streamshore

Team 12 - Product Backlog

Nancy Agarwal, Harshitha Janardan, Seth Maxwell, Pranjali Raturi, Ian Ryan, Jack Sovich

Problem Statement

The rise of new platforms such as YouTube have revolutionized how we consume media in Western culture. While online digital media is perhaps easiest to consume alone, many consumers desire to share media viewing with others. *streamshore* is designed to fulfill this niche, and make the viewing of online videos a shared experience similar to going to the movies together - no matter how far apart. The project supports the creation of virtual viewing rooms for a group to interact amongst themselves while watching a live stream of submitted videos.

Background Information

Audience

Many casual YouTube viewers desire to watch a video in real-time with someone else to share a reaction without being in the same physical location.

Similar Platforms

There are various similar platforms that allow users to watch videos synced together at the same time. A web application that is similar to *streamshore* is Facebook Watch Party. Both *streamshore* and Facebook watch party allow for groups of users to watch videos in sync and use the live chat feature. Another platform is *watch2gether.com* where you can join online chat rooms and watch videos with your friends. Additionally, *plug.dj* exists as a music-focused listen party service.

Limitations

The main limitation that many similar platforms have is that they are hard to navigate. For Facebook Watch Party only supports videos that are hosted on Facebook. However, *streamshore* allows users to watch Youtube videos, which is a more common video service platform compared to Facebook Watch Party. *Watch2gether.com* is difficult to navigate and has an inaccurate search engine which we plan on improving alongside the user experience. *plug.dj* serves as a valuable example for room management, but is primarily aimed at music consumption, and thus alienates the more casual audience, which *streamshore* targets.

Requirements (Backlog)

Functional Requirements

1. As a user, I would like to create an account on *streamshore*.

2. As a user, I would like to log in to via my account credentials.
3. As a user, I would like to recover my account password in the case that I forget it.
4. As a user, I would like to verify my email address.
5. As a user, I would like to edit my display name.
6. As a user, I would like to choose a preset color scheme.
7. As a user, I would like to change my password.
8. As a user, I would like to add other users as friends.
9. As a user, I would like to view a list of my friends.
10. As a user, I would like to see which viewing rooms my friends are in.
11. As a user, I would like to easily join private rooms hosted by friends.
12. As a user, I would like to see the three most previous rooms I participated in.
13. As a user, I would like to invite others to the room I am currently in.
14. As a user, I would like to add videos to a room's queue.
15. As a user, I would like to see videos that are in the video queue.
16. As a user, I would like to create a room to watch videos.
17. As a user, I would like to know how many people are online in a particular room and view a list.
18. As a user, I would like to participate in a room as an anonymous user without logging in.
19. As a user, I would like to toggle my view of the live chat or online users.
20. As a user, I would like to directly mention someone in the live chat.
21. As a user, I would like to direct message my friends privately (if time allows).
22. As a user, I would like to see history of direct messages (if time allows).
23. As a user, I would like to be notified when I receive a direct message (if time allows).
24. As a user, I would like to participate in a live chat with others in the room.
25. As a user, I would like to like or dislike a video that is currently playing.
26. As a user, I would like to like or dislike a video that is in the queue.
27. As a user, I would like to play and pause the video locally and have it resume back to the room's video timestamp.
28. As a user, I would like to locally adjust the video's volume.
29. As a user, I would like to make a video fullscreen.
30. As a user, I would like to view the videos in a theater mode.
31. As a user, I would like to create a playlist of videos to submit to the room's queue.
32. As a user, I would like to discover rooms that are marked public.
33. As a user, I would like to search for rooms by title.
34. As a user, I would like to to navigate easily to other pages on the site.
35. As a user, I would like to have a step-by-step guide on how to submit a video to the video queue.
36. As a user, I would like to submit videos from additional platforms other than YouTube (if time allows).
37. As a user, I would like to report a user to an administrator.
38. As a user, I would like to report a room manager to an administrator.

39. As a user, I would like to become a premium user by attaining a set threshold of users in a room.
40. As a premium user, I would like to have my public rooms featured at the top of the homepage.
41. As a premium user, I would like to have a special identifier.
42. As a premium user, I would like to have unlimited rooms (normal users limited to a certain amount).
43. As a room manager, I would like to set a name, URL, and description when creating a room.
44. As a room manager, I would like to modify the description of my room.
45. As a room manager, I would like to manage chat messages.
46. As a room manager, I would like to mute a user.
47. As a room manager, I would like to promote a user to be a room manager.
48. As a room manager, I would like to push a video to the front of the queue.
49. As a room manager, I would like to remove a video from the queue.
50. As a room manager, I would like to restrict who can submit videos.
51. As a room manager, I would like to host both private (hidden, link required) and public (easily discoverable) viewing rooms.
52. As a room manager, I would like to host a friends-only viewing room.
53. As a room manager, I would like to enable a live chat filter.
54. As a room manager, I would like to disable the live chat in a room I have created.
55. As a platform administrator, I would like to ban a user.
56. As a platform administrator, I would like to view user reports.
57. As a platform administrator, I would like to remove rooms.
58. As a platform administrator, I would like to send announcement emails to all registered users.
59. As a platform administrator, I would like to view a list of all active rooms.
60. As a platform administrator, I would like to view a list of all registered users
61. As a platform administrator, I would like to block the use of certain words in room names, URLs, and/or descriptions as well as user display names.

Non-Functional Requirements

Framework

We are building a web application with a separate frontend and backend. We intend to use Vue.js (JavaScript) as our framework for the frontend, and Phoenix (Elixir) as our framework for the backend. Utilizing a separate frontend and backend allows us to use languages with different goals (design being a priority for frontend, and performance being a priority for backend). In addition to Phoenix, we will use Ecto with MySQL to manage the database.

Security

Proper practices will be used to minimize risk of common vulnerabilities, such as SQL injection and man-in-the-middle attacks (ex. form validation and https). Additionally, there will be a

privileged user system, both site-wide (for administration), and per-room (for room management and moderation).

Usability

The interface will be created in a way that makes it easy to navigate and understand every feature. It will be easy to create user accounts and chat rooms. We want to make it simple for the users to add videos from YouTube and queue them. Users should be able to participate, even if in a limited manner, without creating an account, as requiring account creation tends to be a barrier to entry for first-time users. Search will also be a primary feature, as having to leave the site to find videos takes users out of the experience - the search should not only be present, but accurately represent YouTube search results (something *watch2gether* greatly fails at).

Scalability

Our backend, Elixir, is deliberately chosen so that our application will scale sufficiently to handle any amount of users. Using Discord as a case study (https://blog.discordapp.com/scaling-elixir-f9b8e1e7c29b), it is evident that with proper design, an Elixir backend can scale to handle a vast amount of users concurrently. The system should be able to handle a workload of at least 10,000 concurrent requests with a chat delay of less than 1 second from message send to message received, provided a standard, working internet connection. Given our intended division of tasks, the backend should be the limitation for concurrent users, not the frontend.

Hosting/Deployment

Both the frontend and backend should have a 24 hour availability. In order to accomplish this, a VPS will be utilized for deployment of the backend.