**SW Engineering CSC 648/848 Section 4**

# **EduBridge**

By

**Syntaax Squaad**

**Team 4**

| **Member** | **Role** |
| --- | --- |
| Naisarg Halvadiya | Team Lead |
| Shail Patel | Backend Lead |
| Dylan Nguyen | Scrum Master I |
| James Dixon | Scrum Master II |
| Pankuri Khare | Frontend Lead |
| Riken Kapadia | GitHub Master |

Milestone 2

03/22/2024

| **Milestone** | **Date** |
| --- | --- |
| 0 | 02/21/2024 |
| 1 | 03/06/2024 |
| 2 | 03/22/2024 |

# **Data Definitions**

## **user: An entity that stores registered user info**

### Attributes:

* + 1. userID - users unique Identifier
    2. displayName - users name
    3. email - user email
    4. password - user password
    5. userRole - the user role used to identify privileges, ie student, teacher, admin

### Privileges:

* + 1. Privileges are provided through the users role

## **role: Stores role type and permissions**

### Attributes:

* + 1. roleID - the roles identifier
    2. roleName - student , teacher, admin

### Privileges:

* + 1. student
       1. View content
       2. Upload content
    2. teacher
       1. View content
       2. Upload content
       3. Filter / Moderate content
    3. admin
       1. Set user roles

## **userRole: This entity sets the relationship between roles and users**

### Attributes:

* + 1. userID - the id of the user
    2. roleID - the id of role assigned to the user

## **content: Represents uploaded material the application serves**

### Attributes:

* + 1. contentID - content identifier
    2. userID - uploader id
    3. name - name of content
    4. description - contents description
    5. type - file type of item
    6. uploadDate - date content was uploaded
    7. location - crm location / file path
    8. viewCount - view count of content
    9. visibility - ie public/private

## **comment:**

### Attributes:

* + 1. commentID - comment identifier
    2. contentID - content the comments on
    3. userID - id of commenter
    4. text - comment
    5. uploadDate - comment date

## **like:**

### Attributes:

* + 1. likeID - identifier
    2. contentID - content liked
    3. userID - user who liked
    4. date - date liked

## **dislike:**

### Attributes:

* + 1. dislikeID - identifier
    2. contentID - content disliked
    3. userID - user who disliked
    4. date - date disliked

# **Functional Requirements**

## **User Creation / Management**

* + ID: 01
  + Description: Users should be able to register and log in. Users shall have roles such as admin, student, instructor, and assistant.
  + Priority: 1

### 1.1. Registration:

* Users shall register using a username, email, and password.

### 1.2. Login:

* Users shall be able to log in using their username and password

### 1.3. Role Management:

* A role must be assigned to each user, by default with lowest permissions student
* Admin accounts manage user roles.

## **Student Authentication**

* + ID: 02
  + Description: Users must have a .edu email.
  + Priority: 1

2.1. Email Verification:

* Users will only be allowed to register with a .edu email.
* Users will have to access this email to retrieve a OTP to verify the email.

## **Content Upload**

* + ID 03/1
  + Description: Users must be able to upload diverse types of content
  + Priority: 1

3.1. File Format Compatibility:

* Users shall be able to upload common video,audio, and text files.
  + MP4
  + PDF
  + DOCX

## **Content Playback**

* + ID: 03/2
  + Description: The application shall offer appropriate playback for common file types
  + Priority: 1

4.1. Video Playback:

* Users shall be able to watch uploaded videos in application.

4.2. Text Viewer:

* Users shall be able to view text files through a built in application renderer.

## **Content Download**

* + ID: 04
  + Description: Users shall download content from the platform for offline viewing
  + Priority: 2

## **Comment / Feedback**

* + ID: 05
  + Description: Users shall comment on uploaded content and reply to other comments. Additionally Users shall like or dislike content.
  + Priority: 2

6.1. Comment Posting:

* Users shall be able to post comments on content
* Users shall be able to post private feedback on content for uploaders
* Comments shall support rich text formatting and attachments

6.2. Reply System:

* Users shall be able to comment on other comments.

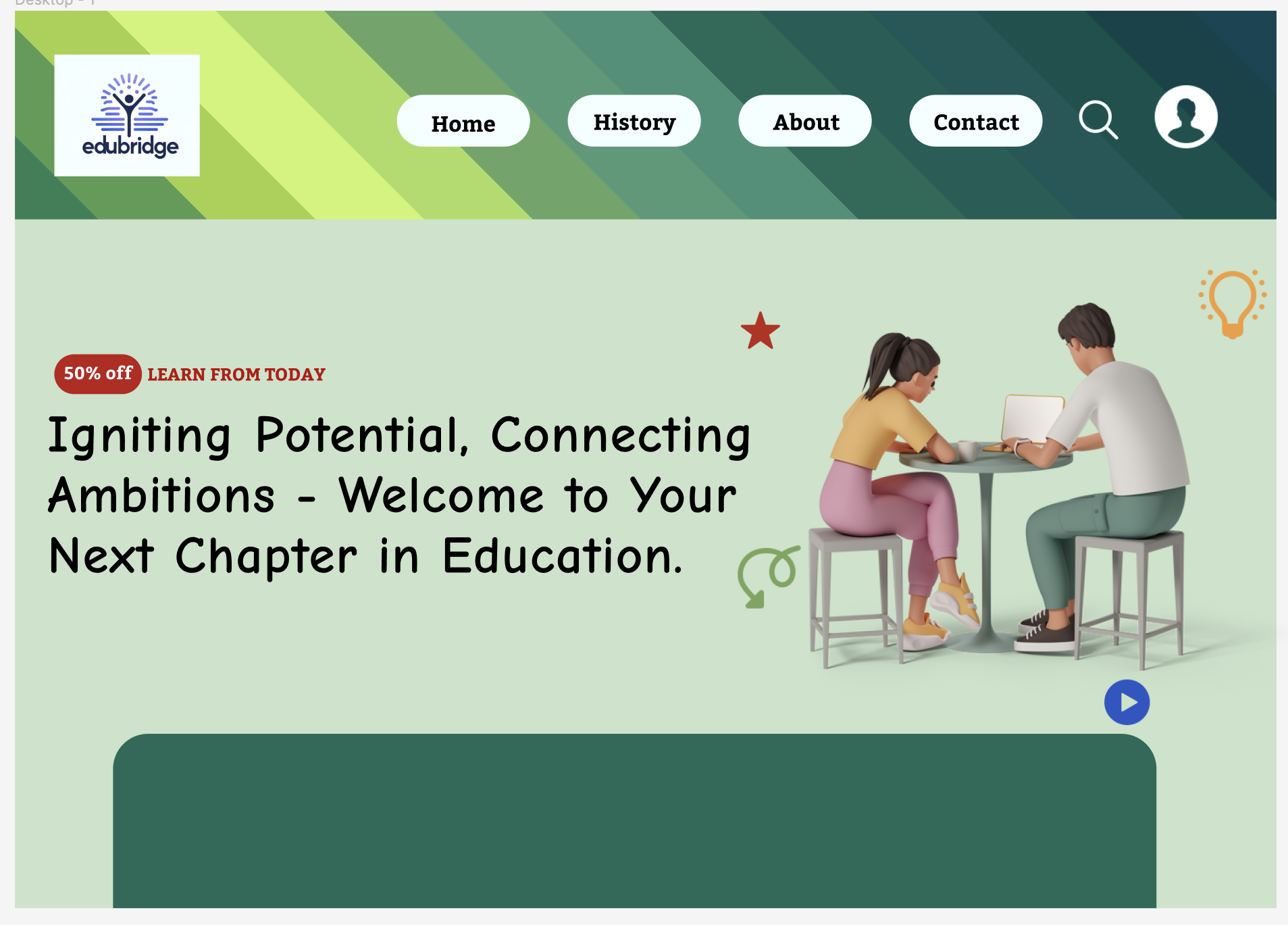
6.3. Like System:

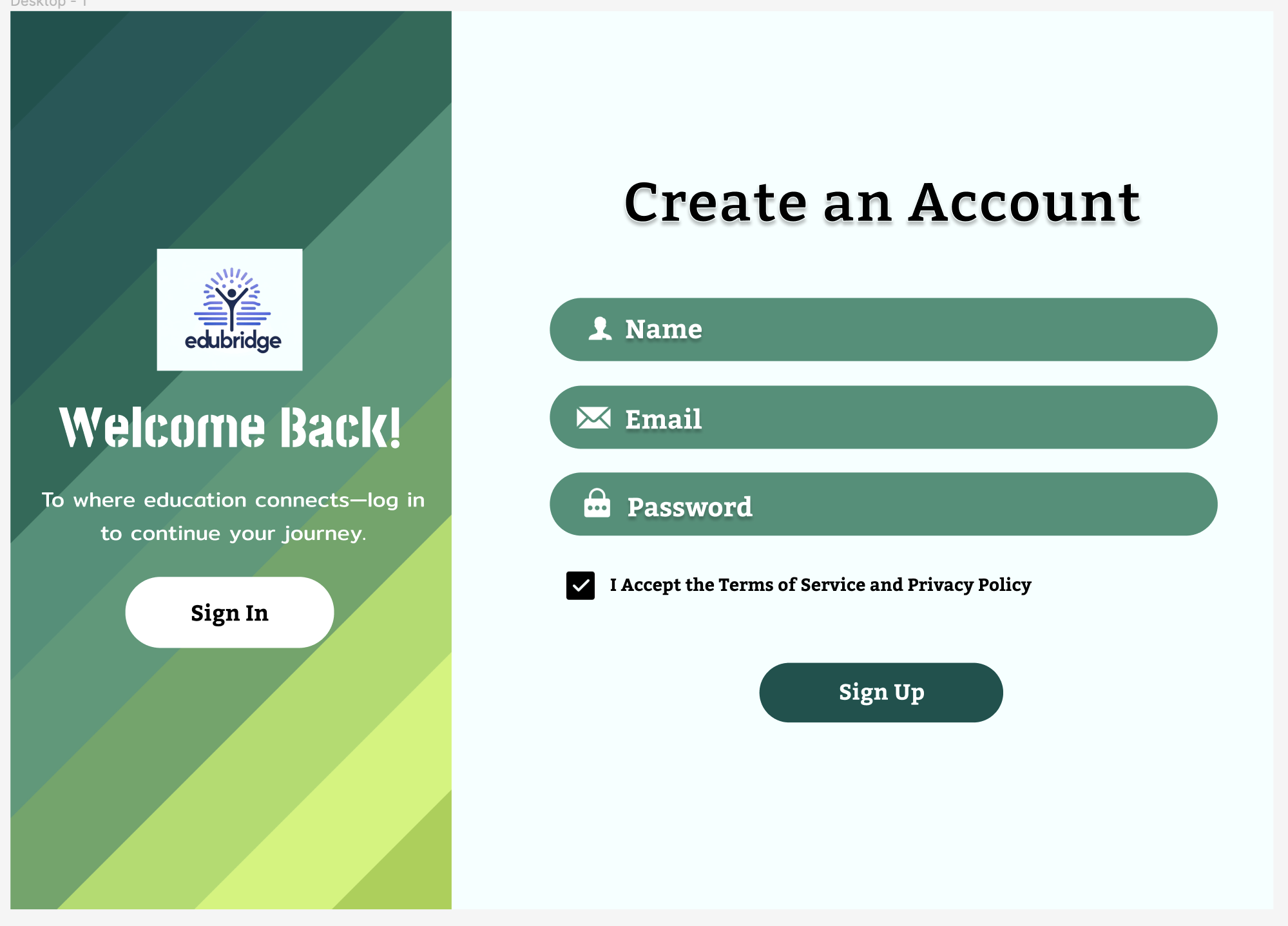
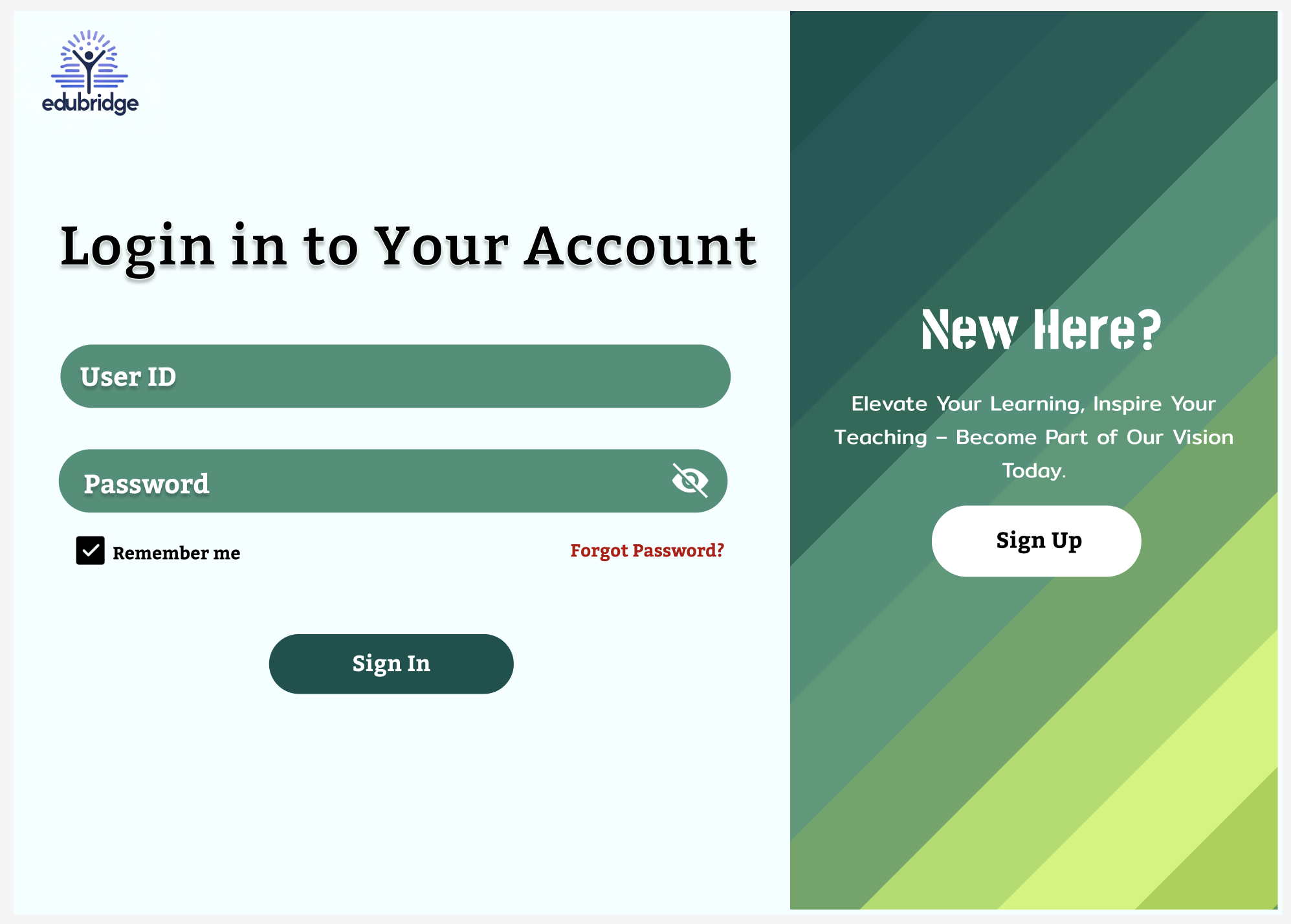
* Users shall be able to like content or comments.
* Users shall be able to dislike content or comments.

# **UI Mockups and UX Flows:**

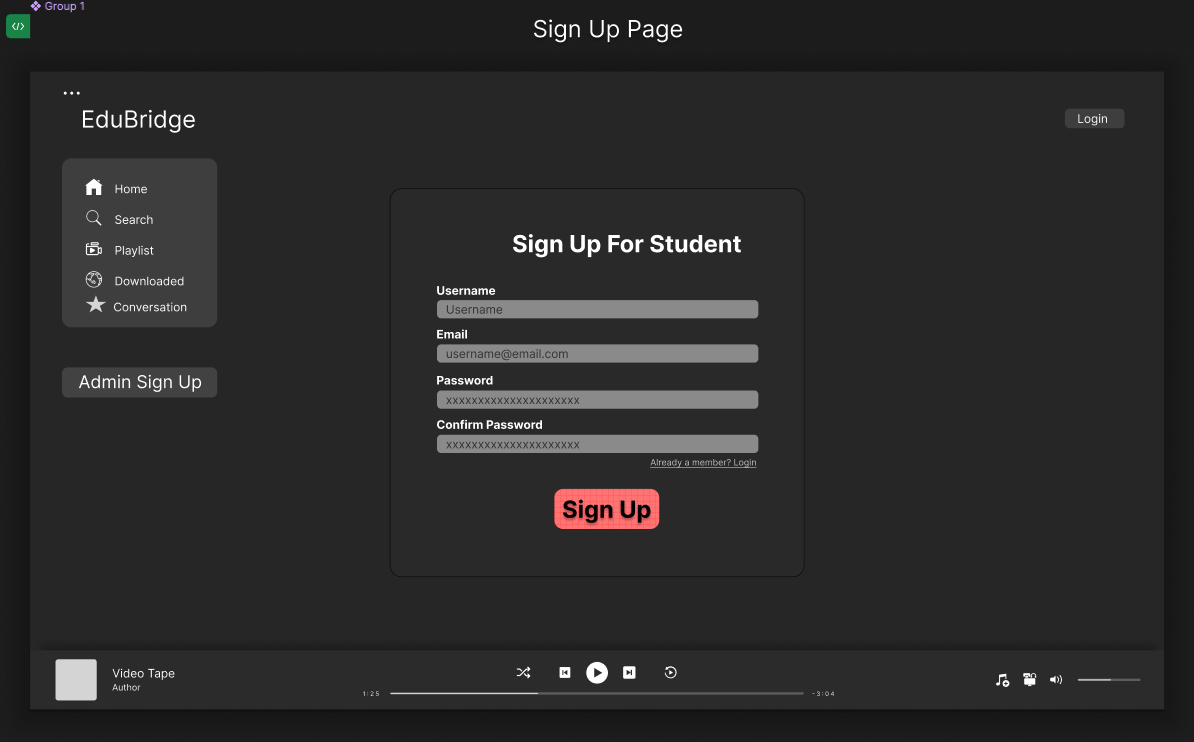
[**Click Here for Interactive Module of UX Flows**](https://www.figma.com/proto/m1EUA4BGHZRvkjO5H0n9Tl/EduBridge?type=design&node-id=50-122&t=Jjk1a8gzOtXaop5x-1&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=50%3A122&mode=design)

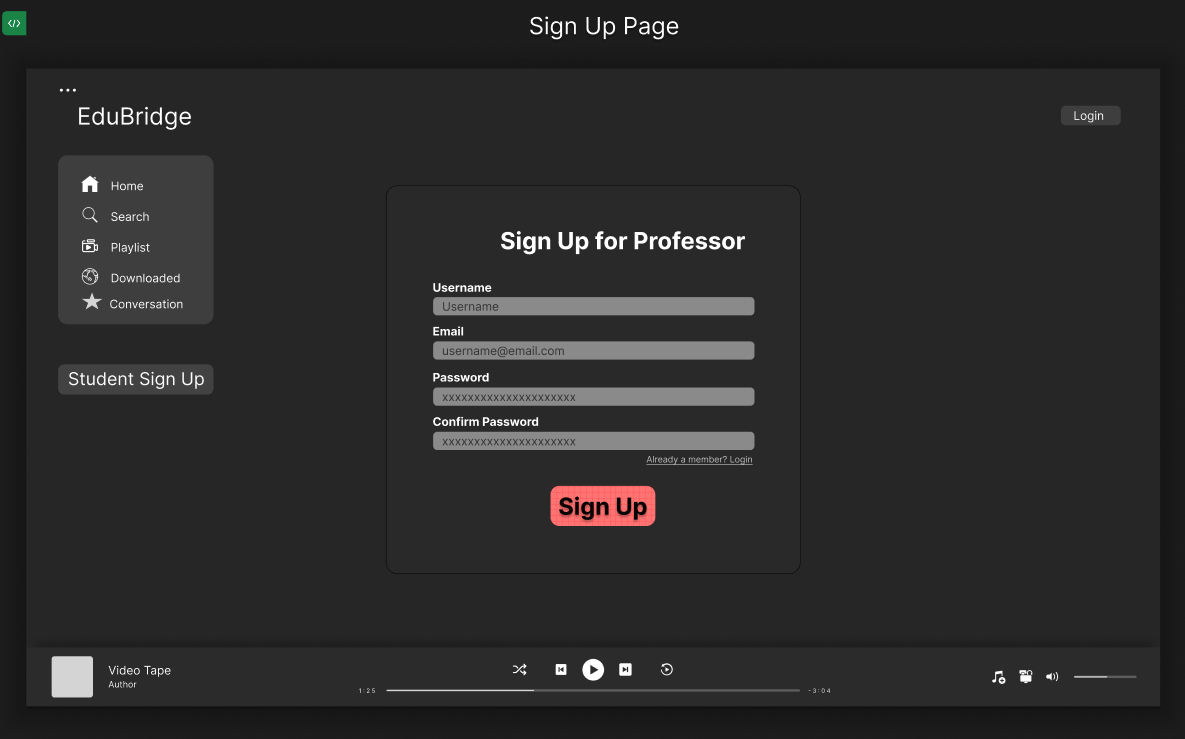
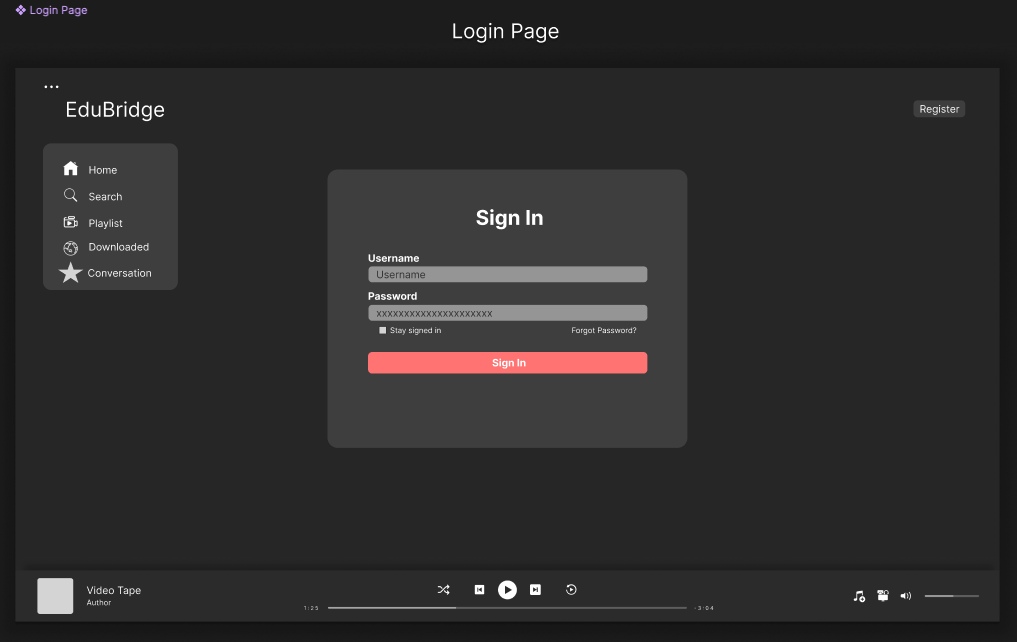
**Basic Expected Output:**

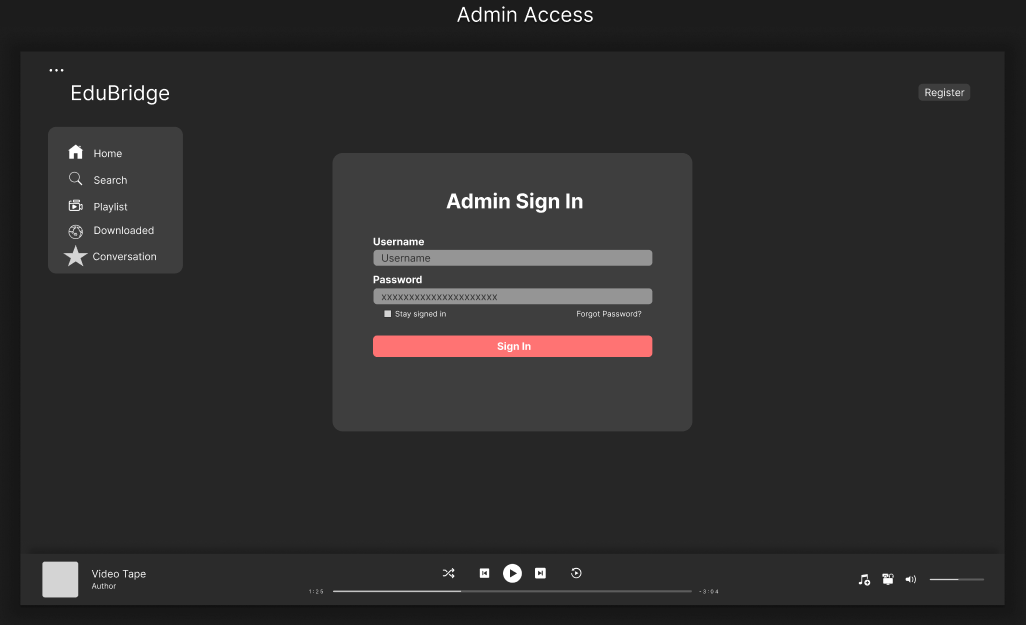
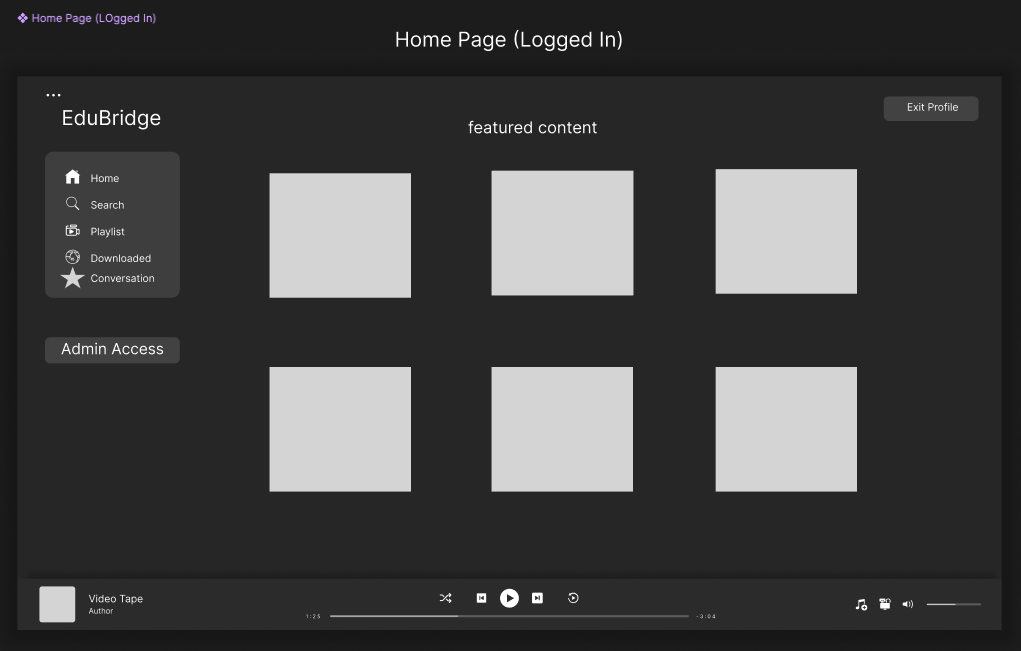


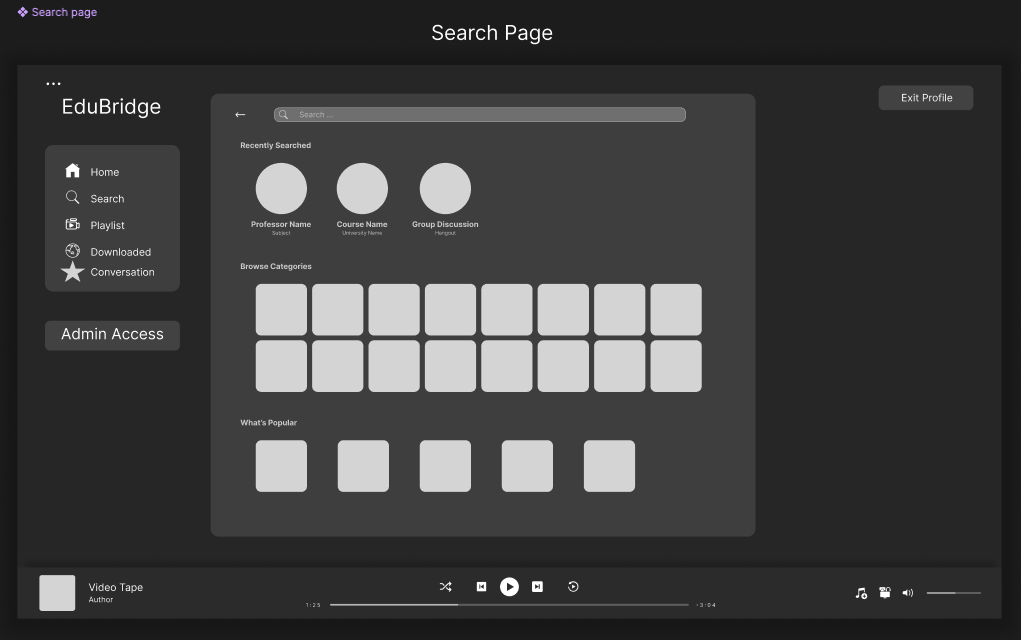
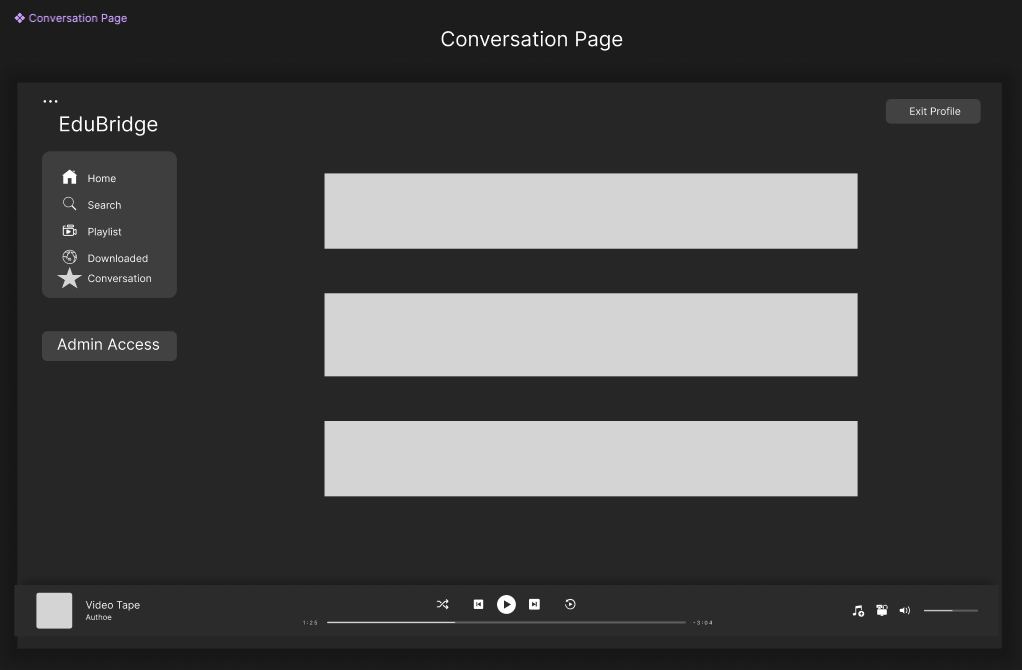


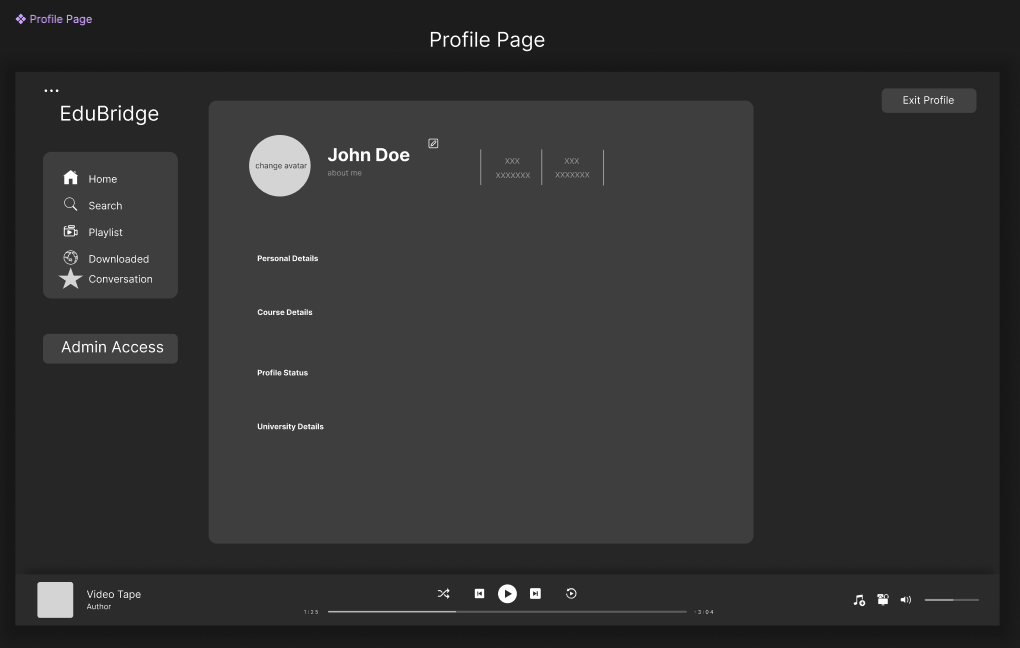
**UI/UX Mockups:**











# **High Level Architecture, Database Organization**

## **MySQL Database Organization:**

USERS // added, deleted,searched, displayed

user\_id (primary key)

name

email

password

is\_Authenticated

ROLES // added, deleted

role\_id (primary key)

role\_name

USER\_ROLES // added, deleted, displayed

user\_id (primary key, foreign key to USERS.user\_id)

role\_id (primary key, foreign key to ROLES.role\_id)

CONTENT // added, deleted,searched, displayed

content\_id

user\_id (foreign key to USERS.user\_id)

title

description

type

upload\_date

location

view\_count

visibility

COMMENTS // added, deleted, displayed

comment\_id (primary key)

content\_id(foreign key to CONTENT.content\_id)

user\_id (foreign key to USERS.user\_id)

text

upload\_date

LIKES // added, deleted, displayed

like\_id (primary key)

content\_id(foreign key to CONTENT.content\_id)

user\_id (foreign key to USERS.user\_id)

date

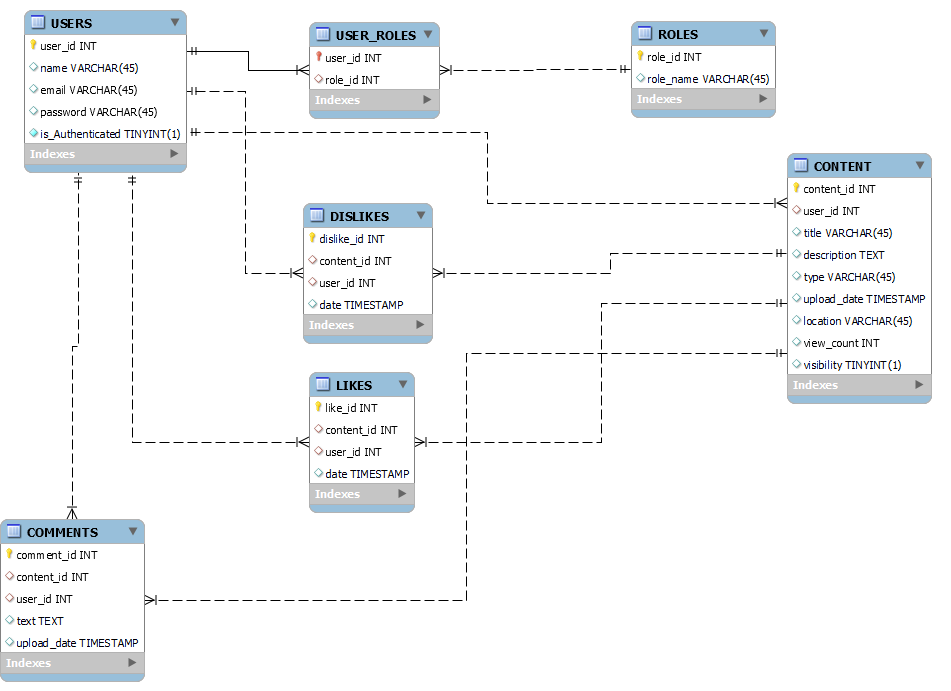
DISLIKES // added, deleted, displayed

dislike\_id (primary key)

content\_id(foreign key to CONTENT.content\_id)

user\_id (foreign key to USERS.user\_id)

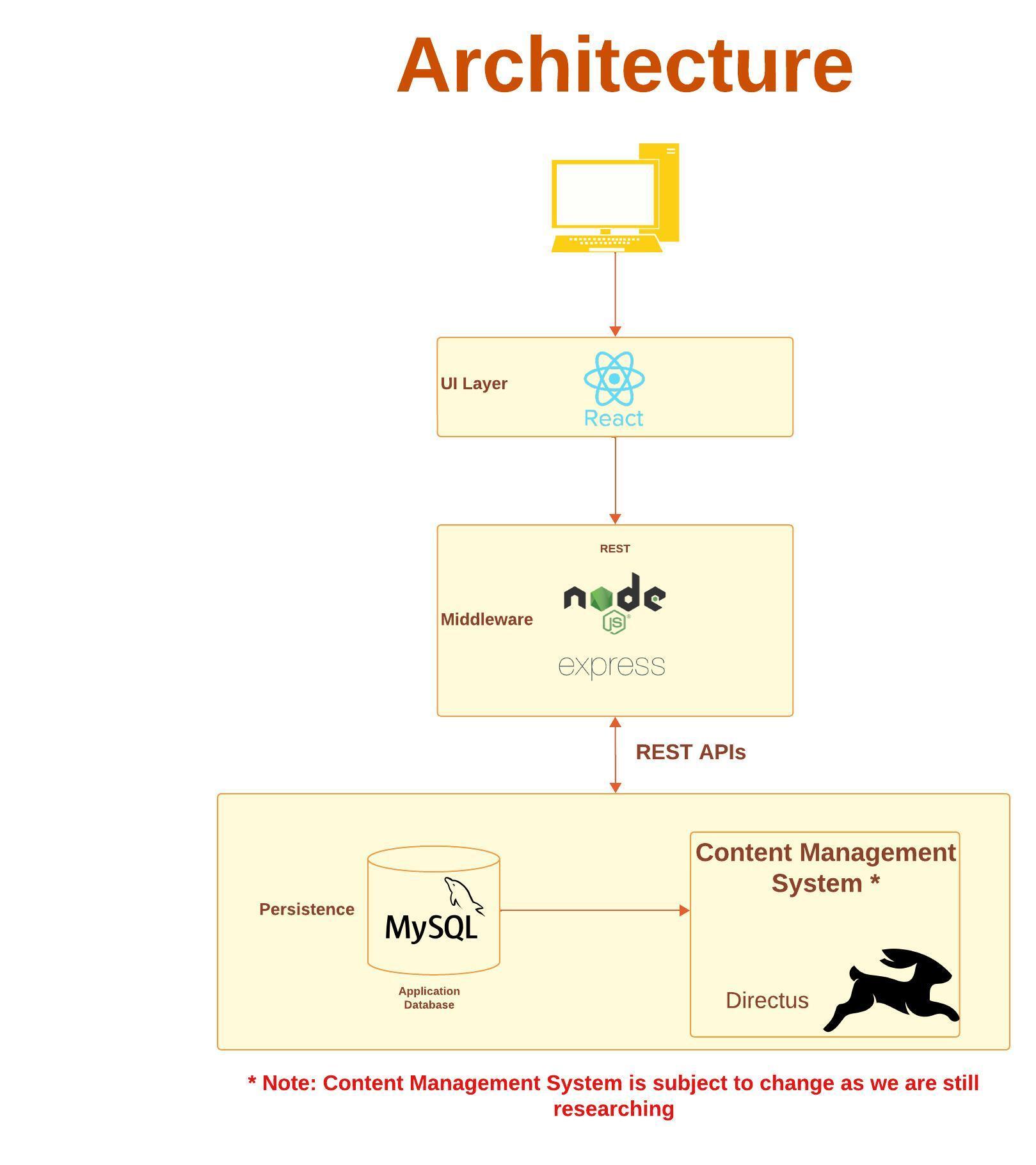
date



## **Current APIs & Architecture:**

### **Express Endpoints:**

| **Endpoint** | **Method** | **Description** | **Input** | **Process** | **Output** |
| --- | --- | --- | --- | --- | --- |
| /signup | POST | Receives email and password from the user. | Email, Password | Checks if the user is already created, checks if the email is .edu, encrypts password and adds user to the database. | Confirmation message |
| /login | POST | Receives email and password from the user. | Email, Password | Checks if a user is in the database. | Returns authorization |
| /users | GET | Returns all users in the database. | None | N/A | List of users |
| /verify-otp | POST | Enters email, password, and OTP to validate the user’s authenticity. | Email, Password, OTP | Checks the OTP validity for the given user. | Verification status |

****

# **Identify actual key risks for your project at this time**

## **Skills Risks**

### Risk

* We are currently looking into CMS to handle our content. It is possible we may need help finding a CMS that will work and is affordable.
* Need to authenticate the API calls.

### Mitigation

* Research more on various open-source CMS.
* Add basic auth to the API calls.

## **Schedule / Teamwork Risks**

### Risk

Arranging meetings is difficult, individually everyone is very busy especially now that we are later in our education where the class intensity may rise more.

### Mitigation

We have meetings as often as we can and often meet in smaller teams for specific tasks, i.e. frontend team may have to meet each other to discuss specific ui elements, or those working on the database need to meet with each other.

Everyone is very active in discord where we keep each other up to date, and coordinate. Additionally we get to utilize online video calls to help mitigate the time cost from commuting.

## **Legal/Content Risks**

### Risk

As with any content hosting platform, we come to the problem of user-uploaded content. Users have the potential to upload any kind of illegal, immoral, or unkind content.

### Mitigation

We will have user uploads verified through teacher/administrator roles.

# **Project Management**

In our scrum meetings, which were held twice a week, each member started with their given tasks and proceeded to carry them out. Additionally, updates about their assigned tasks were updated through Discord which allowed everyone to stay aware of the current progress.

* Naisarg focused on the comparison of functionalities offered by Alfresco, drupal, directus, and Nuxeo CMS while setting them up locally to see which one best fits the use case.
* Shail crafted the backend functionalities for the users (login and sign-up).
* Pankuri and Riken collaborated on the UX as well as the project representation of UX flow through Figma.
* James and Dylan focused on the database functionalities and schema design.