NEXUS

Senior Capstone Project Fall 2021

Project Final Report

11/11/2021

Group: Raider Devs (Group 5)

Morgan Fisher (President) [Log-In Page (Front End), Executive Summary]

Mallory Rasco (Business Analyst) [Product Description, Level 3 Course Directory, Courses Microservice (Front-End)]

Arthur Jones (Designer) [Registration Microservice, Gantt]

Hardik Poudel (Product Developer) [Standards, Courses Microservice (Back-End)]

Darrel Jackson (Product Tester) [Registration Microservice]

Matt Padgett (Product Manager) [Authentication, User Management, and DevOps]

Table of Contents

Individual Contributions

Morgan Fisher

I had worked on implementing the login page, and the UI associated with it including the Light/Dark Mode feature in the Login Page. As well, I worked with Arthur with designing the layout and look of the registration page, and brainstorming ideas on how we were going to have the mobile interface more friendly for the user. As the team contact person, I was responsible for submitting the various reports throughout the semester and making sure they were properly formatted for submission.

Arthur Jones

I oversaw creating the registration, term, general layout of the pages, and some APIs that the registration page needed for class information. Most of my time went into the registration page as that contained the most information and logic by letting the user register and view registered classes.

Hardik Poudel

I worked on the Api integration on the frontend and build REST Apis to manage the databases for various administration tasks. I added the functionalities in the application to View/Create/Update databases of Course directory, class location, term, and campus. I also worked on designing database schemas and writing the Api endpoints.

Mallory Rasco

My area of work was the front-end of several pages for the Administration tab. The main focus was on the Course Directory service, but I also worked on the Terms, Campuses, and Class Location pages. They all had a similar set up, with a menu up top to navigate through viewing, creating, and updating the information saved for those groups.

Darrel Jackson

My main focus was the back end of the registration. I was tasked to create the APIs that were connected to the registration page. Most of my time was spent on designing the Registration APIs and how that would work and having the Registration API connect successfully to the MongoDB.

Matt Padgett

I had the goal of implementing secure authentication and authorization within the Nexus family of microservices, handling team DevOps needs, and configuring the MongoDB instance for use by the different developers. As far as authentication and authorization goes there will be more information later in the paper regarding their specific configurations and implementations. Our DevOps processes included managing 2 Ubuntu 20.04 Linux Servers named ubuntu-svr-001 and ubuntu-svr-002. Both of these servers are equipped with MicroK8s Kubernetes Nodes enabling a cluster for resource and load balancing of incoming network requests. I setup our MongoDB instance using MongoDB Cloud hosted on Microsoft Azure to mitigate networking issues that were outside of my control.

Executive Summary

The project Nexus, by the group Raider Devs, is essentially a redesign of TTU Raiderlink in order to give the user a more friendly experience and optimizing performance as well. We are focusing on a redesign of the TTU system, focusing on the elements of registration for this project. As well, we plan to have our company have a design that can be marketed and sold to other universities as well, bringing out university web infrastructure out of the 20th century into a modern design and implementation.

Here in our Technical Report, we have several contents that further explain to the reader a technical design, specifications, and how we plan to have it look for the user. The first section is our Project Goals and Objectives which will explain the aim and scope of the project. In our second section we go over the specifications and requirements of the project. Then, we have our Product Standards section, where we talk about what engineering/software standards we plan to abide by for our implementation. Our next section is our Level 1, Level 2, and Level 3 diagrams. In our level three diagram section we showcase these into four subsystems, including authentication, registration, courses, and degree requirements. After this, we have our Implementation and Functionalities section, where we explain the portions of the project that were performed by the given group member at the top of each section. This includes Authentication Microservice, Registration Microservice, Degre Requirements, Courses Microservice, Kubernetes and Microservice Lifecycle, and Log-In page. We then move on to our Gantt Chart, where we have a display of our tasks and timeframe for completing our project, and lastly, we have our references section.

Goals and Objectives

Project Nexus overall goal is to create a student information service (SIS). In this student information service students can register for classes faster during class registration period. The faculty will be able to manage the student's registration a lot easier without server crashes unlike the system currently being used. Nexus also allows students to view their degree requirements, letting the students know how much left they have on their degree.

Product Description

As a brief recap of our project, we will be doing a redesign of the TTU Raiderlink/ Banner System, both in the user interface design and system structure. We will be bringing the UI into the 21st century with a more modern look and make it more user-friendly. Behind the scenes, our product will have a microservice architecture, as opposed to the monolithic one currently in use. This will be able to handle large server loads, in situations like Registration, without overloading and crashing. This will have an increased performance, efficiency, stability, and a reduced cost.

Product Standards

The stakeholders of Nexus are the students and faculty members of Texas Tech University. We are determined to provide the best possible application and will follow the standards to ensure the privacy and security of our users.

- Nexus shall follow the General Data Protection Regulation (GDPR) to protect the personal data of the user.
- Nexus shall follow the code of ethics from ACM/IEEE to make the analysis, design, test and maintain the software.
- Nexus shall follow ISO/IEC 27001 to provide security for digital information.

We will also follow the standards from Texas Tech as follows:

- Nexus will not feature any advertisement for third party merchants, products or services.
- Nexus maintains its services respecting the privacy of its user and will not disclose any personal information of the user.

- Nexus will only be used for class registration system of Texas Tech University only and will abide by law.
- Nexus will be secure to store university databases.

Level 1 Diagram

For the level one diagram, we made a high-level overview of what we think the project would do. For this diagram we have two actors; one student and one faculty/university.
The student will be able to register for classes through the system and receive information back from the system and have the ability to view their grades.

For the faculty/university, they can add a class to the system, view student enrollment, and post midterm/final grades. Then the Nexus system is connected to a MongoDB database.

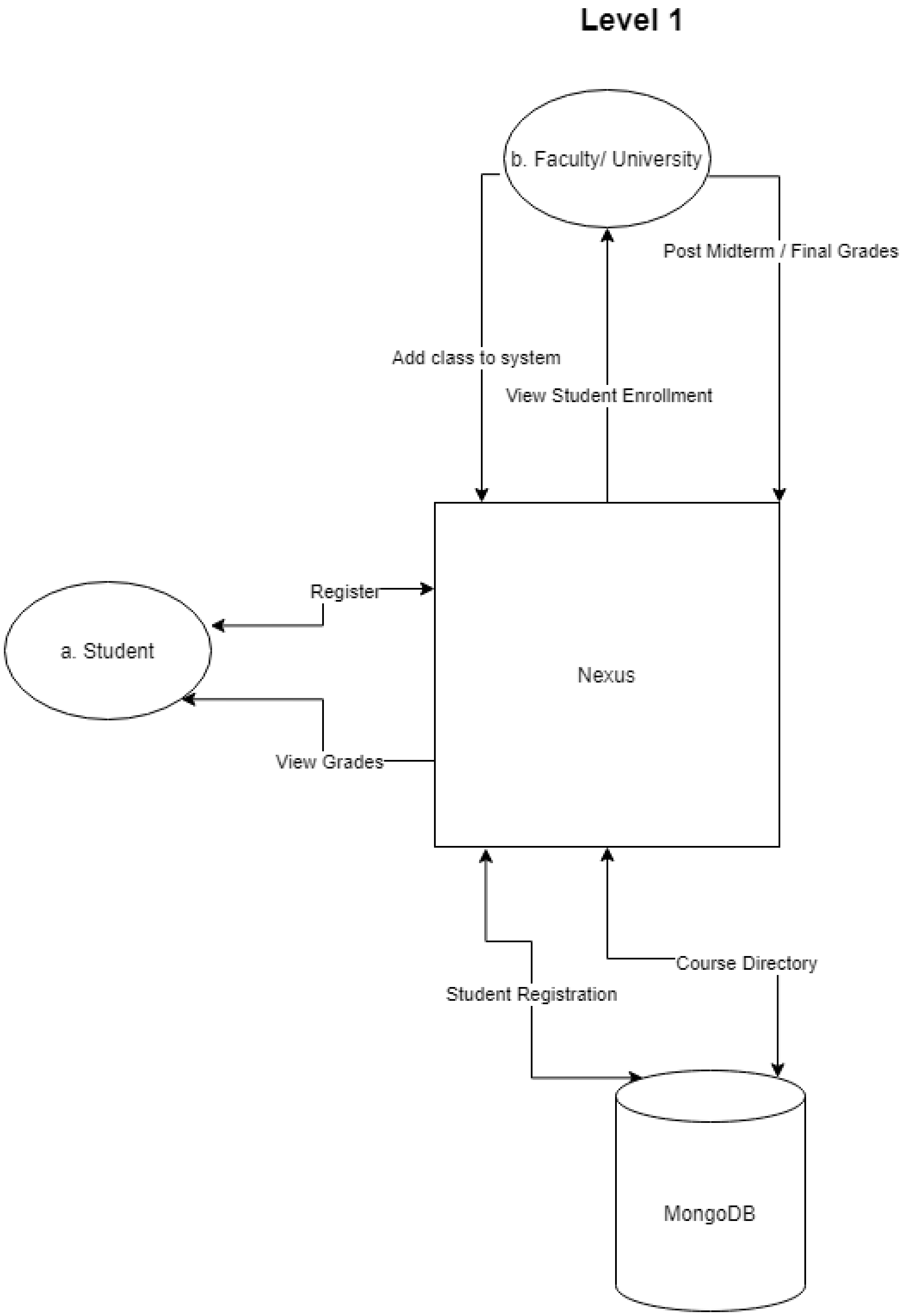


Figure 1 - Level 1 Diagram

Level 2 Diagram

For the level two diagram, we expanded the Nexus system to include four different systems; authentication, registration, degree requirements, and course directory and four different databases; registration, course directory, degree requirements, and users for the level two diagram. The authentication service will combine all services. First, the authentication service will ensure that a specific user can view and change other services by correctly authenticating the user. Next, the course directory service will allow a user to view the course directory. Next, the registration service will allow the user to view and register for classes. Next, the degree requirements will allow the user to access the classes they can register for. Finally, each service is connected to its respective databases.

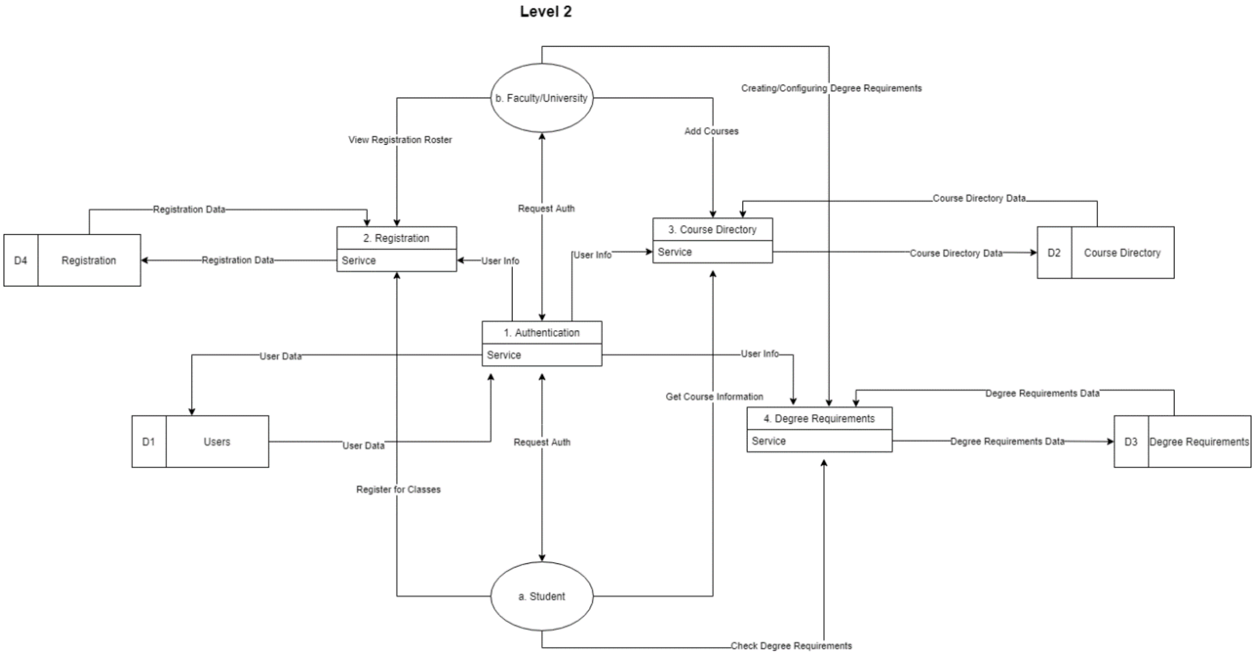


Figure 2 - Level 2 Diagram

Level 3 Diagrams

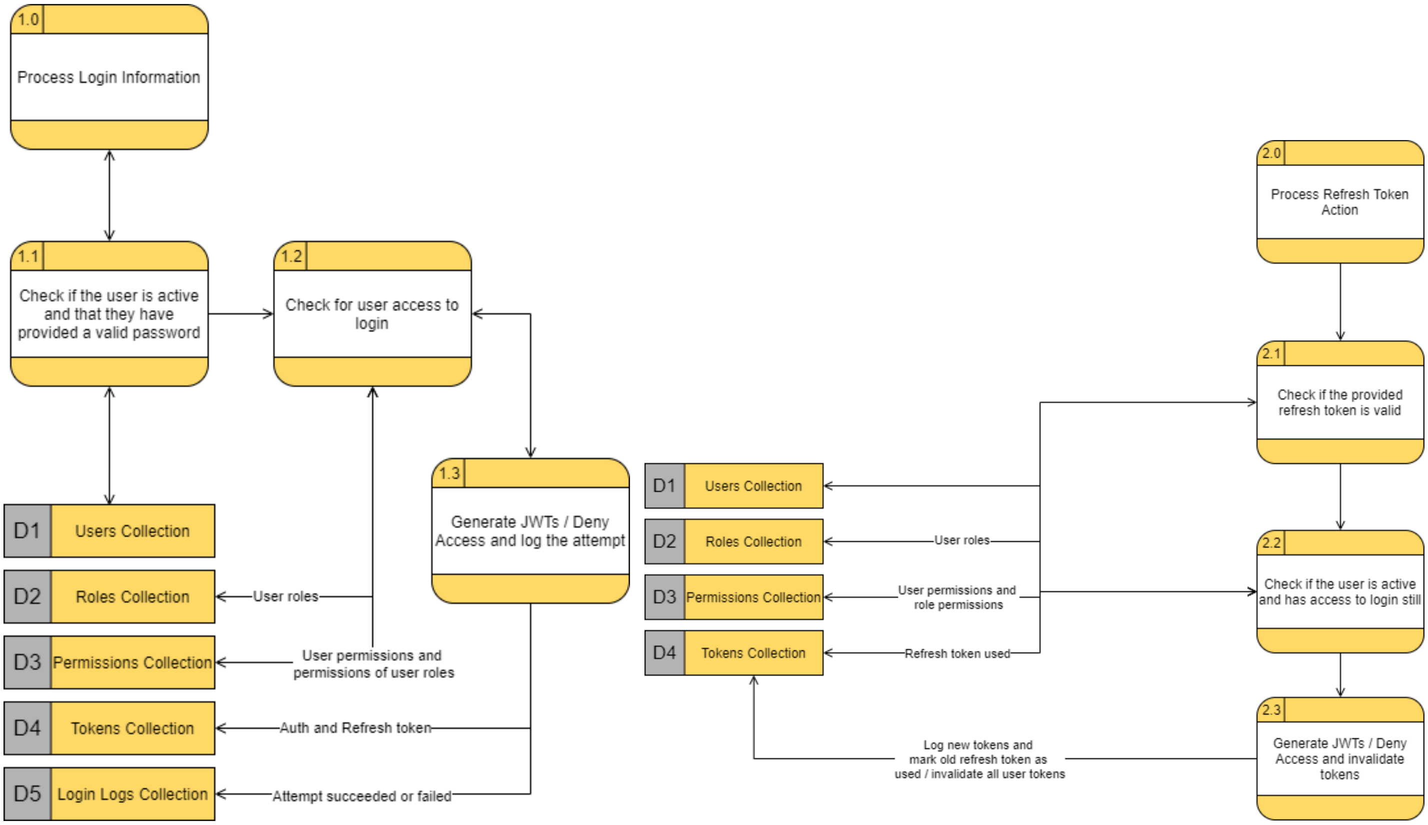
We have broken down the authentication, registration, course directory, and degree requirements services for the level three diagrams.

Authentication Level 3 Diagram

During the initial authentication process, called “login”, we verify that the user is providing a valid username and password. Once these pieces of information have been validated when then confirms that the user’s record is active and that they possess the login permission. If these both appear to be true, the system will then generate two JSON Web Tokens or JWTs for short. The generated JWTs consist of an authentication/authorization token containing the listing of permissions that a user has and a refresh token containing basic user information for security purposes. The authentication token has a defined lifetime of 15 minutes. After this time period has expired the application would automatically detect this issue and use the corresponding refresh token to generate a new token family. The refresh token itself has a defined lifetime of 4 hours. Within the 4-hour window the token can be used to generate a new token family (an authentication token and a refresh token) one time. If for some reason a refresh token is used more than once the server will invalidate all user authentication and refresh tokens and revoke access to the system prompting the user to login with their username and password again for the greatest security. A refresh process is also documented. When a user submits a refresh token the cryptography of that token is checked and then the expiration validity of that token is checked. If both of these are true and the user is still active and has access to the login permission the system will then generate a new token family and store those tokens.

Registration Level 3 Diagram

For the registration level three diagram, first, we check if the student can register during that time by checking the registration dates database tables and cross-referencing the student's classification with the registration dates. For example, if the student is a senior, it checks the senior field in the database and looks at the date in that field. If the student is not able to register during that time, we will display a warning. If so, the student will continue with the registration process. Then the student will choose a class in the system and try to register for that class. The system will then check if the student can register for that class by checking the student's degree


1.0
Process Login Information
1.1
Check if the user is active and that they have provided a valid password
1.2
Check for user access to login
1.3
Generate JWTs / Deny Access and log the attempt
D1 Users Collection
D2 Roles Collection
D3 Permissions Collection
D4 Tokens Collection
D5 Login Logs Collection
User roles
User permissions and permissions of user roles
Auth and Refresh token
Attempt succeeded or failed
2.0
Process Refresh Token Action
2.1
Check if the provided refresh token is valid
2.2
Check if the user is active and has access to login still
2.3
Generate JWTs / Deny Access and invalidate tokens
D1 Users Collection
D2 Roles Collection
D3 Permissions Collection
D4 Tokens Collection
User roles
User permissions and role permissions
Refresh token used
Log new tokens and mark old refresh token as used / invalidate all user tokens

requirements. If the student is not able to register for that class, we will display a warning. If so, the student will register for that course and update the user's registration courses database table.

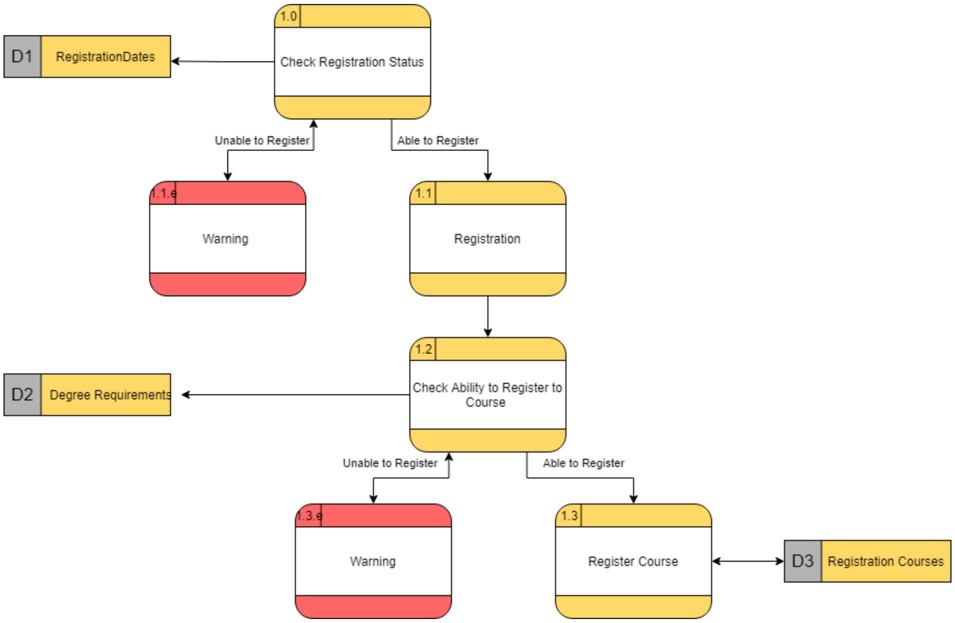


Figure 1 - Registration Level 3

Course Directory Level 3 Diagram

When people login to the system, they will receive different permissions from Authentication for how they can interact with the course directory service, depending on their roles. Students can only read the information stored in these databases, while faculty can read and write to them.

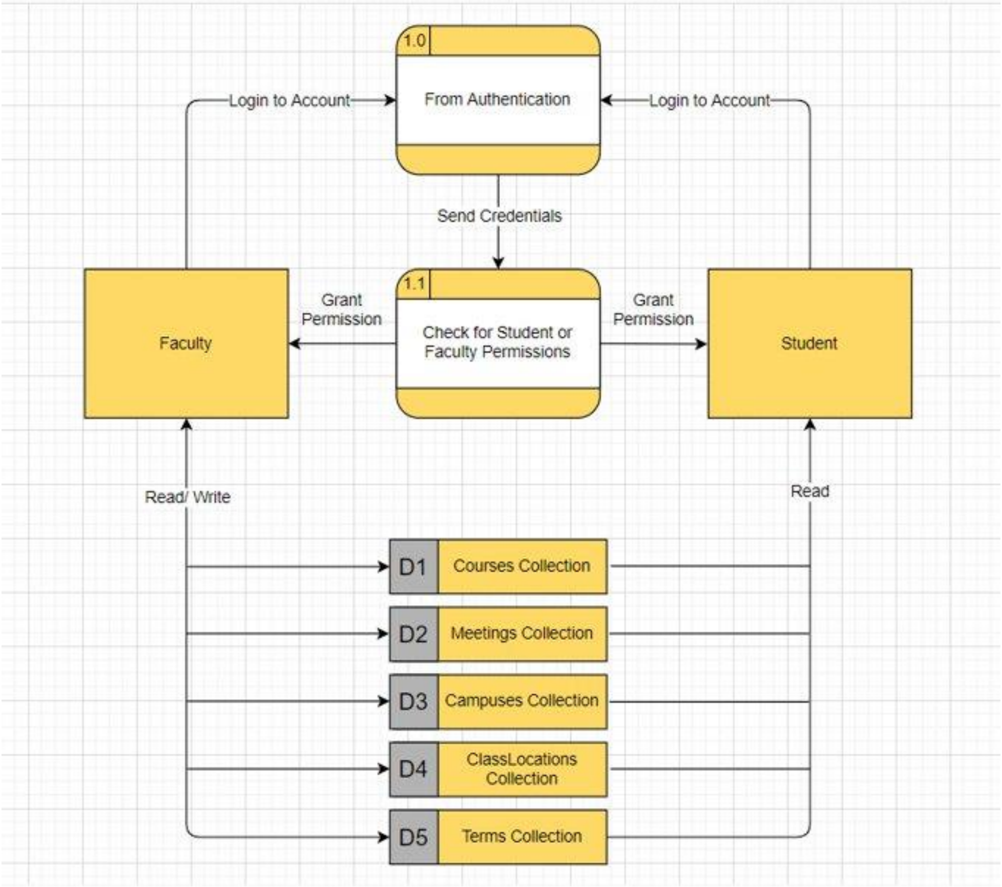


Figure 2 – Courses level 3

Degree Requirements Level 3 Diagram

When login from the authentication service the user is now in the user's degree requirements. It accesses the degree requirements database and the degree catalog year database. With this the user can change the information that is originally be provided like a “what if” if they want to see how their progress would look in a different catalog year or even in a different major. After all that the user is then able to see their major requirements, core requirements and minor requirements if they have those and this keeps track of what is left over.

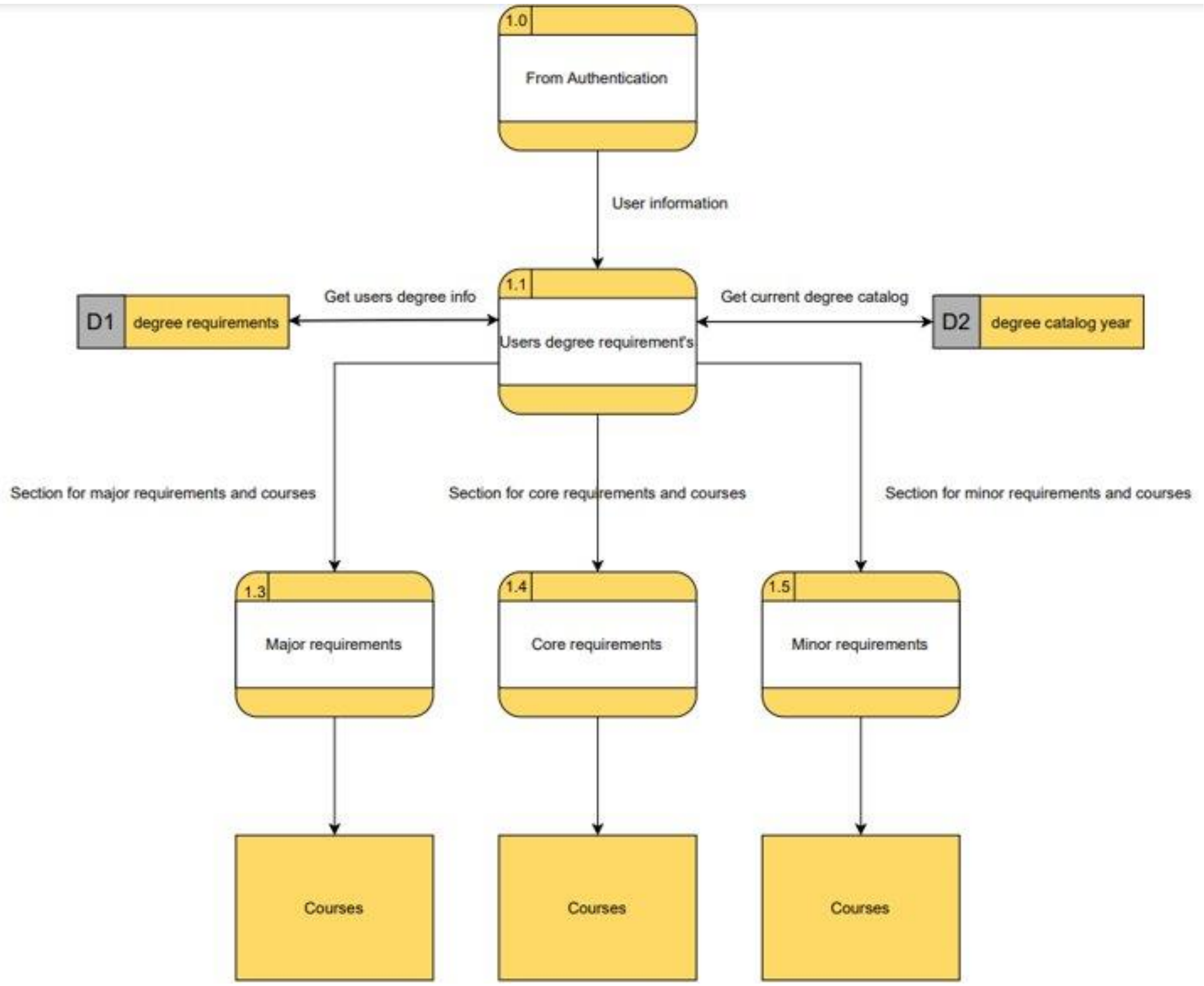


Figure 3 - Degree requirements level 3

Implementations and Functionalities (Backend and Frontend)

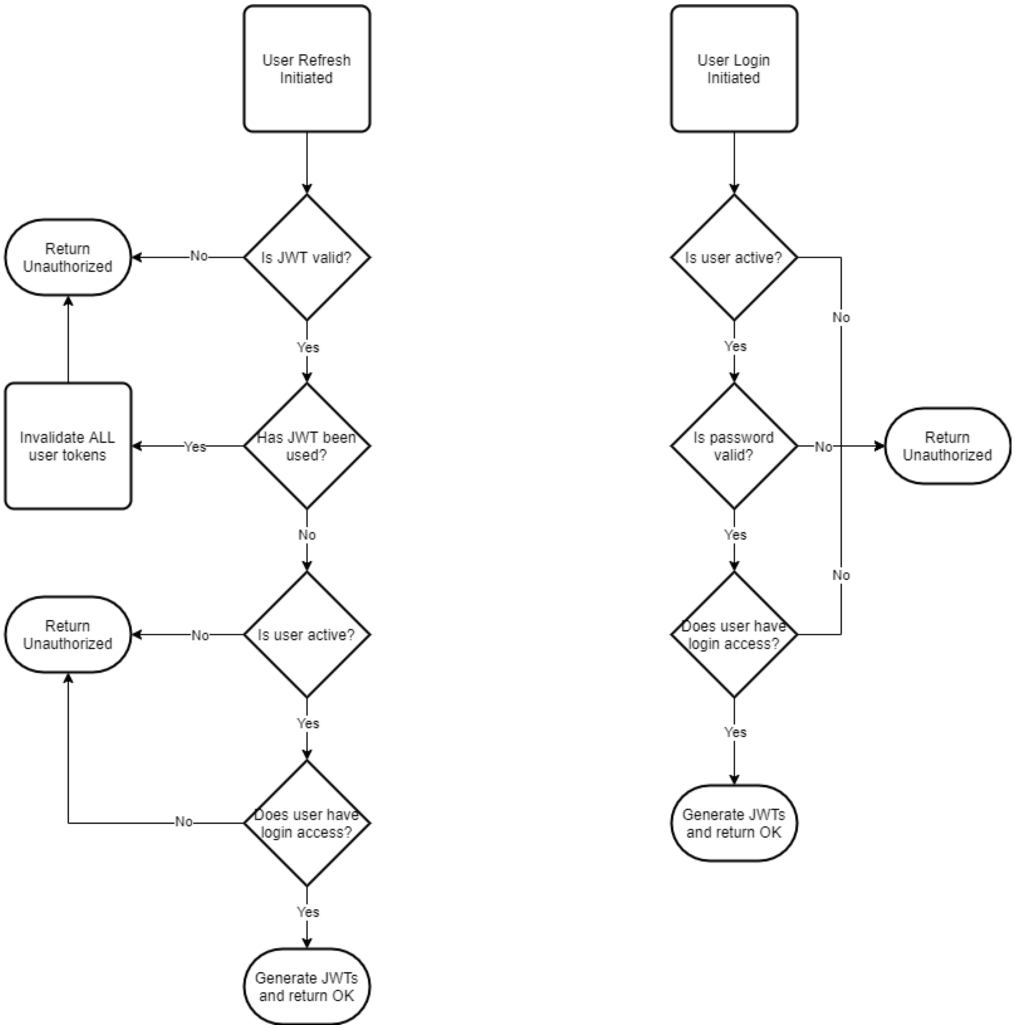
Authentication Microservice – Matt

The authentication and authorization microservice is a core functionality in Nexus. It consists of a .NET Core Web API and a gRPC service for handling user information. This service controlled core user information such as names, addresses, phone numbers, email addresses, and – theoretically – social security numbers. These security principles were upheld using the concept of a set of two JSON Web Tokens (JWTs). These JWTs were divided into their respective concepts of Authorization Token and Refresh Token.

An Authorization Token (or Access Token) is the key to the kingdom so to speak. This token is used and included in every single HTTP request made to a Nexus microservice. It holds crucial pieces of information about the currently authenticated user: their user identifier and an array of keys which identify the permissions that they have the access to perform. What makes this token secure is the server-side encryption key used. While a JWT is readable by anyone with the token, it cannot be changed by anyone other than the issuing server. Additionally, JWTs have an expiration date and cannot be used after that date if the server is implemented properly. When a request is made the Authorization Token is first verified to be valid by checking the encryption integrity and verifying that the token is not expired. Once this check has been completed the user identifier is pulled from the claims of the token and is sent over gRPC call to check if that user identifier has access to utilize the specified endpoint. If not a 401 response is sent back to the client. Otherwise, the endpoint actions are executed as normal.

Before discussing the uses of a Refresh Token we must first understand why they how they exist in the first place. In the "/user/login" endpoint a user will provide two pieces of identifying information (their username and password). These will be used according to the flow chart at the bottom of this section. We will check if the user is active, if they have provided a valid password, and that the user has the permissions needed to login to the application. If all of these are true an Authorization Token and a Refresh Token are generated and returned to the client for browser local storage. Otherwise, a 401 header will be returned to the client and the user will not be granted any permissions within their current security context.

Refresh Tokens are slightly different than Authorization Tokens. They only contain the user identifier. The major difference between a Refresh and Authorization Token is that an Authorization Token is only valid for 5 minutes while a Refresh Token is valid for a period of 4 hours. Another important note is that a Refresh Token can be used only once while a Authorization Token does not have a limit. According to the flowchart at the bottom of this section the refresh flow is as follows. We check if the JWT provided is valid by encryption key and expiration date. If this is not true we return a 401 Unauthorized header to the user and invalidate all of their existing tokens forcing the user to log back in on all devices or clients. The same thing will happen if the token is active but has already been used. If both of these things pass their checks we will check if the specified user is still active in the system and still have the login permission. If these pass we will generate a new token family of an Authorization Token and Refresh Token and return them to the client for local browser storage. This is modeled after the concepts of authentication used by popular authorization provider Auth0. This was developed as an in-house alternative to keep the need for increased security and not sharing crucial user information to an external source.

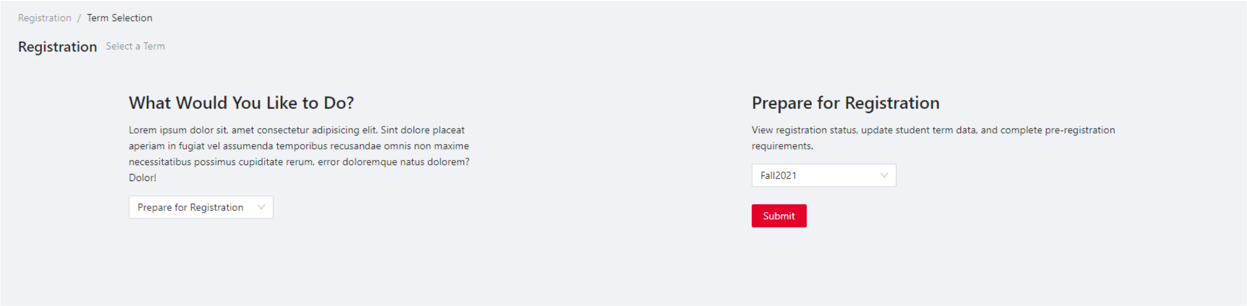


Registration Microservice – Darrel / Arthur

First, we started with making the Terms page. With the terms page, a user is able to select what they would like to do on the left side and then select a term on the right side. Depending on the selection of the left side, the content on the right side will change reflecting that selection. The Terms page pulls its available terms list from the Courses API that was made by Mallory/Hardik.

After implementing the terms page, we focused on the registration page. This page contained most of the intricate work for the registration frontend and backend. We tailored the page to be very similar to the current TTU banner registration page. However, we wanted to bring some of the features to a more modern era. We also focused on mobile registration since the current TTU mobile registration is nearly impossible to do. At the top of the registration page, you can see the user can search for classes. A user can filter depending on the subject name, course number, and CRN. If the user filters their search query, it will return that filtered response. If not, it will show all queries. After searching for a class, a user can add a particular class to their class list. However, if a user is trying to add a class that is already on their class list, it will give the user an error saying they cannot add a duplicate class. Also, if a user is trying to add a class, not in their major, it will give an error saying the class is not major/minor. Once a class is added to the user's class list if on desktop, they will see it on the scheduler on the bottom left and a detailed view on the bottom right. The scheduler will allow the user to see what time slot the class takes up, and the detailed view shows the user some more details on the class. When the class title is clicked, it will show a popup of even more details of the selected class. The popup is currently not populating the details of the current class. However, that is the intended purpose. A user can either register or delete a class within the detailed view on the bottom right. Depending on the option, the Registration API made by Darrel and myself will search for that class and then delete or register it for the user.

For the Registration Classes API Service, it was made to hold all the visual class information for the registration page. It holds information like; _id: its own auto generated id, meetingId: the ID that contains all the class's information, userId: the id of the user it belongs to, registationStatus: the status of that particular class if it is registered or not, title: the title of the course, courseType: if the course is a lecture or online, hours: how many credit hours the course is, crn: the crn of the course, times: a list of times of the course, and details: a place that holds a combined string of the department abbreviation, course number, and section number.



Registration / Term Selection

Registration Select a Term

What Would You Like to Do?

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Sint dolore placeat aperiam in fugiat vel assumenda temporibus recusandae omnis non maxime necessitatibus possimus cupiditate rerum, error doloremque natus dolorem? Dolor!

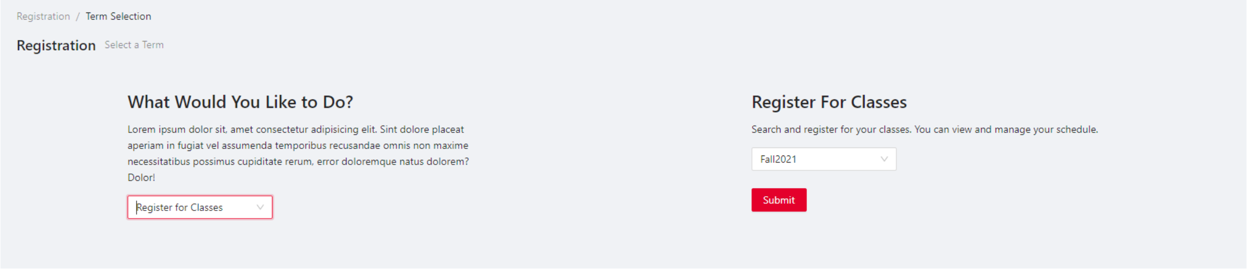
Prepare for Registration

Prepare for Registration

View registration status, update student term data, and complete pre-registration requirements.

Fall2021

Figure 4 - Term Selection 1



Registration / Term Selection

Registration Select a Term

What Would You Like to Do?

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Sint dolore placeat aperiam in fugiat vel assumenda temporibus recusandae omnis non maxime necessitatibus possimus cupiditate rerum, error doloremque natus dolorem? Dolor!

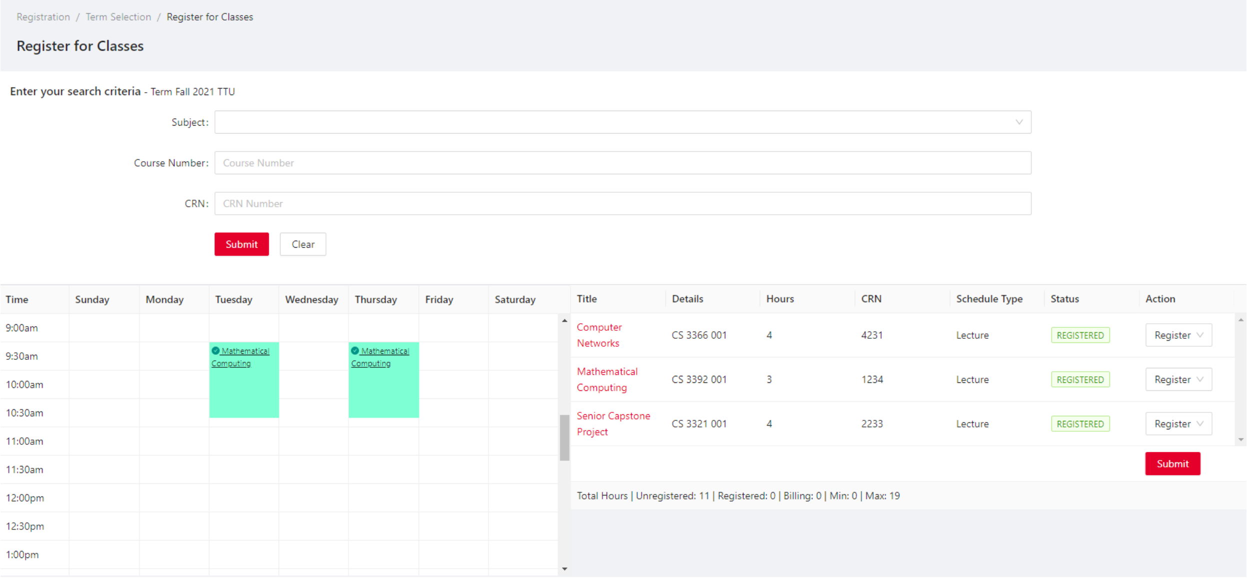
Register for Classes

Register For Classes

Search and register for your classes. You can view and manage your schedule.

Fall2021

Figure 5 - Term Selection 2



Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Subject:

Course Number: Course Number

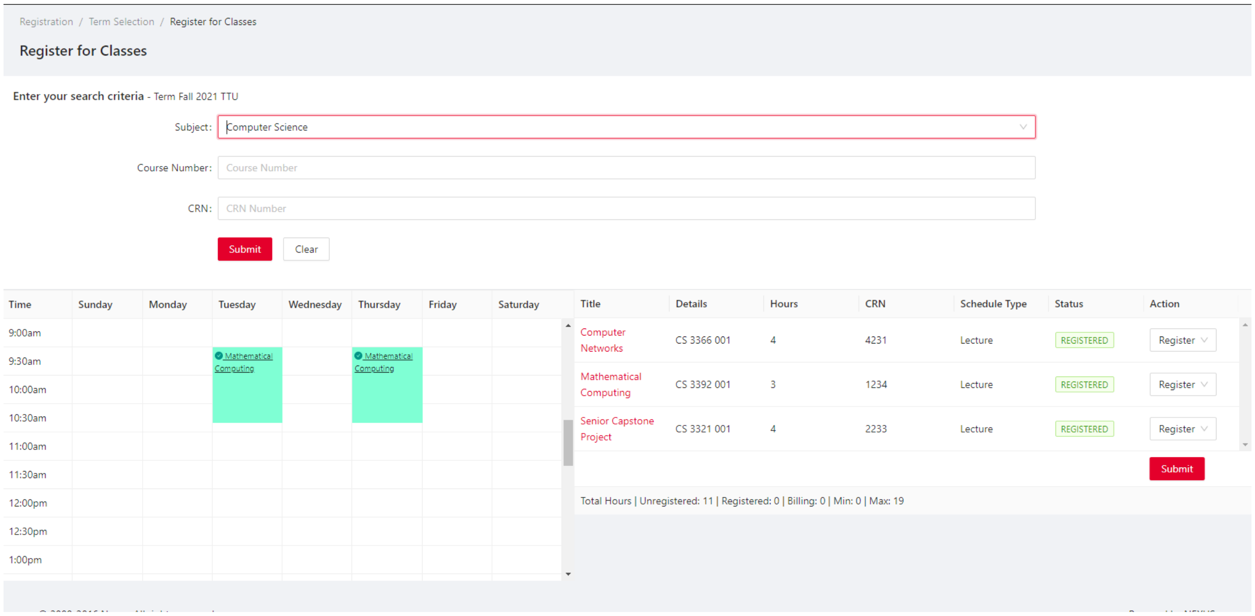
CRN: CRN Number

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							
12:30pm							
1:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register

Total Hours | Unregistered: 11 | Registered: 0 | Billing: 0 | Min: 0 | Max: 19

Figure 6 - Registration Initial View



Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Subject: Computer Science

Course Number: Course Number

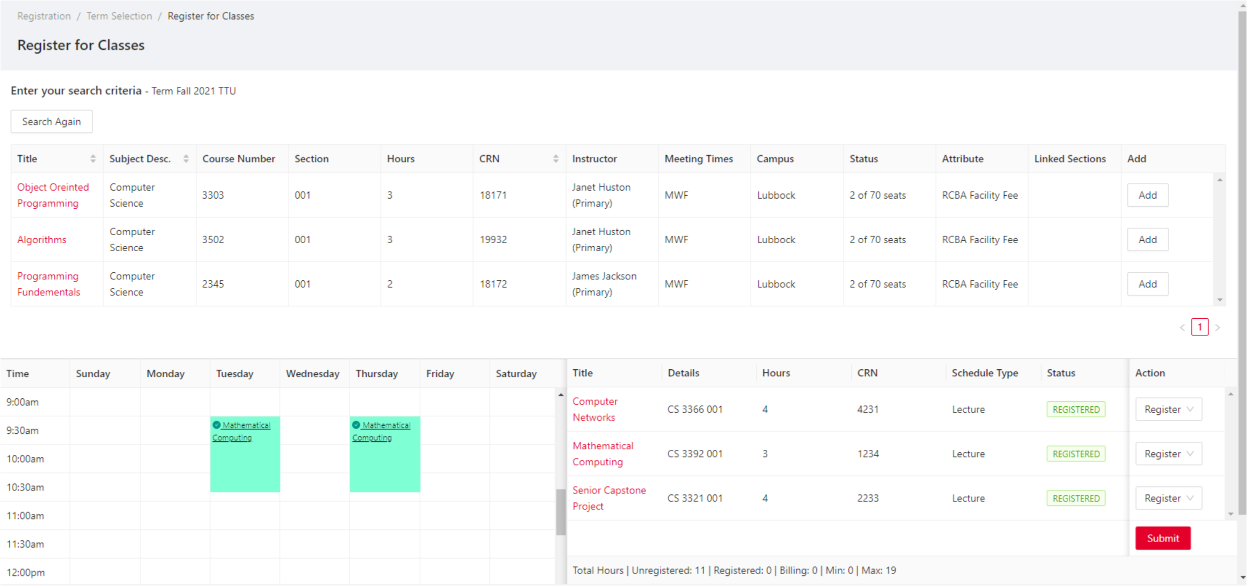
CRN: CRN Number

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							
12:30pm							
1:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register

Total Hours | Unregistered: 11 | Registered: 0 | Billing: 0 | Min: 0 | Max: 19

Figure 7 - Registration Subject Selection Case



Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

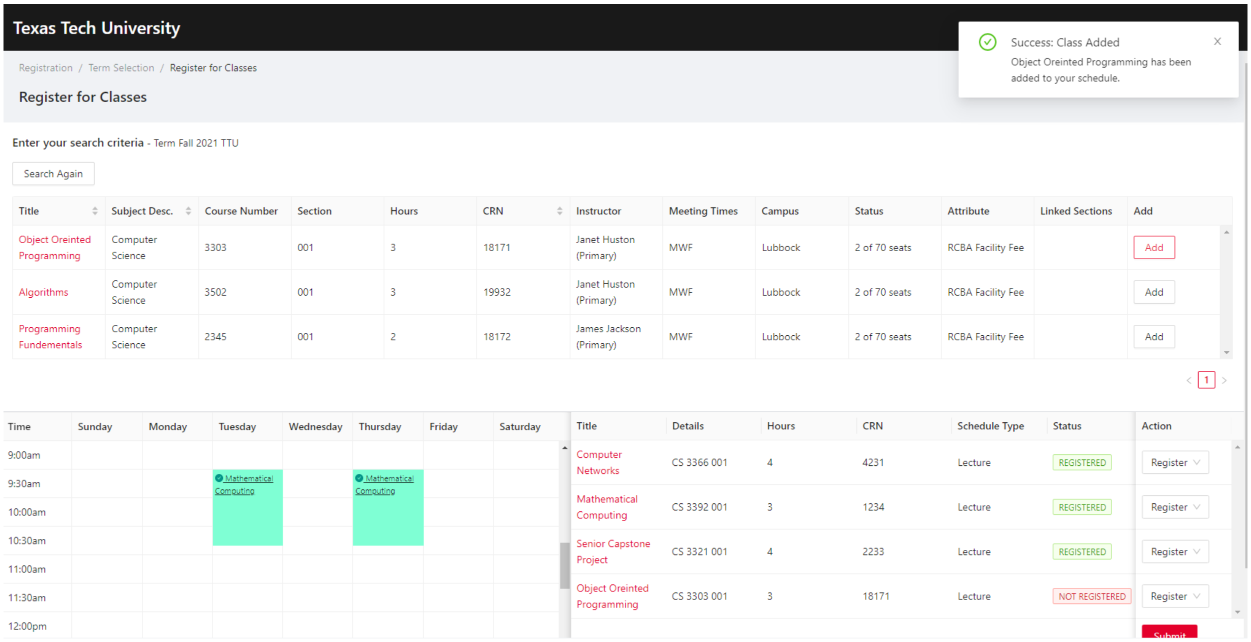
Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Object Oreinted Programming	Computer Science	3303	001	3	18171	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Algorithms	Computer Science	3502	001	3	19932	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Programming Fundementals	Computer Science	2345	001	2	18172	James Jackson (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register

Total Hours | Unregistered: 11 | Registered: 0 | Billing: 0 | Min: 0 | Max: 19

Figure 8 - Registration Subject Selection Case Results



Texas Tech University

Success: Class Added

Object Oreinted Programming has been added to your schedule.

Registration / Term Selection / Register for Classes

Register for Classes

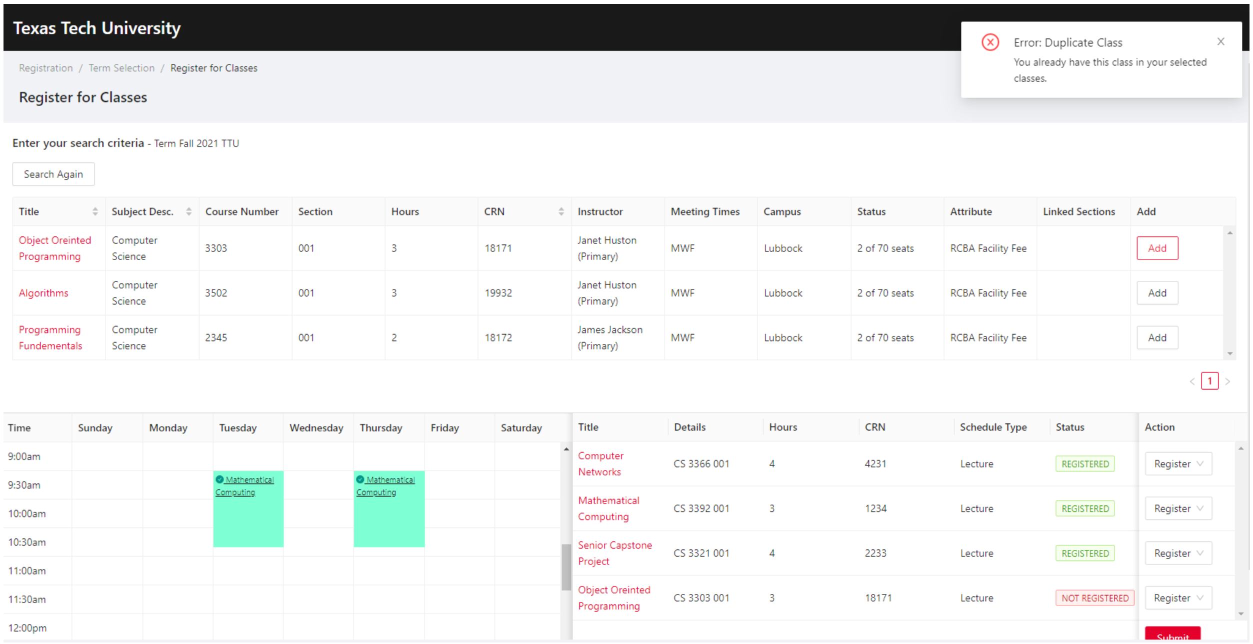
Enter your search criteria - Term Fall 2021 TTU

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Object Oreinted Programming	Computer Science	3303	001	3	18171	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Algorithms	Computer Science	3502	001	3	19932	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Programming Fundementals	Computer Science	2345	001	2	18172	James Jackson (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	NOT REGISTERED	Register

Figure 9 - Registration Adding a Class



Texas Tech University

Error: Duplicate Class

You already have this class in your selected classes.

Registration / Term Selection / Register for Classes

Register for Classes

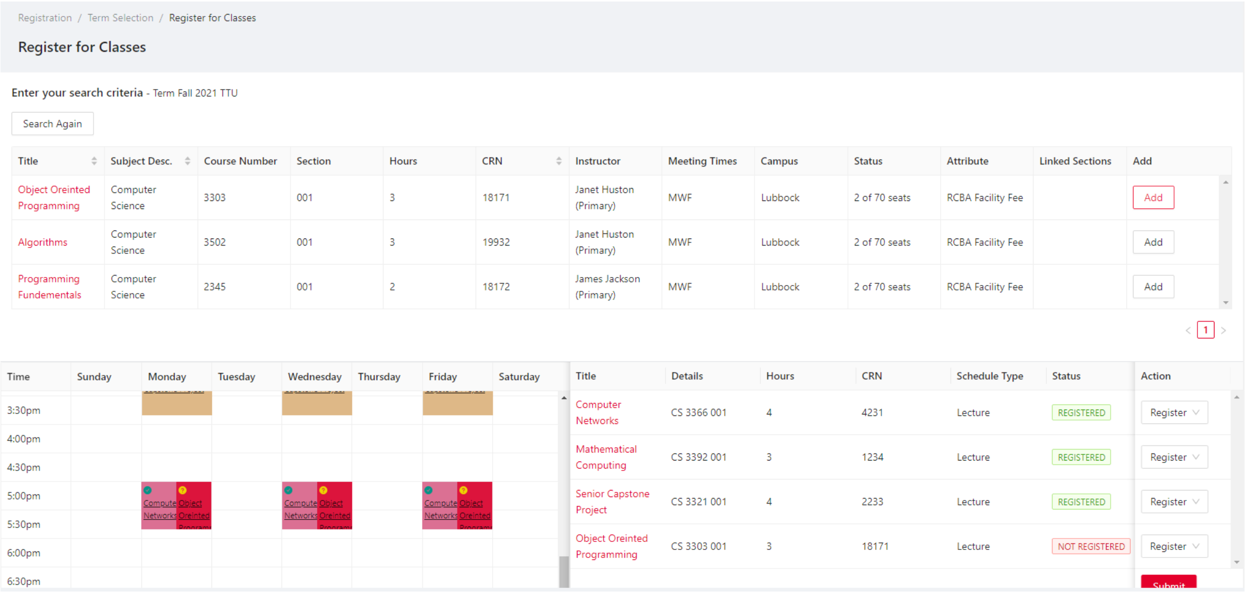
Enter your search criteria - Term Fall 2021 TTU

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Object Oreinted Programming	Computer Science	3303	001	3	18171	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Algorithms	Computer Science	3502	001	3	19932	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Programming Fundementals	Computer Science	2345	001	2	18172	James Jackson (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	NOT REGISTERED	Register

Figure 10 - Registration Adding a Duplicate Class Error Case



Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Search Again

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Object Oreinted Programming	Computer Science	3303	001	3	18171	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Algorithms	Computer Science	3502	001	3	19932	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add
Programming Fundementals	Computer Science	2345	001	2	18172	James Jackson (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

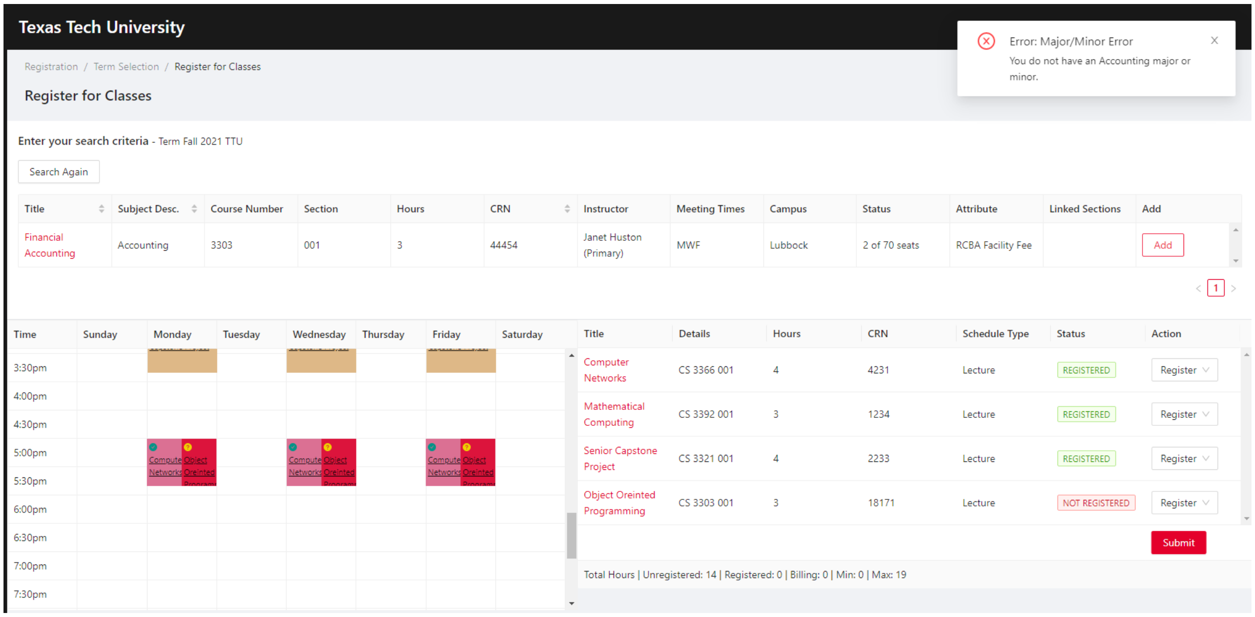
1

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
3:30pm							
4:00pm							
4:30pm							
5:00pm		Compute Networks Object Oreinted Program		Compute Networks Object Oreinted Program		Compute Networks Object Oreinted Program	
5:30pm							
6:00pm							
6:30pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	NOT REGISTERED	Register

Submit

Figure 11 - Registration General View with Updated Components



Texas Tech University

Error: Major/Minor Error

You do not have an Accounting major or minor.

Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Search Again

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Financial Accounting	Accounting	3303	001	3	44454	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

1

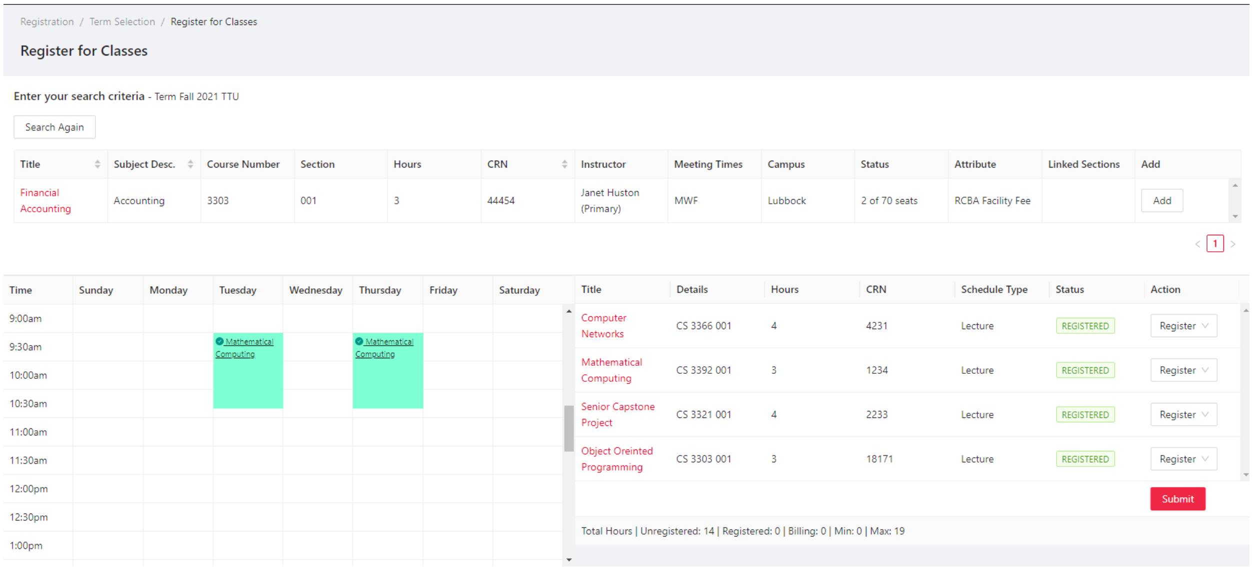
Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
3:30pm							
4:00pm							
4:30pm							
5:00pm		Compute Networks Object Oreinted Program		Compute Networks Object Oreinted Program		Compute Networks Object Oreinted Program	
5:30pm							
6:00pm							
6:30pm							
7:00pm							
7:30pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	NOT REGISTERED	Register

Submit

Total Hours | Unregistered: 14 | Registered: 0 | Billing: 0 | Min: 0 | Max: 19

Figure 12 - Registration Adding a Non-Major/Minor Error Case



Registration / Term Selection / Register for Classes

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Search Again

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Financial Accounting	Accounting	3303	001	3	44454	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

< 1 >

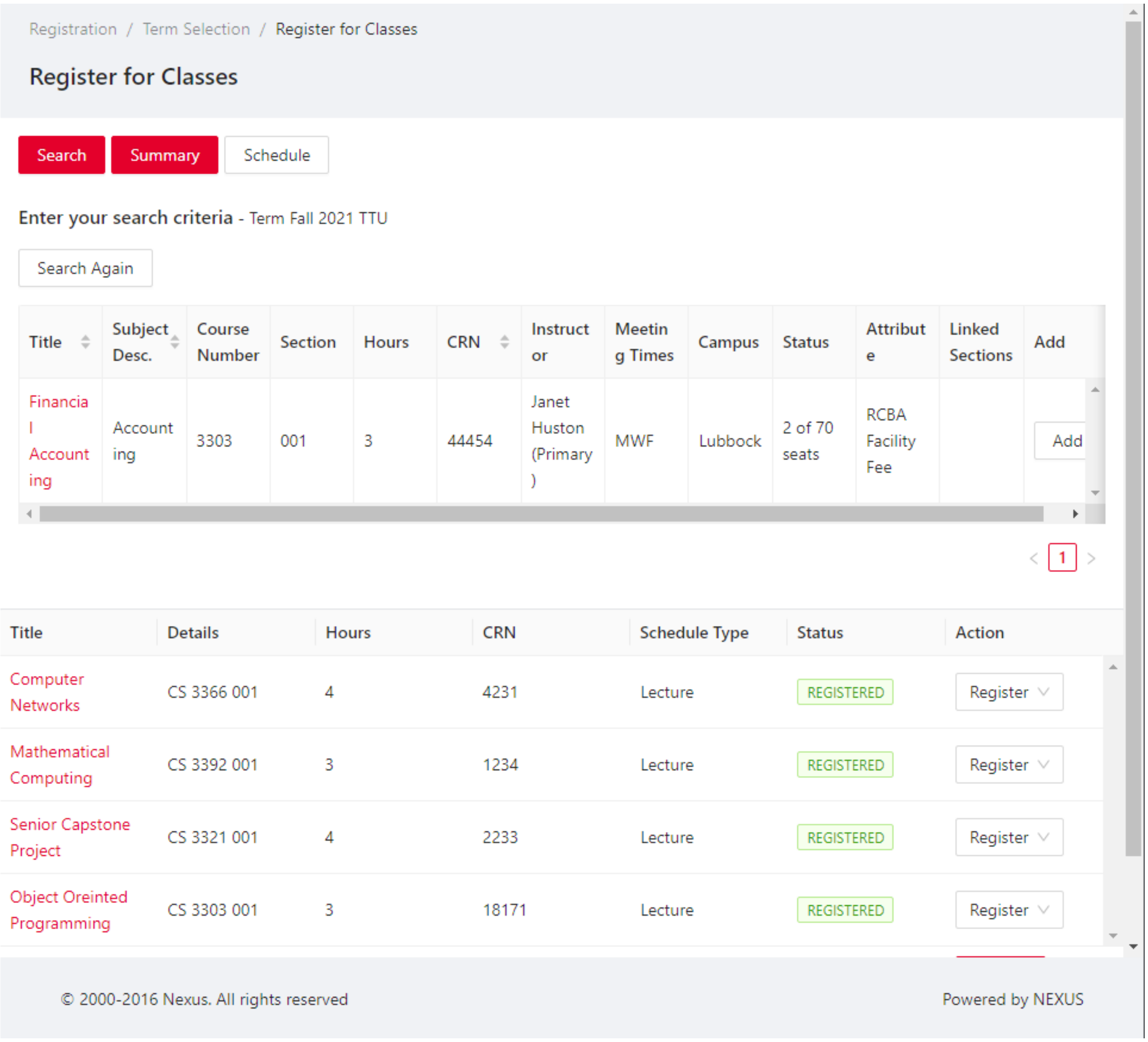
Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am			Mathematical Computing		Mathematical Computing		
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							
12:30pm							
1:00pm							

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	REGISTERED	Register

Submit

Total Hours | Unregistered: 14 | Registered: 0 | Billing: 0 | Min: 0 | Max: 19

Figure 13 - Registration Successfully Registering for a Class



Register for Classes

Enter your search criteria - Term Fall 2021 TTU

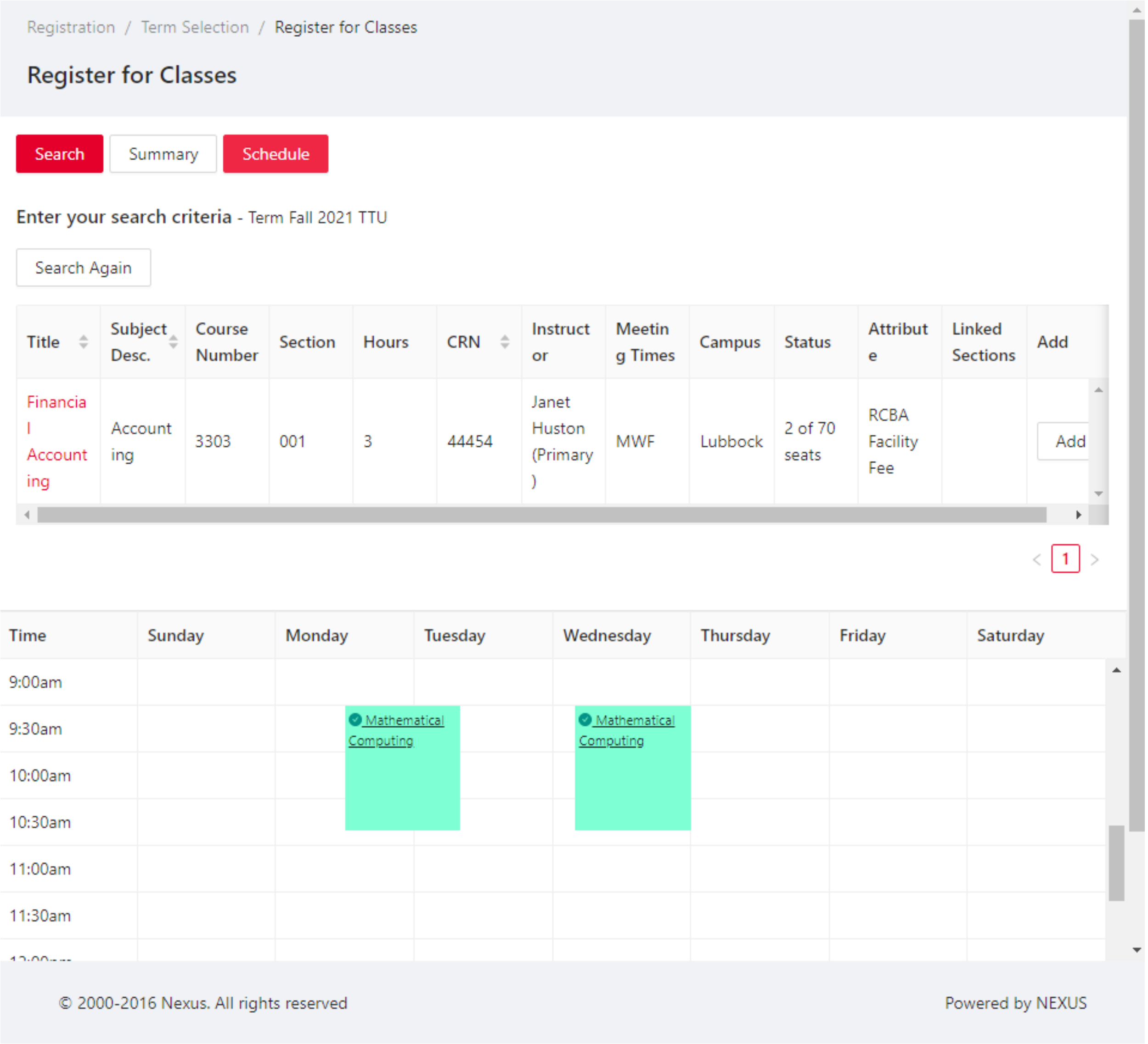
Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Financial Accounting	Accounting	3303	001	3	44454	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

Title	Details	Hours	CRN	Schedule Type	Status	Action
Computer Networks	CS 3366 001	4	4231	Lecture	REGISTERED	Register
Mathematical Computing	CS 3392 001	3	1234	Lecture	REGISTERED	Register
Senior Capstone Project	CS 3321 001	4	2233	Lecture	REGISTERED	Register
Object Oreinted Programming	CS 3303 001	3	18171	Lecture	REGISTERED	Register

© 2000-2016 Nexus. All rights reserved

Powered by NEXUS

Figure 14 - Registration Mobile View 1


Registration / Term Selection / Register for Classes
Search
Summary
Schedule
Search Again
1
© 2000-2016 Nexus. All rights reserved
Powered by NEXUS

Register for Classes

Enter your search criteria - Term Fall 2021 TTU

Title	Subject Desc.	Course Number	Section	Hours	CRN	Instructor	Meeting Times	Campus	Status	Attribute	Linked Sections	Add
Financial Accounting	Accounting	3303	001	3	44454	Janet Huston (Primary)	MWF	Lubbock	2 of 70 seats	RCBA Facility Fee		Add

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
9:00am							
9:30am		✓ Mathematical Computing		✓ Mathematical Computing			
10:00am							
10:30am							
11:00am							
11:30am							

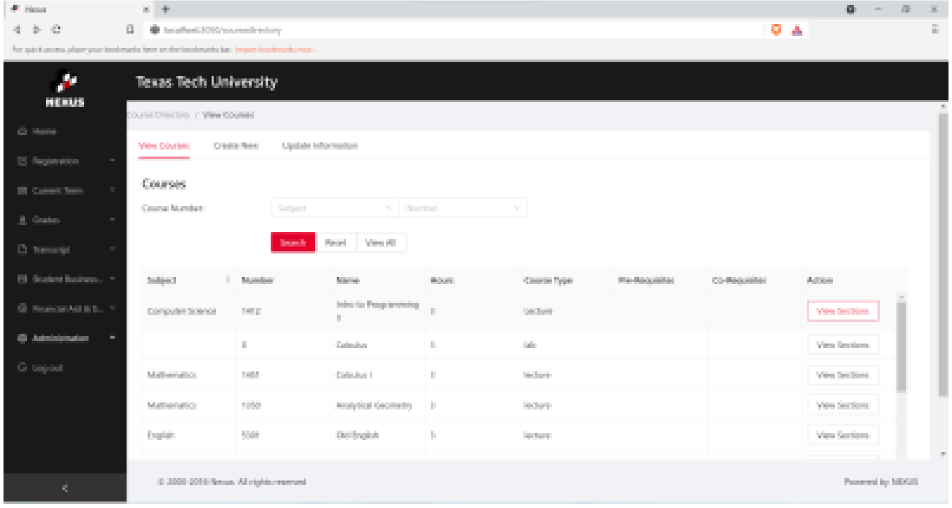
Figure 15 - Registration Mobile View 2

Degree Requirements Microservice – Matt

The degree requirements microservice is a very simple one. It is one that would have much more content if Nexus we to continue into the next stage of development. In its current state it is simply a resource API that provides a list of the different degrees offered by a university utilizing Neuxs. It handles majors, minors, and degree programs available. There is not much more to say on this API as it is a simple consumption only resource in the current stage.

Courses Microservice – Mallory / Hardik

The front-end of the Course Directory page is split into three sections: viewing, creating, and updating for the courses and individual sections. This is all navigable through a menu across the top of the page.

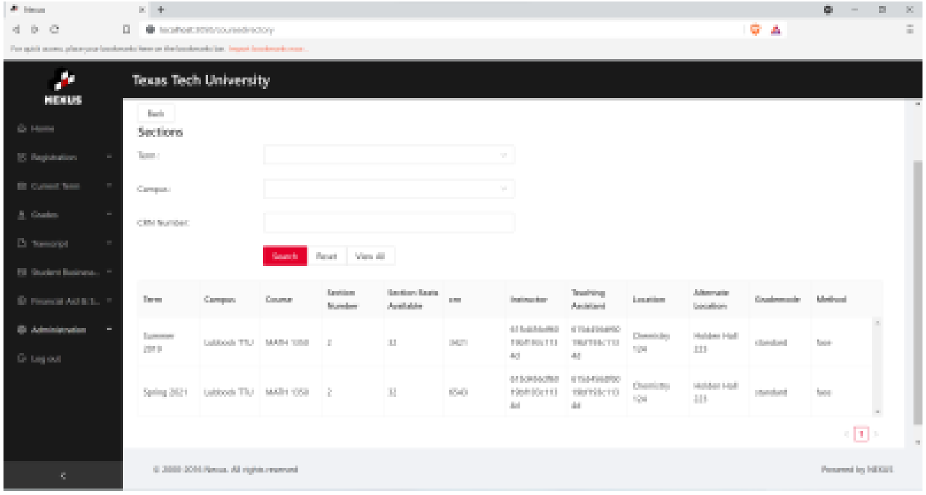


Subject	Number	Name	Hours	Course Type	Pre-Requisites	Co-Requisites	Action
Computer Science	1412	Intro to Programming II	3	lecture			View Sections
	0	Calculus	1	lab			View Sections
Mathematics	1451	Calculus I	3	lecture			View Sections
Mathematics	1350	Analytical Geometry	3	lecture			View Sections
English	5301	Old English	3	lecture			View Sections

© 2000-2016 Nexus. All rights reserved

Powered by NEXUS

Figure 16 – Courses View



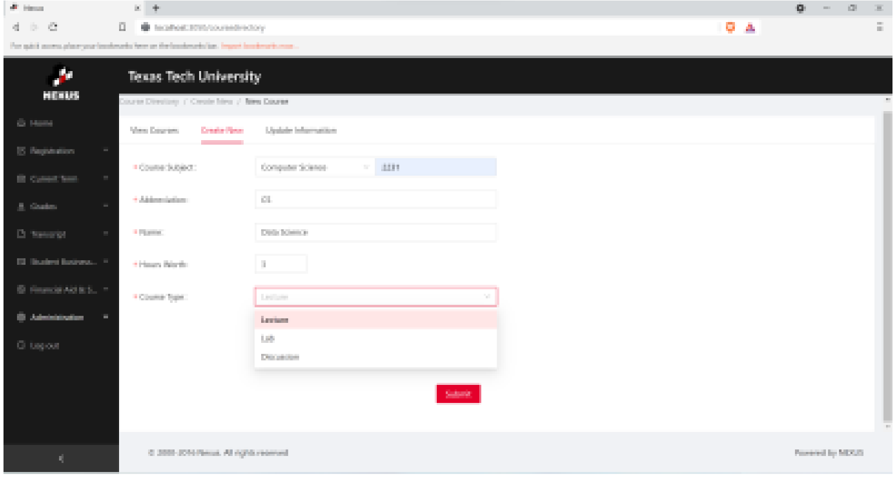
Term	Campus	Course	Section Number	Section Seats Available	crn	Instructor	Teaching Assistant	Location	Alternate Location	Grademode	Method
Summer 2019	Lubbock TTU	MATH 1350	2	32	3421	[illegible]	[illegible]	Chemistry 124	Holden Hall 223	standard	face
Spring 2021	Lubbock TTU	MATH 1350	2	32	6543	[illegible]	[illegible]	Chemistry 124	Holden Hall 223	standard	face

© 2000-2016 Nexus. All rights reserved

Powered by NEXUS

Figure 17 – Section View

The viewing section is a simple table. There is a search bar at the top for the course subject and number, and when the table filters out the chosen course, there is a button to open another table with the corresponding sections. There is also an input field to filter out the sections in a particular term or from a specific campus.



Course Directory / Create New / New Course

* Course Subject: Computer Science 2231

* Abbreviation: CS

* Name: Data Science

* Hours Worth: 3

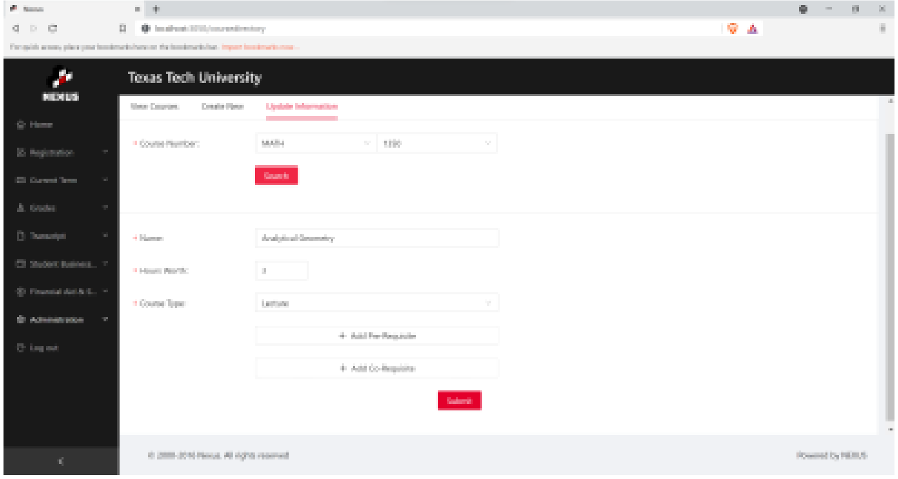
* Course Type: Lecture

Lecture
Lab
Discussion

© 2000-2014 Nexus. All rights reserved

Figure 18 – Create Course

When creating a new course or section, there is a straightforward form with input fields for all the information needed. The required ones are adequately labeled, to help the user notice them, and will highlight red with a message if are left blank when trying to submit. All the fields were chosen to make filling out the form as easily as possible. There are many drop-down menus, like for the course subjects, terms, and instructors, with search functionality. This prevents users from trying to enter any incompatible data.



* Course Number: MATH 1330

* Name: Analytical Geometry

* Hours Worth: 3

* Course Type: Lecture

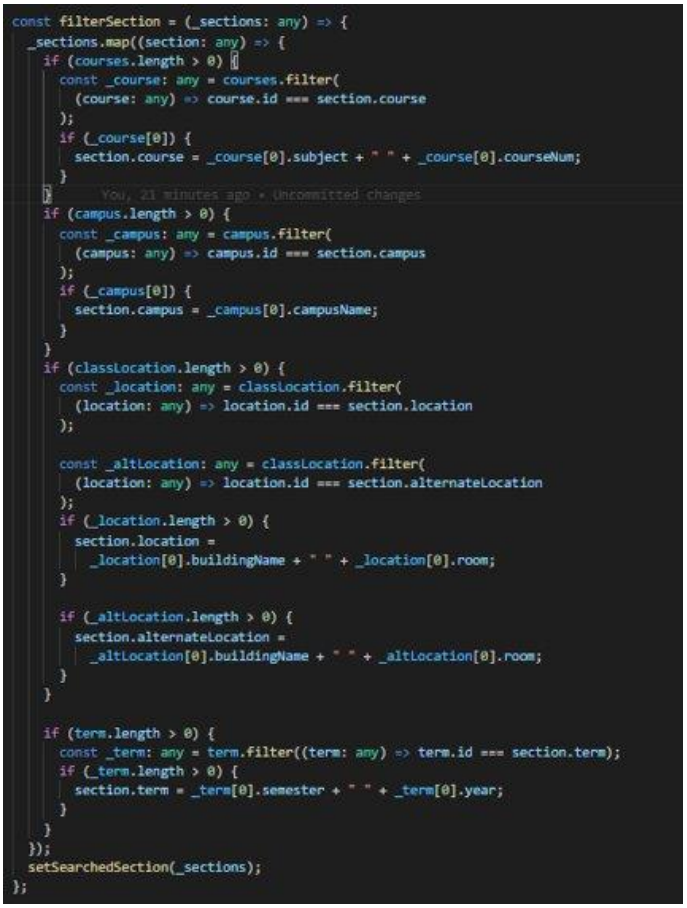
Figure 19 – Update Course

If a course or section's data needs to be updated, there is a page for that too. There are input fields at the top of the page to find the correct course to be changed. Once the search data is input, the fields below automatically populate with the current information. The user is then free to change what they need and re-submit. For the sections, the search information brings up a table with the possible sections that match the course number that was input. From those, the user can choose the one to update, and the fields populate with the data chosen.

The basic setup for the Terms, Campuses, and Class Location pages matches the Course Directory's. Everything was designed and implemented to create an easy and simple experience for the users.

Backend of the administration tasks has Api controllers to fetch data from the database and integrate it to the frontend. There are models of the database, services, and controllers to perform CRUD operations with http requests (GET, POST, PUT, and DELETE).

Some of the data fields in the database in mongo contains object ids to associate between databases. During the Api integration of those data in the frontend, we wanted to display it in a human readable format instead of long numbers and characters. So, we used the map and filter functions of JavaScript to reassign the field data with their names to display it in the table.



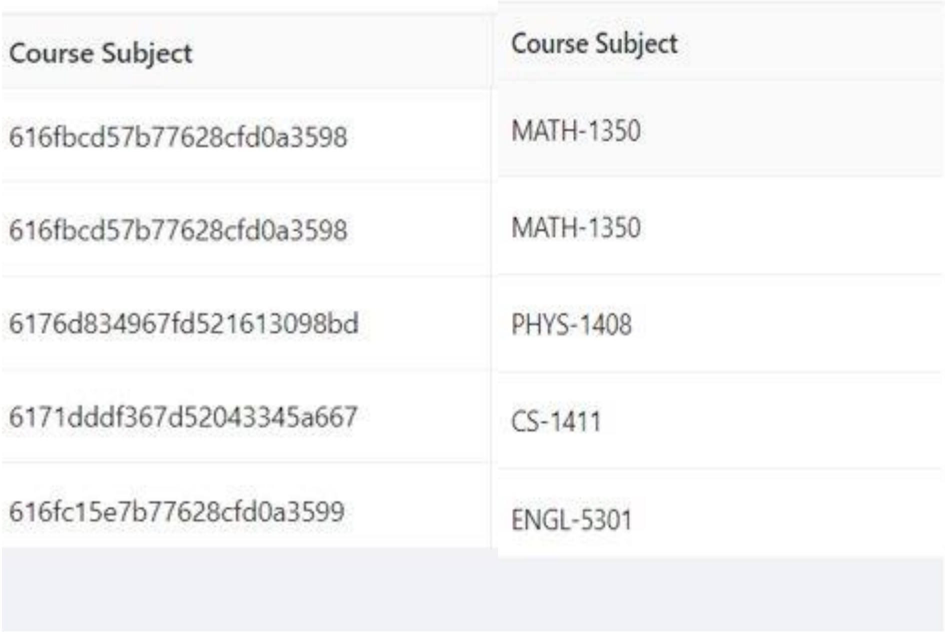
```
const filterSection = (_sections: any) => {
  _sections.map((section: any) => {
    if (courses.length > 0) {
      const _course: any = courses.filter(
        (course: any) => course.id === section.course
      );
      if (_course[0]) {
        section.course = _course[0].subject + " " + _course[0].courseNum;
      }
    }      You, 21 minutes ago • Uncommitted changes
    if (campus.length > 0) {
      const _campus: any = campus.filter(
        (campus: any) => campus.id === section.campus
      );
      if (_campus[0]) {
        section.campus = _campus[0].campusName;
      }
    }
    if (classLocation.length > 0) {
      const _location: any = classLocation.filter(
        (location: any) => location.id === section.location
      );

      const _altLocation: any = classLocation.filter(
        (location: any) => location.id === section.alternateLocation
      );
      if (_location.length > 0) {
        section.location =
          _location[0].buildingName + " " + _location[0].room;
      }

      if (_altLocation.length > 0) {
        section.alternateLocation =
          _altLocation[0].buildingName + " " + _altLocation[0].room;
      }
    }

    if (term.length > 0) {
      const _term: any = term.filter((term: any) => term.id === section.term);
      if (_term.length > 0) {
        section.term = _term[0].semester + " " + _term[0].year;
      }
    }
  });
  setSearchedSection(_sections);
};
```

Figure 20– Code snippet of using map and filter function



Course Subject	Course Subject
616fbcd57b77628cfd0a3598	MATH-1350
616fbcd57b77628cfd0a3598	MATH-1350
6176d834967fd521613098bd	PHYS-1408
6171dddf367d52043345a667	CS-1411
616fc15e7b77628cfd0a3599	ENGL-5301

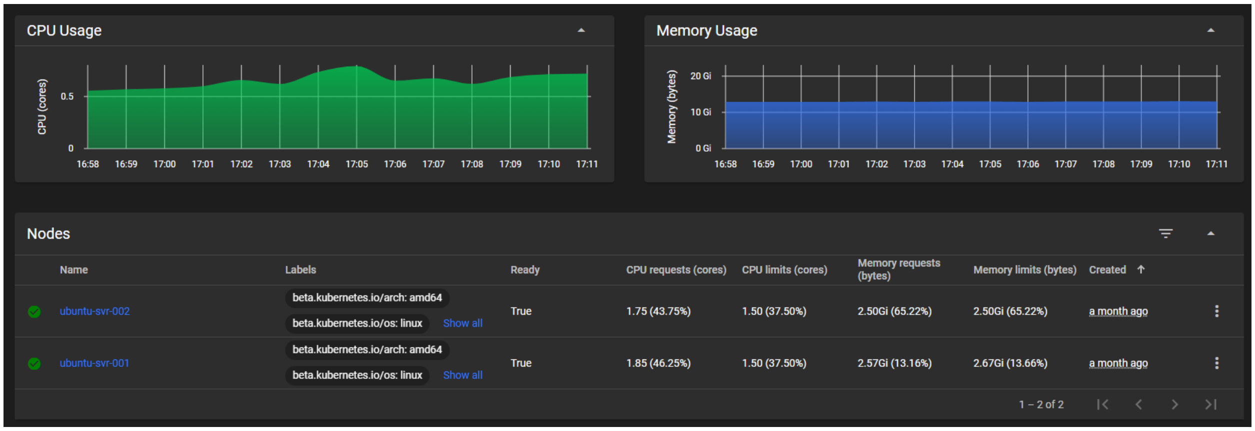
Figure 21- Without filtering the data and with filtering the data

Kubernetes and Microservice Lifecycle – Matt

The Kubernetes configuration for Nexus is a generic one for many large scale MicroK8s deployments. Each of the services are Dockerized using specific build pipelines that are defined in Bitbucket. Once each of these are Dockerized and pushed to Docker Hub, the images can be pulled down into Kubernetes for container orchestration. At the bottom of this section are some images showing the Kubernetes implementation as well as an image showing the successful build pipelines in Bitbucket which are indicated by the green checkmark on the far right of the image. Line items without a checkmark were not setup with build pipelines as they have no need to be dockerized. In the first image it can be seen that both of the nodes are active and ready with their respective CPU and Memory limits.

In the second image it can be seen that each of the Dockerized microservices are running as Pods and are initially scaled to 2 pods per service. When a university's need arises during times of heavy registration for example that particular service can be scaled up according to the number of resources in the Kubernetes cluster. The lifecycle of a Pod is relatively simple. The Pod will run indefinitely unless there is an unrecoverable error inside of that Pod with the host process defined in the Dockerfile. If this is the case, Kubernetes will automatically restart the Pod from its base image. This requires no external help from humans.

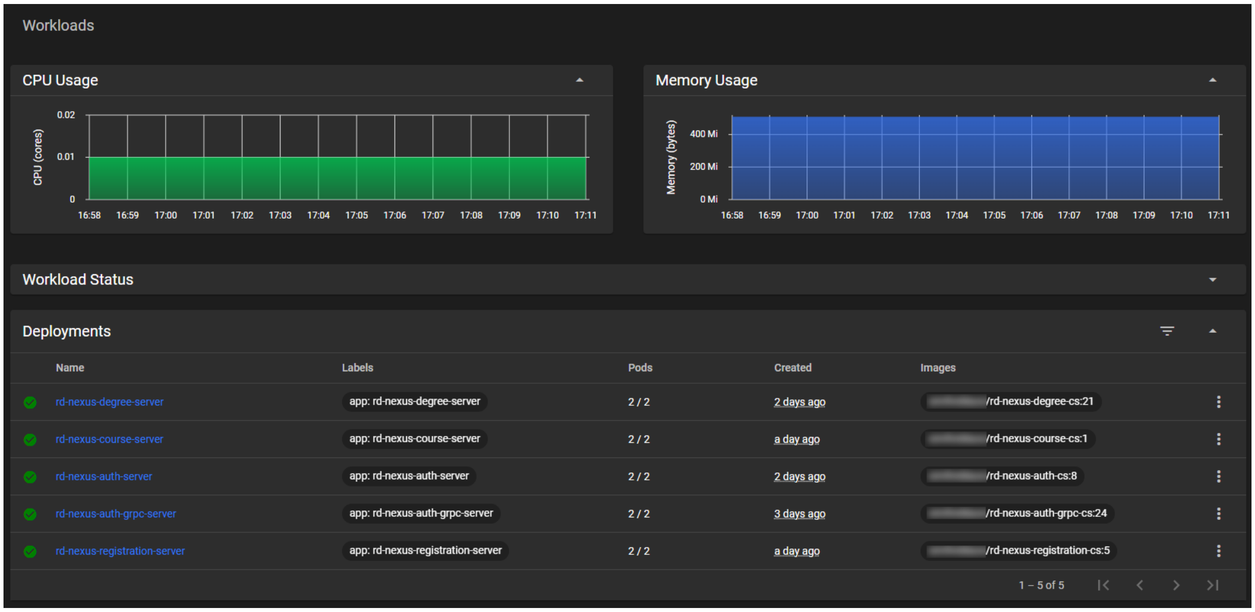
When it comes time to upgrade a node in the cluster or to upgrade a particular microservice, Kubernetes handles it in a very elegant way. If a node needs to be upgraded a command "kubectl drain node *node_name*" can be run to transfer all of the workloads off of that node so it can safely be upgraded or decommissioned without a second of production downtime. The process is somewhat similar for upgrades to deployments. Kubernetes will automatically spin up new pods with the new version and wait for them to begin running and stable before terminating the old pods. This all happens without a second of downtime. In this initial version of Nexus each pod is provisioned with 256 MB of host RAM as well as 300 m (units out of 1000) of CPU time per core. This would be $1000 * core\_count$ total units.



Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Created ↑
✓ ubuntu-svr-002	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux Show all	True	1.75 (43.75%)	1.50 (37.50%)	2.50Gi (65.22%)	2.50Gi (65.22%)	a month ago
✓ ubuntu-svr-001	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux Show all	True	1.85 (46.25%)	1.50 (37.50%)	2.57Gi (13.16%)	2.67Gi (13.66%)	a month ago

1 – 2 of 2



Workloads

CPU Usage

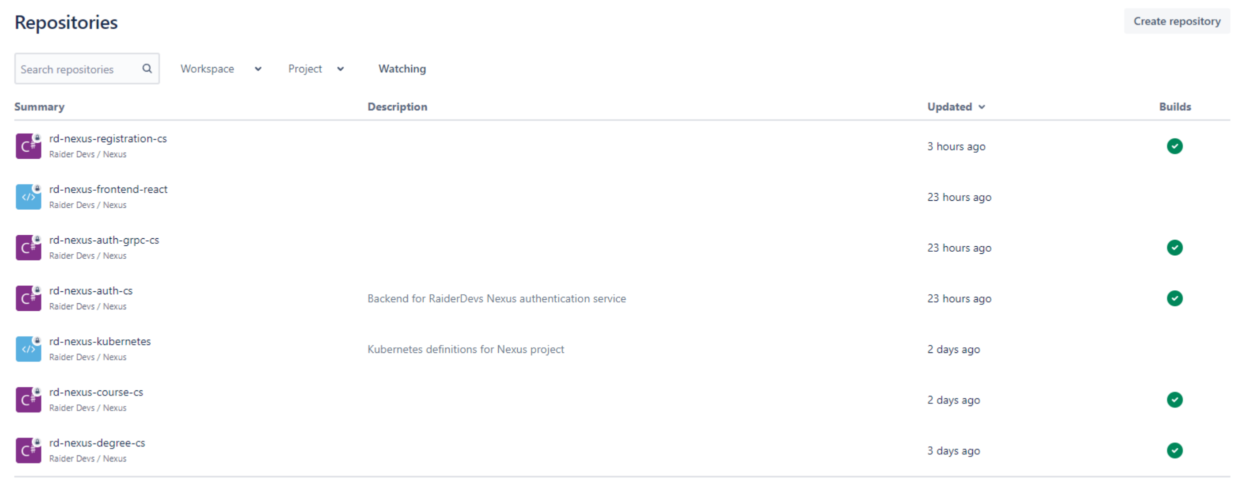
Memory Usage

Workload Status

Deployments

	Name	Labels	Pods	Created	Images
✓	rd-nexus-degree-server	app: rd-nexus-degree-server	2 / 2	2 days ago	[redacted]/rd-nexus-degree-cs:21
✓	rd-nexus-course-server	app: rd-nexus-course-server	2 / 2	a day ago	[redacted]/rd-nexus-course-cs:1
✓	rd-nexus-auth-server	app: rd-nexus-auth-server	2 / 2	2 days ago	[redacted]/rd-nexus-auth-cs:8
✓	rd-nexus-auth-grpc-server	app: rd-nexus-auth-grpc-server	2 / 2	3 days ago	[redacted]/rd-nexus-auth-grpc-cs:24
✓	rd-nexus-registration-server	app: rd-nexus-registration-server	2 / 2	a day ago	[redacted]/rd-nexus-registration-cs:5

1 – 5 of 5

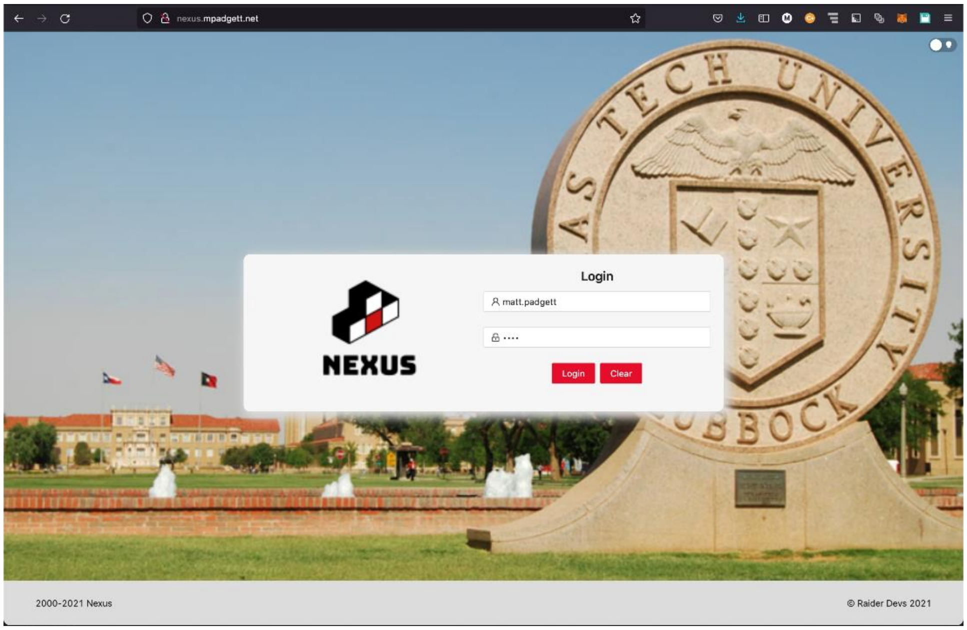


Repositories

Summary	Description	Updated	Builds
rd-nexus-registration-cs Raider Devs / Nexus		3 hours ago	✓
rd-nexus-frontend-react Raider Devs / Nexus		23 hours ago	
rd-nexus-auth-grpc-cs Raider Devs / Nexus		23 hours ago	✓
rd-nexus-auth-cs Raider Devs / Nexus	Backend for RaiderDevs Nexus authentication service	23 hours ago	✓
rd-nexus-kubernetes Raider Devs / Nexus	Kubernetes definitions for Nexus project	2 days ago	
rd-nexus-course-cs Raider Devs / Nexus		2 days ago	✓
rd-nexus-degree-cs Raider Devs / Nexus		3 days ago	✓

Log-in Page – Morgan

We wanted to modernize the designs of the current TTU registration system. To do this, we are making it more mobile-friendly and changing some of the layouts to make it easier to use. Below we will discuss the different pages and why we came to that type of design. We decided to add a dark mode and light mode feature for the desktop/tablet version, making it easier on the eyes for the user. Here below we have our final design for our desktop/tablet light mode login page.



nexus.mpadgett.net

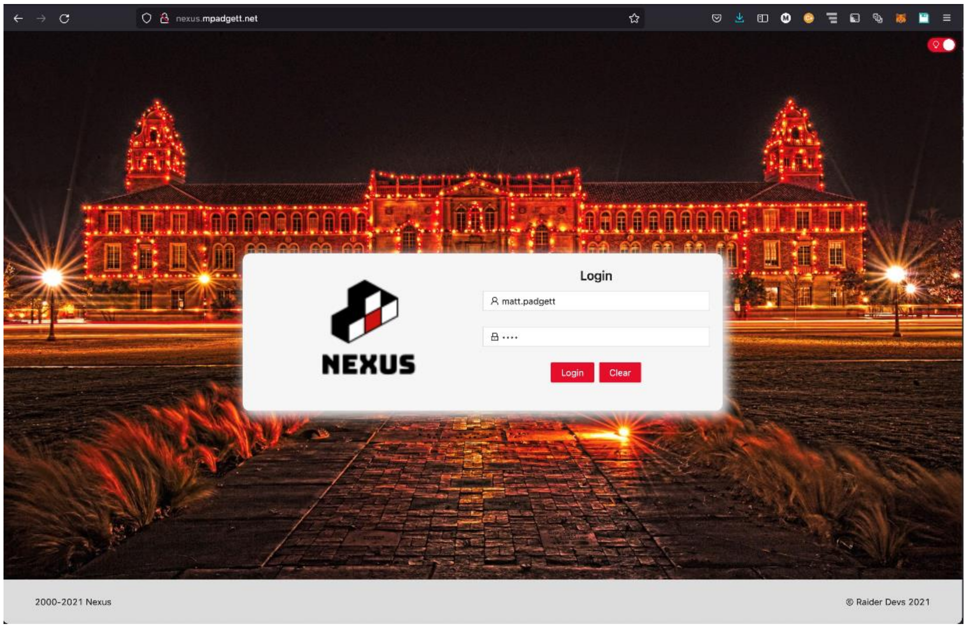
Login

matt.padgett

••••

Figure 16 - Login Page, Light Mode

Here below we have our final design for our desktop/tablet dark mode login page.

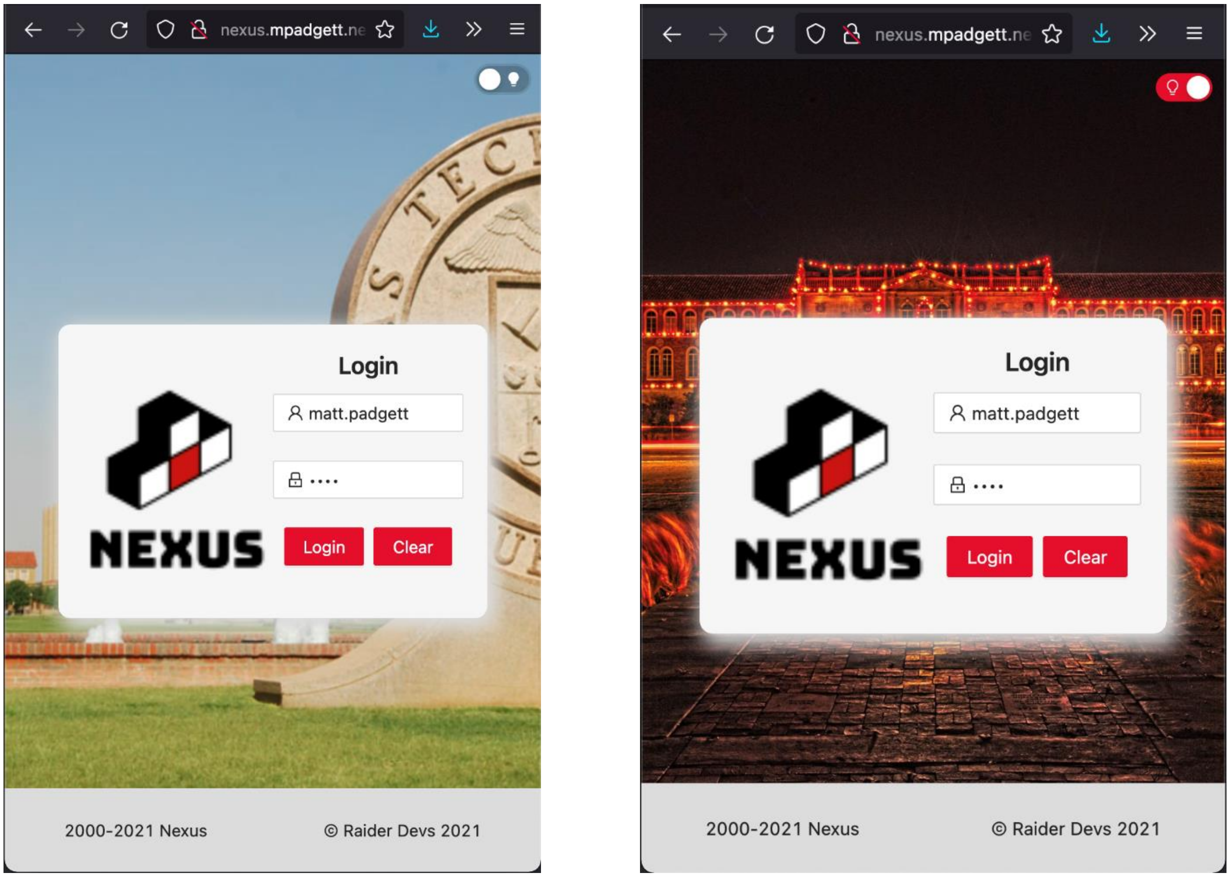


matt.padgett

••••

Figure 17 - Login Page, Dark Mode

Here below we have our final design for our mobile login page.



nexus.mpadgett.ne

Login

matt.padgett

••••

NEXUS

2000-2021 Nexus

© Raider Devs 2021

nexus.mpadgett.ne

Login

matt.padgett

••••

NEXUS

2000-2021 Nexus

© Raider Devs 2021

Figure 18 – Light Mode/Dark Mode Login Page, Mobile

Gantt Chart

Below you can see two versions of our updated Gantt Charts, in the first Gantt Chart we have a general overview of where we are at. As you can see, we have completed the course! Thank you for reading this report!

For the second Gantt chart, you can see a more detailed view of our tasks from our Jira board. As you can see we have completed every task!

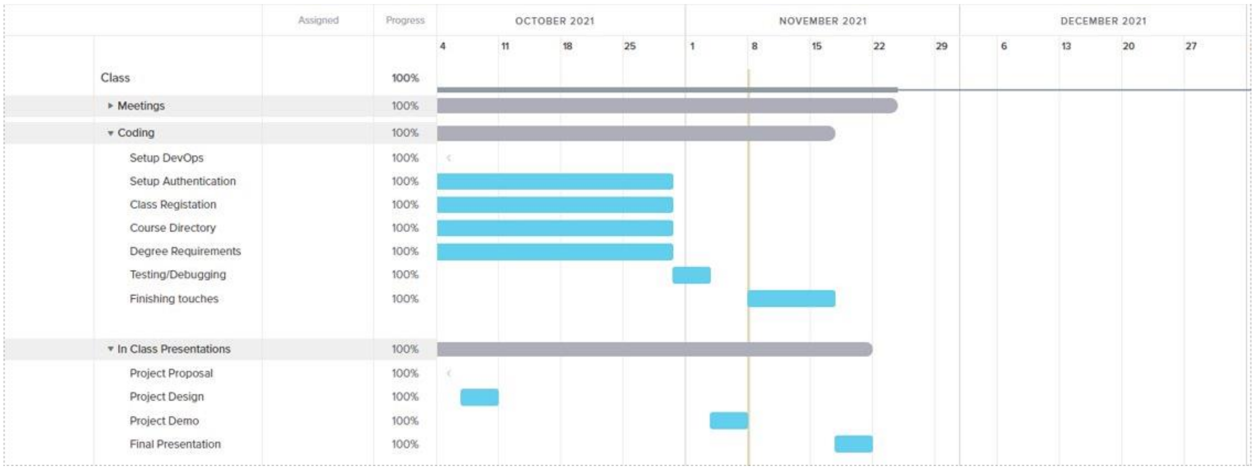


Figure 19 - Gantt 1

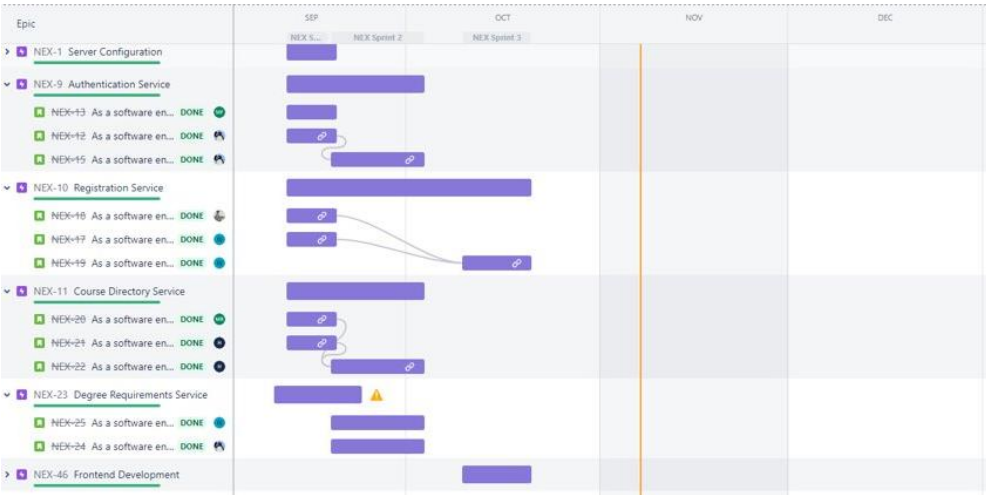
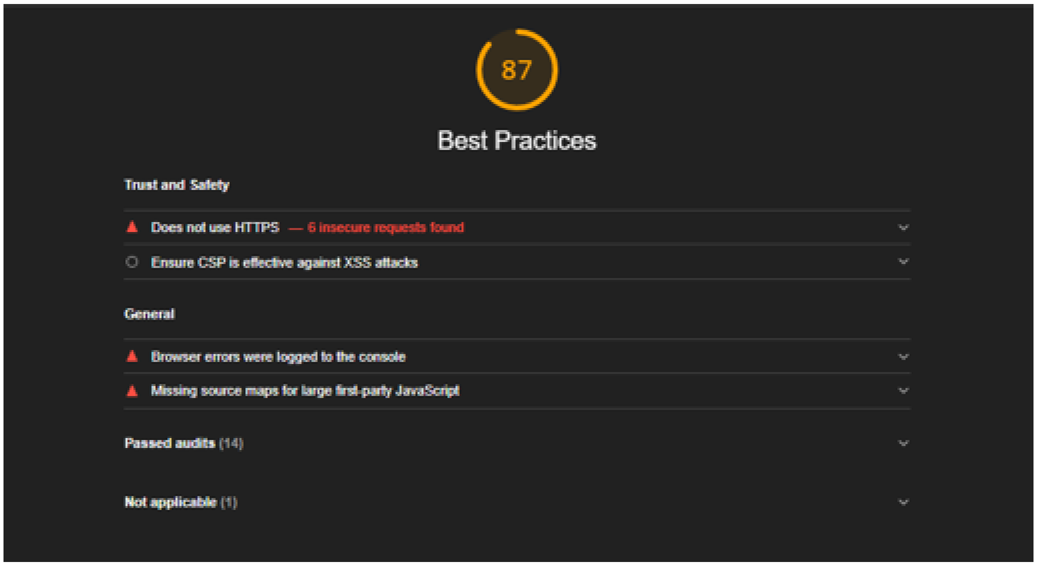


Figure 20 - Gantt

Lighthouse Statistics

Below you can see some of our lighthouse statistics, for performance we have a rating of 74. This is because of minification and some unused JavaScript files. If we were to add in a bundler, our performance would be in the high 90s. For best practices, we have an 87. This is due to not having an SSL/TLS certificate allowing the webpage to be HTTPS. Getting a certificate is pricey and we were not able to buy this during the time of our project. However, if we were to go live this is something we would have invested in and we would a score of 100.



87

Best Practices

Trust and Safety

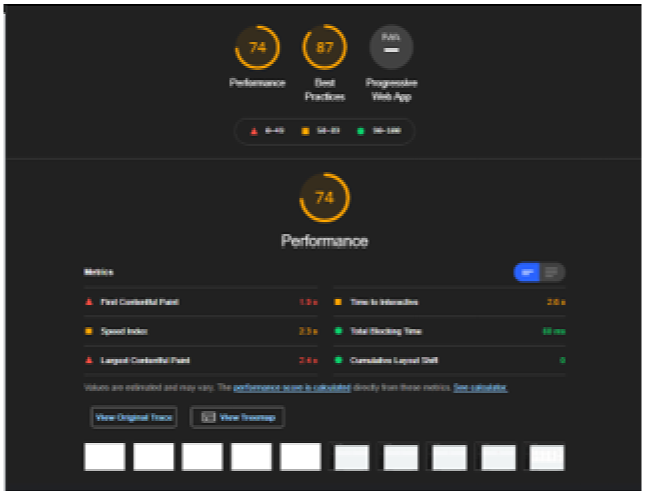
- ▲ Does not use HTTPS — 6 insecure requests found
- ○ Ensure CSP is effective against XSS attacks

General

- ▲ Browser errors were logged to the console
- ▲ Missing source maps for large first-party JavaScript

Passed audits (14)

Not applicable (1)



74 Performance | 87 Best Practices | PWA — Progressive Web App

▲ 0–49 ■ 50–89 ● 90–100

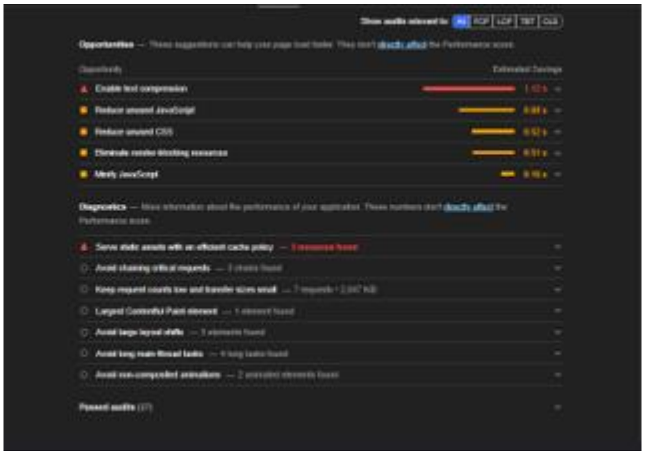
74

Performance

Metrics

▲	First Contentful Paint	1.9 s	■	Time to Interactive	2.6 s
■	Speed Index	2.3 s	●	Total Blocking Time	10 ms
▲	Largest Contentful Paint	2.6 s	●	Cumulative Layout Shift	0

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.



Opportunities — These suggestions can help your page load faster. They don't directly affect the Performance score.

Opportunity	Estimated Savings
▲ Enable text compression	1.12 s
■ Reduce unused JavaScript	0.88 s
■ Reduce unused CSS	0.52 s
■ Eliminate render-blocking resources	0.51 s
■ Minify JavaScript	0.16 s

Diagnostics — More information about the performance of your application. These numbers don't directly affect the Performance score.

- ▲ Serve static assets with an efficient cache policy — 3 resources found
- ○ Avoid chaining critical requests — 3 chains found
- ○ Keep request counts low and transfer sizes small — 7 requests • 2,087 KiB
- ○ Largest Contentful Paint element — 1 element found
- ○ Avoid large layout shifts — 3 elements found
- ○ Avoid long main-thread tasks — 4 long tasks found
- ○ Avoid non-composited animations — 2 animated elements found

Passed audits (37)

References

Document 32021R0876. EUR. (n.d.). Retrieved October 10, 2021, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32021R0876.

Software engineering code - ACM ethics. ACM Ethics - The Official Site of the Association for Computing Machinery's Committee on Professional Ethics. (2018, December 19). Retrieved October 10, 2021, from https://ethics.acm.org/code-of-ethics/software-engineering-code/.

ISO/IEC 27001 - information security management. ISO. (2020, April 3). Retrieved October 10, 2021, from https://www.iso.org/isoiec-27001-information-security.html.

State of Texas and Texas Tech University. (n.d.). *Texas Tech University*. TTU. Retrieved October 10, 2021, from https://www.ttu.edu/webguidelines/standards/.