Will Gillette, Jess Sokolski, Kacey La, Matt Quigley, Ryan Fitzgerald

Dr. Mongan

CS-375: Software Engineering

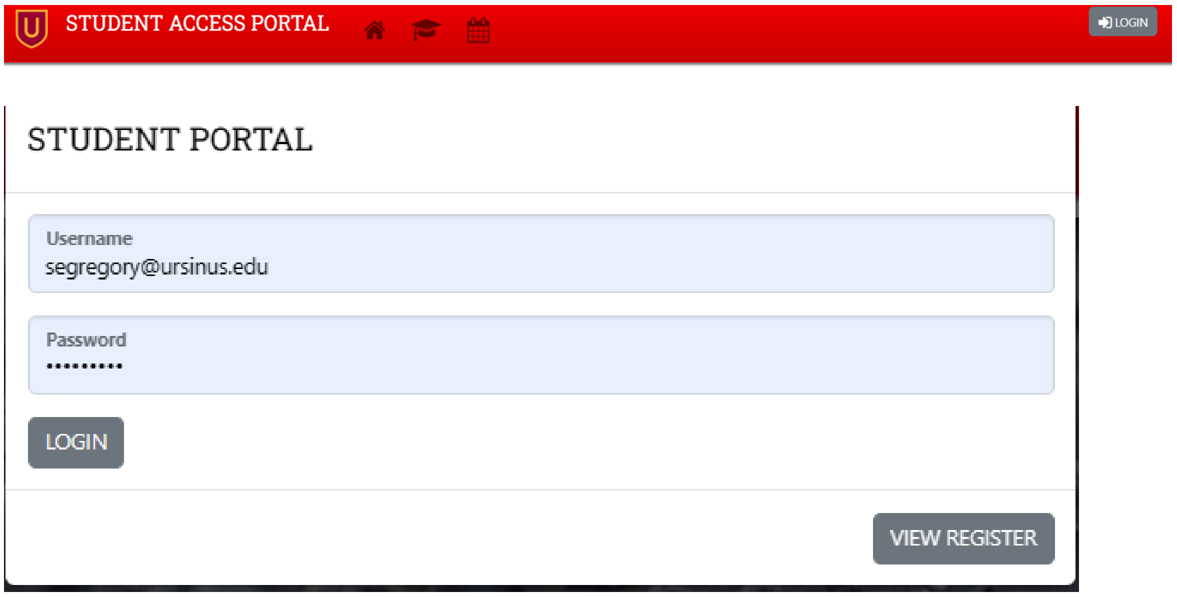
17 April 2023

Software Final Documentation: Academic Planner

USER:

Each section below indicates how users can operate each feature within the degree planner application. The corresponding user requirements are outlined below.

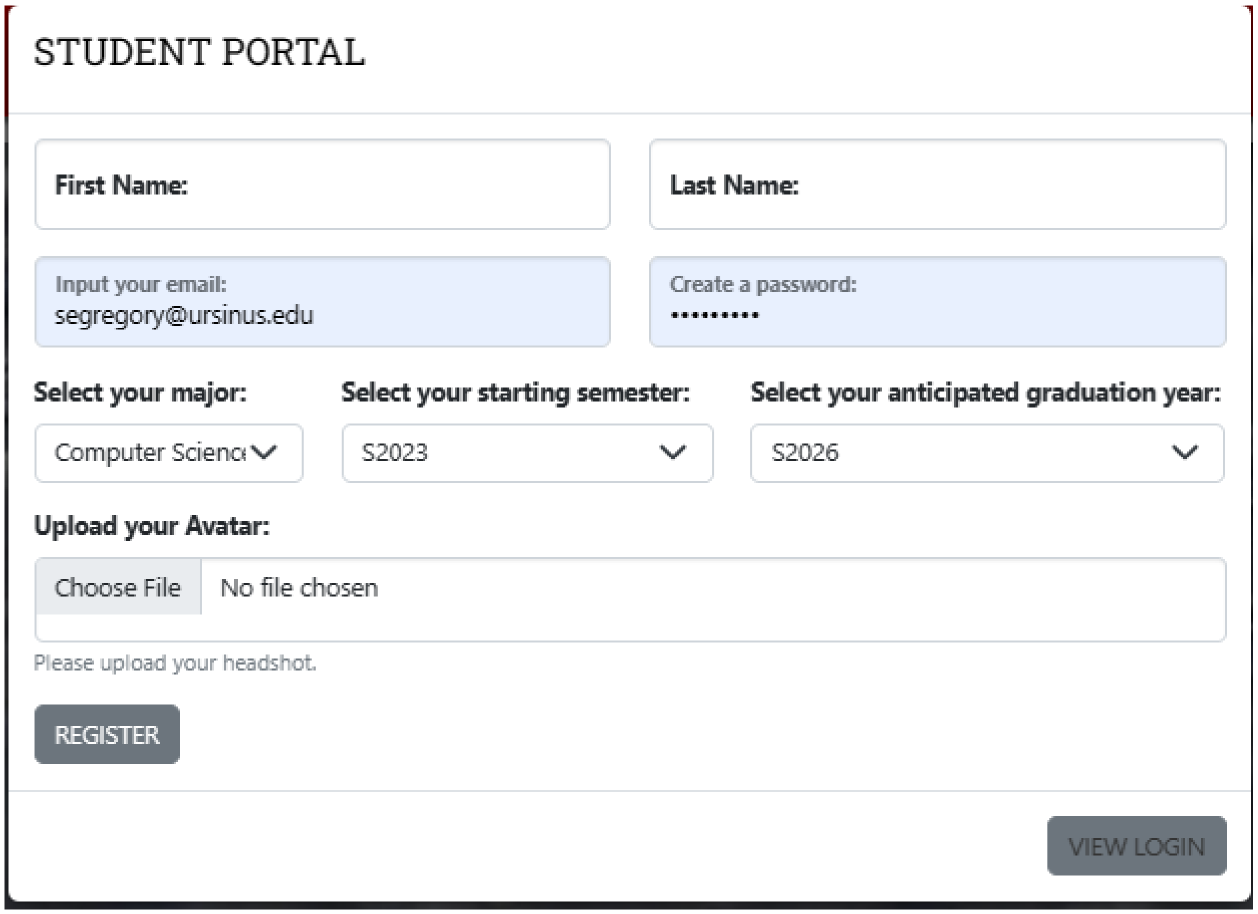
Home Page:



Username: segregory@ursinus.edu
Password: •••••••••

To begin using the Student Access Portal, one can hit the **LOGIN** button in the top right corner of the home page, toggling the **LOGIN** form (Requirement 1.1a). From here, one can enter his/her credentials and then press the **LOGIN** button. Following this, if the login process is successful, the modal will close and allow one to now access the other pages of the application (Requirement 1.2). Otherwise, an alert message will appear on the view indicating the reason why this process has failed. If one has not already registered with the system, they can select

VIEW REGISTER in the bottom right corner of the Student Portal box to toggle the register form (Requirement 1.1a). Following this, one will need to enter all the parameters in the box to register with the system. Passwords must be 6-15 characters long with at least one capital letter and number. Uploaded headshots must either be a PNG, JPG, or JPEG file (Requirement 1.3). After entering input into all the fields, one can press the **REGISTER** button, displaying a feedback alert on the view indicating whether the registration process was successful (Requirement 1.1a). Upon registering successfully, one can return to the login form by selecting the **VIEW LOGIN** button in the bottom right corner and enter the respective credentials. The **LOGOUT** button is in the same position as the **LOGIN** button, and clicking this will sign one out.


STUDENT PORTAL

First Name:

Last Name:

Input your email: segregory@ursinus.edu

Create a password: •••••••••

Select your major: Computer Scienc

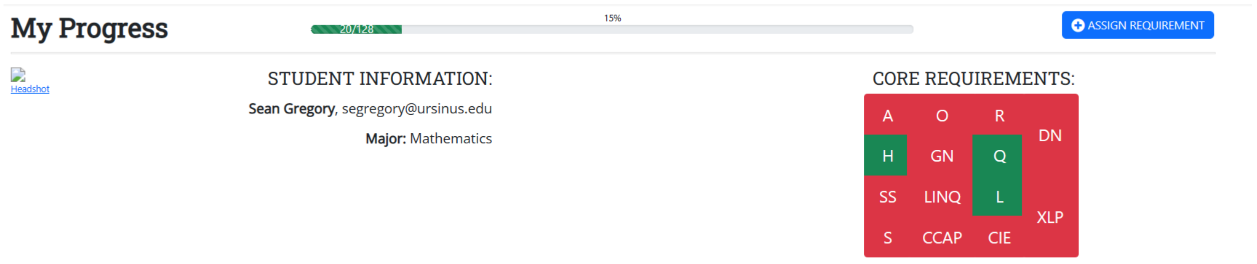
Select your starting semester: S2023

Select your anticipated graduation year: S2026

Upload your Avatar: Choose File No file chosen

Please upload your headshot.

<u>My Progress Page:</u>



STUDENT INFORMATION:

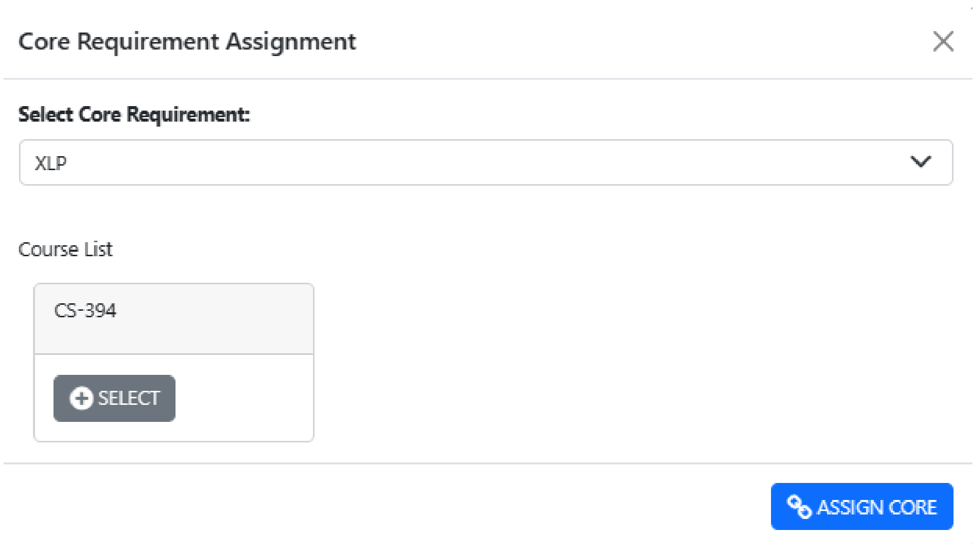
Sean Gregory, segregory@ursinus.edu

Major: Mathematics

CORE REQUIREMENTS:

🟥 A	🟥 O	🟥 R	🟥 DN
🟩 H	🟥 GN	🟩 Q	
🟥 SS	🟥 LINQ	🟩 L	🟥 XLP
🟥 S	🟥 CCAP	🟥 CIE	

Once one has reached the **My Progress** page, one can see which Core Requirements one has completed in the **CORE REQUIREMENTS** box to the right (Requirement 2.2a). If a requirement is green, one has completed it. Otherwise, a red fill indicates that one must assign a planned course to that respective core requirement. At the top of the page, one can see their credits progress bar, displaying the proportion of planned credits compared to the graduation required amount, 128 credits (Requirement 2.3). At the top right, one can press the **ASSIGN REQUIREMENT** button to display the core requirement form, enabling one to assign courses to a core requirement (Requirement 2.2c). Additionally, clicking any of the red filled boxes toggles this form as well. On the other hand, clicking on any of the green filled boxes displays a pop-up notification indicating which course one has assigned to the respective core requirement.


Core Requirement Assignment

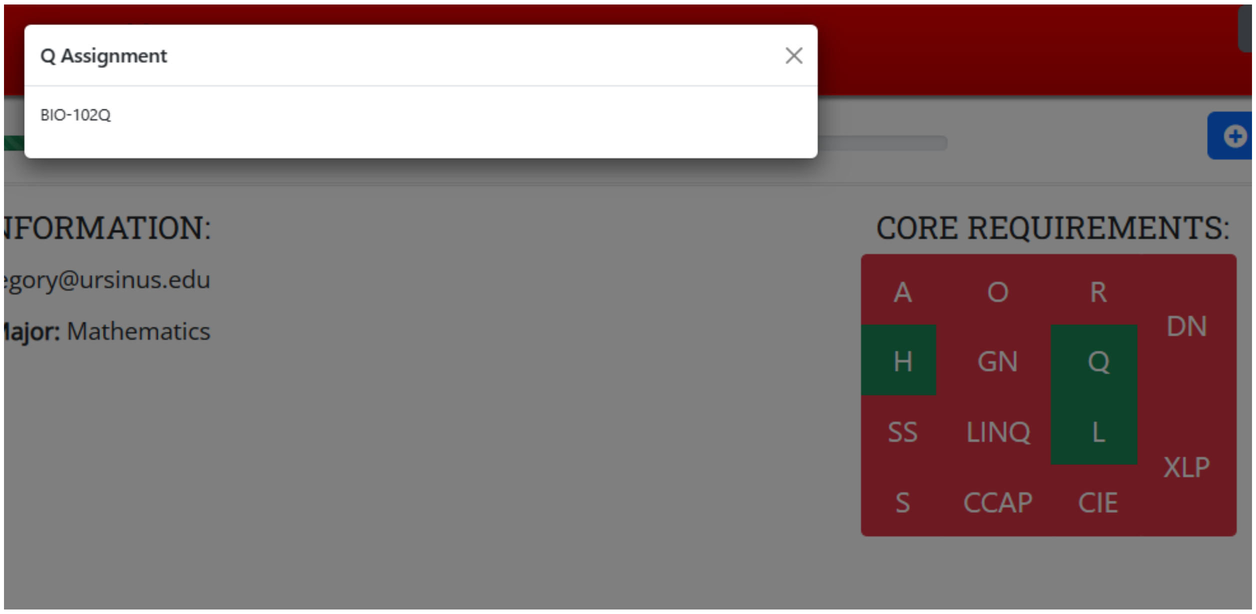
Select Core Requirement:

XLP

Course List

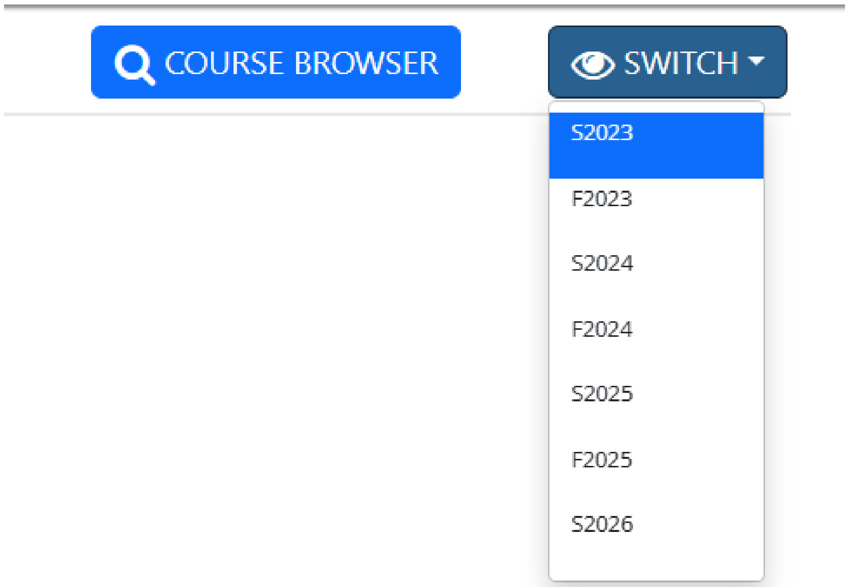
CS-394

The core requirement assign form allows users to choose a requirement from the Select Core Requirement drop down bar (Requirement 2.2c/2.4). After selecting a requirement, the course list will show which planned courses fulfill the chosen requirement, allowing one to assign an eligible planned course to a core requirement. After selecting the course, one can hit **ASSIGN CORE** to assign the course to the respective core requirement and verify that a green fill appears in the core requirements table in that core requirement’s cell. To see which course fulfills a given requirement, one can click on the requirement’s box, displaying a pop-up window with the course’s name.

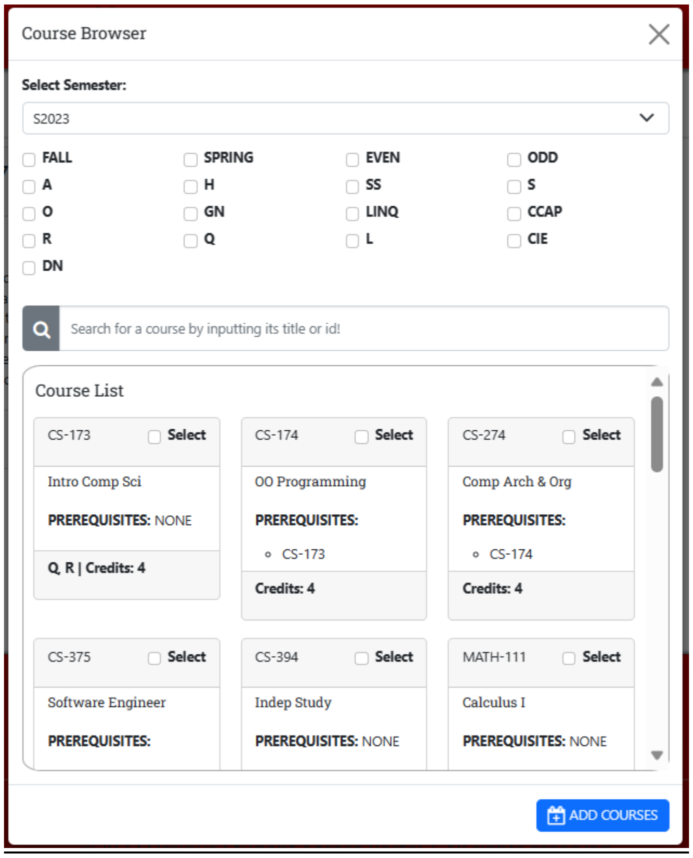


Degree Planning Page:

One can navigate to the degree planning page by clicking on the calendar icon in the header (Requirement 1.4). From this page, one can view his/her degree plan and add and remove courses accordingly (Requirement 3.2a). To view a semester's plan, one can select the **SWITCH** button, showing a dropdown list of available semesters depending on the input starting semester and anticipated graduation date on the registration form (Requirement 3.2c).



To add courses, one can press the **COURSE BROWSER** button to toggle the course planning form (Requirement 3.1). From here, one can filter courses based on whether the semester is in an odd or even year and which core requirements the course will fulfill (Requirement 3.1c). Note that by default, the initial displayed courses are dependent on the selected semester. For instance, selecting the spring semester for an odd year will exclude courses that are only offered in the fall or even years (Requirement 3.1d). After selecting the filters, you can select the courses you want by clicking the box next to Select (Requirement 3.1e). Once you have selected all of the courses you want, hit the **ADD COURSES** button (Requirement 3.3). The courses will now be viewable on your Course Plan under the respective semester.


Course Browser

Select Semester: S2023

☐ FALL ☐ SPRING ☐ EVEN ☐ ODD
☐ A ☐ H ☐ SS ☐ S
☐ O ☐ GN ☐ LINQ ☐ CCAP
☐ R ☐ Q ☐ L ☐ CIE
☐ DN

Search for a course by inputting its title or id!

Course List

CS-173 ☐ Select
Intro Comp Sci
PREREQUISITES: NONE
Q, R | Credits: 4

CS-174 ☐ Select
OO Programming
PREREQUISITES:
- CS-173
Credits: 4

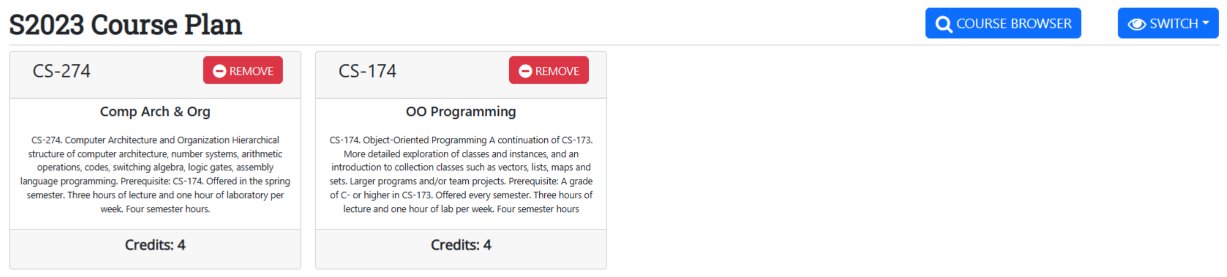
CS-274 ☐ Select
Comp Arch & Org
PREREQUISITES:
- CS-174
Credits: 4

CS-375 ☐ Select
Software Engineer
PREREQUISITES:

CS-394 ☐ Select
Indep Study
PREREQUISITES: NONE

MATH-111 ☐ Select
Calculus I
PREREQUISITES: NONE

Each course will be shown with its name, description, and number of credits. To remove a course, one can press the corresponding **REMOVE** button located in the top right corner of the respective course (Requirement 3.2b/3.3). An alert will appear on the view indicating that the planned course has been removed from the respective plan.



Comp Arch & Org

CS-274. Computer Architecture and Organization Hierarchical structure of computer architecture, number systems, arithmetic operations, codes, switching algebra, logic gates, assembly language programming. Prerequisite: CS-174. Offered in the spring semester. Three hours of lecture and one hour of laboratory per week. Four semester hours.

Credits: 4

OO Programming

CS-174. Object-Oriented Programming A continuation of CS-173. More detailed exploration of classes and instances, and an introduction to collection classes such as vectors, lists, maps and sets. Larger programs and/or team projects. Prerequisite: A grade of C- or higher in CS-173. Offered every semester. Three hours of lecture and one hour of lab per week. Four semester hours

Credits: 4

TECHNICAL:

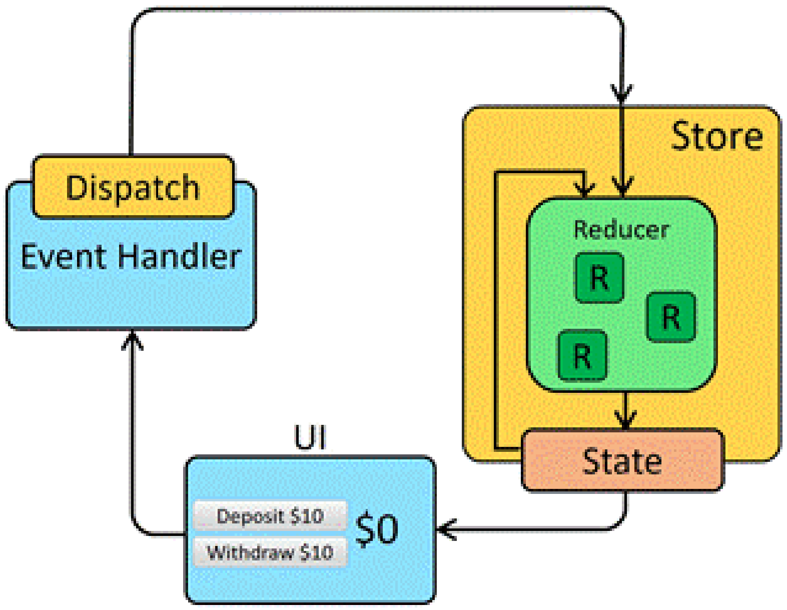
Deployment:

To deploy the system, one must clone the repository from GitHub, located at https://github.com/wigillette/GatewayRevamp and navigate to the local copy via the terminal. To proceed, one must install **Node.js** if not already installed. Following this, one can open a second terminal, also navigating to the local copy. On one terminal, one must run **npm run build**, initializing the view and client side of the application, and installing any necessary client-side packages. On the second terminal, one must run **npm run start**, installing the server-side packages in the application, and deploying the app on PORT 3001 of localhost. Thus, to access the application, one must navigation to http://localhost:3001 in their browser.

Overview:

This application uses a React-Redux model for its client-side functionalities, especially anything pertaining to user interfaces. Each UI entity is called a component, and the client uses a “Redux store” as a global container for state information that each component can access. The store is divided into reducers, which each store information for a respective piece of the application on the front end. To update the Redux store, the client dispatches actions to it, and the store determines which reducer to use when activating that action. Finally, each time the store updates, each component re-renders to reflect the updated information. Figure 1 describes this process.

<u>Figure 1</u>



<u>Client-Side Model:</u>

As elaborated upon in the Design Report, there are components for each of the three pages in the application: the home page, the progress page, and the degree planning page. Each of these pages also has a respective reducer, and we apply the model in Figure 1 to manage events and updates. Some of the information contained in the store includes the user’s degree plan and core requirement assignments fetched from the server. To elaborate, when the progress page loads, the client initially sends a GET request to the server to obtain a dictionary from the database describing what courses the respective user assigned to the core requirements. Similarly, after accessing the degree planning page, the client fetches the list of courses that the user assigned to each semester in their plan. The list of available semesters is determined by the starting semester and anticipated graduation semester that the user inputted in their registration form. As described in the user section, there are forms for adding courses to the degree plan and assigning courses to core requirements, each of which send POST requests to the server upon submission. Upon receiving a response from the server, either an alert appears indicating an error or the view updates reflecting the changes requested on the form. Essentially, the client side leverages the model in Figure 1 to easily update information on the view and send information across all the pages in the application.

To handle user authentication, upon logging in, the server generates a JSON web token, storing it in the client. When a user performs certain actions, such as removing or adding a course in the degree plan, the server validates the client's access token by determining whether it has expired.

Server-Side Model:

The server side of the application contains folders for Models, Routes, Middleware, and Controllers. The routes folder contains JavaScript modules that initialize the API endpoints for the various GET and POST requests in the application, while the controllers folder contains the implementation of the functions that correspond to these endpoints. Finally, the models folder contains the "main" and "test" database files that the controllers access when making database requests. After performing database queries, the controllers then send a response to the client indicating whether the process was successful and any necessary information. For example, when the user assigns a course to a core requirement, the server responds to the client with the updated assignments list, allowing the client to dispatch an update to the store, re-rendering the respective components. The middleware folder validates that the user's JSON web token has not expired, determining whether they can perform certain actions such as updating the list of core requirement assignments or changing the degree plan.

The controllers are the meat of the functionality of the server side. Thus, there are various validation tests that the server conducts prior to performing database queries. For each controller function, the server validates that the parameters are in their respective format. For instance, the register function must validate that the password, email, and headshot are in their proper respective formats as described in the user section. Additionally, when updating the degree plan, the server must verify that the semester key is in the correct format (S2022, F2021, etc.) and that it is a part of the list of valid semesters for the user (by comparing the user's start date and anticipated graduation date from the database). The registration controller also must utilize database queries for tests that determine whether the inputted email is unique (there is not an existing account in the database), the inputted major is in a list of available majors, and the inputted start and graduation semesters are in a list of available semesters. If any of the validation tests fail, the server responds to the client with an error message describing the reasoning behind this, which the client then displays on the view of the respective page. Thus, the server takes extreme measures to ensure that erroneous input does not cause any updates to the database, creating a fluid experience for the user. Beyond these validation checks, the application also utilizes the Jest library to conduct unit tests to verify that corner cases or faulty input do not cause errors.

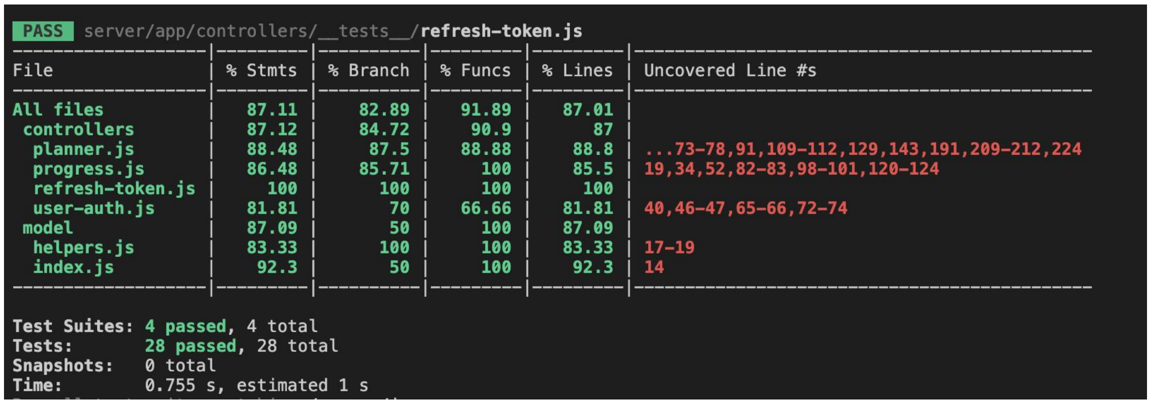
Code Review Improvements:

On April 26th, 2023, our class conducted a code review to provide technical suggestions and feedback. A few suggestions about our application include caching some of the results of database queries to enhance performance. Additionally, our API routes are action words as opposed to nouns. It would make more logical sense to use nouns because the type of request indicates the type of action being conducted. Finally, some aspects of our application code lacked documentation, making it challenging for outside developers to immediately understand the structure and format of the variables in our code. In the future, we hope to implement these changes to enhance our application and improve readability.

Testing/Code Coverage Report:

In this application, Jest is the unit testing framework for the backend. Each module consists of functions that ran the SQL queries or helper functions that helped process and aggregate the results. The helper functions are deterministic and easy to test, but the SQL proves difficult since the queries have the side effect of modifying the database and affecting future tests. To resolve this, the team copied the initialized database and ran tests against the duplicate. Figure 2 displays the results of the tests and the total code coverage.

Figure 2

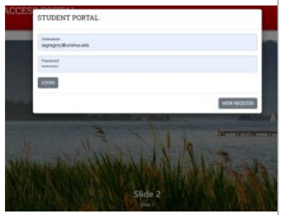
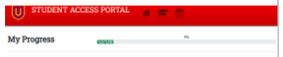


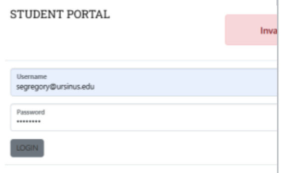
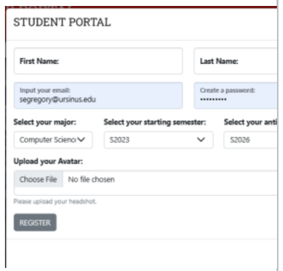
```
PASS  server/app/controllers/__tests__/refresh-token.js
--------------------|---------|----------|---------|---------|-------------------------------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------------------------------
All files           |   87.11 |    82.89 |   91.89 |   87.01 |
 controllers        |   87.12 |    84.72 |    90.9 |      87 |
  planner.js        |   88.48 |     87.5 |   88.88 |    88.8 | ...73-78,91,109-112,129,143,191,209-212,224
  progress.js       |   86.48 |    85.71 |     100 |    85.5 | 19,34,52,82-83,98-101,120-124
  refresh-token.js  |     100 |      100 |     100 |     100 |
  user-auth.js      |   81.81 |       70 |   66.66 |   81.81 | 40,46-47,65-66,72-74
 model              |   87.09 |       50 |     100 |   87.09 |
  helpers.js        |   83.33 |      100 |     100 |   83.33 | 17-19
  index.js          |    92.3 |       50 |     100 |    92.3 | 14
--------------------|---------|----------|---------|---------|-------------------------------------------

Test Suites: 4 passed, 4 total
Tests:       28 passed, 28 total
Snapshots:   0 total
Time:        0.755 s, estimated 1 s
```

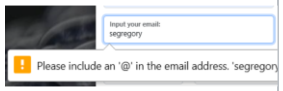
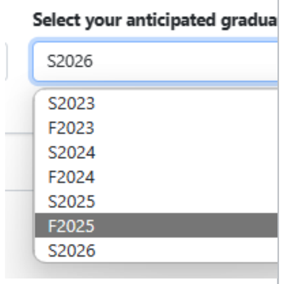
The team achieved a total of 87% on the code coverage report. The uncovered lines belong to the "catch" functions. Unfortunately, the team had trouble testing this because it involves intentionally causing the database to throw an error, an arduous task with the current setup. One potential solution is to use Jest's spy or mock functionality to override what the database returns. However, the team was unable to get the desired functionality from Jest.

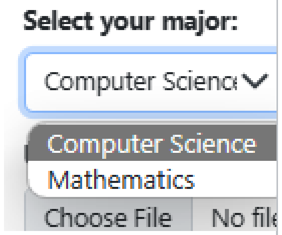
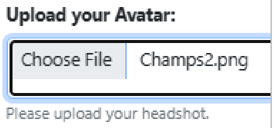
User Acceptance Test Results

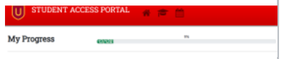
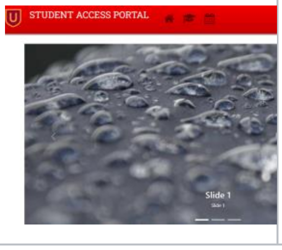
Below are the results of our user acceptance tests, as conducted by our group members:

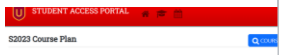
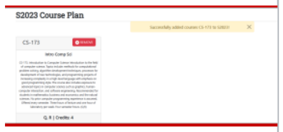
Test Number	Task	Expected Output	Actual Output	Requirement ID
Precondition: Login Form	Click the login button on the right side of the header (located at the top of the page) and be directed to the login form popup.	Ensure that the login form appears with email and password input fields.	 STUDENT PORTAL Username Password LOGIN VIEW REGISTER Slide 2	**1.1a:** Login Form
Login Success	Enter your respective email and password and press the sign-in button located at the bottom	Gain access to the protected routes of the application: the degree planner and my progress page. The login button		**1.1a:** Login Form

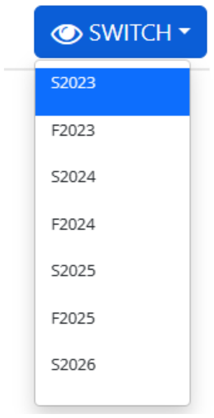
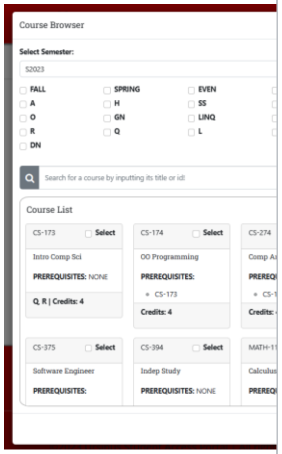
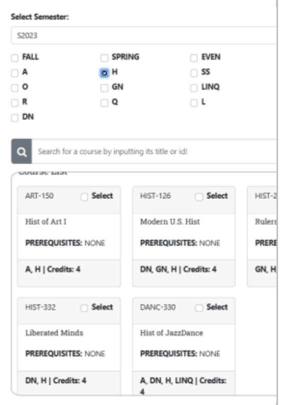
	right of the screen.	should now update to say "logout"		
Login Fail	Follow the same steps as "login success" but enter an invalid email or password on the login form.	**Receive any of these feedback messages:** "Invalid password", "Database not initialized"	 STUDENT PORTAL Inva Username segregory@ursinus.edu Password *********	**1.1a:** **Login Form**
Logout	Press the logout button located to the right of the header.	Verify that you are redirected to the home page and can no longer access the "my progress" and "degree planner" pages.	 Redirected to Home Page upon clicking	**1.2:** Token System
Preconditi on: Registrati on Form	If not registered, click the "login button" on the right side of the header and	View the form for user registration with the "first name", "last name", "anticipated graduation	 STUDENT PORTAL First Name: Last Name: Input your email: segregory@ursinus.edu Create a password: ********* Select your major: Computer Scienc Select your starting semester: S2023 Select your anti S2026 Upload your Avatar: Choose File No file chosen Please upload your headshot.	**1.1a:** Registration Form

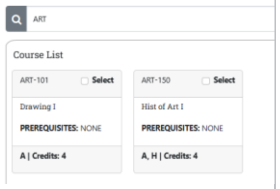
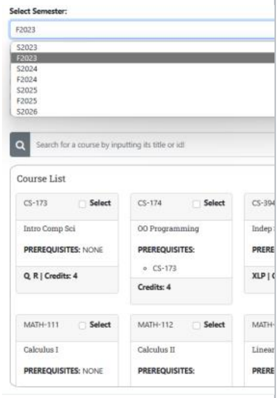
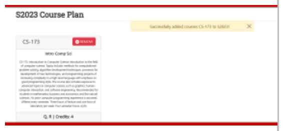
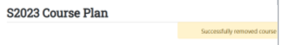
	click the "create account" button in the bottom right corner of the registration form.	date", "major", and "headshot" fields.		
Email/Password	In the email slot, please enter a valid email. In the password slot, please enter a password at least eight characters long with at least one capital letter.	Receive an alert if there are invalid formats in your email and/or password input	 Input your email: segregory Please include an '@' in the email address. 'segregory	**1.3:** Register Validation
Anticipated Graduation Date	Select a graduation year from the respective dropdown menu.	The dropdown updates to indicate your selection.	 Select your anticipated gradua S2026 S2023 F2023 S2024 F2024 S2025 F2025 S2026	**1.3:** Register Validation

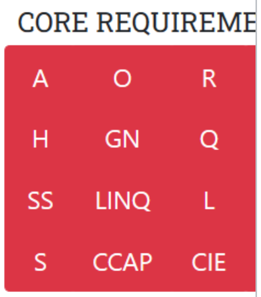
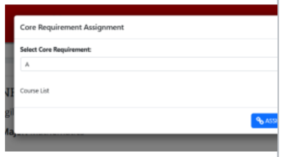
Major	Select a major from the dropdown menu	The dropdown updates to indicate your selection.	 Select your major: Computer Scienc Computer Science Mathematics Choose File No fil	**1.3:** Register Validation
Headshot	Upload a photo of your headshot in either a PNG, JPG, or JPEG format.	Ensure the form displays the file name. Receive an alert if there is an invalid file format.	 Upload your Avatar: Choose File Champs2.png Please upload your headshot.	**1.3:** Register Validation
Create Account	After all selections are made, press the register button located in the bottom right of the screen.	Receive a feedback message indicating whether your registration process is successful. If so, you will now be able to return to the sign-in form and input your new credentials. If not, you will receive an error message indicating the problem,		**1.3:** Register Validation

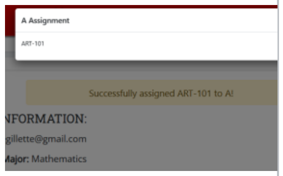
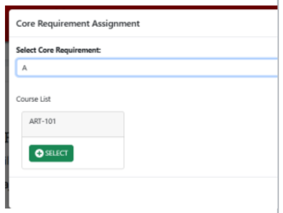
		such as, “Account with that email already exists”, “Database not initialized”, or “Invalid <field> format.”		
Navigation	Access the different pages of the application from the header located at the very top of the page.	**Precondition**: Sign in and gain access to the course planner and my progress pages. If not logged in, be redirected to the home page when attempting to access those pages.		**1.4:** **React Router**
1.1 Navigation	Click the graduation cap icon	Access the “my progress” page if logged in.	 My Progress	**1.4:** Progress Navigation
1.2 Navigation	Click the home icon	Redirection to the landing page of the application.		**1.4:** Home Navigation

1.3 Navigation	Click the calendar icon	Access the “degree planner” page if logged in.		**1.4:** Planner Navigation
Precondition: Degree Planner	Complete Test #1.3 first.	Verify that you see a label at the top left of the page indicating your first eligible semester along with its respective course plan in the center of the page. Ensure that you see an “Add Courses” button and “Switch Semester” dropdown on the right of the page.	Successfully added courses CS-173 to S2023! S2023 Course Plan CS-173 REMOVE Intro Comp Sci Q. 8 | Credits: 4	**3.2a:** Fetch Plans

2. View a different semester's plan	Access the dropdown menu located in the top right corner of the page and select from a list of eligible semesters.	You should see your respective plan for the chosen term or an error message indicating that there was an issue accessing the information for this term.	 SWITCH S2023 F2023 S2024 F2024 S2025 F2025 S2026	**3.2c:** Semester Plan Dropdown
Precondition: Course Browser Modal	Navigate to the degree planner. Click the Course Browser icon located in the top right corner of the screen	View the course browser popup form. Verify that the first fifty courses of the course catalog appear in the search results box at the bottom of the form.	Course Browser Select Semester: S2023 ☐ FALL ☐ SPRING ☐ EVEN ☐ A ☐ H ☐ SS ☐ O ☐ GN ☐ LINQ ☐ R ☐ Q ☐ L ☐ DN Search for a course by inputting its title or id! Course List CS-173 ☐ Select Intro Comp Sci PREREQUISITES: NONE Q, R | Credits: 4 CS-174 ☐ Select OO Programming PREREQUISITES: • CS-173 Credits: 4 CS-274 Comp A PREREQ • CS-1 Credits: CS-375 ☐ Select Software Engineer PREREQUISITES: CS-394 ☐ Select Indep Study PREREQUISITES: NONE MATH-11 Calculus PREREQ	**3.1:** Course Browser Form
3. Course Filters	Toggle the various "core requirement" and "semester offered" filters	Verify that the course browser results updates so that all the courses that match those filters appear.	Select Semester: S2023 ☐ FALL ☐ SPRING ☐ EVEN ☐ A ☑ H ☐ SS ☐ O ☐ GN ☐ LINQ ☐ R ☐ Q ☐ L ☐ DN Search for a course by inputting its title or id! ART-150 ☐ Select Hist of Art I PREREQUISITES: NONE A, H | Credits: 4 HIST-126 ☐ Select Modern U.S. Hist PREREQUISITES: NONE DN, GN, H | Credits: 4 HIST-2 Rulers PRERE GN, H HIST-332 ☐ Select Liberated Minds PREREQUISITES: NONE DN, H | Credits: 4 DANC-330 ☐ Select Hist of JazzDance PREREQUISITES: NONE A, DN, H, LINQ | Credits: 4	**3.1c:** Course Filters

4. Search Course	Search by course ID or name and press the magnifying glass button to the left of the text box.	Ensure that the respective courses that fit your search query appear.	 ART Course List ART-101 Select Drawing I **PREREQUISITES:** NONE **A | Credits: 4** ART-150 Select Hist of Art I **PREREQUISITES:** NONE **A, H | Credits: 4**	**3.1a:** Course Search Bar
5. Switch Semester	From the switch semester dropdown menu, select the semester in which you wish to add the courses	The dropdown should update to indicate the new selection	 **Select Semester:** F2023 S2023 F2023 S2024 F2024 S2025 F2025 S2026 Search for a course by inputting its title or id! Course List CS-173 Select Intro Comp Sci **PREREQUISITES:** NONE **Q, R | Credits: 4** CS-174 Select OO Programming **PREREQUISITES:** ◦ CS-173 **Credits: 4** CS-394 Indep **PRERE** **XLP | C** MATH-111 Select Calculus I **PREREQUISITES:** NONE MATH-112 Select Calculus II **PREREQUISITES:** MATH- Linear **PRERE**	**3.1d:** Course Selection Semester Dropdown
6. Add Courses	Press the add courses button located at the bottom right of the form.	Follow test #2 and ensure that the respective semester's plan has updated.	 S2023 Course Plan Successfully added courses CS-173 to S2023! CS-173 REMOVE Intro Comp Sci [illegible] Q, R | Credits: 4	**3.1e:** Browser Form Submission Button **3.3:** API Endpoint
7. Remove Course	Follow test #2 to navigate to the semester's plan where the respective course is located.	The respective semester's plan should update so that you no longer see the container for the	 S2023 Course Plan Successfully removed course	**3.2b:** Course Removal Button **3.3:** API Endpoint

	Press the remove button located in the top right of the course's information container.	respective course.		
Precondition: My Progress Page	Follow test #1.1 to navigate to the "my progress" page.	Verify that there is a core requirements table located in the center of this page.	 CORE REQUIREME A O R H GN Q SS LINQ L S CCAP CIE	**2.1:** Fetch Assignments
8. Red Requirements	Click on cells in the core requirements table that are filled red.	Verify that a form pops up with a core requirement dropdown menu and a list of eligible courses from your plan that fulfill that requirement. Ensure that the default value of the dropdown menu is the name of the respective	 Core Requirement Assignment Select Core Requirement: A Course List	**2.2a:** Core Fulfillment

		core requirement that you pressed.		
9. Green Requirements	Click on cells in the core requirements table that are filled green.	A popup notification should display on the screen indicating which course fulfills this requirement.	 A Assignment ART-101 Successfully assigned ART-101 to A! NFORMATION: gillette@gmail.com Major: Mathematics	**2.2a:** Core Fulfillment
10. Requirement Assignment Form	Click the requirement assignment form toggle button located in the top right corner of the page.	Verify that you have the same expected output as test #8, with no default value of the core requirement dropdown menu.	 Core Requirement Assignment Select Core Requirement: A Course List ART-101 SELECT	**2.2c/2.4:** Core Assignments
11. Credits Progress Bar	At the top left corner of the screen, there should be a progress bar indicating how many credits you	Verify that the computed number of credits is consisted with your degree plan and the proportion of	3%	**2.3:** Progress Bar

	have mapped out compared to the minimum number needed to graduate (128).	the computed number out of 128 looks correct visually on the progress bar.		