# COMP3900-H15A-capSquad - Project Proposal

Daniel Latimer z5115175
Connor O'Shea z5115177
Kevin Chan z5113136
Oliver Richards z5157383

Peter Kerr
z5115807

October 4, 2020

## Contents

**1 Background** **2**
1.1 Problem Domain . . . 2
1.2 Existing Systems . . . 2

**2 User Stories** **3**
2.1 Product Backlog . . . 3
2.1.1 Project Objectives Stories . . . 3
2.1.2 System Stories . . . 5
2.1.3 Novel Features Stories . . . 6
2.2 Sprints . . . 7

**3 Interface and Flow Diagrams** **8**

**4 System Architecture** **8**

**A Recommendation Engine Flow Diagram** **11**

**B Storyboards** **12**

# 1 Background

## 1.1 Problem Domain

With over 100 million monthly listeners[9] and a steadily increasing user base, there is no doubt that podcasts are a greatly enriching source of information and entertainment for a large variety of individuals.

Although they are highly valuable, it can be difficult to find podcasts that are of interest to a particular user amidst the 1 million[9] that are already available. Thus, podcast streaming services (such as UltraCast) have been created, to provide a centralised place for exploring and discovering new podcasts that are valuable to the listener.

However, all of the web based podcast streaming services available lack many important features, and their interfaces leave much to be desired. For example, there is no streaming service that allows the user to bookmark certain parts of a podcast, nor take notes at certain timestamps. It is even difficult to find a service that allows the listener to change the playback speed of the podcast.

capSquad believes that there is no one service that combines all of the most important features together into a single package, leaving space in the market for a superior podcast streaming platform. We seek to fill this gap with our new web-based podcast streaming service, UltraCast.

## 1.2 Existing Systems

There are a number of existing podcast streaming services on the market. Some of the key failings that capSquad aims to rectify with UltraCast are listed in the following section, using Spotify[5], Sticher[6] and Player FM[7] as case studies. A summary of the comparison is shown in table 1.

All three solutions provide functionalities for users to listen and subscribe to podcasts. Podcasts are recommended to users, however, the basis for these recommendations is not clear.

They also allow content creators to add podcasts to the respective platform. In this regard Player FM is distinct from Spotify and Sticher, Player FM aggregates podcasts from iTunes or RSS feeds which are added by content creators or listeners. Player FM does not serve the podcasts itself.

As table 1 shows, there are a number of key features that are not available on any platform. A podcast service which provides these features will improve the usability and engagement of the platform for listeners.

Table 1: Comparison of existing podcast services

| Feature | Spotify | Stitcher | Player FM |
|---|---|---|---|
| Can see number of subscribers for podcasts | x | x | ✓ |
| Finished episodes are marked as played | ✓ | ✓ | x |
| Notifications for new episodes added to subscribed podcasts | x | x | x |
| Centralised place to view previously listened to episodes | x | x | x |
| Recommended podcasts are stylised to the user | ✓ | x | ✓ |
| Can skip to previous episode | ✓ | x | ✓ |
| Can adjust playback speed | x | x | ✓ |
| Closed captions | x | x | x |
| Ability to bookmark timestamps / take notes | x | x | x |
| Can follow other users | ✓ | x | x |
| Can see the latest episode for each subscribed podcast in a preview panel | x | x | x |

# 2 User Stories

## 2.1 Product Backlog

The 29 user stories which make up the product backlog, as shown in Figure 1, were grouped into three categories as described in the sections 2.1.1, 2.1.2 and 2.1.3. Screenshots of the stories which make up the backlog can be found in sections 2.1.1, 2.1.2 and 2.1.3.

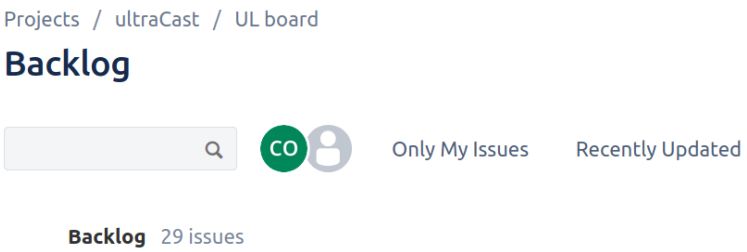



Figure 1: UltraCast Backlog Count

### 2.1.1 Project Objectives Stories

The project objective stories were derived directly from the project objectives. These JIRA stories can be found in Figure 2 below. The mapping of each project objective to the final user story was summarised in Table 2.

Table 2: Project Objectives to Stories Mapping

| **Project Objective** | **Story Key** |
|---|---|
| Listeners must be able to search for podcasts that interest them by keywords, resulting in a list of matching podcast titles, where the total number of subscriptions on the UltraCast platform (function described later) for each podcast is shown next to the title | UL-2<br>UL-3 |
| Listeners must be able to select a podcast show from returned search results to view its full details, including its title, description, any author details that exist, as well as a list of episodes for the show | UL-4 |
| Listeners must be able to play a selected episode within a podcast show, and once that episode starts being played, the listener must be able to also clearly see this episode marked as "Played" | UL-5<br>UL-6 |
| Listeners must be able to subscribe or unsubscribe from a podcast show | UL-7<br>UL-8 |
| Listeners must be able to see the latest episode available for each show that they subscribed to in a "Podcast Subscription Preview" panel | UL-30 |
| Listeners must be notified by the platform when a new episode for a show they are subscribed appears | UL-9 |
| Listeners must be able to see a history of the podcast episodes that they have played, sorted in order from most recently played to least recently played | UL-10<br>UL-11 |
| UltraCast must be able to recommend new podcast shows to a listener based on at least information about the podcast shows they are subscribed to, podcast episodes they have recently played, and their past podcast searches | UL-12<br>UL-13 |

| T | Key ↑ | Summary |
|---|---|---|
| | UL-2 | As a listener, I want to be able to use keywords to search for podcasts, so I can find podcasts that interest me |
| | UL-3 | As a listener, I want to be able to see the total number of subscribers for each podcast returned from searches, so I can see how popular they are |
| | UL-4 | As a listener, I want to be able to view the title, description, author details and list of episodes for a podcast, so I can see if it matches my interests |
| | UL-5 | As a listener, I want to be able play episodes, so I can listen to them |
| | UL-6 | As a listener, I want episodes I've started playing to be marked as played, so I can keep track of what I have already listened to |
| | UL-7 | As a listener, I want to be able to subscribe to a podcast, so I can keep up to date with podcasts that interest me |
| | UL-8 | As a listener, I want to be able to unsubscribe to a podcast, so I can stop following podcasts that no longer interest me |
| | UL-9 | As a listener, I want to receive a notification when a new episode for a podcast I am subscribed to is released, so I can keep up to date with episodes that interest me |
| | UL-10 | As a listener, I want to be able to view my podcast episode history, so I can review what I have listened to |
| | UL-11 | As a listener, I want to be able to sort by most recently played for my episode history, so that I can more easily find episodes I have listened to previously |
| | UL-12 | As a listener, I want to be recommended to new podcasts, so I can more easily find new podcasts that interest me |
| | UL-13 | As a listener, I want recommended podcasts to be based on my existing subscriptions, recently played episodes and past searches, so there is a greater chance I will find a podcast that interests me |
| | UL-30 | As a listener, I want to be able to see the latest episode for each subscribed podcast in a "Podcast Subscription Preview" panel, so I can easily keep up to date with the podcasts I am interested in |

Figure 2: JIRA Objective User Stories

### 2.1.2 System Stories

The system stories were designed to address common features offered by existing offerings in the same problem domain. They can be seen in Figure 3 below.

| T | Key ↑ | Summary |
|---|---|---|
| | UL-14 | As a user, I want to be able to login, so that I have a custom experience and can be identified |
| | UL-15 | As a content creator, I want to be able to create podcasts, so I can publish episodes under a common group |
| | UL-16 | As a content creator, I want to be able to delete podcasts, so I can remove podcasts and episodes within |
| | UL-17 | As a content creator, I want to be able to update podcasts, so I can easily add or remove podcast content |
| | UL-18 | As a listener, I want to be able to pause episodes, so I can resume a podcast later when it suits me |
| | UL-19 | As a listener, I want to be able to adjust the volume of episodes, so I can keep volume at a level different to other apps/programs |
| | UL-20 | As a listener, I want to be able to skip to the next, previous episode as well as the start of the current episode, so I can easily navigate between episodes |
| | UL-21 | As a listener, I want to be able to jump to a particular point in an episode, so I can easily navigate within an episode |
| | UL-22 | As a listener, I want to be able to adjust playback speed, so I can listen to episodes at a pace that suits me |
| | UL-23 | As a listener, I want to be able to auto-play episodes within a podcast, so I can easily listen to episodes sequentially |
| | UL-24 | As a listener, I want to be able to view a title, length, upload date and progress for episodes, so I have a better idea of what to expect before listening |

Figure 3: JIRA System User Stories

### 2.1.3 Novel Features Stories

The novel feature user stories, as shown in Figure 4, were designed to create desirable features that are either uncommon or not available in other mainstream offerings in the same problem domain.

| T | Key ↑ | Summary |
|---|---|---|
| | UL-25 | As a listener, I want closed captions to be available, so I can understand what is being said if audio is an issue |
| | UL-26 | As a listener, I want to be able to "bookmark" points in episodes, with a title and short description and view bookmarks on a separate page, so I can track of points of interest |
| | UL-27 | As a content creator, I want to be able to view episode and podcast metrics; plays, subscribers viewer location, viewer age, so I can better tailor my content |
| | UL-28 | As a listener, I want to be able to follow friends, so I can see what they are listening to |
| | UL-29 | As a listener, I want to be able to save previous searches as "streams", so I can avoid searching for the same thing multiple times |

Figure 4: JIRA Novel User Stories

Table 3: Comparison of proposed novel features with competitors. User stories are shown in figure 4

| User Story | Spotify | Stitcher | Player FM |
|---|---|---|---|
| UL-25 | X | X | X |
| UL-26 | X | X | X |
| UL-27 | ✓ | X | X |
| UL-28 | ✓ | X[1] | X |
| UL-29 | X | X | X |

[1] Some limited metrics on listening platform and number of view available

## 2.2 Sprints

The sprint planning will occur on Thursdays and the sprint duration will be exactly one week as seen in Table 4.

Table 4: Sprint Dates

| Week | Sprint Start | Sprint End |
|---|---|---|
| 3-4 | 01/10/20 | 08/10/20 |
| 4-5 | 08/10/20 | 15/10/20 |
| 5-6 | 15/10/20 | 22/10/20 |
| 6-7 | 22/10/20 | 29/10/20 |
| 7-8 | 29/10/20 | 05/10/20 |
| 8-9 | 05/10/20 | 12/10/20 |

Relatively few stories were selected for the first sprint, as seen in Figure 5 as there will be a decent amount of work in setting up the project infrastructure. The sprints that were selected were those that focused on creating the data models (e.g. a podcast schema) which will be used later on by other stories, such as UL-2 (search for podcasts).

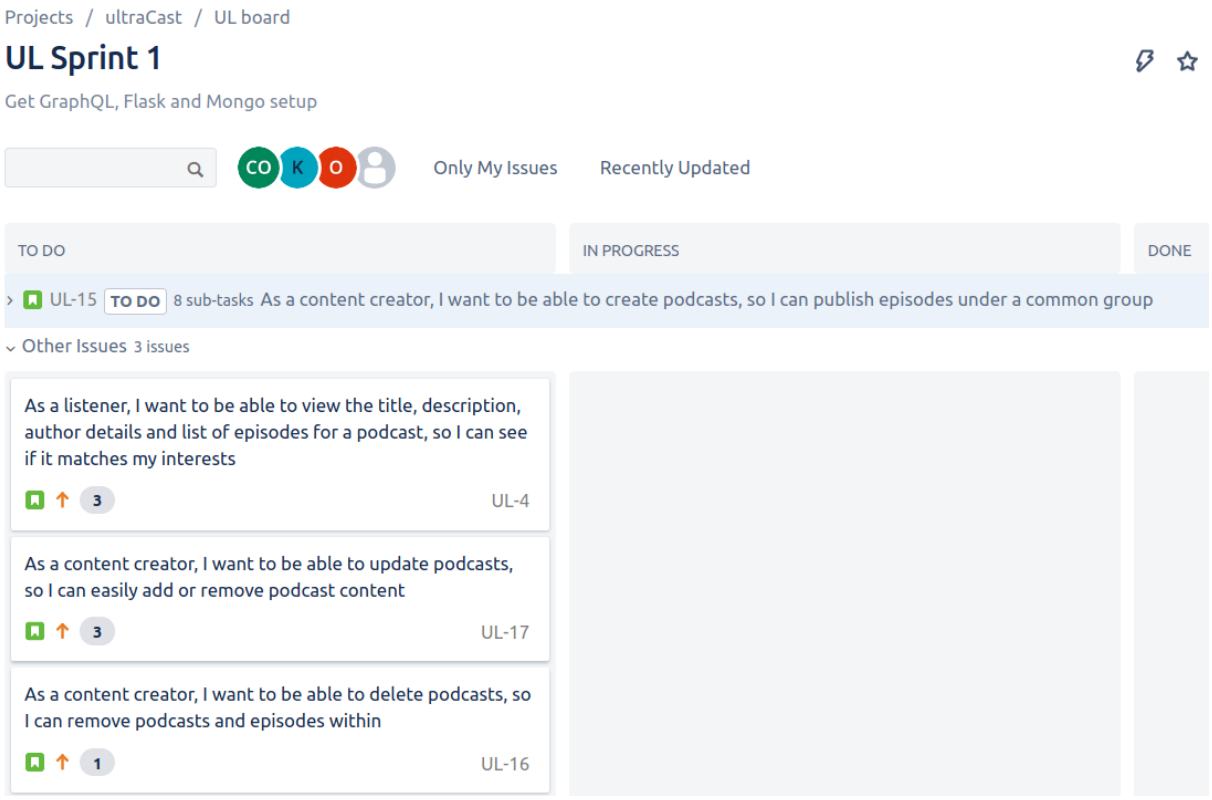



Figure 5: First Sprint (01/10/20 - 08/10/20)

# 3 Interface and Flow Diagrams

Identifying recommended podcasts for users involves a non-trivial back-end implementation. Thus a flow diagram of the process is shown in appendix A.

Storyboards for all user stories are shown in appendix B.

# 4 System Architecture

The proposed system architecture can be seen in Figure 6. It can be seen that our end users will be podcast listeners and content creators. First, we will use MongoDB[8] to store our data, a NoSQL database that is popular for its high scalability. This service will be interfaced with the MongoDB-Python driver, available on the MongoDB website[8]. Next, we will be using Flask[2] for our web-server: a micro-framework that allows us to quickly develop an MVP solution. Flask is written in Python, so connecting to the database via the MongoDB-Python driver should be straightforward. Additionally, we will have a recommendation service that will generate recommended podcasts based on the users listening history. Finally, we will have a React[4] front-end application that will enable our users to login, search and play podcasts, and get recommendations on ones they may be interested in. The React application and Flask application will communicate through a GraphQL API: a scalable alternative to the popular REST API[3].

The architecture has been designed with the final demonstration in mind, hence, the business and presentation layers are shown to be hosted on the VLab machine. Currently, MongoDB is not supported by Debian 6 (the Linux environment on the VLab machine),

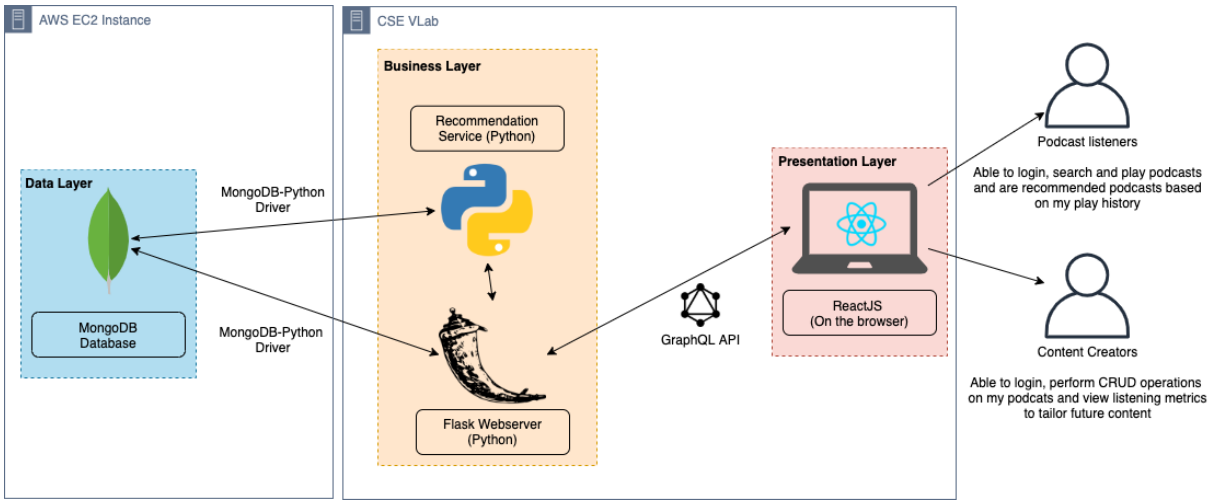



Figure 6: Proposed System Architecture

so we have opted to put the data layer onto an AWS EC2 instance[1].

# References


[1] Amazon ec2: Secure resizable compute capacity to support virtually any workload. https://aws.amazon.com/ec2/. Accessed: 2020-10-03.

[2] Flask: web development, one drop at a time. https://flask.palletsprojects.com/en/1.1.x/. Accessed: 2020-10-03.

[3] Graphql. https://graphql.org/. Accessed: 2020-10-03.

[4] React: A javascript library for building user interfaces. https://reactjs.org/. Accessed: 2020-10-03.

[5] Spotify. https://www.spotify.com, October 2008. Accessed: 2020-10-02.

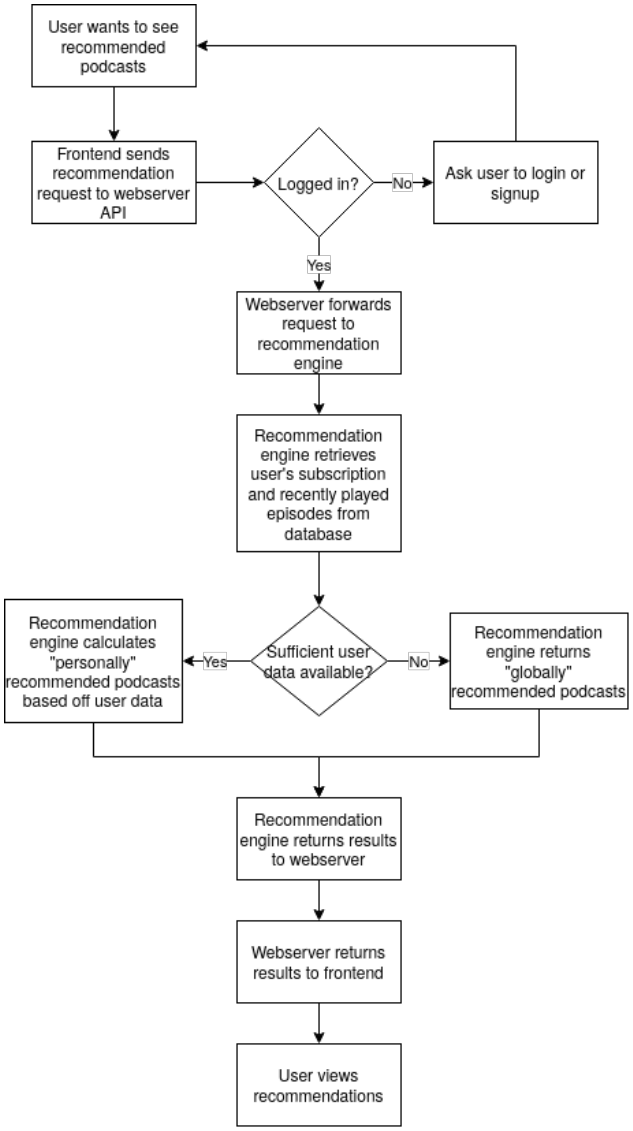
[6] Sticher. https://www.sticher.com, August 2008. Accessed: 2020-10-02.

[7] Player fm. https://www.player.fm, November 2011. Accessed: 2020-10-02.

[8] MongoDB. The most popular database for modern apps — MongoDB, 2020. http://mongodb.com, Last accessed on 2020-10-03.

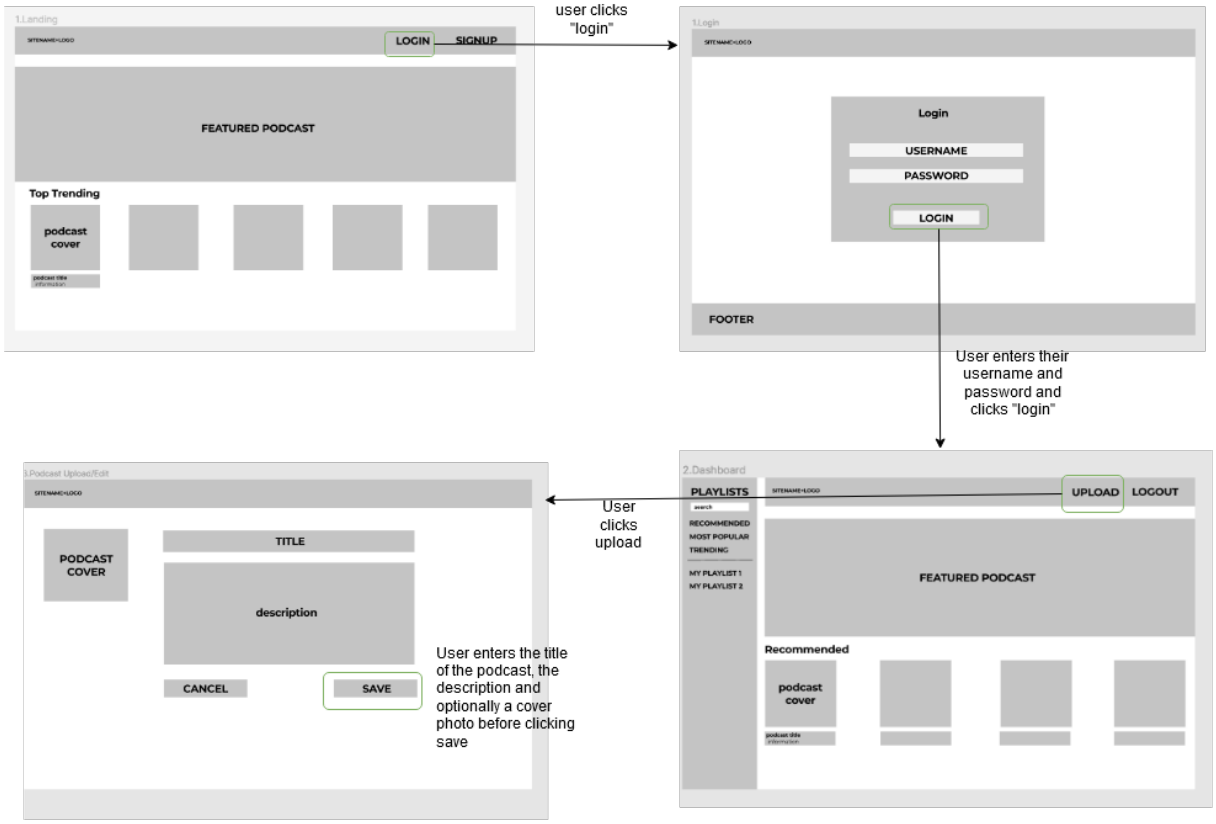
[9] Gavin Whitner. Podcast statistics (2020). https://musicoomph.com/podcast-statistics/, Sep 2020. Accessed: 2020-10-02.

# A Recommendation Engine Flow Diagram

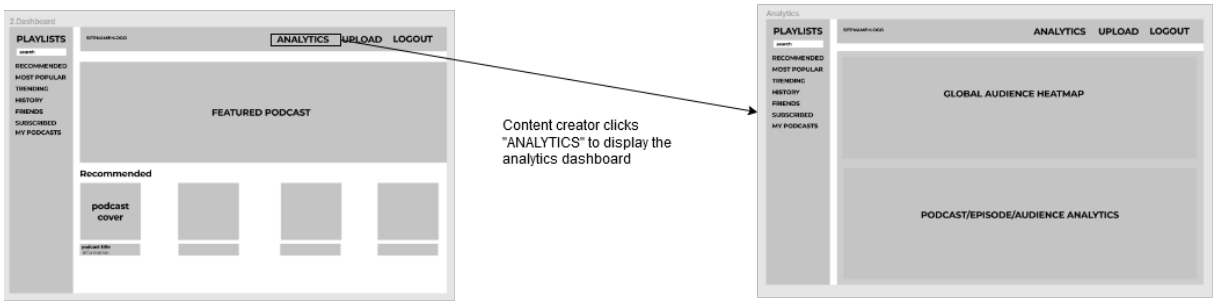


# B Storyboards

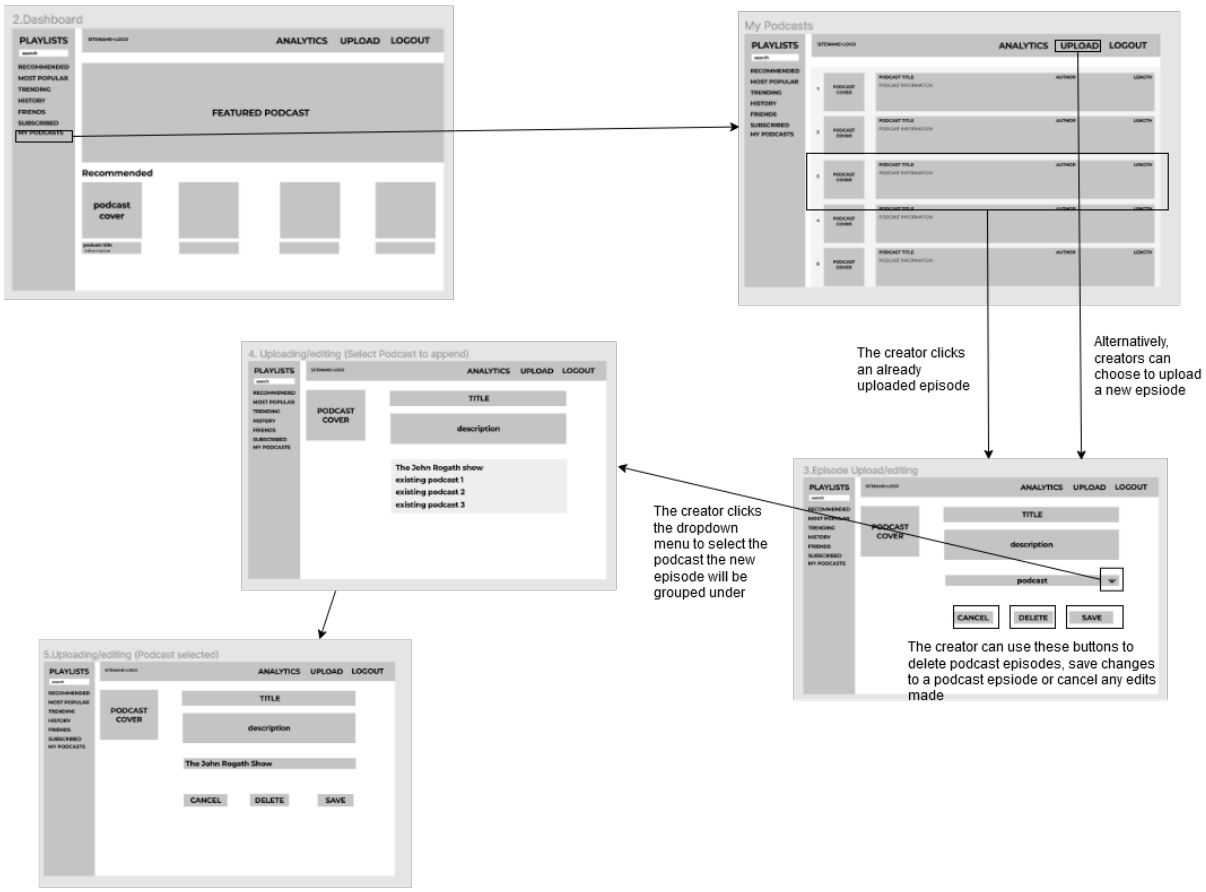
**Content Creator wants to upload a new podast**



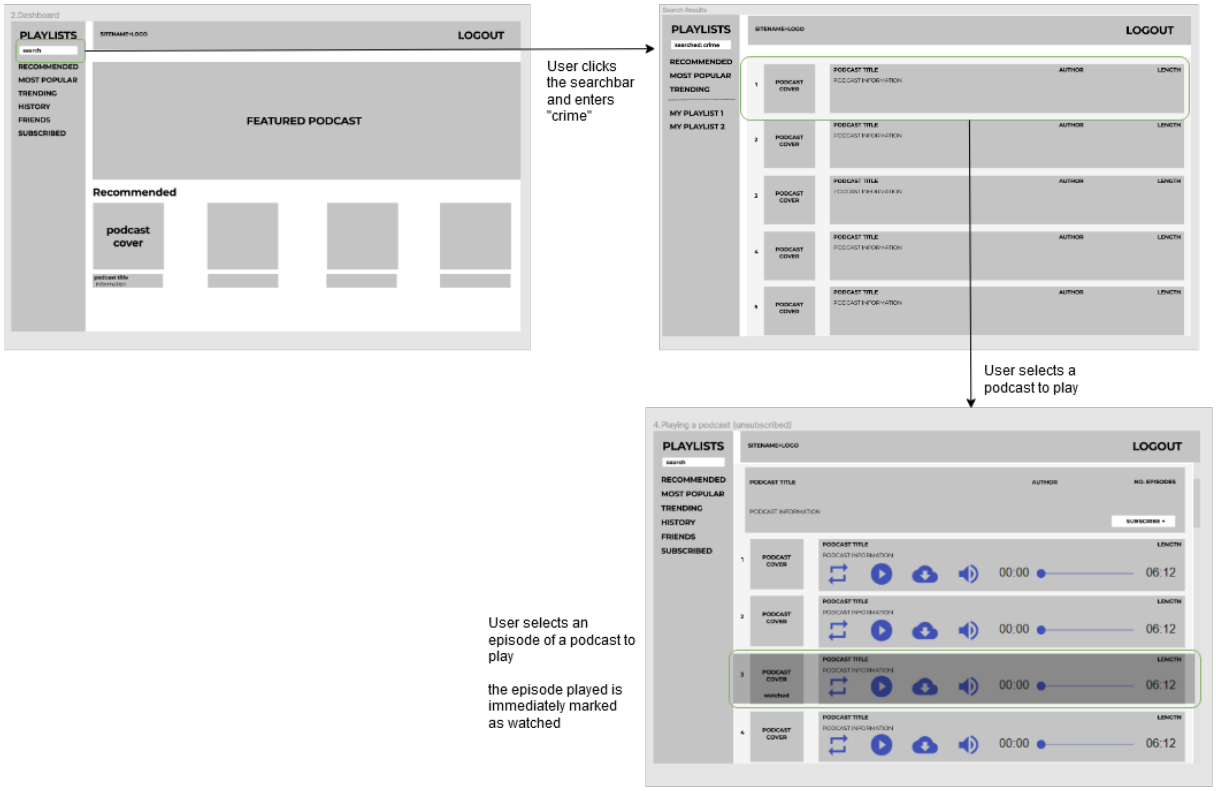

**A content creator wants to view analytics on uploaded podcasts**



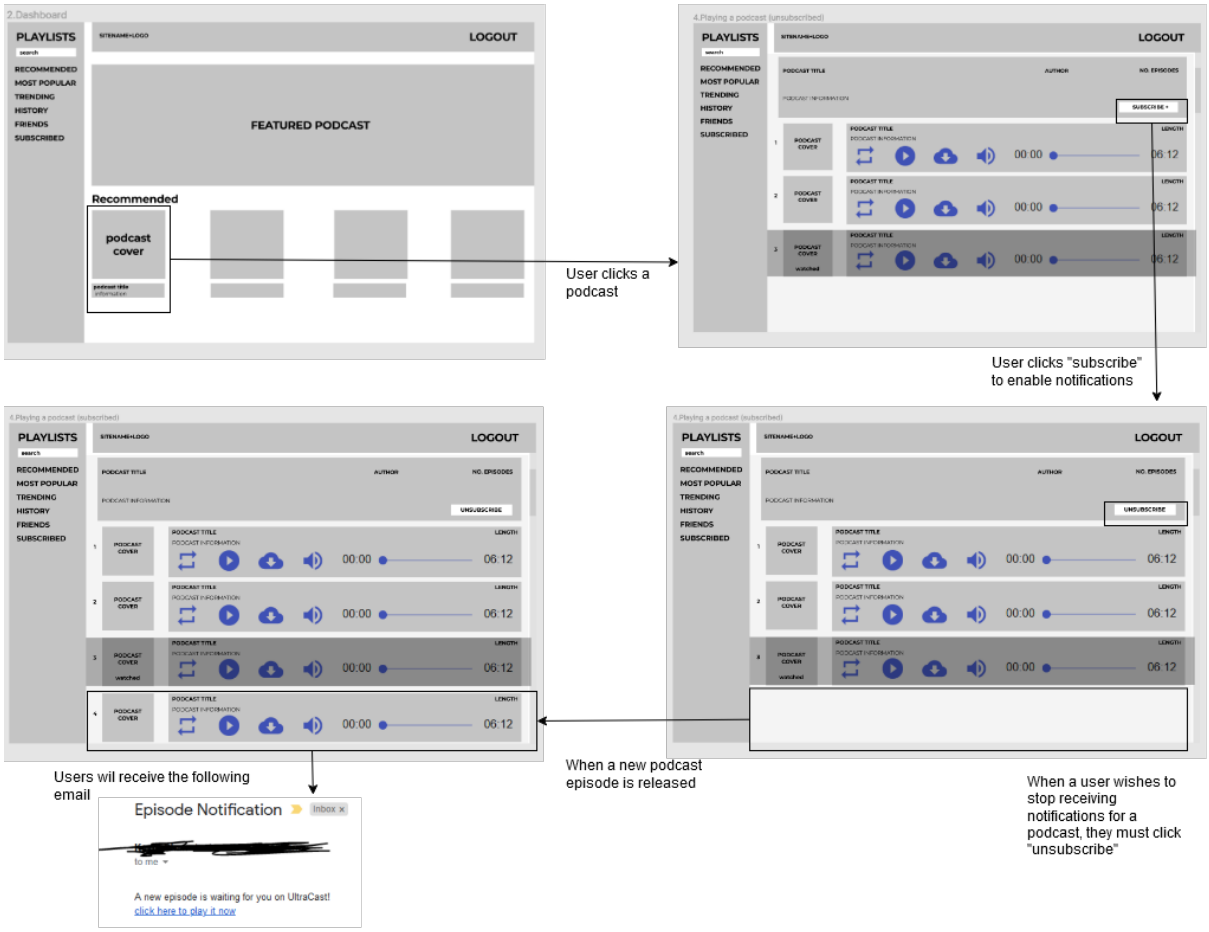
## A content creator wants yo upload/edit/delete a podcast episode



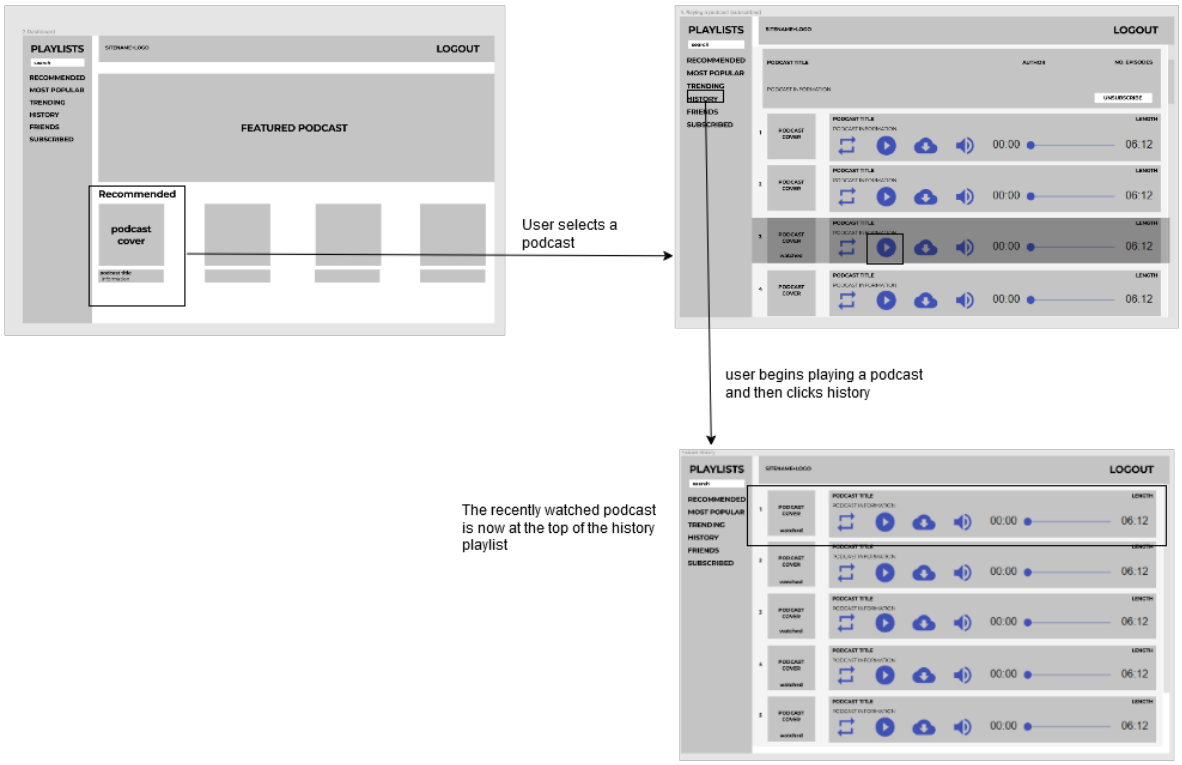

## Existing user wants to find podcasts using a keyword



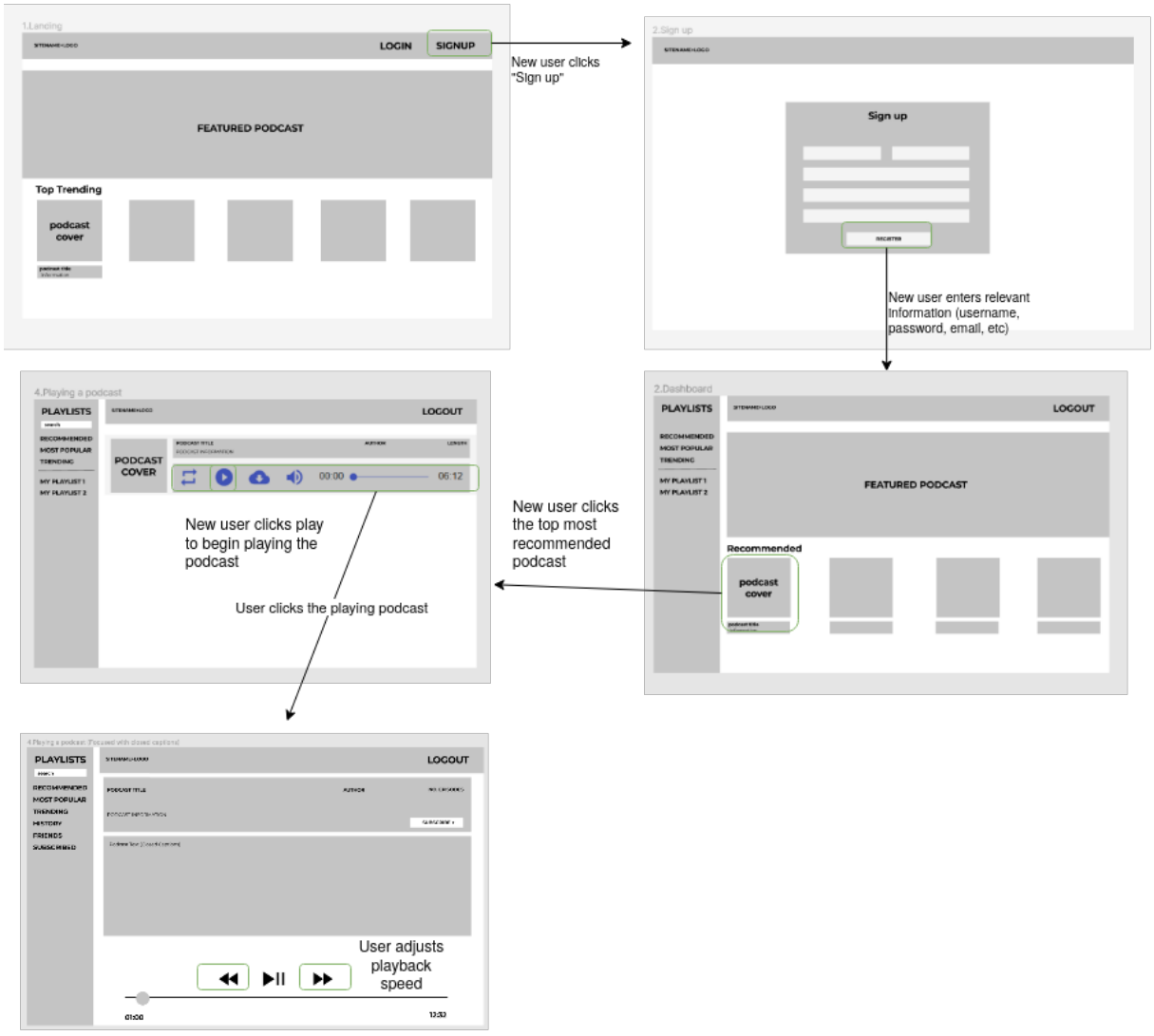
## An existing user wants to subscribe/unsubscribe to podcasts



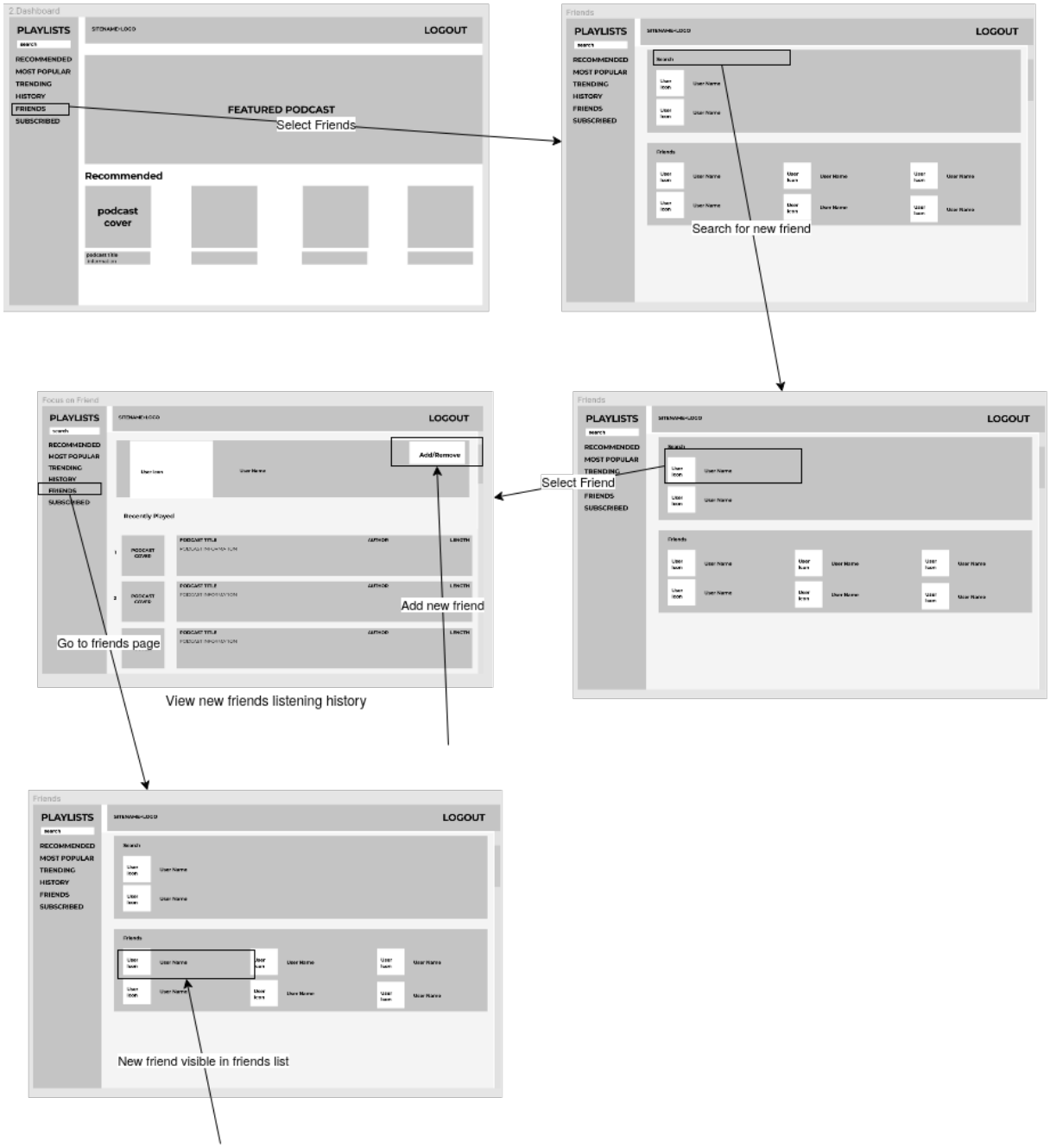

## An existing user wants to replay a recently watched podcast episode



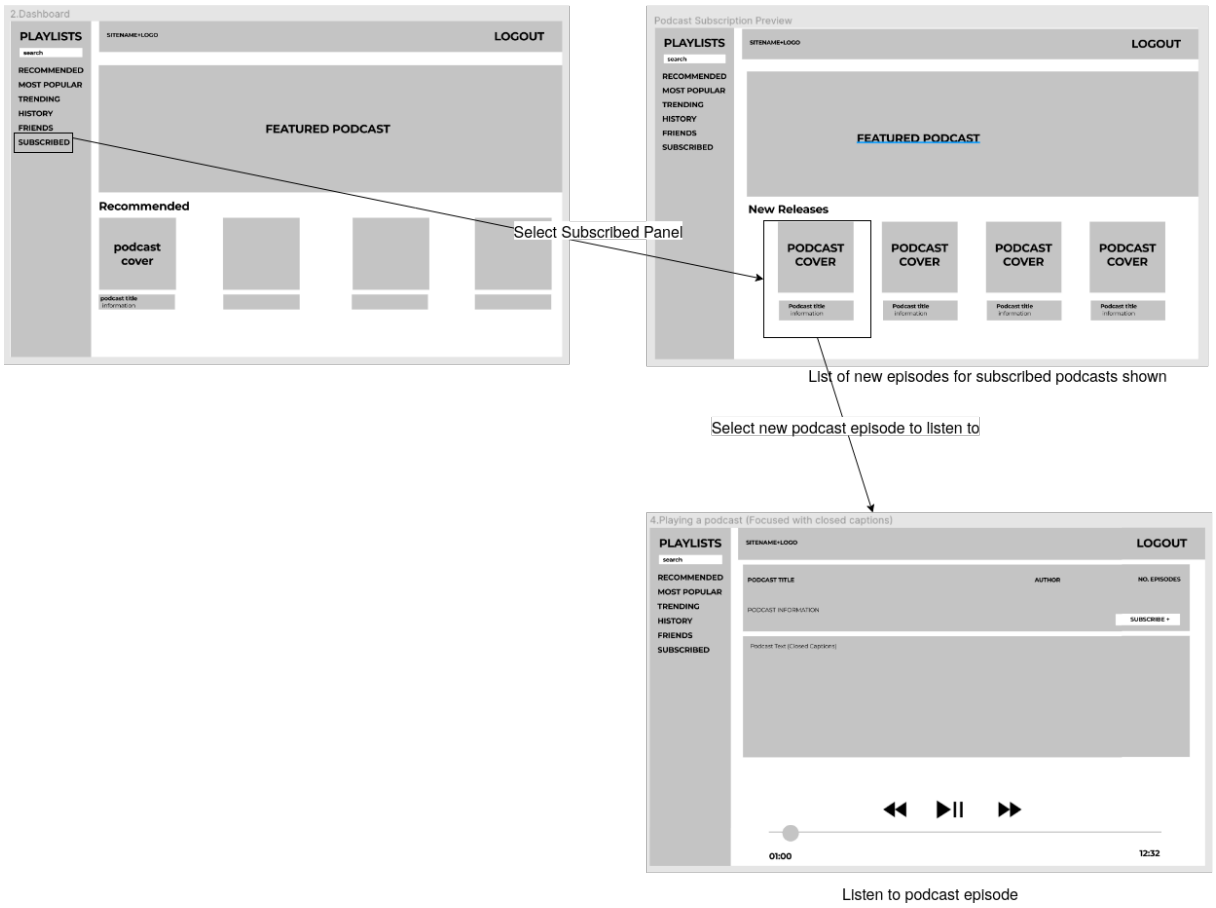
## New user wants to play recommended podcasts



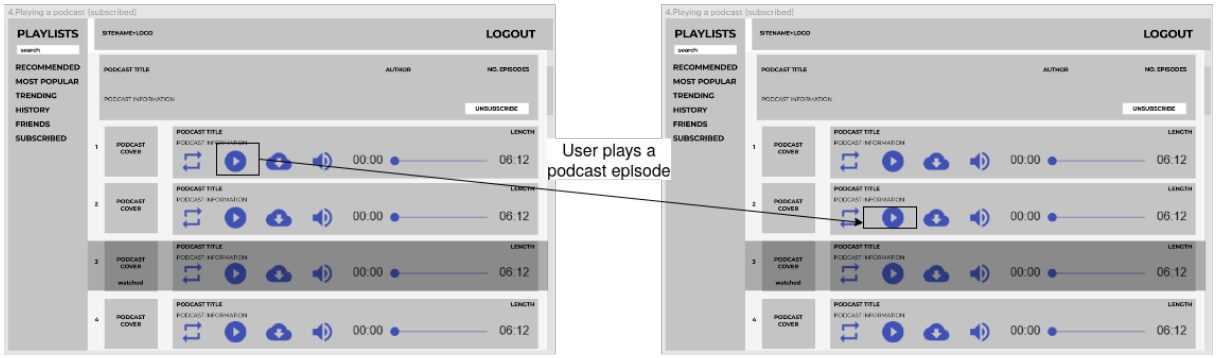
**A user wants to add a new friend and view their listening history**



**A user wants to listen to a new episode of a podcast they subscribed to**




**A user wants to play multiple podcast episodes sequentially**



## A listener wants to make a search, save it as a stream and then view the stream later