and returns the user information as well as the token assigned during login. Similar to LoginAPI, RegisterAPI provides the same response however the differing Register serializer returns a new user object.

4. Register and Login API Postman Test



To test the register and login functionalities, I used Postman in order to pass the necessary form data. The first artifact depicts the registration of a user with a user, email and password. The response was successful, returning the new user object as well as an authorization token. The login response on the bottom provides a new token with an expiry and the token.

5. Built in Logout API Postman Test

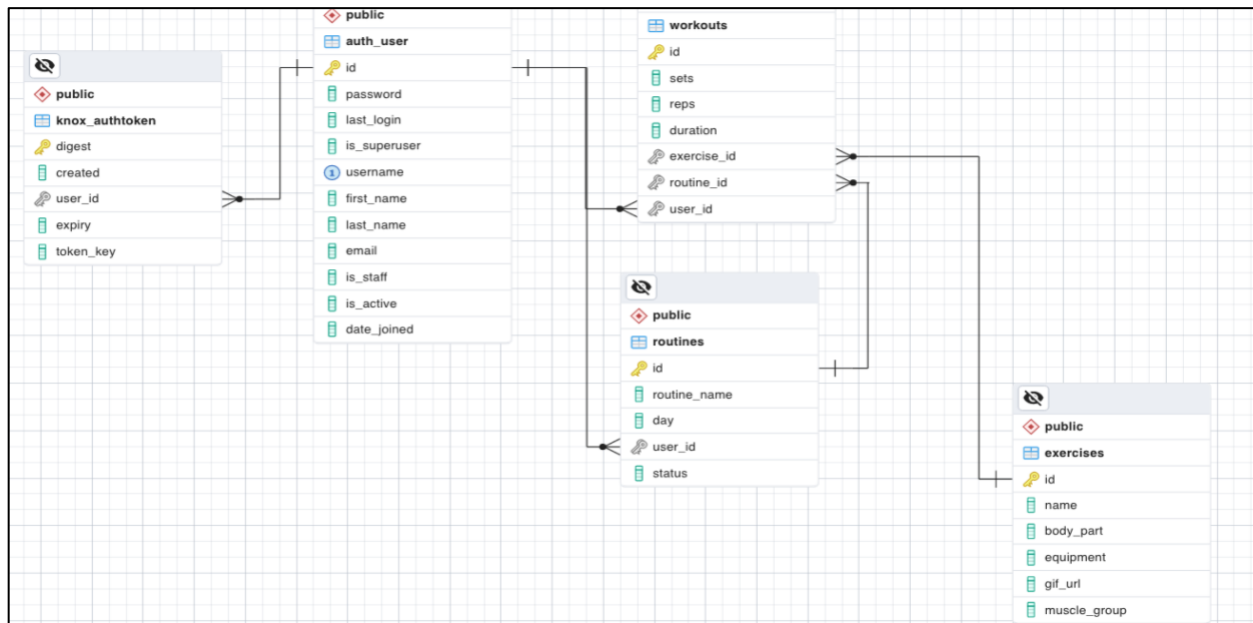


The bundled logout functionality in the Knox authentication system includes a logout function. This function is a post function that takes the token of the user and clears it from the database. A successful call returns a 204 HTTP status.

ROUTINE AND WORKOUT API CRUD IMPLEMENTATION

IMPLEMENTING ROUTINE AND WORKOUT FUNCTIONALITY

1. Adding Workouts and Routines Table to PostgreSQL DB



In order to store user data on the routines and workouts they create, there will need to be an update to the initial ERD database schema. By planning out how we are storing data for users and their workouts, it will make coding the database model in Django easier. This is due to the fact that you need to instruct the relations between the models in the code. In the final system above users can access workouts and routines they have created or added. A workout is made by pulling an item from the exercise list out of the database and adding additional information on how the workout will be performed. The next artifact depicts the table layout of the database after creating the routine and workout tables.

Tables (14)
> auth_group
> auth_group_permissions
> auth_permission
> auth_user
> auth_user_groups
> auth_user_user_permissions
> django_admin_log
> django_content_type
> django_migrations
> django_session
> exercises
> knox_authtoken
> routines
> workouts

2. Creating Routine and Workout Database Models

```
class Routine(models.Model):
    DAY_CHOICES = [
        ('sunday', "Sunday"),
        ('monday', "Monday"),
        ('tuesday', "Tuesday"),
        ('wednesday', "Wednesday"),
        ('thursday', "Thursday"),
        ('friday', "Friday"),
        ('saturday', "Saturday"),
    ]

    STATUSES = (
        ('Started', 'Started'),
        ('Finished', 'Finished'),
    )

    routine_name = models.CharField(blank=False, null = False)
    day = models.CharField( choices = DAY_CHOICES)
    status = models.CharField(max_length=8, choices=STATUSES, default=STATUSES[0][0])
    user = models.ForeignKey(User, related_name="routines", on_delete=models.CASCADE, blank=False, null=False)

    class Meta:
        managed = True
        db_table = 'routines'

    def __str__(self):
        return '{} {}'.format(self.routine_name, self.day, self.status,self.user)

class Workout(models.Model):
    exercise = models.ForeignKey(Exercises,
                                on_delete=models.CASCADE,related_name= 'exercise')
    sets = models.IntegerField(blank=True, null=True)
    reps = models.IntegerField(blank=True, null=True)
    duration = models.IntegerField(blank = True, null = True)
    routine = models.ForeignKey(Routine, on_delete=models.CASCADE,related_name = "routine_workouts")
    user = models.ForeignKey(User, related_name="workout", on_delete=models.CASCADE, blank=False, null=False)

    class Meta:
        db_table = 'workouts'
        managed = True

    def __str__(self):
        return '{} {}'.format(self.exercise.name,self.sets,self.reps,self.duration,self.routine)
```

This segment of code details the implementation of the Routine and Workout models in the back-end. These are two core components of the system. Each model contains a link to the user in order to identify ownership in the database and whilst serving the data. Additionally, the workouts created by the user are always required to be linked to a user's routine as well as an exercise in the database. For the routine model, days are choice bound due to the structure of adding routines to a certain day of the week when planning your routines.

3. Nested Workout and Routine Serializer

```
class WorkoutSerializer(serializers.ModelSerializer):
    name = serializers.CharField(source = 'exercise.name', read_only = True)
    body_part = serializers.CharField(source = 'exercise.body_part', read_only = True)
    gif_url = serializers.CharField(source = 'exercise.gif_url', read_only = True)
    user = serializers.CharField(source = 'user.username', read_only = True,)

    class Meta:
        model = Workout
        fields = ['exercise', 'user', 'name', 'body_part', 'sets', 'reps', 'duration', 'gif_url', 'routine']

    def create(self, validated_data):
        request_user = self.context['request'].user
        if validated_data:
            instance = Workout.objects.create(user=request_user, **validated_data)

        return instance

class RoutineSerializer(serializers.ModelSerializer):
    routine_workouts = WorkoutSerializer(many = True, read_only = True)
    user = serializers.CharField(source = 'user.username', read_only = True,)

    class Meta:
        model = Routine
        fields = ['id', 'user', 'routine_name', 'day', 'routine_workouts']

    def create(self, validated_data):
        request_user = self.context['request'].user
        if validated_data:
            instance = Routine.objects.create(user=request_user, **validated_data)

        return instance
```

This snippet of code details the solution I used in order to display front-end compatible details about the routines/workouts the user has created. When viewing the routines, a user must also be able to see the details of each workout they have added to that given routine. By including a field provided by the workout serializer, the routine serializer will show all the routine details through their one-to-many connection.

4. Implementing CRUD Function Views for Routine and Workout Model

```
class WorkoutViewSet(viewsets.ModelViewSet):
    permission_classes = [IsAuthenticated,]
    serializer_class = WorkoutSerializer
    queryset = Workout.objects.all()
    def get_queryset(self, *args, **kwargs):
        return Workout.objects.all().filter(user=self.request.user.id)

class RoutineViewSet(viewsets.ModelViewSet):
    permission_classes = [IsAuthenticated,]
    serializer_class = RoutineSerializer
    queryset = Routine.objects.all()

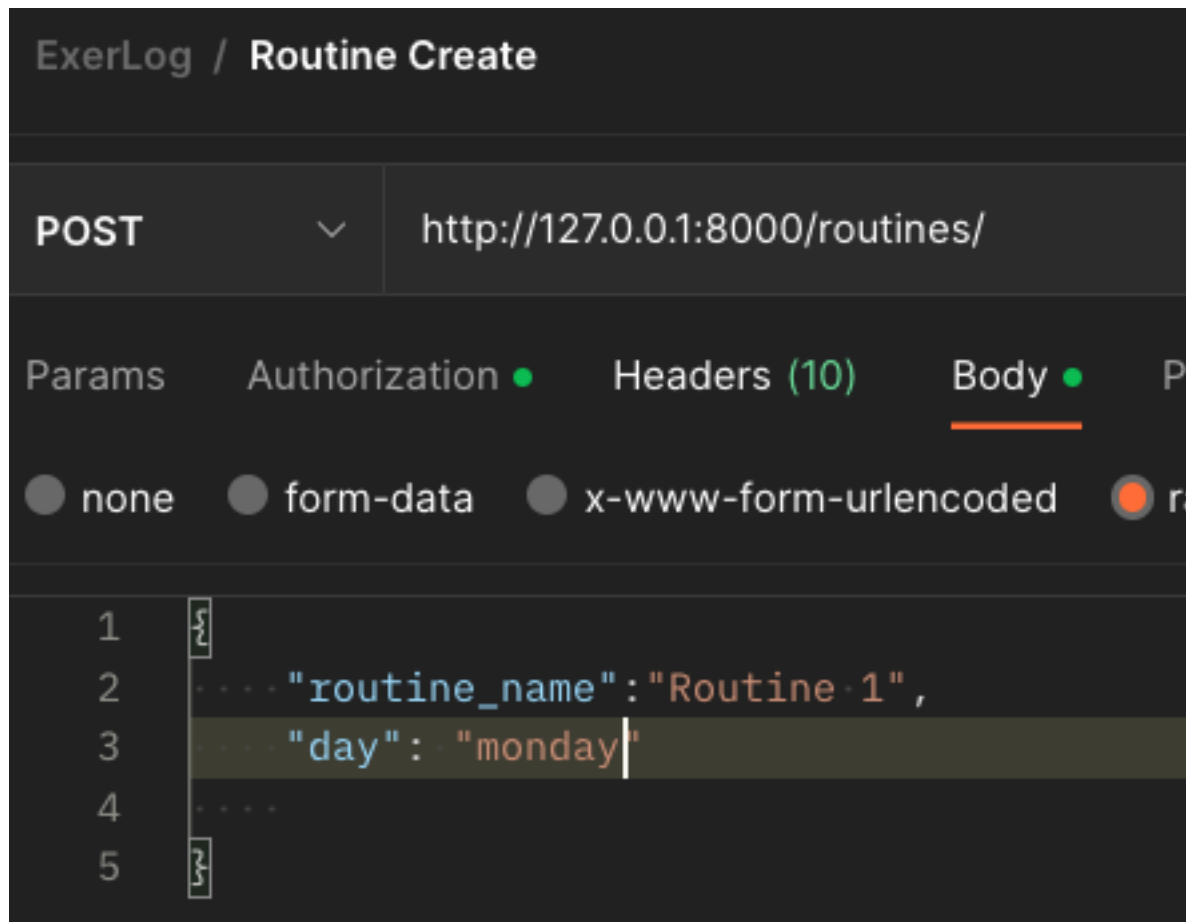
    def get_queryset(self, *args, **kwargs):
        return Routine.objects.all().filter(user=self.request.user.id)
```

In order to serve endpoints to create, view, delete, and update their routines, I had to implement a Model ViewSet for each class. These ViewSets are restricted to authenticated users by way of login. The view function is overwritten in order to return the view of only the routines and workouts belonging to the user.

TESTING USER ROUTINE BUILDING PROCCSS

ROUTINE CREATION AND VIEWING

1. User Routine POST PostMan Input



After implementing the operations, I moved to the testing phase of the process. First, I tested the post return address and inputted the form data in the JSON format above.

2. User Routine GET Postman Output

```
{
  "id": 58,
  "user": "test1",
  "routine_name": "Routine 1",
  "day": "monday",
  "routine_workouts": []
}
```

Testing the get routine yielded this result. Correctly showing the routines a user has created, along with the associated user, name, day, and the routine workouts. This is empty since the user has not added any workouts to their routine yet.

ADDING WORKOUT TO ROUTINE AND VIEWING ROUTINES

1. POST workout to Routine

POST /workouts/

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "exercise": 1216,
  "name": "lever lying two-one leg curl",
  "body_part": "hamstrings",
  "sets": 5,
  "reps": 15,
  "duration": null,
  "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/3195.gif",
  "routine": 55
}
```

Raw data

HTML form

Exercise	<input type="text" value="lever lying two-one leg curl hamstrings"/>
Sets	<input type="text" value="5"/>
Reps	<input type="text" value="15"/>
Duration	<input type="text"/>
Routine	<input type="text" value="Routine 1 monday"/>

POST

The integrated DRF testing functionality was utilized in order to test this step. The artifact shows the results of filling out the form and posting the workout. On the form fields, a routine is required in order to add it into a certain routine that a user has created. The exercise field is required and only allows the selections of exercises on the exercise list from the database. After posting the form, we can see the exercise ID, name, body part, equipment and many more fields are displayed. This is the successful creation of a workout.

2. Get Routines after adding workouts

```
{
  "id": 55,
  "user": "test1",
  "routine_name": "Routine 1",
  "day": "monday",
  "routine_workouts": [
    {
      "exercise": 1002,
      "name": "dumbbell one arm seated bicep curl on exercise ball",
      "body_part": "biceps",
      "sets": 5,
      "reps": 15,
      "duration": null,
      "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/1668.gif",
      "routine": 55
    },
    {
      "exercise": 1216,
      "name": "lever lying two-one leg curl",
      "body_part": "hamstrings",
      "sets": 5,
      "reps": 15,
      "duration": null,
      "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/3195.gif",
      "routine": 55
    }
  ]
},
{
  "id": 56,
  "user": "test1",
  "routine_name": "Routine 2",
  "day": "tuesday",
  "routine_workouts": [
    {
      "exercise": 970,
      "name": "cable overhead curl",
      "body_part": "biceps",
      "sets": 2,
      "reps": 12,
      "duration": 12,
      "gif_url": "http://d205bpvrqc9yn1.cloudfront.net/1636.gif",
      "routine": 56
    }
  ]
}
```

In theory, the user would create multiple routines and have various workouts that are different for each routine. That being said, after implementing and testing the creation of routines and the posting of workouts, it was time to test for a real scenario. After creating many routines and adding various workouts, it was vital to have a successful and comprehensible views of all your workouts. This is the result of the get implementation when a user retrieves all their routines. Detail view of each routine is available as well by using the routine id.

SWAGGER API DOCUMENTATION

The screenshot displays the Swagger API Documentation for ExerLog, version 1.0.0. The interface includes a header with the API title and version, a 'Schemes' dropdown set to 'HTTP', and buttons for 'Django Login' and 'Authorize'. A 'Filter by tag' input field is located above the main list of endpoints. The endpoints are organized into several categories, each with a header and a list of methods:

- exercises**:
 - GET /exercises/ (exercises_list)
 - POST /exercises/ (exercises_create)
 - GET /exercises/{id}/ (exercises_read)
 - PUT /exercises/{id}/ (exercises_update)
 - PATCH /exercises/{id}/ (exercises_partial_update)
 - DELETE /exercises/{id}/ (exercises_delete)
- login**:
 - POST /login/ (login_create)
- logout**:
 - POST /logout/ (logout_create)
- logoutall**:
 - POST /logoutall/ (logoutall_create)
- register**:
 - POST /register/ (register_create)
- routines**:
 - GET /routines/ (routines_list)
 - POST /routines/ (routines_create)
 - GET /routines/{id}/ (routines_read)
 - PUT /routines/{id}/ (routines_update)
 - PATCH /routines/{id}/ (routines_partial_update)
 - DELETE /routines/{id}/ (routines_delete)
- workouts**:
 - GET /workouts/ (workouts_list)
 - POST /workouts/ (workouts_create)
 - GET /workouts/{id}/ (workouts_read)
 - PUT /workouts/{id}/ (workouts_update)

After completing the testing and implantation of the intended functionality, I needed a way to document the functionality along the way. I served a Django Rest Framework Root view, however I found it wasn't enough to supplement the separate development of the front-end. I began looking for ways to document the first API I had created. After expert recommendation, I implemented code in order to show an interactive Swagger API. This was pivotal in documenting and showing the resulting functionality of the back-end portion of ExerLog

PART VI – ANNOTATED BIBLIOGRAPHY

API AND WEB DEVELOPMENT RESOURCES

REST API Crash Course + Python API Demo

Curry, Caleb. “Rest Api Crash Course - Introduction + Full Python Api Tutorial.” *YouTube*, 19 Nov. 2020, www.youtube.com/watch?v=qbLc5a9jdXo&t=599s.

Caleb Curry is a widely recognized YouTube resource when it comes to coding. With limited knowledge in web-development, I needed a foundation and structure before I started coding. That being said, this video taught me the fundamentals of how API functions as well as states within a program. Endpoints were clearly described as well as an implementation demo in the language I would be using for ExerLog.

DATABASE RESOURCES

ExerLog utilized a relational database that is accessible by users and developers alike. Resources regarding copying formats from Django and PostgreSQL servers were used. Database related actions were researched in order to remind myself of past teachings and best practices. SQL resources of various formats were used in order to increase database management capacities during this project.

Kaggle Exercise Dataset

Cantagallo, Edoardo. “Fitness Exercises Dataset.” *Kaggle*, 20 Mar. 2022, www.kaggle.com/datasets/edoardoba/fitness-exercises-with-animations.

I utilized a pre-existing database from Kaggle as the structure of my exercise collection. Users are able to filter exercises based on the pre-defined columns found in the dataset as well as see a viewable gif link from this dataset.

PYTHON AND DJANGO/DJANGO REST FRAMEWORK RESOURCES

During the course of the project, I used Python’s full-stack framework Django. I aimed for a REST API for simplicity. That being said, developing a back-end is something I’ve never done. While I am familiar with Python, I am not familiar with building API’s nevertheless Django. This aspect of my project was the area I spent a heavy amount of research. The documentation page

from both Django and DRF served as a cornerstone of my processes. Various video tutorials around YouTube were also watched for context specific actions or tutorials.

Django REST Framework Documentation

Christie, Tom. "Django Rest Framework." *Home - Django REST Framework*, www.django-rest-framework.org/. Accessed 30 May 2023.

This was one of my most used resource due to the nature of my back-end implementation strategy. Django's REST Framework is a flexible toolkit allowing for the development of a well-structured Web API. This was appealing due to Django's compatibility with relational databases such as PostgreSQL. I used this documentation almost every step of the way in order to learn the framework. Especially when it came time to implement the API functionality as well as the endpoints. The root view allows for easier development on projects for front-end and backend which was attractive about this tool kit as well.

Django Documentation

Django. "Django." *Django Project*, docs.djangoproject.com/en/4.2/. Accessed 30 May 2023.

Python's full stack framework Django was utilized framework for the back-end. As an all batteries included framework, you could essentially write your entire website within the environment of this project. That being said it is a loaded framework that required a lot of reading and learning. I used the regular Django documentation mainly for setting up data interactions and models. Additionally, ideas regarding queries, instances as well as integrated authentication packages were researched.

Django and PostgreSQL Connection

KenBroTech. "How to Connect Django Project to Already Existing Database." *YouTube*, 6 Apr. 2021, www.youtube.com/watch?v=CkYvpKZyEqI.

This YouTube video was a tutorial I leaned on in order to get a functioning database connection between the database server and the Django server. With no familiarity in database integration in Python, I needed a solution that would allow me to specifically integrate my PostgreSQL database. Not only that but this resource allowed me to create a Django Model based on the initial existing ExerLog database containing the exercises table.

Django ViewSets

Very Academy. “Django Rest Framework Series - Viewsets and Routers with React Front-End Example - Part-4.” *YouTube*, 16 Sept. 2020, www.youtube.com/watch?v=dCbFOZurCQk&t=926s.

Along with the DRF documentation, I used this video in order to help me get a grasp on the capability of ViewSets. More importantly this tutorial series takes place in an environment utilizing both Django and React which was a bonus. I gained a lot of knowledge about overwriting integrated Model ViewSet functions as well as routing them to a URL path.

Django Authentication

Sayyed, H. “Django Rest Framework Tutorial – Register Login Logout Api - Django Tutorial.” *StudyGyaan*, 9 May 2021, studygyaan.com/django/django-rest-framework-tutorial-register-login-logout.

This guide/tutorial was the corner stone of creating the authentication system for the project. Using one of Django’s packages called Knox, I followed this guide in order to implement a Token Based authentication system. This multi-token authentication packages provided back-end API implementation steps for functions such as user registration, login, logout, and logoutall. It still uses the Users model provided by Django in their provided authorization scheme. This allows me to forego the creation of a user table in the database since Django and Knox handle that already.

JS, REACT AND FRONT-END RESOURCES

With no initial knowledge in JavaScript, CSS, HTML and React, I relied on resources in order to implement some functionality in the client-side. Documentation was key as always in the library and languages I used. Tutorials served as guiding implementations and understanding concepts while writing the code line by line. These were often helpful because they were implementations in the same technology context as my project.

Front-End Authentication

Płoński, Piotr. “React Routing and Components for Signup and Login.” *React and Django Tutorial*, 25 Oct. 2020, saasitive.com/tutorial/react-routing-components-signup-login/.

I leaned heavily on this tutorial in order to get a working form of registration, login and logout on the client-side. With technologies using React and Django, this tutorial provided me with specific context based on the technology stack. However, they use a different authentication

package. I utilized Knox rather than Djoser which is the authorization package used in the tutorial. In this course actions, components, and JS techniques were taught in driving the authentication system in the front-end as well. That being said many tweaks were made in the structure to customize it to ExerLog.

Front-End Authentication Serializer Connections

Traversy Media. “Full Stack React & Django [5] - Django Token Authentication.” *YouTube*, 31 Jan. 2019, www.youtube.com/watch?v=0d7cIfiydAc&t=2054s.

This Full Stack React & Django token authentication walkthrough was an excellent resource from Traversy Media. It helped me adjust and correct serializers regarding their connection to the front-end content. I was having difficulties implementing the authentication for quite some time. The token and username of the user was not correctly being pulled from the login call to the backend. I am still currently working on implementing this.

Axios Documentation

Axios. “Getting Started.” *Getting Started / Axios Docs*, axios-http.com/docs/intro. Accessed 30 May 2023.

Axios was the React library I used to fetch data from the API. With React and JavaScript being pretty foreign to me, I leaned on this documentation in order to get authentication functions running. I am still actively investigating storing tokens and data correctly within the local storage in the front end.

Axios Implementation and Example

PedroTech. “How to Make an API Request in Reactjs with Axios and Fetch - Tutorial [2020].” *YouTube*, 20 Oct. 2020, www.youtube.com/watch?v=rpg1jOvGCtQ.

Pedro Tech’s YouTube tutorial on fetching data from an API was extremely helpful in helping me get started with a library in React called Axios. This library gives you a lot of flexibility when it comes to processing requests. This video explained the details of code written in order to make these API calls.