**Team:** Citrix Are For Kids!
**Project Title:** Study Together
**Authors:**
O'shea Anaya
Shadee Barzin
Andrew Ferguson
Brendan Murphy
Peter Werner

# Product Requirements Design 2.0

## Introduction

### Problem Statement

Throughout their high school and university experiences, students spend much time collaborating with peers, teachers, and TAs, as well as participating in group projects, study groups, and review sessions. However for many students, physically getting together in a group setting can be a challenge. There are many non-traditional students such as those who work full or part-time, parents, and disabled people, who find it difficult to set aside the time to travel to another location for a group meeting when they have many other priorities. Additionally, for students, finding a physical meeting space on campus can be difficult, especially during high traffic times such as midterms and finals. There are also often groups of students from various schools and courses who get together to study for exams such as the SAT, GRE, and MCAT, who would benefit greatly from an online platform that allows them to create ongoing study groups while practicing for these exams. Currently, the best solutions for remote collaboration and studying are a combination of Skype or Google Hangouts and Google Drive, but these lack integration with one another and are also not focused on student study groups.

### Project Outcome & Innovations

The aim of this project is to make studying with peers significantly easier. Study Together is a platform that combines video/audio meetings, screen sharing, and document editing to make group collaboration possible without being physically near the group members. It also provides a matchmaking service for students looking to join a study group; the application displays available study sessions for specific courses and allows students to request to join an existing study session if they find one that suits their needs. These features not only improve studying with a group when people are unable to meet in person, but they make conventional, in-person study meetings obsolete by providing invaluable study tools and built-in smarts.

Study Together allows students to enter their school and courses for the term. Based on this information, the application creates a unique dashboard for the user that displays a list of

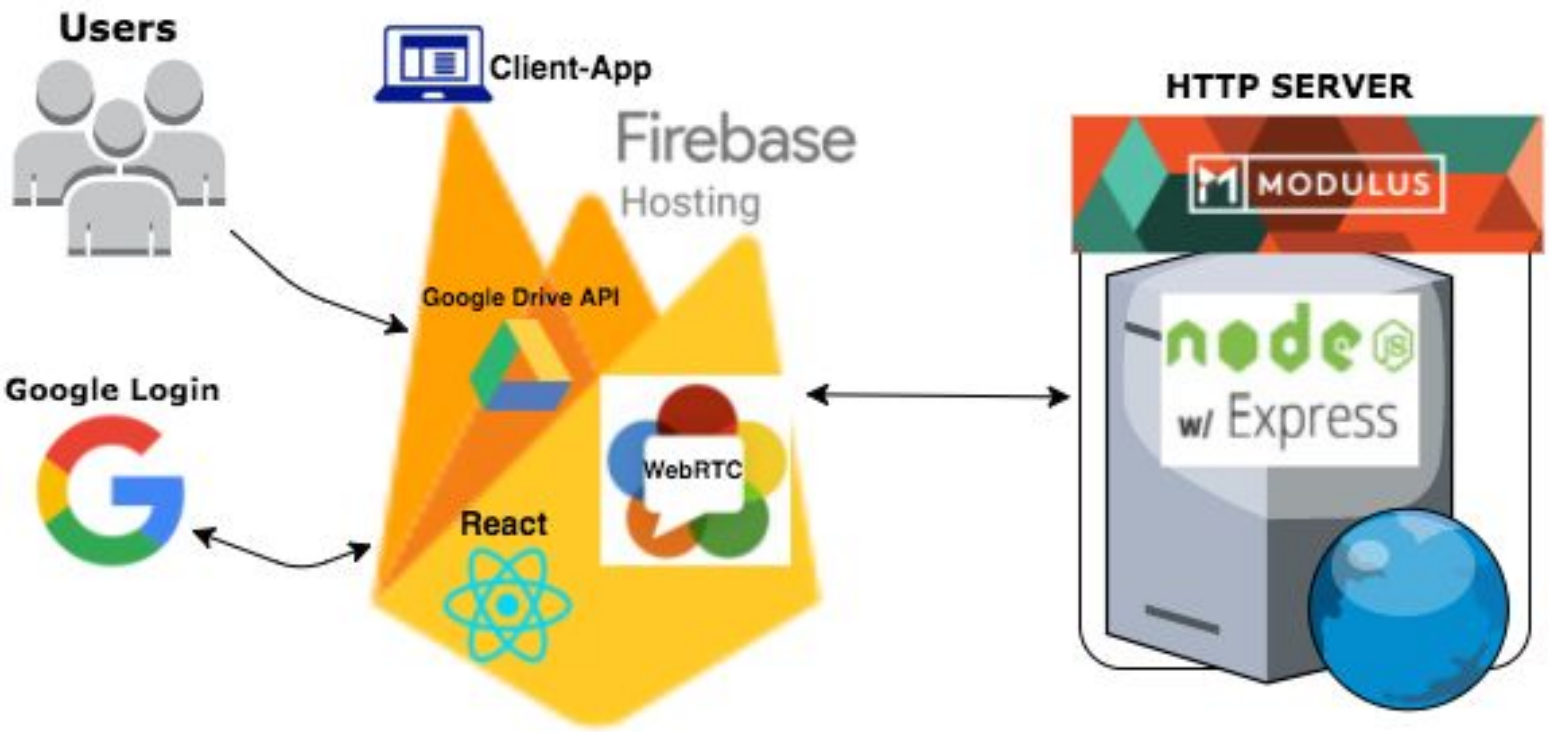
their classes in the center. The user is able to view each class's page, where all current study groups will be displayed as well as file uploads from classmates; these files may be lecture notes, notes from a review session, a photo of the blackboard during lecture, or any other useful information that a student thinks could be helpful for his or her classmates. From this page, a user is able to start a study session and make it public to the whole class so that anyone wanting to join a study group can simply navigate to the class's page, find the group, and request to join. The aim of this matchmaking feature is to make finding a study group simple and intuitive.

The main feature of Study Together is the creation of study groups. From the chat sidebar, users are then able to create study groups and invite specific people if they wish to keep their group private. Invited members of the group are also able to invite additional members to the group if they wish to do so. This creates an additional chat, specific for the members in the group. These sessions are not viewable from a specific class's page in the "Active Sessions" section of a class page unless explicitly declared public by the session's creator. In a study group session, users are able to use audio and video to speak to each other, as well as text chat to communicate additional information such as web pages, images, and files with other group members. The study group session is also connected to Google Drive to cohesively integrate collaborative document editing in the application.While in a session, the pullout sidebar contains tabs with the group's chat and a file list. The file list tab displays to the user all Google Drive files shared with the members of that specific session. This way, all the group members can work together on a single file while speaking to one another, or even in separate docs to share relevant information about different topics. To make remote collaboration even better, Study Together has built-in smarts that automatically bring up web pages, articles, and images relevant to the topic being discussed in the group meeting.

The application also contains a chat feature, which is collapsible from the side of the screen. The chat bar is separated into two sections: one for the user's classes, where a list of all their classmates is displayed, and one for all the user's contacts, which may be previous classmates or friends from other schools. Users can also add contacts by simply searching for the person. The chat feature is separate from the video session, allowing users to coordinate meetups, discuss topics at hand, and preserve history of communication. The application also provides a chat transcript of the messages sent between users once the session is completed, which can be used as a summary of the whole group study session. If so desired, the file can then be uploaded to a specific course's upload section for the whole class to access and review for an upcoming exam or project.

Study Together's most prominent innovation is providing multiple forms of communication that students currently use separately all into one, cohesive application as well as providing an interface for students in the same courses to share relevant and helpful information with their peers. It allows an entire class of students to study with one another without the hassle of finding a physical meeting space and the communication difficulties that come with studying with a large group.

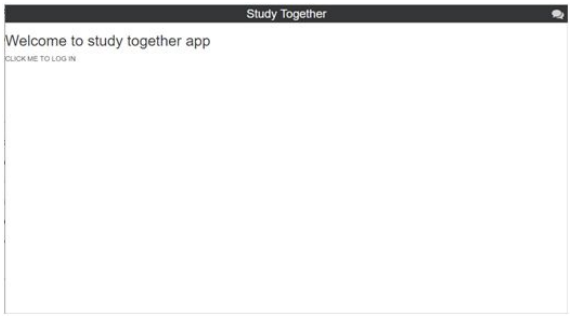
## System Architecture



**User Interaction and Design**

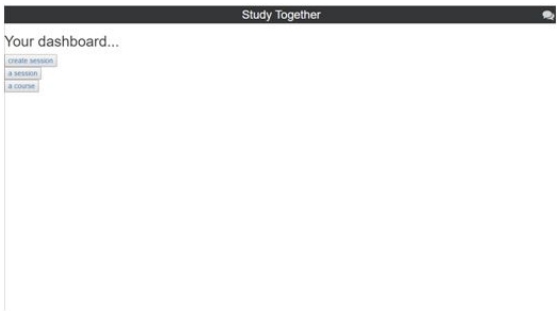
When a user of Study Together first accesses the web page, they are brought to the log in page.

Log In Page:



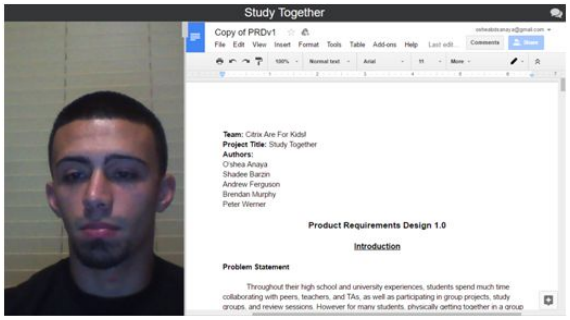

Once logged in, the user is led directly to their dashboard; on the dashboard page, the user is able to create a session, join a previously created session, or access any of his or her course pages.

Dashboard Page:



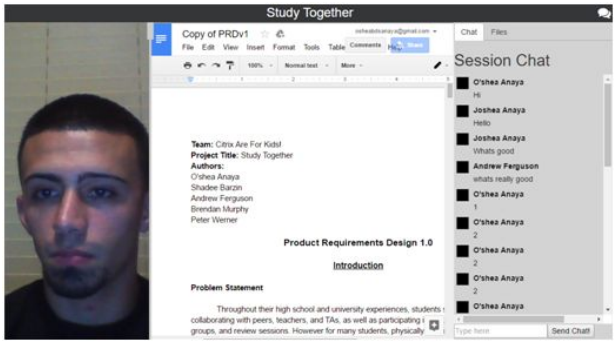

The session layout is comprised of two parts: the video component and the Google Drive component. One the left side of the screen is the video component holds the videos of the members of the study session and can hold more than five videos. On the right is the Google Drive component, where the user can launch any document in the session's Google Drive directory.

Session Layout:



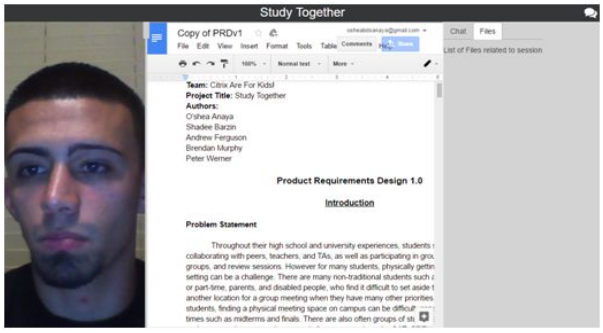

From the far right is a pull out side bar that contains tabs for both the chat and file tab. The chat tab holds the text chat for the study group members that are in the session, whether they are currently online or not.

Chat Tab:




The file tab contains a list of the files in the study group's Google Drive directory. This directory persists between study sessions, and the files added to it are available to all members of the study group, online or not.

File Tab:



# Requirements

## User Stories

1. First Time User Login
    a. As a student, I create an account so I can begin using the application.
    b. Acceptance Criteria
        i. User is logged into account
            1. User ID in database
        ii. Can generate and refresh an auth token
        iii. User is led to user management page (where he/she enters school info, class info, user info)
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/HomeLayout.jsx
    d. Time Estimate: 2 days
2. Existing User Login
    a. As a student, I can log into my account so I can study with my peers.
    b. Acceptance Criteria
        i. User is logged into account
            1. User ID in database
        ii. Can generate and refresh an auth token
        iii. Can view and interact with dashboard
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/HomeLayout.jsx
    d. Time Estimate: 2 days
3. Logout
    a. As a student, I can logout of my account when I am done using the application.
    b. Acceptance Criteria
        i. User cannot access any of his/her courses or study sessions
        ii. User cannot access his/her dashboard
        iii. User is led directly to login page
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/test/App.test.js
    d. Time Estimate: 1/2 days
4. Viewing Course Page
    a. As the member of a specific class, I can view the course's profile page
    b. Acceptance Criteria
        i. User can see the active study sessions and their related topics
        ii. User can view the list of course file uploads (the relevant files other users have uploaded)

        iii. User can see the create study session button for the class
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/CourseLayout.jsx
    d. Time Estimate: 1/2 days
5. Create a Study Session
    a. As a student, I can create a study session so I can begin a study group.
    b. Acceptance Criteria
        i. New video session is created with a group key, that is stored in the database
        ii. Creator can specify the class, project, test, topic, etc that the group is studying for (since there can be multiple subjects/exams to study for a certain class)
        iii. Creator is automatically added to the video session
        iv. Video, audio, and text chat are automatically started
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/SessionLayout.jsx
    d. Time Estimate: 5 days
6. Join a Study Session
    a. As a student, I can join a previously created study session so I can study with my peers.
    b. Acceptance Criteria
        i. User is added to the preexisting video session using a group key
        ii. User is led directly to the study session
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/server-app/test/test.js
    d. Time Estimate: 1 days
7. Join a Study Session with an open Google Doc
    a. As a student, I can join a previously created study session so I can study with my peers.
    b. Acceptance Criteria
        i. User is added to the preexisting video session using a group key
        ii. User is led directly to the study session
        iii. Specified Google Doc opens automatically
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/SessionLayout.jsx
    d. Time Estimate: 2 days
8. Leaving a Study Session
    a. As the member of a study session, I can leave the study group so I can join a different group or leave the site completely.
    b. Acceptance Criteria

        i. User's id is removed from the group in the database
        ii. User is removed from the study session
        iii. User is led to his/her dashboard
        iv. The study session persists without the user
    c. Time Estimate: 1 days
9. Viewing Dashboard
    a. As a student, I can view my dashboard so I can see and interact with the list of my classes and the create study session button.
    b. Acceptance Criteria
        i. User's class list is displayed on the screen with all relevant info (number of active study sessions)
        ii. User is able to click and interact with the classes
        iii. User can create a study session by clicking the create study session button
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/layouts/DashboardLayout.jsx
    d. Time Estimate: 1/2 days
10. Open Chat/File Sidebar
    a. As a student, I can click on the chat icon so I can expand the chat/file side
    b. Acceptance Criteria
        i. After icon is clicked, sidebar is expanded
        ii. User can view the contents of the sidebar (Chat or File Tab)
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/components/SidebarSlideComponent.jsx
    d. Time Estimate: 3 days
11. Close Chat/File Sidebar
    a. As a student, I can click on the Chat icon so I can expand and the chat/file sidebar
    b. Acceptance Criteria
        i. After icon is clicked (when expanded), sidebar is collapsed
        ii. User can no longer view the contents of the sidebar
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/components/SidebarSlideComponent.jsx
    d. Time Estimate: 1 days
12. Switch To File Tab
    a. As a student, I can click on the the File tab inside the sidebar so I can switch to the File Tab
    b. Acceptance Criteria
        i. After File tab is clicked, the File tab is opened
        ii. User can view and interact with the contents in the File tab

    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/components/ChatComponent.jsx
    d. Time Estimate: 3 days
13. Switch To Chat Tab
    a. As a student, I can click on the the Chat tab inside the sidebar so I can switch to the Chat Tab
    b. Acceptance Criteria
        i. After Chat tab is clicked, the Chat tab is opened
        ii. User can view and interact with the contents in the Chat tab
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/components/ChatComponent.jsx
    d. Time Estimate: 2 days
14. Send a Chat Message
    a. As a student, I can send text messages so I can communicate with other users
    b. Acceptance Criteria
        i. Message can be sent to server and received by the group(/person)
        ii. Message is stored in chat history
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/client-app/src/components/ChatComponent.jsx
    d. Time Estimate: 4 days
15. Open a Google Drive file while in a Session
    a. As a student, I can click on a Google Drive file in the File Tab so I can open the file in the session
    b. Acceptance Criteria
        i. After a file is clicked, the File is opened in the session
        ii. User can view the contents of the Google Drive file
    c. Time Estimate: 4 days
16. Edit the Google Drive file while in a session
    a. As a student, I can view and edit the Google Drive file opened in the session so I can make changes to the file
    b. Acceptance Criteria
        i. User can modify the Google Drive file
        ii. Google Drive file is updated with all changes (from any user) in real time
    c. Time Estimate: 3 days
17. Add files to a session/the session's file list in sidebar
    a. As a student, I can add a Google Drive file to the list of Files in the File Tab so I can share files with other users in the session
    b. Acceptance Criteria
        i. User can add a file to the file list

        ii. All users in the session are able to view and interact with the file after it is added
    c. Time Estimate: 4 days
18. Invite Users to a Study Session
    a. As the member of a study session, I can invite additional users to join my study group so I can study with more people.
    b. Acceptance Criteria
        i. Invited users receive a notification saying they have been invited to a study group
        ii. Invited users can accept or reject the invitation.
    c. Implementation/Test Cases: https://github.com/peterwerner/citrix-capstone/blob/master/server-app/test/test.js
    d. Time Estimate: 2 days
19. Accept/Reject Invitation to a Study Session
    a. As an invited user of a study session, I can accept or reject the invitation sent to me by a member of the study group so I can decide whether I want to be in the group or not
    b. Acceptance Criteria
        i. If the invite is accepted, the user is added to the preexisting video session using a group key
        ii. If the invite is accepted, the user is led directly to the study session
        iii. If the invite is rejected, nothing happens
    c. Time Estimate: 2 days
20. Adding Files to Course Page
    a. As the member of a specific class, I can upload files to the course page so I can post information I find relevant for other members of my class.
    b. Acceptance Criteria
        i. Google Drive file appears in the file upload section of the course page
        ii. File is sorted by most recent date added
        iii. File topic tags are viewable below the icon
    c. Time Estimate: 5 days
21. Send a File/Photo/etc. via Message
    a. As a student, I can send a file so I can share the file with another student/group through chat
    b. Acceptance Criteria
        i. File can be sent to server and received by the group(/person)
        ii. File is stored in chat history
    c. Time Estimate: 6 days
22. User Profile Management
    a. As a student, I can modify my profile information so I can control my account
    b. Acceptance Criteria
        i. User can add/modify school(s) and class info
        ii. School(s) and Class information are stored/updated in database

iii. User is led to the dashboard

c. Time Estimate: 5 days

**Prototyping code**

New code should be tested locally first, by running the server and client locally.

- Running the server locally - from the *server-app* directory, run *npm start*
- Running the client locally - from the *client-app* directory, run *npm start*

Github unit tests: https://github.com/peterwerner/citrix-capstone/tree/server-app.chai

**Deploying code**

Code should be deployed once it is merged to master (see **GitHub Commits/Issues**).

- Deploying the server - from the *server-app* directory, run *npm run zip*, then upload the server archive to Modulus
- Deploying the client - from the *client-app* directory, run *npm run build*, then run *npm run deploy*. (You must have the Firebase CLI installed.)

**Tests**

Before merging to master and deploying, the automated tests should be run.

- Testing the server - from the *server-app* directory, run *npm run test*
- Testing the client - from the *client-app* directory, run *npm run test*

**GitHub Commits/Issues**

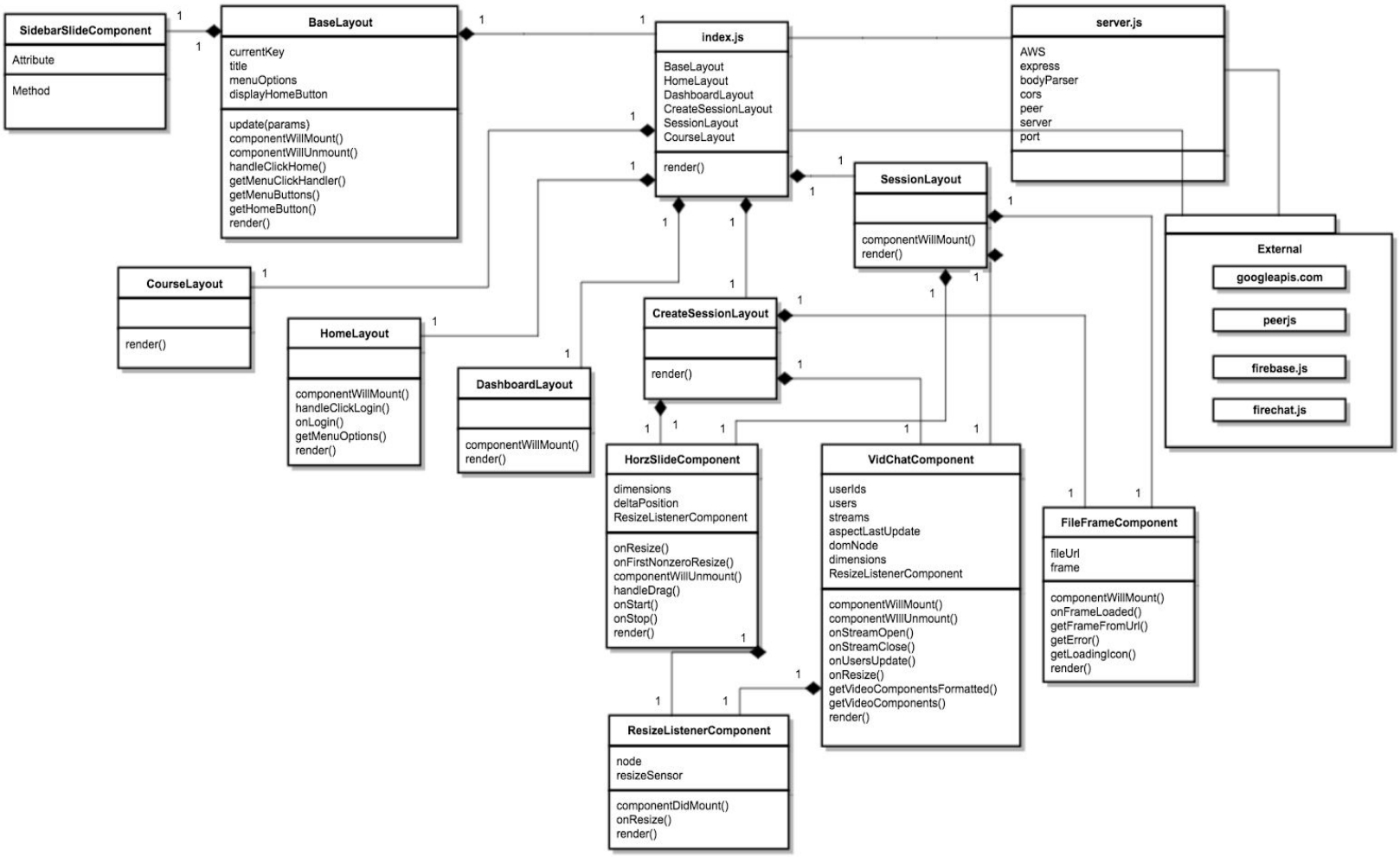
Committing new code:

1. Ensure the automated tests pass.
2. Commit to a new branch.
3. Open a pull request. Assign at least two other people to review it.
4. Once reviewed, merge with master locally.
5. Ensure the automated tests pass.
6. Merge the pull request to the master branch.
7. Deploy.

Issues are tracked on Waffle.io (integrated with GitHub issues). When you create a pull request for or merge in code resolving an issue, move the issue to the appropriate column on Waffle.io.

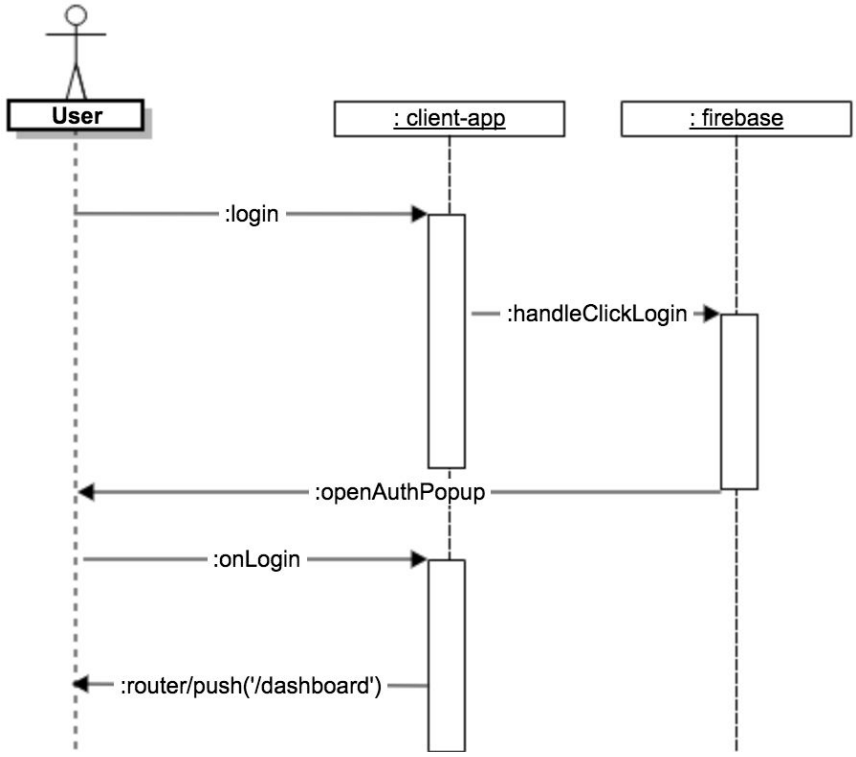
# System Models

## Class Diagram

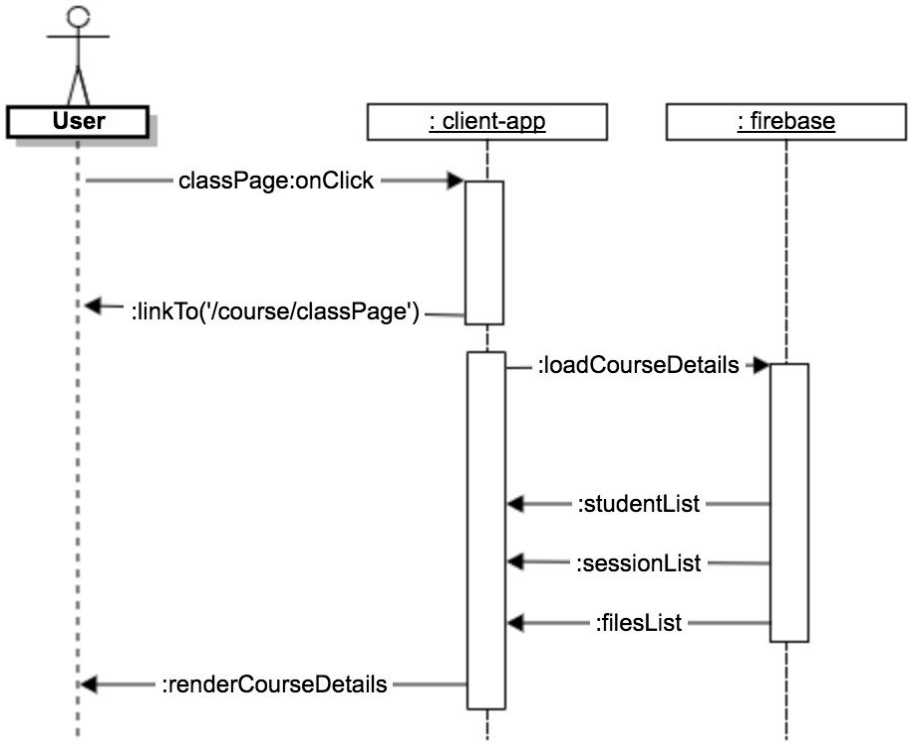


**Sequence Diagrams**

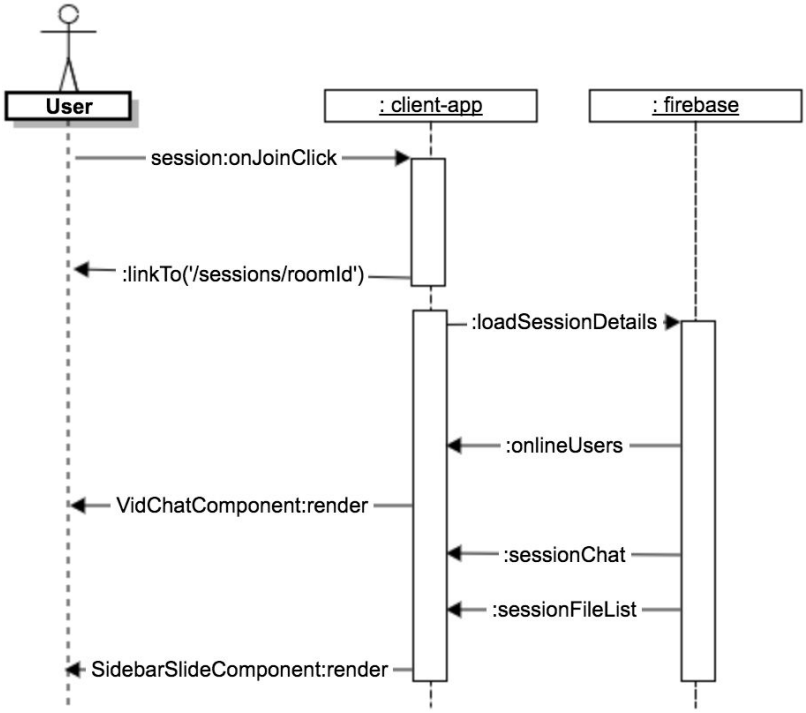
Log In



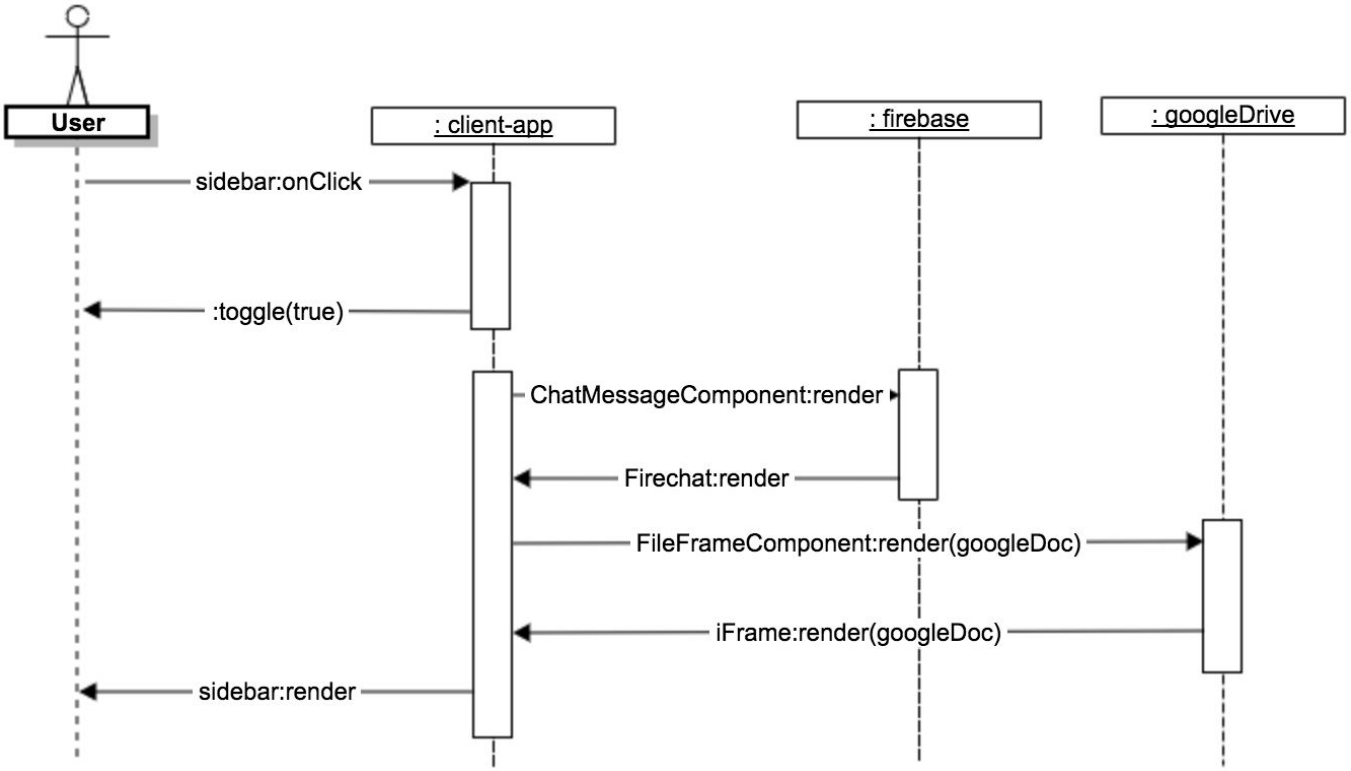

Navigating to a Class Page



Joining a Study Session




Opening Chat/File List Sidebar



# Appendix

**Technologies Employed**

Client-side

- Static website stored using Firebase's Hosting for static website hosting
- Front-end is generated by React components
- Communicates with API endpoints on the server (CORS - cross origin resource sharing)
    - FireChat for text chat - the client listens for data from and sends data to the server
    - WebRTC/PeerJS for video chat - the client requests a peer-to-peer connection to other clients from the server, sends data to and receives data from peer clients directly
- Communicates with Google APIs for public user profile information

Server-side

- Node JS server running on Modulus
- Responds to requests from the client (CORS - cross origin resource sharing)
    - Firebase Firechat for text chat, PeerJS (WebRTC wrapper) for peer-to-peer video chat - the server listens for data from the client, brokers connections between users, and sends data back to the client
    - Communicates with Google APIs for identity verification and access to private user information and files hosted on Google Drive

Database

- Firebase's Real-time Database is a non-relational database - accessed by the server to persist, query, and retrieve data
    - User information (school IDs, course IDs, group IDs, Prof Pic, etc.)
    - Group information (users IDs, file IDs)
    - School information (course IDs)
    - Course information (user IDs, group IDs)
- Firebase Hosting
    - The client website is hosted using Firebase's hosting for static websites
    - Files uploaded by users are stored on google drive, and are retrieved through the database (file IDs are stored in the database)

Firechat

- A Firebase powered chat widget
    - Allows for text communication between users
    - Chat log stored on Firebase by key for later use and observance
    - Extends to multiple chat rooms (public or private)

**Glossary of Terms**

| Term | Definition |
|---|---|
| Firebase | Platform with numerous features to build infrastructure for our app |
| Firebase Database | Firebase's cloud hosted, NoSQL, real-time database |
| Firechat | Firebase powered chat widget allowing for an easily implementable text chat feature |
| Google Drive | Secure cloud storage that allows for real-time editing and file sharing between users |
| Google Hangouts | Platform for messaging other users via text, audio, and video calls |
| Modulus | Scalable app platform for deploying, hosting, and managing Node.js, Java, and MongoDB applications |
| NodeJS | Open-source, cross-platform JavaScript runtime environment for developing a diverse variety of tools and applications |
| PeerJS | Wrapper library for WebRTC allowing peer-to-peer video and audio communication between client's browsers |
| ReactJS | Open-source JavaScript library for data rendered as HTML |
| WebRTC | Peer-to-peer communication over the web, using websockets (used by PeerJS) |
| Websockets | Browser-to-browser communication protocol (used by PeerJS) |