

Apollo

Design Document

CS 307

Team 4:

Jacob Beene

Ali Darwish

Brandon Jiang

Edward Potapov

Daniel Sowers

Saimonish Tunguturu

INDEX

INDEX	1
Purpose	2
Functional Requirements:	2
Non-Functional Requirements:	5
Design Outline	6
System Overview:	6
High-level System Structure:	7
Design Issues	8
Functional Issues:	8
Non-functional Issues:	9
Design Details	11
Class Design	11
Sequence Diagrams	14
State Diagram	19
UI Mockup	20

Purpose

Most Purdue students have gone through the process of desperately looking for information about the course they are signing up for and connecting with those who have previously taken it. Our purpose with this project is to create an efficient way for Purdue students to gather data on their potential courses based on the experiences of those who have previously taken the course, and also allow a way for current students to connect with their classmates and professors. With Apollo, students will no longer need to browse various platforms in order to find the information for a course they are looking for. This will be done by combining aspects of a social networking site such as chat-rooms with a review site and compiling crowd-sourced data based on official course data scraped from Purdue's websites and organizing this data into an index of courses.

Functional Requirements:

Account

1. As a user, I want to be able to create an account.
2. As a user, I want to be able to set my role (student vs prof).
3. As a user, I want to be able to add my major (students) / area of practice (professors) to my profile.
4. As a user, I want to be able to link my social media accounts (Facebook, Twitter, Instagram) onto my Apollo profile.
5. As a user, I want to be able to add my class to my profile.
6. As a user, I want to add my current courses to my profile.
7. As a user, I want to be able to add a profile picture to my profile.
8. As a user, I want the ability to make my personal information secure (public vs. private profiles).
9. As a user, I want to be able to reset my password and username.
10. As a professor, I want to add an "about me" section on my account.
11. As a developer, I want to verify that new users are students / professors at Purdue.

Class Homepage

12. As a user, I want to be able to go to a class' homepage when I click on a class.
13. As a user, I want to see a description of the class on its homepage.
14. As a professor, I want to be able to add my own description of the class on its homepage.

15. As a professor, I want to be able to upload new information or correct misinformation provided about the course I am teaching.
16. As a professor, I want to link an up to date PDF of various class resources such as syllabi from previous semesters.
17. As a user, I want to be able to see the professor(s) for a class on its respective homepage.
18. As a user, I want to see the grade distributions of previous semesters in an elegant pie chart showing the percentages of different grades obtained in the class.
19. As a user, I want to be able to see what semesters a class is being / has been offered in recent years.
20. As a user, I want to be able to see what times a class is being offered during the current (or upcoming) semester.
21. As a user, I want to see the ratings / reviews for the class.
22. As a user, I want to be able to leave a detailed rating / review for the class.
23. As a user, I want to be able to upvote / downvote reviews or questions that users post.
24. As a user, I want to be able to star / pin / favorite classes that I have interest in.
25. As a developer, I want course content to update with each new semester.
26. As a user, I want to see the difficulty level (based on their ratings) of pairing this course with a course I choose.
27. As a user, I want to see a leaderboard of the highest ranked classes at Purdue based on various categories such as material, difficulty, quality, and likability.
28. As a user, I want to be able to post questions and receive answers from other users.

Dining Court Homepage

29. a user, I want to be able to view a dining courts homepage when I click on it.
30. As a user, I want to be able to view the menu for a specified dining court.
31. As a user, I want to be able to view the hours for a specified dining court.

Map

32. As a user, I would like to view an interactive map of campus on the map page
33. As a user, I would like search for buildings on the map

- 34. As a user, I would like to filter map pinpoints by libraries, restrooms, dorms, dining halls, etc.
- 35. As a user, I would like to get a brief blurb about the history of various locations across campus.

Other

- 36. As a user, I want to be able to switch between light and dark mode
- 37. As a user, I want to be able to see information about Purdue buildings and history.
- 38. As a user, I want to be able to view the schedules for Purdue athletics

Search

- 39. As a user, I want to be able to search for classes
- 40. As a user, I want to be able to search for other users on Apollo
- 41. As a user, I want to be able to search the Purdue directory
- 42. As a user, I want to be able to search for dining courts
- 43. As a user, I want to be able to search for buildings on campus
- 44. As a user, I want to be able to search for events.
- 45. As a user, I want to be able to search up study groups.
- 46. As a user, I want to be able to search for clubs.
- 47. As a user, I want to be able to search for study spaces.

Social Network Interactions

- 48. As a user, I want to be able to view other users' profile pages when I click on their name
- 49. As a student, I want to be able to join group chats for classes
- 50. As a developer, I want the class group chat to reset after each semester.
- 51. As a student, I want to be able to organize or attend study sessions / groups
- 52. As a student, I want to be able to see a ranking of all the different study spaces students use.
- 53. As a user, I want to be able to add / remove other users from my friends list
- 54. As a user, I want to be able to add other users to my block list
- 55. As a user, I want to be able to receive email notifications for certain responses and posts
- 56. As a student, I want to see other people's plan of study in terms of future classes they would like to take or are looking to take.

Non-Functional Requirements:

Scalability

1. As a user, I do not want to experience degradation in performance under load.
2. As a user, I do not want to lose data.
3. As a user, I want a reasonable response rate.

Usability/Clarity

4. As a user, I want an easy to understand user interface.
5. As a user, I want to navigate pages within the site without any issues.

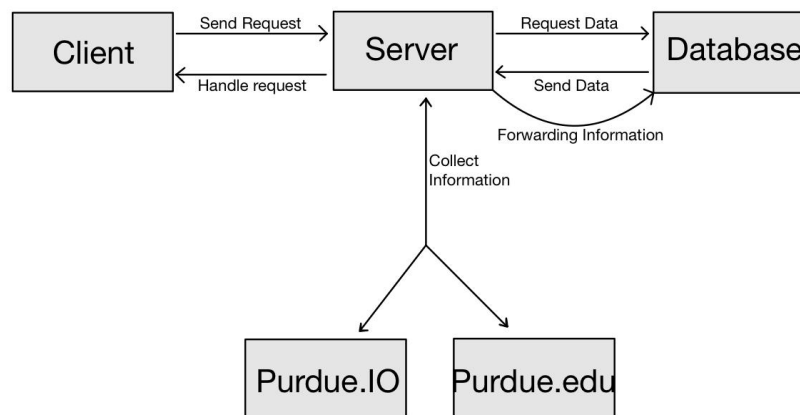
Security

6. As a user, I want my account to be secure.
7. As a user, I want my private information to be kept private.

Design Outline

System Overview:

This project is a web application that allows its users to gather information about courses, review their courses, and interact with those who are taking or have previously taken given courses. In this application, a client-server model will be used where a single server will handle all client requests by communicating with our database. Our database will be where we store all user data and course information along with various other data collected from online resources such as Purdue websites.



Client

1. Our client will provide an interface for the user to communicate with our server.
2. Our client will be implemented using the React framework.
3. The client will send requests to the server for processing.
4. The data received from the server will be presented to the user.

Server

1. Our server will function as a bridge between the client and database.
2. Our server will be implemented using the Node and Express frameworks
3. The server will request the data from the database needed to satisfy the request made by the user.
4. The server will parse the data retrieved from the database and present it to the user after performing necessary operations on the data.

Database

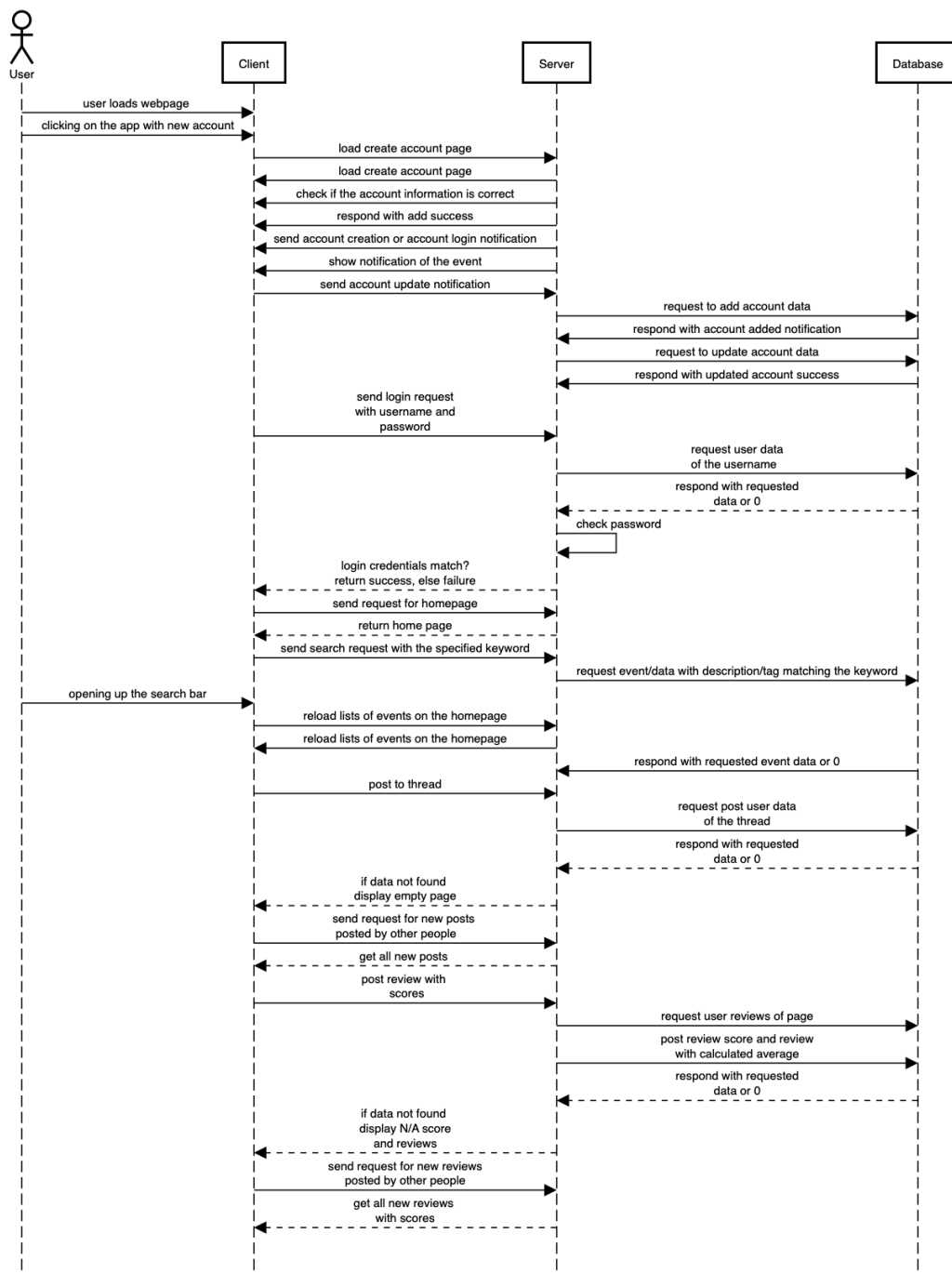
1. Our database will store all data.
2. Our database will be implemented using MongoDB.
3. Along with the data gathered from the users, the database will also store data gathered from online resources such as Purdue University websites.

Online Resources

1. Online resources refers to data we will be scraping from Purdue websites and other websites as needed.

High-level System Structure:

The high-level structure of our system shows all the different interactions of the client, backend, and database. The system is always started by the user that gets some data on the client. The client then asks the backend to provide that data and do some calculations and collect specifics about certain course pages, building pages, and event pages. The backend manages it and then sends requests for the database which provides the data it is looking for. This data is brought forward and alters the frontend client as a result.



Design Issues

Functional Issues:

1. How to verify Purdue students/instructor accounts?
 - Option 1: Use a moderation system to check whether a registered user is affiliated with Purdue using a Purdue ID number to verify.
 - Option 2: Use an email authentication service to verify that the user has a Purdue email.
 - Option 3: Don't require any authentication for signing up. Just register with username and password
 - Decision: We decided on using an email authentication service to verify that the user has a verified Purdue email. The reason being that we want some form of verification that the user is in fact a Purdue student or professor. Since we won't have a list of Purdue IDs on hand to verify that the one provided is a specific user we will go with an email instead. We will make sure that the user verifies the email by clicking a verification link sent to the email. For professors we will need to have a moderation system to verify that the user used a professor's email so that they can edit information on a class page.
2. How will a user search for classes, buildings, and events?
 - Option 1: Search for each item in a search bar similar to a google search bar.
 - Option 2: Search with an advanced search bar that includes specific details such as departments, buildings, etc.
 - Option 3: Have the homepage be grouped in different sections for different pages and the user clicks on the link.
 - Decision: A search bar works well to navigate a large amount of courses rather than going through a list of them on the homepage. We opted for a simpler search bar since we notice having filters bogs down the speed of searching up a class. This will be our main means of navigating the site.
3. Who will be able to view classes, buildings, and events?
 - Option 1: Users will be able to see all the pages.
 - Option 2: Students and professors will only be able to view pages relating to what they are taking or teaching.
 - Decision: For our application we will let people search for classes regardless of their status of taking the class. We will also allow users to post in courses they might not have taken/taught for. This is to give some freedom where the user can discuss

classes that they are considering taking but haven't yet. The idea to restrict them might be helpful for avoiding spam, but overall it will cause more harm than good.

4. Who will be able to rate and review courses and buildings?
 - Option 1: All users will be able to.
 - Option 2: People that have taken the class, and everybody can rate buildings.
 - Decision: Our decision for this topic is to stick with the second option. The reasoning is to keep the review system as ethical as possible. The intention for registering with a Purdue email is to specifically prevent any non-Purdue affiliated people on to the site and spam review a page. This also applies to Purdue affiliated people spam reviewing courses that they have not taken to hopefully prevent negative hoard reviews. For a student to review a course they should have already taken the course or are currently registered to take the course. Specifying which course they are taking will be done on account creation and at the start of each semester. Reviews will also entail various components such as difficulty, quality, quantity of material, and relevance. These subscores will be calculated into a super score out of 5 points.
5. How will content be updated for different pages?
 - Option 1: Both students and professors will have the ability to post new content through the thread functionality of the course pages.
 - Option 2: The ability to post specific content such as core content will be reserved to professors while students will be able to provide information in the thread discussions.
 - Decision: Again we want to stick with a more ethical solution for our project, and while it is nice to have a platform that caters to both professors and students alike. We still have to have some moderation for the platform so that content doesn't become inaccurate or uncontrolled. That's why professor accounts will have more power to edit course descriptions and content to what they see fit. They will be able to edit core information and attach other content that might be useful for the students to see before taking a class.

Non-functional Issues:

1. What is going to be our frontend framework?
 - Option 1: Raw HTML, CSS, and Javascript
 - Option 2: Angular
 - Option 3: Vue.js
 - Decision: We will be doing with React since our group has more experience with React. It is also more popular with more widespread support for the library. It will also be more applicable for our usage since the MERN stack is more common nowadays than the MEAN stack. Also generally we want a way to manipulate the UI without directly accessing the DOM.

2. What is going to be the backend framework?
 - Option 1: Java
 - Option 2: Express.js/Node.js
 - Option 3: Python/Django
 - Decision: We wanted to stick with the usage of Javascript for the whole application. Even though we mostly have experience with Java as a backend language, we find it more fitting to be on the same page of languages used throughout the project. It is also relatively easier to build an API using Express.js on top of Node.js.
3. What is going to be our database solution?
 - Option 1: Relational Database such as MySQL
 - Option 2: NoSQL database such as MongoDB
 - Decision: Since our stack seems to be more focused on Javascript the intuitive direction we would take for our database solution is MongoDB. Since we aren't as experienced with relational databases MongoDB's NoSQL style of storing data is more intuitive. MongoDB will be using a document based database in the form of JSON files, which would be easier to manage instead of searching for the right entry in a SQL database. It is also easier to learn with minimal startup costs.
4. What is going to be our method of collecting basic information about classes, buildings and other information relevant to Purdue?
 - Option 1: We crowd-source the majority of the information by providing the skeleton of the pages and users add to them.
 - Option 2: We scrape the information from Purdue sites and curate the basic information about the classes and let professors edit the classes from there.
 - Option 3: Use both methods for data collection.
 - Decision: For our method of collecting data, we are going to be scraping data from Purdue web pages, specifically pages relating to courses and facilities to be the main information for our course pages and building pages. This will provide the basis for the app without the problems that come with crowd sourcing information. Instead of just providing the stubs to the class there is some information that allows users to develop their ideas about it and share information through the threads. We will also use APIs to collect the basic stub information such as the courses that are happening during the current semester.

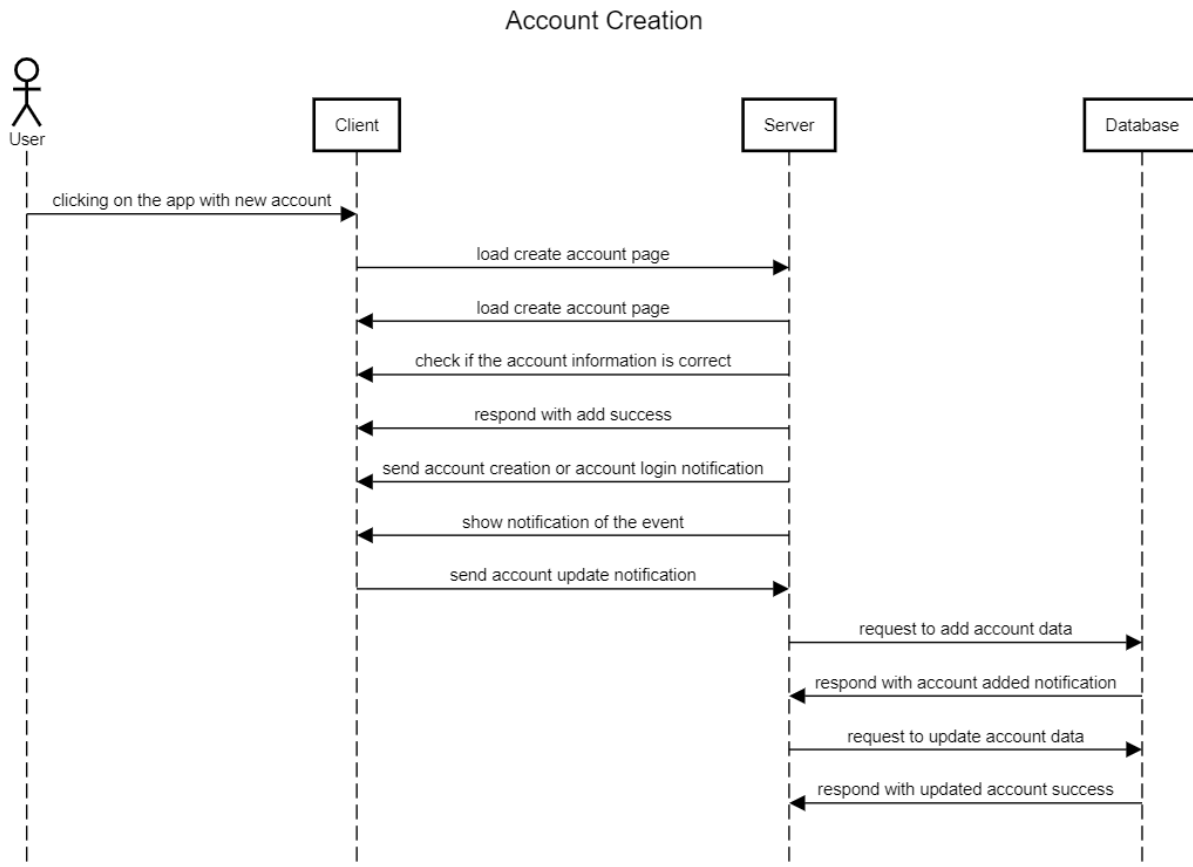
- Other attributes of a professor are: department, list of courses they will be teaching, office, whether they are on tenure, a list of ratings which will be of type Rating, and an average rating.
- The class will also include an addReview() function, which will take in a new review for the Professor, add it to the list, and calculate the new average.
- Student
 - Student is a class that will implement User. It will have the same functions and attributes.
 - It will also have the student's major, their current class year (Freshman, Sophomore, Junior, Senior), expected graduation date, whether they live on or off campus, and a list of courses they are currently taking.
- Course
 - Course is a class that contains details about a certain course at Purdue offered in the current semester.
 - It will have a unique id (used for backend).
 - Course attributes: Course name, department, course number, a list of ratings for the course, average rating, a description, and a thread where Users can post and engage.
 - It will have get() and set() methods for all attributes, as well as an addRating(Rating rating) function.
- Thread
 - Thread is a class for storing information about a thread. It will contain a list of ThreadPosts as well as the date it was created.
 - get() and set() method for each attribute.
- TextField <<Interface>>
 - Text field is a generic interface for all types of text that will be posted.
 - It will store the content of the field, the date it was created, the creator, and the date it was last modified.
 - There will be get() and set() methods for each variable stored.
- ThreadPost
 - ThreadPost is a class for posts made on Threads and will implement TextField, and will have the same attributes and functions
 - ThreadPost will also have the number of likes and dislikes, as well as the number of comments, and a list of comments. As well as an addComment(Comment comment) function to add comments.
- Comment
 - Comment is a class for comments made on posts, and will implement TextField.
 - Comments will also have a number of likes and dislikes.
- Rating
 - Rating is a class that is used for leaving ratings for courses and professors. It will implement TextField.
 - Rating will also have a number rating field.

- Room
 - Room is a class that will store all relevant information to lecture rooms/halls on campus.
 - It will store: Building name, room number, total capacity, and whether there are supported speakers for hearing aids.
 - Room will have a get() and set() method for each attribute.
- Building
 - Building is a class that will store all relevant information for a building on campus.
 - It will store: Address, a list of rooms in the building and a description containing more information and history about the building.
 - Building class will have a get() and set() method for each attribute.

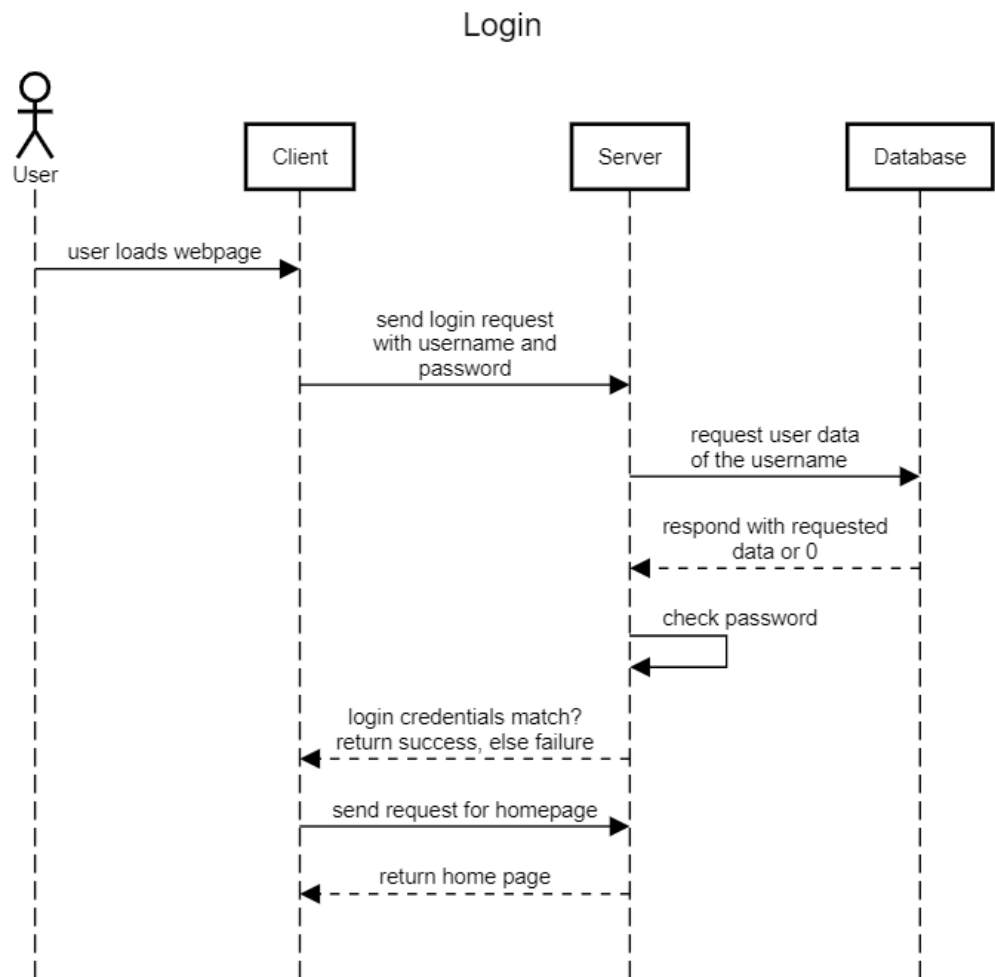
Sequence Diagrams

These sequence diagrams show the different parts of our system involving the client, server, and database. Our main components that we will have to account for is account creation, logging in, searching for pages, posting to a thread, and reviewing. These components will start with the frontend and then make requests from the backend. Our backend will grab information from our database to then bring it to the frontend.

Account Creation:

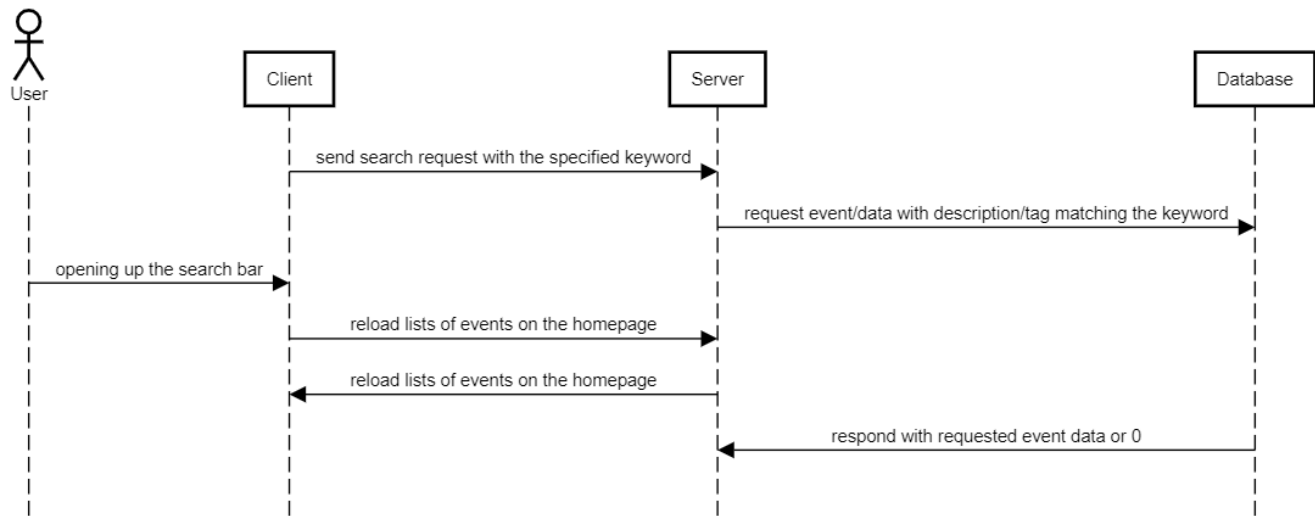


User Login:

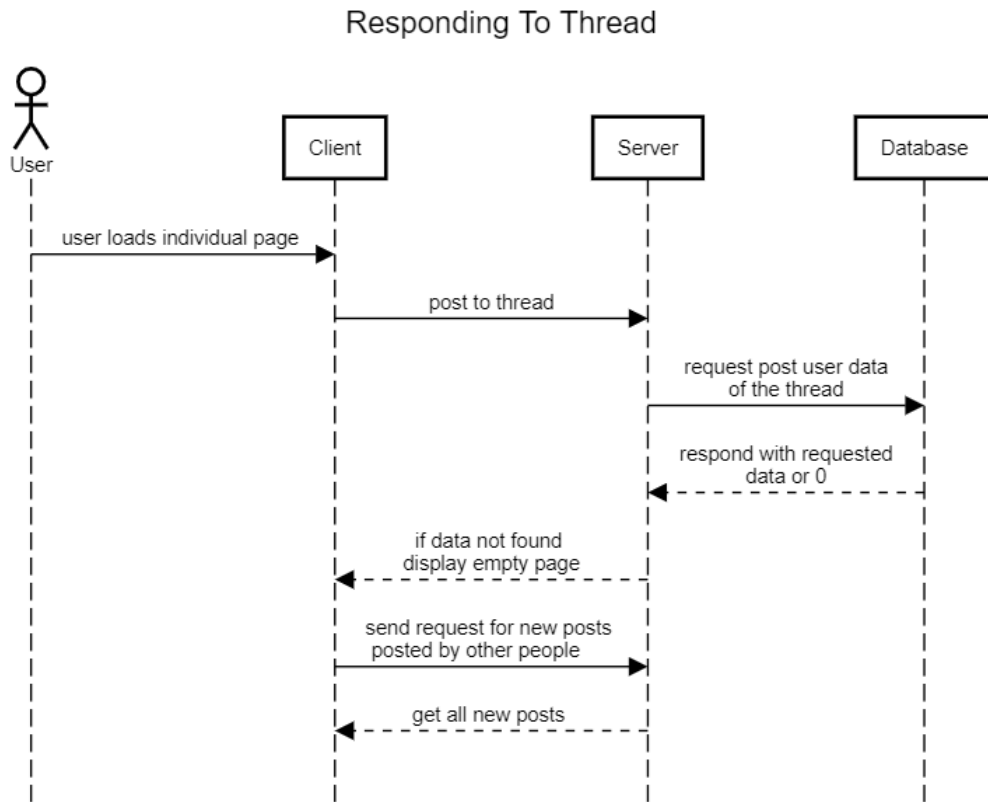


Searching:

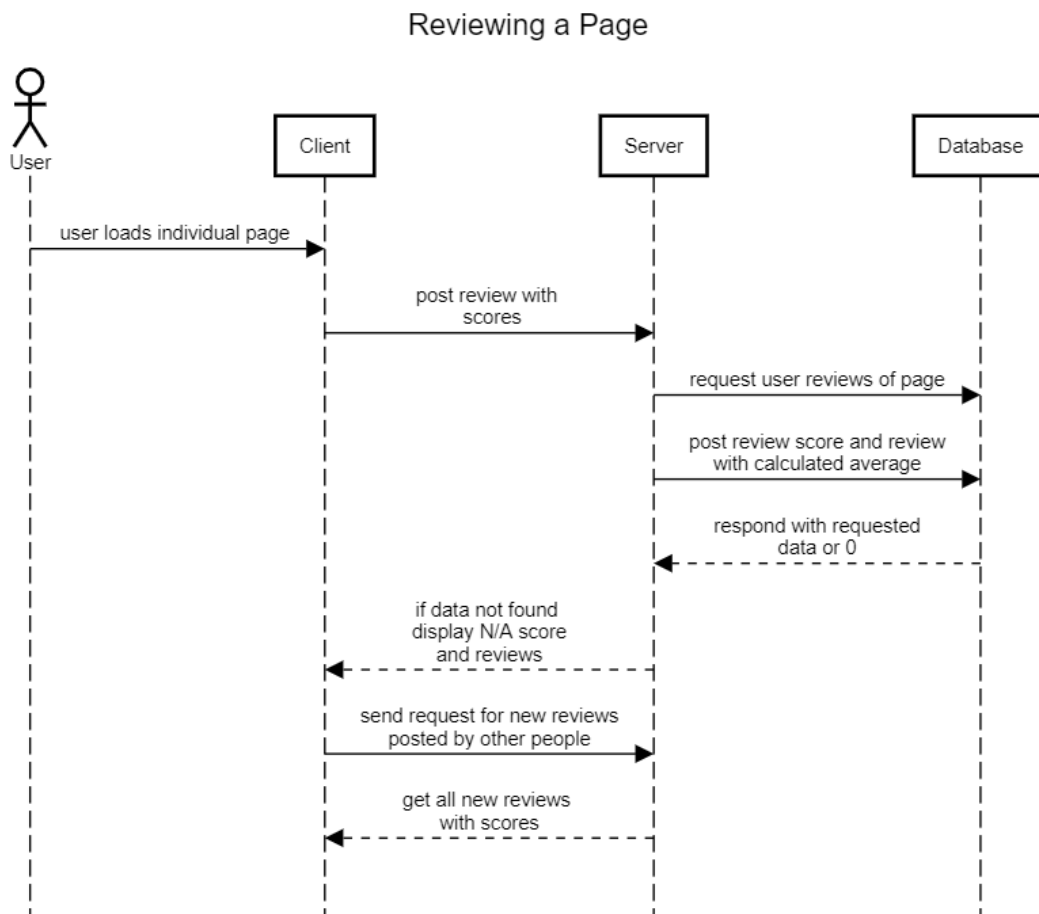
Searching for a Page



Responding to a thread:

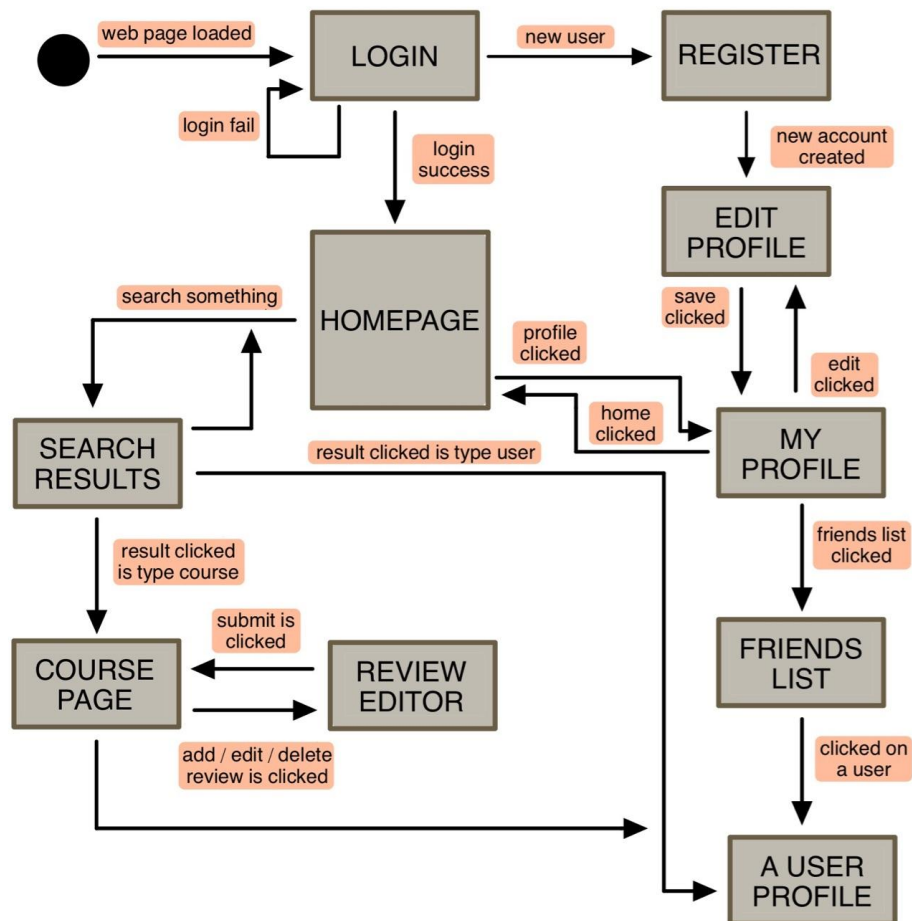


Leaving a review for a page:



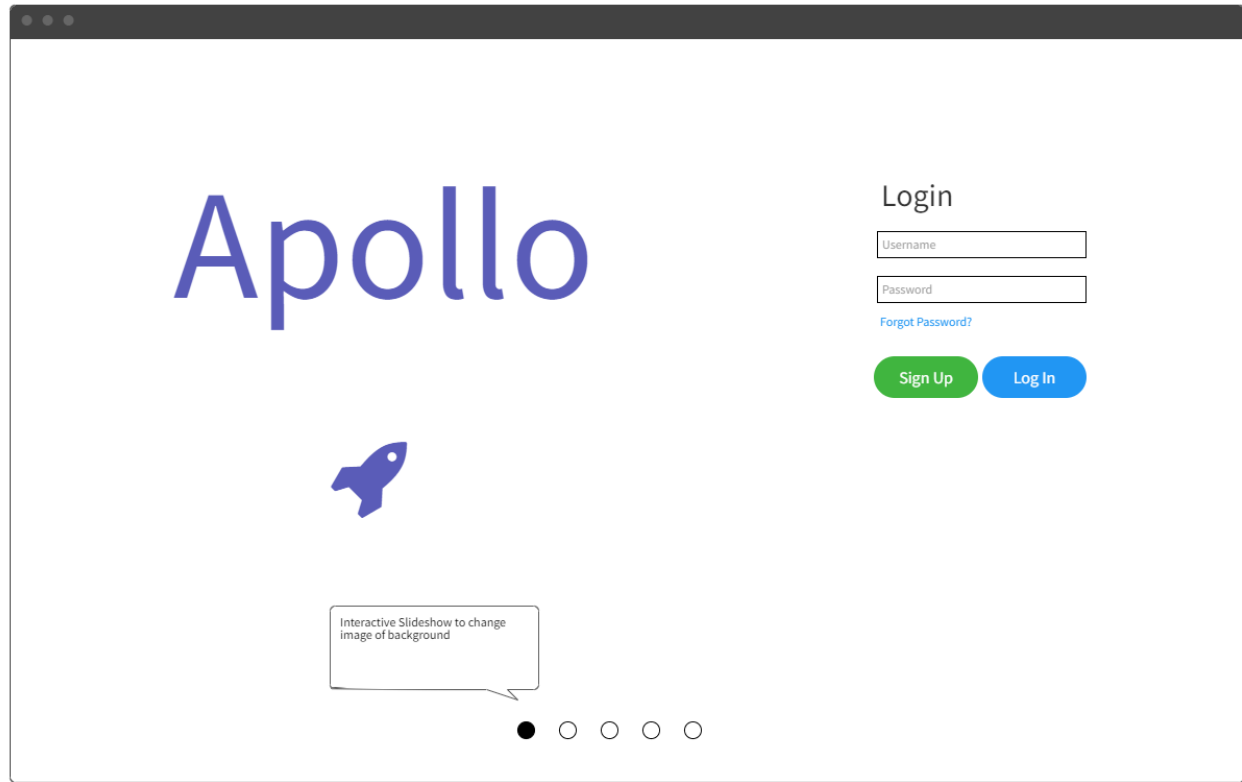
State Diagram

The state diagram visualizes the various states that Apollo may be in. Initially, it launches into the LOGIN page. Once logged in, a user can navigate to their profile or use the search bar to search for classes, users, etc. At any point past the LOGIN / REGISTER state, the user may click the Apollo logo in the top left and return to the HOMEPAGE.

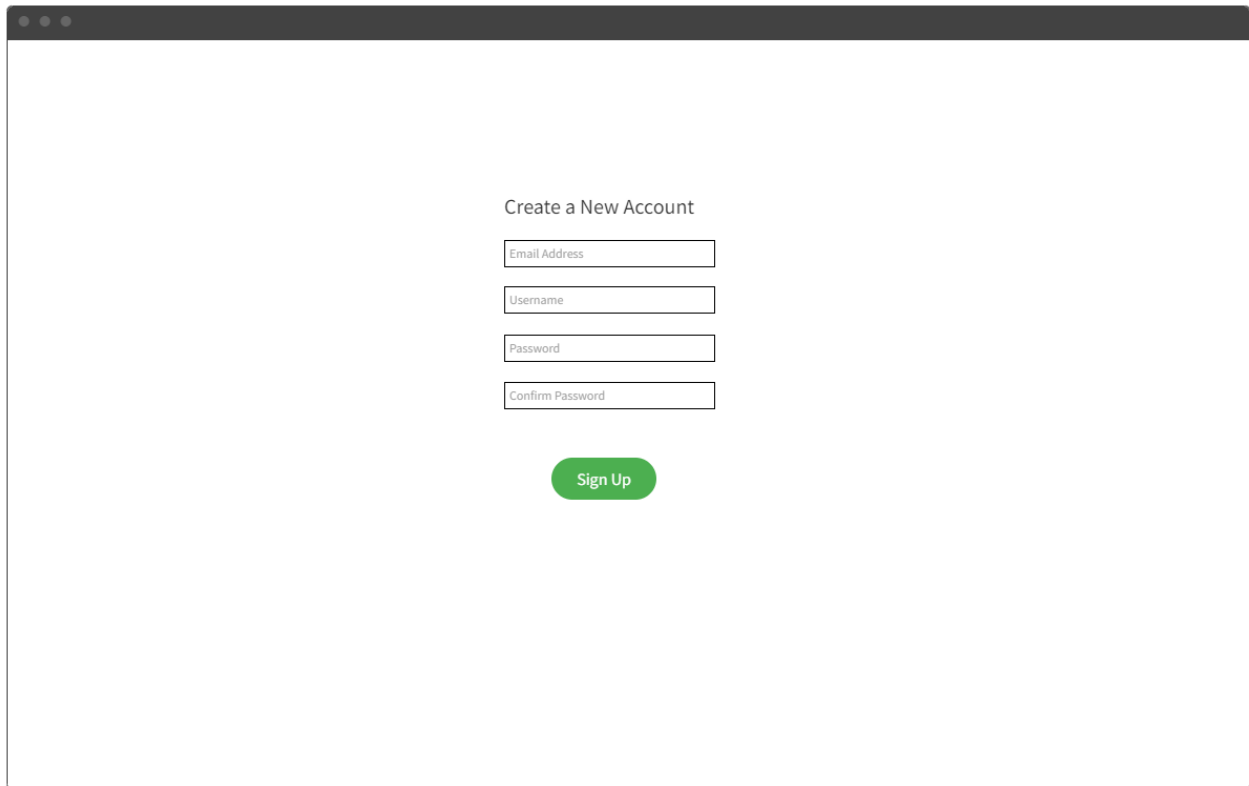


UI Mockup

Our main goal when trying to engineer the UI is to make it as simple as possible for the user. The UI is very organized and accessible for any user. We used icons and bright colored buttons for the user to navigate our site.



Above is the login page when accessing the website. All users need to register an account in order for access because our target audience will mainly be Purdue students and faculty. A user will simply put their information in the text boxes and press the login button to access the content. There is also an interactive slideshow to pick up all the excess whitespace in the Login page. The background images will be the same for every user.



Create a New Account

Email Address

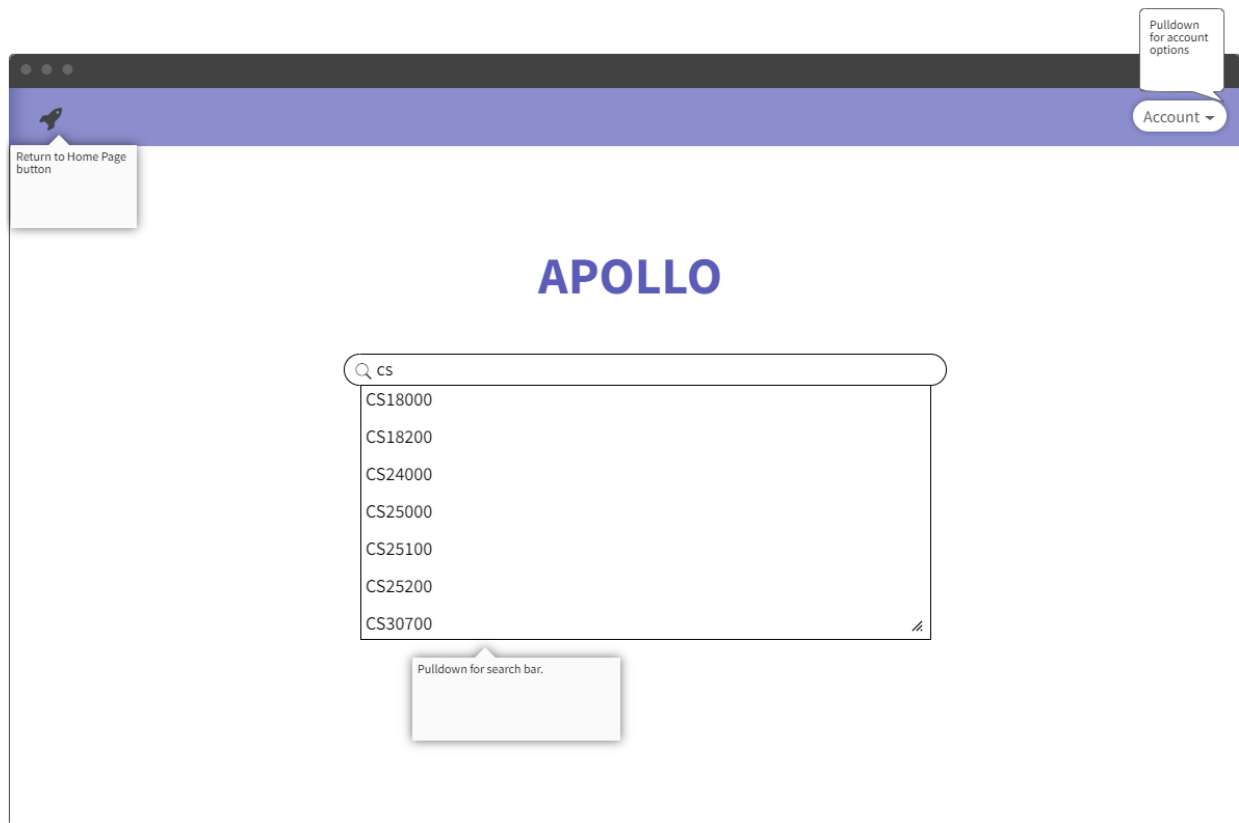
Username

Password

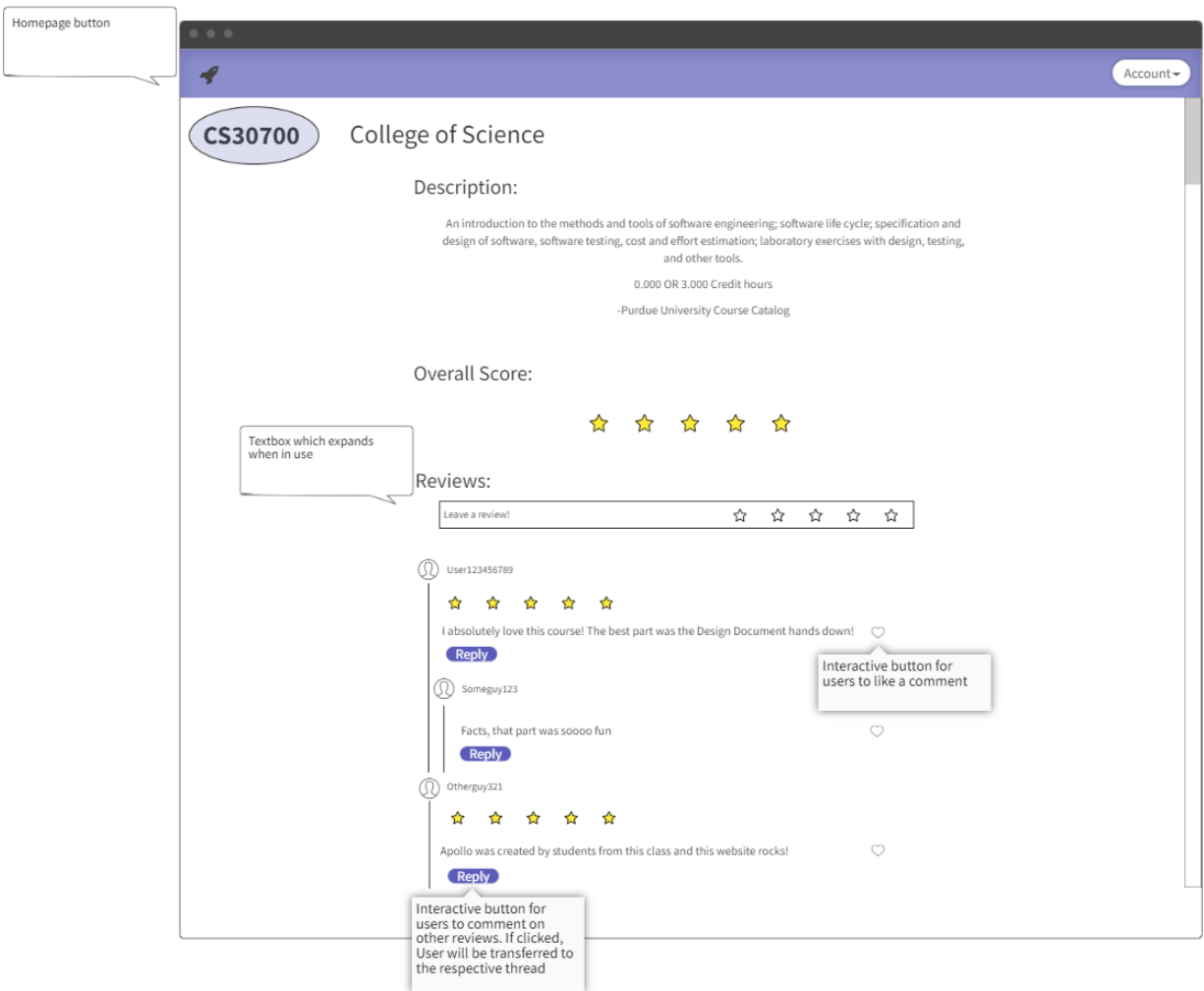
Confirm Password

Sign Up

This is the sign up page. The user will be redirected here if they click the sign up button on the previous login page. Each email address can only be used once to create an account to prevent spam or duplicate accounts. The UI will prompt the user if the email is currently being used.



Next, is the homepage. It may look very simple, but this is the rough way we wanted to design it to keep it as simple as possible for ease of access. The top left Icon will return the user to the homepage and the top right corner is how the user will access their account. The user will type the course ID and results in the pulldown should correspond to what the user is trying to input.



Once the user clicks on the desired course, they will be redirected to the course review page. This page will feature the description of the course via Purdue Catalog, other user's reviews and the overall score of the course. A User will be able to create a new post or reply to other user's posts. Our idea was to have a thread like comment system where it will be easier to reply and correspond each reply to each post. Each review will display the user's username and their score. The overall score will be the average between all submitted reviews. To give a score to the course, the user can click on the star starting from left to right. For example, if the third star is clicked the two stars on the left of it will be autofilled indicating that it will be a 3 star score. All reviews or replies can be liked from other users.

Apollo Account

CS30700 College of Science

Description:

An introduction to the methods and tools of software engineering; software life cycle; specification and design of software, software testing, cost and effort estimation; laboratory exercises with design, testing, and other tools.

0.000 OR 3.000 Credit hours

-Purdue University Course Catalog

Overall Score:

★★★★★

Reviews:

Leave a review! ☆ ☆ ☆ ☆ ☆

User123456789

★★★★★

I absolutely love this course! The best part was the Design Document hands down! ❤️

[Reply](#)

Someguy123

Facts, that part was sooooo fun ❤️

[Reply](#)

Otherguy321

★★★★★

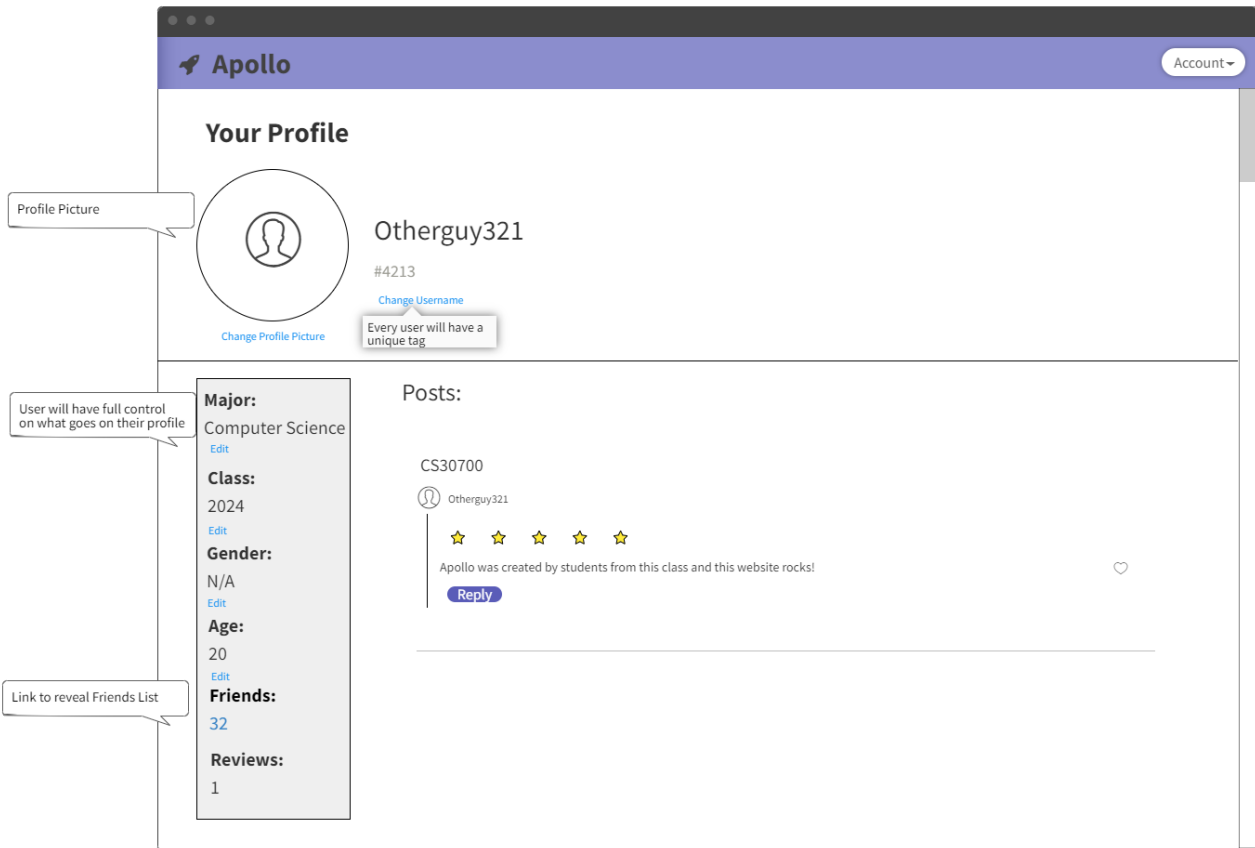
Apollo was created by students from this class and this website rocks! ❤️

What's your reply?

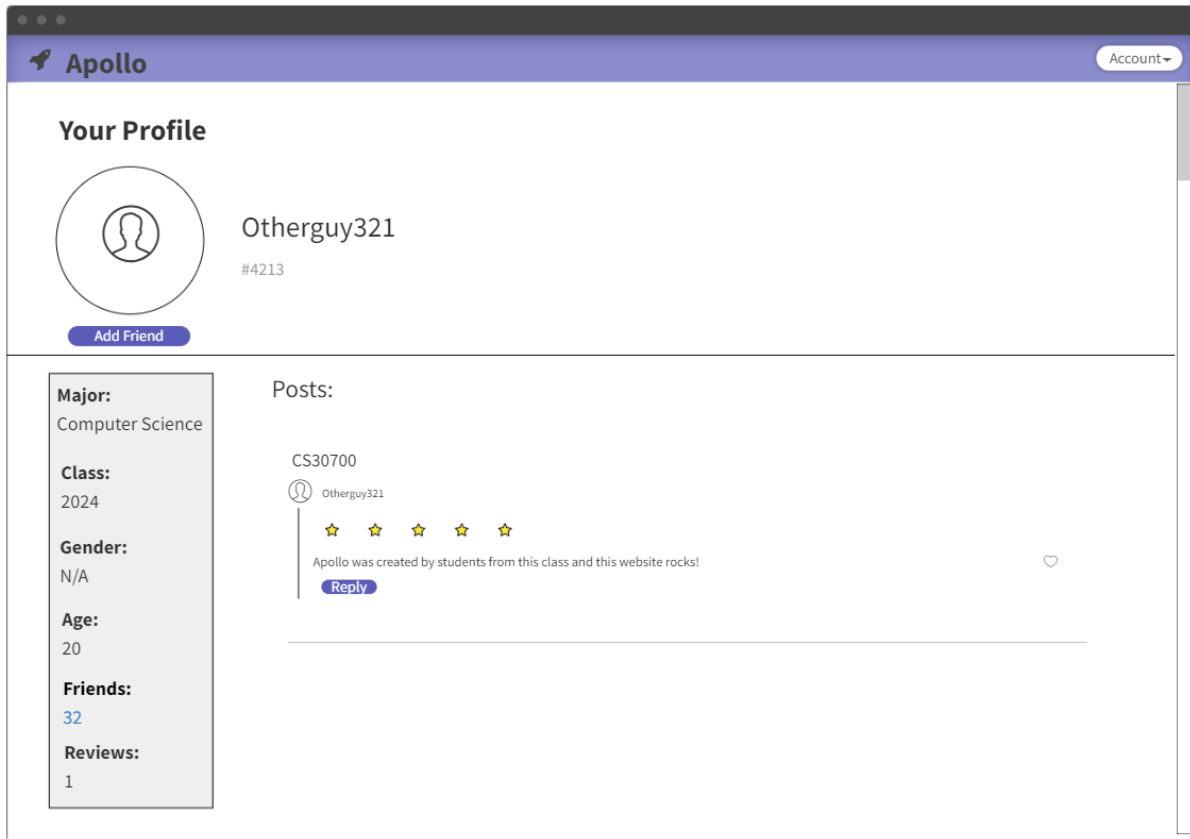
[Post](#)

Text box will pop up when user clicks on reply button

Clicking the reply button will pop a text box for the user to type in. The user cannot score a course when replying to another review. The reply will be under the post they responded to.



This is the user's profile page. The profile page will consist of a profile picture, their username, user ID, their personal info and all their previous posts. The user will have full control on what goes on their profile. They can edit any aspect of their profile page.



Finally, this is a user's perspective on another user's profile. The visiting user will only be able to view, reply to their past reviews, and add the user to their friends list.