

# Project: Study Sphere

## Members of the group:

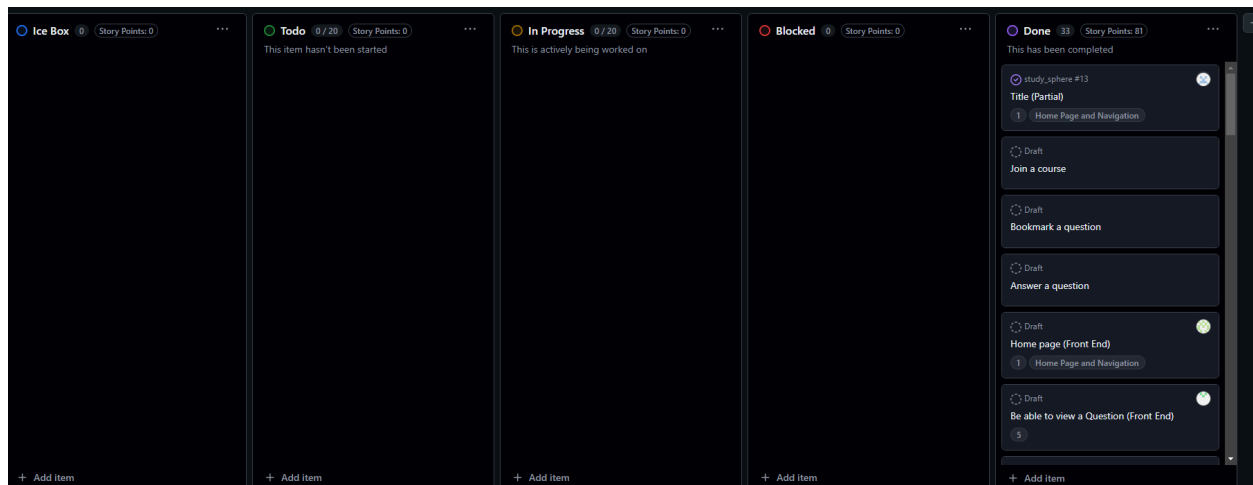
- Diego Castro (*diego-A-castro*), Vibha Joshi (*viyo7860*), Jun Kusano (*juku2287*), Siraaj Sandhu (*siraajsandhu* and *saaaji*, 2nd account) Henry Van Cleave (*heva2420*)

## Description:

Study Sphere is a homework helper website designed for students to help other students in the same classes. Study Sphere is a forum-like website where students can create and/or join classes to interact with others to get their problems fixed quickly. All of the classes house the questions sent in by students for other students to answer, these answers are then ranked based on a voting system wherein the answers with the most 'likes' are pushed to the top so that other students can easily view them. These answers can be bookmarked, saving them to the profile page for quick access in the future. To be able to answer questions or create classes the users must sign in and or make an account on Study Sphere. These users are saved on a database and the passwords are hashed for security purposes. Once the user is signed in they have access to all the features of the website: being able to save class questions, enter the class-specific live chats, and answer questions. With the implementation of cloud storage via AWS, users can upload images in question and answer posts to allow for a more streamlined understanding between users, however, with the understanding of scalability, these images can't be saved to a local database.

## GitHub project board:

- <https://github.com/users/siraajsandhu/projects/1/views/1>



## Video:

- Google Drive link to the video (shared with all):  
[https://drive.google.com/file/d/1Y5cOQS9zbD\\_iRmEKYHXc0ggJS2NAX4X2/view?usp=drive\\_link](https://drive.google.com/file/d/1Y5cOQS9zbD_iRmEKYHXc0ggJS2NAX4X2/view?usp=drive_link)

- Link to unlisted video on YouTube (same as video uploaded to drive, added just in case other link doesn't work): <https://www.youtube.com/watch?v=-WvWn2I-8E>

#### VCS Link:

- **Repository:** [https://github.com/siraaJsandhu/study\\_sphere](https://github.com/siraaJsandhu/study_sphere)

#### Contributions:

- **Diego Castro:** For the development of the website I helped with making parts of the bookmark API route, beginning the register API route and the nav-bar partial. For the bookmark route I helped in making the base code for it and writing the SQL queries which would remove or add the question to the users to the bookmarks table. In the very beginning of the project I also added the base of the nav-bar partial which was improved by SiraaJ. I also started the register API route which was also later improved by other members to add more functionalities.
- **Vibha Joshi:** I created the tables to store user, question, answer, and course information, making sure to link related tables accordingly. I also adjusted the CSS of the website so that everything was cohesive and matched the purple and white theme of StudySphere. Additionally, I developed the create class and search bar functionality of the page, which was later improved by SiraaJ.
- **Jun Kusano:** I developed both the front-end and back-end of the profile page. Additionally, I created the API route for user login and built the front-end interface for posting questions. I utilized technologies such as HTML, CSS, Handlebars, and SQL across these pages to ensure consistency and functionality. By applying the DRY (Don't Repeat Yourself) principle to the profile page, the code of the page is simple and easy to read.
- **SiraaJ Sandhu:** Developed front- and back-end of the register and login routes, including registration preferences. Worked on the front- and back-ends for the home, class, and question pages and added additional functionality to the search bar back-end. API endpoints included viewing class live chat, creating classes, and answering questions. Fixed general errors with the app including adding SASS/SCSS to make sure modifications to Bootstrap CSS would apply correctly and issues with posting new questions to classes. Implemented AWS API (Amazon) to upload and fetch images attached to posts from cloud. Implemented front- and back-end for like & dislike actions on answers.
- **Henry Van Cleave:** I created the front end for the login, home, and course pages. Regarding the back end, I worked on the course and fixed many smaller errors in other sections. I worked on the CSS page so that the website would not look terrible. The majority of my time was spent on bug-fixing the website other than the main pages that I produced. I thought it would be a good idea to have the ability for the users to browse the website without having to sign in, I noticed that this is a feature that almost all websites out there utilize.
- Some screenshots of the git log have been added below, but full commit history is available in the actual git repository:

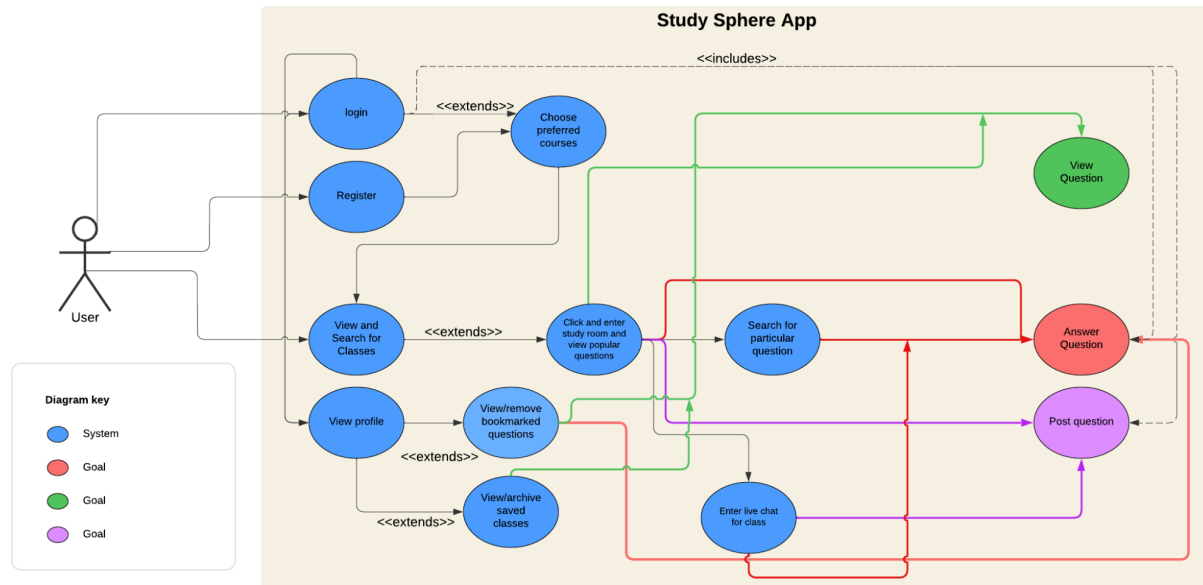
added to home page and added ability to search for /create a new class	92b3142	<>
• saaaji committed 3 weeks ago		
create classes (fixed merge conflicts)	ae98d7a	<>
• vijo7860 committed 3 weeks ago		
class page	594d677	<>
• juku2287 committed 3 weeks ago		
class page front-end adn back-end	23423b8	<>
• juku2287 committed 3 weeks ago		
Merge pull request #29 from siraajsandhu/logout	2a783bb	<>
• siraajsandhu authored 3 weeks ago		
implemented logout route	7473f8a	<>
• saaaji committed 3 weeks ago		
Merge pull request #28 from siraajsandhu/siraaj_profile_updated	a48b8fa	<>
• siraajsandhu authored 3 weeks ago		
merged with main	52eaa3f	<>
• saaaji committed 3 weeks ago		
fixed classes page. can now delete classes from profile	233d915	<>
• saaaji committed 3 weeks ago		
fixed problems with search/profile	7997746	<>
• saaaji committed 3 weeks ago		

added profile front-end and back-end	136d743	<>
• juku2287 committed on Nov 4		
API route for profile is added	ed5c8e5	<>
• juku2287 committed on Nov 4		
updated index.js with register API routes	f5233f3	<>
• diego-A-castro committed on Nov 4		
nav-bar with dropdown menu, title, and username displaying	e521bf3	<>
• diego-A-castro committed on Nov 4		
Merge pull request #8 from siraajsandhu/Henry_Login_Register_F	e643c29	<>
• heva2420 authored on Nov 4		
At this point im just trying different things until it decides to work. I changed to a new terminal, and things are moving smoother now.	2e7d72d	<>
• heva2420 committed on Nov 4		
Henry VC: trying to get this to merge into main	1a8ff6f	<>
• heva2420 committed on Nov 4		
Merge branch 'main' into jun_profile_f	4bd25a2	<>
• juku2287 committed on Nov 4		
began implementing registration preferences, frontend page and backend API route. Users can search course names and add or remove them from their list of preferences. Submitting will add these cour...	1dac3e5	<>
• saaaji committed on Nov 4		

Commits on Dec 4, 2024		
merged with main	43603cc	<>
• saaaji committed 5 days ago		
merged with main/css changes	885635e	<>
• saaaji committed 5 days ago		
buttons	d4b8e55	<>
• vijo7860 committed 5 days ago		
nav bar button	8709e4d	<>
• vijo7860 committed 5 days ago		
borders	a8039d6	<>
• vijo7860 committed 5 days ago		
chatting is mostly done. users can join the chat for their respective class	b8698af	<>
• saaaji committed 5 days ago		
--	48f428e	<>
• vijo7860 committed 5 days ago		
removing borders of buttons	5667c08	<>
• vijo7860 committed 5 days ago		

## Use Case Diagram:

- We identified three general use cases or “goals” for the application, depicted below in the use case diagram: view question, answer question, and post a question. We planned API routes and systems accordingly to help users reach these end goals, including login and registration pages to enable user interaction with the site and generation of content, home and class pages to assist in organization and easy access of questions through search features, and a profile page to help users organize questions they were either interested in or had asked themselves.



## Test results:

- **Test Plan and Results:** We decided to test four different endpoints of our application using 10 total unit tests to handle potential edge cases: the /register, /login, /create, and /profile endpoints. The tests are described below as well as their type (positive or negative) and success criteria. The 11th unit test depicted in the screenshot below is from the dummy endpoint implemented in Lab 11.
  - /register
    - *(positive)* Successful creation of account with valid new user account info. Expect HTTP status 200 and successful redirect
    - *(negative)* Unsuccessful creation of account after submitting invalid password length. Expect HTTP status 400 and successful redirect
    - *(negative)* Unsuccessful creation of account after submitting existing username. Expect HTTP status 400 and successful redirect
    - *(negative)* Unsuccessful creation of account after submitting invalid username length. Expect HTTP status 400 and successful redirect
  - /login
    - *(negative)* Unsuccessful login after entering invalid account credentials. Expect HTTP status 400 and successful redirect.
    - *(positive)* Successful login after entering valid account credentials. Expect HTTP status 200 and successful redirect to /profile
  - /create
    - *(positive)* Successful creation of a new class after entering valid class info. Expect HTTP status 200 and successful redirect to new class page
    - *(negative)* Unsuccessful creation of a new class after entering existing class info. Expect HTTP status 400 and successful redirect
  - /profile

- *(positive)* Successful password change after submitting valid user credentials and new password. Expect HTTP status 200 and successful redirect
- *(negative)* Unsuccessful password change after submitting invalid user credentials and new password. Expect HTTP status 400 and successful redirect
- **Screenshot of Passing Results (11 out of 11):**
  - Due to the amount of debug info printed in the terminal the full output is not particularly useful. Detailed description of all test cases is presented above.

```

1      ✓ positive: /create (1)
1      ✓ negative: /create (1)
1
1      Testing /profile/update API
1      ✓ positive: /profile/update (1) (142ms)
1      ✓ negative: /profile/update (1) (72ms)
1
1
1      11 passing (644ms)
1

```

- **Observations**
  - `./register`: Users are creating a new profile on the register page. This is consistent with the use case because users will be prompted to provide info that will be used to update the user database. When a user provides an invalid input when registering (for example, their password is an invalid length), there is a deviation from the expected action. As a solution, there is a message that informs the user their username or password don't meet the necessary requirements.
  - `./create`: Users are creating a new class because the class they're looking for does not currently exist in the database. This is consistent with the use case because users can add to the course database if it does not already exist; in other words, it is how the course data table can grow to fit user preferences. However, there is a deviation from the use case if the user enters information for a class that already exists in the course table. To account for this, a message will display informing the user that the class they wish to create already exists in the database.
  - `./login` and `./profile` did not have significant issues that warranted changes to the application

## Deployment:

- **Deployment link:** <https://study-sphere-res6.onrender.com>
- The app was deployed using Render's deployment service. Successful deployment entailed creation of a web service and an external SQL database service, the first of which had to be configured with correct environment variables, including PostgreSQL credentials and AWS API keys. All code was automatically retrieved from our remote repository but initializing the database required executing a local shell script that connected to the database service and created and modified all SQL tables as needed. Example usage is depicted in our video demo, and it may be accessed through the deployment link above.




- Please note that by their policy Render's free service will slow down after periods of little use and is especially slow with uploading images to AWS. For testing it is advisable to use smaller images.

### Wireframes:

- These wireframes were developed during our initial proposal process. Each page has been sketched below.

Login page

 Study Sphere

Login


username:

password:

[register an account](#)

---

Register page

 Study Sphere


Register

username:

password:

Confirm Password:

After registration, choose classes for profile

 Study Sphere

Add classes:


Search for class:

1300
2270
2400

Chosen classes:

---

Home Page →

 Study Sphere [username/login](#)

Search for classes:

1300	2270
image	image

Popular classes

2400	3308
image	image

Class

☰

Study Sphere

CSC11300

user/login

Search for question ...

GO

...

...

...

Post New Question

Popular Questions

?	...
?	...
?	...
?	...

Chat

Chat page

☰

StudySphere

CSC11300

Chw

User1	...
User2	...
User1	...
User7	...
User2	...
User1	...

Send message...

SEND

New Question Page

☰

StudySphere

CSC11300

user/login

Write your question...

Upload images:


images:

Image 1

Submit



## Profile Page

 Logout

User Name

My Bookmarks

Search through bookmarks

Question 1	...	Remove
Question 2	...	Remove
Question 3	...	Remove
Question 4	...	Remove
Question 5	...	Remove

My Classes


1300 ☹


2400 ☹

3308 ☹

Change Username / password

## Question Page

 Study Skills ESC11300 user/login

☒ ☐ Question: bookmark 

< question content >

image1.png

image2.png

New answer

Write your answer here...

Upload Images

Images:

Answer.png ☹

Post

☒ ☐ Answers

< answer content >

Image.png

☒ ☐ < answer 2 content >

Image2.png