

Design Document

Project Name: Eventor

Team 25 - Sean Rettig, Vance Menapace, Raul Lara-Concha, William Kettleborough

Purpose

In today's world, platforms fail to provide a meaningful solution for discovering and sharing events, leading to fragmented engagement and interaction. Oftentimes, it is difficult to know where to look in order to have the most up-to-date information on events that you're interested in. Our aim is to differentiate our project by providing what other platforms such as "Facebook Events" and "Purdue Boilerlink" lack: a single centralized place to track all the events you are interested in, and a place for those events to be shared.

The system we are designing includes a feed for users to look at which is served by the web server as a front end to the database. The user is also able to create events for others to attend. The purpose of our system is to allow users to find and view nearby events, as well as connect with other users and their interests.

Requirements (Backlog)

Functional

1. As a user, I would like to create an account.
2. As a user, I would like to sign into an account.
3. As a user, I would like to reset my password if I forgot it.
4. As a user, I would like to be able to delete my account.
5. As a user, I would like to manage my account information including my username and password.
6. As a user, I would like to be able to set the information on my profile, including a profile picture, display name, and a biography or status.
7. As a user, I would like to have a list of events that I have created on my profile.
8. As a user, I would like the ability to post an event onto my profile. Adding an address, description, event type, links, and other additional information.
9. As a user, I would like to allow other users to RSVP or indicate interest for my events and see a list of users who have done so.
10. As a user, I would like to have a calendar on the app that shows all the events I am RSVPed for or interested in going to.
11. As a user, I would like to be able to follow other users.

12. As a user, I would like to be able to have a feed of the most recent events posted by users that I follow.
13. As a user, I would like the ability to like and share events.
14. As a user, I would like the ability to view analytics on events, such as likes, shares, and views.
15. As a user, I would like to be able to search for events based on my location or interests.
16. As a user, I would like to be able to search for profiles of other users.
17. As a user, I would like to be able to filter my search to better find what I am looking for.
18. As a user, I would like to allow other users to comment on my events.
19. As a user, I would like to be able to like and reply to other user's comments.
20. As a user, I would like the ability to block other users causing me to not see any of their posts about events and preventing them from commenting or interacting on my profile.
21. As a user, I would like the ability to unblock users I have blocked.
22. As a user, I would like the ability to delete comments from my posts.
23. As a user, I would like to set other users as moderators with the ability to delete comments from my posts.
24. As a user, I would like the platform to automatically block posts and comments on posts that may contain obscene or offensive content.
25. As a user, I would like the ability to delete my own comments.
26. As a user, I would like the ability to edit posts that I have made.
27. As a user, I would like the ability to delete posts I have made.
28. As a user, I would like to share a link to the event somewhere else, and have an embedded stylized version of the event show up.
29. As a user who is an Artist, I would like the ability to have a custom formatted event type for going on tour, which shows all of my tour locations and dates.
30. As a user who is an Artist, I would like the ability to have a custom formatted event type for album drops/music releases, which allows me to embed a track list within my post and share links to different listening platforms.
31. As a user, I would like the ability to create Twitter-like posts on my profile to get information to fans who follow me or visit my profile.
32. As a user, I would like the ability to attach image(s) to my posts.
33. As a user, I would like the ability to use markdown in my posts for increased control over text formatting.
34. As a user, I would like the ability to co-host an event with another user, making the same event show up as posted by both of us.
35. As a user, I would like to post updates to a specific event.
36. As a user, I would like the ability to view my notifications in a dedicated UI location (e.g. a "notification bell" dropdown that shows my notifications in order of recency).
37. As a user, I would like the ability to opt-in to notifications from users whose posts I am interested in.
38. As a user, I would like the ability to configure which types of posts I get notifications from.
39. [If Time Allows]: As a user, I would like to see what the weather will be at the location of an in-person event.

40. [If Time Allows]: As a user, I would like the ability to share a map for my in-person events so attendees can find their way around easier.
41. [If Time Allows]: As a user, I would like the ability to add an event to my external calendar app.
42. [If Time Allows]: As a user, I would like the ability to link my calendar from this app to my external calendar app (automatically updating).
43. [If Time Allows]: As a user, I would like an interactive user tutorial for how to use the app.

Non-Functional

Security and Authentication

- Users will be able to log into their own accounts or sign up for a new account. Passwords will be stored securely in a SHA256 hashed form.
- [If Time Allows]: Allow users to sign in through providers such as Google, Facebook, Apple, etc.

Web Application

A web interface to host the application. We will use the MERN stack to create our application. MongoDB will be used as our database to store necessary information. ExpressJS will be used to support Node and provide middleware and APIs. ReactJS is used to construct the UI and UX. NodeJS will be used to serve the website.

- As a user, I would like to have 24/7 access to the app.
- As a user, I would like the system to have the capability to handle me posting at least one event per minute.
- As a developer, we would like to be able to host at least 100 concurrent users in their general use of the app.
- As a developer, I would like to learn how to use all of these systems to create the application.
- As a developer, I would like to wireframe the User Interface before coding it (prototyping).
- As a user, I would like to access this service on mobile as well as desktop.
- As a mobile user, I would like the User Interface to be optimized for mobile use.

Usability

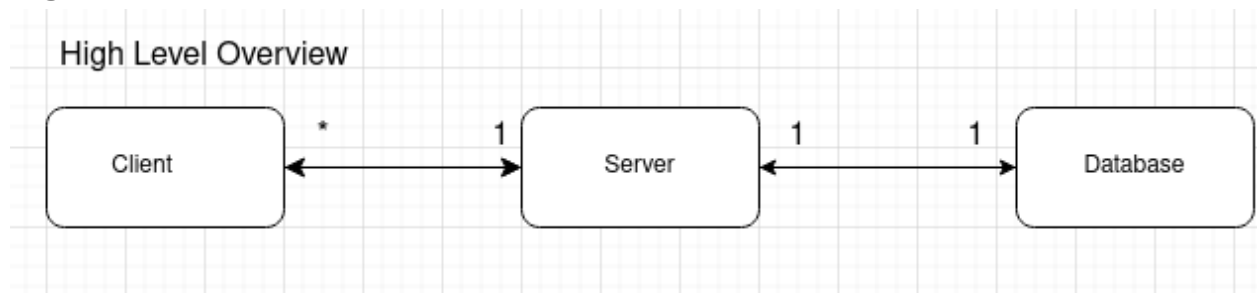
The user interface will be intuitive and easy for the average user to navigate. The interface will be designed so users can accomplish a task such as post an event, RSVP for an event, etcetera, as quickly and simply as possible by minimizing the number of clicks required to do so.

Architecture and Performance

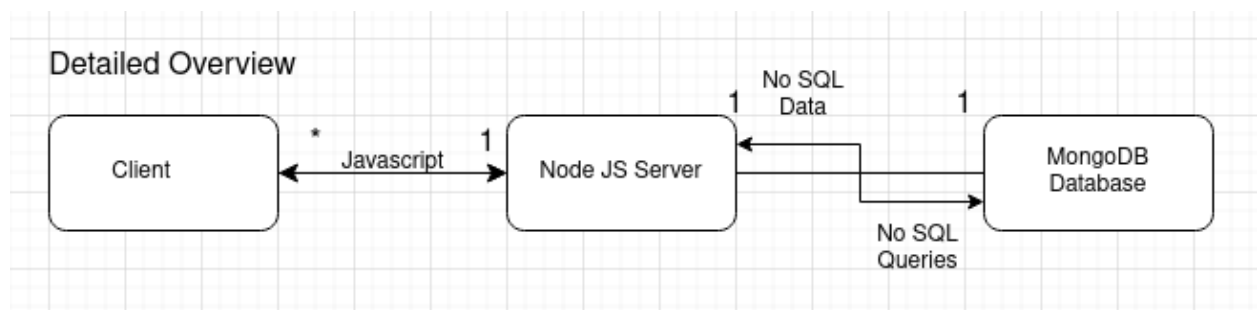
Must be able to run on the free tier of AWS.

Design Outline

High Level Overview



Detailed Overview



Our general design will include the web client, web server, and database.

Eventor will use the client-server-database model. This model works very well for common web applications. The server handles all requests for html and javascript, and then when the server needs data it requests it from the database. The client provides the user interface for the user and sends the requests to the server. It is a thin-client since it does not need to have much computational power, the server does most of the work. We picked this model not only because it is common for the previous reasons, but also because it is easy to learn. We don't have an abundance of experience with this framework, so that is important to us.

- Client
 - The client is only a thin-client and will be the user's browser. Web pages are a very common way to have application interfaces instead of building out a local interface that communicates to a server
- NodeJS server: NodeJS is used as the server because it includes many common applications that a web server will need including communicating with the client as well as communicating with the database.
- MongoDB database: This database is easy to setup and use specifically for web servers as they work well directly with JSON documents.

Design Issues

Functional Issues

1. What information is required for creating an account?
 - Option 1 - Username, Password
 - Option 2 - Username, Password, Email
 - Option 3 - Username, Displayname, Password, Email

Choice: Option 3

Justification: We want to split up username and displayname, because we need usernames to be unique in order for them to be the primary key. Display names are simply there for users to be able to show up to other users how they want. We also need an email in case a user needs to reset a password.

2. Should posts be able to have multiple associated owners?
 - Option 1 - Each post can only have one owner
 - Option 2 - Each post can have multiple owners

Choice: Option 2

Justification: For large events it would be more convenient for event managers to have multiple users who can modify the event.

3. Should posts be able to have multiple events?
 - Option 1 - Each post can only have one embedded event
 - Option 2 - Each post can have multiple embedded events

Choice: Option 1

Justification: Having multiple events in the same post would lead to visual clutter making it hard to understand. It would also increase complexity and development time. We can instead use our list of specific event types to cater to users who may have additional information to display for an event.

4. Should single-ticketed and multi-ticketed events be split into different event types in the class diagram?
 - Option 1 - Yes, split into two different event types
 - Option 2 - No, they are the same event type
 - Choice: Option 2

Justification: The data structures looked almost exactly the same with a single ticketed event having a single destination attribute and a multi

ticketed event having an array of destinations. We can just merge these and always use an array of destinations, and a single event inside the array can represent a single ticketed event as opposed to multi.

5. How should our discovery page work?
 - Option 1 - Followers of the people you follow
 - Option 2 - It is automatic based on location and interests
 - Option 3 - We first ask the user what they are looking for (i.e. location, interests) and we show those.

Choice: Option 3

Justification: We don't want to store the user location or their interests as part of their userdata. We want the discovery feed to be more manual than most social medias but we can save their own searches to make it easier on the user to be more powerful.

6. How should replying to other comments work?
 - Option 1 - Don't allow replying to other user's comments.
 - Option 2 - Allow unlimited replying down a tree of comments.
 - Option 3 - Allow only two layers of comments, "root" comments that are on the post, and replies which can only be added to "root" comments.

Choice: Option 3

Justification: It seems to be the best balance between usability and flexibility. We don't want the user to have to go down a rabbit hole of infinite comments to find the end of one chain, instead conversations are forced to be grouped in this way, but still allows replying to other user's comments which can be an essential feedback feature.

7. Should all posts be required to have an event?
 - Option 1 - Allow posts without events attached.
 - Option 2 - Require events attached to posts.

Choice: Option 1

Justification: If we don't allow bare text posts, people may just create placeholder events that only exist to allow them to make a text post. Additionally, we would like our users to be able to post updates and other information to their followers, and to do this it is critical we don't lock down the posting feature in this way.

Non-Functional Issues

1. How should the server be hosted?
 - Option 1 - Self host
 - Option 2 - AWS free tier
 - Option 3 - Firebase

Choice: Option 2

Justification: Using AWS EC2 allows us to run whatever services we want without worrying about any issues with selfhosting. It also allows us to not be dependent on firebase, but on our own deployment.

2. What stack should we use?
 - Option 1 - MERN stack
 - Option 2 - MEAN stack
 - Option 3 - Ruby On Rails
 - Option 4 - .NET

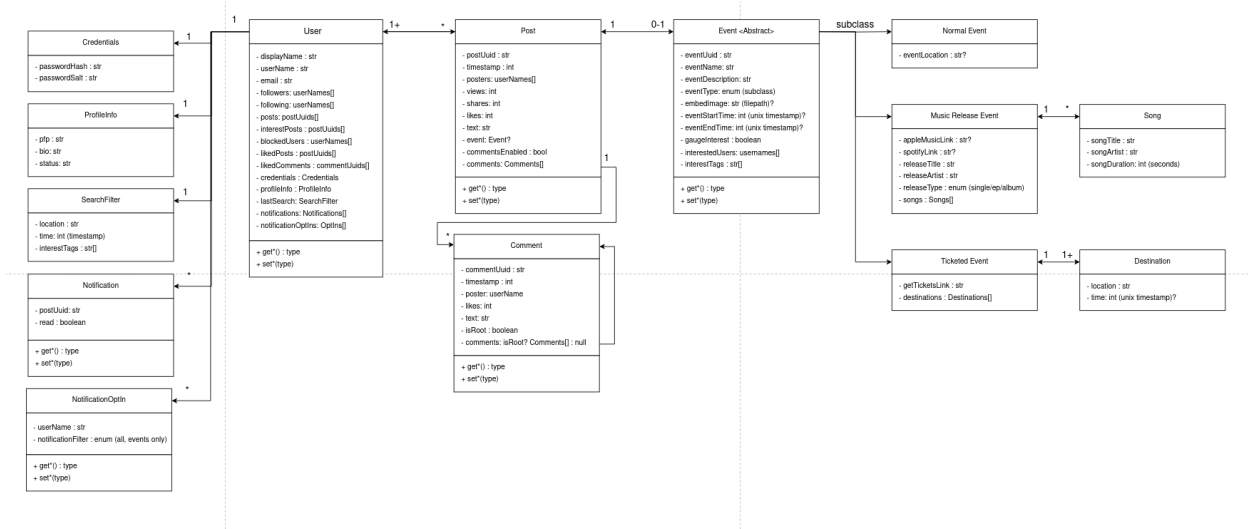
Choice: Option 1

Justification: MERN Stack seems to suit our needs best. .NET is not required because we aren't dealing with anything Windows specific. MEAN stack is very similar to MERN stack, but we decided that we wanted to learn React more than Angular. Ruby On Rails is also not as good of a choice because we do not want to use Ruby.

Design Details

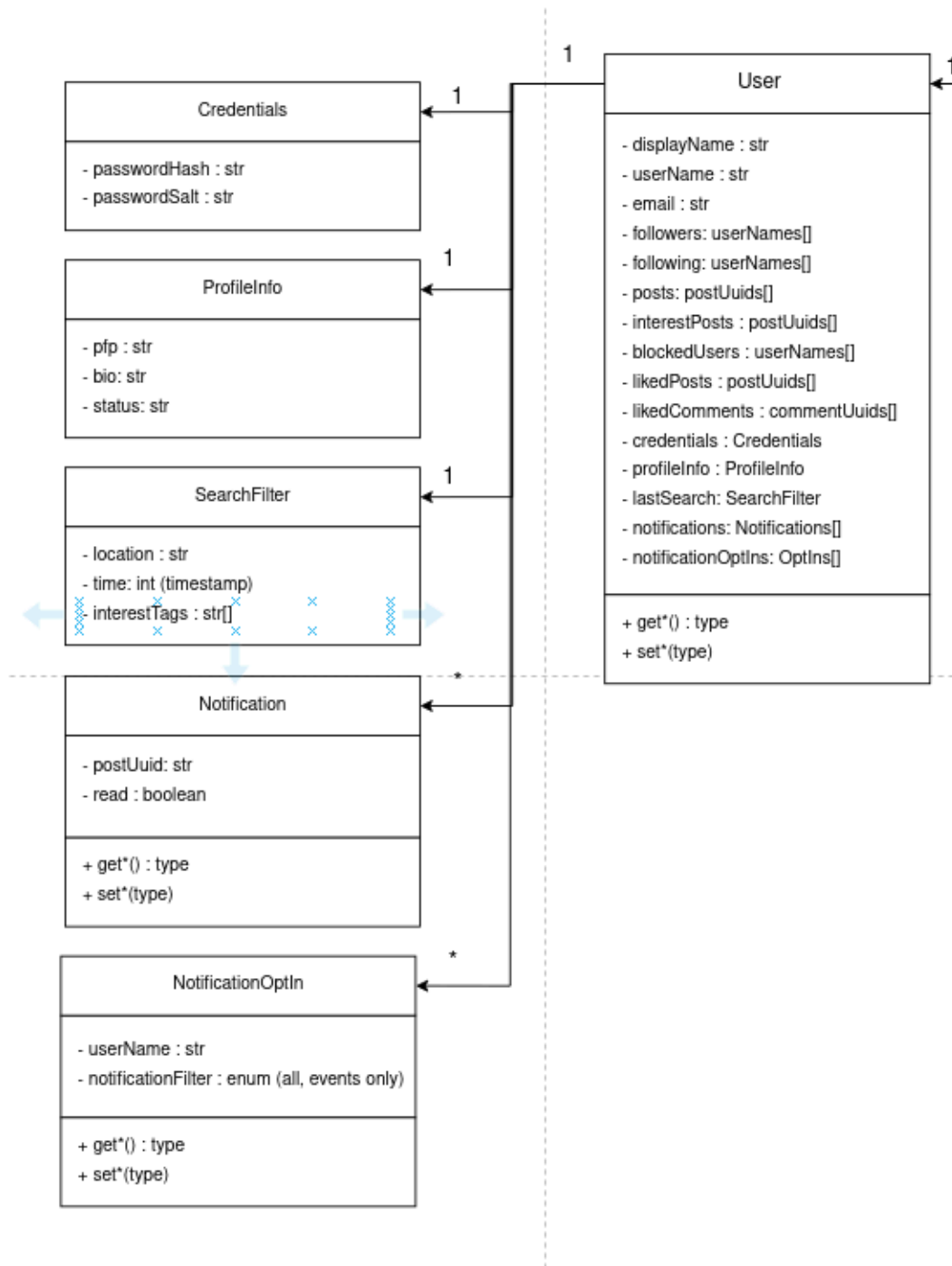
Class Diagrams

Whole Class Diagram:

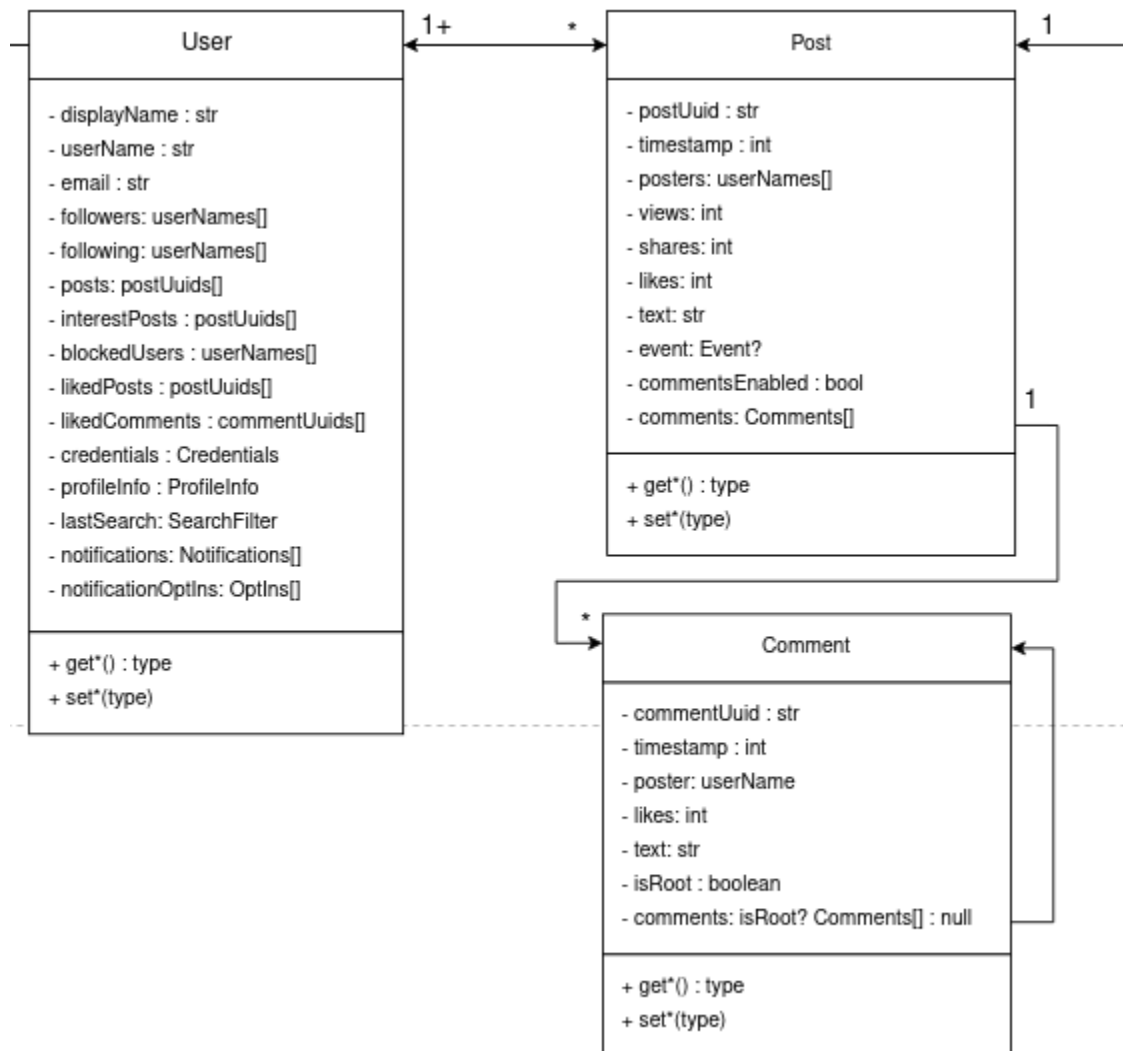


This will be gone into in further detail below.

User class and it's helper classes



User <-> Post relationship:



1. User

- Each user object contains all data related to the function of a user.
- At any time the user will be able to modify their data

2. Post

- A standard social media post that allows users to communicate with other users
- Can contain an embedded event
- Users can leave comments on posts

3. Credentials

- Stores information related to the login system and cryptography

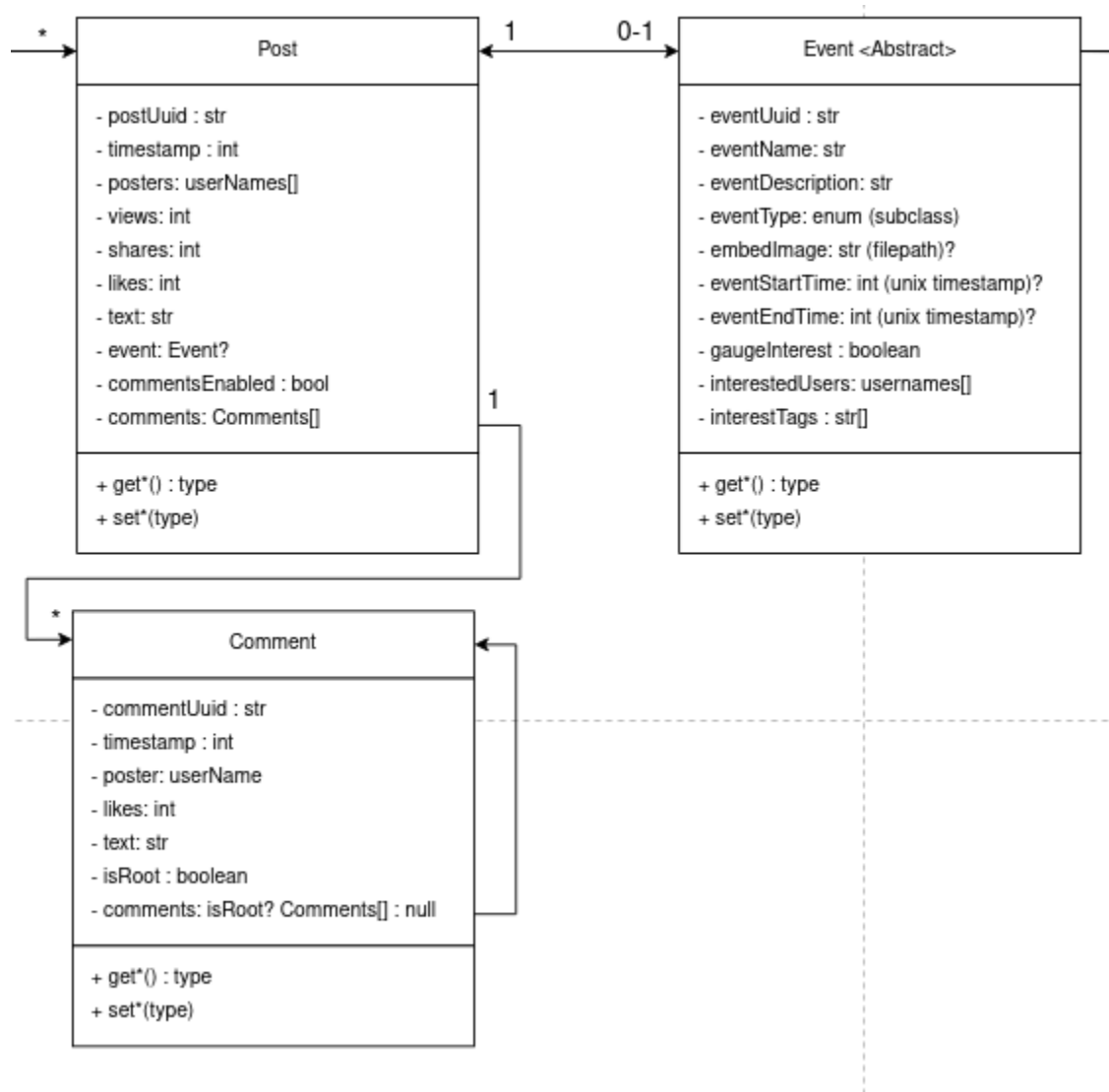
4. ProfileInfo

- Stores a user's personal info

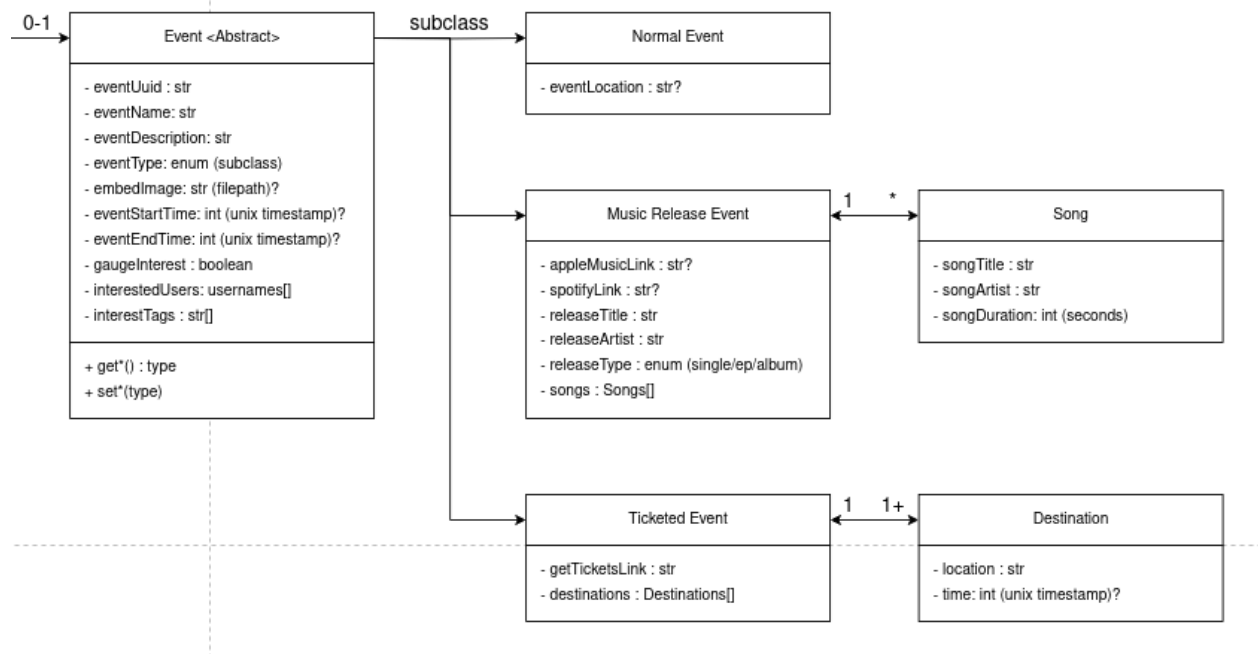
5. SearchFilter

- Stores the filters of the most recent discovery search made by the user to auto populate next time they search.

Post <-> Event relationship



Event -> Subclasses:



Descriptions of Classes and Interaction Between Classes

6. Event <Abstract>

a. Normal Event

- i. A standard event, without much special going on, that allows the poster to specify a location for the event if they desire.

b. Music Release Event

- i. Online event announcing a music release. Allows the poster to link to Apple Music and Spotify as well as provide information of the music release to embed a preview within the post.

c. Ticketed Event

- i. An event post format that allows the poster to add a link to “Get Tickets” for an event. Supports multiple destinations for cases such as an artist going on tour.

7. Song

- a. Contains information related to the song
- b. Used to preview the songs within an album

8. Destination

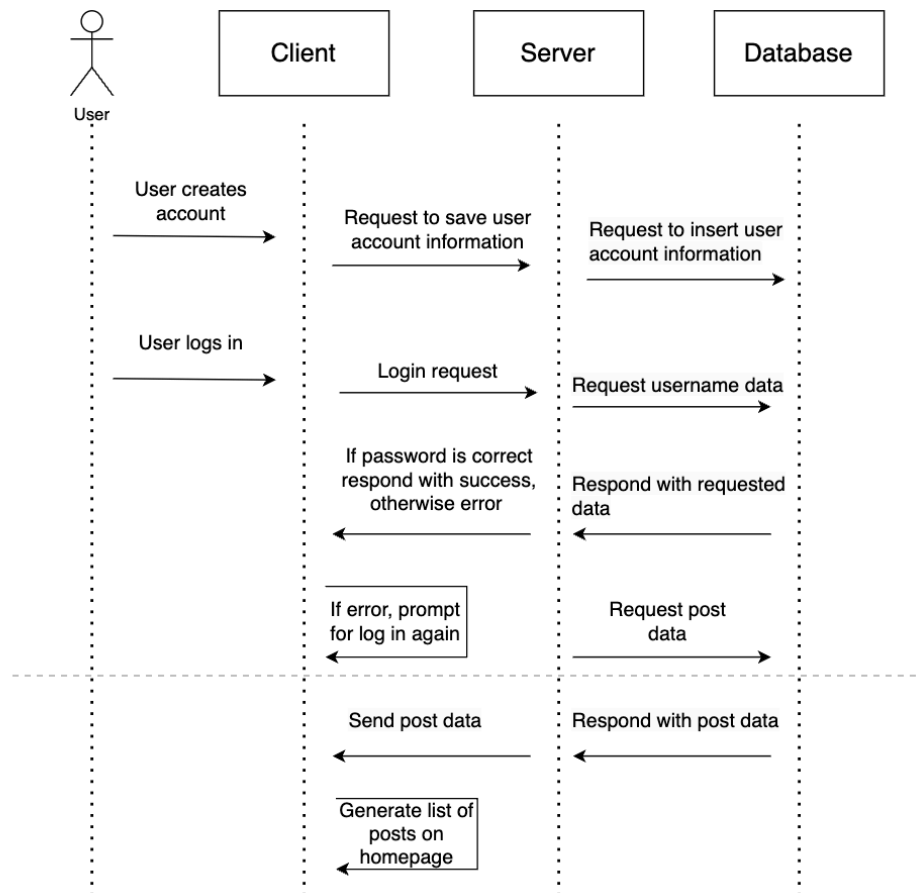
- a. Stores the time and location of an individual destination within a tour or other ticketed event

9. Comment

- a. Can either comment directly on the post (root comment) or can reply to a root comment
- b.

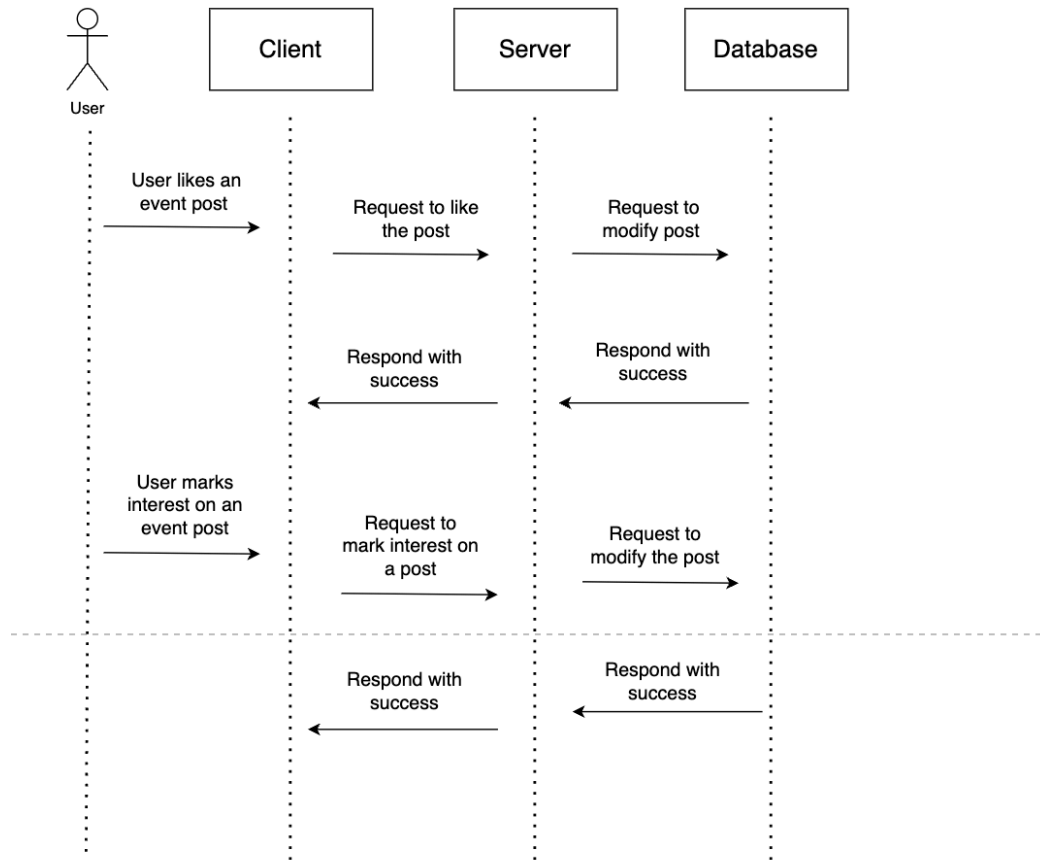
Sequence Diagrams

1. Sequence of events when a user creates an account and logs in



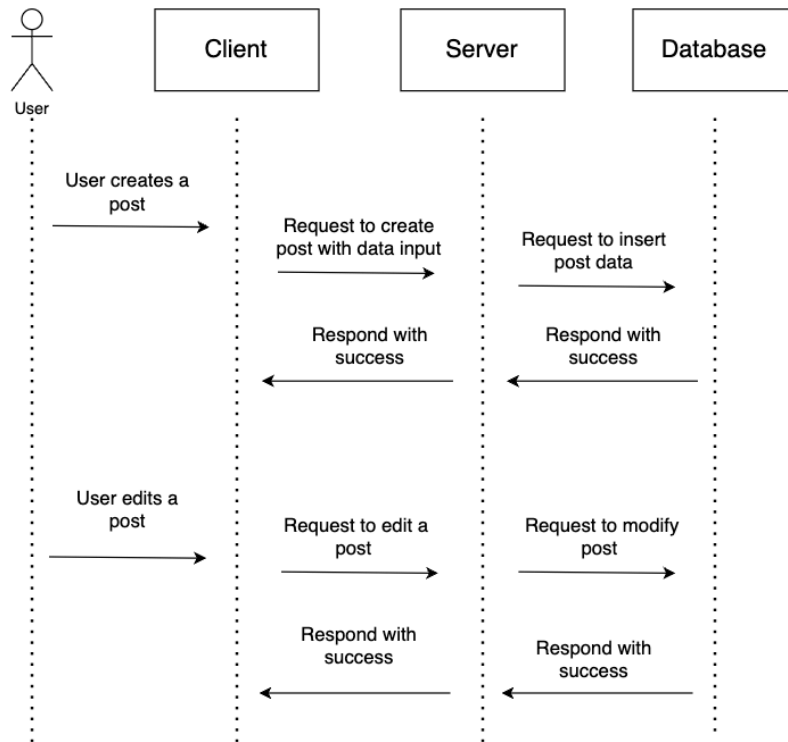
This diagram shows the sequence of events that happen when a user creates an account and then logs in. The diagram is showing the two simple events that the user does. The basic idea is that the user will input all of their information and send it to the server. The server must save it to the database and then generate the next page.

2. Sequence of events when a user likes and marks interest on an event



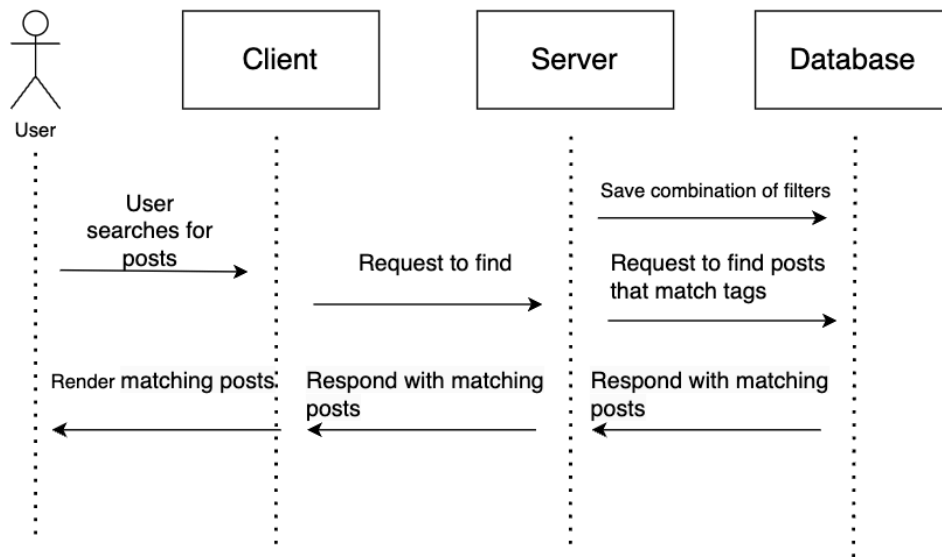
This diagram shows the sequence of events that happen when a user likes and marks interest for an event. Again, the idea here is that the user is generating information that the server must save to the database. The user is shown feedback when their action is completed.

3. Sequence of events when a user creates and edits a post



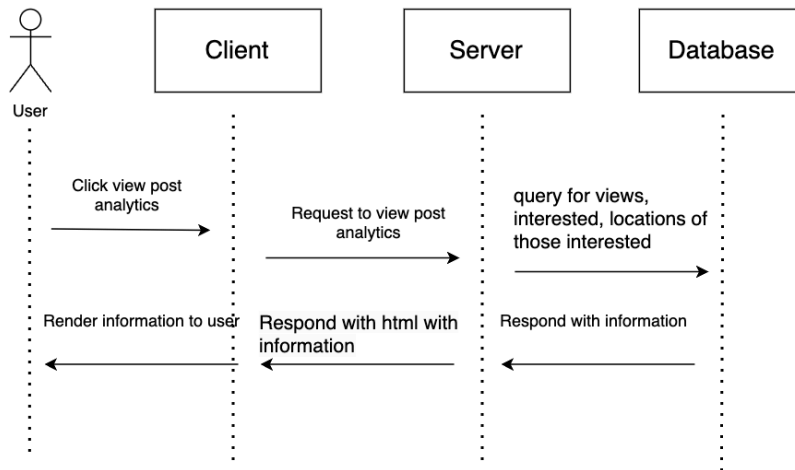
This diagram shows the sequence of events that happen when a user creates and edits a post. The user is again generating data for the server to save to the database. When editing the post, they are first shown what the previous data was, and are allowed to edit. The user is shown feedback to let them know they are done.

4. Sequence of events when a user searches for posts



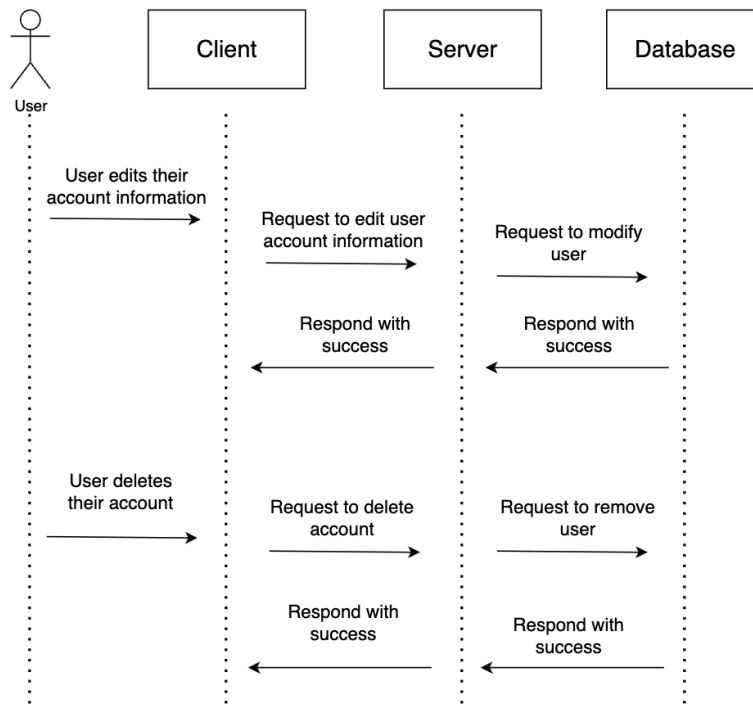
This diagram shows the sequence of events that happen when a user searches for posts. When the user is searching for the post, they also include a set of parameters with their search. One being location, and the other being tags. The server finds matching posts in the database, and then shows them to the user.

5. Sequence of events when a user views post analytics



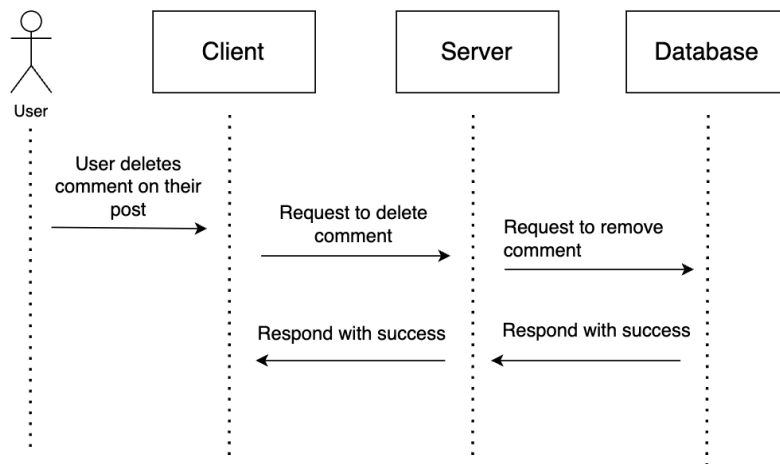
This diagram shows the sequence of events that happen when a user wants to view post analytics. The server must request the numbers and data from the database in order for it to be shown to the user.

6. Sequence of events when a user edits and deletes their account



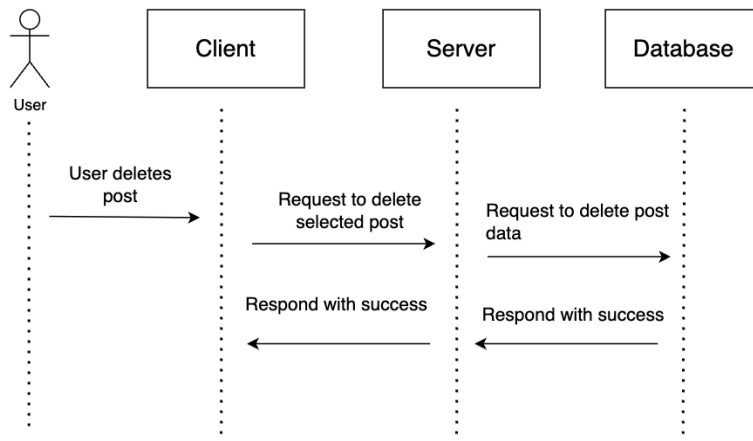
This diagram shows the sequence of events that happen when a user edits or deletes their account. Editing account information includes showing the user previous information, as well then sending information back to the server for it to save to the database. If the user deletes their account, we delete their information from the database.

7. Sequence of events when a user deletes a comment on their post



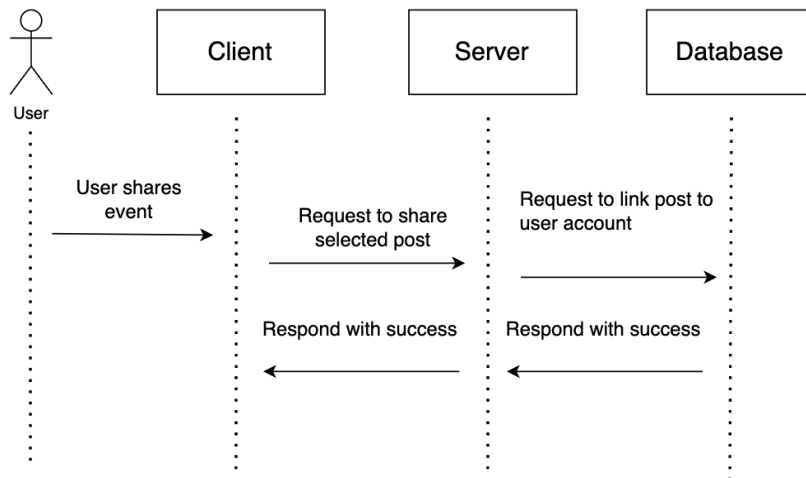
This diagram shows the sequence of events that happen when a user deletes a comment on their post. The user sends a request to the server, which then deletes the information from the database.

8. Sequence of events when a user deletes a post



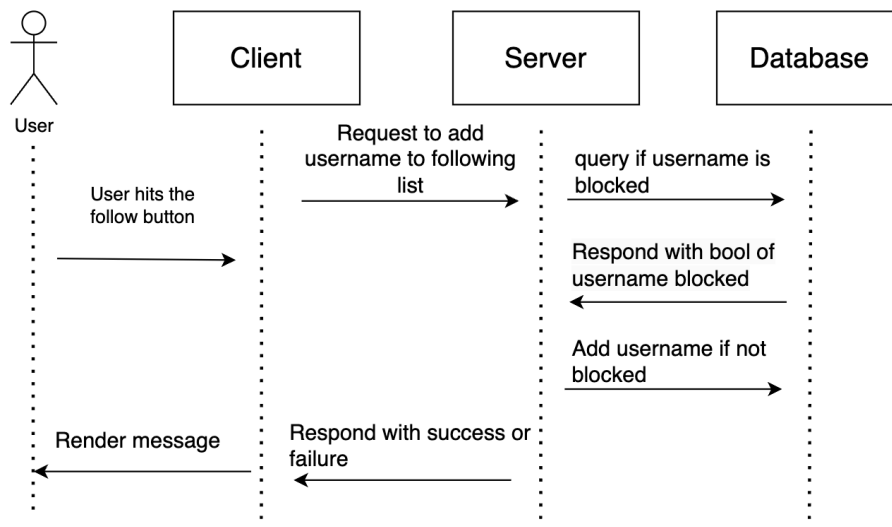
This diagram shows the sequence of events that happen when a user deletes a post. Once the user sends the request to delete the post by pressing a button, the server deletes the post from the database, and then sends feedback.

9. Sequence of events when a user shares an event



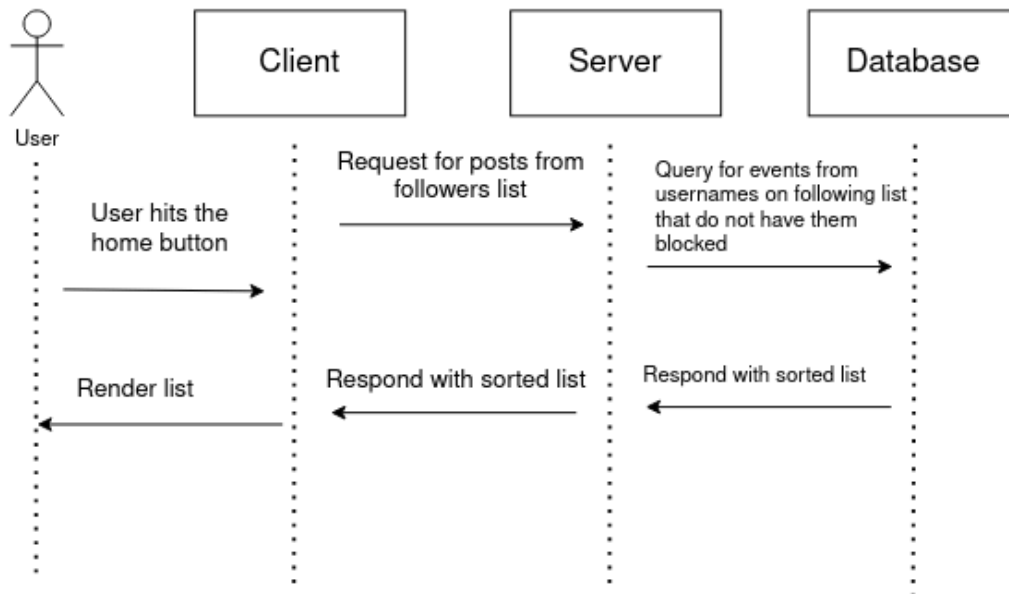
This diagram shows the sequence of events that happen when a user shares an event on their feed. The selected post must be linked to their profile in the database. Feedback is then sent to the user.

10. Sequence of events for following someone



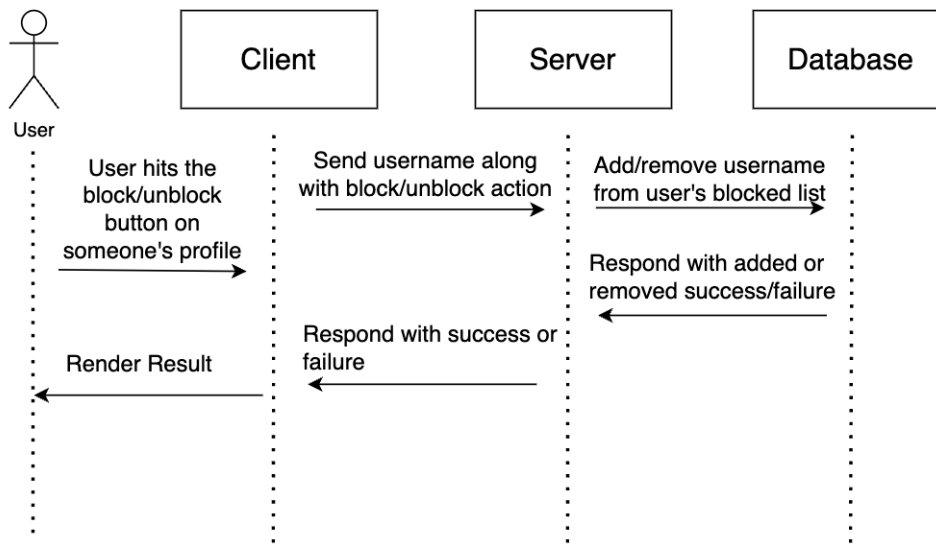
This diagram shows the sequence of events that happen when a user follows someone. We keep multiple lists of followers and following, so when the user follows someone, the server must update that information in both places.

11. Sequence of events for showing the user's feed



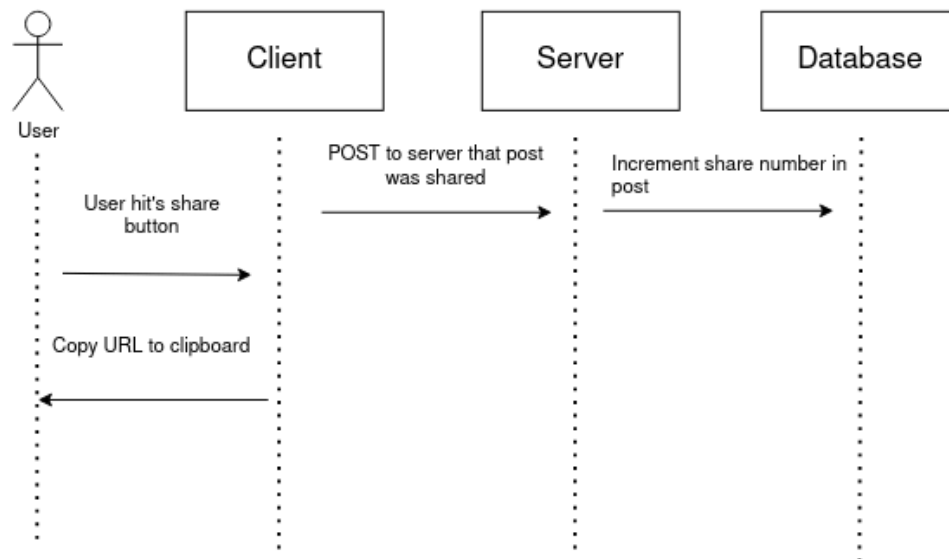
This diagram shows the sequence of events that happen when showing a user's feed. They hit the button which requests from the server the information. The server queries the information from the database and shows it back to the user.

12. Sequence of events for blocking and unblocking a user



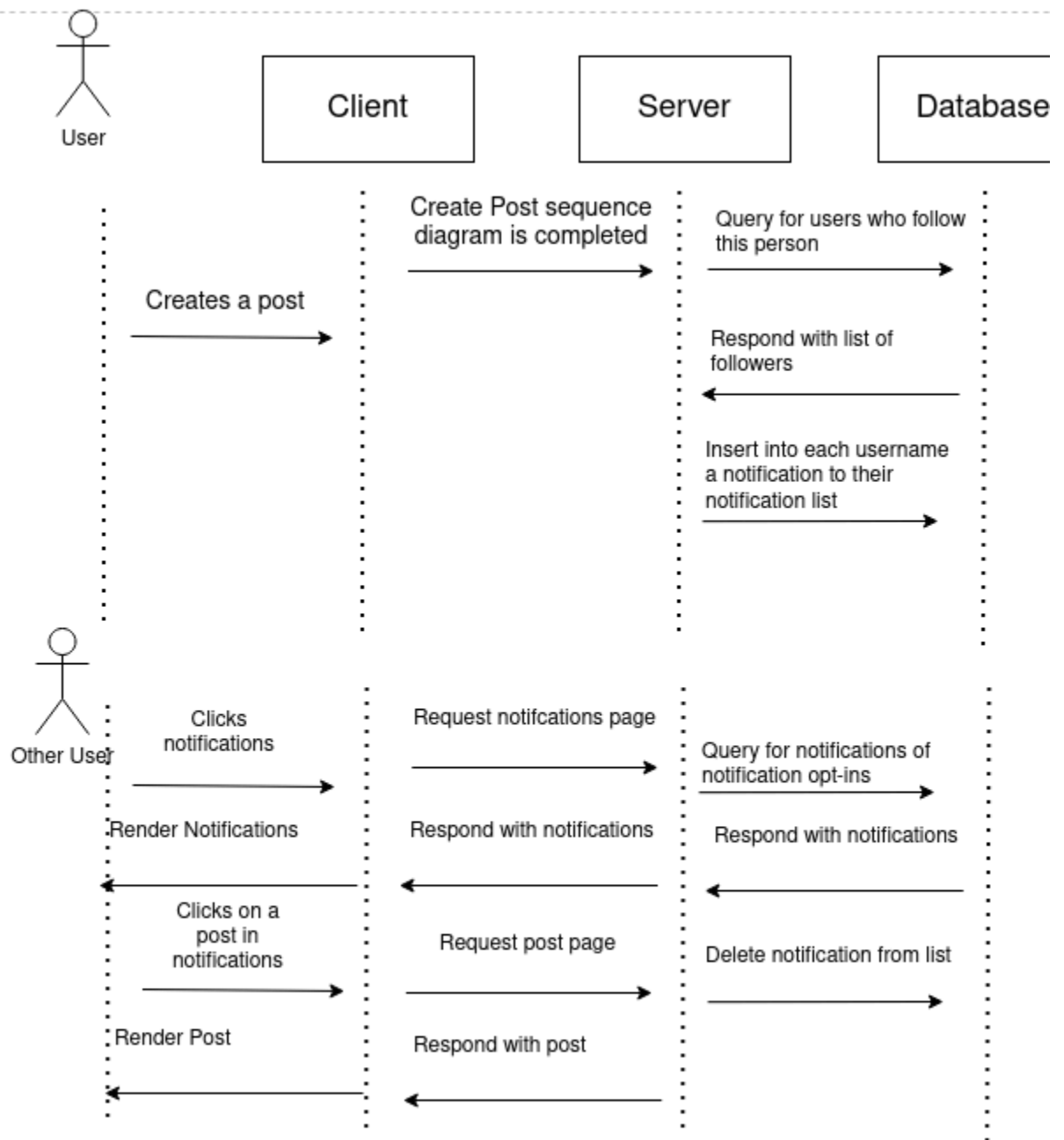
This diagram shows the sequence of events that happen when a user blocks or unblocks a user. They send the action to the server, and the server updates the database depending on what type of action it is.

13. Sequence of events for sharing a link



This diagram shows the sequence of events that happen when a user shares an event. While a user could copy the url and share that, we want the share button to send a POST to the server so it can increment a share counter in the database.

14. Sequence for a user's notification feed



This diagram shows the sequence of events that happens for a user's notifications. When a user sets another user with a notification status, a sequence of events will happen when that user posts an event. It checks to see who is following them, and of those, who has notifications set to on. It then adds that post to each of the followers notification lists.

When a user clicks notifications, they request from the server and database all the posts

which they want to be notified of. Once the user clicks on a post, the post is removed from their notifications list.

UI Mockups

Login Page

EVENTOR

Post. Meet. Share

Login

New user? [Sign up](#)

[Forgot Password?](#) [Reset Password](#)

Sign Up Page

EVENTOR

Post. Meet. Share

Sign up

Already a user? [Login](#)

Feed Page

EVENTOR



Suga Sean
@sugasean2

My Feed

Refreshing Feed

Ben

@ben10

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt molestie ultricies. Aliquam erat volutpat. Proin dictum nibh at lectus faucibus vehicula

1st

100

1

0

Joe

@joem

Pull up.

21

3

2

8

Suga Sean

Posted a new event: "Whale Festival tomorrow 8pm ET..."

All caught up!



Suga Sean
@sugasean2

Discover

What are you looking for?

Nothing Here Yet

Enter a search to get started with Discovery.

EVENTOR



Suga Sean
@sugasean2

Discover

Kanye Wes|



(search filters here)

Discover

Nothing Here Yet

Enter a search to get started with Discovery.



Suga Sean
@sugasean2

Discover

Kanye West




Kanye West

@kanyewest

1.1M Followers


Follow



Kanye West

@kanyewest

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt molestie ultricies. Aliquam erat volutpat. Proin dictum nibh at lectus faucibus vehicula





User Profile Page

EVENTOR



Suga Sean
@sugasean2



Suga Sean
@sugasean2

109 Followers

"Every day is a new beginning" Edit Status

About

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt molestie ultricies. Aliquam erat volutpat. Proin dictum nibh at lectus faucibus vehicula



What will you be hosting next?



Post

Nothing Here Yet

Create a post for it to show up on your profile!

EVENTOR



Suga Sean
@sugasean2



Joe Mama
@joe32
"Yo mamma so fat!"

13 Followers

Follow



About

Yo mamma is so fat, I took a picture of her last Christmas and it's still printing.

Nothing Here Yet

Joe Mama hasn't made any posts yet!

EVENTOR



Suga Sean
@sugasean2



Kanye West
@ye
"GOD BREATHED ON EARTH"

1.1M+ Followers

Following



About

Keep your nose out the sky, keep your heart to god, and keep your face to the raising sun.



Kanye West
@kanyewest

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tincidunt molestie ultricies. Aliquam erat volutpat. Proin dictum nibh at lectus faucibus vehicula



End of Feed

You've reached the end of Kanye West's posts!