Project: Student Course Advising Website – **Milestone 3**

Student name: Brendan Hearrell

Student UIN: 01219737

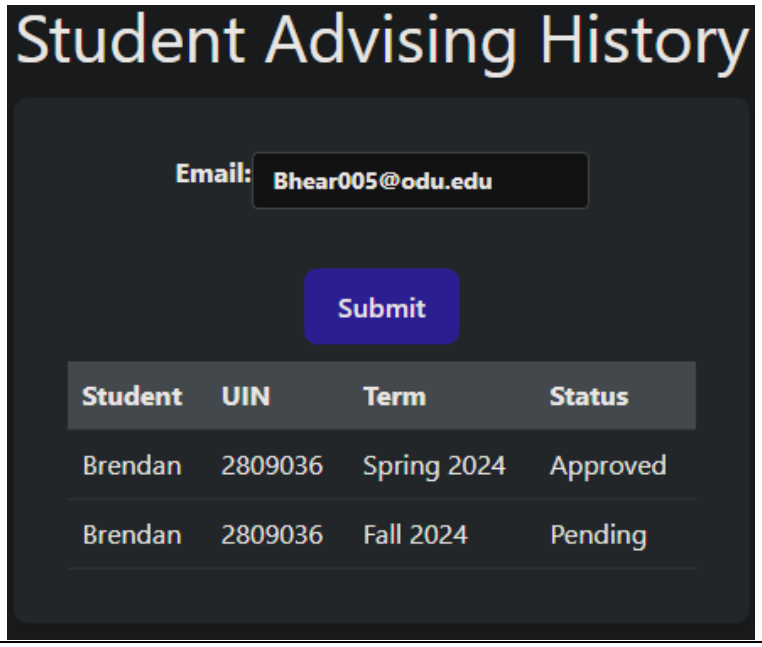
1. Overview

The website is an advising tool for students attending college. The first portion of the project covers account creation, email validation, user login, two factor authentication, account editing, and differentiating between normal user accounts and administrator accounts. The language used for the project will be JavaScript which will utilize the React library and Vite for local development. Node.js will be used as the runtime environment for JavaScript and Express.js will be used for the back-end framework.

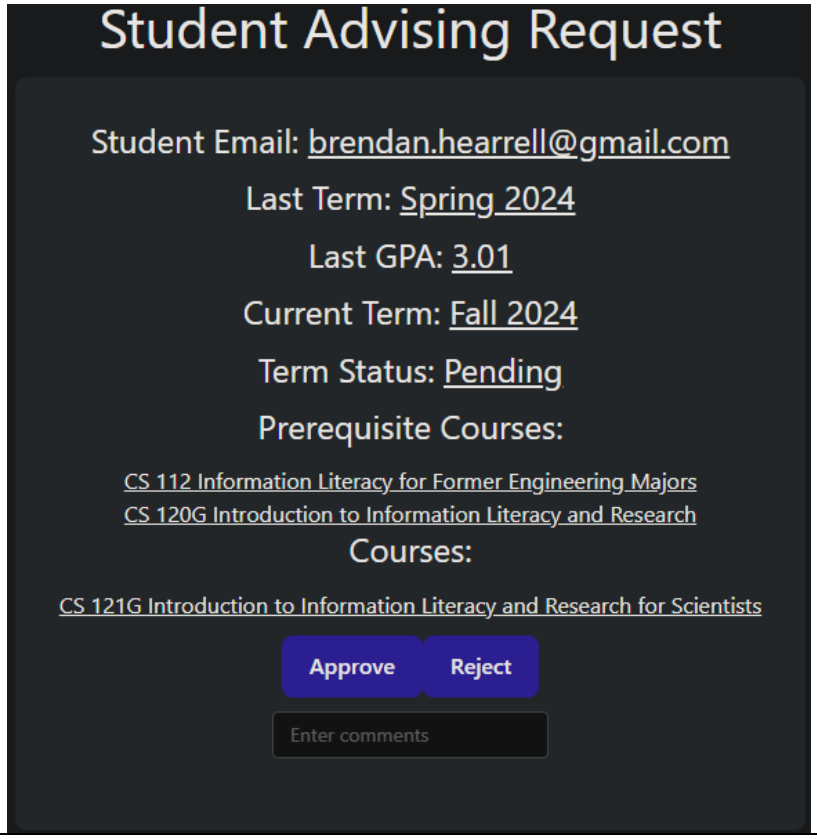
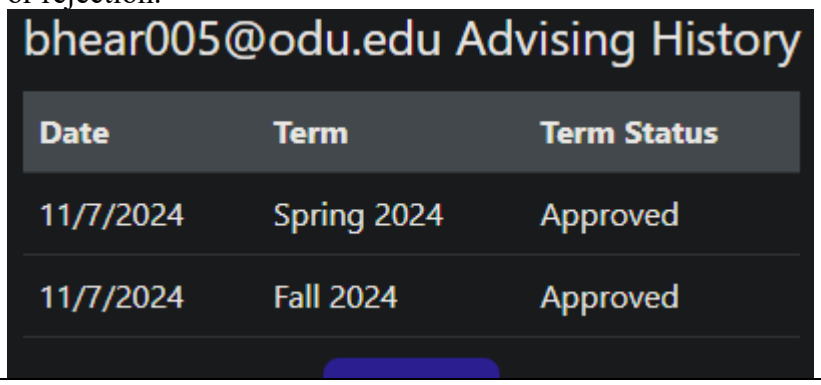
The third portion of the project introduces the course advising features for admin to approve or reject student advising submissions. The admin has access to a search bar where they can enter a student email address, and all advising history will be listed. They can then either accept or reject the request along with a message to the student on whether their choices were appropriate or instructions on what to change. The student will then be able to view their advising history with the status of their approval or rejection. Also, in this milestone reCAPTCHA and a password rule (At least 8 characters, mix of lowercase and uppercase letters, a mix of numbers and special characters) have been added to the registration and login portions of the site. A favicon has been added to the website tab. Test cases have been added to the backend of the application to test API endpoints.

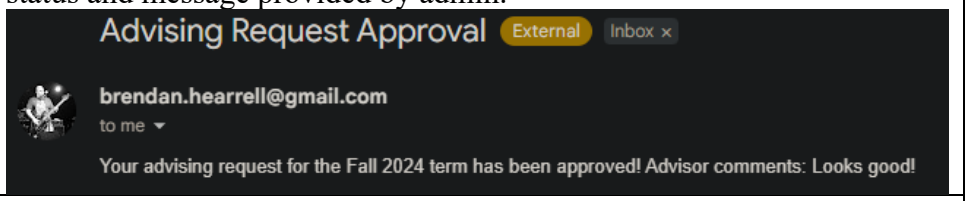
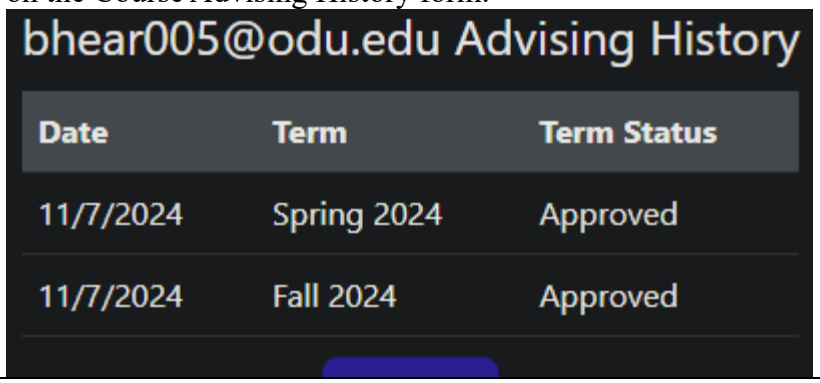
[Page intentionally left blank.]

2. Milestone Accomplishments

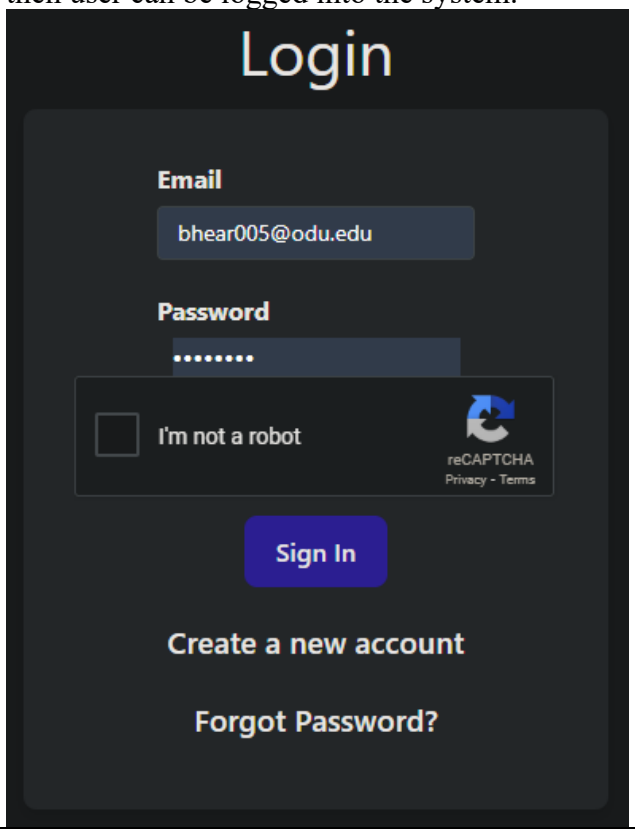
Fulfilled	Feature	Specification
Yes	1	Develop a screen to display advising sheets submitted by CS department students. 

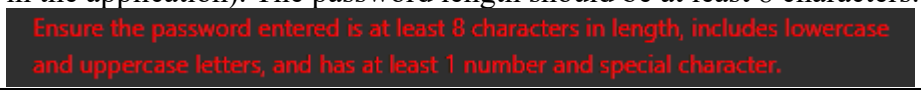
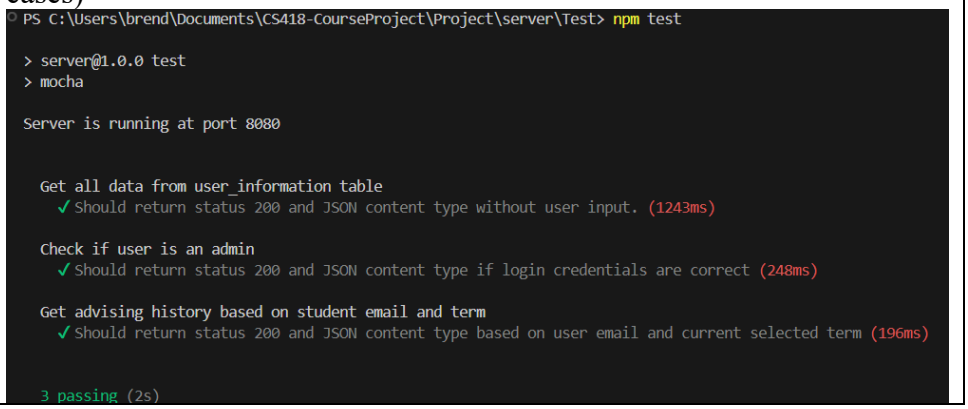
[Page intentionally left blank.]

Yes	2	Clicking on a student's name will redirect to a page displaying the student-submitted record. On this page, there will be options to approve or reject the record. When the admin submits their decision, they must also provide a text message wit their feedback on the advising sheet. After clicking the submit button, the system will redirect to the advising sheet form (as described in point 1), and the new status of the student's records upon submission of approval or rejection. 
Yes	3	Implement status update of student records upon submission of approval or rejection. 

Yes	4	Upon submission, student will receive an email where they can see the status and message provided by admin. 
Yes	5	Now student will be able to see the update status of their advising sheet on the Course Advising History form. 

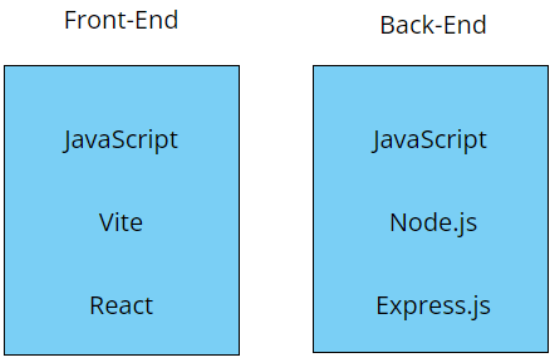
[Page intentionally left blank.]

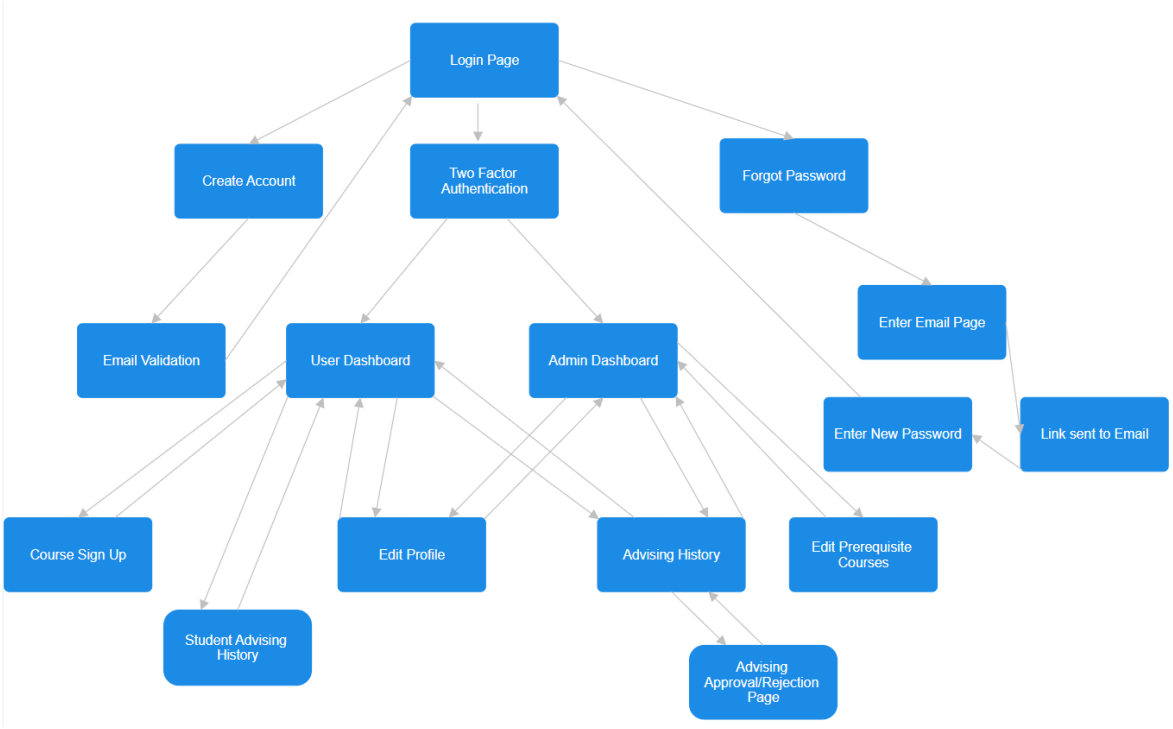
Yes	6	Add reCAPTCHA to the login page. Verify the reCAPTCHA is verified then user can be logged into the system. 
Yes	7	Prevent your application from clickjacking attack. Implement the prevention of click jacking. Show the clickjacking prevention by using <iframe> in .html form. `<iframe src="https://advising-portal-be407.web.app/" width="500" height="500"> </iframe>` 

Yes	8	Add a favicon to the website. 
Yes	9	Add a password rule requiring a mix of capital letters, lowercase letters, special characters, and numbers (implement regex for all password fields in the application). The password length should be at least 8 characters. 
Yes	10	Create test cases and execute in you BE application (Create at least 3 test cases) 

[Page intentionally left blank.]

3. Architecture





[Page intentionally left blank.]

4. Database Design

The Classes table contains all listed courses including fields; ID, Course_Level, Course_Name, and Prerequisite. Course_Level indicates the level of the course taken for identification. Course_Name contains the entire course name. Prerequisite contains the Boolean value indicating the admin selected prerequisite courses.

Field	Type	Key	Example
ID	Int	Primary	1
Course_Level	Varchar		200
Course_Name	Varchar		CS 250 Programming with C++
Prerequisite	Boolean (tinyint)		1

The course_plan table contains all “general” courses that are not indicated as prerequisite and include the fields; ID, Current_Term, Email, Course_Name, and Status. Current_Term indicates the term that the student requested. Email correlates to the user email. Course_Name indicates the user requested course to be taken. Status indicates the approval status from the admin.

Field	Type	Key	Example
ID	Int	Primary	1
Current_Term	Varchar		Fall 2024
Email	Varchar		Bhear005@odu.edu
Course_Name	Varchar		CS 250 Programming with C++
Status	varchar		Pending

The prerequisite table contains all the same fields as the course_plan table.

Field	Type	Key	Example
ID	Int	Primary	1
Current_Term	Varchar		Fall 2024
Email	Varchar		Bhear005@odu.edu
Course_Name	Varchar		CS 250 Programming with C++
Status	varchar		Pending

[Page intentionally left blank.]

The advising_history table contains the fields; ID, Name, UIN, Email, Submission_Date, Term, Last_Term, Last_GPA, and Term_Status. The Name and UIN field corresponds to the user who submitted the advising request. The Email field corresponds to the user email address. The Submission_Date indicates the date that the request was received, rejected, or approved. The Term field indicates the current term the user is signing up for. The Last_Term field indicates the previous term the user attended. The Last_GPA field indicates the most recent GPA of the user. The Term_Status indicates the admin approval status.

Field	Type	Key	Example
ID	Int	Primary	1
Name	Varchar		Brendan
UIN	Varchar		12345679
Email	Varchar		Bhear005@odu.edu
Submission_Date	Varchar		11/7/2024
Term	Varchar		Fall 2024
Last_Term	Varchar		Spring 2024
Last_GPA	Float		3.01
Term_Status	Varchar		Pending

5. Implementation

Develop a screen to display advising sheets submitted by CS department students.

The prerequisite form for the administrator contains all courses loaded dynamically from the database. When the checkbox is checked the input is managed through an API call and the course is added to the bottom of the page. This indicates that the Boolean value was successfully changed in the database to indicate a prerequisite course.

Code contained within: Project -> client -> src -> components -> AdvisingHistory.jsx

Project -> server -> routes -> advisinghistory.js

[Page intentionally left blank.]

Clicking on a student's name will redirect to a page displaying the student-submitted record. On this page, there will be options to approve or reject the record. When the admin submits their decision, they must also provide a text message with their feedback on the advising sheet. After clicking the submit button, the system will redirect to the advising sheet form (as described in point 1), and the new status of the student's records upon submission of approval or rejection.

Code contained within: Project -> client -> src -> components -> CurrentTerm.jsx

Project -> server -> routes -> classes.js

Implement status update of student records upon submission of approval or rejection.

Code contained within: Project -> client -> src -> components -> A_H_StudentView.jsx

Project -> server -> routes -> classes.js

Upon submission, student will receive an email where they can see the status and message provided by admin.

Code contained within: Project -> server -> routes -> advisinghistory.js (/approve or /reject)

Now student will be able to see the update status of their advising sheet on the Course Advising History form.

Code contained within: Project -> client -> src -> components -> A_H_StudentView.jsx

Project -> server -> routes -> classes.js

Add reCAPTCHA to the login page. Verify the reCAPTCHA is verified then user can be logged into the system.

Code contained within: Project -> client -> src -> components -> Login.jsx

Project -> client -> .env (reCAPTCHA keys)

Prevent your application from clickjacking attack. Implement the prevention of click jacking. Show the clickjacking prevention by using <iframe> in .html form.

Code contained within: Project -> client -> src -> main.jsx

Project -> Clickjacking -> index.html (execution example)

Add a favicon to the website.

Code contained within: Project -> client -> public -> favicon-32x32.png

Add a password rule requiring a mix of capital letters, lowercase letters, special characters, and numbers (implement regex for all password fields in the application). The password length should be at least 8 characters.

Code contained within: Project -> client -> src -> components -> CreateAccount.jsx

Project -> client -> src -> components -> EditProfile.jsx

Project -> client -> src -> components -> Login.jsx

Create test cases and execute in you BE application (Create at least 3 test cases)

Code contained within: Project -> server -> Test -> test.js

(npm test to execute)