

# BoilerHouse



CS 307 – Design Document

Team 10

Adhi Babu, Arin Asawa, Rohit Kannan, Victor Gao

# Index

<b>Purpose:</b> .....	<b>3</b>
<b>Design Requirements</b> .....	<b>4</b>
Functional Requirements.....	4
User accounts.....	4
User Profile.....	4
Clubs.....	4
Additional Stories.....	6
Non-Functional Requirements.....	6
<b>Design Issues</b> .....	<b>7</b>
<b>Functional Issues</b> .....	<b>7</b>
<b>Non-Functional Issues:</b> .....	<b>9</b>
<b>Design Outline:</b> .....	<b>10</b>
High Level Overview:.....	10
Detailed Overview:.....	10
React Client.....	11
Django Server.....	11
AWS RDS Database.....	11
<b>Design Details</b> .....	<b>11</b>
<b>Classes:</b> .....	<b>11</b>
1. User.....	11
2. Profile.....	12
3. Club Creation Application.....	12
4. Club.....	12
5. Meeting.....	12
6. Review.....	12
7. Club Application.....	13
<b>Class Diagrams</b> .....	<b>14</b>
<b>Sequence Diagrams</b> .....	<b>15</b>
Sequence Diagram for Creating/Updating/Logging into Accounts.....	15
Sequence Diagram for Creating Clubs.....	17
Sequence Diagram for Creating/Updating Meetings.....	19
Sequence Diagram for Creating/Updating Review.....	21
Navigation Flow Map.....	23
<b>UI Mockup</b> .....	<b>24</b>
Home Page:.....	24
Login Page:.....	25
Sign Up Page:.....	26
Page when viewing a Specific Club:.....	27

## **Purpose:**

University students both come to college or want to seek out clubs or extracurriculars that they can participate in. These clubs can actually be a community that these students may be comfortable with and may make lifelong friendships in. BoilerHouse aims to provide a place where students can find organizations that are personalized to them and provides several features for matching interests and creating connections with others.

Currently, to find clubs and organizations here at Purdue, there is a website, namely BoilerLink. Though Boilerlink serves its purpose by listing clubs and meeting times, there aren't any features that allow students to filter through or find clubs that play on each student's interests. Even though Boilerlink displays when a club meets throughout a week, there isn't a filter for it. A student has to scroll through all club listings to see what is available that fits their personal schedule. BoilerHouse matches students to different clubs based on interests that the student places on their own profile, as well as allowing several filters across all aspects, making the app much more intuitive and personalizable.

Another feature that Boilerhouse allows is the ability to communicate and connect with peers who have mutual interests and share clubs. This chat feature can allow newcomers to ask questions to the admins about the club as well as enabling students to create new relationships and build up their own community.

## Design Requirements

### Functional Requirements

#### User accounts

1. As a user, I want to be able to register as a student account
2. As a user, I want to be able to register as an administrative account
3. As a user, I want to be able to log into and manage my account
4. As a user, I want to be able to reset my password if I forgot it

#### User Profile

5. As a user, I want to be able to add information to a profile page consisting of a name, graduation year, bio, major, and various other personal information
6. As a user, I want to be able to change/delete information to a profile page consisting of a name, graduation year, bio, major, and various other personal information
7. As a user, I want to be able to upload a profile picture
8. As a user, I want to be able to change and delete a profile picture
9. As a user, I want to be able to input the times of the week in which I am free
10. As a user, I want to be able to set my interests
11. As a user, I want to be able to update, edit, or delete my interests
12. As a user with an administrative account, I want to be able to manage and delete other accounts, including setting a user as a club officer
13. As a user with club officer permissions, I want to be able to set roles for other club officers
14. As a user with club officer permissions, I want to be able to set meeting times whenever possible for my club and give required permissions for meetings if needed.

#### Clubs

15. As a user, I want to be able to see a complete list of available clubs
16. As a user, I want to be able to see club overviews that display information about its members (what majors members are in, member interests)
17. As a user, I want to be able to get recommendations for clubs that match my interests
18. As a user, I want to be able to apply to create a club
19. As a user, I want to be able to join/apply to a club
20. As a user, I want to be able to search through clubs with filtering
21. As a user with an administrative account, I want to be able to view club applications
22. As a user with an administrative account, I want to be able to accept/delete club applications
23. As a user, I want to be able to filter clubs by what major they are catered to
24. As a user, I want to filter by club dues
25. As a user, I want to filter by relative club size

26. As a user, I want to filter by the culture of clubs
27. As a user, I want to filter by when I am free
28. As a user, I want to filter by the amount of time I am willing to commit every week
29. As a user, I want to be able to apply to receive an officer role in a club
30. As a user who has an officer role in a club, I want to be able to send out notifications to club members through email
31. As a user who has an officer role in a club, I want to be able to set the culture, time commitment, tags, targeted audience, and other detailed information
32. As a user who has an officer role in a club, I want to be able to edit/delete the culture, time commitment, tags, targeted audience, and other detailed information
33. As a user who has an officer role in a club, I want to be able to create meetings on a schedule
34. As a user who has an officer role in a club, I want to be able to update meetings on schedules
35. As a user who has an officer role in a club, I want to be able to delete meetings on schedules
36. As a user, I want to be able to have a calendar view showing my upcoming meetings
37. As a user, I want to be able to freely add new meetings into the calendar view
38. As a user, I want to be able to freely edit and delete meetings in the calendar view
39. As a user who has an officer role in a club, I want to be able to set default contacts for clubs
40. As a user who has an officer role in a club, I want to be able to keep track of what members have paid dues
41. As a user who has an officer role, I want to be able to upload images, files, and other media to my club description page
42. As a user regardless of an officer role, I want to be able to message other members and admins of clubs through the platform
43. As a user, I want to be able to write a rating for any club I am a part of
44. As a user, I want to be able to read other people's ratings of different clubs
45. As a user, I want to be able to be able to see other students who have similar interests as me
46. As a user, I want to be able to look at clubs that other similar interest students are a part of
47. As a user, I want to subscribe to notifications from a club without joining yet
48. As a user, I want to receive notifications for important and upcoming meetings
49. As a user, I want to receive notifications for due dates (i.e club dues)
50. As a user, I want to receive notifications for new messages
51. As a user, I want to receive notifications for joining a club

## Additional Stories

52. If time allows, as a user with admin status, I want to be able to restrict access to or kick students out of my club
53. If time allows, as a user, I want to be able to register/login with my Google account
54. If time allows, as a user, I want to be able to register/login using my Purdue credentials
55. If time allows, as a user who has an officer role in a club, I want to be able to record attendance, if needed, for that club
56. If time allows, as a user, I want to be able to scan a QR code that will check me into a club meeting
57. If time allows, as a user, I want to be able to receive notifications from new clubs that match my interests
58. If time allows, as a user who has an officer role in a club, I want to be able to send out notifications to club members through SMS
59. If time allows, as a user, I want to be able to block other users so they can't message me
60. If time allows, as a user, I want to be able to receive a similarity rating between me and clubs
61. If time allows, as a professor or faculty member, I want to be able to browse a list of clubs, and be able to sponsor those I am interested in
62. If time allows, as a user, I would like to be able to have a privacy setting which prevents others from seeing my major and interests
63. If time allows, as a user, I want to receive notifications for new connections with other peers

## Non-Functional Requirements

- Security: With its own login system, BoilerHouse will need to store credentials for users in a secure database, and we need to be able to get a response from the database fairly quickly (within 300ms).
- Usability: Boiler House needs to be user friendly for all users, both club admins and those looking to join. The color scheme will fit Purdue and the UI will be implemented so everyone can have access to all the features of the application with ease. We will be using React to implement our front-end.
- Scalability: The application will be designed to allow Purdue students and possibly faculty to use, and be able to support up 1,000 concurrent users or requests. Our databases also need to be structured in a way such that they can store the information

for up to thousands of users, and still be able to return queries in a reasonable time (within 500ms)

- **Accessibility:** If time allows, we want to ensure that our UI design accommodates those that are colorblind or those with impaired vision. In addition, if time allows, we want to make our application accessible for different language barriers.

## Design Issues

### Functional Issues

- 1) Do users need to create an account/log in to use BoilerHouse to browse clubs?
  - Option 1: Only require users to log in to be able to use the app's full functionality other than simply browsing clubs
  - Option 2: Users don't need to log in, they can browse clubs as guest

**Choice:** Both options

**Justification:** We realize that some users may just want to utilize BoilerHouse to browse a list of clubs using our comprehensive filters. We want to be able to allow users to freely browse a list of available clubs without having to register an account and input their information. We also want users to have an option to filter clubs based on their interests, and having an account system that saves user preferences will keep everything more organized and streamlined.

- 2) Should users be able to log in using external accounts?
  - Option 1: Users can log in through Google or Apple accounts
  - Option 2: Users can only log in through our login system

**Choice:** Option 2

**Justification:**

We chose option 2 because it would require considerably more work to integrate Google and Apple login APIs into our website. Having our own login system will be easier and won't take away from the usability of the website.

- 3) Who is able to create clubs, and how do we verify that those clubs are legitimate?
  - Option 1: Anyone can create clubs

- Option 2: Only admins can create clubs, users need to apply to become admins
- Option 3: Users can create clubs, but they must be approved by admins before they get added to the database. Once approved, that user is granted the role of an officer for that club, and they can assign the officer role to any user(s) they wish.

**Choice:** Option 3

**Justification:** Although it would make it easier on the users if anyone can just create clubs, we do not believe that it would be a good idea, as there is the potential that some users with malicious intents will make a series of fake club postings and meetings. We also do not want only admins to be able to create clubs. The purpose of admins is to look over and manage the accounts and clubs in the database. Having an approval system in place ensures that all clubs in the database are legitimate.

- 4) What if users leave inappropriate reviews under club pages?
- Option 1: Admin analyzes and deletes the inappropriate reviews after they are reported by club officers
  - Option 2: Club officers can delete inappropriate reviews
  - Option 3: Implement code that checks for inappropriate content in a review and prevents the user from posting such a review
  - Option 4: Do not handle this

**Choice:** Option 1

**Justification:** It is important to handle inappropriate comments to make sure the Experience is nice for all users. Thus, not handling it is not a good approach. Having club officers delete inappropriate comments is not a good option as club officers can misuse this and delete any critical comments. Implementing code that checks for inappropriate content will require checking for hard-coded terms in a user's review/rating. Although this is feasible, it is important to note that comments can be inappropriate without the use of profane language. Thus, this approach will not be 100% effective. The most effective way to get rid of inappropriate reviews without allowing for misuse is to have a human review them.



**Non-Functional Issues:**

1) What framework should we use for our backend?

- Option 1: Django
- Option 2: .NET
- Option 3: Springboot
- Option 4: Node.js

**Choice: Option 1**

**Justification:** We chose Django because all the group members are familiar and comfortable with using Python. The rest of the frameworks require the use of languages other than Python and thus there will be a learning curve for members of the group.

2) What framework should we use for the front end?

- Option 1: React.js
- Option 2: Angular.js
- Option 3: Vue.js
- Option 4: React Native

**Choice: Option 1**

**Justification:** We chose React.js because it provides a nice way of breaking the UI into reusable components in a declarative fashion. We believe this would lead to better code organization. React also has a massive developer community supporting it, meaning we will be able to find frameworks/extensions useful to us with ease.

3) What database should we use to host the data?

- Option 1: Google Firebase
- Option 2: AWS
- Option 3: MongoDB
- Option 4: Microsoft Azure

**Choice: Option 2**

**Justification:** We chose AWS because most of the group members have used their services in personal projects and it has a track record of being very reliable. AWS also offers a free tier which is useful for testing as we develop the application. The free tier comes with more than enough storage capacity and request limits for our testing purposes. In the future, should we wish to scale

the application, AWS makes it easy to upgrade to have more storage and request limits.

4) Should we use SQL or NoSQL?

- Option 1: SQL
- Option 2: NoSQL

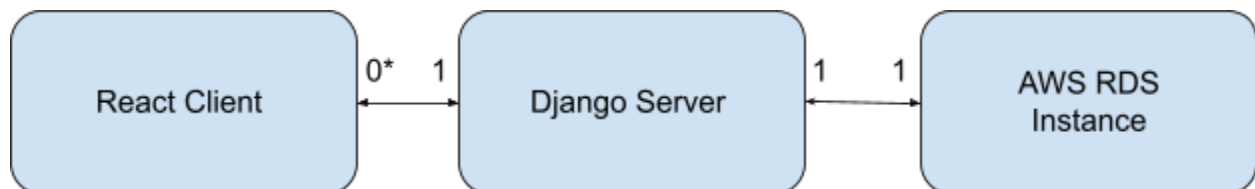
**Choice: Option 1**

**Justification:** We chose SQL because most of the group members have experience with relational databases, making it an obvious choice. We also have planned out a fixed schema, making implementing a SQL database and handling the relationships between tables easier than trying to implement the relationship with a NoSQL database.

## Design Outline:

BoilerHouse will have many clients that must be able to interact with the database, and be able to retrieve, store, and update various information. Our back end service will handle the requests sent from the clients, and pass them onto the database. On the way back, the server will consolidate the data returned by the database, performing any calculations, such as similarity calculations or finding recommended clubs, before serving that data back to the clients. Our client communicates to our server using a request via the Python Django SDK. The server then processes this and sends an SQL query to our database. Once data from the database is returned, the server then serves this data back to the clients.

## High Level Overview:



## Detailed Overview:



## React Client

The React client is the frontend that the user will interact with. It displays our UI and sends information to our backend to be processed. It will use the fetch api to send requests to our Django server and obtain information.

## Django Server

The Django Server is our backend that will handle all of our service layer handling. It will be a server hosted on a machine with various endpoints. When the endpoints are called by a client, a service layer method will be called and the request will be validated, processed, and it will use a connection to the database to add/modify/delete the entries if needed. Then the backend will return a response to the client.

## AWS RDS Database

The AWS RDS Database is a SQL relational database hosted by Amazon Web Services. We will use this database to persist our object entries. We will use a key to access this database through the service layer.

## Design Details

### Classes:

#### 1. User

- a. User object is created when a user creates an account in the login page
- b. User object is created with a username and password
- c. When a user object is created, a profile one is created as well and they are linked
- d. After the object is created the password can be edited from the profile page
- e. Users with administrative accounts can set administrative access to other users
- f. Users with administrative accounts can view all profiles on a profile list page, where they can edit/delete users
- g. Users with officer roles can appoint others to have officer roles

## **2. Profile**

- a. Profile object is created when a user creates an account on the login page
- b. Profile object is created with a name, bio, profile picture, interests, year, and major.
- c. Schedule and other information can be edited from the profile page on a later date.
- d. Profiles are default set to public but can be privatized

## **3. Club Creation Application**

- a. Club creation applications are created by users on a create club page
- b. Club creation applications are created with a club name, officers, club description, meeting times, one-line description, and estimated dues
- c. Club creation applications can be viewed, denied, or approved by administrative accounts
- d. Club creation applications can be withdrawn, and will automatically expire in 60 days

## **4. Club**

- a. Clubs are created when club creation applications are approved
- b. Clubs are created with officers, a name, a description, estimated dues, meeting times, one-line description
- c. Club information can be edited from a club home page by the club's officers
- d. Clubs can be viewed from the club list page
- e. Announcements can be added to the club by the club's officers

## **5. Meeting**

- a. Users create meetings from a club's home page by the club's officers
- b. Meetings are created with a time, whether it is general or for officers, agenda, clubId, and whether an announcement should be made or not
- c. If an announcement should be made, an email is sent out to the target members
- d. Meetings can be edited from the club's home page by officers

## **6. Review**

- a. Users create reviews from a club's home page
- b. Reviews are created with a score (out of 5), author, timestamp, and a comment

- c. Reviews can be viewed on a club's homepage
- d. Reviews can be edited from the club's home page

## **7. Club Application**

- a. Users with club officer permissions can create club applications from a club's home page
- b. Club application is created with a list of questions
- c. Club application can be edited from a club's home page

## Class Diagrams



## Sequence Diagrams

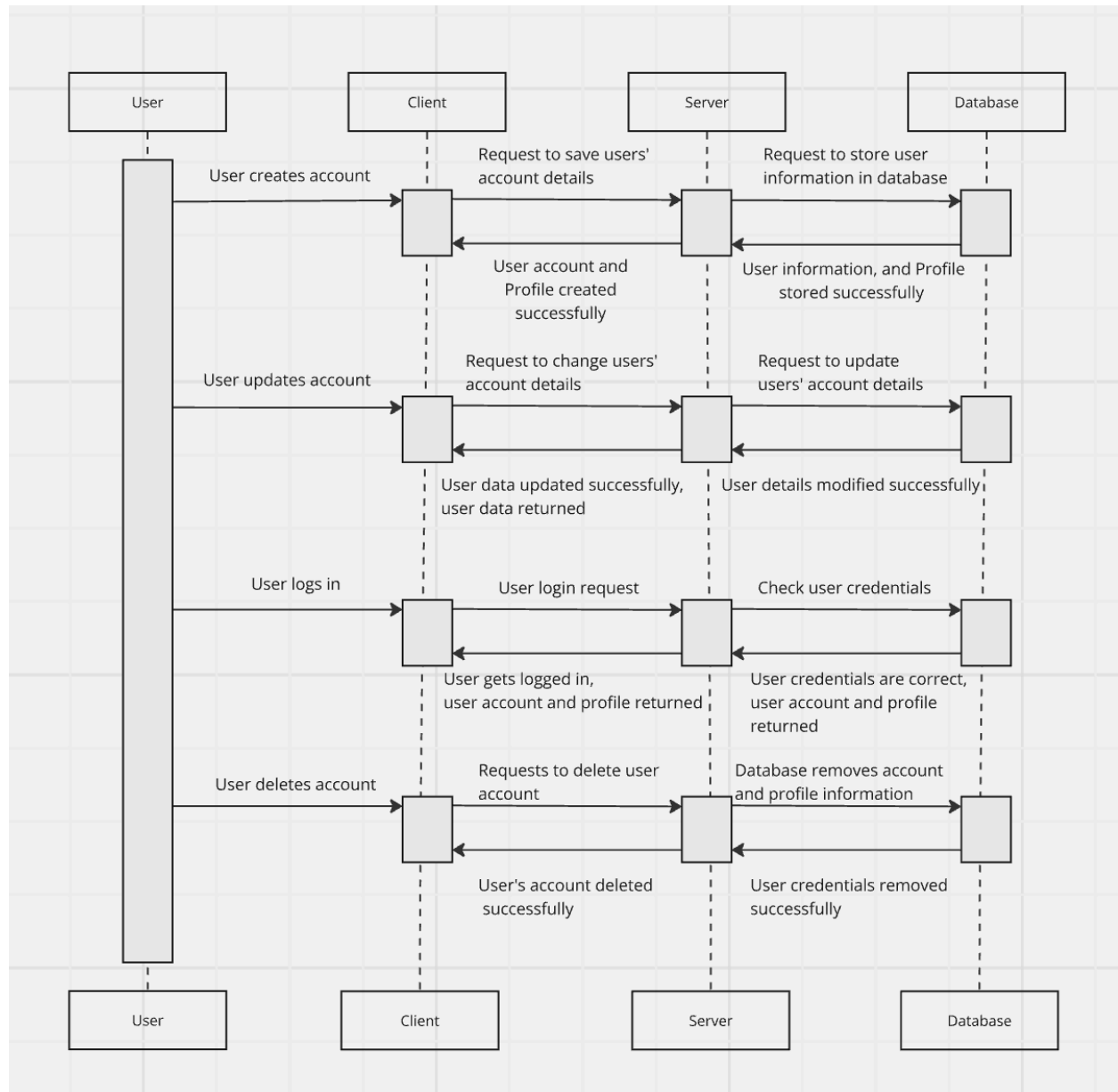
### Sequence Diagram for Creating/Updating/Logging into Accounts

Users must first create an account when they want to log in. To do this, they click the 'Create an Account' button and fill in all necessary information. The client then sends a POST request with all of the user's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200, and the user is logged in.

To update a user's information they must click on the update profile button and change what fields they desire. The client then sends a POST request with all of the user's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200.

To log in, a user must click on the log-in button and input a username and password. The client then sends a GET request with the user's information to the server. The server then does validation checks on the request and checks if these credentials are valid. If this action is successful, a response is sent to the client with status code 200 and a JWT token is passed to the client.

To delete a user's account they must click on the delete profile button, or an admin must click delete account on a user's account. The client then sends a DELETE request with the target's username to the server. The server then does validation checks on the request and attempts to remove the data from the database. If this action is successful, a response is sent to the client with status code 200.



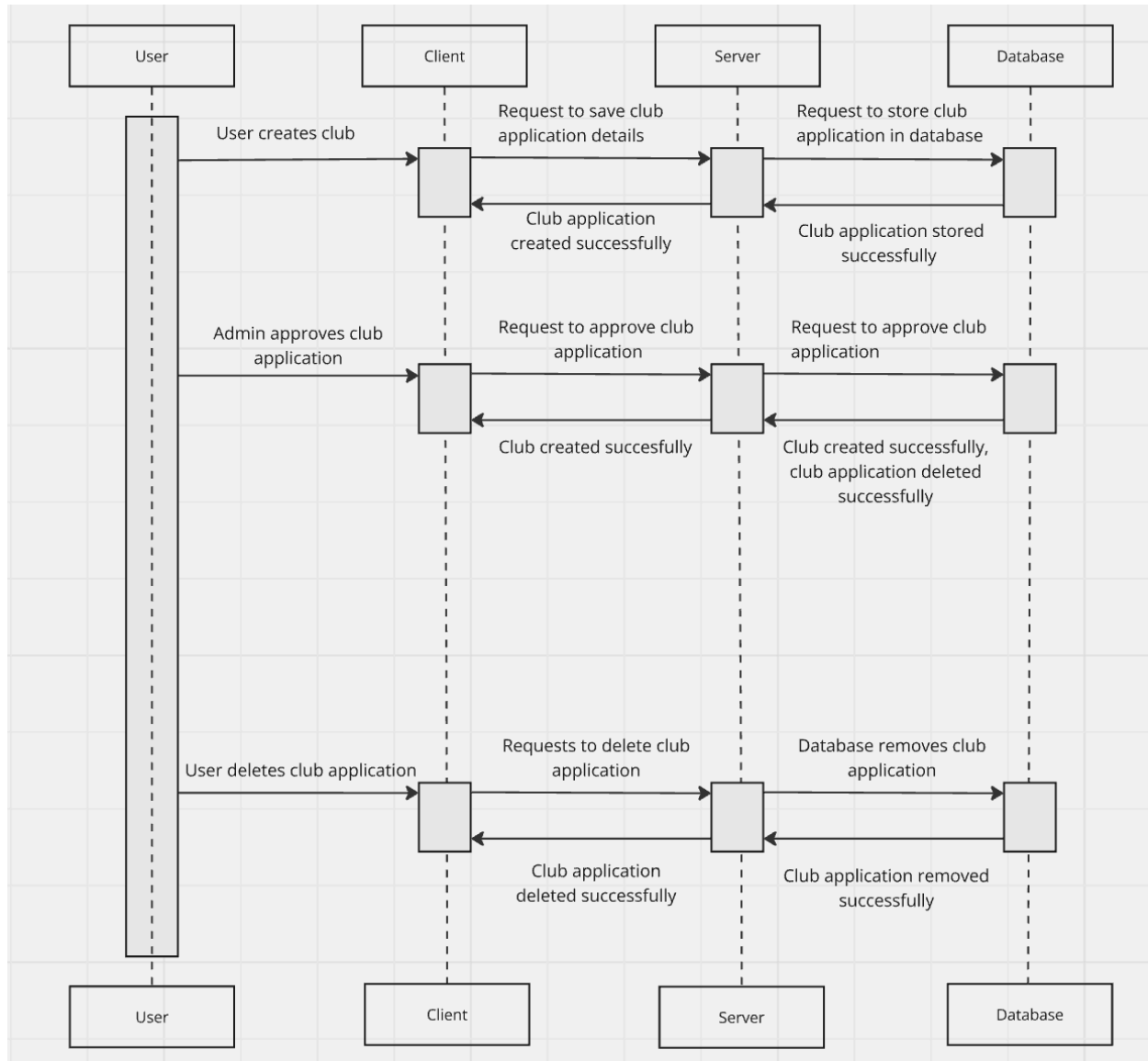


## Sequence Diagram for Creating Clubs

Users must create a club application to create a club. To do this, they click the ‘Create a Club’ button and fill in all necessary information. The client then sends a POST request with all of the club’s information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200, and the user is logged in.

To approve a club’s application, an admin user must click on a club application and press ‘Approve’. The client then sends a POST request with all of the club’s information to the server. The server then does validation checks on the request and attempts to delete the club application and persist the data of the new club in the database. If this action is successful, a response is sent to the client with status code 200.

To delete a club application, a user must click on the delete application button, or an admin must click deny on a club’s application. The client then sends a DELETE request with the club application’s information to the server. The server then does validation checks on the request and attempts to remove the data from the database. If this action is successful, a response is sent to the client with status code 200.



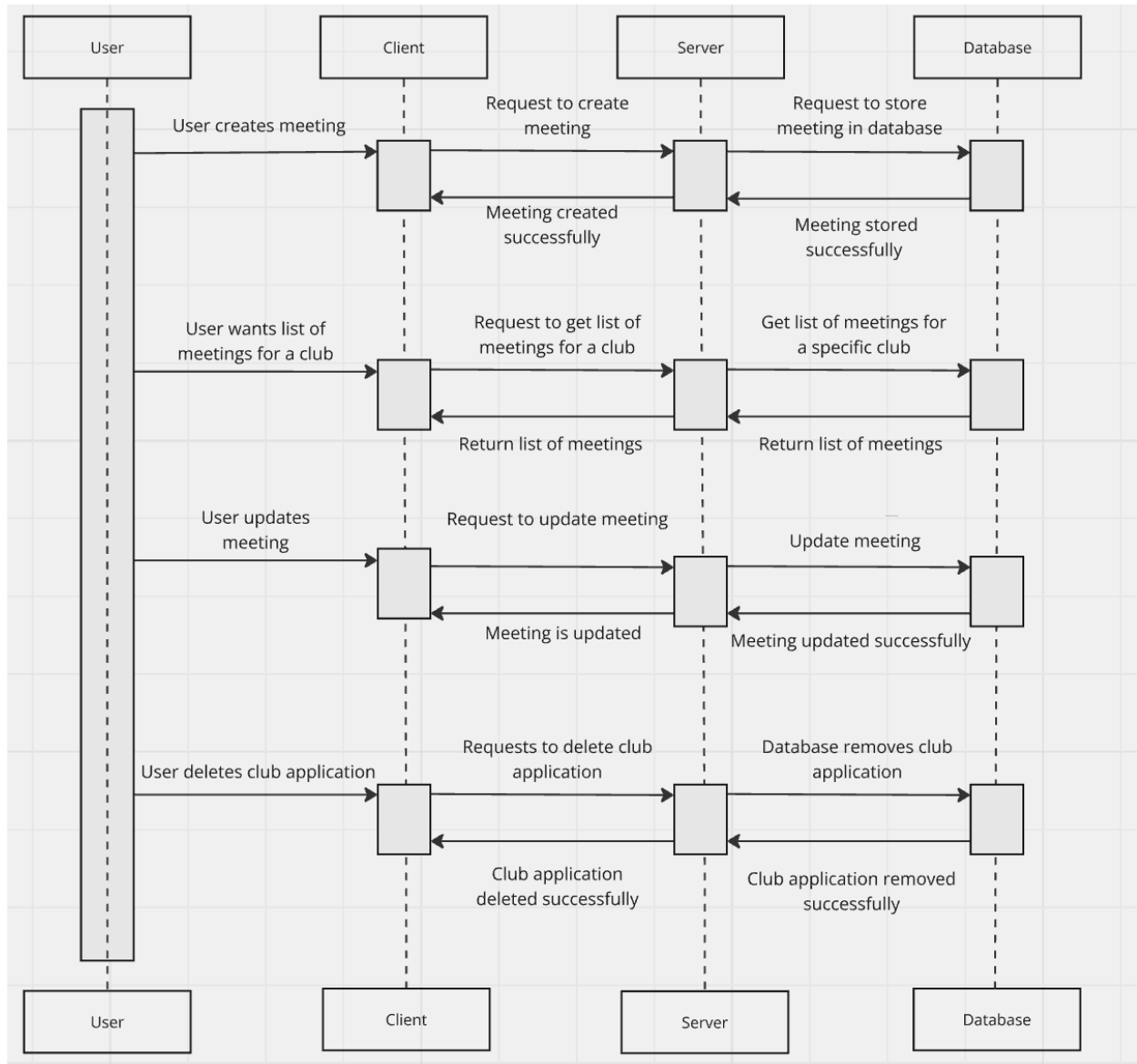
## Sequence Diagram for Creating/Updating Meetings

To create a meeting, a user must navigate to the club that they are an officer for, and click the 'Add an Account' button, and fill in all necessary information. The client then sends a POST request with all of the meeting's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200.

To view a club's meetings, a user must click go to the club's home page. The client then sends a GET request with the club's information to the server. The server then does validation checks on the request and checks if there are meetings for the specified club. If this action is successful, a response is sent to the client with status code 200 and the meetings are passed in the response body.

To update a meeting's information, a user must click on the meeting they want to change and change the fields they desire. The client then sends a POST request with all of the meeting's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200.

To delete a meeting, a user must navigate to a club's home page and then click on the meeting they want to delete. Then they must click on the delete meeting button. The client then sends a DELETE request with the meeting's information to the server. The server then does validation checks on the request and attempts to remove the data from the database. If this action is successful, a response is sent to the client with status code 200.



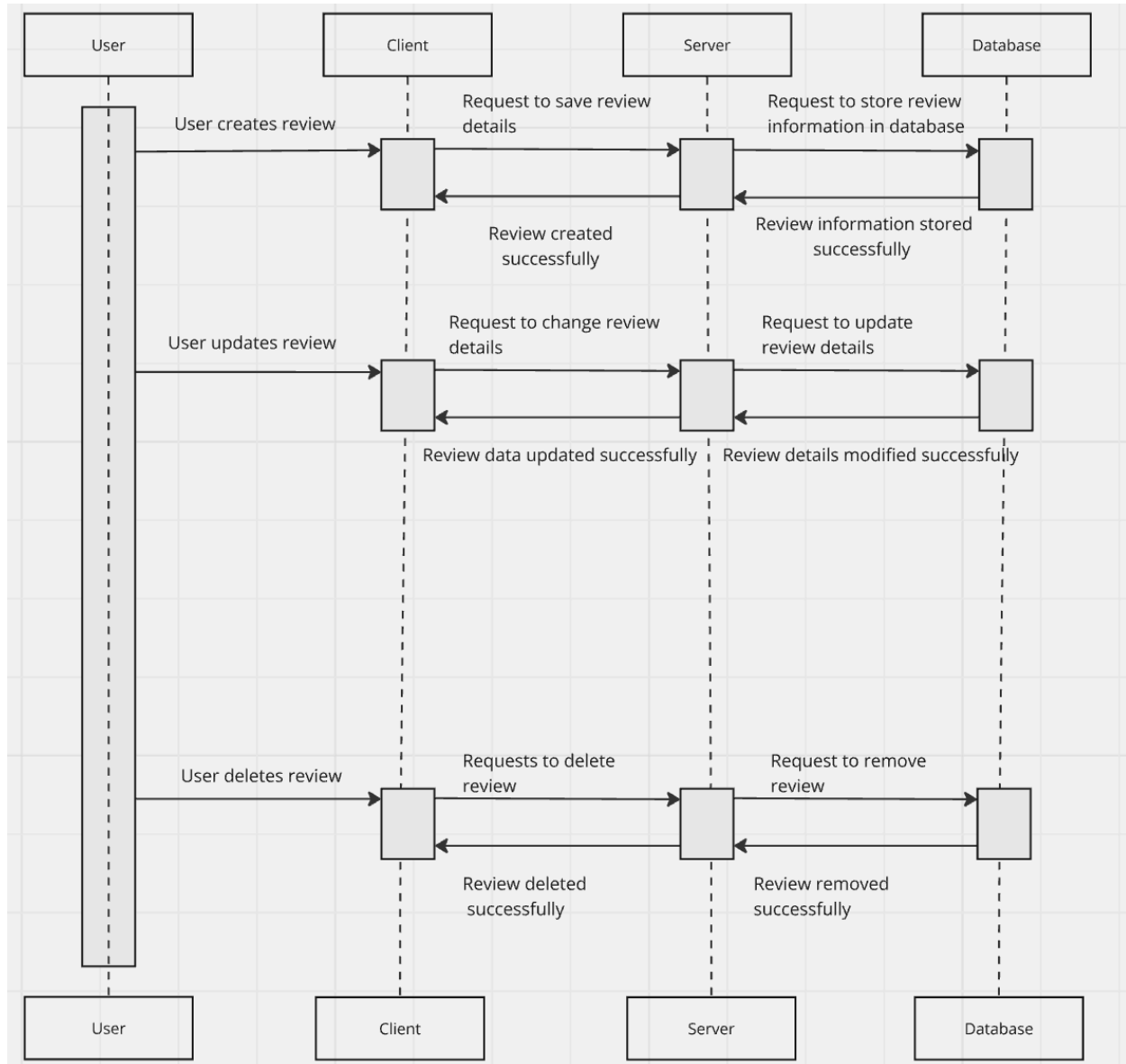
## Sequence Diagram for Creating/Updating Review

To create a review, a user must navigate to the club that they are writing a review for, navigate to the comment section, write a comment, add a rating, and then click the 'Post' button. The client then sends a POST request with all of the review's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200.

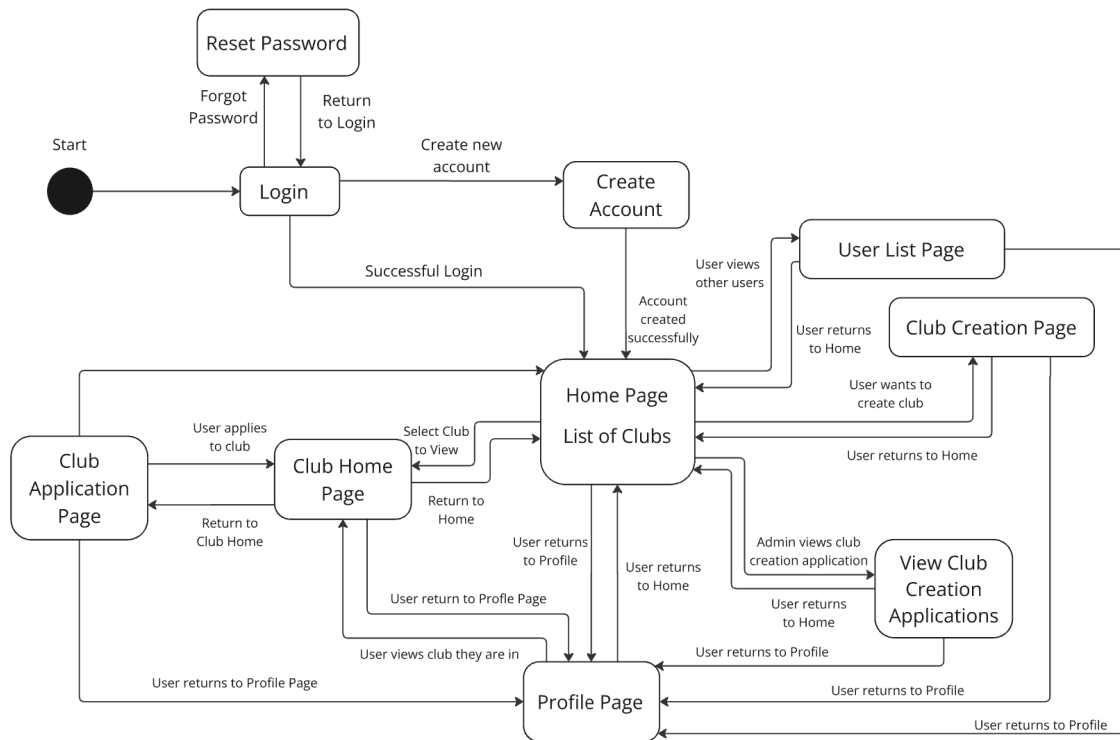
To view a club's reviews, a user must click go to the club's home page. The client then sends a GET request with the club's information to the server. The server then does validation checks on the request and checks if there are reviews for the specified club. If this action is successful, a response is sent to the client with status code 200 and the reviews are passed in the response body.

To update a review's information, a user must click on the review, that they own, that they want to change and change the fields they desire. The client then sends a POST request with all of the review's information to the server. The server then does validation checks on the request and attempts to persist the data in the database. If this action is successful, a response is sent to the client with status code 200.

To delete a review, a user must navigate to a club's home page and then click on the review, that they own, that they want to delete. Then they must click on the delete button. The client then sends a DELETE request with the review's information to the server. The server then does validation checks on the request and attempts to remove the data from the database. If this action is successful, a response is sent to the client with status code 200.

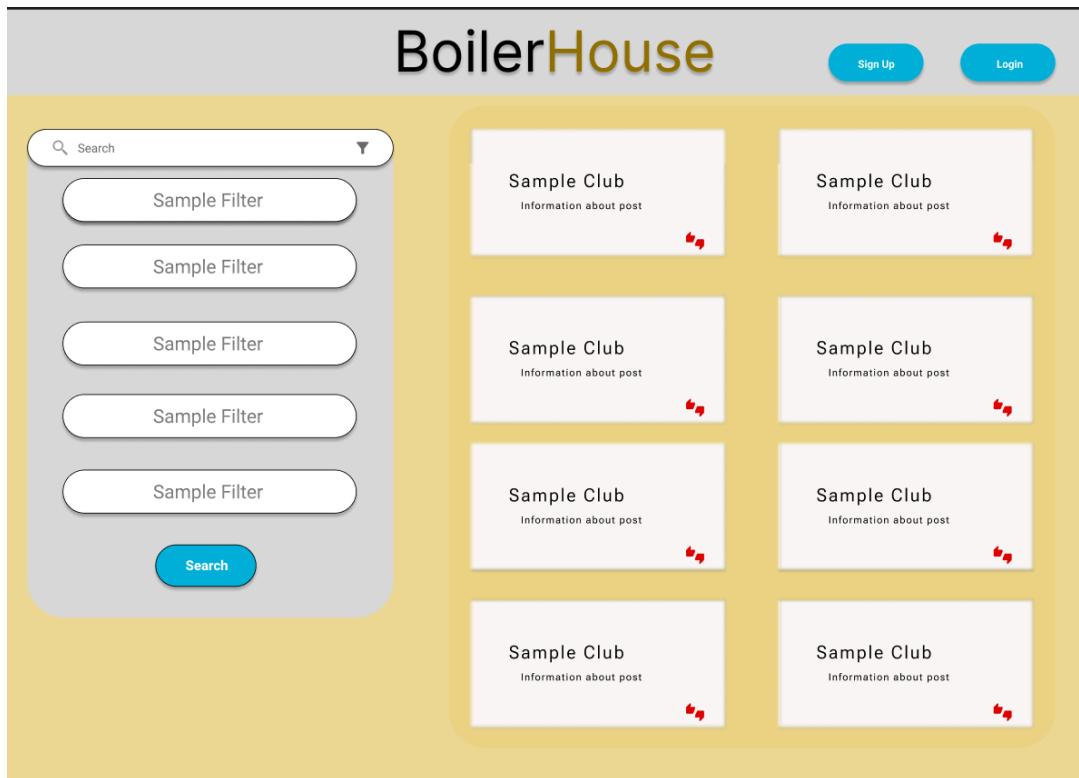


## Navigation Flow Map

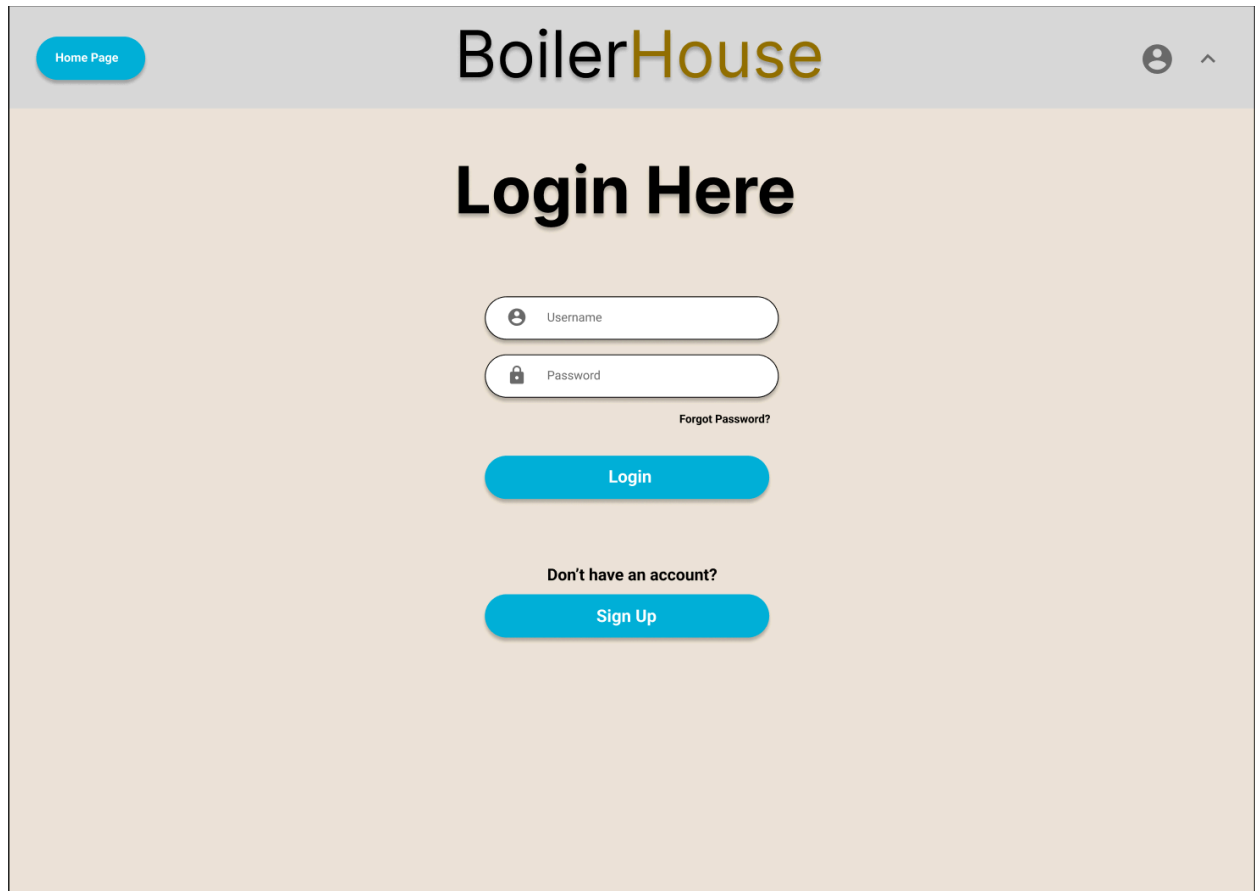


# UI Mockup

## Home Page:





**Login Page:**

The image shows a web page mockup for a login interface. At the top, there is a grey header bar. On the left of the header is a blue button labeled "Home Page". In the center of the header is the "BoilerHouse" logo, with "Boiler" in black and "House" in a gold color. On the right of the header is a user profile icon and an upward-pointing chevron. Below the header, the main content area has a light beige background. Centered in this area is the text "Login Here" in a large, bold, black font. Below this text are two white input fields with rounded corners. The first field is labeled "Username" and has a small person icon on its left. The second field is labeled "Password" and has a small lock icon on its left. To the right of the password field is a link that says "Forgot Password?". Below the input fields is a large blue button labeled "Login". Below the "Login" button is the text "Don't have an account?". At the bottom of the form is another large blue button labeled "Sign Up".

Home Page

BoilerHouse

Login Here

Username

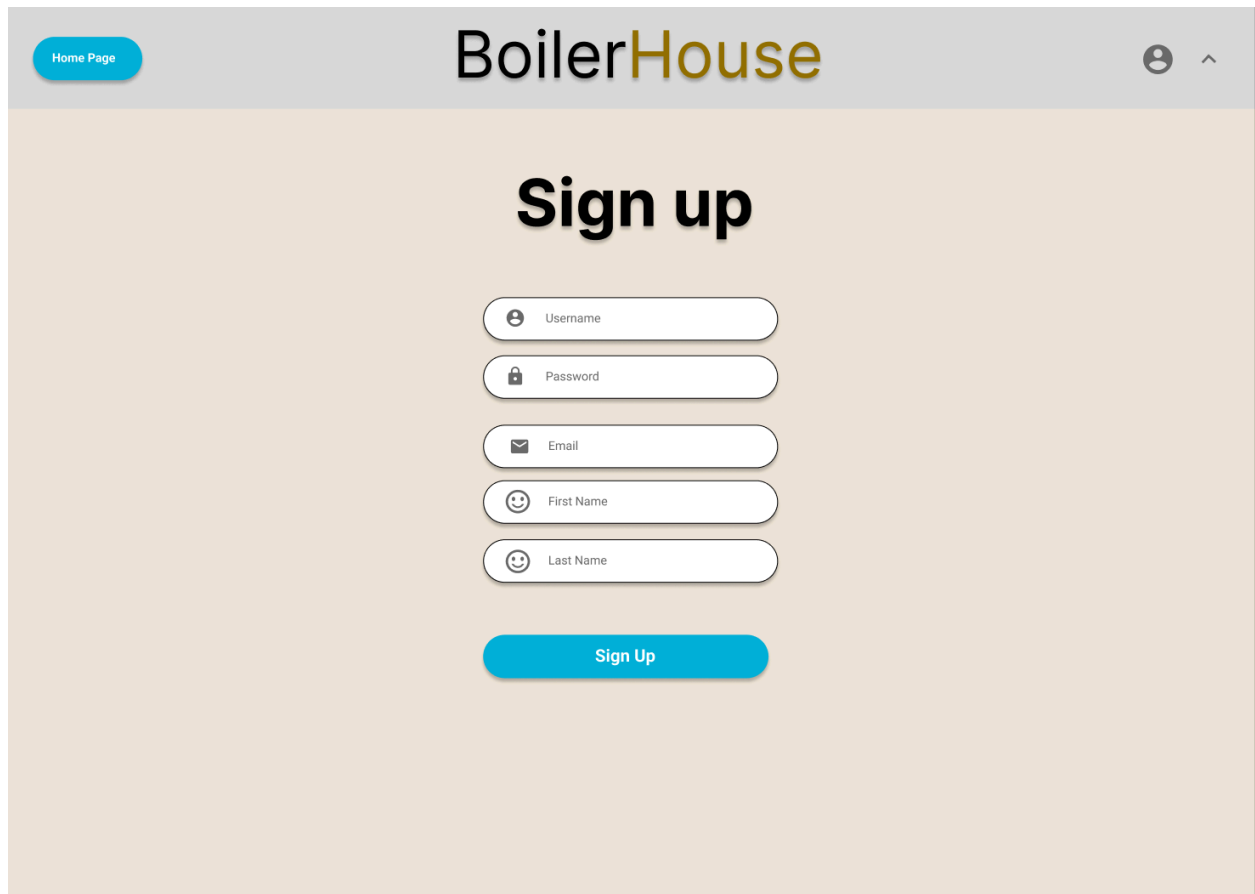
Password

Forgot Password?

Login

Don't have an account?

Sign Up



**Sign Up Page:**

The image is a mockup of a web page for signing up. At the top, there is a grey header bar. On the left of the header is a blue button with the text "Home Page". In the center of the header is the logo "BoilerHouse", where "Boiler" is in black and "House" is in a gold color. On the right of the header is a user profile icon and a small upward-pointing chevron. Below the header, the main content area has a light beige background. In the center of this area, the text "Sign up" is written in a large, bold, black font. Below this text are five white input fields with rounded corners, each containing a small icon on the left and a label on the right: "Username" (person icon), "Password" (lock icon), "Email" (envelope icon), "First Name" (smiley face icon), and "Last Name" (smiley face icon). At the bottom of the form is a blue button with the text "Sign Up".

**Page when viewing a Specific Club:**

[Home Page](#)

# BoilerHouse



## Sample Club

Interest Match Percentage

Apply!

### Ratings and Reviews

Mike Anderson

Former Club Member

★★★★☆

Amanda Conrad

Club Treasurer

★★★★☆

### Club Info:

### Meeting times:

### Club Admin(s):

Amanda Conrad

Club Treasurer

555-555-5555

Brad Davis

Club President

555-555-5555