VastCast

Software Requirements & Design

Raychel Delaney
Emma Elston
Riley Kohler
Michelle Pearson
Erika Pope

4/03/18

Change History

Version	Summary	Authors	Date
1.0	Initial Draft	All	2/6/18
1.1	Enhanced Project Description and Functional / Non-Functional Requirements	All	2/8/18
1.2	Added Diagrams, Completed Requirements, Started Website, and Started Presentation 1	All	2/15/18
1.3	Added Diagrams, and Finished Presentation 1	All	2/22/18
2.0	Changed High-Level Class diagram to Low-Level, added Sequence Diagrams	Michelle Pearson	3/29/18
2.1	Update Sequence Diagrams and add Final Changes	All	4/3/18

Table of Contents

---

1. Introduction

1.1 Purpose

The purpose of this document is to outline the requirements and specifications for VastCast. VastCast is a user-friendly, android podcast application. This document outlines the general features, functional, non-functional requirements of the application as well as competing podcast applications on the market. The document also contains the UML documentation of specific use case, class, and activity diagrams for the VastCast application development.

1.2 Scope

The application will be built for Android. The application should provide a user-friendly experience, while still addressing all the user's needs. VastCast will utilize RSS feeds to retrieve desirable podcasts for users to download or stream on their devices. VastCast will provide features such as search, download, subscribe, silent skip, add, remove, and play to give the users the ultimate experience.

1.3 Goals

Improve interface by creating a simple, user-friendly design that integrates features with customers in mind. Include, and improve on features provided in other podcast apps. These features include improved searching and syncing, as well as a focus on tracking the location in podcasts and those have been played. Another key goal is to improve setting options for enhancing user personalization.

1.4 Definitions

Discover: The "store" like section of the application allowing users to find new podcast content

Download: When an episode is downloaded, it is stored on your phone's storage space and no longer requires wifi or cellular connection to play.

Episode: A single audio file from a podcast series

Library: User's already downloaded and saved podcasts

Playlist: A pre-made or user-made collection of episodes

Podcast: A podcast is an episodic series of digital audio or video files which a user can download and listen to.

Queue: An ordered list of podcasts to be played next

RSS feed: RSS is a type of web feed which allows users to access updates to online content in a standardized, computer-readable format

Silence Skipping: An algorithm which detects silence in audio and attempts to omit it without affecting surrounding audio.

Streaming: A process which allows a user to start consuming content over a network before its whole file has been downloaded.

Subscription/Follow: A podcast saved by the user to receive future updates. Subscribing to or following podcasts allows for quick access by the user.

2. Project Description

2.1 Overview

The app is focused on the user experience and provides features for discovery, listening, download management, settings, and user-defined preferences. The user can access the different features in the app from the discover screen.

2.2 Discover Features

When the app starts up it will begin on the discover screen. The discover screen includes access to podcasts that have recently been listened to as well as top podcasts on the app, which is designated by ratings. Users also have access to a search functionality from this feature in order to find

podcasts. Users will access settings and account information, such as playlist libraries, from this feature.

2.3 Listening Features

The listening screen is accessed when a user selects a podcast to listen to. This screen provides a user the ability to play/pause the podcast, the buttons for which are located at the bottom of the screen for ease of use on an Android phone. It also includes the options to have variable multi-second time skips (i.e. 15 seconds, 30 seconds, etc) and the option to speed up the podcast. The application also maintains a record of when a podcast has been listened to. If the user has not finished listening, it will keep record of where the user left off. From this screen, users can also save an episode to a playlist in the user's library, or add an episode to the queue.

2.4 Management Features

The management feature can be accessed from any other feature in the applications. The management feature can be used for managing user settings. User setting features include managing the user's account, as well as the option to allow auto downloading of podcasts that they follow, to auto delete podcasts for a playlist after they have been listened too, or to allow autoplay when a podcast episode is opened in the listening feature. The features allow the user to personalize the features of the application for the best user experience. The management features also include management of user-created or user-subscribed playlists and the queue.

2.5 Potential Bonus Features

If time allows the app will include other extra features. These features include Chromecast and/or Amazon Echo support and information from podcast related media, such as linked blogs or social media. A possible discover feature would be the addition of recommended podcasts based on a user's followed podcasts, highest rated episodes, or other listening habits. A possible bonus listening feature would be an algorithm that allows the user to skip extended silence. A possible bonus management

feature would be the option for a user to allow push notifications when a podcast they follow uploads a new episode.

3. Functional Requirements

3.1 Podcasts Player

1. User can play, pause, rewind, fast forward, silent skipping, and save its state.

Priority: High

2. User can see the start time, end time, and current play time of the podcast as well as can skip to a specific time.

Priority: High

3. User can view podcast cover art, bio, attached blog posts, and more from the podcaster.

Priority: Medium

4. User can view next up as well as previously played.

Priority Low

3.2 Search Bar

1. User can use the search bar to find podcasts by podcast and podcastor name.

Priority: High

2. User can use the search bar to find podcasts by podcastor name.

Priority: Medium

3. While typing in the search bar, android autocorrect will predict related podcast names based on existing typed words.

Priority: Low

4. User can use the search bar to find podcasts by categories.

Priority: Low

3.3 Settings Options

1. The settings menu will allow users to view and edit their user information, including changing their login password and associated gmail.

Priority: High

2. The user is able to view information about the app as well as have access to contact the creators.

Priority: Medium

3. The settings menu will allow users to change the theme associated with VastCast. An example includes a night mode.

Priority: Medium

3.4 Playlist Management

1. User can add, remove, and arrange podcasts in Playlist/Queue

Priority: High

2. User can create a new Playlist

Priority: High

3. User can auto download podcasts they follow, auto delete podcasts from a playlist after they have been listened to, and auto play an episode when it is opened

Priority: Medium

3.5 Download Management

1. User can turn on the “Auto-Download” setting to automatically download podcasts from podcasters they are subscribed to when they come out.

Priority: Medium

2. User can turn on the “Auto-Delete” setting to automatically delete finished podcasts.

Priority: Low

4. Non-Functional Requirements

4.1 Background Playback

1. Allows playback when the app is in the background which can be controlled through notifications.

4.2 Add RSS Feeds

1. Allows users to add RSS feeds for a given podcast manually using URLs.

4.3 Syncing

1. Users' podcasts are synced between devices using database information.

4.4 User-Focused Design

1. Design choices in the application aim to provide an intuitive interface that is powerful and simple.
2. Tutorial loads the first time a user launches the app to teach new users how to maximize functionality from VastCast.

5. Competing Applications

Competing Apps	**Syncing**	**Queueing**	**Silence Skipper**	**Linked Media**	**Cost**	**User Settings**	**Usability**
Spotify	No - only tracks on one device	Yes	No	No	Ads	Yes	Lacking
Apple Podcasts*	Yes	Yes	No	Yes	Free	Yes	Good
Google Play	No	No	No	No	Free	Yes *(weak)*	Lacking
Overcast*	Yes	Yes	Yes	Yes	Ads	Yes	Strong
Podbean	Yes	No	Yes	No	Ads	Yes	Weak

* applications only available on iOS
+ 'User Settings' refers to settings beyond expected capabilities (i.e. managing podcast downloads, managing account)

In making comparisons to competing podcast applications, we assumed the applications had basic features (i.e. play/pause functionality, search, account management). Our comparisons instead focused on the key distinctive features in our application.

Spotify, while a strong music player and has a great queueing feature, is lacking in podcasts options - many popular podcasts are not hosted on this application.

Apple Podcasts features syncing, but only allows downloads on wifi. It feature continuous playback and many personalize user settings including enabling auto-delete for played episodes and auto-downloading new or unplayed episodes. It does not feature a silence/pause skipper and has an user-friendly interface, although one not as strong as some other podcast applications. This application is also not available for Android.

Google Play has improved a lot over the years but is still lacking in core functionalities. The application lacks the ability to create personal playlists for podcasts, is missing many popular shows, such as *Serial,* and has a poorly designed search functionality.

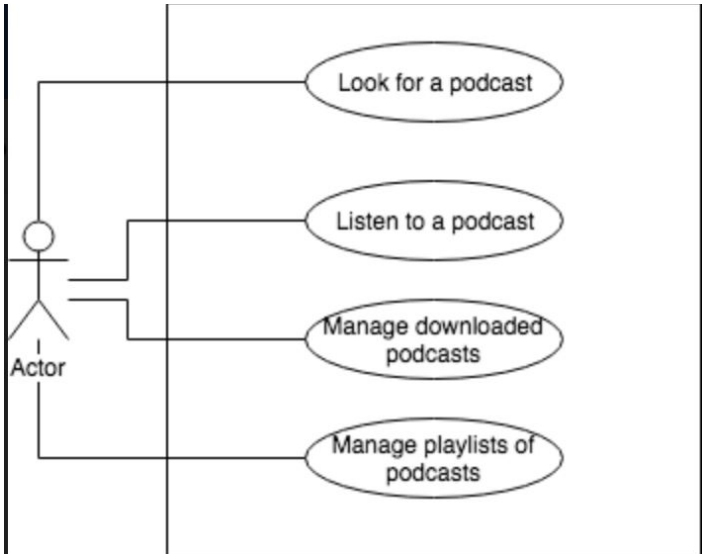
Podbean has a large focus on making hosting podcasts on their application easier. On the user end the application does feature great user settings, including auto-delete and auto-download options; however, the interface can be confusing to navigate.

The strongest podcast application by far is Overcast. It features the best access to personalized user settings, which can be specified for overall listening settings down to an episode-by-episode level. It features options for skipping silences, sleep timers, and has the most user-friendly interface. It's focus on user personalization makes it the one of most popular for users. This application, however, is not available for Android.

A major struggle applications seem to share is access to podcasts

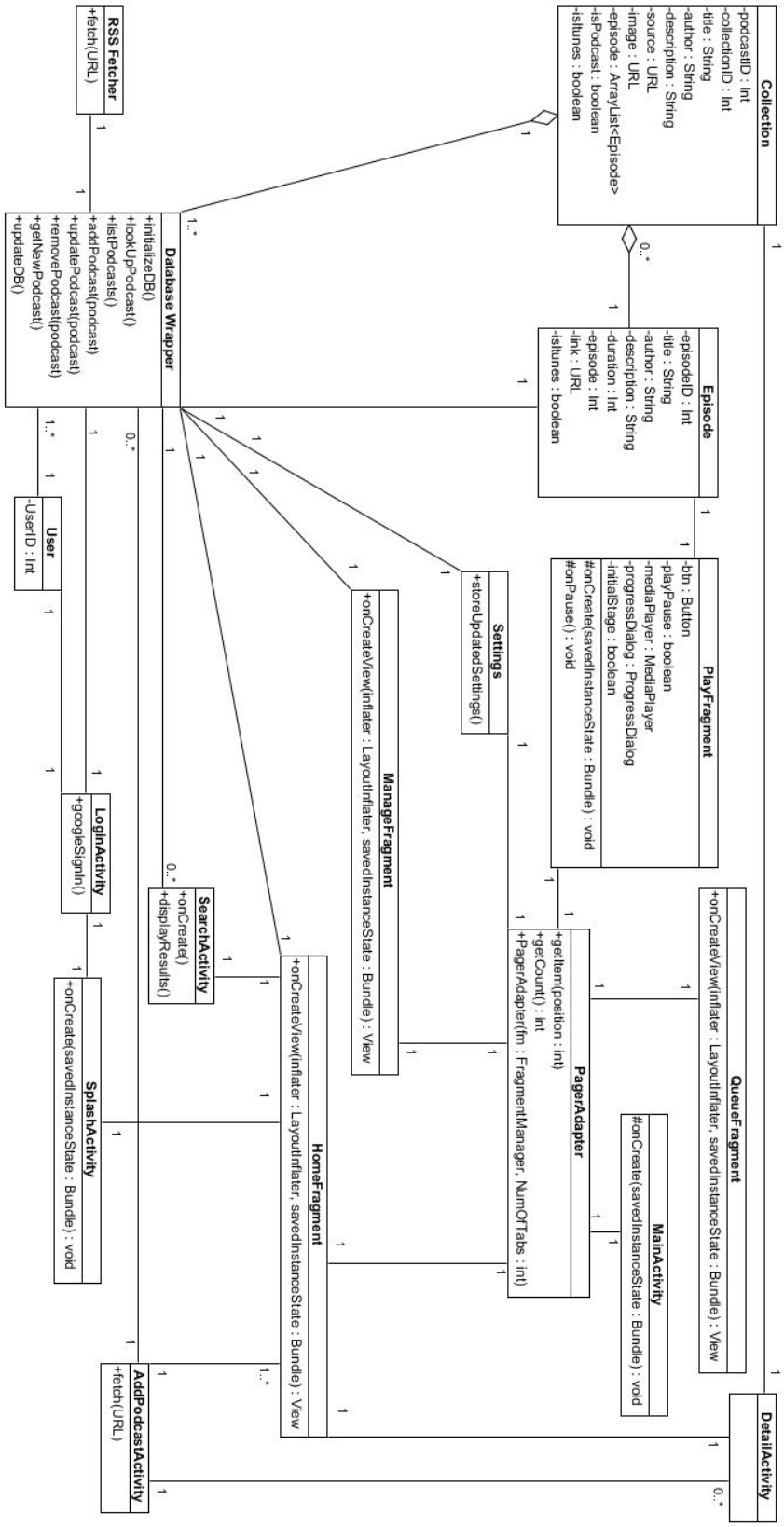
VastCast takes into account the strengths and weaknesses of each application, focusing on the strengths of Overcast specifically, and combines them into a user-oriented application focused on the creating the most personalized listening environment for podcasts.

6. Use Case Diagram



The use cases of our app include searching, playing, and manage the podcast. VastCast is designed to focus around these uses and leave out bloat that doesn't help to improve those use cases.

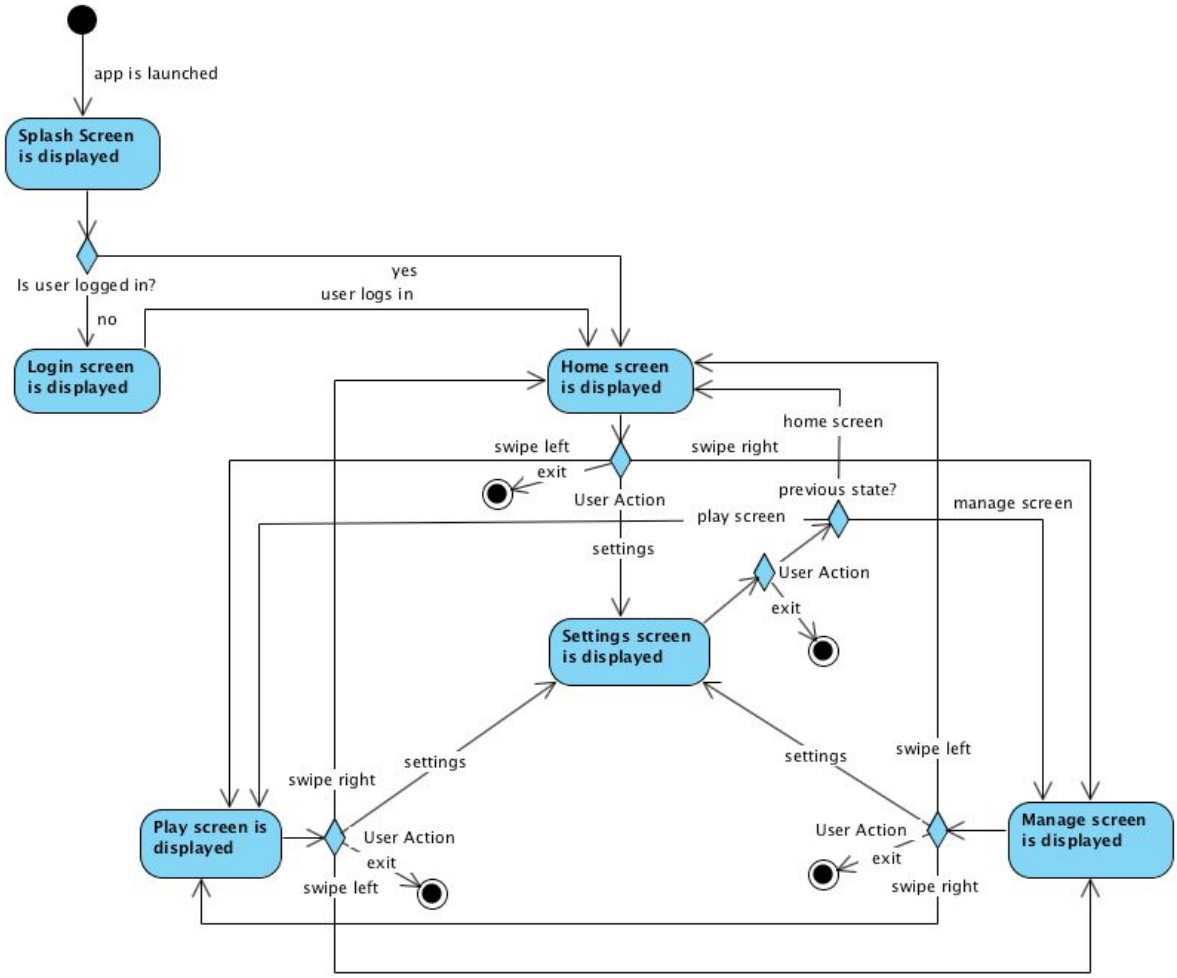
7. Low-Level Class Diagram



The class diagram showcases the different classes that we anticipate using in the development of the VastCast application. Collections hold a set of episodes and have a variety of variables to parse data from the Database Wrapper in a meaningful way. The Database Wrapper stores information about the podcasts and populates its data using the RSS Fetcher class, which pulls the data via URLs. The User class stores a UserID that will be used to identify the user in the database when storing data about their downloaded and played podcasts. The PlayFragment, Settings, ManageFragment, QueueFragment, and HomeFragment classes are each tied to the PagerAdapter class which is tied to the Main Activity class. This design is because the pages in our application will all be fragments, which require a main activity and a pager adapter to connect the fragments together. These fragments all have OnCreateView functions. The DetailActivity class works with the HomeFragment, AddPodcastActivity, and Collection classes to display details about podcasts as they are added to collections. The SearchActivity connects to the HomeFragment and DatabaseWrapper to allow for searching among podcasts. It also contains a function that returns the search results requested by the user. The AddPodcastActivity calls fetch using an RSS URL to add new podcasts to the database. Finally, the SplashActivity is the home page for our app, followed by the LoginActivity that utilizes Google Sign In.

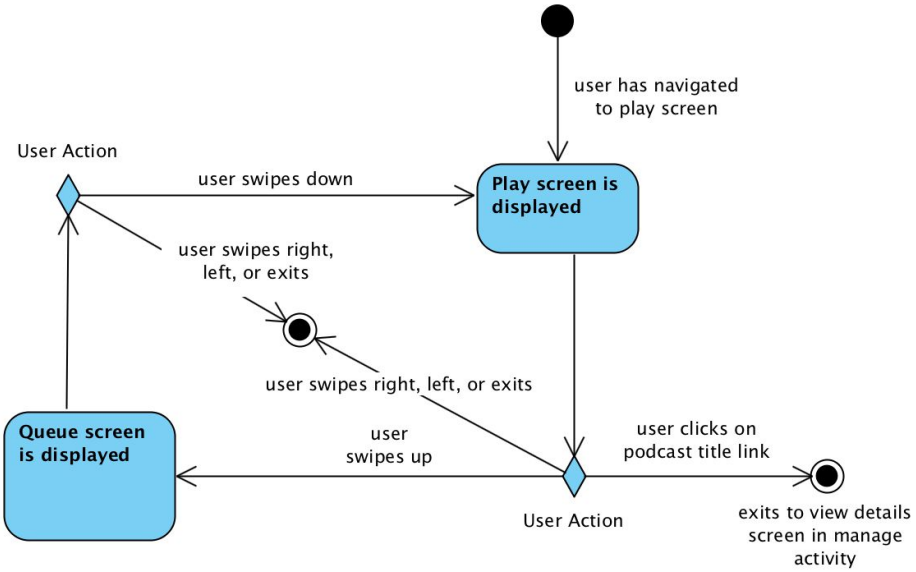
8. Activity Diagrams

8.1 Overview Activity Diagram



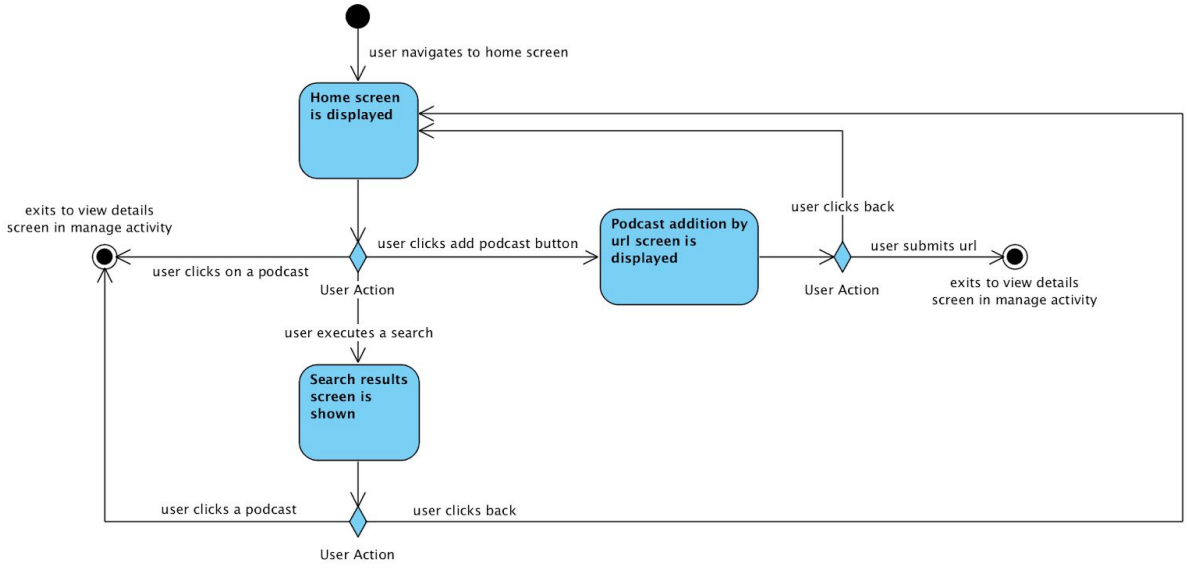
This diagram walks through the overarching views of the app. VastCast shows a splash screen at startup and if no login data is saved it directs the user to a login screen. After the login check, the home screen is displayed and the user enters a circular structure whereby they can access any screen by swiping from the current screen. The diagram also depicts how the user can access the settings screen or exit the application from any view.

8.2 Play Activity Diagram



This diagram depicts the user activities from the play screen. The user can swipe up to view the queue screen, click on a podcast link to view details about the podcast, or swipe left or right to access a different view.

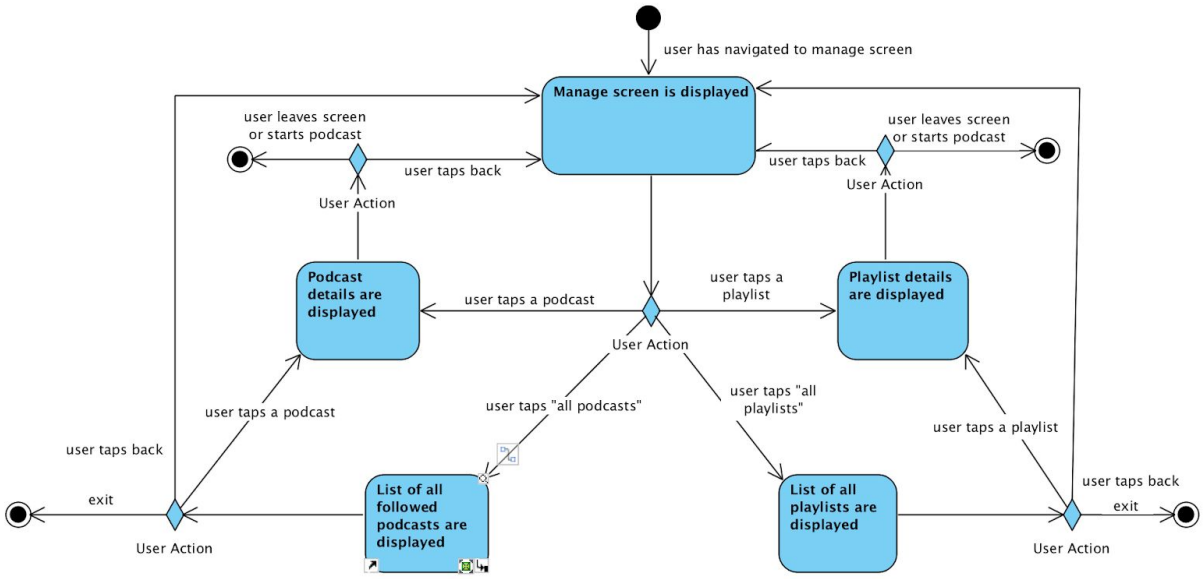
8.3 Home Activity Diagram



From the home screen, the user can search for a podcast, click on a podcast in a displayed category to view its details and episodes, or add a podcast by url. If the user executes a search, the user can either select a podcast from the search, which will exit to

the view details screen, or terminate the search by clicking back. If the user chooses to add a podcast from url, a podcast addition box will pop up and the user can submit their url. If successful, the app will populate a view details page for the new podcast.

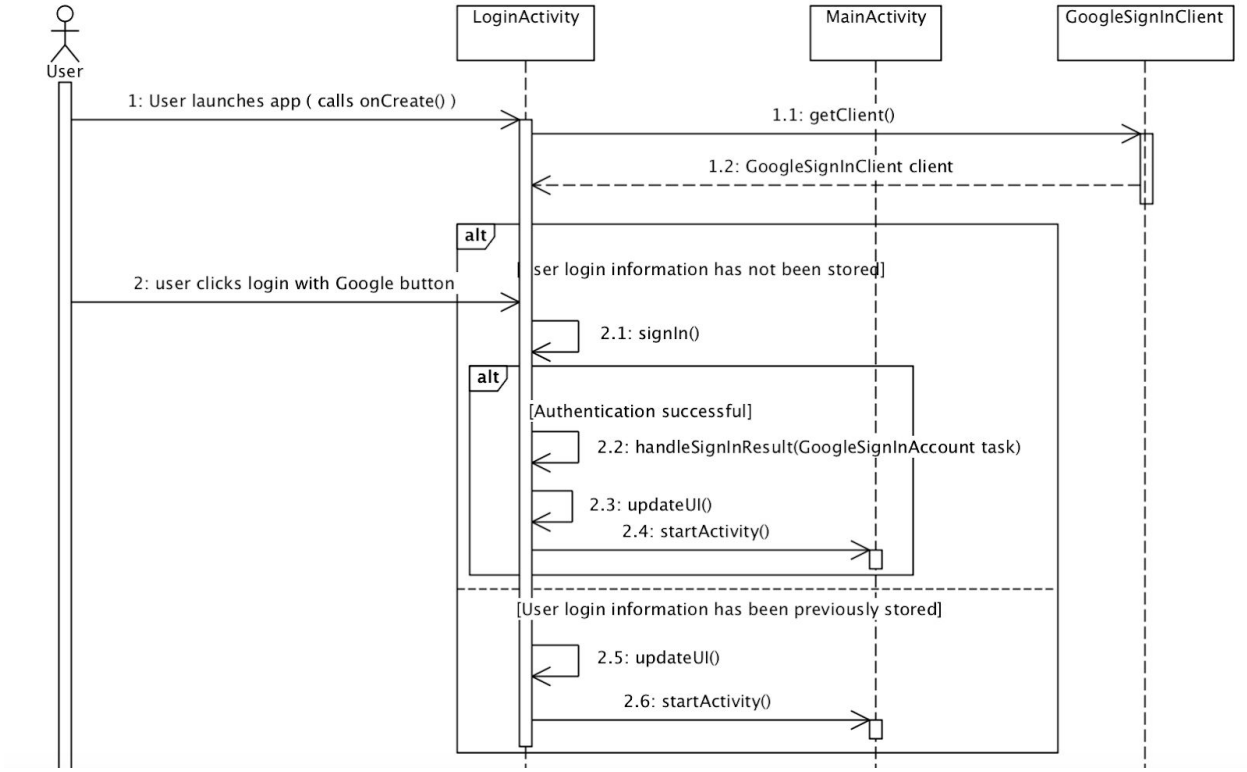
8.4 Manage Activity Diagram



The manage screen has two sections: a list of podcasts and a list of playlists. Each section has a button that the user can press to expand the section (either “all podcasts” or “all playlists”). The user can either tap a podcast from the initial view or choose to view more. If a specific podcast or playlist is tapped, the details for that collection are displayed. If the user taps “all podcasts” or “all playlists”, that section will expand and the user can then select a specific podcast or playlist which will then display details.

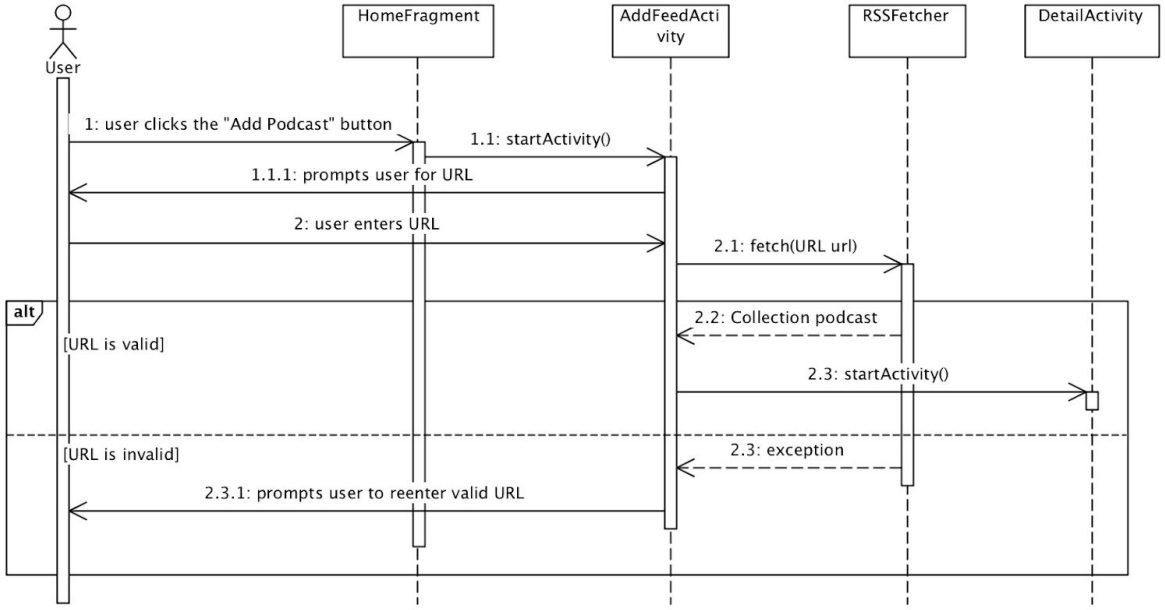
9. Sequence Diagrams

9.1 Login Sequence Diagram



When the app is launched, it always creates LoginActivity first. LoginActivity utilizes Google's database to request user's email addresses using the GoogleSignInClient. If user information is found, the MainActivity is immediately launched using the updateUI function and the user never sees the LoginActivity screen. This should be true for all cases except the first use of the app or when the user has intentionally logged out. If user information is not found in the database, the LoginActivity screen will be displayed and the user can click the "Log in with Google" button, at which point the system will attempt to authenticate the user with their Google login. If this is successful, MainActivity will be launched.

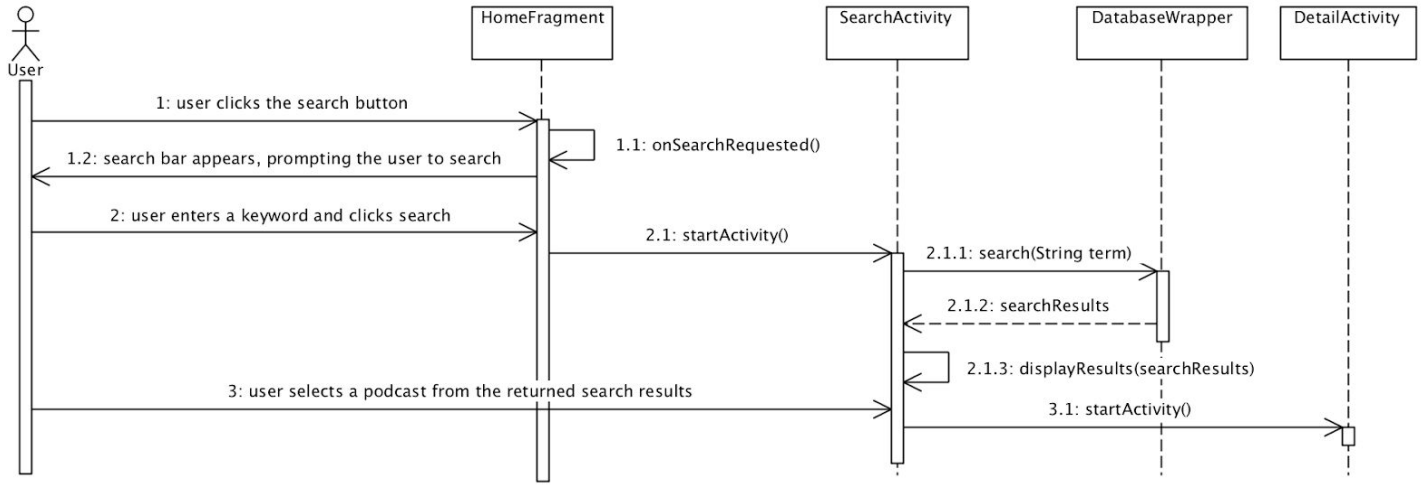
9.2 Add Podcast Sequence Diagram



NOTE: For the sake of simplicity, we have omitted everything leading up to Fragment pages in this and in all subsequent diagrams.

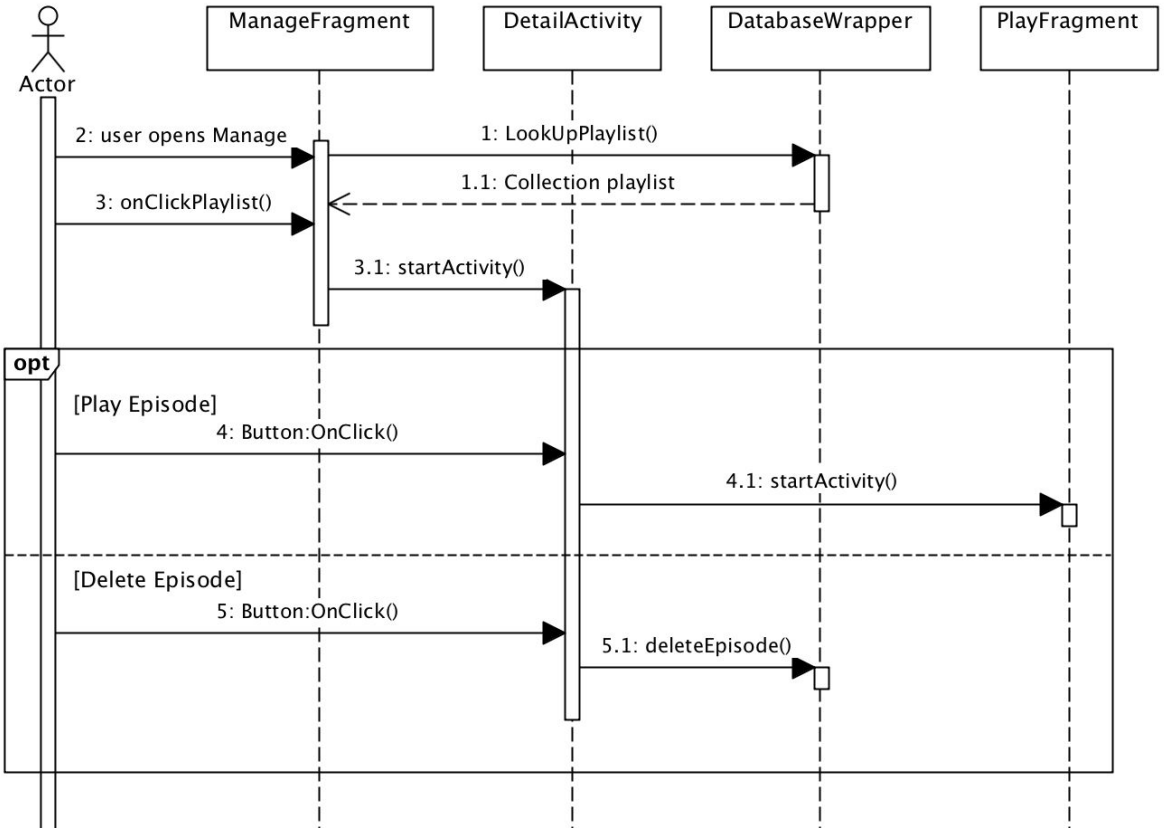
To add a podcast, the user starts on the HomeFragment page and clicks the “Add Podcast” button. This brings up a new AddFeedActivity page where the user can enter an RSS URL. Once the URL has been submitted by the user, the AddFeedActivity calls fetch() on the RSSFetcher class. This returns either a Collection holding the podcast information or an exception. If the URL is valid and a proper Collection is returned, the DetailActivity page is launched with all of the podcast information. From that page, users can play an episode, add episode to queue, subscribe, mark as played, download an episode/podcast, or add an episode to a playlist.

9.3 Search Sequence Diagram



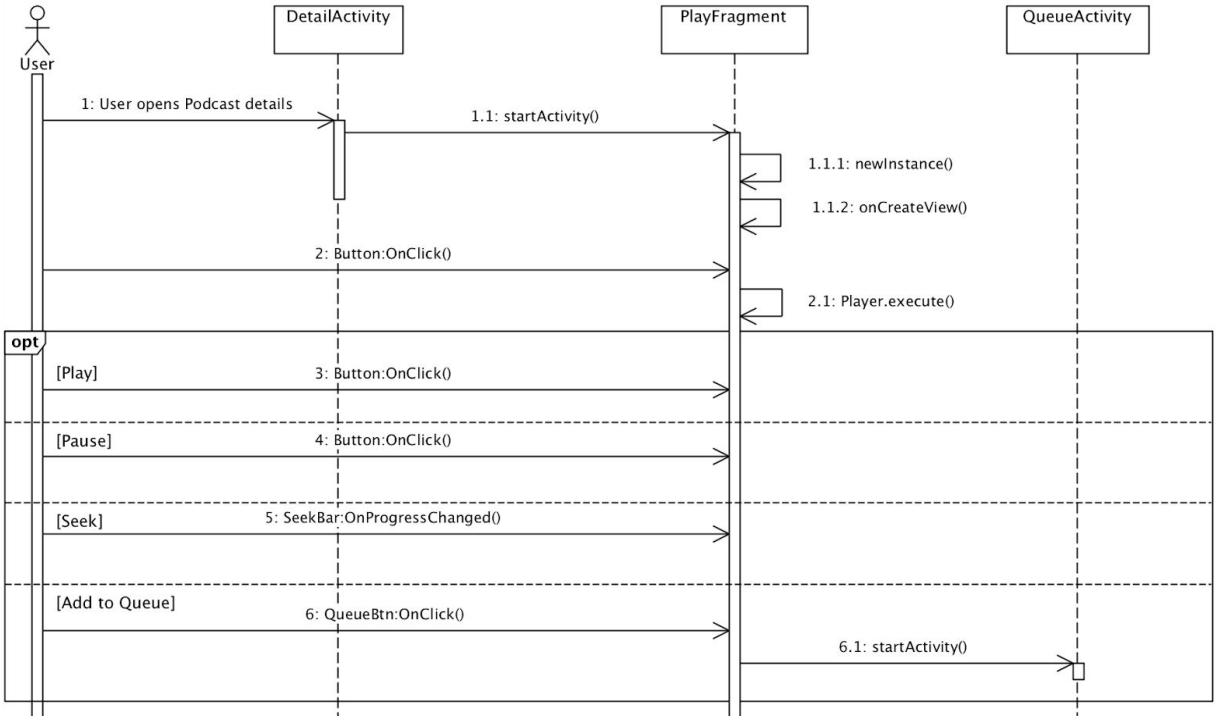
A search button will be located at the top of the HomeFragment. When that button is clicked, the search bar and keyboard appear so that the user can enter a search term. The user will press “Enter” on the keyboard or “search” next to the search bar once they have entered their desired search criteria, and the SearchActivity will be started. This activity calls a search to the database, which returns the results of the search. These results are then passed in to displayResults in the SearchActivity, which displays the results visually for the user. From there, the user is able to select any podcast from the results and it will be displayed in the DetailActivity.

9.4 Manage Playlist Sequence Diagram



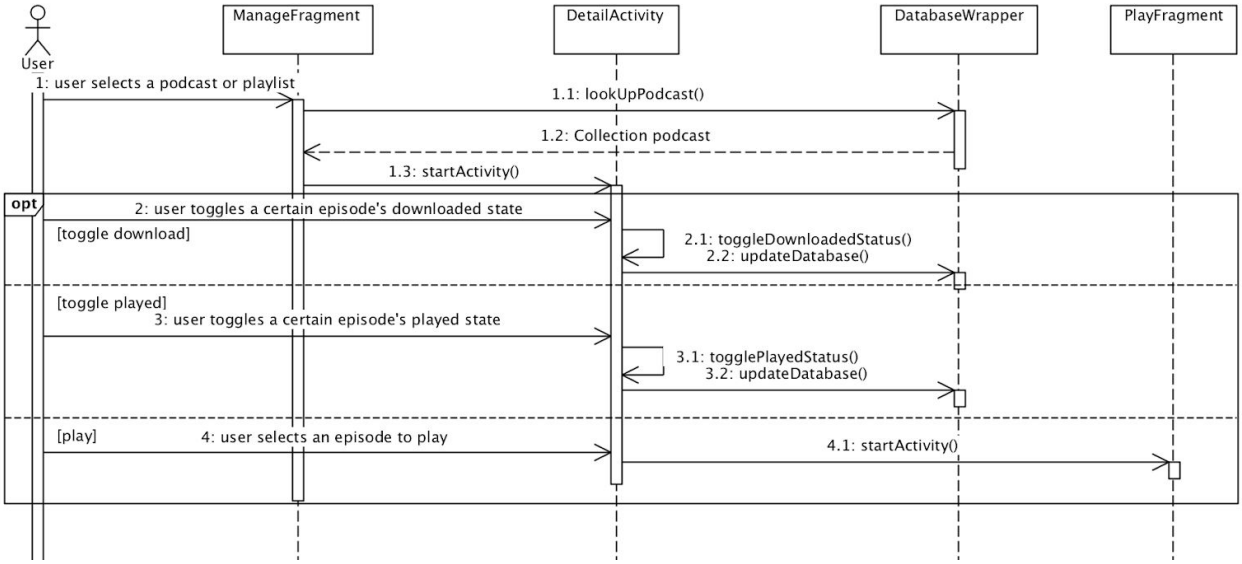
The user will open the ManageFragment through the MainActivity. The user will be able to view a list the list of their playlists that have been pulled from the database as a collection. The user will then select the playlist they wish to view which will then open that playlist in the DetailActivity. The chosen playlist will display the episodes in the selected playlist. Users will be able to select an episode from their playlist to play, which will open the PlayFragment. Alternatively, users will be able to click a button that will delete an episode from their playlist.

9.5 Listen to Podcast Sequence Diagram



Playing a podcast will always begin from the DetailActivity. A user will select an episode from the displayed podcast details, which will launch the MainActivity and pass the necessary podcast and episode information to the new activity. A field in the intent passed to the main activity signals that the PlayFragment is to be used. When the PlayFragment opens, a new instance of the fragment is created. OnCreateView() handles the setup of different features in the UI including the play/pause button, the seek bar for moving through the podcast, and the 'add to queue' button. On the opening of the Play screen the UI will also load the album art in an asynchronous task. The user will select the play button, which will run Player.execute() and initialize the media player appropriately based on the provided information from the DetailActivity. This initialization includes preparing the media player and setting the appropriate information for the seek bar. Once the media player is prepared, the audio will start playing and the user will be able to interact with the audio. The user may play and pause the audio using the audio control button. The user may also seek through the podcast using the seek bar. The user may also choose to add the podcast to their queue, which will open the QueueActivity.

9.6 Manage Podcast Download Sequence Diagram



From the ManageFragment, the user can pick any podcast or playlist. Once a podcast or playlist is selected, it will be fetched from the database and returned as a Collection. From there, podcasts and playlists are treated the same way. DetailActivity is launched, displaying the list of episodes. The user then has a few options: they can toggle the download or play state of an episode, which will be reflected in a database update, or they can select a podcast to play. If they choose to play a podcast, that podcast will be passed to the PlayFragment and the podcast will begin playing.