GitHub Username: jessica-hsu

App Name: PodcastBeMine

Description:

Search for your favorite podcasts and save your personal favorites to listen to later.

Write a brief summary of what your app does. What problem does your app solve?

The app will first ask the user to sign in with Google sign in. That way the same app on a phone could be have multiple users. A main menu page will ask user if they would like to logout, search for podcasts, or view favorites. The user can search for podcasts using keywords and results are displayed on a page. Clicking on a result will lead to a details page and can start the podcast with a button click. Users can also add and remove podcasts from their favorite list or add it to the home screen widget. The favorites list will hold the user's saved podcasts. Users can also listen to podcasts from there.

Who is your intended user? (For example, is this an app for dog owners? Families? Students? Travelers?)

The database where the app will get podcasts contains hundreds of thousands of podcasts, so nearly every topic will have a podcast or episode available. It is intended for anyone with any interests who like to search and listen to podcasts on a mobile phone. One app could be used for multiple users since it has a Google sign in function.

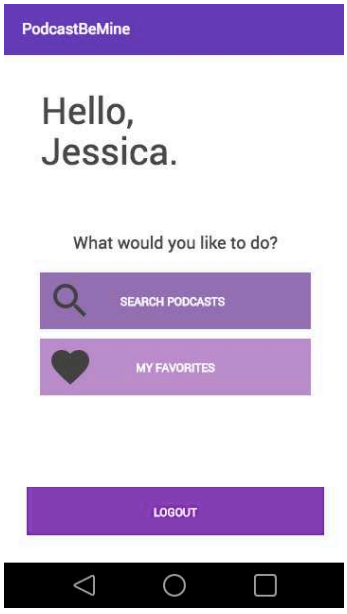
List the main features of your app. For example:

Sign in with Google. Search for podcasts or episodes with a keyword or phrase. View results and podcast details. Play podcast. Save favorite podcasts to a database later. Play a podcast on home screen widget.

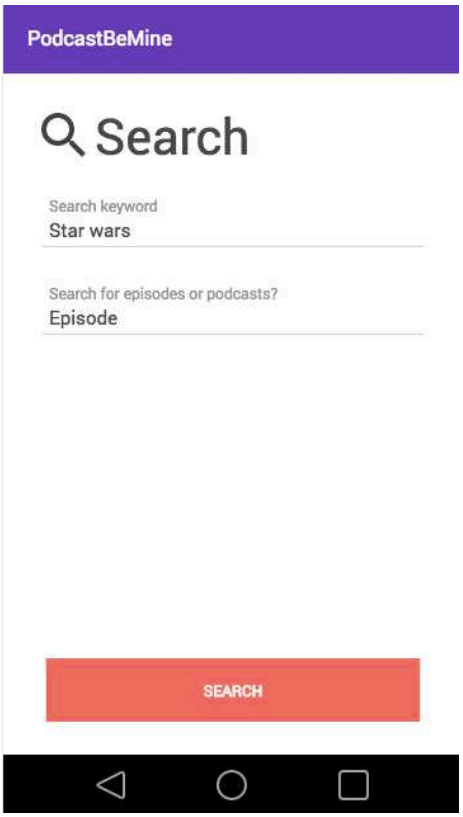
User Interface Mocks

1) Home page: Users will be asked to use Google sign in to start using the application.

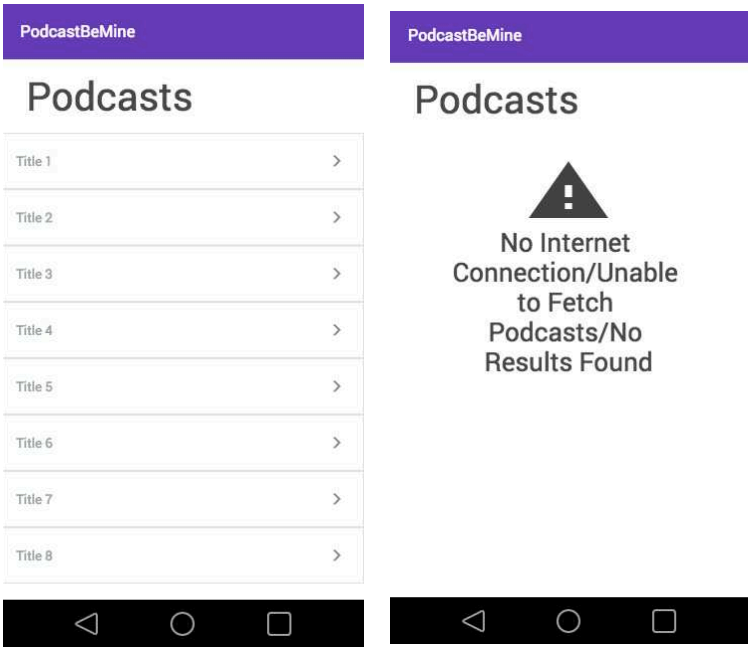
2) Main menu. After sign in, users will be taken to the main menu. It will have a personalized greeting and a few action buttons: Search Podcasts, View favorites, or Logout. Clicking logout will take user back to home screen.



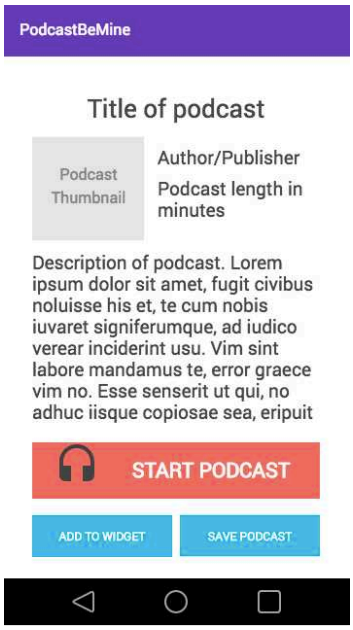
3) Search page. Clicking the search podcasts button from main menu will get the user here. User will enter keyword(s) and decide whether to search for podcasts or episodes.



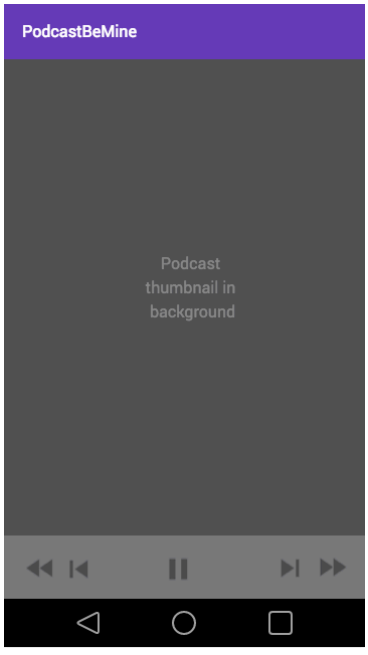
4) Results page. After clicking Search button from the search page, users will be able to see a list of podcast titles based on the search keyword. Maximum number of results will be 30. Clicking the "View Favorites" button from main menu will also take the user to this page, but will show titles that the users save. The second error screen shows only some error occurs (i.e. no internet, search API error, etc) or if there are no results.

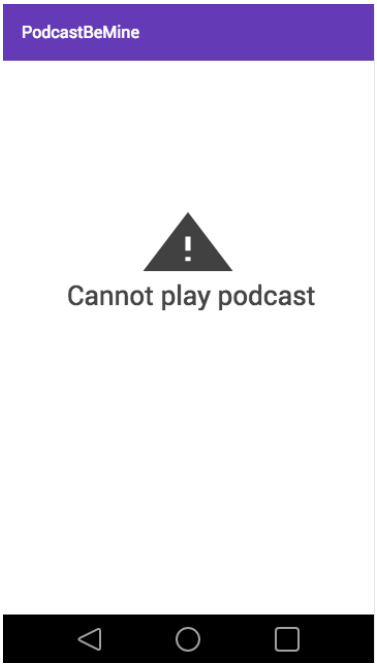


5) Podcast details page. Clicking on one of the results shown in results page will display details about the podcast: title, author/publishing company, audio length in minutes, and description. There are three action buttons: Start Podcast, Add to widget, Save to Favorites. Podcasts added to widget will let user play podcast on a home screen widget. Save/Remove from favorites will either save or delete the podcast from your favorites list.



6) Play podcast page. Starting a podcast will open a new page to show an audio player. If some error happens, the screen will show an error message.





How will your app handle data persistence? Describe how your app with handle data. (For example, will you build a Content Provider or use Firebase Realtime Database?)

Data persistence will be handled with Firebase Realtime Database.

Describe any edge or corner cases in the UX.

Edge cases may include having no Internet connection. To handle this, there will be an Internet connection check before searching for details or playing the podcast. If no connection found, the next screen will display a simple error message instead.
Another case would be going from media player to search menu, to main menu, etc. The app will handle this by menu button that will lead to search page, main menu, or logout. This will be available on every view.

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

1. Exoplayer will be used to play podcasts. I already have experience with Exoplayer and Exoplayer can handle audio file redirect since URL to the audio files will come from an URL redirect.
2. Picasso to handle easy loading and caching images and thumbnails in a simple way.

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

1. Google Sign in will be used in the home page. It will keep track of the current user and use that information to create personalized favorites list when the mobile phone has multiple users.
2. Firebase Realtime Database will be used to store favorite podcasts. Favorite podcasts will be loaded from Firebase.

Next Steps: Required Tasks

Task 1: Project Setup/Configuration

Add the libraries for Exoplayer, Google Sign in, Firebase Realtime Database, and Picasso to your app's Gradle. Modify the Mainfest to allow for Internet Connection.

Follow Google's tutorial to set up your project configurations to use for Google Sign In and Firebase. Go to ListenNotes API and subscribe to the free tier to get an API key. ListenNotes is where the app will pull podcasts from.

Task 2: Create UI and Activity for each screen

Use the mockups provided above to create the home page, main menu, search page, results page, details page, and play podcast page. Create activities for the home page, main menu, search page, results page, details page, and play podcast page. Add dummy data to results page and details page.

Task 3: Implement Google Sign in

Follow Google's Google Sign in to implement the API on homepage. Add required code so that the user stays signed in at all times (even when user closes app) until user hits the Logout button. Add method that will check for Internet connection. If no Internet connection, a snack bar should pop up when user hits the "Sign in with Google button". Redirect to main menu if sign in is successful.

Task 4: Create classes Async task for searching podcasts and parsing details

Create an async task class that will handle the search API call. Create helper classes to handle the HTTP Get call and to parse JSON into a model. Integrate the helper classes in your Async task class.

Task 5: Add Onclick handlers to each of the buttons in every activity.

On Main Menu Activity: Logout button should log user out and go back to home page. Search podcast button will open up the Search activity. View Favorites will open up Results activity.
On Search Activity: Search button should open up Results activity.
On Results Activity: Clicking on a result should open up Details activity.
On Details Activity: Clicking on Start Podcast should open up Play Podcast activity. Hitting Add to Widget or Save to Favorites should open snack bar indicating button was clicked (real implementation to come later)

Task 6: Implement Async Task in Search Activity

Before starting the async task, check for Internet connection. If there is none, go directly to Results Activity but hide the results and display the No Internet Connection message. Begin Async task, get the results back, and pass the results to the Results Activity.

Task 7: Implement Recycler View in Results Activity

Remove all dummy data. Results Activity will use recycler view to display all results. Modify the activity to implement recycler view. Add additional layout xml files if necessary. Read the results passed from Search Activity and load only the title in the recycler view. Add an onclick handler to each recycler view item. This onclick handler should open the Play Podcast Activity and pass the audio URL to the activity.

Task 8: Implement Exoplayer in Play Podcast Activity

Follow tutorials to play the podcast using exoplayer. Check for Internet Connection first. If there is none, show an error message instead of Exoplayer.

Task 9: Implement Firebase Realtime Database

Follow tutorials to set up Firebase Realtime Database in Android Studio. Modify the onclick handler for Add/Remove to Favorites button in Results Activity to add to database if podcast id doesn't exists. Remove if Podcast id does exist. Change button text to say Remove if podcast is already in the database.

Task 10: Read from database function

Go back to the main menu activity. Modify the "View favorites" button's onclick handler to read the database first. Check for Internet connection. If there is no Internet connection, go to Results Activity and display an error message. Implement function to read from database. Pass results to Results Activity.

Task 11: Create Widget

Create a widget with Android Studio. Modify widget layout and class to display and set up exoplayer. Update the Details activity's "Add to Widget" button to send the audio URL to the Widget class.

Task 12: Data validation

Add methods to validate data before parsing them or sending them to the next activity, etc. Find generic “no image available” or “no audio available” icons to use when there is no valid URL for podcasts thumbnail or audio. Do this for every class. Add Internet Connection check when necessary.