

StudyManager

Group 32

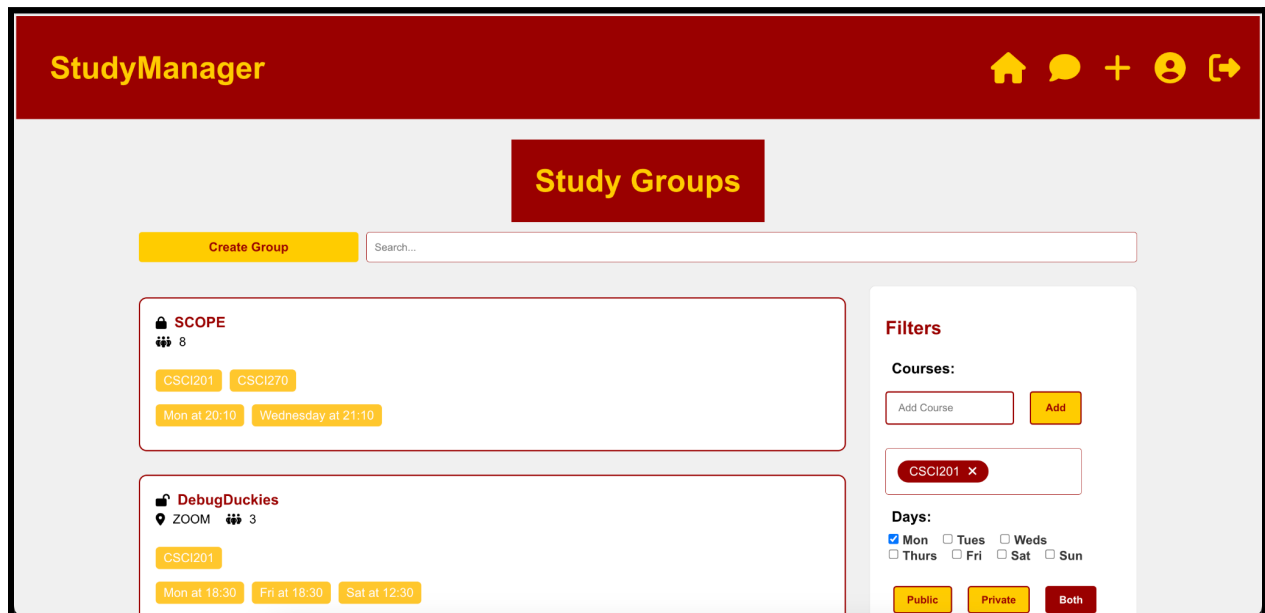


Table of Contents

Table of Contents	2
CSCI-201 Group 32 Project Proposal	3
CSCI-201 Group 32 High-Level Requirements	4
Overview	4
1.0 User Management:	4
2.0 Study Group Management:	4
3.0 Chat & Communication:	5
4.0 Web Interface & Aesthetics:	5
CSCI Group 32 - Technical Specifications	6
1.0 User Management (2 people, 8-12 hours):	6
2.0 Study Group Management (2 people, 35 hours):	6
3.0 Chat & Communication (2 people, 8-12 hours):	7
4.0 Web Interface & Aesthetics (2 people, 8-12 hours):	8
CSCI Group 32 - Detailed Design	9
1.0 User Management	9
2.0 Study Group Management (except for UI)	14
3.0 Web Interface & Aesthetics:	19
4.0 Chat & Communication:	21
CSCI Group 32 - Testing Plan	25
1.0 User Management	25
2.0 Study Group Management	27
3.0 Web Interface & Aesthetics	29
4.0 Chat and Communication	31
CSCI Group 32 - Deployment Document	33
Contact Information	35

CSCI-201 Group 32 Project Proposal

Project Proposal:

USC StudyManager

It would be great if students could have a way to easily find and join study groups with other students who share similar courses, study habits, and availability. The StudyManager would act like a social network but solely for academics and group studies. This would eliminate the hassle of searching for study partners and help foster a collaborative learning environment. Implementing the project as a web application would make the forum simpler to navigate.

CSCI-201 Group 32 High-Level Requirements

Overview

This web application's objective is to enable students to engage in and create study sessions. This fosters a more productive environment, allowing them to gain a deeper understanding of the subject compared to just studying alone. A community dedicated to studying not only enhances academic achievement but also promotes collaboration among peers.

1.0 User Management:

1.1 Registration

- Users need to create an account with their name, email, major, and password.

1.2 Login

- Users should be able to log in through their email.
- Users can choose to log in as a guest as well, which will not require an email but should give limited access to the website (Unable to access chat and join groups).

1.3 Profile Management

- Users should be able to see the profile information they submitted once registered.

2.0 Study Group Management:

2.1 Study Group Creation

- Authenticated students can create their own study groups.
 - Students can choose to create a private study group or a public study group
 - Study groups should include the course/courses they are hoping to study.
 - Each study group should display meeting times and the location where they will be meeting whether it be a room number, address, or online.

2.2 Public Study Groups

- Each study group should have its own page.
 - Authenticated students should be able to view and join various study groups.
 - Guest users should only see the study group on the homepage, which shows limited information.
 - The users who are part of the study group should be shown on the side.

2.3 Private Study Group

- Private study groups should act the same as public study groups, but can only be joined via a code.
- Confidential information like the name of users should be hidden.

2.4 Study Group Discovery

- On the home page in the web application, we would like our application to display a list of available public study groups.
 - The study groups should display the courses being studied, time, number of people in the group, and location in a minimized format on the homepage.
 - Users should be able to sort and search through the list
 - Underneath these public results, private study groups that you are enrolled in are displayed

3.0 Chat & Communication:

3.1 Real-Time Chat

- An open chat room is available to users who are logged in.
 - Messages will be displayed on a continuous wall.
 - The user's username will be displayed near the messages that they send.
 - Old messages can be displayed by scrolling up the chat interface.

4.0 Web Interface & Aesthetics:

4.1 Aesthetics

- Ensure the website's aesthetics are visually appealing by implementing:
 - Thoughtfully chosen fonts
 - Engaging layout designs
 - A logo that reflects the purpose of our site
 - Choosing a theme to reflect our USC demographic

4.2 Intuitiveness

- Prioritize an intuitive interface to enhance user experience, including
 - Streamlined navigation
 - User-friendly features

CSCI Group 32 - Technical Specifications

1.0 User Management (2 people, 8-12 hours):

- Frontend:
 - Utilize ReactJS to create a responsive and user-friendly interface for registration, login, and profile management
 - Registration page - a form with fields for student email, name, major, and password.
 - Login page - input fields for the user's email and incorporate a "Login as Guest" option
 - Profile Management - displays the user's current profile information from registration.
 - Study Groups - provide a section where users can view a list of study groups.
- Backend:
 - Information collected during the registration process will be stored in MySQL.
 - The database should contain fields for student email, name, major, and courses
 - The information from the database will be retrieved and updated when students edit their profile information, such as their major and courses.
 - Java will serve as the intermediary between the front-end application and database to handle the request. It will also run the APIs for the frontend to call
 - It will perform any necessary validations and update the entry in the database with the new information provided by the student
- User Interface:
 - The login page, registration page, and profile information editor page should all be designed with the aid of Figma

2.0 Study Group Management (2 people, 35 hours):

- Frontend:
 - Authenticated users should see a "Create Study Group" button on the home page.
 - On the creation pages there should be a field for the following items:
 - Name of the a unique study group without spaces
 - An input field for people to insert multiple courses.
 - A short-field textbox for the location.
 - A formatted input field to select the meeting times on a day-time system.

- A toggle to switch between private and public groups with a code field for private groups.
 - There should be a submission button at the bottom to publish the study group.
 - After creation the user should be taken to the page of the new study group.
- A page that will display the public study groups with name, privacy, location, courses, time, and number of students.
- Guest users should not be able to access study group pages nor create them, only view them on the home screen.
- Backend:
 - Access to private study groups will require an access code that is validated against the database.
 - Each study group should have its own ID. Additionally, if a study group is private, users should be able join by entering the correct code.
 - Each study group should have its own entry in a database with the following keys. UUID, name, visibility, location. And then there should be 2 reference tables using foreign keys. One table should include the day and time frame for each study group. And another table of the courses.
- User Interface:
 - There should be a search bar to search based on study groups. This should be a close match that compares the entire string.
 - On the homepage each study group should have its own entry, displaying their information. Guest users should only see the study group's name, courses, and time.

3.0 Chat & Communication (2 people, 8-12 hours):

- Frontend:
 - Implement the chat interface using ReactJS, having Figma as a way to draft designs.
 - Show the username of the sender beside the message using document cookies.
 - Create a map that keeps track of the placement of messages.
 - Javascript to set up real-time messaging
 - Check if the user is logged in or not so they can access the chat.
 - AJAX calls Pusher to send messages to all clients currently in the chat.
- Back-End:
 - MySQL and Java Servlets

- Messages are stored with keys for the message itself and the sender's UserID
- Old Messages are retrieved and displayed when the chat is reloaded.

4.0 Web Interface & Aesthetics (2 people, 8-12 hours):

4.1 Aesthetics & Intuitiveness

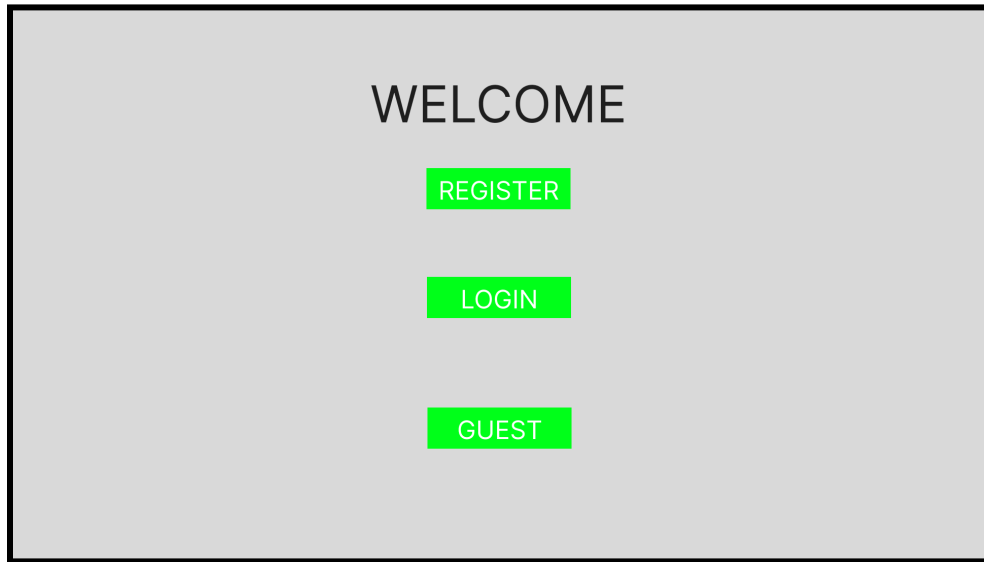
- Implement an aesthetically pleasing as well as user-friendly interface using Figma
- When users navigate to our website, the logo will appear along with the sign-in button.
 - All of the necessary fields will be clickable so the user can sign in
- Once they sign in they will be able to see our user-friendly layout, making it easy to navigate through the website.
 - At the top of the page will be our logo with a header that says "Homepage"
 - Underneath the header will be the "Create Study Group" button
 - Using Figma, this button will be clickable to help the user navigate to the page that will yield their desired outcome
 - The home page will also show the list of both public and private study groups the user is a part of, distinguishing them as "Public" and "Private"
 - This will also use Figma by showing each study group as a clickable icon
 - Clicking on a study group's icon will take the user to the study group's home page
- An Icon that allows users to navigate to their account info
- Next to the drop-down menu, there will be a search icon
 - Users will be able to click on this to search for study groups they would like to join or are already a part of
- Our website will have a very specific color scheme to it as well
 - The headers will be in yellow with a red background to be USC-themed
 - Any other writing will be in black

CSCI Group 32 - Detailed Design

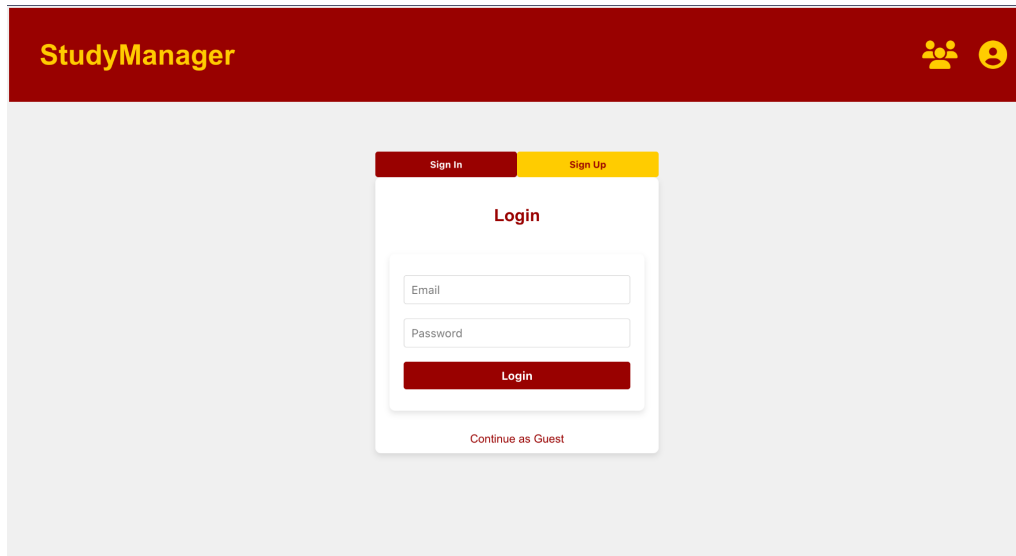
1.0 User Management

Frontend Design

Wireframe:



Final:



1.1 Registration Page - Class Name: Registration Page

Wireframe:

REGISTER

<input type="text" value="FIRST NAME"/> <input type="text" value="LAST NAME"/> <input type="text" value="MAJOR"/> <input type="text" value="EMAIL"/>	<p>CURRENT COURSES</p> <div>EE 259 <input checked="" type="checkbox"/></div> <div>CSCI 104 <input checked="" type="checkbox"/></div> <div>WRIT 240 <input checked="" type="checkbox"/></div> <div>NEW <input checked="" type="checkbox"/></div>	<p>PREVIOUS COURSES</p> <div>EE 259 <input checked="" type="checkbox"/></div> <div>CSCI 104 <input checked="" type="checkbox"/></div> <div>WRIT 240 <input checked="" type="checkbox"/></div> <div>NEW <input checked="" type="checkbox"/></div>
---	---	--

Final:

StudyManager

Sign In
Sign Up

Register

[Continue as Guest](#)

- info.Email- variable of String type to hold the student's email address
- info.firstname- variable of String type to hold their student's first name
- info.lastname- variable of String type to hold their student's lastname
- info.major- variable of String type to hold the student's major

1.2 Login Page - Class Name: LoginPage

Wireframe:

A simple login form wireframe on a light gray background. At the top center is the word "LOGIN" in a large, black, sans-serif font. Below it is a white rectangular input field with the word "EMAIL" in small, black, uppercase letters on the left side. Underneath the input field is a bright green rectangular button with the word "Login" in white, centered text.

Final:

A final design for a login form. At the top is a dark red header bar. On the left of the header is the text "StudyManager" in yellow. On the right are two yellow icons: a group of three people and a single person. Below the header is a light gray area containing a white login card. The card has a dark red "Sign In" button and a yellow "Sign Up" button at the top. Below these is the word "Login" in dark red. Underneath are two white input fields, one labeled "Email" and one labeled "Password". Below the fields is a dark red "Login" button. At the bottom of the card is a link that says "Continue as Guest" in small, dark red text.

- email - variable of String type to hold the value of the email field
- password - variable of String type to hold the value of the password field
- handleGuest() - method to log in the user as a guest

1.3 Profile Management - Class Name: ProfileManagement

Wireframe:

ACCOUNT MANAGEMENT

CURRENT COURSES

EE 259 ☐

CSCI 104 ☐

WRIT 240 ☐

NEW ☐

Final:

Account Info

Account Information

First Name: Bad

Last Name: Bruin

Email: BadBruin@usc.edu

Major: EE

- Profile - object that will contain specific user information
 - name - variable of String type to hold the user's name
 - major - a variable of String type to hold the user's major information
 - currentCourses - a list that contains the user's current courses' information

Backend Design

1.1 Registration Page

- addToDatabase(info.Email, info.FirstName, info.LastName, password): add data collected during this step into a new database entry

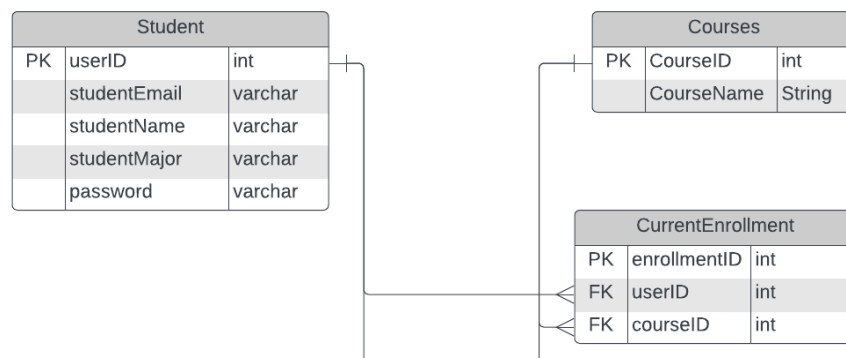
1.2 Login Page

- fetchUsername(email, password): fetches an entry in the database if it finds a matching user, calls checkPassword(password), and sets the “loggedIn” variable to true. Otherwise, it sends an error message.
- checkPassword(password): checks that the password entered matches what is in the database. Otherwise, it sends an error message.
- If no user is authenticated at this, various checks throughout the site will provide guest functionality for the rest of the session.

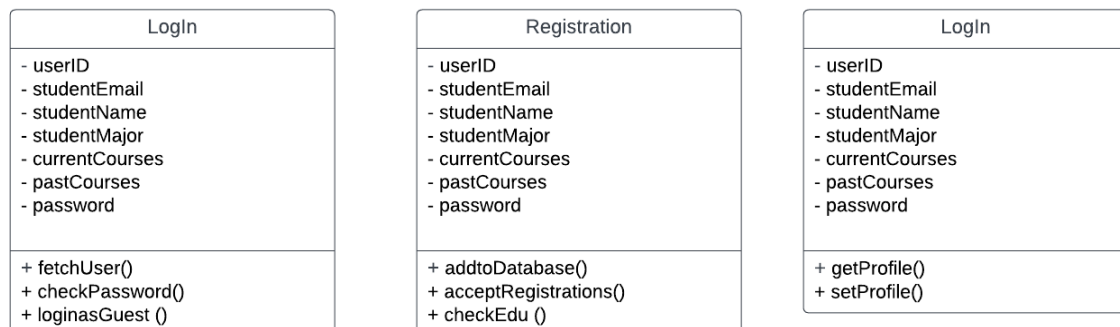
1.3 Profile Management

- getUserAccount(username): fetches the profile information for a logged-in user to display
- User(info.Email, info.FirstName, info.LastName, Major): Updates the profile information in the database when the user hits save.

User Management Database



User Management Class Diagram

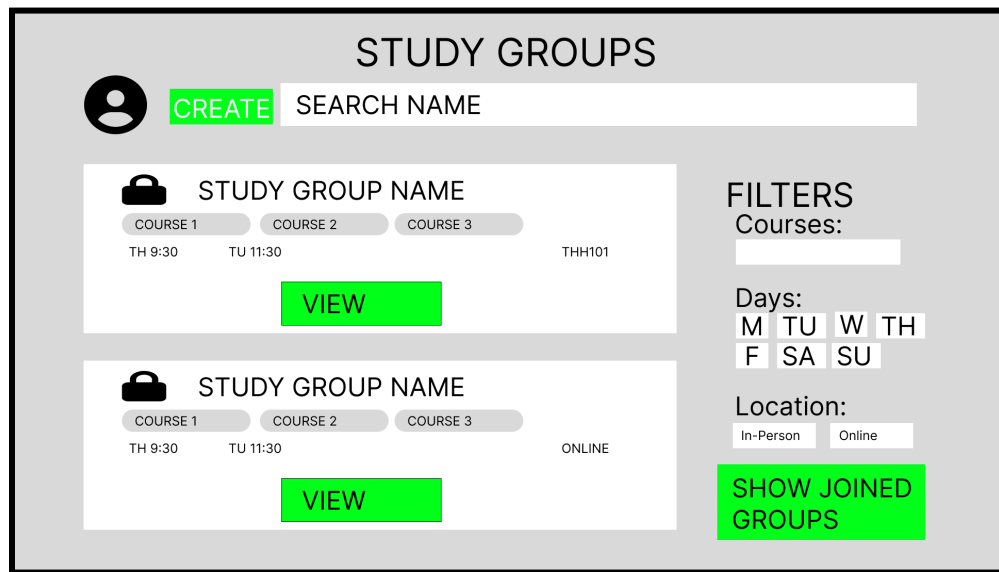


2.0 Study Group Management (except for UI)

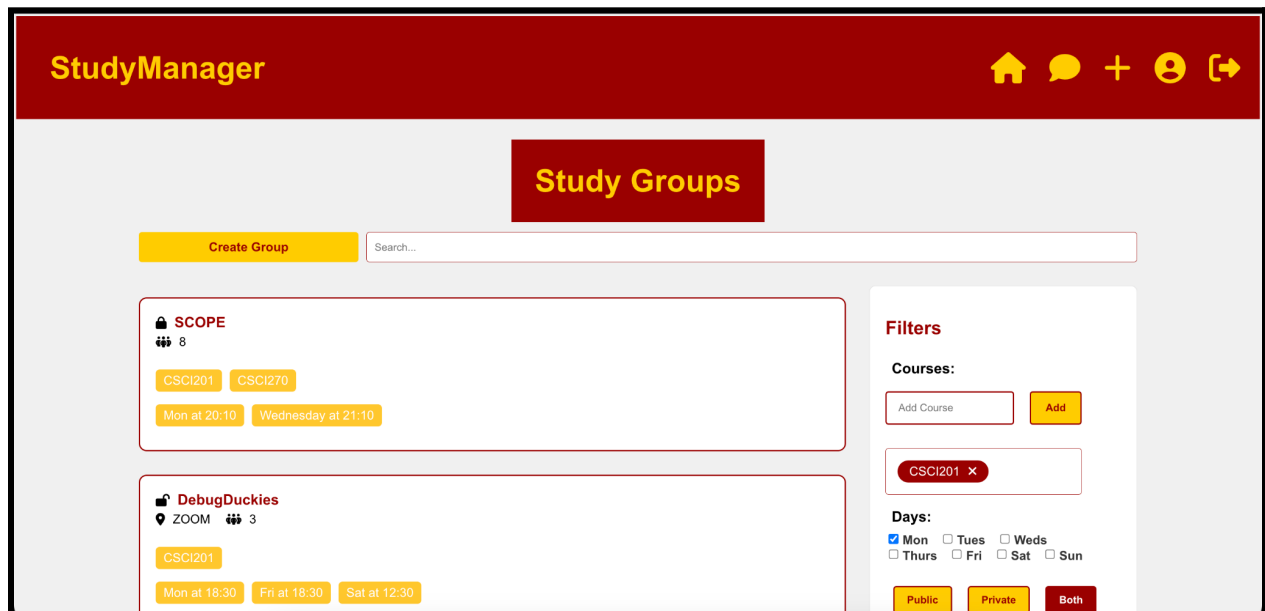
Frontend Design

2.1 Home Page - Class Name: HomePage

Wireframe:



Final:



- createStudyGroupButton: This variable represents a button that allows users to create a new study group. When clicked, it triggers an event that navigates to the Study Group Creation Page.
- createStudyGroup: This method is triggered from createStudyGroupButton. It activates an event that navigates to the Study Group Creation Page.

2.2 Study Group Creation Page - Class Name: StudyGroupCreationPage

Wireframe:

CREATE GROUP

STUDY GROUP NAME

COURSES:

CSCI-102

CSCI-101

CSCI-201

MEETING TIMES:

M

TU

11:30AM

W

TH

F

SA

2:00PM

SU

LOCATION:

ONLINE

ROOM

PRIVACY:

PUBLIC

PRIVATE

CODE

CREATE

Final:

The image shows a web form for creating a study group. It contains the following elements:

- A text input field containing "USCRocks".
- An "Add Course" button.
- A course selection dropdown menu with "CSCI201" selected and a close button (X).
- A day/time selector with a dropdown showing "Mon", a separator "--:--", and an "Add Day & Time" button.
- A time selection dropdown menu with "Tues at 15:00" and "Thurs at 16:30" selected and close buttons (X).
- A location input field containing "GFS101".
- A "Public" button and a "Private" button.
- A "SECRET" input field.
- A large "Create Study Group" button at the bottom.

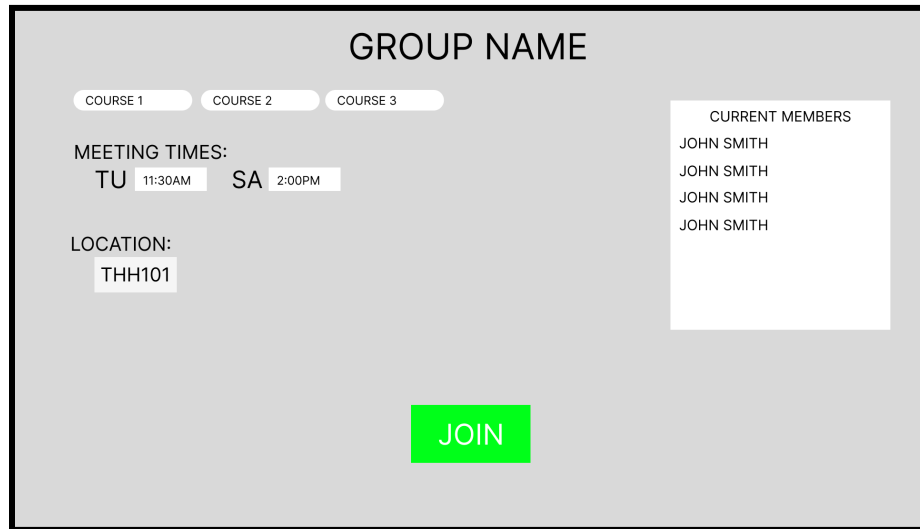
- `group_name`: This variable represents an input field where the user can enter the name of the study group they want to create. Its value can be accessed through a getter method.
- `isPrivate`: A variable representing whether the study group will be private or public. It has a method to determine the current state, which returns either a boolean to indicate whether a study group is public or not.
- `location`: A variable for the text field where users can specify the location of the study group. It has a method to get the user's input, which returns a list of study groups.
- `timeFrameInput`: This variable represents an input field where the user can specify the day of the week and time frame for the study group. It includes a method to process the input, which will return a list that contains the study groups that were found.
- `submitButton` (`button[type= "submit"]`): A variable that is associated with the submit button that, when clicked, triggers the submission of the study group creation form. It has an associated event handler to handle form submission.

2.3 Study Group Creation Success Page - Class Name: `StudyGroupCreationSuccessPage`

- successMessage: A method that is used to display a success message to the user after a study group has been successfully created. It can set and display the string that indicates that a user has successfully created a study group page.

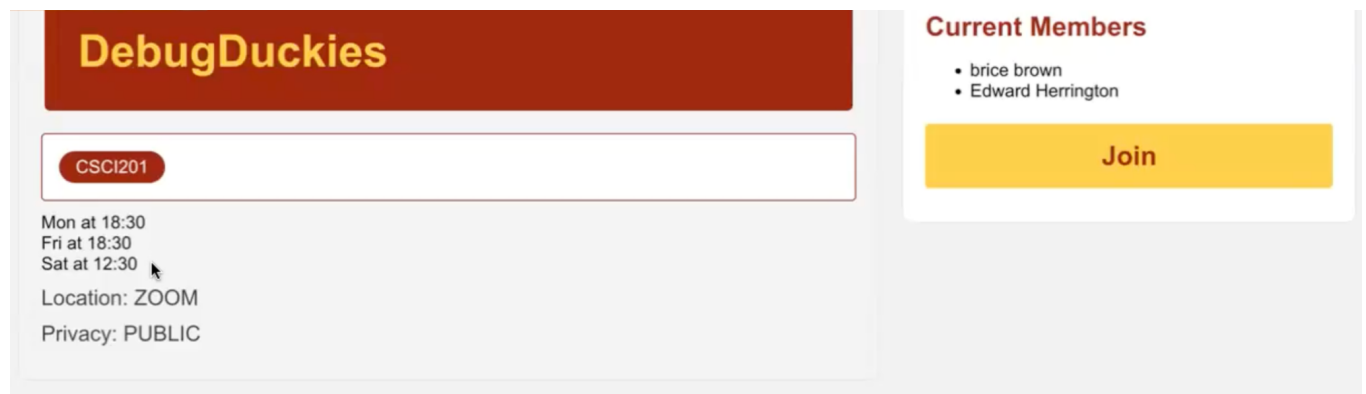
2.4 Study Group Management Page - Class Name: StudyGroupManagementPage

Wireframe:



The wireframe shows a rectangular box representing the page layout. At the top center is the text "GROUP NAME". Below this are three horizontal buttons labeled "COURSE 1", "COURSE 2", and "COURSE 3". To the left, under the heading "MEETING TIMES:", are two buttons: "TU 11:30AM" and "SA 2:00PM". Below that, under "LOCATION:", is a button labeled "THH101". To the right of these elements is a white rectangular box titled "CURRENT MEMBERS" containing four lines of text, each "JOHN SMITH". At the bottom center is a large green button labeled "JOIN".

Final:



The final design features a dark red header with the text "DebugDuckies" in yellow. Below the header is a white input field containing "CSCI201" in a red pill-shaped button. To the right of the input field is a "Current Members" section with a list of two members: "brice brown" and "Edward Herrington". Below the list is a yellow "Join" button. On the left side of the input field, the meeting times are listed: "Mon at 18:30", "Fri at 18:30", and "Sat at 12:30". Below the times is the location "Location: ZOOM" and the privacy setting "Privacy: PUBLIC".

2.5 Public Study Groups Page - Class Name: PublicStudyGroupsPage

- publicStudyGroupEntries: This variable represents a list of public study groups. It may have getter and setter methods for fetching and updating the public study group entries.

Backend Design

2.6 Authentication and Access Control - Class Name: AuthenticationManager

- access-code: A variable that stores the access code required to join private study groups. It has methods for setting, getting, and validating access codes.

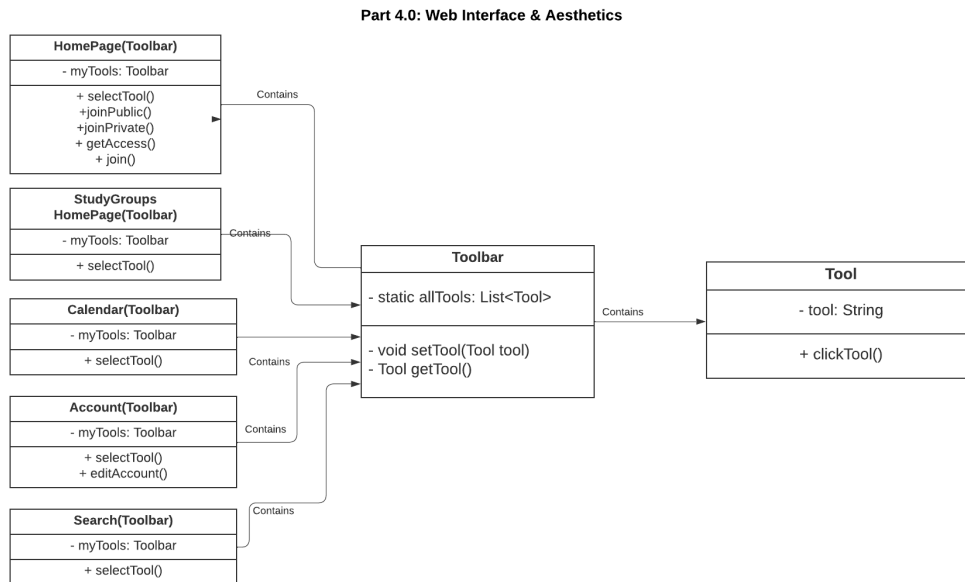
2.7 Database Structure - Class Name: DatabaseManager

- courseTable: A table in the database that stores information about courses offered. This variable has methods to retrieve course data.
- studyGroupTable: A table in the database that stores data related to study groups. This variable has methods for create, reading, updating, and deleting operations on study group data.
- meeting_day/ meeting_time: Tables in the database that store information about the day and time frames for study groups. This variable may include methods to update and retrieve time frame data.
- courseReferenceTable: A table that links study groups to courses. It may have methods for associating study groups with relevant courses and retrieving study groups relating to those courses.

2.8 Study Group Access - Class Name: StudyGroupAccessManager

- accessCodeValidation: A method that handles the validation of access codes when users attempt to access private study groups. It may return a boolean indicating whether the code is valid.

3.0 Web Interface & Aesthetics:



Frontend Design:

3.1 Home Page - Class Name: **HomePage**

-Home Page header at the top-center with the logo in the top-left corner, “Welcome to ‘Our Website’” includes a brief description

-Public and private study groups that the user is a part of are listed on the study group home page. Each study group will show the name of it and be clickable. When clicked, it will take the user to the main page of that study group

-joinPublicButton: A variable under the “Public Groups” header at the top of the list where the user will be presented with a text field

-joinPublicField: A variable for the text field associated with the public group the user wants to join: search for a study group using the name of the study group (case-insensitive) as the only way to find a group. Once the group is found, the user can click the “join” button

joinPrivateButton: A variable under the “Private Groups” header at the top of the list where the user will be presented with a text field

-joinPrivateField: A variable for the text field associated with the private group the user wants to join: search for a study group using the name of the study group (case-insensitive) as the only way to find a group. Once the group is found, the user can click the “access code” button and then will be prompted for the Access Code.

-accessCodeButton: A variable associated with the access code required to join a private study group. Once clicked, a text field will appear

-accessCodeField: A variable for the text field associated with the access code. The user can then type the access code associated with the private study group they are trying to join. Once they type the access code a “join” button will appear

-joinButton: A variable inside both Public and Private queries, that once the user finds the group they are looking for, they select “join” to join that group

3.2 Toolbar: ClassName: toolbarManager

-Toolbar in the top right corner, accessible on any page. Home to many icons that can perform different functions

-homeButton: A variable in the toolbar associated with the home page. When it is clicked, it will take the user back to their home page.

-studyGroupsButton: A variable in the toolbar associated with the homepage containing all study groups in our database. When it is clicked, it will take the user to the study groups home page.

-accountButton: A variable in the toolbar associated with the user’s account. When it is clicked, it will take the user to their account where they can see their account information, including their email, name, major, current courses, and past courses

-editAccountButton: A variable on the account page where users can update their information, namely their major, current courses, and past courses. Using Figma, this will look like a bunch of text boxes labeled with the specific piece of information they will replace

-searchButton: A variable in the toolbar associated with the study groups available on the website. When it is clicked, it will present the user with a field to type in their search queries

-searchField: A variable for the text field associated with searching for study groups. The user can search using keywords: name of the course they want to study and the name of the study group.

Backend Design:

3.1 Home Page - Class Name: HomePage

-joinPublic(): Once the user clicks the button, it prompts the user for the name of the group they want to join, it checks this name (case-insensitive) against the list of groups in the database and presents the user with the group or “No group found” message. The join button appears (frontend).

-joinPrivate(): Once the user clicks the button, it prompts the user for the name of the group they want to join, it checks this name (case-insensitive) against the list of groups in the database and presents the user with the group or “No group found” message. The access code button appears (frontend).

-getAccess(): Once the user clicks the button, it prompts the user for the access code. It then verifies this access code against the access code for this study group (case-sensitive). The join button appears (frontend)

-join(): Once the user clicks this button, they will be added to the list of students in this study group. This study group will then appear on the user’s homepage.

3.2 Toolbar - ClassName: toolbarManager

-Homepage.html: Takes the user back to the homepage

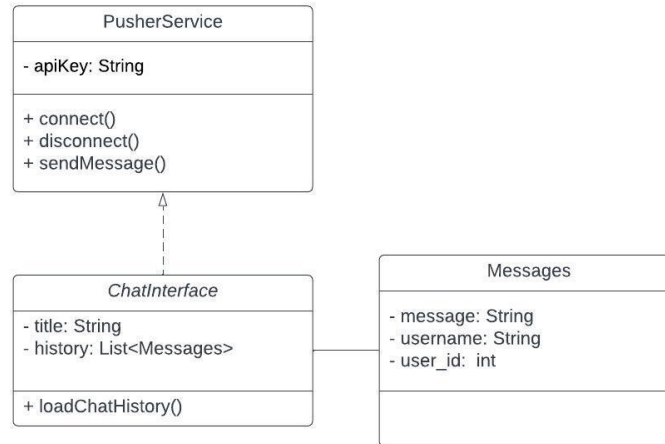
-goToStudyGroups(): Takes the user to the Study Groups Homepage

-goToAccount(): Takes the user to their account page

-editAccount(): Allows the user to make updates to their major, current courses, and past courses. Given what the user enters, this function will replace that information on the object of that user’s account

-goToSearch(): Takes the user to the search page. Given what the user enters, this function will check the information against the list of courses (to find study groups that study this course) and the list of study group names. It will then display any matches given user input or display “No match found”

4.0 Chat & Communication:



4.1 Connecting to Pusher - PusherService

Implementation

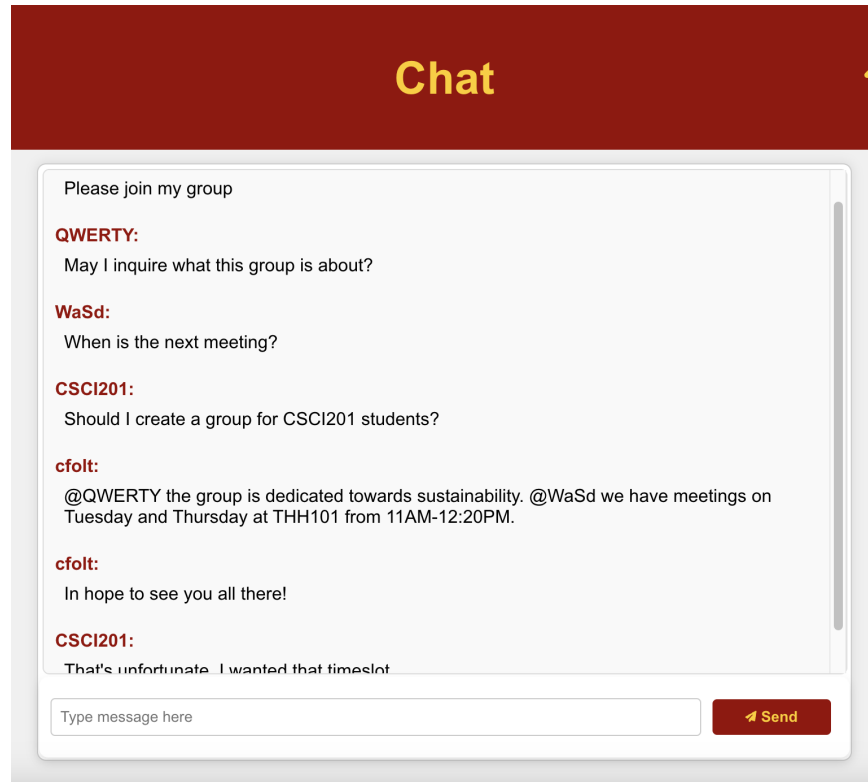
- There will be a server constantly listening for events from clients connected to the same channel.
- Frontend will capture user input and send the message data to Pusher using AJAX.
- Clients listening in on the event of that channel will receive the message that the server will distribute.

4.2 Maneuvering the Chat Interface - ChatInterface

Wireframe:



Final:



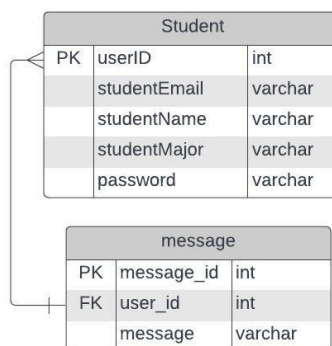
loadChatHistory() - The method loads and displays the chat history, allowing users to scroll back and view previous messages.

4.3 Sending A Message - Messages

Implementation

- Users will have an input line where they can type their message. To send it they can either click the send button or press the 'Enter' key on the keyboard.
- Create a method using JDBC that will store/retrieve messages in the database.

4.4 Database



CSCI Group 32 - Testing Plan

1.0 User Management

1.1 Login

Name	Type	Input	Expected Result
Check for Required Inputs: Email	Black Box	Email input is empty but password is "password"	checkLogin returns -1; Alert message "Please enter your email"
Check for Required Inputs: Password	Black Box	Email input is " test@usc.edu " but password is empty	checkLogin returns -1; Alert message "Please enter your password"
Check for valid login information: Correct	Unit test	Email input is " test@usc.edu " and password input is "password"	checkLogin returns -1; The program fetches the correct user account to load into the session.
Check for valid login information: Incorrect password	White box	Email input is " test@usc.edu " and password input is "password"	checkLogin returns -1; The program displays "incorrect login information" in red text and is accepting new input.
Check for valid login information: Incorrect email	White box	Email input is " test@usc.edu " and password input is "password"	checkLogin returns -1; The program displays "incorrect login information" in red text and is accepting new input.
Check for changes to the profile page if user information is correct	Regression test	Email input is " test@usc.edu " and password input is "password"	The program continues on to the profile page.

1.2 Registration

Name	Type	Input	Expected Result
Check for Required Inputs: Email	Black Box	Email input is empty but all other inputs are filled	Red message "Please fill in all required information"
Check for Required Inputs:	Black Box	Password input is empty	Red message "Please fill all required"

Password		but all other inputs are filled	information”
Check for Required Inputs: First name	Black Box	First name input is empty but all other inputs are filled	Red message “Please fill all required information”
Check for Required Inputs: Last name	Black Box	Last name input is empty but all other inputs are filled	Red message “Please fill all required information”
Check for Required Inputs: Major	Black Box	Major input is empty but all other inputs are filled	Red message “Please fill all required information”
Check for correct user added	White Box	Put all inputs for a new user with current courses and previous courses.	A new database entry is added with all corresponding input
Add a current course to the list	Unit Test	Add “EE 249” to current courses.	A new entry appears under the current courses tab on the web page
Remove a current course to the list	Unit Test	Remove “EE 249” from current courses	A entry is remove on the courses tab on the web page

1.3 Profile Page

Name	Type	Input	Expected Result
Check for Required Inputs: Name	Black Box	Name field is blank	Display error message prompting user to fill in the Name field
Check for Required Inputs: Major	Black Box	Major field is blank	Display error message prompting user to fill in the Major field
Check for Required Inputs: Current Courses	Black Box	Current courses field is blank	Display error message prompting user to fill in the Current Courses field
User Updates Major Field	Black Box	User updates their major	Profile page updates and displays new major
User Updates Name Field	Black Box	User updates their Name	Profile page updates and displays new Name
User Updates Current Courses	Black Box	User updates their current	Profile page updates and

Field		courses	displays the updated list of courses
User Updates Email Field	Black Box	User updates their email	Profile page updates and displays the new email
User Updates Password Field	Black Box	User updates their password	Profile page updates and ensures password security

2.0 Study Group Management

2.1 Study Group Creation Page

Name	Type	Input	Expected Result
Courses Input Field	Black Box	“AAA-111,” “BBB-242”	“AAA-111” and “BBB-242” should be displayed to the side of the courses input field.
Location Input Field, Online	Black Box	Click Online Button	The online button should highlight and the location should gray out, if there was a location it should be cleared.
Location Input Field, Physical	Black Box	Room, “THH101”	The Online option should gray out and the user should be able to type in the location text field.
Privacy Selection	Black Box	Private	The user should get a prompt to enter a code to be associated with the private room.
Submit Form	White Box	Study group name, courses, meeting times, location, privacy	Every field should be populated upon submission and there should be a code also sent to the backend if the mode is private.
Submit Form Result	Black Box	Clicking submit button	The user should be navigated to a page displaying the success of the study group.

Submission API	White Box	Clicking submit button	A JSON object containing all the users information they submitted in the document should be read from the API into the backend.
----------------	-----------	------------------------	---

2.2 Study Group Creation Success Page

Name	Type	Input	Expected Result
Creation Message	Unit Test	Entering valid study group parameters	A message of success should be displayed to users once they create a valid study group.

2.3 Study Group Management Page

Name	Type	Input	Expected Result
Visual Inspection, static	Unit Test	Load in a premade JSON file of sample data	Properly displays all data from the JSON.
Visual Inspection, dynamic	White Box	Load in the study group under normal conditions.	Properly displays data associated/stored under the study group.
Guest Mode	Black Box	Access study group page as a guest	All information should be shown except location, member names and join button.
Private Mode	Black Box	Access study group page as a user	Show everything like in guest mode except the Join button has a code next to it to join.
Joining Private Group, Success	White Box	Input correct code to study group.	Joins the group's member list and displays all information.
Joining Private Group, Failed	Black Box	Input incorrect code.	Give an error and reset the code box.

2.4 Public Study Groups Page

Name	Type	Input	Expected Result
View Study Groups	Black Box	Clicking view button	Should show all of the public study

			groups that are available
Filter Study Groups	Black Box	Clicking filter study groups button and selecting a way to filter them	Study groups shown should be restricted to the filtered parameter i.e. only those on Wednesdays or only those at 6:00.
Join Study Group	White Box	Clicking on a study group and pressing the join button	User should be connected to the study group object and vice versa.

3.0 Web Interface & Aesthetics

3.1 Home Page - Class Name: HomePage

Name	Type	Input	Expected Result
Authenticated User Homepage	Black Box	User logs in and is taken to their homepage	Homepage should have a toolbar with buttons: home, account, study groups, and search and should also include a list of the public and private study groups they are a member of with join buttons under both sections
Guest User Homepage	Black Box	User continues as guest and is taken to the homepage	Homepage should be mostly blank with a toolbar that should include join buttons under the public and private sections but no list of study groups
Click a Study Group	Black Box	Authenticated user clicks on a Study Group that they are a member of	The user should be taken to the homepage of that study group
Join Public Button	Black Box	User clicks on the join public button to join a public study group	Authenticated User: loads a page where they can specify the name of the group they want to join Guest user: Takes them to the login/registration page
Public Name Field	White Box	User specifies the name of the group	Group Exists: They can click join which then takes them to the group's homepage Group DNE: The page displays a "No Matches Found" Message (back button)
Join Private Button	Black Box	User clicks on the join private button to join	Authenticated User: loads a page where they can specify the name of the group they want to join

		a private study group	Guest user: Takes them to the login/registration page
Private Name Field	White Box	User specifies the name of the group	Group Exists: They can click “access code” which then loads a page to type in the access code Group DNE: The page displays a “No Matches Found” Message (back button)
Correct Access Code	White Box	User types correct access code for the group	The user should be taken into the study group’s homepage and a “Success” message should pop up
Incorrect Access Code	White Box	User types incorrect access code for the group	The user should be taken back to the access code page with a message that displays “Invalid Access Code”

3.2 Toolbar: ClassName: toolbarManager

Name	Type	Input	Expected Result
Clicking button to navigate to same page	Black Box	Clicking a button to navigate to the page that the user is currently viewing	Should simply refresh the page that the user is currently viewing without taking them to a different page beforehand. Ex. Viewing home and clicking home should reload home
Clicking button to navigate to different page	Black Box	Clicking a button to navigate to a different page	Should simply load the page corresponding to the button that the user clicks. Viewing home and clicking account should load the account page
Guest clicking “Account” button	Black Box	Guest clicks account button to go to their profile	Should redirect the user to the login/registration page because they are an unauthenticated user and do not have a profile.
Editing Account	Black Box	Clicking edit account within the account page	Takes the user to another page where they fill out a form with the edits they want to make to their major, current courses, and past courses.
Filling Account Fields and “Make Changes” button	Black Box	User fills in fields with desired information	User fills in fields then clicks “Make Changes” which should save the changes they made to their user account and reload the account page with the correct information.
Clicking search button	Black Box	Click search button to take user to search	User will be presented with a field to enter keywords pertaining to the course they are

		page	searching for
Search Field: Group exists	White Box	User enters keywords that match to a study group	The keywords match a study group or study groups in the database. The page displays the ones that match
Search Field: Group does not exist	White Box	User enters keywords that do not match to a study group	The keywords do not match any study groups in the database. The user should be shown a “No Results Found” message

4.0 Chat and Communication

4.1 Connection to Pusher

Name	Type	Input	Expected Result
Connecting to Pusher/Channels	Black Box	Name of channel	Pusher’s Debug Console should state that a client has connected
Pusher Message	Black Box	Send a message to clients binded to the event using Pusher’s debug console	The message is displayed on the clients’ chatbox

4.2 Maneuvering the Chat Interface

Name	Type	Input	Expected Result
Retrieve Messages	Black Box	Chat history in the database	The previous messages are displayed
Store Messages	White Box	Send a message in the input field	The message is stored in the database with the ID of the user who sent it
Chat Authorization	Black Box	User tries to access the chat	They should not have the option to see the navigation to chat if not logged in

4.3 Sending A Message

Name	Type	Input	Expected Result
Sending Messages	Black Box	Click Send Button/Pressing 'Enter/Return' key on the keyboard	The message is sent to other clients and the input field is reset.
Character Limit	Stress	A very long string	The message should still be able to be sent
Empty Message	Black Box	Click Send Button with an empty input field	Nothing should be sent
Emoji Message	Black Box	Message full of emojis	Emojis are displayed correctly

CSCI Group 32 - Deployment Document

Platform

Github Pages

Prerequisites

1. Be an admin part of the <https://github.com/CSCI201-G32> organization.

Deployment

Frontend:

1. Create a branch called gh-pages
2. Have gh-pages pull the react files you want to have hosted (for us git merge main)
3. Run npm install gh-pages --save-dev within the react folder
4. Add this to the package.json file under scripts:
 - a. "predeploy" : "npm run build",
 - b. "deploy" : "gh-pages -d build",
5. Navigate to <https://github.com/CSCI201-G32/StudyGroups/settings/pages>
6. Set branch to gh-pages
7. Frontend is hosted on <https://csci201-g32.github.io/StudyGroups/>
8. Run npm run deploy to make a build version of react

Frontend- Chat:

1. To get the chat functionality, you need to start the server.js service.
2. Type `npm install express/cors`
3. Type `node server.js`
4. You should get the message: listening to port 8000
 - a. Done!

Backend:

1. Local hosting
 - a. Create a dynamic java web project in eclipse.
 - b. Add Java servlet files to Java Resource under default package and util.
 - c. Add MySQL and GSON Jar files.
 - d. Start the tomcat server
 - e. Start the mysql database and run the StudyGroups.sql script to creat the database
2. Hosting on AWS EC2 (not implemented in final version)
 - a. If you'd like to host the instance on AWS EC2 instead, export the above dynamic java web project as a .war file.
 - b. Create an Amazon EC2 instance and enable HTTP traffic to the instance.

- c. Install Java (installation depends on instance OS. For Windows, you would download and install JDK. For Linux, you can to “sudo yum install java -y)
- d. Download and install apache tomcat. Instructions vary by OS instance.
Start tomcat
- e. Place your .war file in the webapps folder inside of where tomcat is installed.

Post-Deployment/Troubleshooting

If the initial deployment fails, recheck the build commands for both the frontend and backend. Try multiple, plausible combinations while looking at the source code and console to resolve the issue.

Frontend:

- Outdated Node, failed build when running npm:
 - run: export NODE_OPTIONS=--openssl-legacy-provider
- Not able to route within the file:
 - Create a file with no extension in public folder named, _redirects and just put this in it /* /index.html 200.

Backend:

- More detailed instructions/troubleshooting can be found here:
 - <https://www.youtube.com/watch?v=9XN0Fd5SX6M>
 - <https://stackoverflow.com/questions/15996741/how-to-deploy-a-eclipse-java-web-dynamic-project-on-amazon-ec2>

Contact Information

Team Members:

- Brice Brown, BriceBro@usc.edu
- Jonathan Aydin, jaydin@usc.edu
- Jennifer Ayissi, ayissi@usc.edu
- Rachel Appenzeller, appenzel@usc.edu
- Carlos Barcelo, cbarcelo@usc.edu
- Alan Au, alanau@usc.edu
- Krishna Biniwale, kbiniwal@usc.edu
- Simi Awujo, awujo@usc.edu