GitHub Username: thatsabhi22

SpaceBinge

Description

SpaceBinge is a video streaming app, showcasing latest and upcoming missions and activities as NASA. These videos on space are exclusively created by NASA and available for general public viewership.

With this app:
• Stream unlimited space videos and documentaries created by NASA
• Discover hours of infotainment content, with new videos being added regularly
• Create your personal ‘Watchlist’ list and easily access your collection
• Extensive search inside vast collection of videos of NASA
• Watch full documentaries and short videos for free
• Offline viewing of the videos, download these space videos effortlessly on your device
• User Profile gets created for personalisation on favorite space videos

Intended User

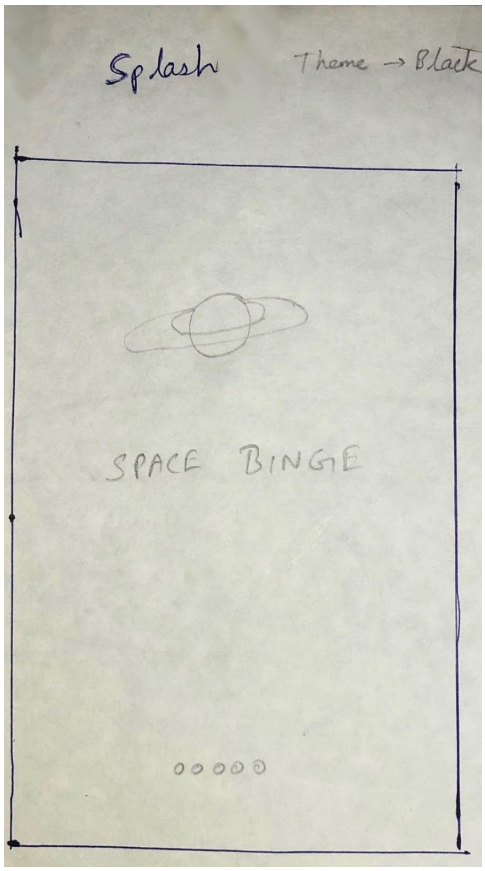
Anyone can use this app, especially space enthusiasts.

Features

- Search Space videos by keywords
- Stream Space videos originally created by NASA
- Download Videos for Offline viewing
- Create your profile for personal watchlist and downloads
- Get latest updates from NASA’s twitter handle in the News section

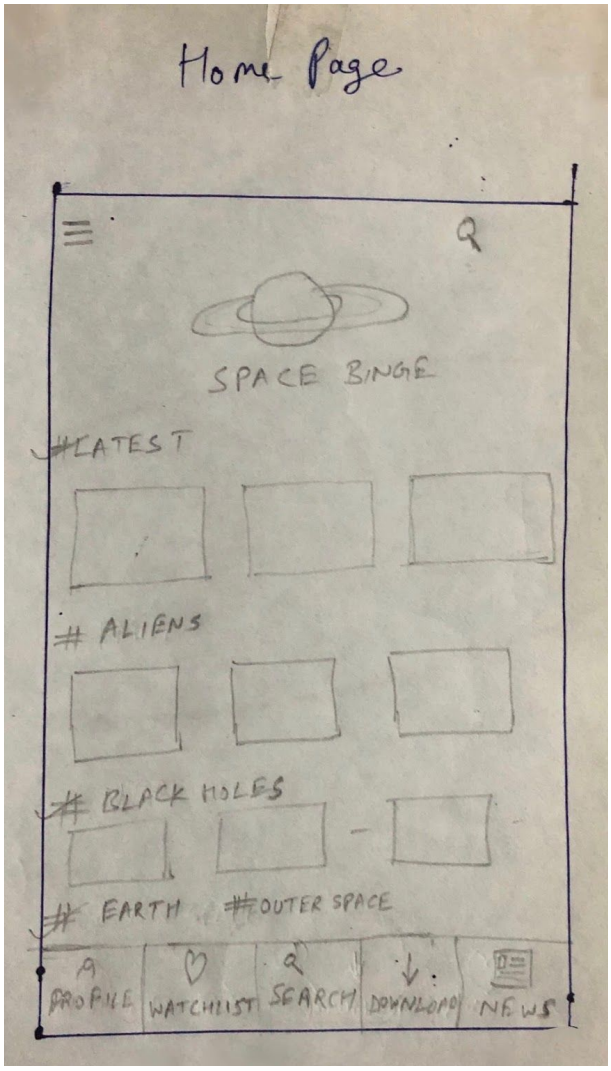
User Interface Mocks

Screen 1



The App opens up with this splash screen

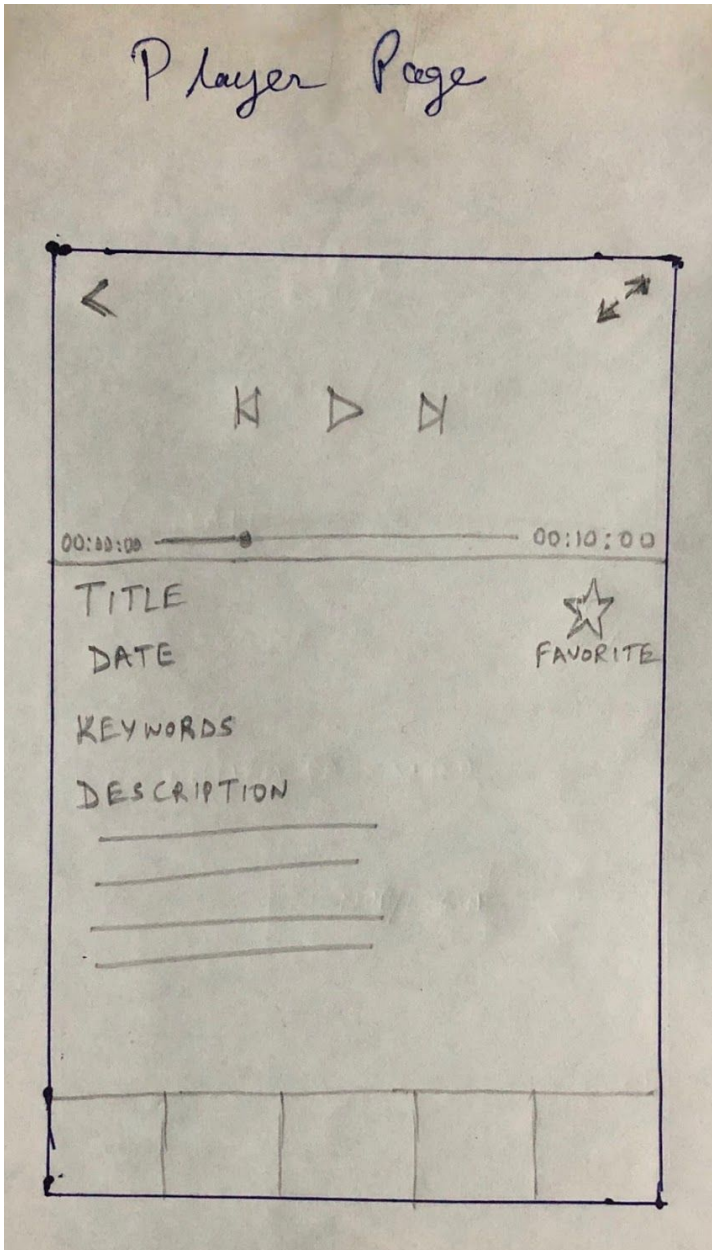
Screen 2



The Homepage contains curated videos collection on various categories in Space.
This also contains the bottom bar that serves for the app navigation between activities.
Primarily

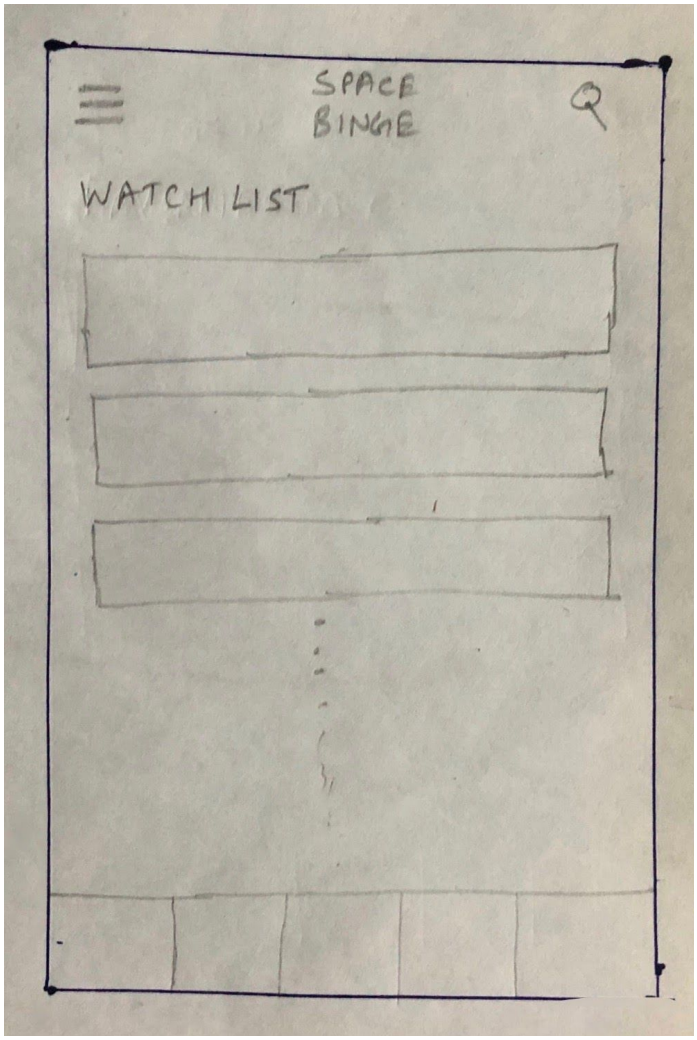
- Watchlist
- Search
- Downloads
- News
- Profile

Screen 3



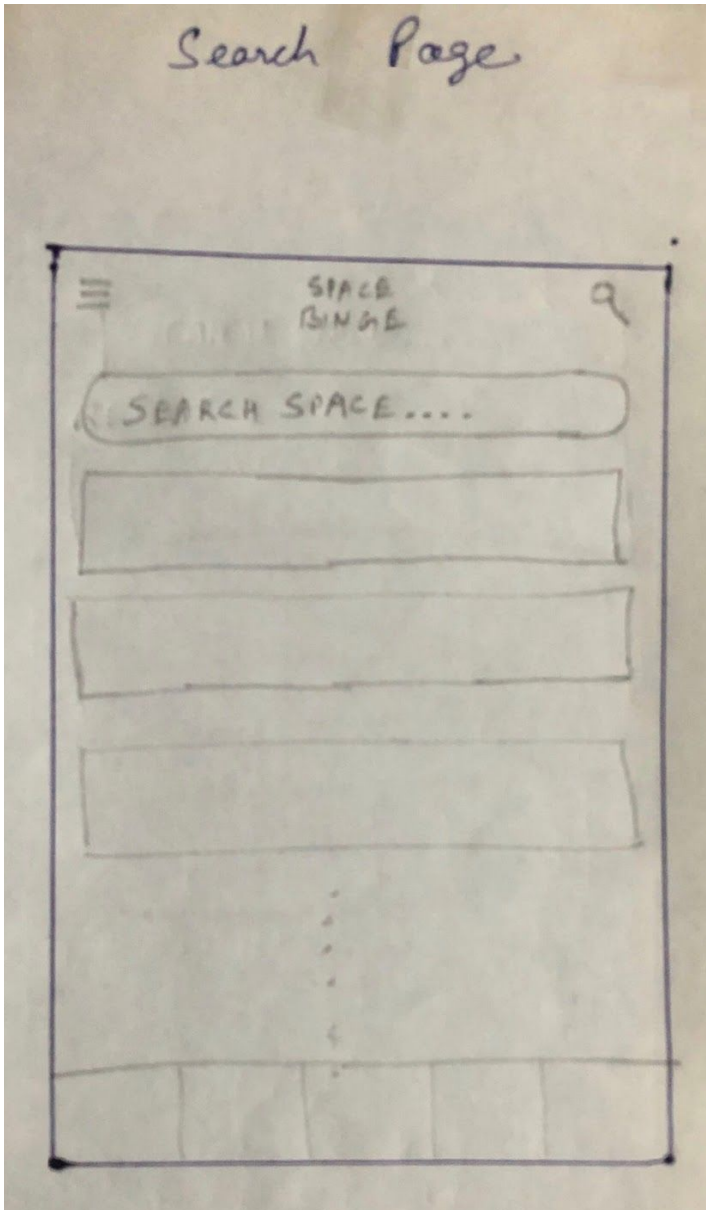
On click of any video from the homepage, User will be navigated to this player page for streaming the respective video. The screen can be viewed on full screen as well.
Users can also mark a video as their favorite. This video will be saved in the watchlist created for that particular user.

Screen 4



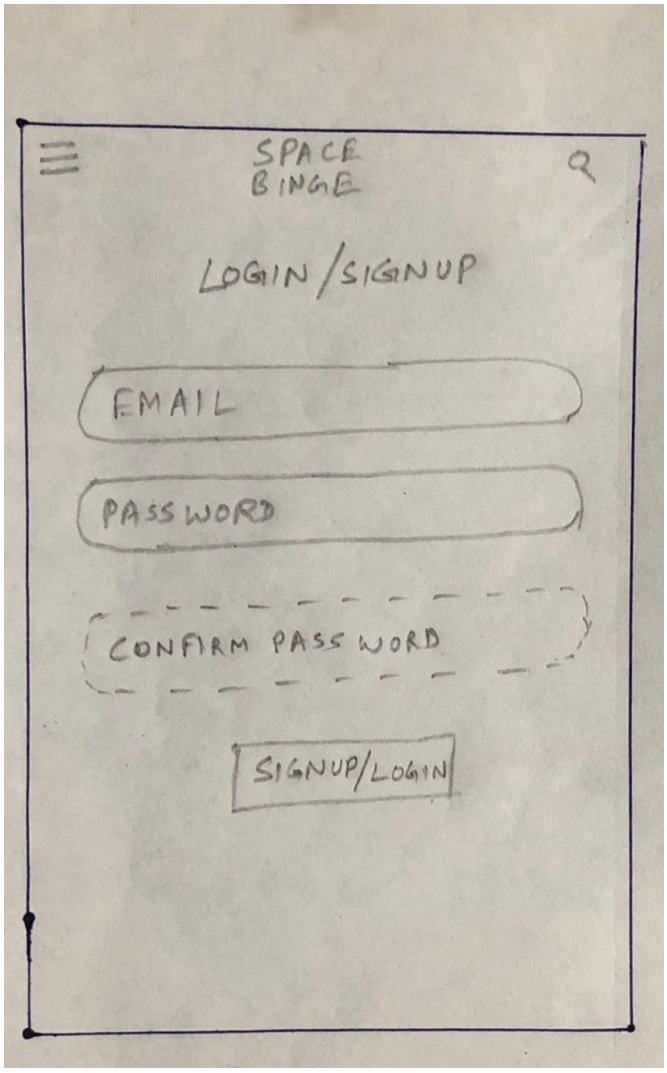
All Videos marked as favorite can be viewed on this screen. Users can either watch them at a later time or watch the favorite videos again from this screen.

Screen 5



This is the search screen of the App. Any topic related to space can be searched on this screen and based on the searched keywords, a search REST API will be called resulting in the related videos search by the User.

Screen 6



This is the signup page, for viewing any video on the app a user has to create his profile from this page. This app is free of cost, the only required before watching any video is to register on the app.

Key Considerations

How will your app handle data persistence?

The latest videos collection is fetched from the internet using the internet API. Also the videos will be watched by streaming from the Internet API.

A User can mark a video to his watch list, in that case the video data will be stored into the local Room database of the app. This will be using Android architecture components Room - Repository - LiveData - ViewModel.

Describe any edge or corner cases in the UX.

As the app opens up, the user lands up to the homepage. He is available with the Bottom Bar to navigate to various screens of the app.

The User will navigate between Homepage, Watchlist, Downloads, Player and News Screen using the bottom bar navigation which is generally used in the video streaming apps.

At any point of time, if the user hit a back button, he/she will be redirected to the homepage which contains the categorised collection of the space videos.

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

The app will utilize stable release versions of all libraries.

- Android Jetpack libraries
    - AndroidX appcompat library to support for the Action Bar, AppCompatActivity.
    - AndroidX Recyclerview library provides support for the RecyclerView widget.
    - AndroidX Cardview library to support the CardView widget.
    - AndroidX ConstraintLayout to build a responsible UI.
    - The Design Support library provides CoordinatorLayout, AppBarLayout, CollapsingToolbarLayout, FloatingActionButton, and Snackbar.

```
implementation 'androidx.appcompat:appcompat:1.1.0'
implementation 'androidx.recyclerview:recyclerview:1.1.0'
```

```
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
implementation 'androidx.cardview:cardview:1.0.0'
implementation 'com.google.android.material:material:1.1.0'
```

- Android Architecture Components (Room, ViewModel, LiveData)
    - Room Persistence Library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite. Room will map our database object to Java object.
    - ViewModel to cache data that needs to survive configuration changes.
    - LiveData makes it easy to keep what's going on screen in sync with the data.

```
implementation ‘androidx.room:room-runtime:2.2.5'
annotationProcessor ‘androidx.room:room-compiler:2.2.5’
```

```
implementation ‘androidx.lifecycle:lifecycle-viewmodel-ktx:2.2.0’
implementation ‘androidx.lifecycle:lifecycle-livedata-ktx:2.2.0’
implementation 'android.arch.lifecycle:extensions:1.1.0'
```

- Picasso to handle the loading and caching of images.

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

- Retrofit for REST API communication.

```
implementation ‘com.squareup.retrofit2:retrofit:2.7.2
```

- Gson to convert Java Objects into their JSON representation.

```
implementation ‘com.squareup.retrofit2:converter-gson:2.7.2"
```

- ExoPlayer to play audio and video locally and over the Internet.

```
implementation 'com.google.android.exoplayer:exoplayer-core:2.10.5'
implementation 'com.google.android.exoplayer:exoplayer-dash:2.10.5'
implementation 'com.google.android.exoplayer:exoplayer-ui:2.10.5'
```

- Timber is a logger with a small, extensible API which provides utility on top of Android's normal Log class.

```
implementation 'com.jakewharton.timber:timber:4.7.1'
```

- Stetho is a sophisticated debug bridge for Android applications. Used for debugging the app on runtime, especially used for debugging the database changes happening at runtime.

```
implementation 'com.facebook.stetho:stetho:1.3.1'
implementation 'com.facebook.stetho:stetho-okhttp3:1.3.1'
```

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

The App will use **Google Mobile Ads** and **Firebase Analytics**. For Google Mobile Ads, banner ads will be used on the bottom of the screen. For Firebase Analytics, initialize analytics and use analytics to log custom events.

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

Create a new project and add a navigation drawer activity. Add dependencies to the dependencies block in the build.gradle file for the module's directory.

- App is written solely in the Java Programming Language.
- App utilizes stable release versions of all libraries, Gradle, and Android Studio.
- Configure libraries
- Create a navigation drawer

Task 2: Implement UI for Each Activity and Fragment

Create each Activity. Create layouts for multiple Activities.

- Build UI for SplashActivity
- Build UI for HomepageActivity
- Build UI for PlayerActivity
- Build UI for SignUpActivity
- Build UI for WatchlistActivity
- Build UI for DownloadsActivity
- Build UI for SearchActivity

Task 3: Fetch data from NASA open API for space videos and images

Add the Internet permission in AndroidManifest.xml to connect to the Internet. To fetch Json data from NASAs open api for space videos and images.

In order to request videos from NASA and return the results on a particular topic for eg “black holes” and media type to be video, the URL will look like the following:
https://images-api.nasa.gov/search?q=black%20hole&media_type=video

Collection object is retrieved in json format.

- Add the Internet permission
- Use Retrofit to turn HTTP API into a Java interface
- Create a POJO

Extract following parameters out of this collection object and form VideoItem entity out of it
id
title
description
video_url
thumbnail_url
nasa_id
href
media_type
date_created

Extracted video_url, thumbnail_url and nasa_id parameters to be used on the player page to stream video data from the NASA open API for images and videos.

Task 4: Set Up the Search Interface

- Add the Search View to the app bar in the Search Screen

Following URL will be hit as a user queries on search screen
https://images-api.nasa.gov/search?q=black%20hole&media_type=video

A collection json will be retrieved from the api. Using Retrofit, will get converted to a Java object. Picking up necessary information, VideoItem object to be created and displayed as the search result on the search screen.

Task 5: Implement the Android Architecture

- Create a database with the name VideoItem.
  - ➔ id
  - ➔ title
  - ➔ description
  - ➔ video_url
  - ➔ thumbnail_url
  - ➔ nasa_id
  - ➔ href
  - ➔ media_type
  - ➔ date_created
- Create an Entity to define a database table
- Create a DAO to provide an API for reading and writing data
- Create a Database which represents a database holder
- Create Repository to handle both databases as well as internet requests in a single class.
- Add the LiveData and ViewModel
- Make sure no unnecessary calls to the database are made

Task 6: Implement media streaming with ExoPlayer

Use ExoPlayer to play a video. Initialize and release video assets when appropriate. Show Media Style notification, with actions that depend on the current Media Session PlaybackState.

- Initialize ExoPlayer
- Create a MediaSource
- Release ExoPlayer
- Restore the playback position
- Customize the PlayerControlView
- Set the Player.EventListener
- Add Media Session and Media Notification

Task 7: Implement a splash screen with an Retrofit

- Create a new Activity for a splash screen
- Load a splash screen before entering the main screen by using an Retrofit call

Task 8: Implement Google Play Services

- Display test ads for a free version
- Connect the app to the Firebase
- Each service imported in the build.gradle is used in the app.
- App creates only one analytics instance.
- Add Log events for the Firebase Analytics

Task 9: Handle Error Cases

- Handle errors when loading images by using onError() method of Picasso Callback
- Redirect to Signup screen, when the user has not signed up.
- Check network connectivity. If there is no network connectivity, show a snackbar message
- App validates all input from servers and users. If data does not exist or is in the wrong format, the app logs this fact and does not crash.

Task 10: Add Visual Polish and Use Material Design

- Change colors and fonts
- Add a touch selector
- Make the app more delightful with material design patterns
- App uses standard and simple transitions between activities
- App theme extends AppCompat
- App uses an app bar and associated toolbars

Task 11: Add a Signing Configuration

- Create a keystore and a key
- Create a signing config in build.gradle
- Assign the signing configuration to a build type
- The keystore and passwords are included in the repository. Keystore is referred to by a relative path.

Task 12: Build an App

- App builds from a clean repository checkout with no additional configuration.
- App builds and deploys using the installRelease Gradle task.
- All app dependencies are managed by Gradle.

Task 13: Confirm to common Standards

- App conforms to common standards found in the Android Nanodegree General Project Guidelines