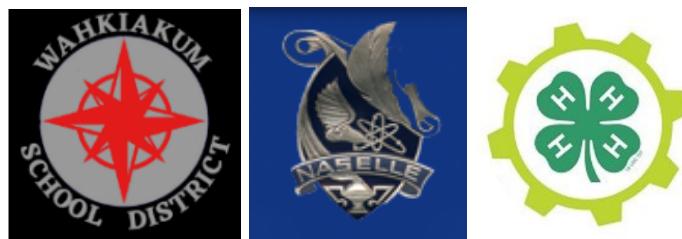


Google Workspaces Alternative Application

Project Solution Approach

Wahkiakum School district, Naselle School District and Wahkiakum 4-H



Team Toto

Albert Lucas

Ben Kaufmann

Daniel Semenko

TABLE OF CONTENTS

Introduction	4
System Overview	4
Architecture Design	4
Overview	4
Subsystem Decomposition	5
Data design	11
User Interface Design	11
Glossary	16
References	17
APPENDICES	19
APPENDIX A - SOFTWARE ARCHITECTURE	19
APPENDIX B - DATA DESIGN	20
APPENDIX C - USER INTERFACE DESIGN	21

I. Introduction

This document is a record of Team Toto's solution design for Wahkiakum School district, Naselle School district, and Wahkiakum 4-H. It briefs the reader on the approach to building a communicative platform available for ChromeOS and Android. It dives deep into the blueprint of the application and projects how a future prototype should portray data and interact with users. This document is intended for developers, faculty, and anyone that is curious about how the application was constructed.

Wahkiakum School district, Naselle School district, and Wahkiakum 4-H are in need of an application that allows student users to communicate with each other across classrooms, clubs, groups, etc. To satisfy these district needs, our team is planning on analyzing, designing, and building a stand-alone application from scratch to improve student learning and connection.

II. System Overview

The application we are creating will be made using the Flutter Software Development Kit. We chose Flutter since it offers cross-platform capabilities which satisfy the need for running on Chromebooks while also being available to most mobile devices. Our team does not have experience working with Flutter, but experienced developers say that it is easy to learn and similar to Django. Flutter allows for different types of architectural patterns, so we will be using the Business Logic Component pattern since we have all worked with it before. In terms of the User Interface, we decided to model our application like similar applications such as Discord and Google Spaces. That means having the user's post spaces accessible from anywhere in the app using a sidebar. Overall, we are taking a cautious approach in developing the system and are sticking with things we have experience with.

III. Architecture Design

III.1. Overview

The Team Toto development team has decided on a Bloc, Business Logic Component, and architectural pattern for this application. The team concluded with the justification that this application is meant to be a Cross-Platform Application. Thus, there will be a user view, a database model, and a business logic-driven control. We believe that given the context of this application, any other architectural design pattern doesn't make sense.

(See [Appendix A Image 1](#))

Our team is using this diagram to base our development around, and it was used for subsystem decomposition, which is expanded upon in the next section. Here, we will provide an overview of every component within the diagram, with expanded details to follow in the next section.

III.2. Subsystem Decomposition

III.2.1. Controller

III.2.1.1. Description

In general, the Bloc design pattern, and any Bloc-based application, will wait and listen for user inputs and will send this data in the form of an Event to the Business Logic Layer of the application. From there, Bloc will send requests to the database and will receive a response in an asynchronous manner just as it did from the user end. Bloc will then transmit the response to emit a state back to the user end of the application. Specific to this application, the Bloc subsystem will represent the connection between the user on the front end of the application, the database, and the state of the application. It will utilize business logic in order to create posts, comments, spaces, and more.

III.2.1.2. Concepts and Algorithms Generated

The Bloc subsystem will be composed of a Bloc class, with some smaller systems as part of the greater Bloc subsystem.

III.2.1.3. Interface Description

We will leave the Bloc Interface Description empty, as the Bloc subsystem consists of the Authentication and Computational Logic sections, and we can go into more detail within those individual subsystems.

III.2.2. Authentication

III.2.2.1. Description

The authentication subsystem is tied to the Bloc and the User Interface subsystems. It is a smaller subsystem, in charge of the constant authentication of user permissions, log-in checks, superuser privileges, and more. Linked closely to the User Interface subsystem, since it transmits states to the User Interface subsystem, and sends back events to the Bloc.

III.2.2.2. Concepts and Algorithms Generated

This subsystem has been created to encapsulate any behavior or functions in regard to authentication, permission handling, and password checking. It is necessary so that the subsystems that utilize authentication don't need to do their own authentication checks, and can instead focus on their more specific functions. This helps promote encapsulation.

III.2.2.3. Interface Description

III.2.2.3.1. Services Provided:

1. **Service name:** Login

Service Provided to: Bloc

Description: This service will authenticate the user's provided credentials with those present in the Firebase database. After authentication, a new user session is created.

2. **Service Name:** Logout

Service Provided to: Bloc

Description: This service will end the user session.

3. *Service Name:* hasPermissionTo

Service Provided to: Bloc

Description: This service, provided to many functions throughout the application, will state whether a given user has any given permission to any given requested resource or action. The permissions are pulled from the Firebase database, dependent on the specific function from the queried table.

III.2.2.3.2. Services Required:

1. Firebase database

III.2.3. Computational Logic

III.2.3.1. Description

The logic handling and running both the data layer and the presentation layer processes. This is where a majority of the code is, as it must handle the connection between the Presentation subsystem and the Data subsystem. Part of the greater Bloc subsystem.

III.2.3.2. Concepts and Algorithms Generated

This subsystem will have methods to handle the many different requests for the many different tables we have. These functions will retrieve data through the data access subsystems in order to run the application. There may be add<>(), update<>(), or delete<>() functions in which data comes from the User Interface to Bloc to update in the Data Layer. There may be get<>() functions in which data comes from the Data Layer to Bloc to display in the User Interface.

III.2.3.3. Interface Description (such as GET, POST, DELETE, PATCH, and PUT)

III.2.3.3.1. Services Provided:

1. *Service name:* GET

Service provided to: Presentation Layer

Description: A GET request, or a get<> function, is used to retrieve/request data from a database or server, in this application, from the Firebase database. It retrieves whatever information is identified within the request.

2. *Service name:* POST

Service provided to: Data Access Logic

Description: A POST request, or a post<> function, is used to create a resource. In this application, it creates a new resource within one of the tables, within the Firebase database.

3. *Service name:* DELETE

Service provided to: Data Access Logic

Description: A DELETE request, or a delete<> function, is used to delete an existing resource. In this application, it deletes an existing resource from a specific table, within the Firebase database.

4. *Service name:* PATCH

Service provided to: Data Access Logic

Description: A PATCH request, or a patch<> function, is used for updating an existing resource. In this application, it updates an existing resource within the specific table, within the Firebase database.

5. *Service name:* PUT

Service provided to: Data Access Logic

Description: A PUT request, or a put<> function, is used for checking if a resource already exists then updating, or otherwise creating a new resource if it doesn't already exist. In this application, it either updates an existing resource within the specific table or creates a new resource within one of the tables, within the Firebase database.

III.2.3.3.2. Services Required:

1. Firebase database
2. User Interface, aka Presentation Layer

III.2.4. Data Layer

III.2.4.1. Description

The Data Layer subsystem will be composed of a Model class, and some smaller systems part of the greater Data Layer subsystem. The general purpose of the model class is to connect and communicate with the database/data provider, take that information and parse/translate it, and send it to the specific Bloc function called from the Bloc subsystem.

III.2.4.2. Concepts and Algorithms Generated

The Data Layer subsystem consists of the Data Access Logic, the Firebase, and the DartKB subsystems. These subsystems work together to allow for the transfer, updating and deleting of information between the Bloc, and the Database.

III.2.4.3. Interface Description

We will leave the Data Layer's Interface Description empty, as the Data Layer's subsystem consists of the Data Access Logic, DartKB, and Firebase sections, and we can go into more detail within those individual subsystems.

III.2.5. Data Access Logic

III.2.5.1. Description

The logic allows for data updating and retrieval. Also known as the Data Access Layer, this subsystem is housed within the Data Layer subsystem and is responsible for providing access to data stored in persistent storage. In this example, the persistent storage is our Firebase database, within which our data is stored.

III.2.5.2. Concepts and Algorithms Generated

The Data Access Logic generates queries to add or retrieve data to the USER, POST, COMMENT, POST SPACE, and PERMISSIONS tables. It must instantiate objects for these data types when necessary and add new entries to the appropriate tables.

III.2.5.3. Interface Description

III.2.5.3.1. Services Provided:

1. *Service name:* Firebase Protocol

Service provided to: Firebase database

Description: This service provides a connection between the Firebase database and the Data Access Logic.

2. *Service name:* AddEntry

Service provided to: Firebase database

Description: This service adds any given new entry to the appropriate model table, using data received from the Bloc subsystem.

3. *Service name:* UpdateEntry

Service provided to: Firebase database

Description: This service updates any given entry in its corresponding table, using data received from the Bloc subsystem.

4. *Service name:* RemoveEntry

Service provided to: Firebase database

Description: This service removes a given entry from its corresponding table, provided by the Bloc subsystem. .

5. *Service name:* returnData

Service provided to: Firebase database

Description: This service receives data from the Firebase database using getData (see below), parses and translates the data to the correct object, and returns it to the process it was called within.

III.2.5.3.2. Services Required:

1. *Service name:* getData

Service Provided from: Bloc

III.2.6. Firebase

III.2.6.1. Description

Firebase is a Backend-as-a-Service acquired and backed by Google known for its realtime database. Firebase provides a variety of tools for developing, handling, and enhancing applications, allowing the application to be stored online.

III.2.6.2. Concepts and Algorithms Generated

This subsystem allows for the connection between the application and the database, which would be stored in Google Cloud.

III.2.6.3. Interface Description

The group is not 100% on how the database will be utilized, between cloud hosting vs server hosting. As we are awaiting a decision from the client, and throughout development will be using Firebase on Google Cloud, we will be leaving the Interface Description empty.

III.2.7. Presentation Layer

III.2.7.1. Description

What the user is physically seeing. The Presentation subsystem controls and handles the application's data representation and user interactions. The Presentation Layer consists of a tree of widgets, originating from the route widget, and extending with more and more as it goes deeper into the application. The widgets interact together, along with some methods from the Bloc subsystem, to run the page that is sent to the client through IIS. The view will be similar, but slightly different for the Mobile view and for the Chromebook view. Both are handled through the same subsystem.

III.2.7.2. Concepts and Algorithms Generated

The Presentation's many functions are dispersed across many smaller subsystems, similar to that of the Data and the Bloc subsystems. However, since it is more general in the Presentation subsystem compared to the others, these will all be handled under the general Presentation subsystem.

III.2.7.3. Interface Description

III.2.7.3.1. Services Provided:

1. **Service name:** DisplayPage

Service provided to: IIS

Description: This service displays all the widgets of the requested page.

2. **Service name:** UpdatePage

Service provided to: IIS, Computational Logic System

Description: This service updates the requested page's widgets after receiving an update from the Computational Logic System.

3. **Service name:** DisplayPopup

Service provided to: IIS

Description: This service displays a popup with the corresponding requested popup.

4. *Service name:* UpdatePopup

Service provided to: IIS, Computational Logic System

Description: This service updates the requested popup after receiving an update from the Computational Logic System.

5. *Service name:* ClosePopup

Service provided to: IIS

Description: This service closes the requested popup.

6. *Service name:* sendGetRequest

Service provided to: Computational Logic subsystem

Description: This service takes user-updated information and sends it to the Computational Logic subsystem to be handled.

III.2.7.3.2. Services Required:

1. IIS
2. Bloc Subsystem

III.2.8. IIS

III.2.8.1. Description

The Internet Information Service is how the user gets their application page loaded from the program to their screen. Basically, users have to be connected to the internet in order to access the application, and the application's back end also has to be connected to the internet to serve users.

III.2.8.2. Concepts and Algorithms Generated

In terms of our project, this subsystem is a gateway/buffer between the user and the Presentation subsystem.

III.2.8.3. Interface Description

III.2.8.3.1. Services Provided:

1. *Service name:* sendData

Service provided to: Presentation

Description: This service sends data in the form of an event from the User to the Bloc subsystem.

2. *Service name:* receiveData

Service provided to: Computational Logic

Description: This service receives state data from the Bloc subsystem, to then display to the User.

III.2.8.3.2. Services Required:

1. Presentation
2. Computational Logic

III.2.9. User

III.2.9.1. Description

This subsystem is self-explanatory, but the user is anyone using the application. They have to be connected to the internet in order to use the application, and their device has to be an allowed device.

III.2.9.2. Concepts and Algorithms Generated

There are no concepts or algorithms present in the User subsystem.

III.2.9.3. Interface Description

The Interface Description is extremely simplified in this “subsystem”. There is user input and the user’s connection to the IIS.

IV. Data design

(See [Appendix B](#) Image 1)

The **User** entity will hold every created user (both Faculty and Students) and will distinguish the different types of users with the **userType** attribute. This entity includes basic profile attributes such as **firstName**, **lastName**, **email**, **password**, and **profilePicture**. The **parentEmail** will be required for students as their parents will be constantly receiving a report of their child’s activity in the application. The **email** attribute is the primary key that the user will be able to log in with. The **User** entity has a many-to-many relationship with **PostSpace**. This relationship has a **Permissions** composite attribute, meaning every **User** will have unique **Permissions** in each **PostSpace**. These permissions include **canComment**, **canInvite**, **canEdit**, **canPost**, **canRemove**, and **canDelete**. The **User** entity also has a one-to-many relationship with **Post**.

The **PostSpace** entity is distinguished by the **spaceName** attribute and has basic attributes such as **spacePicture** and **spaceDescription**. The **isPrivate** attribute indicates whether the Post Space can be discovered by the “search post space” function where users can request to join. The **PostSpace** entity has a one-to-many relationship with **Post**.

The **Post** entity is a weak entity that uses the **timePosted** attribute as a weak key. It uses **isPinned** and **canComment** to hold the settings of each post. Other attributes include **title** and **contents**. The **Post** entity has a one-to-many relationship with **Comment**. Since the **Post** entity is a weak entity, it is identifiable by the primary keys of the **User** and **PostSpace** entities along with its weak key.

The **Comment** entity is a weak entity that uses the **timePosted** attribute as a weak key. The only other attribute is **contents**. Since the **Comment** entity is a weak entity, it is identifiable by the primary keys of the **User** and **PostSpace** entities along with the weak keys of itself and **Post**.

V. User Interface Design

Our team has created UI Mock-Ups for most of the different pages and boxes for our application. While these can give an idea of what the application may look like, pages are all subject to change as the development team or as the client deems fit. The UI Mock-ups are used to help guide development over deciding on the look of the application. Additionally, these mock-ups only show the Chromebook view, not the mobile view - they will look similar. It helps establishes and determines links in a way that's easier to come up with, from that visual perspective. Thus, the mockups should give you an idea of how the application will be used.

Note: We have not created anything pertaining to the **Superuser** pages, since those will look much different. They may include pages such as *Add User*, *View All Users*, *View All Spaces*, etc.

V.1. Login Page

(See [Appendix C Image 1](#))

First, we have the login page. This will be the landing page when users first open up the application. The background is images of Wahkiakum County's landscape but can be any image determined by the client. It can be any particular image, a simple graphic, a blank colored background, or a slideshow of many images. Regardless of which one is used, the images must not contrast in a way that is too distracting from the login block. The application name can be displayed in the corner.

Users will log in with their school-associated email, and a password. Students (whose accounts are created for them) will have to change their password from a temporary password to their own password, provided it meets all password requirements. As long as the email and password match in the database, they can log in. If incorrect, they return to the same page, with a corresponding message of "Incorrect username or password. Please try again". Users can also press "Forgot your password" to go to a password reset page (see [item V.2](#)). Superusers can press "Superuser Login" to go to the Superuser Login page (see [item V.3](#)).

V.2. Forgot Password Page

(See [Appendix C Image 2](#))

Users will come to this page from the Main Login Page (see [item V.1](#)). Users will be able to enter their school-associated email, and submit it to reset their password. We do not know which method we will take for this yet, but it will most likely end up back on the application and look like the password reset page (see [item V.18](#)).

V.3. Superuser Login Page*

(See [Appendix C Image 3](#))

Users will come to this page from the Main Login Page (see [item V.1](#)). Users who have superuser privileges (IT, Ron, etc.) will be provided with a Superuser Key which allows them administrative privileges, and a different Superuser main page view. As per the note above, we do not have UI mock-ups of these pages as they are more specific, and will be completed in the second half (or later) of the development lifecycle.

V.4. Main Home Page

(See [Appendix C Image 4](#))

The Main page is where you come after a user successfully logs in, from the Main Log In page (see [item V.1](#)). The application is split between the current page, the sidebar on the left side, and pop-outs for specific functions. The current page of the Main home page is the landing page of all users. It will contain the personalized Mecha-Mules home page, containing links that the client will provide. It can open those links into new browser tabs. The sidebar will contain a Home button (to return here), buttons for all the class spaces the logged-in user is a part of (see [item V.10](#)), and the user's profile page button (see [item V.5](#)).

V.5. Side Tab Extended

(See [Appendix C Image 5](#))

Here, we can see what happens when you press the logged-in user's profile picture (in the bottom left). It will extend the sidebar (from taking ~1/8th to ~1/3rd of the screen). In this view, the user will see their profile picture, name, a logout button, a link to the notifications page, a link to edit their profile page, and a link to the settings page.

V.6. Mouse-over any Button

(See [Appendix C Image 6](#))

Here, we can see what it looks like when you hold the mouse over any clickable link or button. It will display some text pertaining to its function. In this example, the mouse is over the New Class Space button, which when clicked will take the user to the Join/Create Class Space Popout (see [item V.7](#)).

V.7. Join/Create Class Space Popouts

(See [Appendix C Image 7](#))

Here, we arrive here when the user clicks the New Class Space button from the Sidebar (see [item V.6](#)). The user can see one of these two different popouts, depending on the permissions

given to their account. Faculty (or those with permissions) will see the right image, that is, they can create a class space in addition to joining a class space. To join a class space, users must have a class space code, which will be provided to them when invited by the space owner (see [item V.14](#)).

V.8. Mouse-over Profile Image

(See [Appendix C Image 8](#))

Similar to [Item V.6](#), this is a view of when the user mouses over their profile picture, in the extended profile sidebar (see [item V.5](#)). If they click this, they go to the profile popout (see [item V.9](#)), specifically their own.

V.9. View any profile popout

(See [Appendix C Image 9](#))

We arrive here when the user clicks their own Profile Image through the extended sidebar, or any other user's profile image. The Profile Page popout is a popout that shows a user's profile picture, their name, their description, the class spaces they are a part of, and an edit profile button (if they are viewing their own), or edit/deletes profile aspects buttons (if they are a faculty/superuser viewing a student's profile).

V.10. Class Space Page

(See [Appendix C Image 10](#))

We Arrive at the Class Space when a user clicks the corresponding Class Space button from the sidebar. Here, users can see information about the class space, and the post history, where posts are placed. Posts have some comments (all can be viewed in the post view), and this view shows all posts filtered by the most recent. Any user can create a post. All users can click view post, bringing them to the View Post Page (see [item V.11](#)). Users with administrative privileges can edit or delete any post. We are still considering if we want these buttons to be visible on this page, or only on the view post page.

V.11. View Post Page

(See [Appendix C Image 11](#))

Users arrive at the View Post Page when a user clicks "View Post" from the Class Space Page (see [item V.10](#)). Here, they can see Post details such as the title and post date, the full description, and all comments. Additionally, there is a Post Comment text box, where users can

post a comment response to the current post. Users with administrative privileges can edit or delete any comment, or the current post (see [item V.12](#)).

V.12. Delete Post (or Post Space) Popout

(See [Appendix C Image 12](#))

Users can delete their own posts, post space, or comment. Users with administrative privileges can delete any post, post space, or comment. Users arrive here when a user clicks “Delete Post”, “Delete Comment”, or Delete “Post Space”. Here, the users have to confirm the deletion before the database is updated and the post/post space/comment is deleted.

V.13. Mouse-over Member Count

(See [Appendix C Image 13](#))

Similar to [Item V.6](#), this is a view of when the user mouses over the member count in the Class Space view (see [item V.10](#)). If they click this, they go to the view member list pop out (see [item V.14](#)).

V.14. View Member List Popout

(See [Appendix C Image 14](#))

Users arrive here when they click the member count from the Class Space view. This displays the list of members in the class. Here, any user can view the profile popout of any other user, or message them. Users with administrative privileges can invite/add new users, or remove users from the space.

V.15. Remove User Popout

(See [Appendix C Image 15](#))

Users with administrative privileges arrive here when they click “Remove User” from the View Member List Popout. They have to confirm that they want to remove the user from the space.

V.16. Search for/add User Popout

(See [Appendix C Image 16](#))

Users with administrative privileges arrive here when they click “Add User” from the View Member List Popout. They get to a list of all users in the database, where they can add users to their space, or view user profiles. The list can be filtered (by year, by name, etc.).

V.17. Edit Profile Page

(See [Appendix C Image 17](#))

Users arrive at this page when they click “Edit Profile Page” from the Extended Side Tab (see [item V.5](#)). This allows users to change their profile picture, their description, and their password, in addition to text talking about how their posts/comments/profile pictures/profile descriptions can be edited by users with administrative privileges at any time. There is a “Change Password” button (see [item V.18](#)) and a “Change Profile Picture” button (see [item V.19](#)). Administrative users who are editing a student’s profile page will have a similar page.

V.18. Change Password Popout

(See [Appendix C Image 18](#))

Users arrive at this popout when they click “Change Password” from the Edit Profile Page (see [item V.17](#)). They must provide their old password, their new password, and their new password again to update it. Checks will be performed to verify all this information. Users can click Save Changes. If there are mistakes or errors, a corresponding error message will be displayed. If no errors or mistakes, users return to the previous page, and the database is updated.

V.19. Change Profile Picture Popout

(See [Appendix C Image 19](#))

Users arrive to this popout when they click “Change Profile Picture” from the Edit Profile Page (see [item V.17](#)). They can upload a picture file to display as their profile picture. Users click save changes to update their profile picture in the database and return to the previous page. Administrative users who are editing a student’s profile page will have a similar page.

VI. Glossary

Administrative Moderation - Faculty users with administrative accounts have extra permissions that student users do not, such as editing and deleting other users' posts. These accounts also have permission to moderate the behavior of student users.

Authentication - A form of validating a user's ID and password are within the database and correct. Once validated, users can log in.

Back-End - The server side; Everything necessary for an application that users do not see, including code/scripting, databases, software architecture, security, etc.

Cross-platform - Software that is designed to work on several computing platforms. In this application, software must work on both Chromebooks and mobile devices.

Chromebooks - Runs using Chrome OS. The main platform students use and the main platform for this project.

CIPA Compliance - The Children's Internet Protection Act our application must comply with.

Database - An organized collection of structured information, or data, typically stored electronically in a computer system.

Discord Voice Communication - A very popular VoIP (Voice over Internet Protocol) and instant messaging social platform that the school previously used.

Front-End - The client-side; everything that users actually see on an application, including the web pages, performance, UI, etc.

Full-Stack Application - The entire depth of an application, both front-end and back-end.

Software Architecture - The fundamental structures of a software system and the discipline of creating such structures and systems

Stakeholders - Individuals, groups, or organizations directly involved with, or indirectly affected by, a project, product, service, or enterprise.

UI (User Interface) - The space where interactions between humans and applications occur; all the physical buttons, text boxes, etc. that users interact with within an application.

User - Any entity that has an account on the application. Users will have specific permissions depending if they are faculty or student.

User Records - Data of user's posts, comments, space, and profile. These are all stored on a database used for functions throughout the application.

VII. References

"Children's internet protection act (CIPA)," *Federal Communications Commission*, 28-Apr-2020. [Online]. Available:
<https://www.fcc.gov/consumers/guides/childrens-internet-protection-act>. [Accessed: 04-Oct-2022].

“ChromiumOS Developer Guide,” *ChromiumOS Docs - ChromiumOS Developer Guide*. [Online]. Available:
https://chromium.googlesource.com/chromiumos/docs/+/HEAD/developer_guide.md. [Accessed: 04-Oct-2022].

“Cloud SQL for PostgreSQL Documentation,” Google Cloud. [Online]. Available:
<https://cloud.google.com/sql/docs/postgres>. [Accessed: 08-Oct-2022].

“Dart overview,” Dart. [Online]. Available: <https://dart.dev/overview>. [Accessed: 08-Oct-2022].

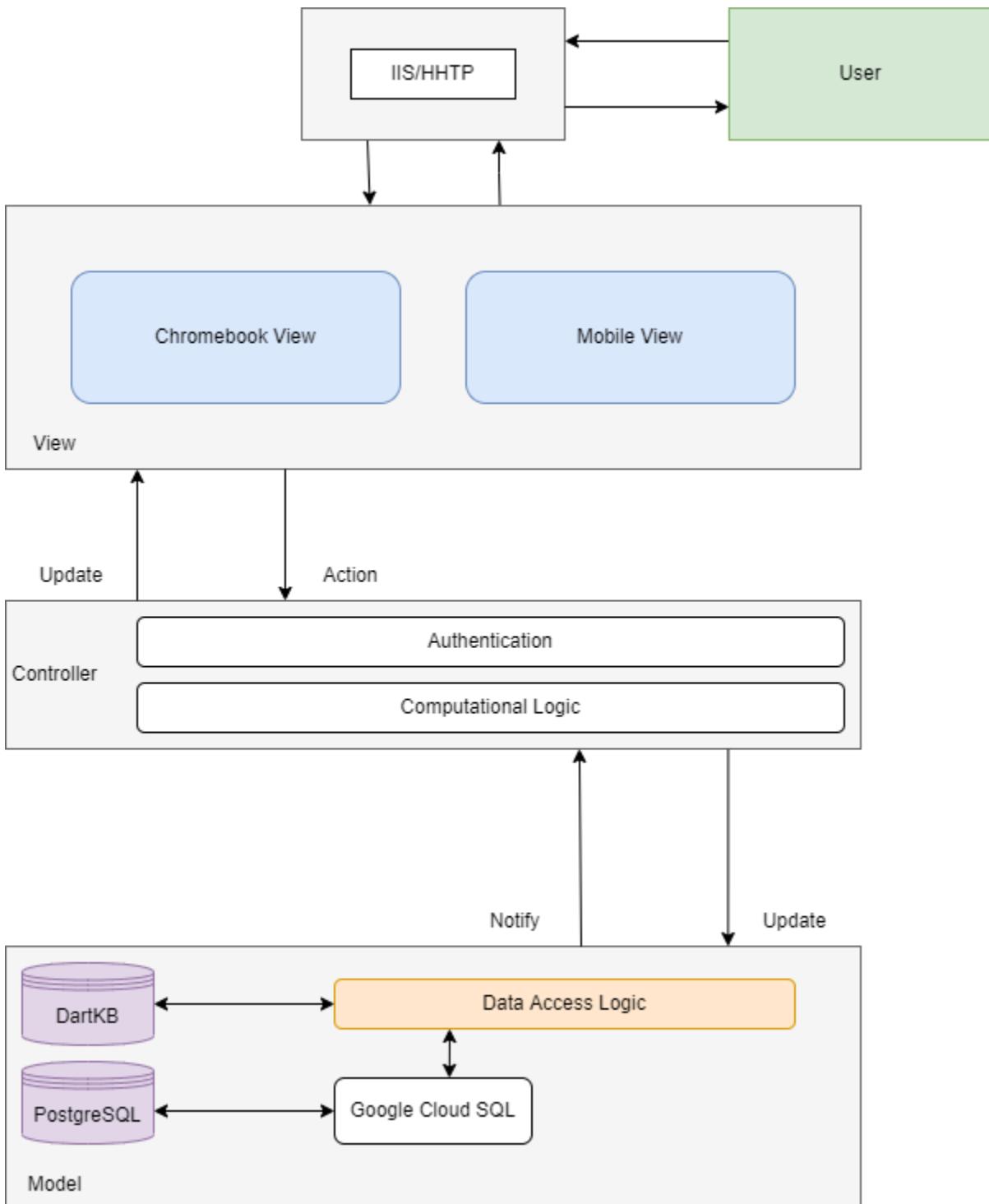
(Dutoit, 2010), 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, 2010.

“IEEE standards,” IEEE. [Online]. Available:
<https://www.ieee.org/content/ieee-org/en/standards/index.html/>. [Accessed: 03-Oct-2022].

S. Berkun, *Making things happen: Mastering project management*. O'Reilly Media, 2015.

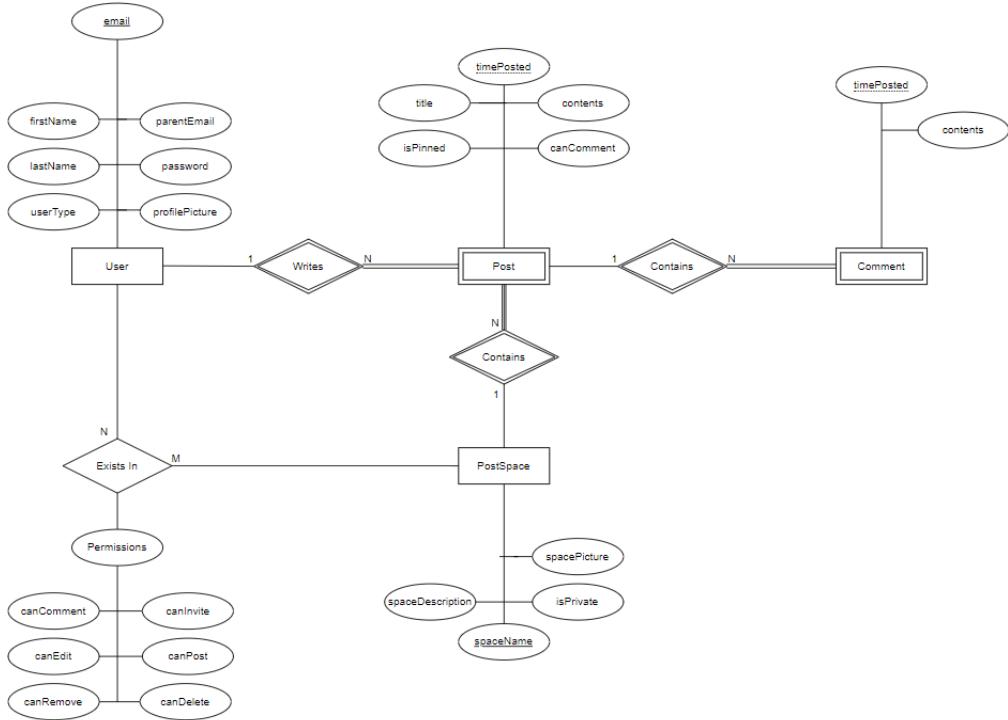
VIII. Appendices

VIII.1. Appendix A - Software Architecture



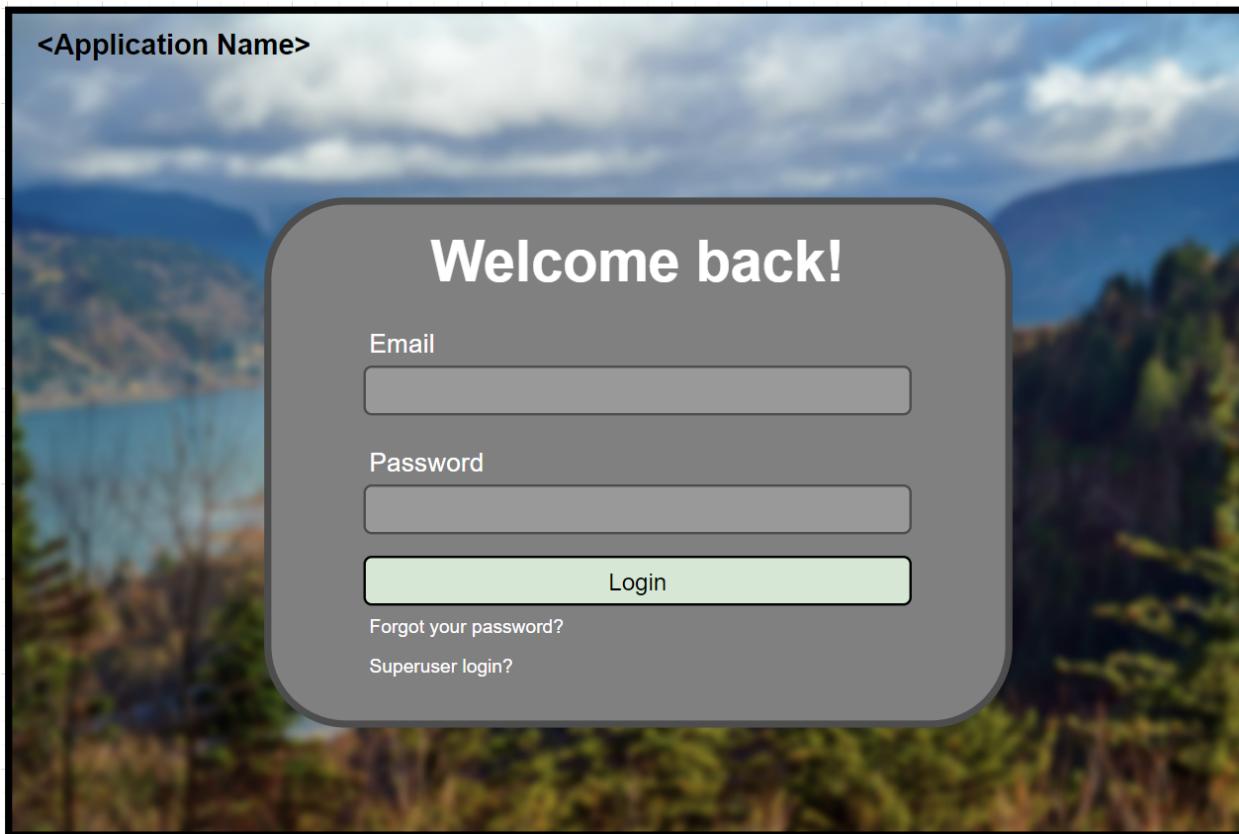
(Image 1)

VIII.2. Appendix B - Data Design

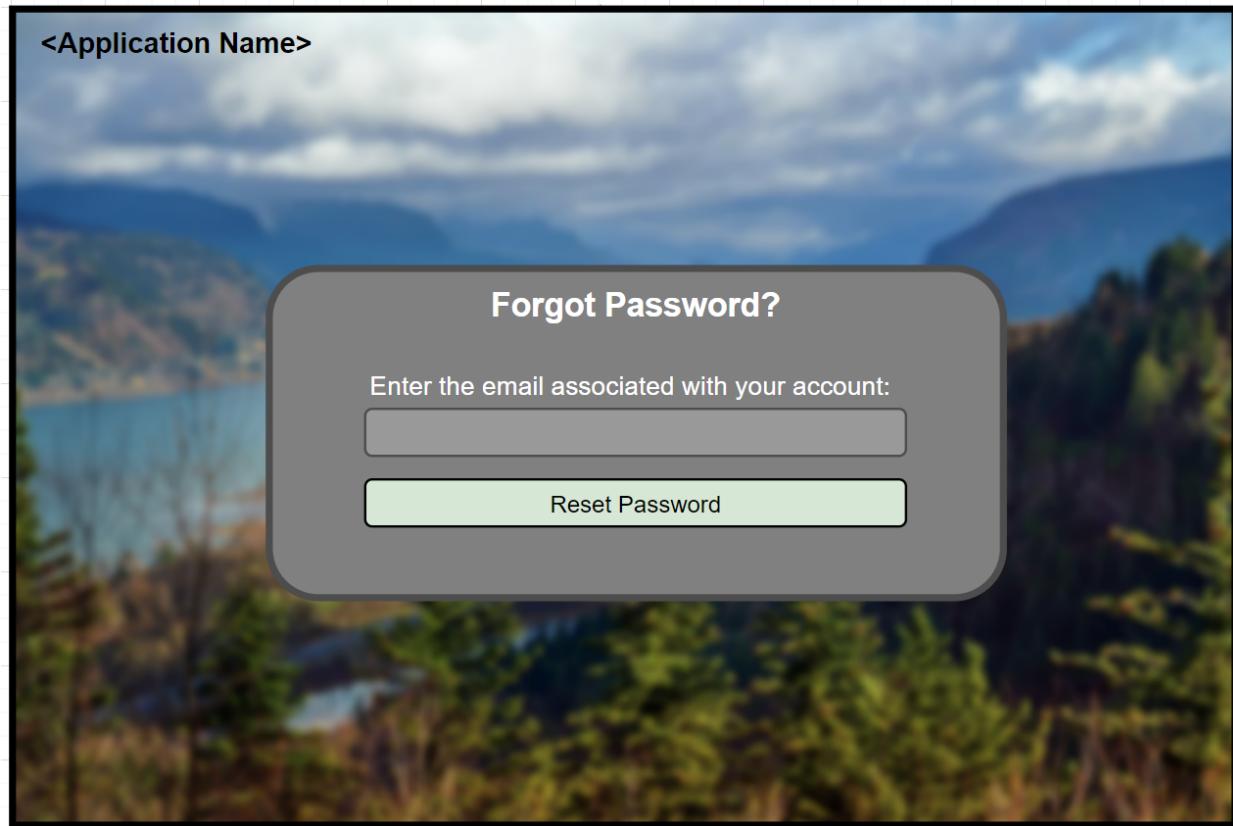


(Image 1)

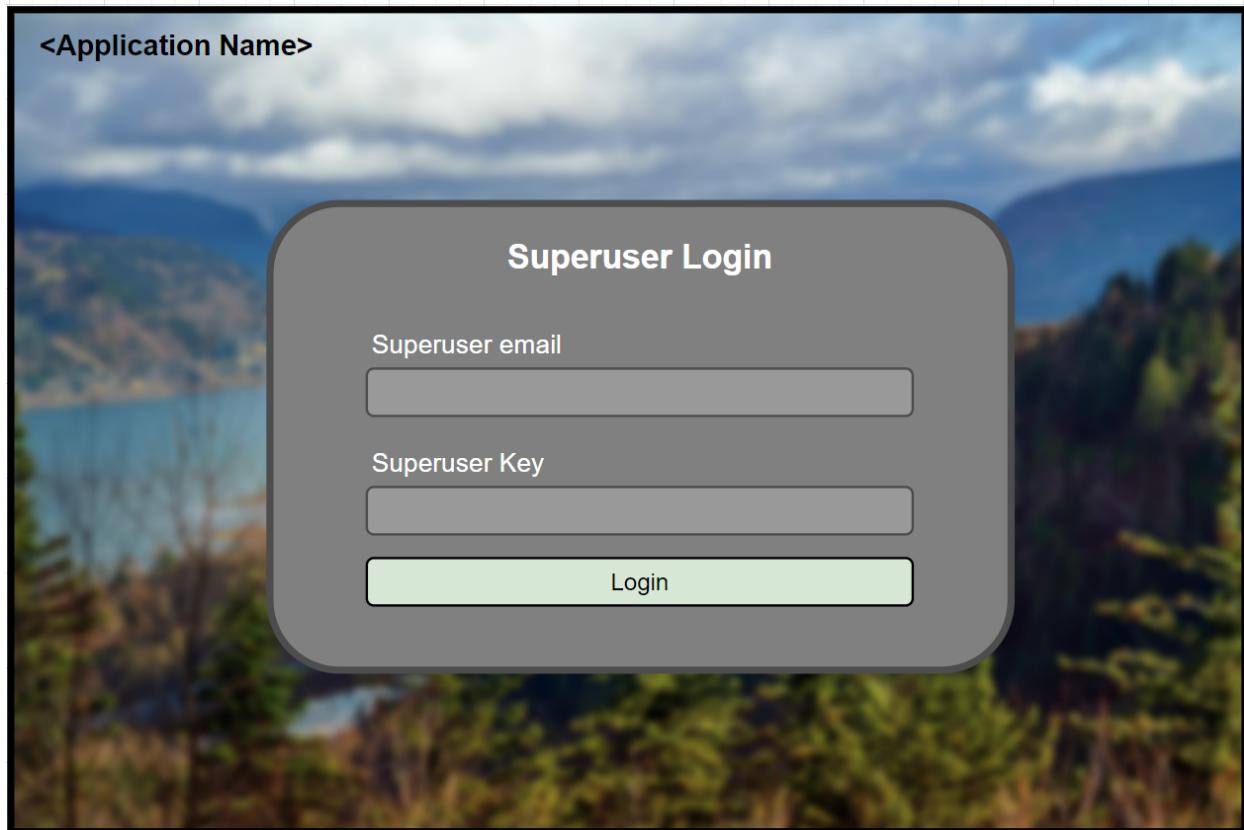
VIII.3. Appendix C - User Interface Design



(Image 1)



(Image 2)



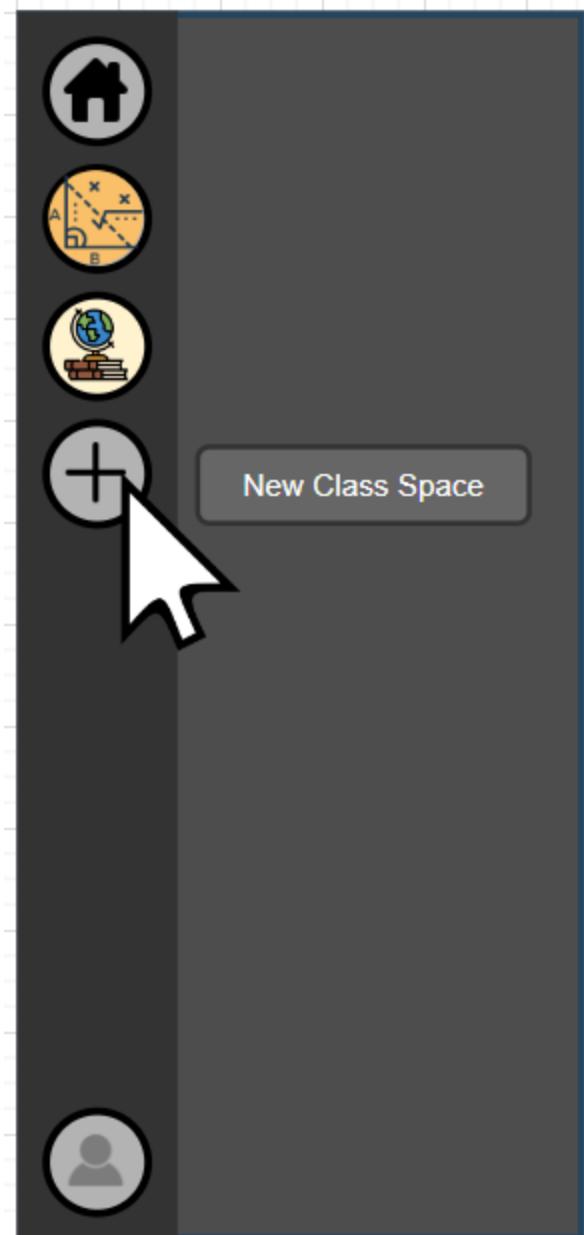
(Image 3)



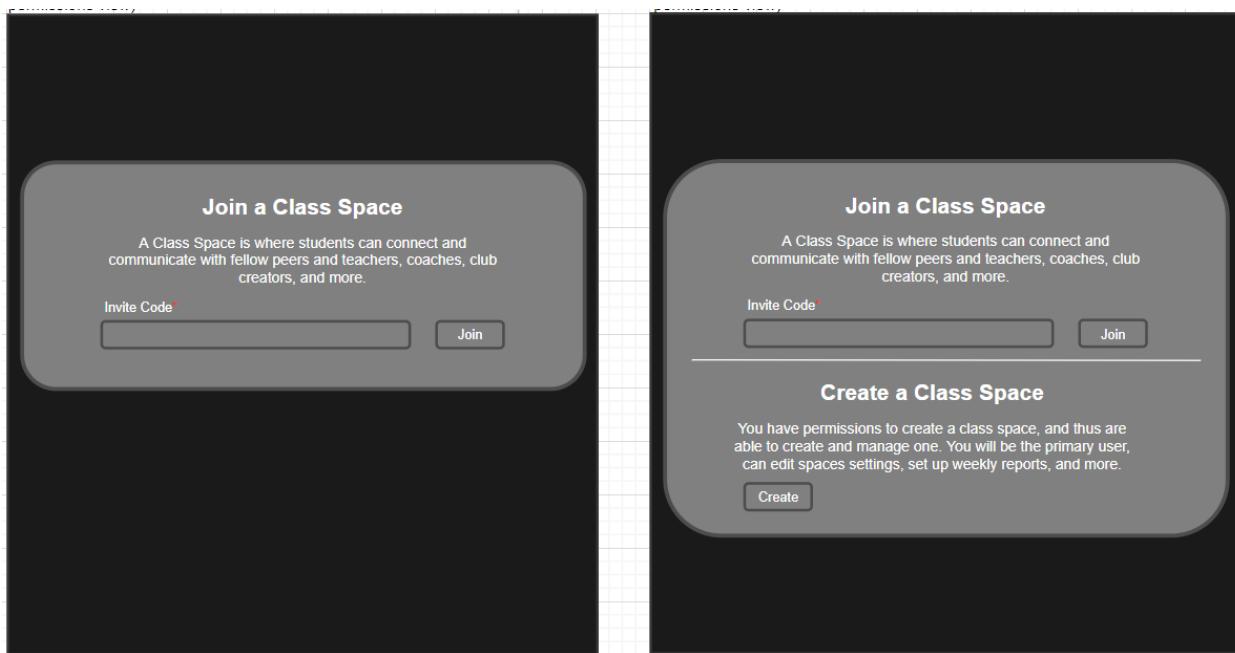
(Image 4)



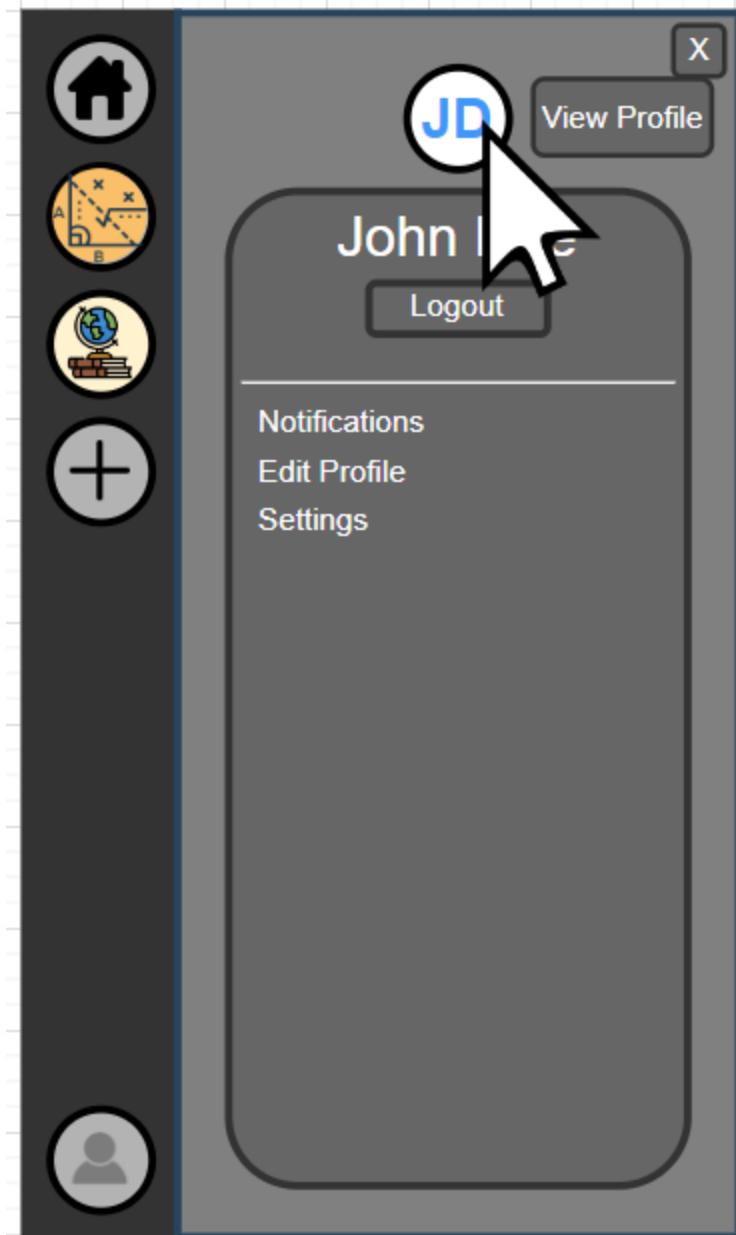
(Image 5)



(Image 6)



(Image 7)



(Image 8)



(Image 9)

The screenshot shows a class space page for "Mr. Doe's Geometry Class Space". On the left is a sidebar with icons for Home, Class Space, Post, Comment, and New Post. The main title is "Mr. Doe's Geometry Class Space" with "24 Members". Below the title is a post by "Mr. Doe" titled "Post 1" posted at "11:59pm, 10/8/2022". To the right is a "View Post" button. The post content is "Post 1 Description". Below the post is a comment by "Sally Exampleson" with the text "This is a student's comment under post 1!". To the right are "Edit Comment" and "Delete Comment" buttons. Below the comment is a comment by "John Doe" with the text "This is the current user's (Faculty John Doe) comment under post 1!". To the right are "Edit Comment" and "Delete Comment" buttons. Below the comments is another post titled "Post 2" with "Post 2 Description" and a link "TotallyLegitLink.com/NotAScam". At the bottom is a "New Post" form with a "Post" button.

(Image 10)

The image shows a dark-themed user interface for a digital platform. On the left side, there is a vertical sidebar with several circular icons: a house, a yellow triangle with a grid, a globe, a plus sign, and a person icon. The main content area has a dark background.

Post 1 Posted by Mr. Doe at 11:59pm, 10/8/2022.

Post 1 Description. This is where the rest of the description of the Post will be.

It can take
a lot of
Space.

Edit Post **Delete Post**

Comments

Sally Exampleson
This is a student's comment under post 1!

Edit Comment **Delete Comment**

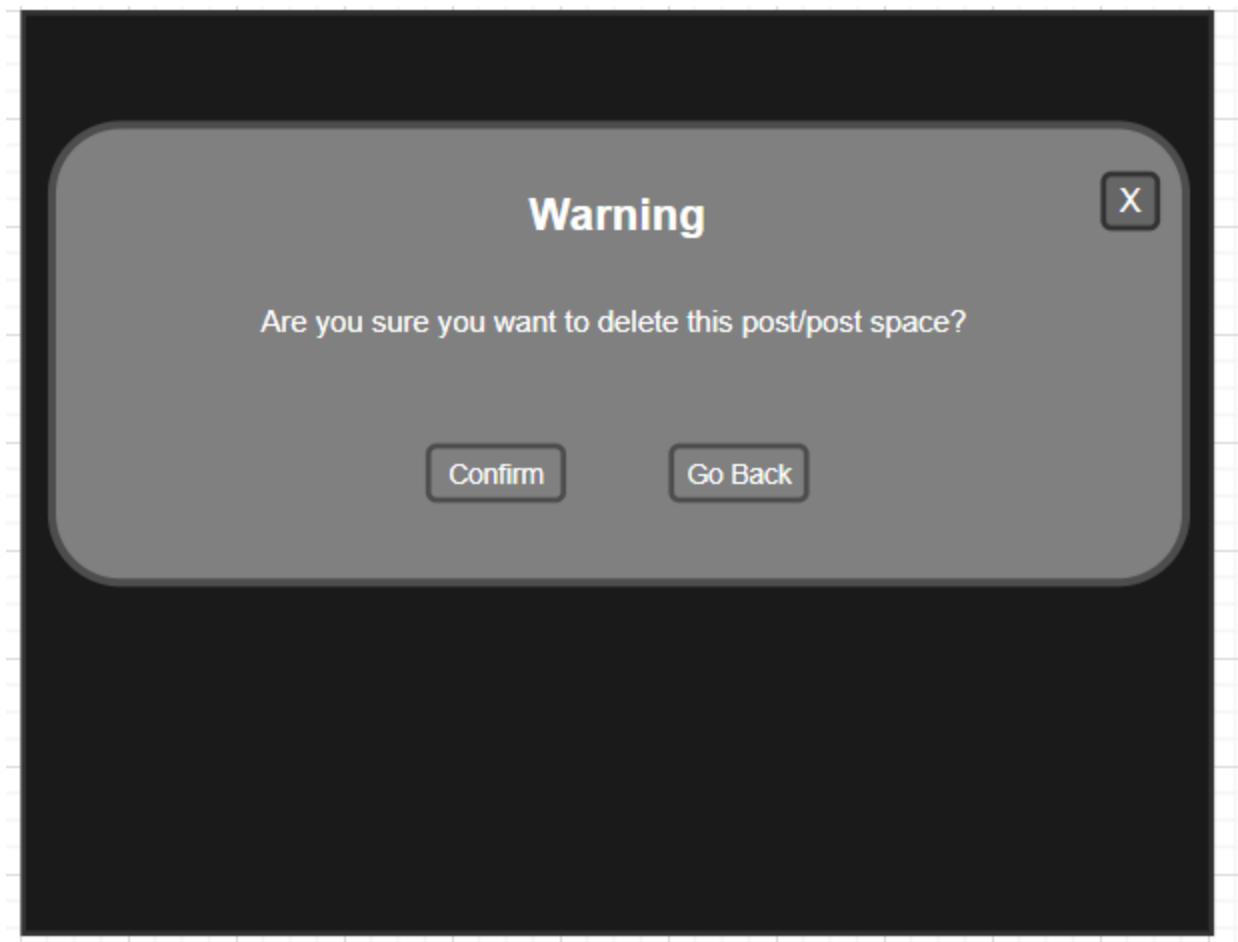
John Doe
This is the current user's (Faculty John Doe) comment under post 1!

Edit Comment **Delete Comment**

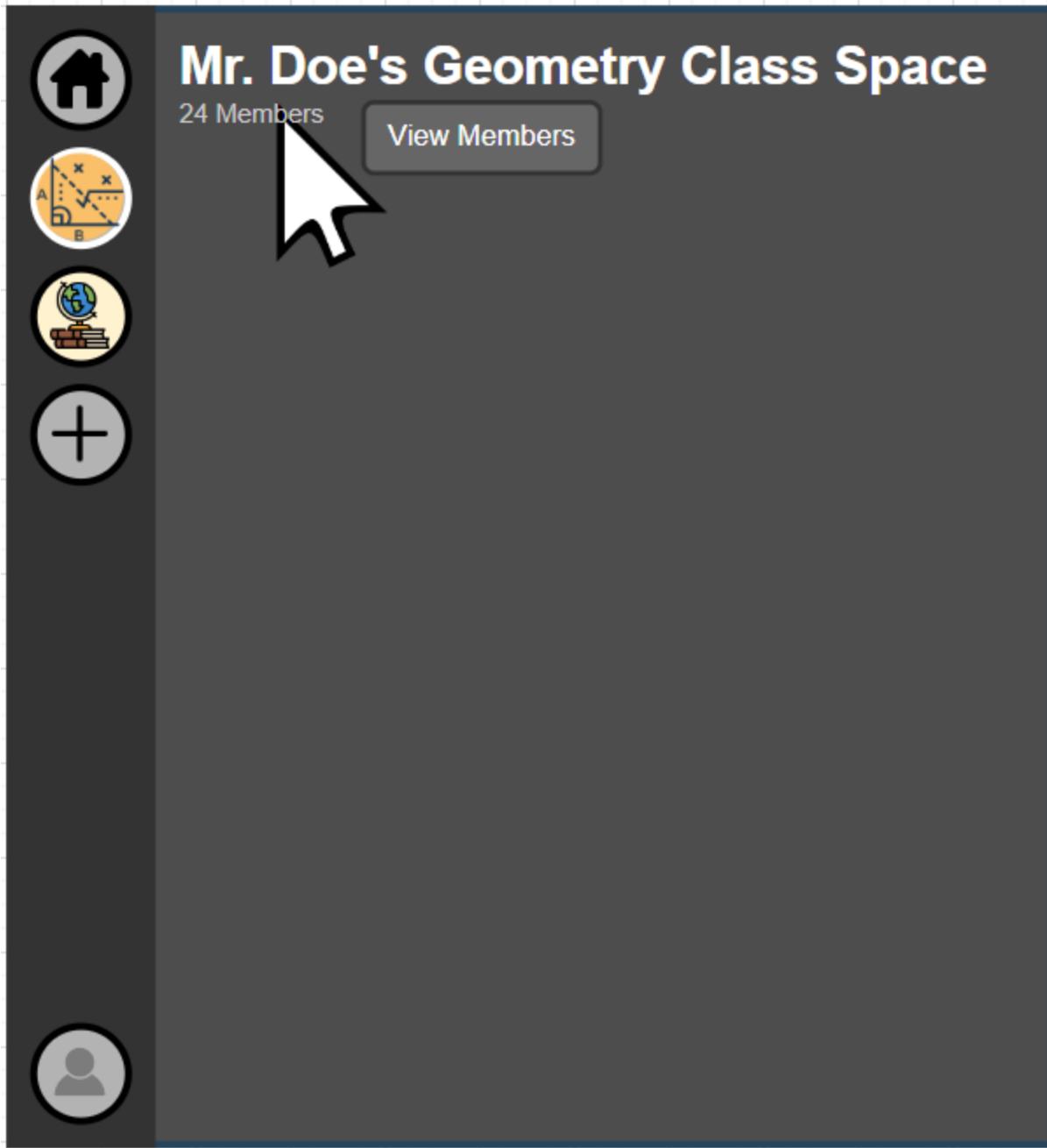
New Comment

Post Comment

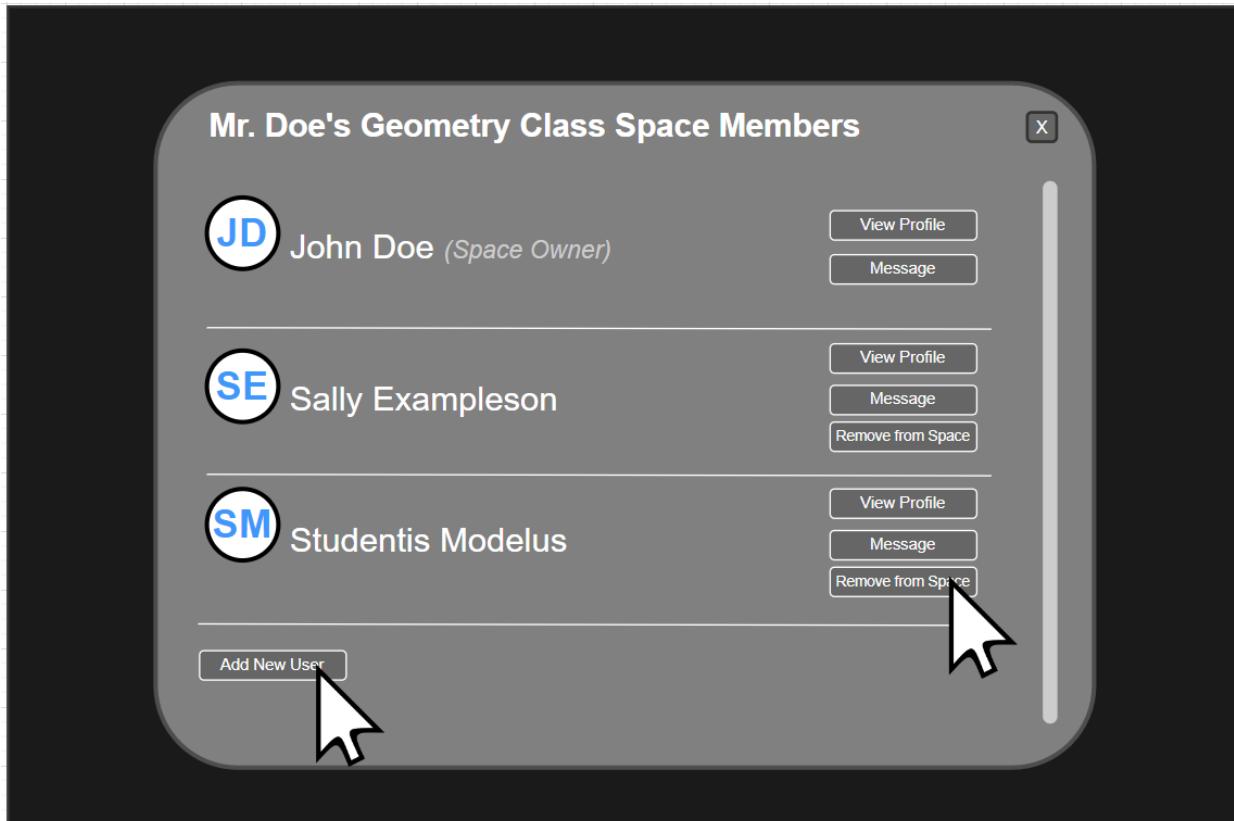
(Image 11)



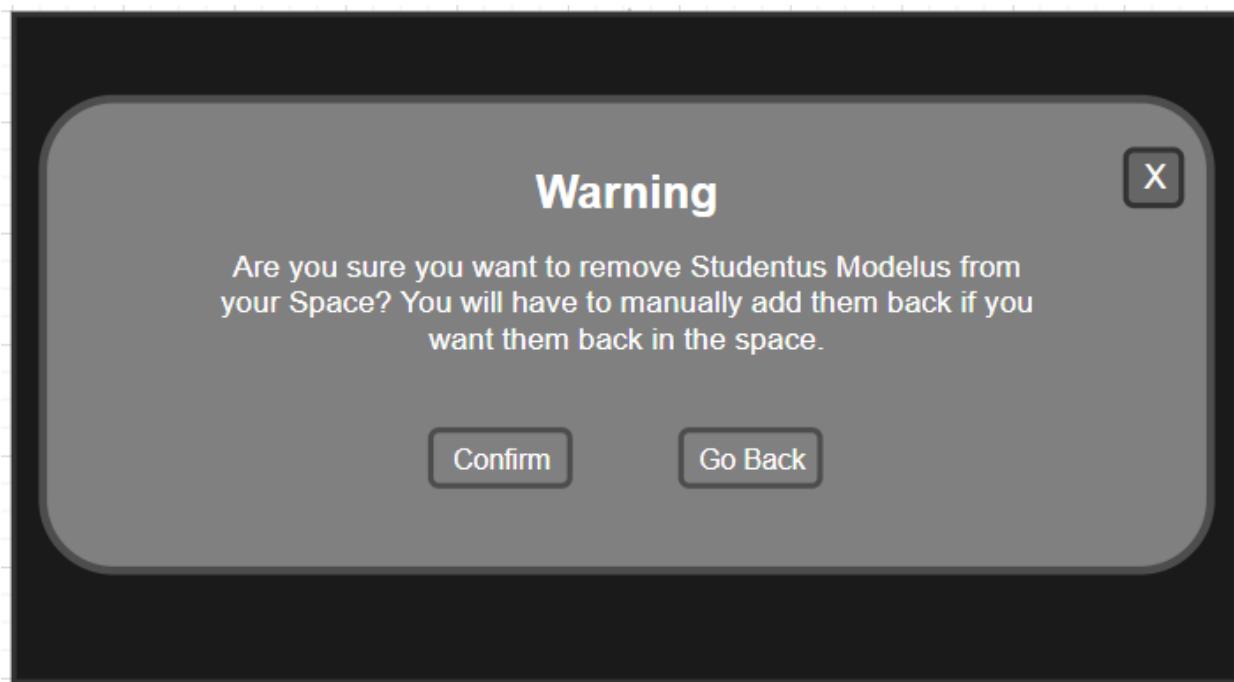
(Image 12)



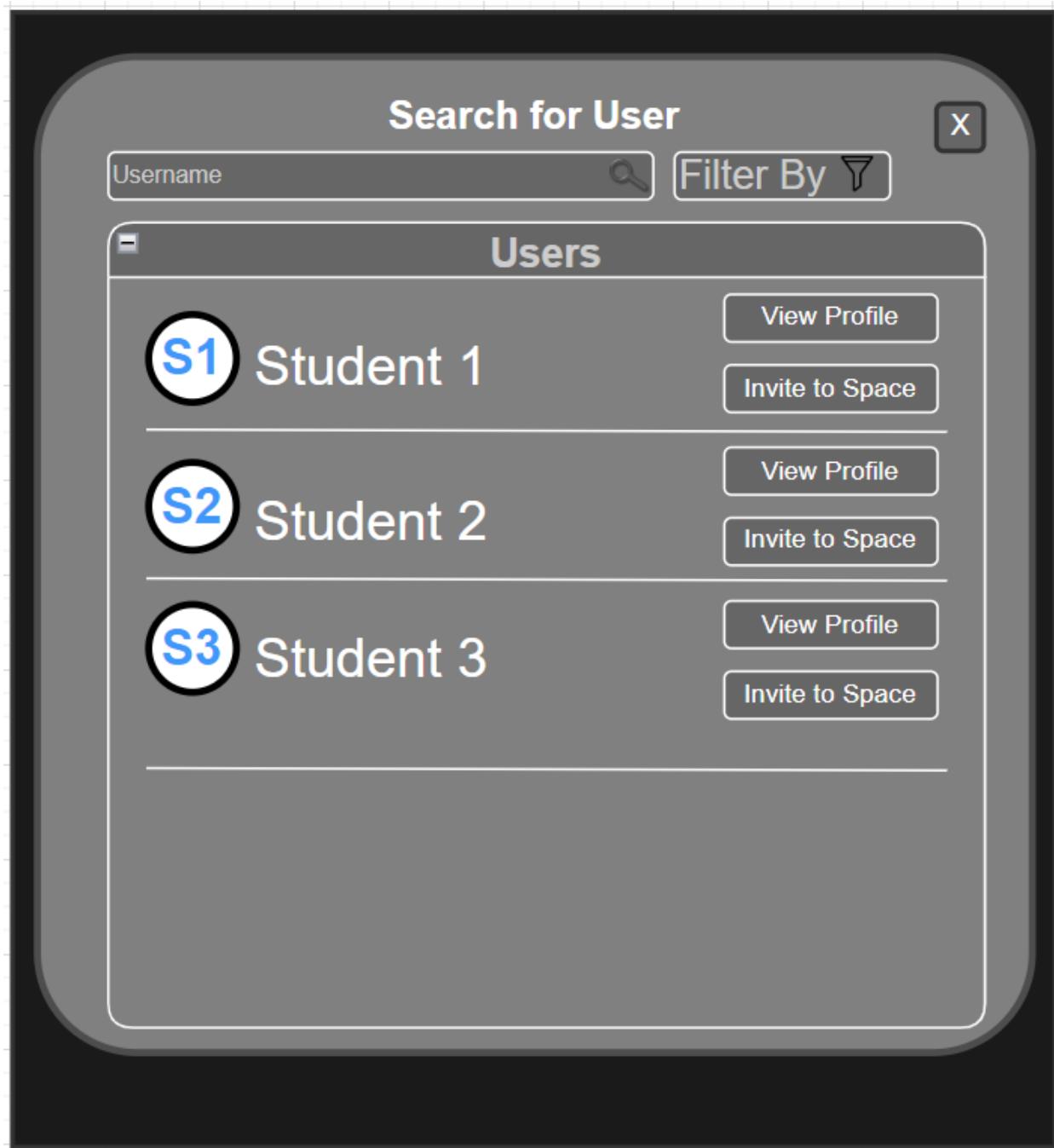
(Image 13)



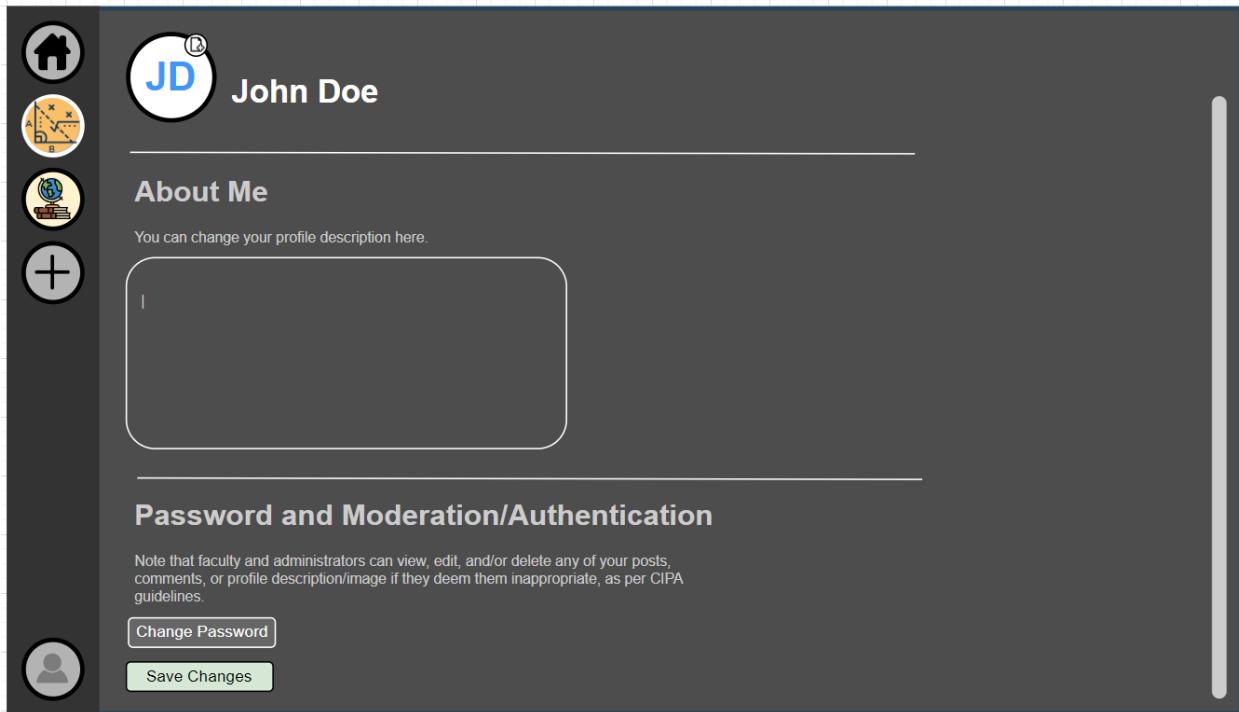
(Image 14)



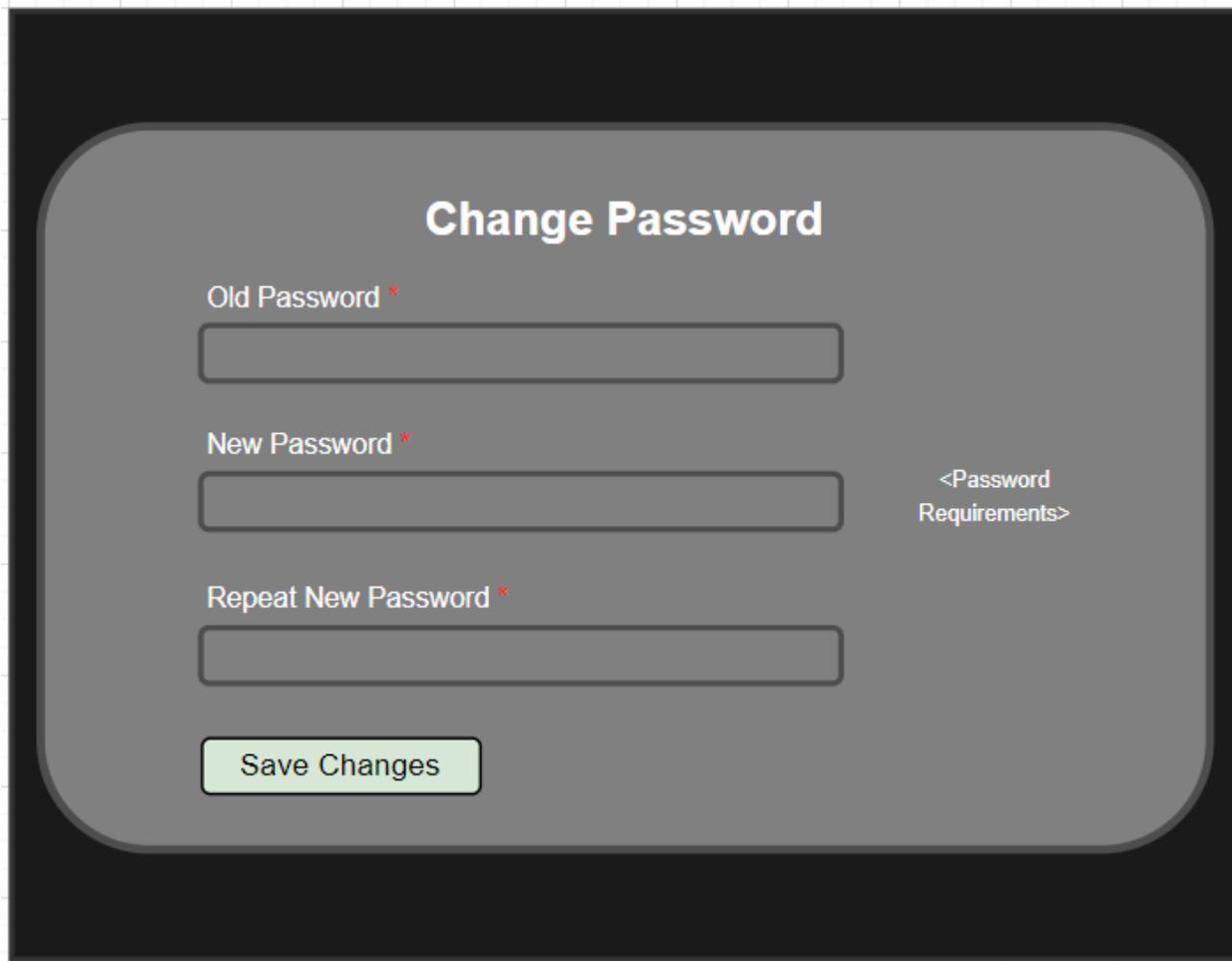
(Image 15)



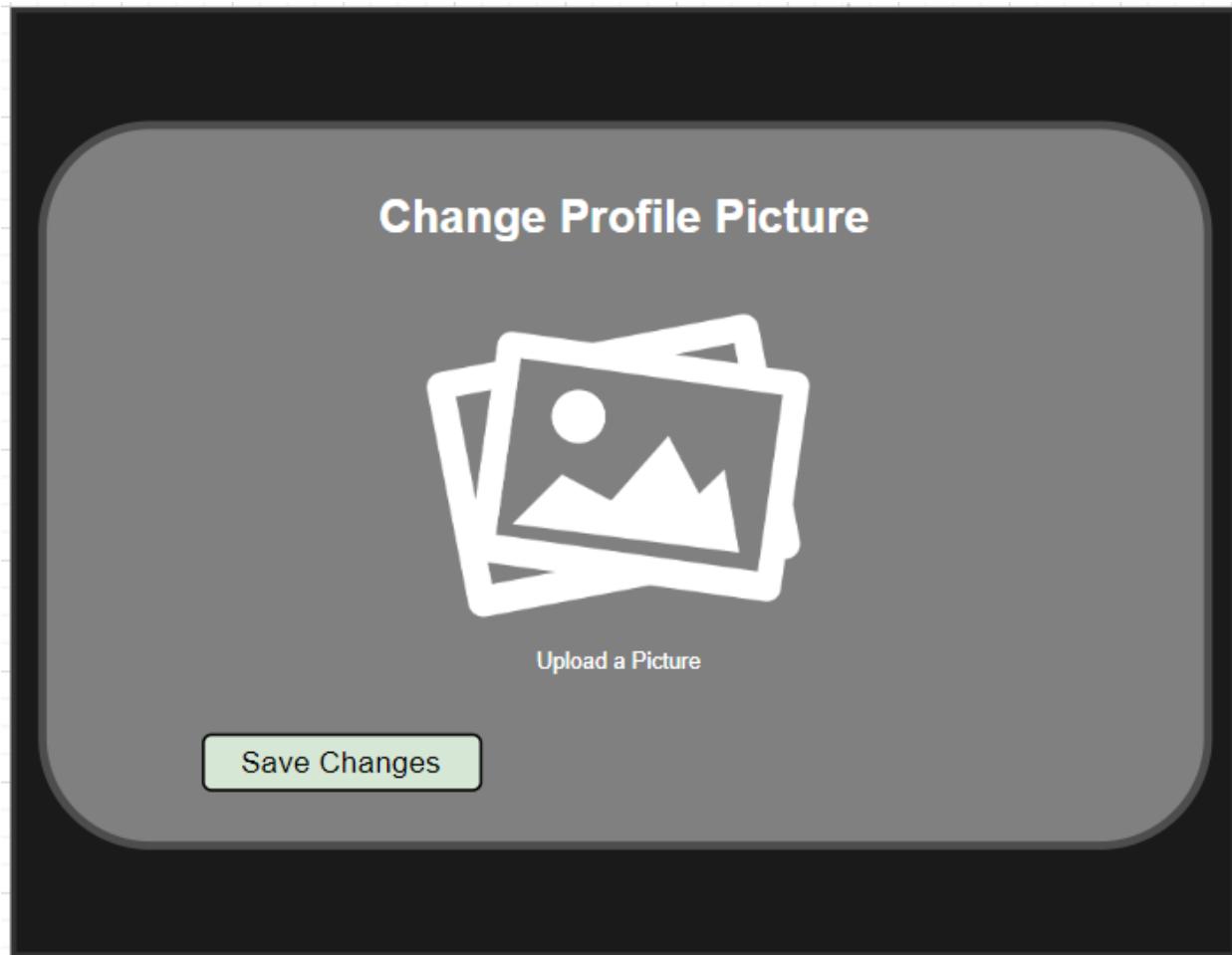
(Image 16)



(Image 17)



(Image 18)



(Image 19)