



Design Document

CS307 - Team 23

Arshpreet Singh

Kaushik Ramachandran

Luciano Handal Baracatt

Pratyaksh Motwani

Sadiq Ahmed A Albinalshaikh

Table of Contents

1. Purpose	3
1.1 Functional Requirements	3
1.2 Non-functional Requirements	6
2. Design Outline	7
2.1 High Level Overview	7
2.2 Sequence of Events Overview	8
3. Design Issues	9
3.1 Functional Issues	9
3.2 Non-functional Issues	11
4. Design Details	13
4.1 Class Diagram	13
4.2 Sequence Diagram	15
4.3 Navigation Flow Diagram	17
4.4 UI Mockup	18

1. Purpose

easyA is a web platform that will allow Purdue students to learn about what other students think about classes they consider taking. The web application will help students decide between electives and course choices that might be required for their majors. It will stand out from similar websites because of the inherited credibility in the website since only active Purdue students can sign up. This website will allow students to search for classes and these search results can be filtered using professors and/or tags defined by the developers. Everyone can read reviews but only logged-in users can write them, upvote, downvote and report them.

When a student posts a review, they would choose the appropriate tags, the professor and give rating out of 5-stars. The search results can later be filtered by professor names and tags. Also, these posts can be sorted chronologically and by upvotes. There would be links to Rate My Professors for each professor on our website. The professors and courses will be added by administrators upon request from the students. Every post can be reported and each report would be taken into account by the developers and if the report seems genuine, the post will be removed by the developers.

1.1 Functional Requirements: (46 User Stories)

User Interface

As a user, I would like:

- a. to have an easy navigitable user interface throughout the website.
- b. to log in from any webpage in *easyA*.
- c. to log out from any webpage in *easyA*.
- d. to easily access *easyA* on my phone or tablet.
- e. to sort reviews chronologically.
- f. to sort reviews based on their upvotes.
- g. to filter specific course reviews by professors teaching the course.
- h. to filter specific course reviews by tags.
- i. to filter a specific professor reviews by course.

Search/Database

As a user, I would like:

- a. to search for courses by name.
- b. to have suggestions while typing in the search bar.
- c. to have an auto-complete dropdown menu for choosing the professor I am reviewing.

Review System

As a user, I would like:

- a. to easily post a review for a course.
- b. to delete my post.
- c. to report inappropriate reviews.
- d. to upvote/downvote reviews.
- e. to reach the reviewer to ask him or her follow-up questions.
- f. to be redirected to the professor's rating from RateMyProfessor.com after clicking on the professor's name in a review post.
- g. to know when the reviewer took the course.
- h. to know what grade the reviewer got.
- i. to tag my reviews.

Authentication

As a user, I would like:

- a. to sign up for an *easyA* account.
- b. to log in to my account.
- c. to log out from my account.
- d. to change my password after logging in.
- e. to have an account menu for changing my password and logging out.
- f. to know that I am signed in or not.
- g. to reset my password if I forget it.
- h. to be prompted to login when I attempt to post, upvote/downvote, or report.

Accessibility

As a user, I would like:

- a. to access reviews for a course without having to sign up.
- b. to share course or professor reviews through social media.
- c. to use the text-to-speech feature to read reviews aloud.
- d. to translate non-English reviews to English.
- e. to see a course rating in the course review page.
- f. to see a course description in the course review page.

Reliability

As a user, I would like:

- a. to know that all reviews are from Purdue students.
- b. to know that my information will be secure.
- c. my identity to be protected from the readers.
- d. to contact the website administrators for any inquiries like a request to add a course or professor to the database.

Administration

As an administrator, I would like:

- a. to ban certain users from using the website services.
- b. only Purdue students with “@purdue.edu” emails to be able to sign up.
- c. newly signed up users to verify their ownership of their Purdue emails.
- d. to be able to delete any user’s post.
- e. to view and take action on reported reviews.
- f. to manage the website by extending control over course descriptions, professors and course listings.
- g. to be notified by email about any user inquiry.

1.2 Non-functional Requirements: (16 User Stories)

Performance

As a user, I would like:

- a. to run the website smoothly without any discomfort.
- b. the webpages load fast in less than 2 seconds.

As a developer, I would like:

- a. the web application to support 100 user requests simultaneously and upgradeable if needed.

Usability

As a user, I would like:

- a. to have a minimalist user interface the serves the purpose.
- b. to be able to interact with the user interface easily.
- c. to navigate the website without getting lost between the pages.

As a developer, I would like:

- a. the web application to load 10 course reviews at a time.

Security

As a user, I would like:

- a. my information to be encrypted from others and administrators 24/7.

As a developer, I would like:

- a. the web application to be safe from outside attacks 24/7.

Scalability

As a developer, I would like:

- a. a 5 GB expandable storage space for the web content.
- b. a 1 GB expandable storage space for the database.
- c. the web application to support storage enough for 40000 users.
- d. to be able to easily add upgrades to the web application.

Deployment

As a developer, I would like:

- a. to work on the website locally through XAMPP.
- b. to use version control (git) to maintain any changes happen to the web application.
- c. to host and publicize the website through a public domain and a public server.

2. Design Outline

2.1 High Level Overview

Our project, *easyA*, will be a web application that is based on Flask (Python) web framework and Firebase Database and Authentication. *easyA* will be a course review platform for Purdue student who are looking for others' opinions on courses. Our project will follow the client-server model. The server will be handling multiple client requests like accessing the webpages or any database queries. The server will be sending the queries to the Firebase Database and respond back appropriately.

Client

- a. The client will be our user interface for the web application.
- b. The client will request or send HTTP requests to the server.
- c. The client will retrieve or receive data then display it in the user interface dynamically.
- d. The client will validate some data before sending them to the server like the signup information using Regex and Javascript.

Server

- a. The server will receive requests from the clients and validates it.
- b. The server will send data queries to the database.
- c. The server will receive the queries from the database.
- d. The server will handle requests from the clients and respond appropriately.

Database

- a. The database will store users, courses, professors, and reviews information.
- b. The database will receive data queries from the server.
- c. The database will execute the queries.
- d. The database will respond back to the server with the appropriate query result.

2.2 Sequence of Events Overview

The general events that function in our client-server model are drawn in the sequence diagram below. In our web application there will be multiple forms of requests that are either functions at the client level or functions that extend up to the database. Guests (non-logged in users) are able to use most of the client services except for posting reviews, upvoting/downvoting, reporting, or posting an inquiry. This diagram is an abstraction of what events will be functioning from a high level point of view meaning that much of the details/computations were hidden in this diagram for clarity.



3. Design Issues

3.1 Functional Issues

1) What kind of Users will be able to view reviews on the page?

- a) Users with a valid account on *easyA*.
- b) Any user that is able to access the page.

We choose **Option B** to represent the user pool, this is key as we want this page to serve as infographic that will allow any students to be able to obtain information about the course, the professor teaching the course and the level of difficulty of the course being taught in the respective semesters. However, in order to hold authenticity of posts only logged in users that are verified Purdue students will be able to write reviews about courses.

2) What kind of Information will we be collecting from the user for signing up, before he is able post reviews?

- c) Purdue Email ID, Password, Name.
- d) Purdue Email ID, Password, Name, Last Name.
- e) Purdue Email ID, Password, Name, Last Name, Phone Number.

We will choose **Option B** to represent our needs. We choose this as these requirements are adequate to represent the user model required to have a working interaction with our application. We also take steps to authenticate that the user is an affiliated current student at Purdue University, this is pivotal as we want to have reviews only from students that are able to write reviews relevant to the current nature in which the course is taught, this step will also ensure that the page does not have bogus and incredible reviews.

3) What kind of information will we display when a user navigates to a particular course page?

- a) Course description, user reviews.
- b) Course description, user reviews, professor of the course.
- c) Course description, user reviews, professor of the course, tags.

We have decided to represent our web application with **Option C**. This was an important design choice as we wished to provide as much about every course as possible. On the course page a user is able to see the name of the professor, if attendance for the course is mandatory, if the course requires a mandatory textbook and has the ability to view the courses overall rating and filter tags to view selective information about courses.

4) How will the user information be displayed when leaving a review about a course?

- a) First Name, Last Name.
- b) Email ID.
- c) Anonymous.

We choose to implement **Option C**. This is a very important design choice as we wish to have our users anonymity in order to protect their reputation and prevent other users from having a bias. Keeping the users anonymous will ensure that they are able to provide information without being concerned for their privacy.

5) What action will be taking on users who abuse the system and post inappropriate reviews?

- a) Users will report the post, and after more than 5 reports the user will be banned from using the system permanently.
- b) Users will banned from the system temporarily.
- c) Users will be immediately remove and permanently blocked.

We choose to implement **Option A**. This is a pivotal as although we encourage reviews, it is unacceptable to post inappropriate reviews or false reviews that could possibly undermine Purdue University. We give the user five strikes, after which we ensure that the user cannot login to the system to post reviews and is blocked from being able to write any future reviews.

3.2 Non-functional Issues

1) What web service should we use for our backend?

Option 1: Amazon Web Services

Option 2: Microsoft Azure

Option 3: Google Firebase

Choice: Firebase

Justification: The Team chose Firebase as their backend database service because the team wanted to focus on building the platform. Firebase is an easy to implement infrastructure. The team also decided to use Firebase Authentication for the users thereby making Firebase the best backend from a compatibility standpoint,

2) What back-end framework should our team use?

Option 1: Flask (Python)

Option 2: Node.js (Javascript)

Choice: Flask (Python)

Justification: The team decided to use Flask due to better compatibility with Firebase. The team would use PyreBase which is a wrapper for Firebase in Python. Also, using Python would involve less time configuring infrastructure and the team believes it is easier to implement. This would give the team a better head start.

3) What front-end framework would the team use?

Option 1: AngularJS

Option 2: ReactJS

Choice: ReactJS

Justification: The team believes that using ReactJS would be a better option because most team members have experience using it before. ReactJS also comes with better documentation and tutorials which make it very easy to learn. Also, components in ReactJS are reusable which makes them easy to implement for a reviewing platform.

4) How will the search UI be handled?

Option 1: Having a dedicated search results page

Option 2: Having a suggestions drop-down menu

Choice: Having a suggestions drop-down menu

Justification: The team decided to have a drop-down menu after deciding that the courses will be added by the developers. Since the developers would be adding the courses after being requested by users, having a result page seemed redundant.

5) How will the team handle uploading media to the server?

Option 1: Use Content Delivery Network (CDN) for images

Option 2: Upload images to the server directly

Choice: Use CDN

Justification: The team chose to use CDN since it will improve the loading speed of the website. Images will be hosted on different server rather than the web application's server which may slow down loading as the server tries to serve the web application, and media altogether.

6) How does the team plan to test the web application?

Option 1: Using a public server

Option 2: Using XAMPP for locally hosting the web application

Choice: XAMPP

Justification: This will help every team member to have their own version of the project which will help with better version control and help from the public server being corrupted if something wrong is pushed to it.

7) What IDE is the team going to use?

Option 1: Visual Studio Code

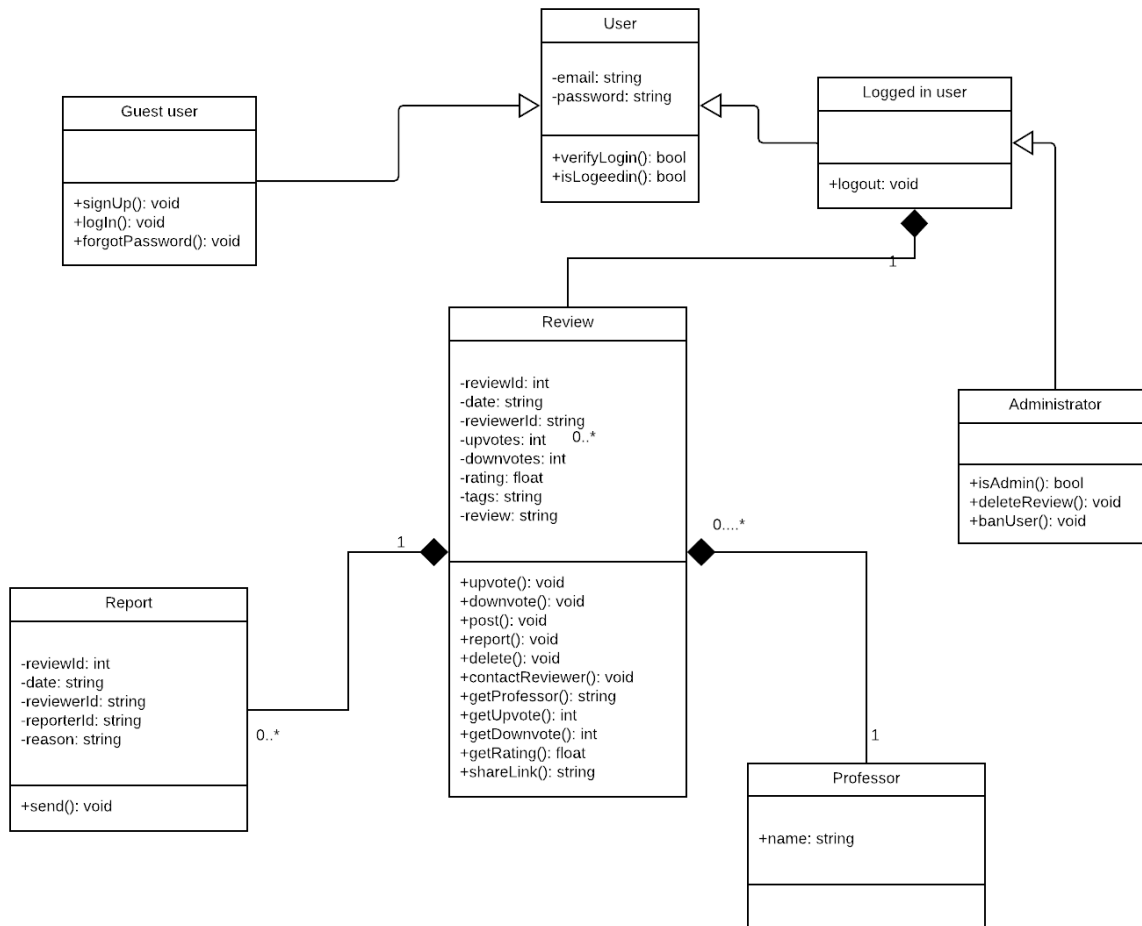
Option 2: PyCharm for Python and Atom for other languages

Choice: Visual Studio Code

Justification: The team decided to Visual Studio Code due to its easy to set up interface. Also, the compatibility of Visual Studio Code with GitHub is better than the others. Another reason was the team's previous experience working with Visual Studio Code.

4. Design Details

4.1 Class diagram



1

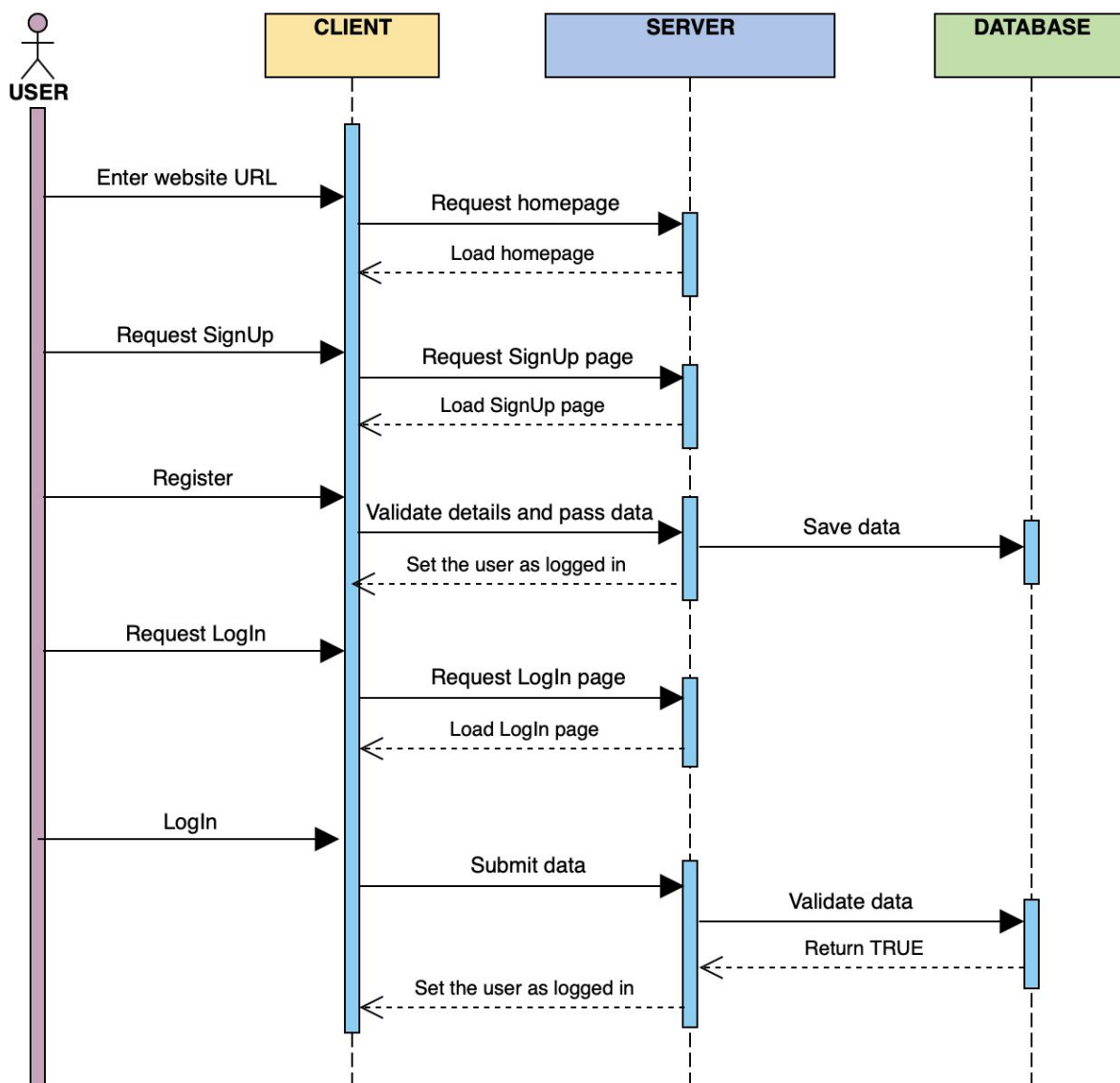
Descriptions of Classes and Interaction Between Classes

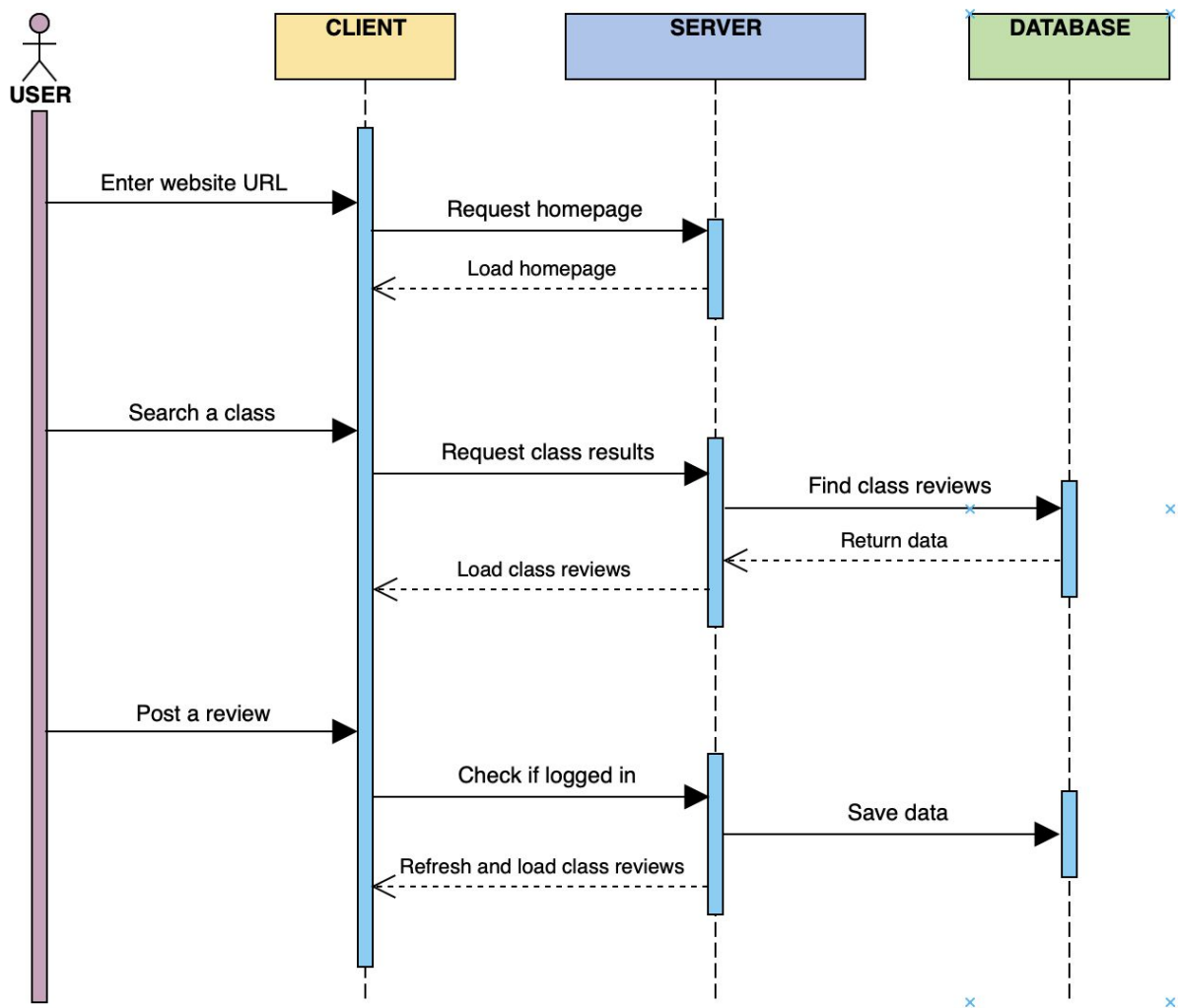
The program would have 7 classes. The four main ones are User, Review, Report and Professor. User would be an abstract class and would have two subclasses, Guest user and Logged in user. Guest user will be allow to log in, sign up and forget password. It would also be allowed to have read-only access to the reviews. Logged in user would get access to the reviews, which includes posting, deleting, reporting up/downvoting, etc. As a subclass of logged in user, there is the Administrator class. An administrator is a logged in user with privileges like deleting others' reviews. Logged in users have a one to many relation with reviews, the reviews will contain the necessary methods to create and delete them. Subsequently, the review class have report and professor. The report class will allow users to report the review and the professor class will allow professors to have entities rather than just strings, allowing to easily avoid duplicity.

- User
 - Abstract class that provides the minimum functionality for any user regardless if it is logged in or not.
 - E-mail: each user will have a unique Purdue email that will serve as an ID.
 - Password: a password to verify the users identity.
- Guest User
 - Guest user will inherit the user attributes.
- Logged in User
 - Logged in user will inherit the user attributes.
- Administrator
 - Administrator will inherit the logged in user attributes.
- Review
 - Review ID: a uniquely assigned number to identify a particular review.
 - Date: the date when the review was posted.
 - ReviewerId: the reviewer's email.
 - Upvotes: the number of upvotes.
 - Downvotes: the number of downvotes.
 - Rating: the rating of the review.
 - Tags: the tags associated with the reviews.
 - Review: the body/paragraph in the written review.
- Report
 - Review ID: a uniquely assigned number to identify a particular review.
 - Date: the date when the report was filed.
 - ReviewerId: the reviewer's email.
 - ReporterId, the reporter's email.
 - Reason: the body/paragraph in the written report.
- Professor
 - Name: the name of the professor.

4.2 Sequence Diagram

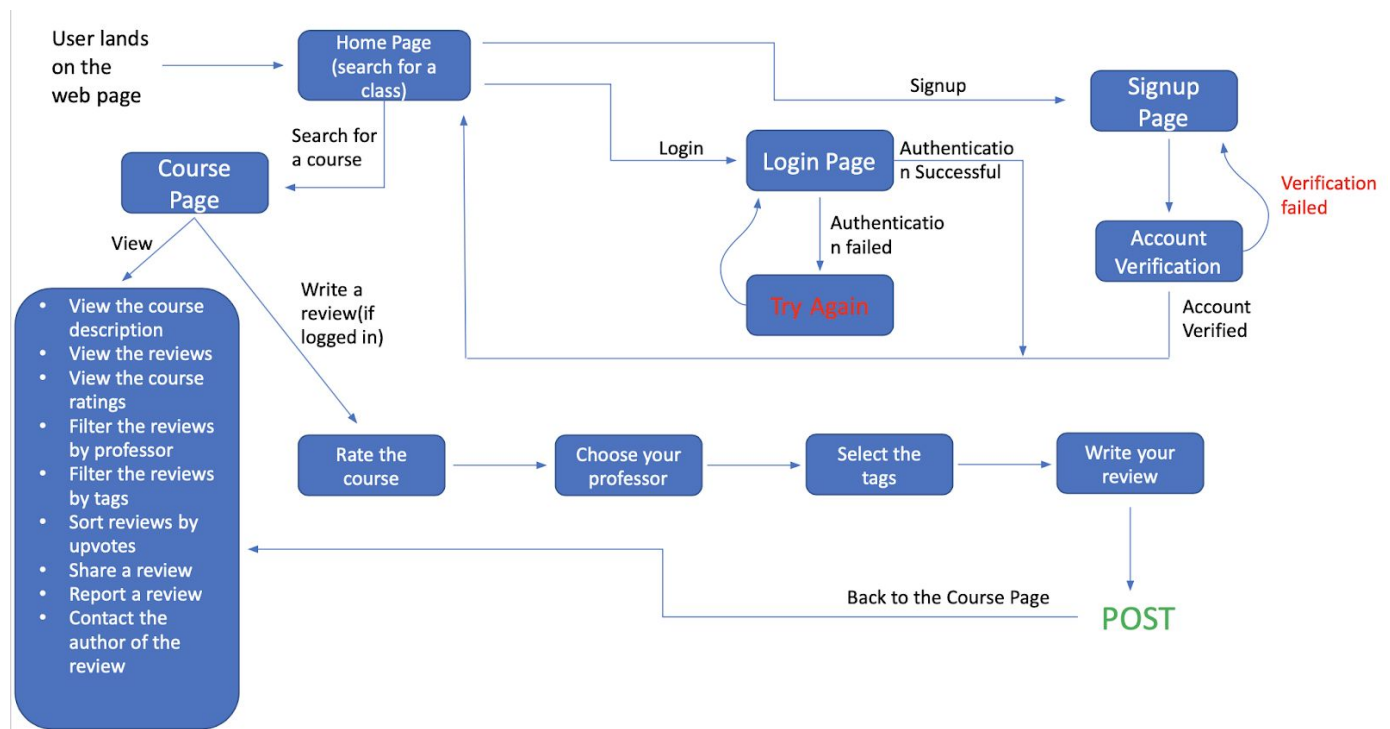
The following sequence diagrams highlight the sequence and interaction flow between various processes and how processes including looking up courses, user login, reviewing courses and sorting posts would behave within the system. The diagram highlights how course data is layered across client and server systems. When a user perform an action or interacts within the webpage, the client server sends a request to the backend server that will get the retrieve the request data from the FireBase Database and send back the information back to the client, to be displayed according by the UI.



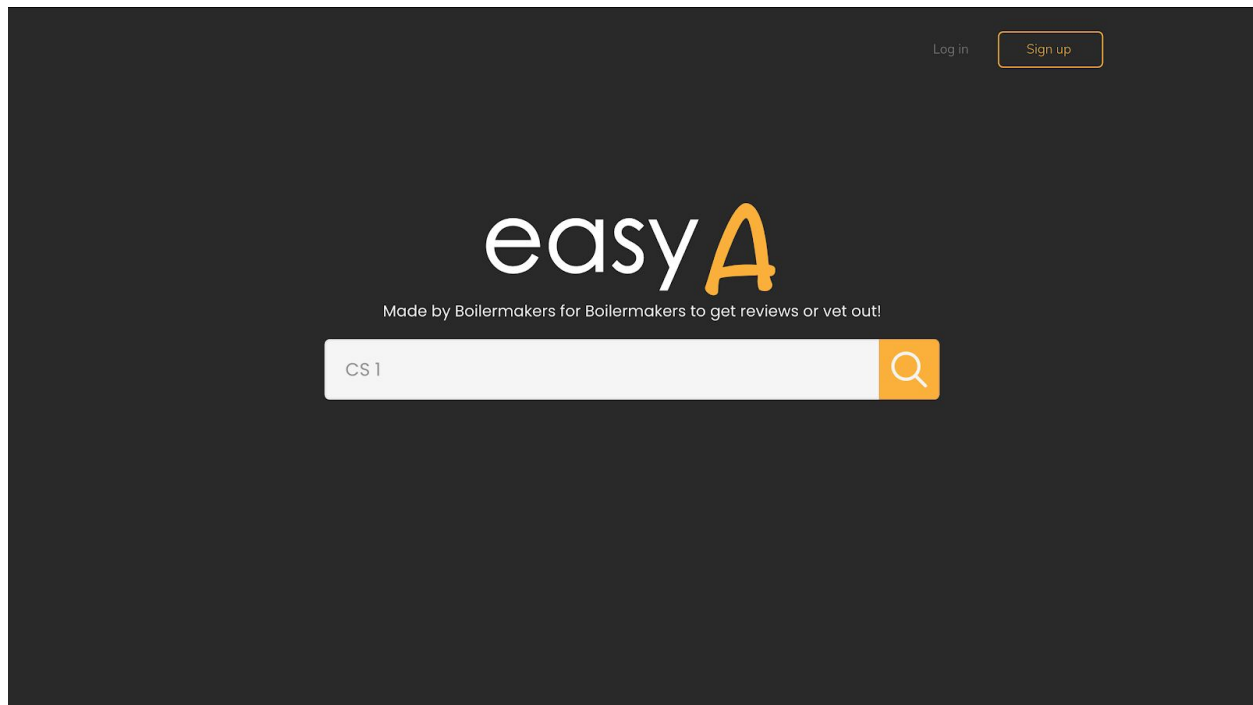


4.3 Navigation Flow Diagram

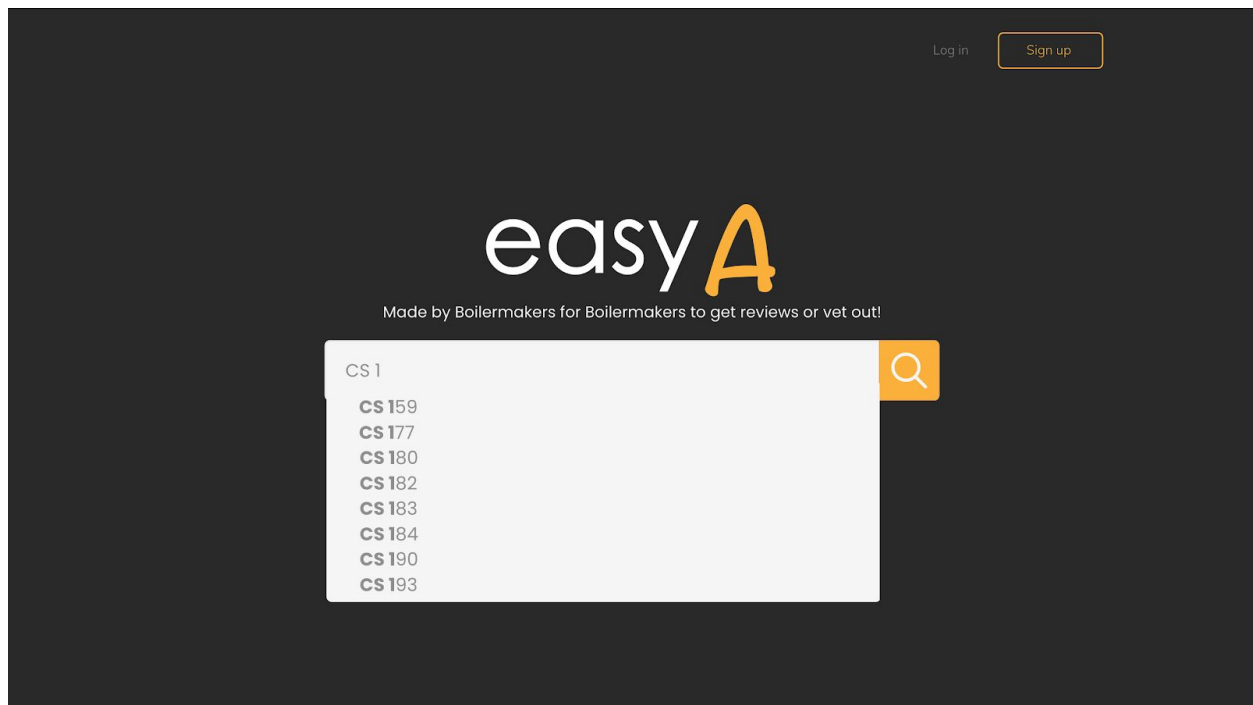
The Navigation Flow map serves to highlight the flow the user serves as he interacts with the flow of the web application. The homepage contains a primary search bar that allows users to look up any course. Once the user selects a course it directs him to the course page that contains the course description and reviews about the course. Although the user is able to look up courses and view reviews about courses, the user cannot write reviews about specific courses without actually logging in to the system. In order to hold authenticity of our system during sign up we ensure that a verification system is in place to verify that the user is a current student at Purdue University. Once verification is complete, the user is able to post review and select specific tags to describe the characteristics of the course.



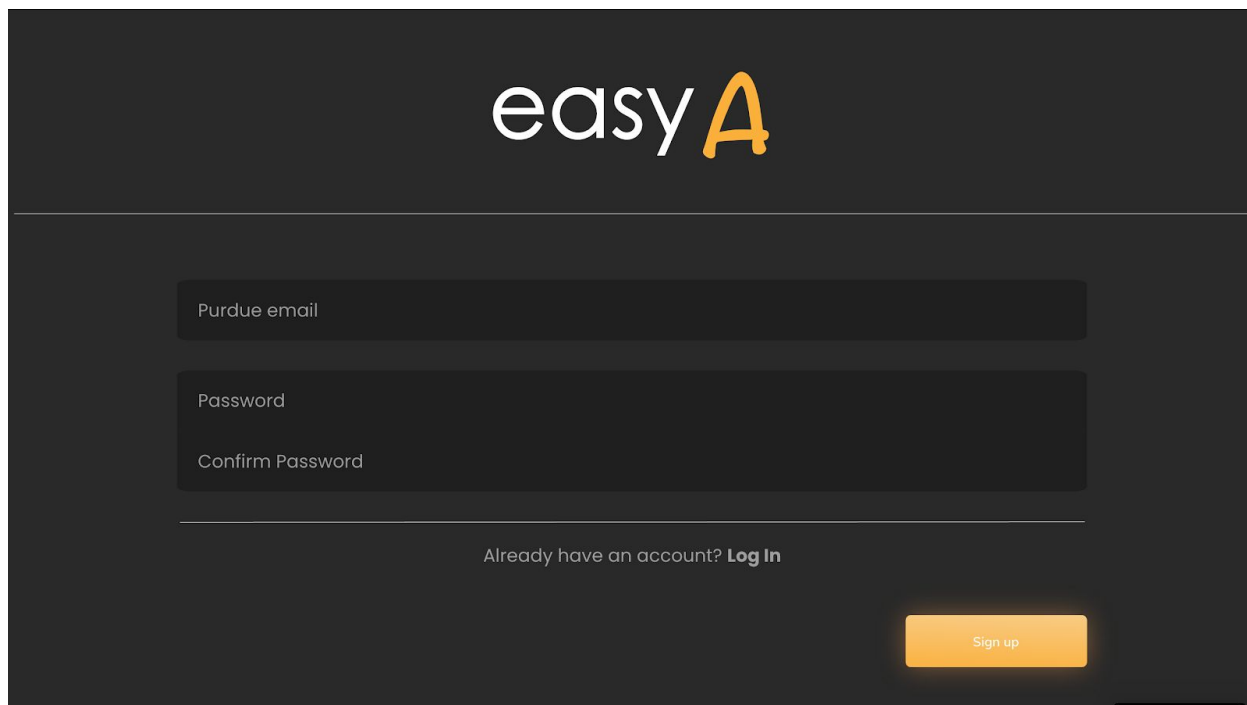
4.4 UI Mockup



Homepage with the search bar, the sign up and log in options.

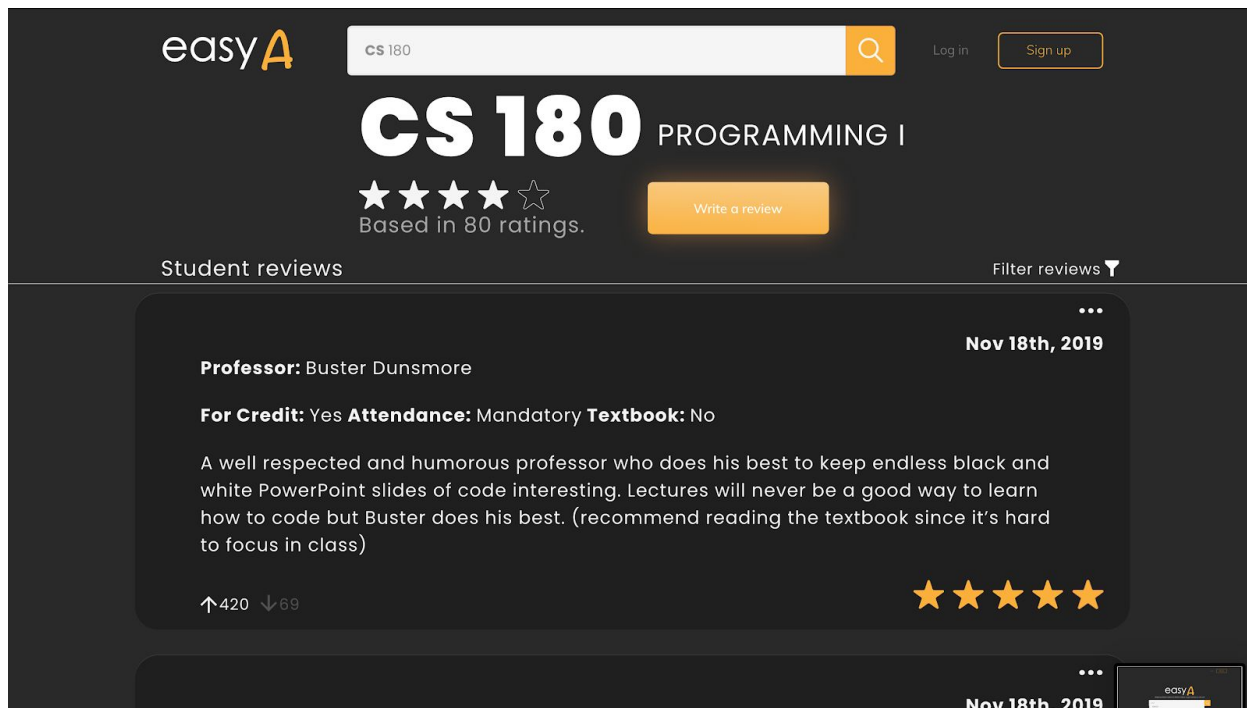


Auto-complete options showing up in the search bar.



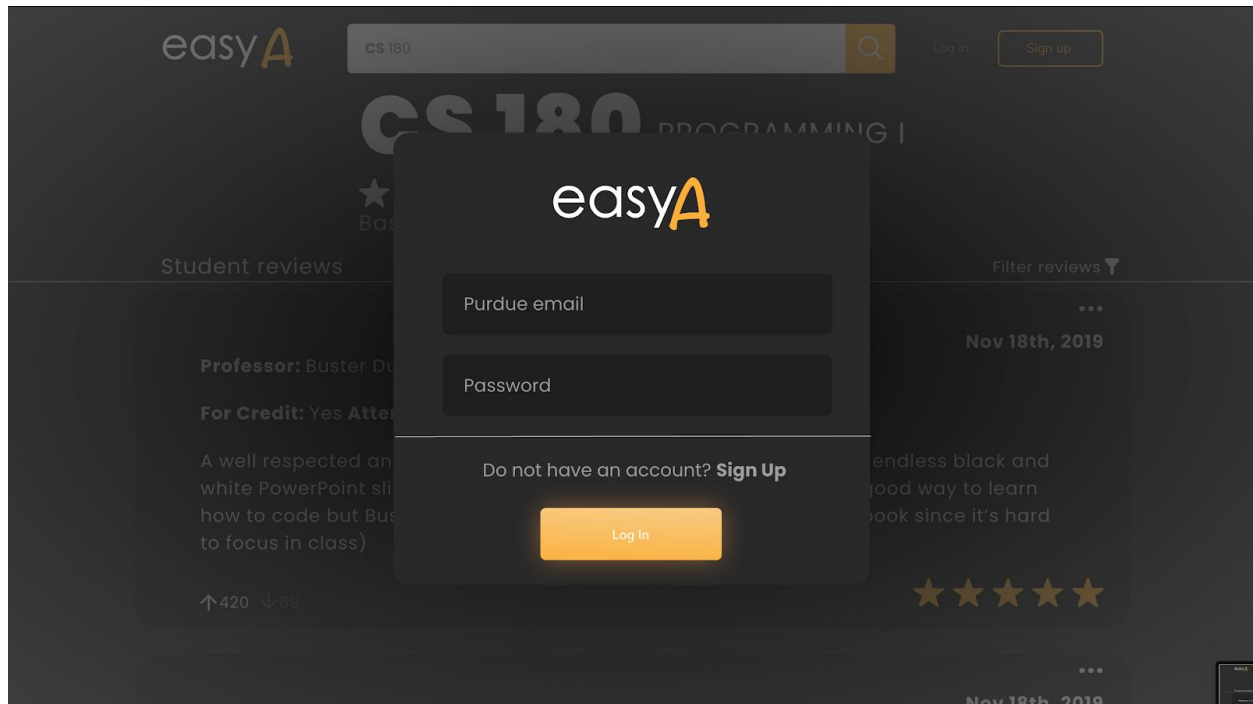
The sign up page features the 'easyA' logo at the top. Below it, there are three input fields: 'Purdue email', 'Password', and 'Confirm Password'. A link 'Already have an account? Log In' is positioned below the password fields. A 'Sign up' button is located at the bottom right of the form area.

Sign up page.

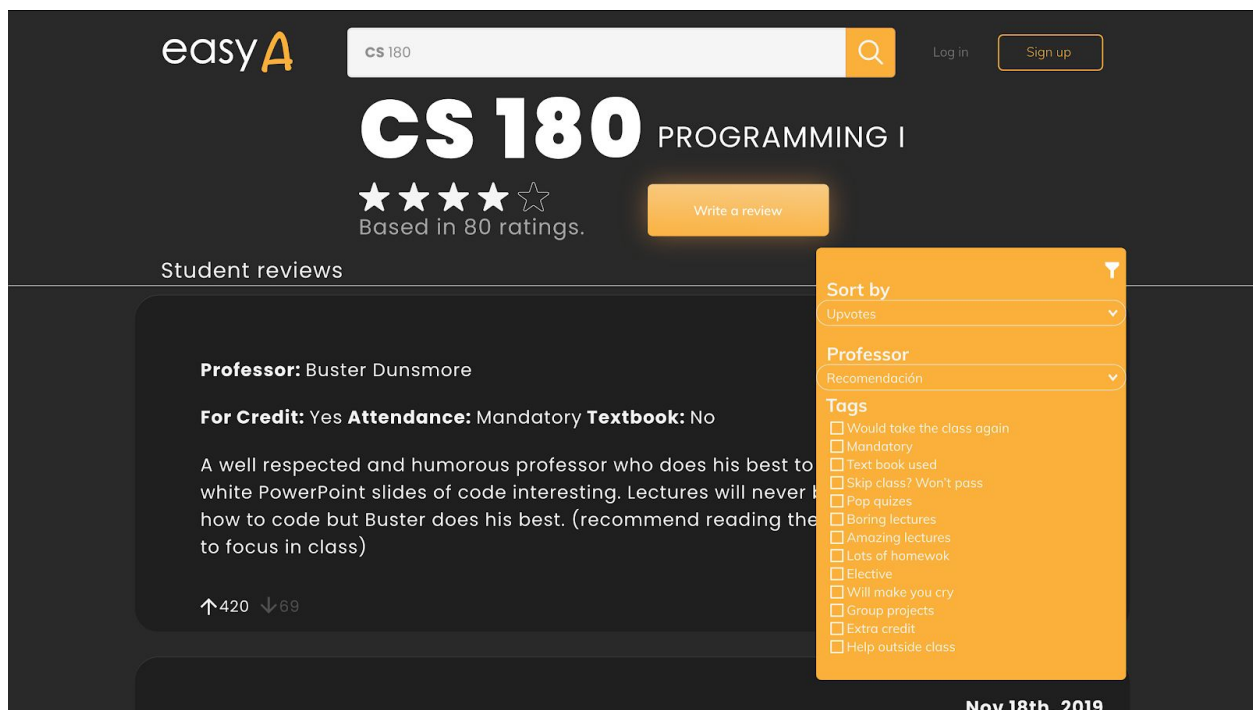


The search results page displays the 'easyA' logo and a search bar containing 'cs 180'. Navigation links for 'Log in' and 'Sign up' are in the top right. The main heading is 'CS 180 PROGRAMMING I', followed by a star rating of 4.5 stars based on 80 ratings, and a 'Write a review' button. Below this, the 'Student reviews' section is shown with a 'Filter reviews' dropdown. The first review is dated 'Nov 18th, 2019' and is by 'Professor: Buster Dunsmore'. It includes details: 'For Credit: Yes', 'Attendance: Mandatory', and 'Textbook: No'. The review text describes the professor as well-respected and humorous, noting that lectures are not the best way to learn but that the professor does his best. The review is rated 5 stars and has 420 upvotes and 69 downvotes. A second review is partially visible at the bottom, also dated 'Nov 18th, 2019'.

Search results showing all the reviews for a class.



Log in pop-up if a guest user tries to upvote, downvote, write a review, write a report, etc.



Filter options on the results page .

The screenshot shows a web interface for 'easyA' with a dark theme. At the top, there's a search bar with 'CS 180' and a magnifying glass icon, and a user profile icon labeled 'ihandal'. Below the header, the course title 'CS 180 PROGRAMMING I' is displayed. The main section is titled 'It's your turn to grade the class!'. It features a form with several input fields and buttons. The 'Professor' field is a text input. The 'Grade' field is a dropdown menu. Below these are several buttons for tags: 'Would take the class again', 'Mandatory attendance', 'Elective', 'Will make you cry', 'Text book used', 'Skip class? Won't pass', 'Lots of homework', 'Group projects', 'Help outside class', 'Pop quizzes', 'Boring lectures', 'Amazing lectures', and 'Extra credit'. A large text area for the review is labeled 'Here's your chance to vent out'. At the bottom left, there are five star icons for rating. At the bottom right, there is a 'Submit' button.

easyA CS 180 ihandal

CS 180 PROGRAMMING I

It's your turn to grade the class!

Professor | Grade ▼

Would take the class again Mandatory attendance Elective Will make you cry

Text book used Skip class? Won't pass Lots of homework Group projects Help outside class

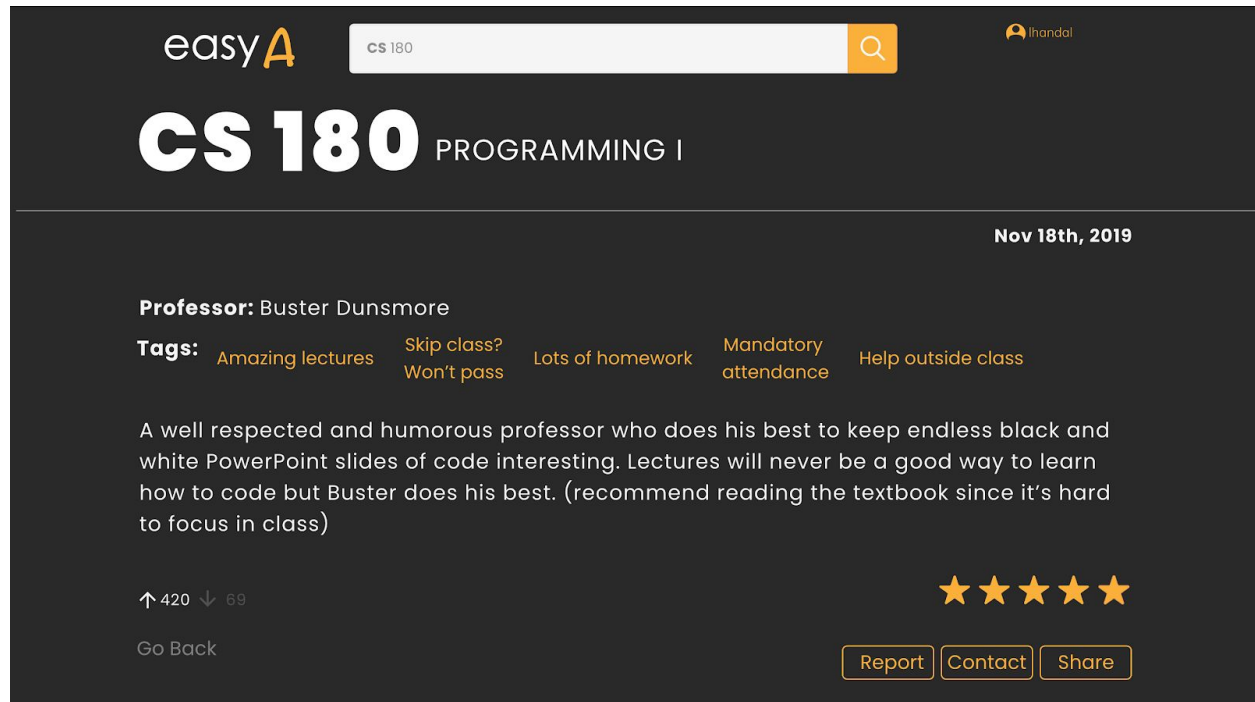
Pop quizzes Boring lectures Amazing lectures Extra credit

Here's your chance to vent out

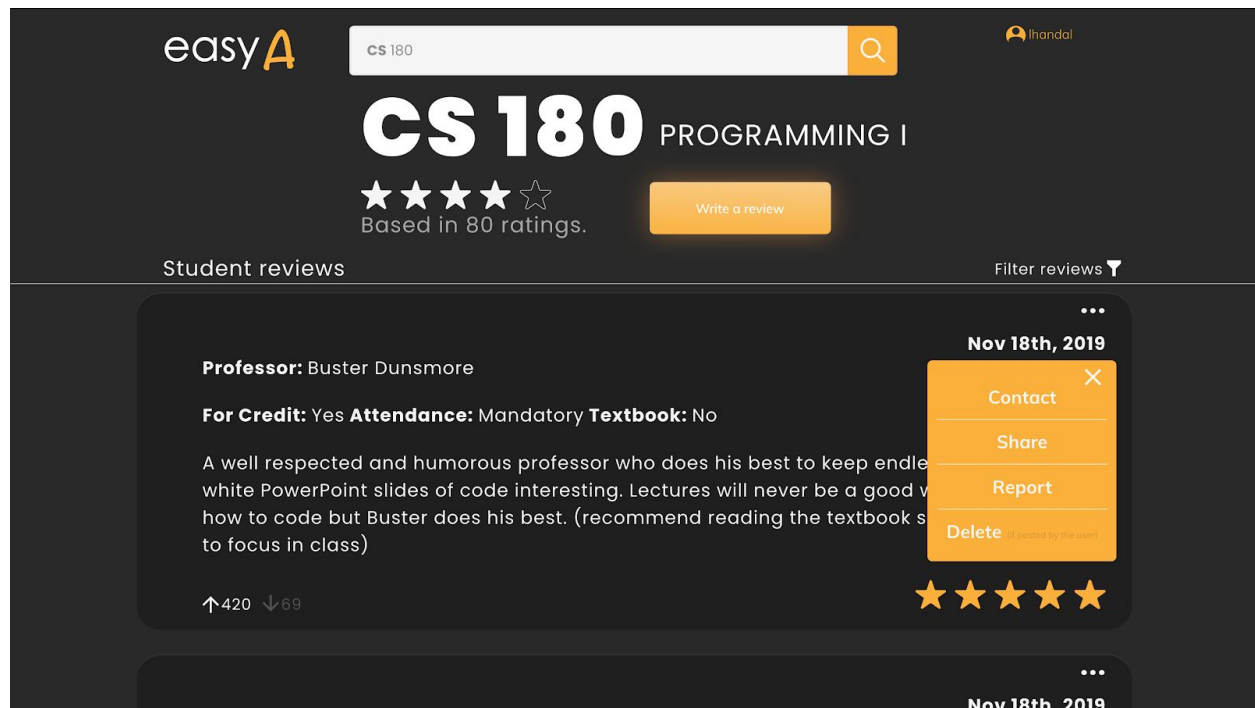
☆☆☆☆☆

Submit

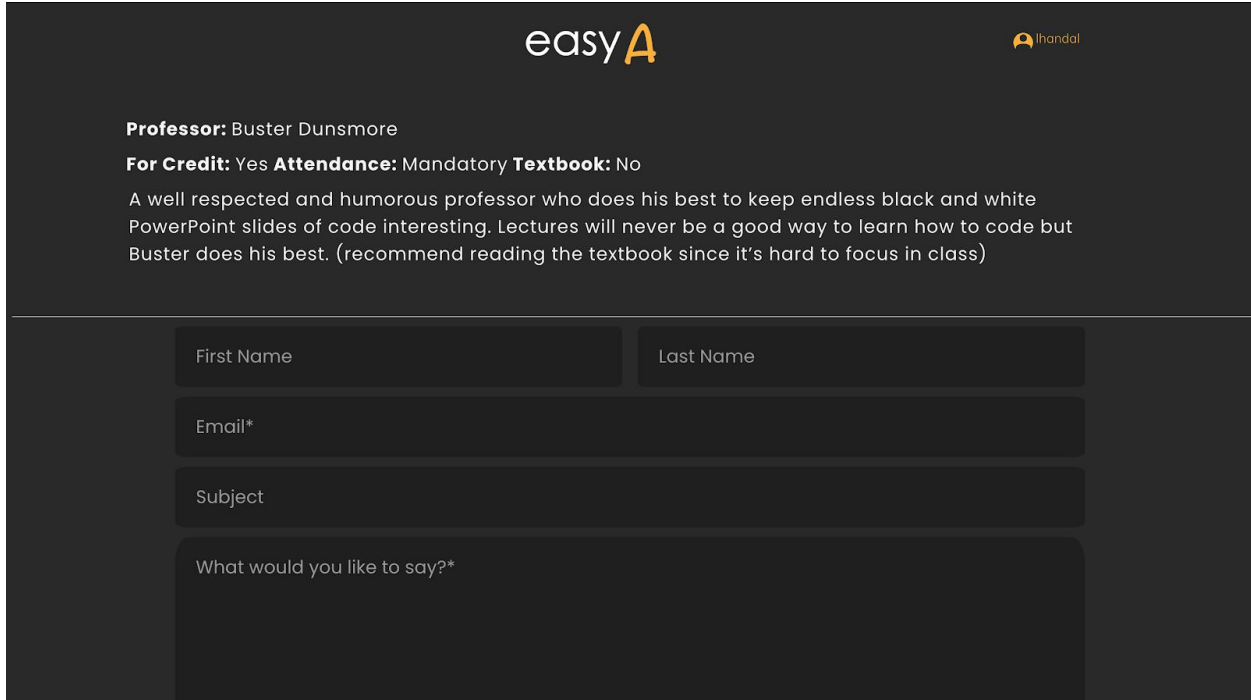
When writing a review the user will have different fields to fill, including the professor name, tags, the actual written review, etc.



Review page that shows up when selecting an specific review.

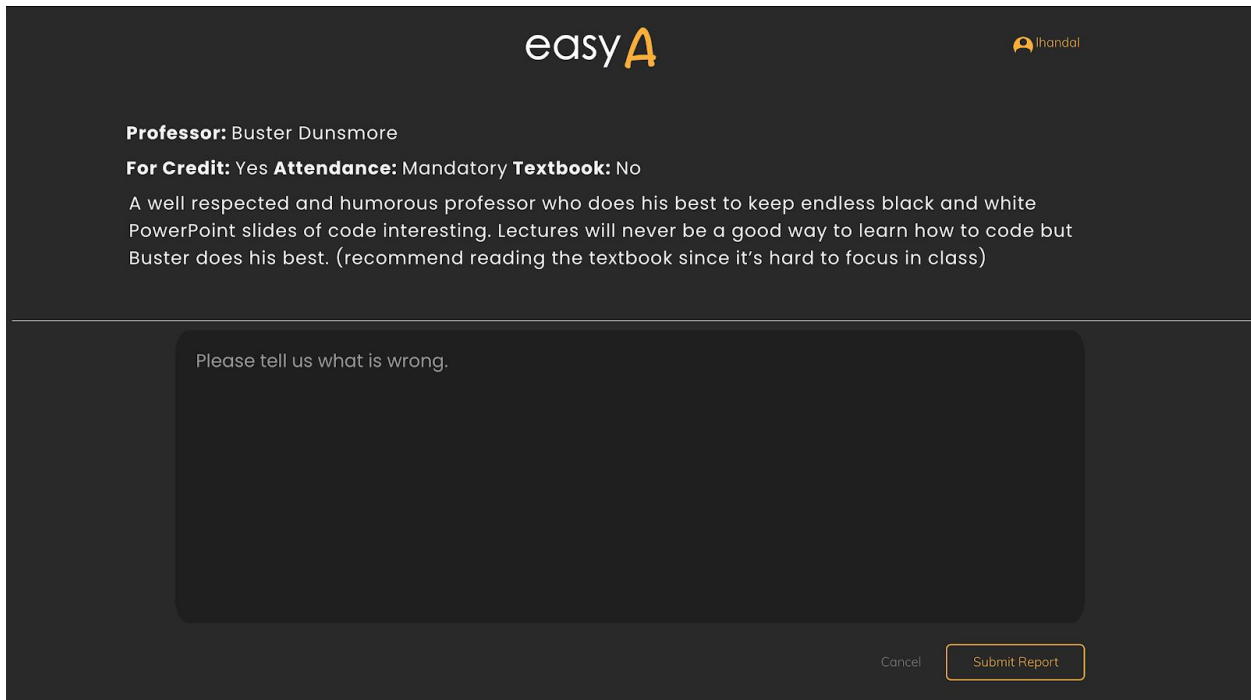


Additional options for the review.



The mockup shows a dark-themed interface. At the top, the 'easyA' logo is on the left and a user profile icon labeled 'ihandal' is on the right. Below the header, the text 'Professor: Buster Dunsmore' is displayed. Underneath, the fields 'For Credit: Yes', 'Attendance: Mandatory', and 'Textbook: No' are shown. A paragraph of text follows: 'A well respected and humorous professor who does his best to keep endless black and white PowerPoint slides of code interesting. Lectures will never be a good way to learn how to code but Buster does his best. (recommend reading the textbook since it's hard to focus in class)'. The main form area contains four input fields: 'First Name', 'Last Name', 'Email*', and 'Subject'. Below these is a larger text area with the placeholder 'What would you like to say?*'.

Contact form for users to reach out to the reviewer with follow up questions.



This mockup is similar to the first one, with the same header and professor information. However, instead of a contact form, it features a large text area with the placeholder 'Please tell us what is wrong.'. At the bottom right, there are two buttons: a 'Cancel' button and a 'Submit Report' button.

Report if the user wants to report an inappropriate reviews.

[Click here to try the interactive version of the mockup.](#)