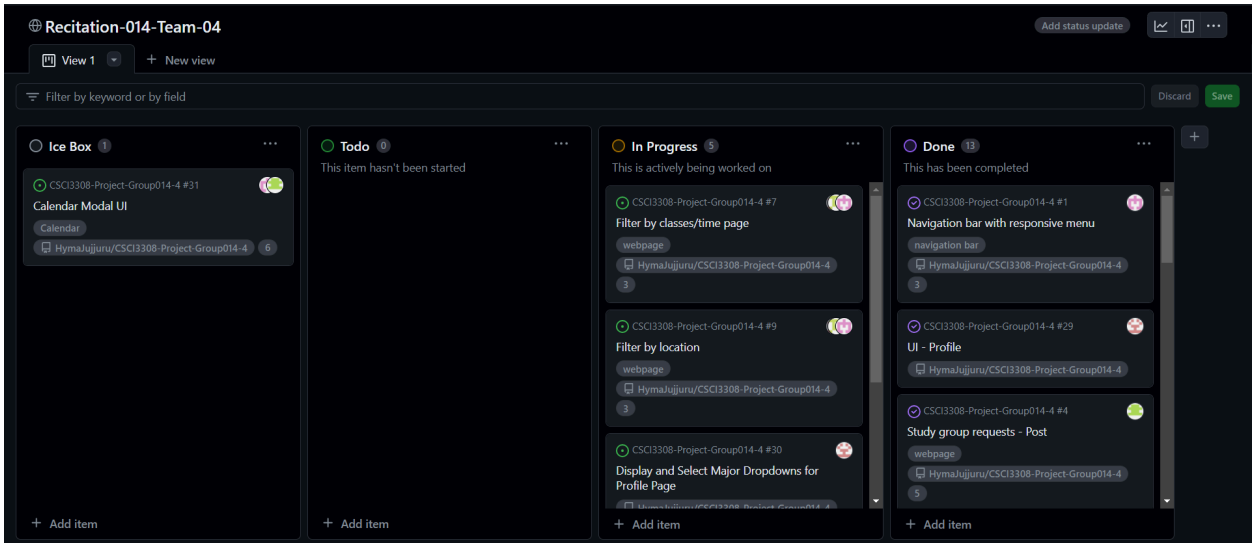
Study Bytes

Hyma Jujjuru, Apollo Hoffman, Santana Reyes, Michael Sexton, Billy Daves

Deployment:
studybytes.eastus.cloudapp.azure.com:3000/login

This application is a campus study group website that displays local study groups happening on or near campus in Boulder. This website is intended to address the problem of finding/being aware of study groups to join on campus. Our target audience is students, and the functionality is to inform/advertise study groups through an easy user interface, with an option to search for specific study groups based on filtering options.
There are 6 different user interfaces for the user to navigate through. The first is the main Study Bytes page, which will prompt the user to either login or sign up for an account. Either option will lead to a pop-up menu for the user to enter their information for registering or signing in. After the user is signed in, a calendar view of general open study groups is displayed. The user has the option to navigate to the advanced events list, profile page, or log out via a navigation bar at the top of the page. The Events pages will list the available study sessions for students to add to their calendars to save the dates. The Profile page allows users to view their events and modify user details. The final navigation bar option "log out" will just log the user out of the application and take them back to the main Study Bytes page, which will again prompt the user to log in or sign up for an account.

Project Tracker:
https://github.com/users/HymaJujjuru/projects/1/views/1



Video:
https://drive.google.com/file/d/1pKEEB4j5FKcbHUxKGWGyOl0sDkPvnTF4/view?usp=sharing

VCS:
https://github.com/HymaJujjuru/CSCI3308-Project-Group014-4/tree/main

Contributions

Hyma: Initially, I worked on creating the framework (navbar, header, footer, layout) of each page so that the rest of the members could focus on dividing up the features. I also worked on creating the events page where events from the database will be populated as cards. This included making the create events modal work like making it open and adding to the database and implementing the endpoints for the events pages and the create events modal. I attempted to create a calendar view of the events to present a better UI which was difficult to implement.

Apollo: I primarily worked on setting up the “Create Event” button, including the api route, the javascript and the html. I also was the primary person who would focus on the lab each week. In the second half of the project, I helped with debugging code in the profile page and the events page in order to integrate the events list and profile page with the functions in our javascript pages. I also attempted to work on a calendar view separate from the Google Calendar with Hyma, however that feature had to be cut in favor of the list view.

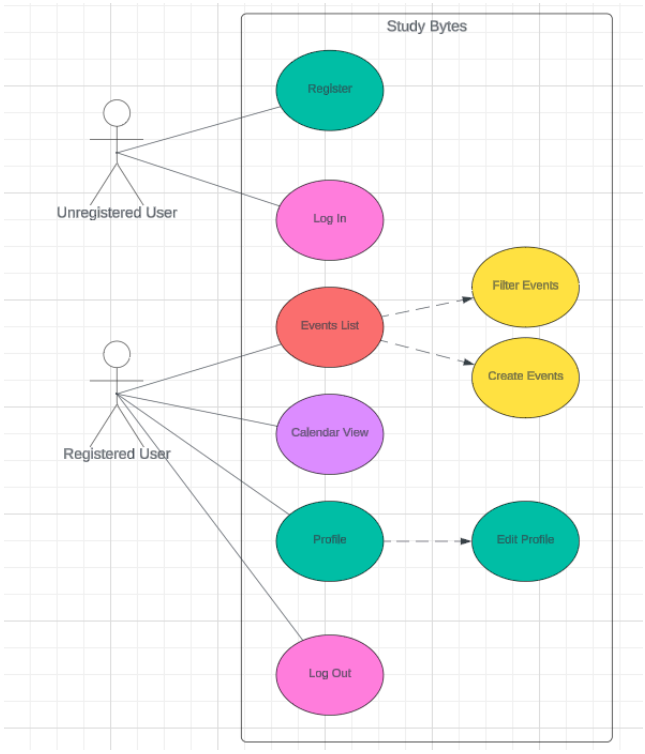
Santana: I primarily focused on the profile page, creating features to alter specific user details such as usernames, major, degree type, and graduation year through specific routes. These routes utilized SQL statements to update corresponding user database entries. Additionally, I contributed to features that were unfortunately omitted from our final product due to time constraints. These included capabilities to modify user-specific study group events directly from the profile page, enabling users to edit or delete their own study group events. I also contributed features for the profile information to be used for additional filtering on our events page, but we opted to cut this due to its scope.

Michael: I worked on creating and inserting data into a postgresSQL database that we used for the core of our website. This includes dummy data for testing/presentation purposes. This database is able to be accessed via apis since it is on its own docker port. I created the database in order to store the relevant information regarding study sessions, students, locations, and courses. I also created a complex filter api route which is able to pull sessions from the database based on criteria that the user enters. The user is able to use just one or multiple different filters at the same time making it a fully functional filter as a user would expect to be able to use.

Billy: I worked on the Google Calendar API implementation. It was mainly straight forward, reading the surplus documentation provided by Google. All we're using it for is displaying the user's primary calendar, and allowing them to move the events we store onto their primary calendar. The user will authorize using the authorize button generating a personal GAPI token. Using the token and the GAPI library we're able to pull their calendar id for the iframe, and insert events into the user's calendar.

Planning Stage

Use Case Diagram:



Wireframe:

Test Results and Observations

1) **Login and Register**
The user acceptance is that the user is able to create an account, and use that account to see other pages.

Observations: When first visiting the app, the user is directed to the Login page, and is only able to navigate to the Registration page. If they have not previously registered, the user will have to register before logging in.
They are required to provide a username and password in both pages. If any fields are not filled in, the username provided in the register tab already exists in the database, or the username provided in the Login page does not exist, an error message will be displayed.
Users are able to see the Login page and a link to the Registration page. It is assumed that the user has previously seen and used a Login page due to how common these are, allowing the user to navigate the page based on their previous knowledge. Due to this previous knowledge and the simplicity of the pages available to the user at this stage, there is little deviance from the expected actions, and the behavior is consistent with the use case.

2) **Create Event**
The user acceptance is that the user can create an event that is visible on the Events page.

Observations: The user is able to see a button on the Events page that says “Create” with a blurb above it that says “Create a study group session!”.
When clicking on the button, a form opens up with these fields: day, start time, end time, location, and recurring. The user is required to provide information in all fields except for recurring, which is a checkbox that, if left unchecked, will create the event for only one day. If any other field is left blank, the page will throw an error and display a message.
Once the form is submitted, the event is visible on the events page, and if the user has authorized the Google API it will be added to their personal calendar as well. Since it is a positive test case, the behavior is consistent with our expectations. However, there are no checks for if the event already exists. It is also easy for a user to get confused due to a lack of confirmation messages, leading to a risk of duplicate events being made. This is a deviation from what we expected, and requires a check for duplicate events and a confirmation message after the user creates the event in order to solve.

3) **Filter Events**:
The user acceptance will be that the user can find events of a specified class and only events from that class.

Observations: The user is able to see a button on the Events page that says “Filter Calendar” with a blurb above it that says “Not seeing what you want? Filter events to your needs!”. When clicking on the button, a form opens up with the fields course code, course number, location, date range, and whether the study sessions should be recurring. None of the fields are required. When one or more fields are submitted, the study sessions on the Events page will only show those that fulfill the requirements set by the user. The user acceptance criteria is fulfilled by the user inputting information into course code and course number fields. The major deviancy in expectation from the user is after events are filtered, as there is confusion about how to remove the filtering. Currently, the way to reset the filters is to refresh the page, but it would be best to add a button to reset filtering in order to prevent user confusion.

4) **User Profile**
The user acceptance is that the user is able to update the username of their account with a new username, and see it appropriately displayed within the profile page.

Observations: If a user wishes to change their profile username, they must do so in the profile page of our application. After registering and logging into the application, they can navigate to the profile page. Here, the user can update their username by inputting their new username into the text box located near the username section, and click submit. The new username will be appropriately displayed on the left hand side of the screen under the username section. The user can log out and re-log back in using the new username and same password associated with their account. Due to the fact that there are other usernames stored on our database, changes may need to be made to check if the new username already exists in the database, and prompt the user to choose a new username. However since this is a positive test case, the behavior is consistent with our expectations, as we are testing with the knowledge that this is a completely new username within the database. Therefore, user acceptance criteria is fulfilled, as the new username is appropriately displayed as expected.