- Description
- Intended User
- Features
- User Interface Mocks
  - Screen 1
  - Screen 2
- Key Considerations
    - How will your app handle data persistence?
    - Describe any 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.
- Next Steps: Required Tasks
  - Task 1: Project Setup
  - Task 2: Implement UI for Each Activity and Fragment
  - Task 3: Your Next Task
  - Task 4: Your Next Task
  - Task 5: Your Next Task

GitHub Username: Gregbaggings

OtakuBook

Description

OtakuBook is an app to browser and mark different animes that is interesting for the you. Track what you want to watch by saving it to your personal list or just check the story and details of an interesting show.

Intended User

Anime fans, TV fans, kids and adults who like series

Features

- Presenting a list of animes by given predefined filters for seasons & latest trending and search for given ones via title,
- Browsing anime details like:
  - genres,
  - airing,
  - synopsis,
  - characters,

    - episode number,
    - and videos.
- Mark a show as favorite and save it for later tracking.

User Interface Mocks

Screen 1

In the main activity (home screen) the user will see the popular movies/animes at top and can click them to directly access their details.

The user can search for a title here. If there is a direct hit, the user is moved to the details activity. If there are more titles (like for example “Naruto” – where more series and movies are containing this text), the user is moved to the search result activity from where can move to the details screen or do other interactions, see below on the search result activity screen details.

With the favorites bottom button, the user can access his/her favorites if she/he is logged. Else a toast message will be shown to request them to login.

The login is available in the top appbar, under the menu of the 3 dot button.
The surprise me will show the details activity with a random anime.

By tapping on the seasons, the user will be moved to the season list activity. All the data obtaining will be handled via asynctasks.

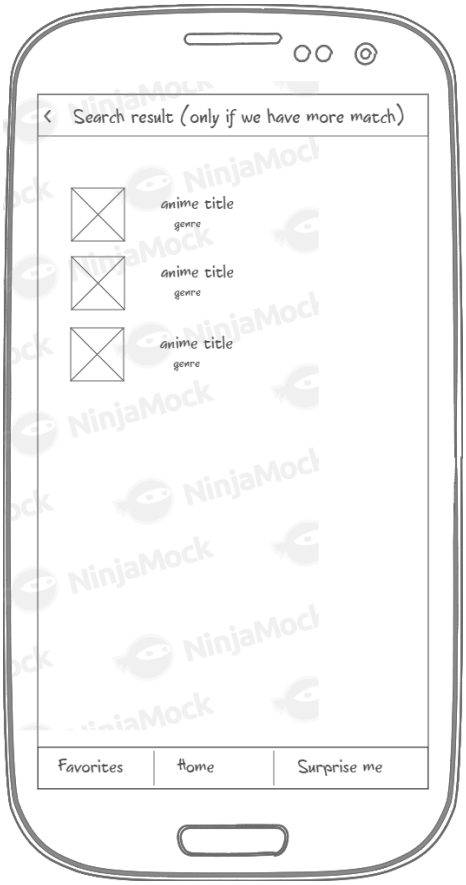
Screen 2

On the details activity the anime details will be presented returned by the API via asynctask. Under the character list there will be an embedded YouTube player to play video(s) related to that anime.

Screen 3

Shows the list of the favorite marked shows. With a swipe action, the user can delete any of them here.

Screen 4



As mentioned above, if there are more hit for the given title, we show this activity and the list of the hits.

Screen 5

On this activity the user can pick a year and season to see what shows were airing at the time.

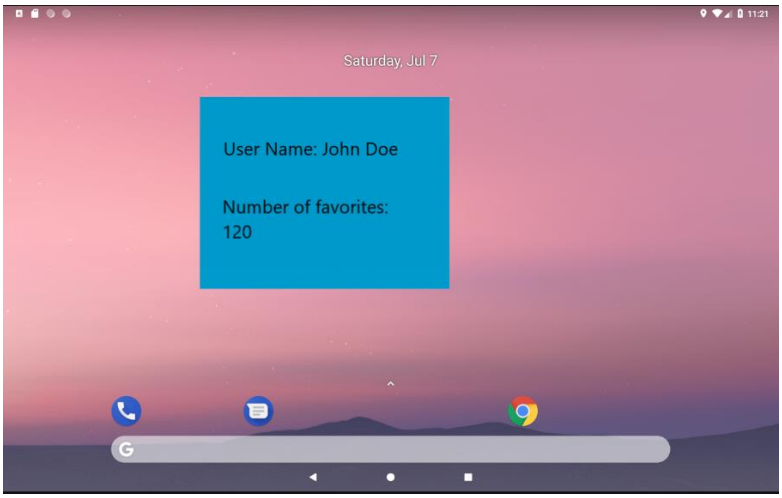
There are 4 seasons from which the user can pick:

- spring,
- summer,
- autumn,
- winter.

The shows will be visible in a recycler view, with grid layout.

Screen 6

In the required widget, I will show the total number of favorites of the user and his/her user name. Note: this is not the final design.



General UI things:
All the imageViews will have content description for the accessibility support and proper image sizing. Other content will use proper colors to make the use of the apps great and maximize the user experience.

Key Considerations

How will your app handle data persistence?

The data will be obtained from the Jikan open API through network. The application will use a live Firebase Database to store and manage the favorites of the users obtained from the mentioned API.

Describe any edge or corner cases in the UX.

At leaving the detail screen, the state of the video player will not be saved as when return the user may not want to continue to watch it.

During obtaining the information for the main screen and the details screen, the application will show a loading indicator. In case of no network, the app will show a toaster message and a sad smiley on the screen to inform the user about the error case.

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

The programming will be in JAVA using Android Studio, with Gradle 3.1.3. The version may differ if any newer version is out!

'com.android.support:appcompat-v7:27.1.1' – for the great layout
'com.android.support:design:27.1.1' – for the great layout
'com.android.support:support-v4:27.1.1' – for the great layout
'com.android.support.constraint:constraint-layout:1.1.2' – for the great layout
'com.android.support:recyclerview-v7:26.1.0' – for the great layout
'com.google.firebase:firebase-core:16.0.1' – to access Firebase features
'com.google.android.gms:play-services-auth:15.0.1'
'com.google.firebase:firebase-auth:16.0.2' – for Google Sign In
'com.google.firebase:firebase-database:16.0.1' – for accessing Firebase Realtime DB
'com.squareup.picasso:picasso:2.5.2' – to load images
'com.squareup.retrofit2:retrofit:2.3.0' – for api requests
'com.google.code.gson:gson:2.8.0' – for json serialization
'com.squareup.retrofit2:converter-gson:2.1.0'
'com.squareup.okhttp3:logging-interceptor:3.4.1'
'com.squareup.okhttp3:okhttp:3.8.0'
'com.jakewharton:butterknife:8.8.1' – for binding views
'com.jakewharton:butterknife-compiler:8.8.1'
'com.github.davidmigloz:youtube-android-player-api-gradle:1.2.2' -for embedded YouTube view

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

On the main screen of the app an ad will be presented via the admob service and the user will be able to login with his/her google account via the Firebase Authentication service. The third service will be a Realtime database though Firebase to store the favorites of the user.

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,
- Set the SDK details,
- Add the dependencies into the build.gradle of the app and setup Firebase for the project

Task 2: Implement initial UI for each Activity

- Create the layout for the MainActivity
- Create the layout for the DetailsActivity
- Create the layout for the SearchResultActivity
- Create the layout for the FavoritesActivity

Task 3: Create the APIRequestHandler class

- Create the async request handler and other necessary classes to fire API requests and obtain - populate the results on the initial UI in the respective layout elements

Task 4: Complete the MainActivity

- Show details in the top recycler view about the most popular animes
- Add the Google sign in feature and store the user name into the extras
- Add the admob

Task 5: Complete the DetailsActivity

- Show details of the opened anime according to the layout sketch
- Implement saving the show to the favorites with the Realtime database

Task 6: Complete the SearchResultActivity

- Show the result in a recycler view according to the layout sketch

Task 7: Complete the FavoritesActivity

- Show the result in a recycler view according to the layout sketch
- Implement swipe based delete for the items

Task 8: Complete the SeasonListActivity

- Show the result in a recycler view according to the layout sketch

Task 9: Complete the Widget

- Create the widget to show the number of favorites and user name.

Task 10: Polish the features if necessary

- Code clean up, minor refactor

Task 11: Layout polish

- Finalizing the layout by using the different xml sources like:
    - string.xml
    - colors.xml
    - etc.
- Applying material design guidelines

Add as many tasks as you need to complete your app.

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**"