GitHub Username: KevinTheGray

Podcast Master

Description

Podcast Master is an app that attempts to give a user full control of their podcast listening experience. All podcasts will need to be fully downloaded before they can be listened to, which will allow for easy offline experiences. Maintaining downloaded podcasts will be a snap, showing you how much space they take up and quick deletion of unwanted podcast episodes. This solves the problem I have with many podcast apps, that focus more on streaming, and where maintenance of downloaded episodes can be a pain.

Intended User

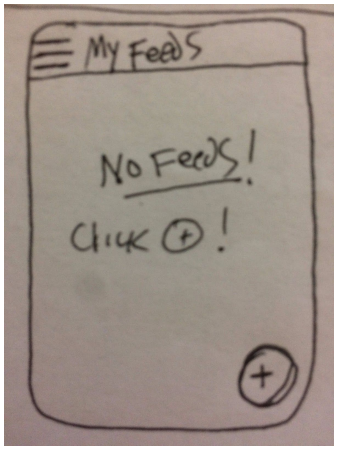
The intended user is anyone who enjoys listening to podcasts and prefers to download their episodes.

Features

- No streaming, so less accidental data usage. Unless you use your data to download.
- Bring in the podcasts you want by entering the URL of the RSS feed.
- Detailed/searchable/filterable list of downloaded podcasts that can be easily maintained.
- Typical media player experience, (play, pause, skip)
- All feeds are stored in a content provider and there are ways to refresh them when desired.
- Lists of episodes and the ability to download any that are desired.

User Interface Mocks

My Feeds - Empty State (Mobile)

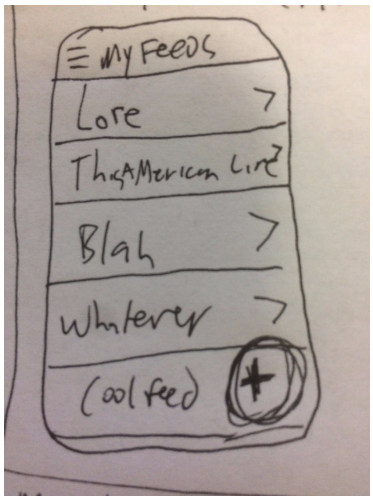


This is the first screen a user will see on their first use. There is an empty view that lets the user know that they have not added any feeds, and can do so by clicking the floating action button. On the top left is a menu drawer where they will be able see a menu that will allow them to select between showing their feed of added podcasts and their downloaded episodes.

My Feeds - Adding Feed State (Mobile)

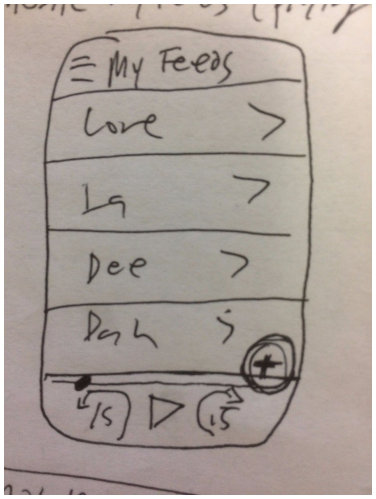
This screen is shown when a user clicks the FAB on the My Feeds screen. The keyboard will appear and a dialog asking for them to enter the URL for their desired RSS feed. After entering, a spinner will appear, and once the feed download is complete, a message will be displayed in the dialog indicating success or failure.

My Feeds - Feeds Added (Mobile)



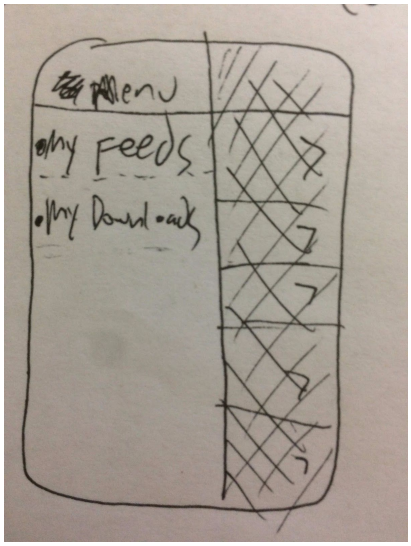
This is what a user will see as they fill up their app with feeds. The FAB will remain there so they can continue adding more. The added feeds will be displayed as cells in a RecyclerView, and a user will be able to click on them to go into a section of the app that will display details of that feed.

My Feeds - Episode Playing (Mobile)



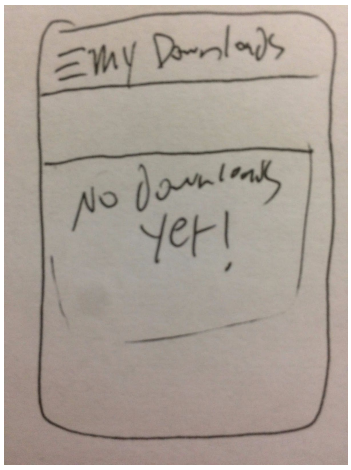
This is an example of when an episode is playing, as you can see in the controls at the bottom. These controls should be visible whenever there is something to control. The user can pause/play/seek 15 forward/seek 15 backward/ and seek with the slider. There will also be details of what is playing, probably at the top of the control.

My Feeds - Open Drawer (Mobile)



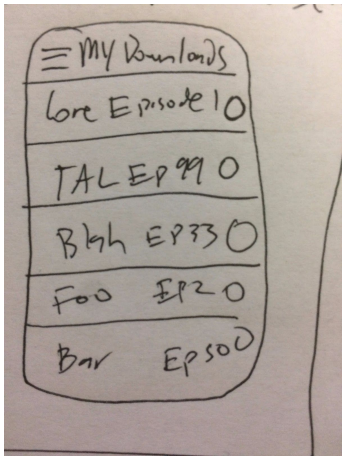
This is an example of the drawer open on the main activity. The drawer will open, and the user will be able to toggle between viewing their feeds and viewing their downloads. They can tap out of the menu by clicking outside of the menu.

My Downloads - Empty (Mobile)



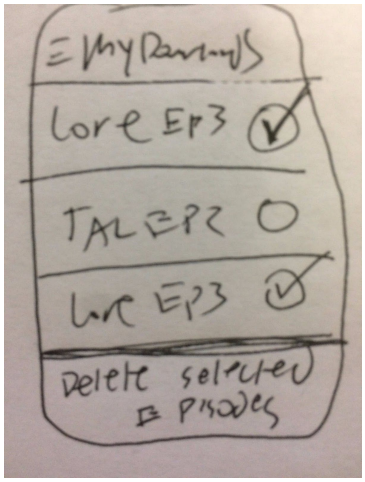
This is similar to the empty feeds screen. If the user enters this screen and has no downloads, they will be shown an empty state message to let them know why the screen is blank.

My Downloads - Episodes Downloaded (Mobile)



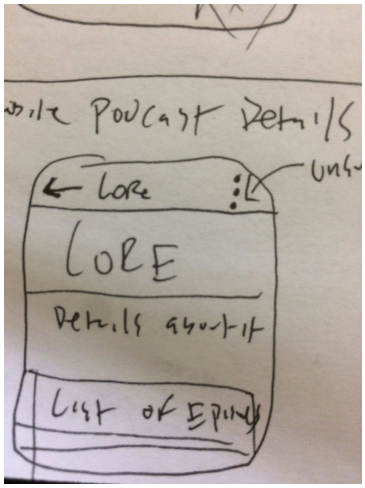
When the user has downloaded episodes, they will be able to see the list of downloads. And each one will have a little circle next to them that can be selected. This will be used to select multiple and delete.

My Downloads - Episodes Selected (Mobile)



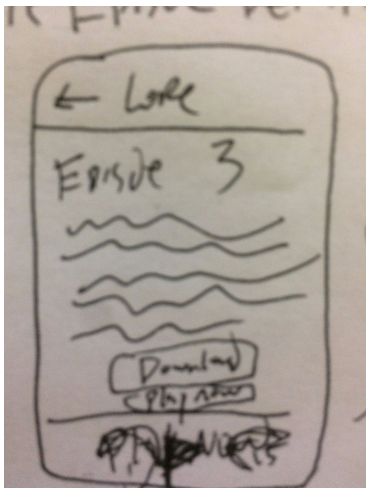
This is the same screen as above, but some items are selected. As the user selects by pressing the circles on the right, a button on the bottom will appear allowing the user to delete all selected downloads. Tapping a cell will bring the user to details about that episode.

Podcast Details (Mobile)



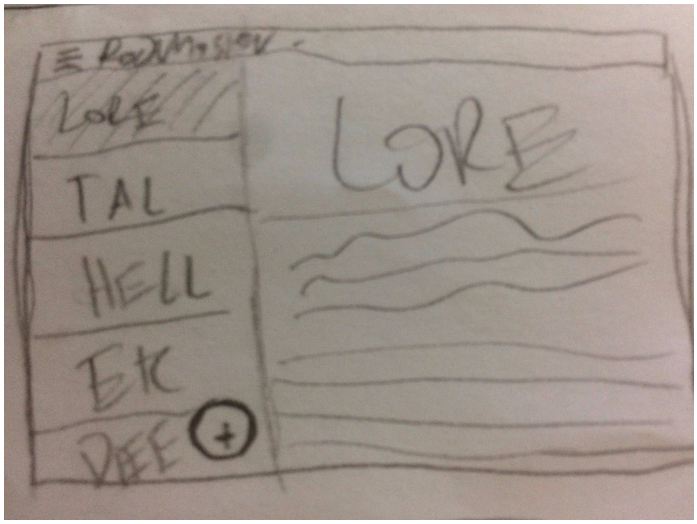
This screen shows the details for a feed, (a podcast), when it is selected from the My Feeds screen. There will be details about the Podcast, and a list of all the episodes that the user can click into. The screen will be scrollable, probably just one big recyclerview. On the top left will be a menu with options to delete the feed, refresh, etc. Also there will be a back button to go back to My Feeds.

Episode Details (Mobile)



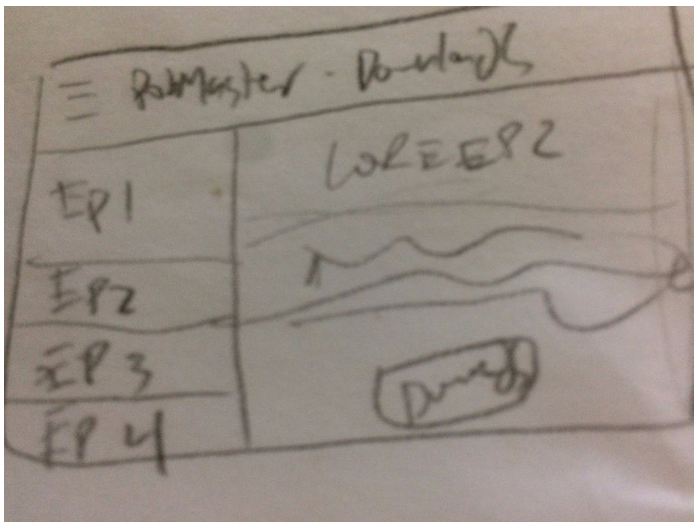
This screen shows the details of an episode when it is selected from the list of episodes or from my downloads. There will be details about it, and buttons to Download/Play Now depending on the state it is currently in. They can also go back to where they came from.

My Feeds with selected Feed (Tablet)



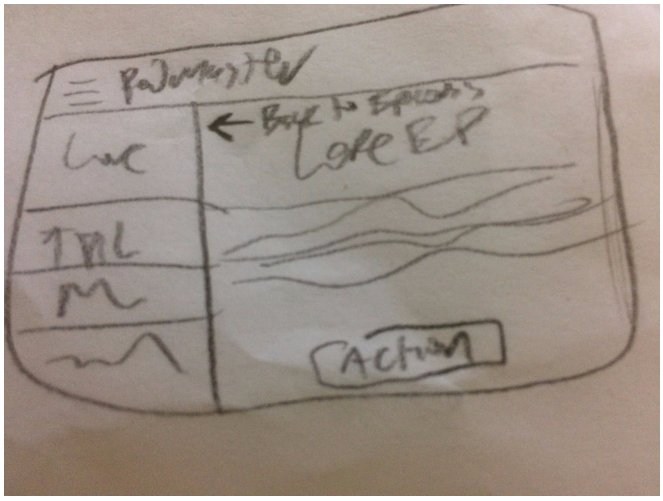
This screen shows a tablet layout for the My Feeds section. On the left will be the list of feeds, and when one is selected, the detail will be shown to the right. The user will be able to select their episodes from the list on the right, and see details, as in the previous mobile section of the episode details.

My Downloads with selected Downloads (Tablet)



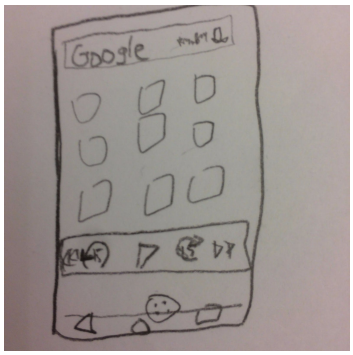
This screen shows the tablet layout for My Downloads. On the left is a list of downloaded episodes, and on the right is the detail of the episode. Similarly as in the above drawings for mobile layout, they can play now or download from here.

My Feeds with selected episode from list (Tablet)



This is similar to the above image, in that the user can see the episode details, and perform and action to either play now or download. However, as it was selected from the feeds list, there will be a back button for the user to return from here to the podcast details.

Widget



This is what the widget will look like on a mobile screen. The user will be able to Play, seek and go back and forward from the home screen.

Key Considerations

How will your app handle data persistence?

I will create by own Content Provider, most likely using a library such as DBFlow. There will be two tables, Podcasts and Episodes. When a feed is requested, I will parse and get necessary information and create a Podcast model with it. Each episode in the feed will be parsed and an

Episode model will be created and added to the database. It will have a reference to the id of the Podcast model, and in that way I will have a sort of relationship between the models. Each episode can be downloaded for listening, and that will be a row in the database which contains the location of the download. If a user requests that a feed be deleted, all episodes with a relationship will be deleted, and so will any downloaded files associated with an Episode.

Describe any corner cases in the UX.

Loading while adding feeds - When a user adds a feed, there will need to be some sort of blocker as the content is loaded, which may take a while depending on the size of the feed. I will handle this by having a spinner popup and block touching on the app until there is a result, either success or failure.

Selecting Feed/Downloads when they are already open - If a user already has “My Feeds” open, and then selects it from the menu, what should happen? I think I will just ignore it, but if it feels more natural, I might restore them back to the original state. I don’t believe this will be the case though.

Empty state on my downloads - When there is an empty state on the downloads, and there is also an empty state on the feeds, the user first needs to download some feeds before they can make downloads, I will need to find a way to make this clear to the user. I could possibly link them back to the My Feeds section if it feels right.

How will removing a feed affect downloads - If a user downloads a feed and then downloads episodes from that feed to listen to, what happens if they delete the feed? I will show a dialog before removing the feed that will explicitly state that all related downloads will be removed as well.

Empty state on a tablet - It’s going to look very empty. I could potentially have one large empty view for this state to get the user started.

Widget when there are no downloaded episodes to play - If there are no episodes downloaded to play, the widget will have a sort of empty state that will link the user to the My Feeds sections.

What if a user closes the app during a download? I think the download will have to cancel, and when it starts up I ask them if they want to resume downloading.

What if a user adds a feed they already have? I’ll allow it, it will be the same as refreshing. Any data that needs to be persisted will be.

Describe any libraries you'll be using and share your reasoning for including them.

Picasso - I will use this to load images where required, and utilize the caching of the images. The primary usage will be loading artwork for podcasts where possible.

DBFlow - I will use this to simplify the creation and persistence of data in my app. I will have a few tables as listed in the data persistence question, and this should help simplify it.

OKHttp - Used to simplify making HTTP requests, which I will need to do to get the podcast feeds.

MaterialDialogs - Used to make fancy dialogs that I need for entering an RSS feed.

Describe how you will implement Google Play Services.

Google Analytics - I will add Google Analytics and use it to see how my users are using my app. Are they downloading a lot of episodes, or only a few? How do they move through the app. Are they using the widget? Etc.

Google Ad Mobs - When a user wishes to download an episode of a podcast, they will be shown a full screen advertisement. And when they close it, the download begins.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Get the project setup, just a clean slate with all the libraries/support libraries/services I need, and also take this time to set up the play services I will need ad mobs and analytics.

- Set up Android Studio project, supporting Lollipop and up only
- Set up design and support libraries in gradle
- Add desired third party libraries to gradle
- Set up Google Analytics developer console
- Set up Google Ad Mobs developer console
- Add Google Services to gradle

Task 2: Get analytics working properly

I like to make sure everything is installed and configured properly first, so first up is analytics.

- Add button to screen that sends analytics event
- Insure analytics are being received

Task 3: Get admobs working properly

Same reason as before, want to get this configured and working properly as soon as possible

- Add button to screen that displays full screen ad
- Insure full screen ad is coming through as intended.

Task 4: Get widget working properly and displayed on home screen.

I had a lot of trouble getting the widget working for Stock Hawk, so I want to get this done and working properly as soon as possible. As I build in the rest of the app and playback functionality, I want to be able to test this at all times and insure it is working as intended. Trying to hook it up after the main work is done was really a pain in Stock Hawk.

- Get widget service working properly and able to get one showing on home screen
- Build proper UI for widget with all buttons on it
- Make sure buttons are communicating back to app and we can differentiate between buttons presses when they are pressed.

Task 5: Get media playback working well

I want to get a sample file working first, be able to play, pause, seek. Also it needs to be available from any activity.

- Add podcast length media file to assets
- Create UI for playback
- Create an activity that always has the playback control in it, and it can be hidden or visible and the layout plays well with that.
- Make it possible to show and hide that control
- Get playback working with the controller. Make it possible to play, pause and seek.
- Do whatever service things we need to do to get the sound playing even when app is in the background
- Make sure the widget is working with it too.

Task 5.5: Create menu drawer for My Feeds and My Downloads

Want to get this and have it designed, ready for use whenever I need it.

- Add drawer to MainActivity
- Add callback for when My Feeds is pressed
- Add callback for when My Downloads is pressed
- Make sure it is designed for use in the rest of app. Build out subclasses and whatever else as necessary.

Task 6: Set up content storage structure

Get the database and models set up so that I can start filling out the views with their data.

- Make requests for RSS feed, and insure responses are coming back.
- Parse XML or whatever format it is in, and create models for them
- Make it so that I can easily create a callback that returns all the data I need per feed, or an error if it all goes to heck.
- Set up content providers for Podcasts and Episodes, and give them the values they need to be persisted.
- Make a feed request and store the podcast and episodes.
- Test that they are persisting as intended by logging

Task 7: Create and implement UI for My Feeds

Start building the UI

- Get two feeds into the database
- Add FAB to proper position in view.
- Create XML UI for the cells of the recycler view that will hold the list of feeds.
- Create empty view XML
- Add recycler view to view
- Add logic for empty state if no feeds are available
- Set up logic to load cells of the database
- Create modal to enter RSS feed using MaterialDialog
- Make request from MaterialDialog and make sure it's sending out
- Show spinner while requests are doing work
- Show success message if success, error message if failure.
- When dialog dismissed, refresh if necessary.

Task 8: Create and implement UI for Podcast Details

Continue building the UI

- Create view for it
- When tapped from My Feeds, transition to empty Podcast Details view
- Make sure we can go back from there to where we were before
- Pass data in Intent or whatever to let the view know what Podcast needs to be loaded and insure it is being received.
- Load required models when view is transitioned to using content loader
- Add recycler view
- Create views for any detail cells, and for the list of cells that will transition to the episodes
- Populate cells with data that was loaded before
- Add ability to delete feed if desired with menu button

Task 9: Create and implement UI for Episode Details

Continue building the UI, and start getting downloads in

- Create view for it, and as above, pass all the stuff we need into it so we can populate the view.
- Add download button to view
- Get the thing downloading, keeping in mind to build around the fact of the download failing. Or the app closing, things like that
- Save file, save file location in database for episode
- Boom, we have downloads.
- Make sure we can play the downloads and control them from the player, and the player is all working as intended and all that.

Task 10: Create and implement UI for My Downloads Details

Continue building the UI, and managing downloads

- Create view for it, and transition to it from menu
- Add recycler view, button for deleting, and build the cell for the downloads
- Make sure we can transition to the episode details when a cell is selected. Should use the same episode details as above.
- Create empty view for downloads. Make sure it hides and shows appropriately.
- With the views, allow the option to select as many as possible, and the delete button should show and hide
- Do logic for deleting downloads

Task 11: Test across all devices, clean up any bugs and fix any oddities that will reveal themselves

Task 12: Submit