Description
Intended User
Features
User Interface Mocks
  MainActivity
  EpisodesList/AudioBooksChapterList
  Episode Detail/AudioBookDetail
  My Podcasts
  Settings/Preference
  Profile
Key Considerations
    How will your app handle data persistence?
    Describe any edge or corner cases in the UX.
    Describe any libraries you'll be using and share your reasoning for including them.
    Describe how you will implement Google Play Services or other external services.
Next Steps: Required Tasks
  Task 1: Project Setup
  Task 2: Implement UI for Each Activity and Fragment
  Task 3: Implement Retrofit Service Calls
  Task 4: Implement Room Database
  Task 5: Implement View Models /Live Data
  Task 6: Implement Mechanism to Synchronize Data with the Server

GitHub Username: maganahalli

Puffin Podcaster

Description

Are you looking to subscribe and listen to digital recordings on internet? Are you looking to find your favorite audio books or tv episodes you love?

Puffin Podcaster gives you instant access to millions of digital recordings available for public on internet for downloading to device and explore and listen to on your own time ad free.

Puffin Podcaster is available on your Android device. Explore right podcast, subscribe and listen to anywhere, anytime.

With Puffin Podcaster, you'll be able to:

- Access millions of Digital recordings or podcasts on internet to public.
- Display podcasts by popularity and recommended.
- Explore podcasts by category like Arts, Technology etc.
- View detailed information about podcasts.
- Subscribe, Play and listen to podcasts.

Intended User

Anyone who is looking for convenient way to get the niche content they want on internet through RSS feed.

Features

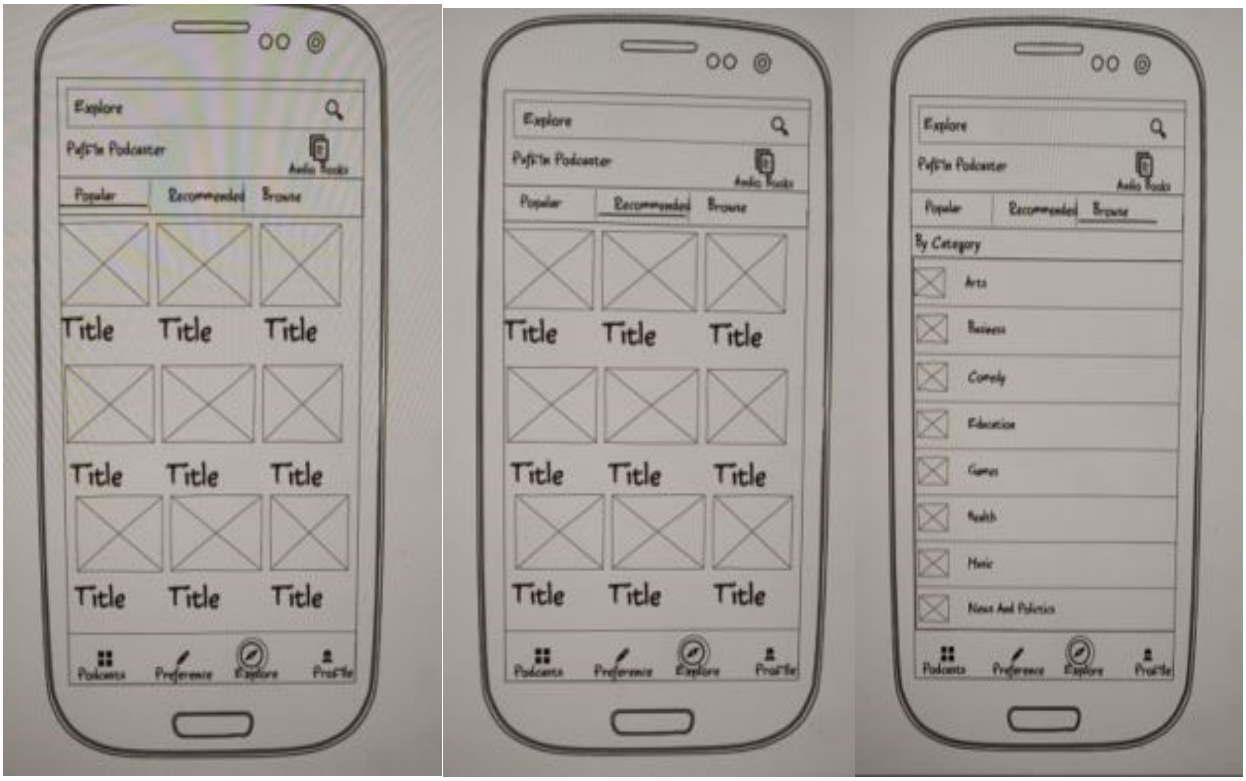
List the main features of your app. For example:

- Organizes podcasts feed so that they can be easily searched.
- Clean and ad free app.
- Subscribe, Play and Listen Podcasts
- Create favorite lists and play later lists.
- Can specify frequency of update of podcasts update.
- Can Specify region so that explore region specific podcasts.
- Saves podcast information in a Room database
- Images are downloaded and displayed using Glide.
- Uses "Listen Notes API" service to obtain podcasts data.

User Interface Mocks

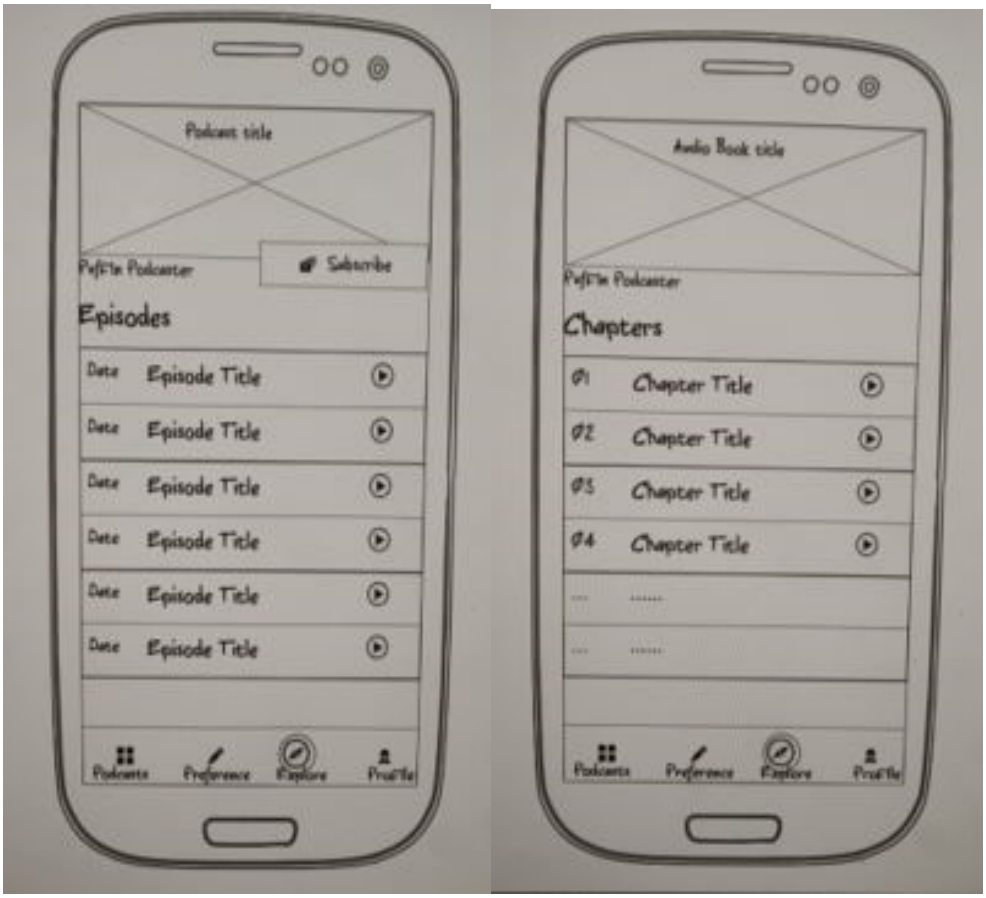
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

MainActivity



Main Activity is landing view for app or when app is launched this view will be showed with tab “Popular” selected and Explore highlighted in Bottom navigation menu.
User can explore available podcasts in popular, recommended and browse by category options.
Free form Search functionality for podcasts available in tool bar via search icon.
A separate Icon is available for Audio books podcast search just below toolbar.
When Audio Book icon is clicked , a new Activity will be opened with Audio Books titles and images ..
With Bottom view menu, user can access main functionality of app from any view .
When user Clicks on Any Tiles in this activity, will be navigated to Episodes List or Audio Books Chapter list view depend on user selection.

EpisodesList/AudioBooksChapterList

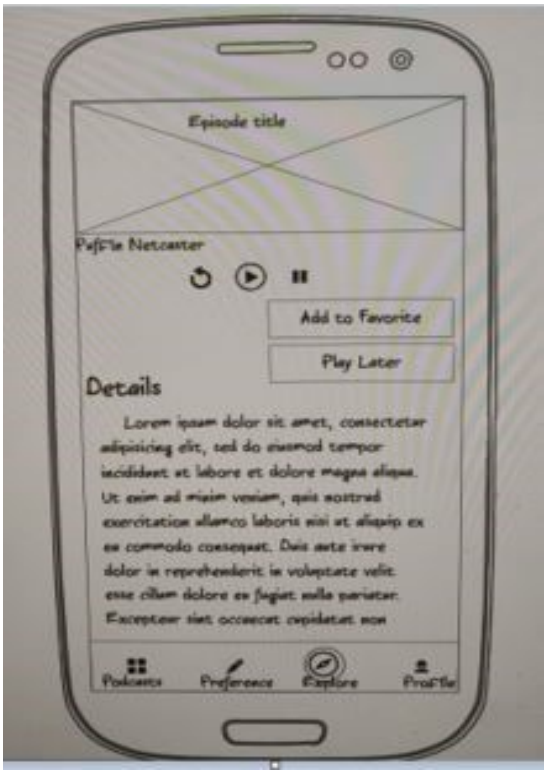


This Activity Displays podcast or Audiobook Background Image with title.
In case of favorite shows or tv episodes, there is option to subscribe to series.

All the episodes or book chapters are displayed as per service response from server.
when user clicks on Play button of list item, user will be taken to Episode or Audio Book detail view.

With Bottom view menu, user can access main functionality of app from this view.

Episode Detail/AudioBookDetail



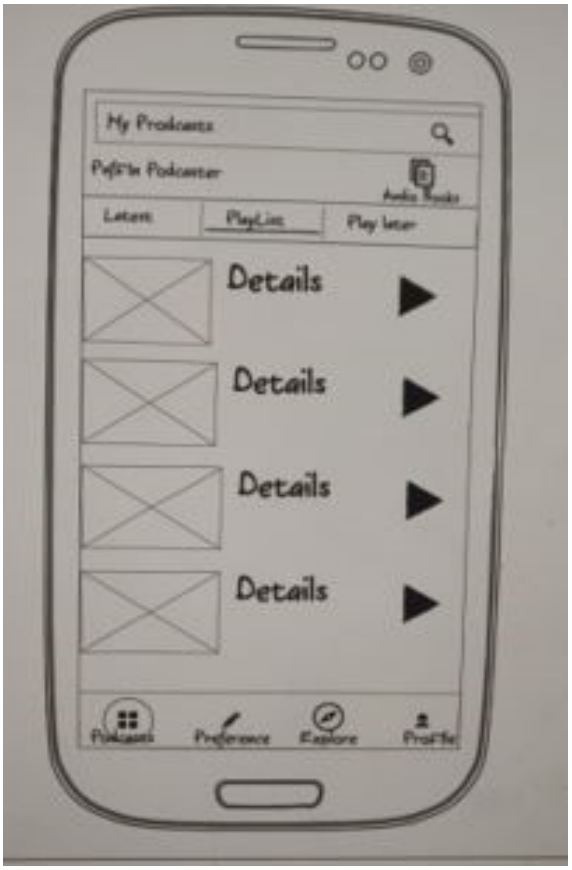
This Activity shows long description of episode or chapter in scrolling and readable format. Audio or Video will be played using Android Exo player. Player will have the option of pause, play and replay functionality.

It has “Add to Favorite/Play List ” And “Play Later” buttons which helps to add episode or audio to Play list or Play later list.

.

With Bottom view menu, user can access main functionality of app from this view.

My Podcasts



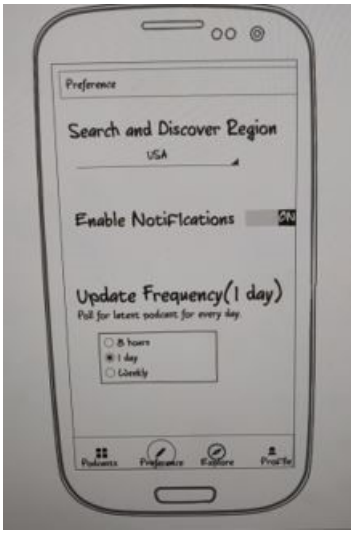
The My Podcasts screen is invoked from Bottom Navigation menu and allows user to customize the podcasts into favorite or Playlist and play later list.

when user clicks on Play button of list item, user will be taken to Episode or Audio Book detail view.

If user has not added any episode or audio to favorite or Play later lists, then empty list along with message will be displayed on the screen to user

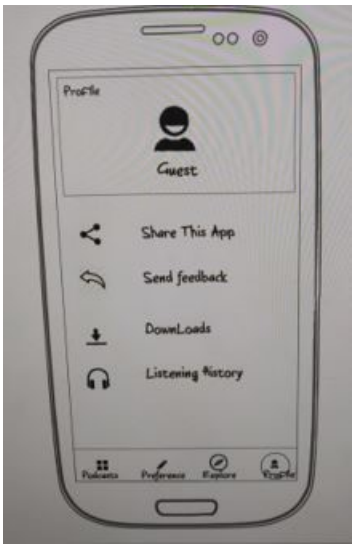
With Bottom view menu, user can access main functionality of app from this view.

Settings/Preference



The preference or setting screen is invoked from Bottom Navigation menu and allows user to customize the region, notification when new episode available and update frequency for podcast data sync.
With Bottom view menu, user can access main functionality of app from this view.

Profile



The Profile screen is invoked from Bottom Navigation menu.
This view lets user give feedback and share the app when they want.
Also provide entry points for downloaded podcasts and listen history statistics

With Bottom view menu, user can access main functionality of app from this view.

Key Considerations

How will your app handle data persistence?

- The Room Database library will be used to save Podcast meta data .
- Work manager will be used to refresh the data periodically.
- When an episode or Audio book is downloaded or played, data will be saved to database.
- Server calls are made only when necessary.

Describe any edge or corner cases in the UX.

● If network access isn't available, the app will display any podcast information that is saved on the device.

● If search results are empty, display an error message on the screen informing the user of the error.

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

Glide will be used to load and caching images provided by URLs from the API calls

```
implementation 'com.github.bumptech.glide:glide:4.9.0'
```

Retrofit will be used to make API service calls . It is a type safe REST Client for Android it has following advantages

- ● Provides powerful framework for authenticating and interacting with API CALLS and sending network requests with OkHttp.
- ● Makes downloading Json or XML data from a WEB Api fairly straightforward.
- ● Callback methods provide success and error methods overrides which helps in describing service call results.
- ● Builds URL which us specific to API call

implementation 'com.squareup.retrofit2.converter-gson:2.6.0'
implementation 'com.squareup.retrofit2.retrofit:2.6.0'

Room will be used to store data locally on the device for quick lookup when service calls are not required.

implementation 'androidx.room:room-runtime:2.1.0'
implementation ‘android.room:.room-compiler:2.1.0'

Android Architecture/Lifecycle Components will be used to establish view models for the various activities and fragments, so they can seamlessly respond to changes in data that affect the UI.

implementation 'androidx.lifecycle.lifecycle-livedata:2.0.0'
implementation 'androidx.lifecycle:lifecycle-extensions:2.0.0'
implementation 'androidx.lifecycle:lifecycle-viewmodel:2.0.0'
implementation 'androidx.lifecycle:lifecycle-compiler:2.0.0'

Exo Player will be used to play video and audio podcasts

Implementation ‘com.google.android.exoplayer:exoplayer-core:2.10.3’
Implementation ‘com.google.android.exoplayer:exoplayer-dash:2.10.3’
Implementation ‘com.google.android.exoplayer:exoplayer-ui:2.10.3’

Describe how you will implement Google Play Services or other external services.

Work manager will be used to refresh user data from the server data periodically.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Obtain Listen Notes API key and store it in the project properties file so the application has access to the key (Note: do not display the key in any public GitHub repositories)
- Create a project in Android Studio called Puffin Podcaster and add it to a GitHub repository.

- Configure the following libraries (described above) in the application's build. gradle file:
  - Retrofit
  - Room
  - Glide
  - Exo Player
  - Android Architecture/Lifecycle Components
  - Jetpack Work Manager

- Add a buildConfigField and revalue declarations for the Listen Notes API key in the application's build.gradle file

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Main Activity that will allow the user to search podcast and will display the appropriate search icons on the toolbar.
- Build UI for Episode or Audio Books list Activity that displays the available podcasts.
- Build UI for Episode detail or Audio Books chapter detail Activity that displays the podcast details.
- Build UI for Settings/Preference Activity that allows the user to change application settings
- Build UI for My Podcasts Activity that shows the details of users favorite or subscribed Podcasts.
- Build UI for Profile activity, that allows an user share feedback and app. Also provides statistics for listening history.
- Create a fragment with a recycler view and adapter that displays podcast related information. This fragment can be re used with different Activities to display information.

Task 3: Implement Retrofit Service Calls

- Create Retrofit Client Instance in order to perform api service calls
- Create Retrofit API interfaces with annotated methods for each service call that will be made (specifying URL and parameters)
- Implement service calls and handle results appropriately

Task 4: Implement Room Database

- Create and configure Room Database Instance

- Create entity pojos for each of the data objects
- Create interfaces for all the DAOs

Task 5: Implement View Models /Live Data

- Create Live Data objects that are accessed through the View Model that will be used by the UI to detect changes. These will mostly be driven by results from api calls and accessing data from the Room database
- Implement View Models that contain methods that allow the UI to access Live Data objects and observe them

Task 6: Implement Mechanism to Synchronize Data with the Server

Implement Jetpack Workmanager to periodically run code that refreshes application saved data from API service.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"