How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: pantmurino

Oh My Board Game!

Description

"Oh My Board Game!" is table top and board games database, based on a community supported and well known web resource: https://boardgamegeek.com/. It contains overviews, ratings, description, reviews and much more for more than 90000 board games.
What if you would like to know what is the most interesting and newly published board game available? Or maybe you would like to see the list of the most rated games? Or maybe you're looking for some specific board game and would like to learn more about it? Found a game for you and would like to save it for later or share with friends?
No problem. You can do all of the above with "Oh My Board Game!" application.

Intended User

Application is intended for table top games and board games players. Would be a good help both for seasoned players and for people just beginning with games.

Common project requirements

- App is written in Java programming language
- App utilizes stable release versions of all libraries, Gradle and Android Studio

Features

- Uses BoardGameGeek.com as a backend
- Gets the list of most rated board games from the backend
- Gets the list of most recent board games from the backend
- Searches for the game specified by user
- Gets additional information about a game from the backend:
    - Overview
    - Rating
    - Approximate playing time
    - Approximate game complexity
    - List of reviews
- Saves information about a game in a local database

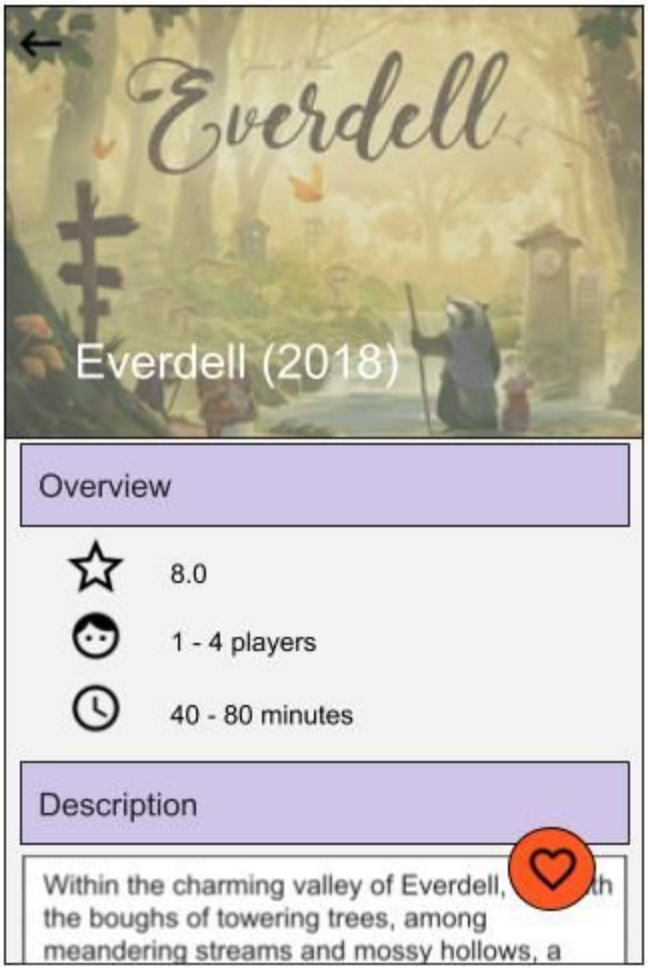
User Interface Mocks

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.

Screen 1

Main screen. Contains a list with thumbnails for board games. Options button allows to sort by date, rating or show favorites. Search button opens a search bar to search for a particular game.

Screen 2



After clicking on a thumbnail in main screen, board game details screen is opened. It contains the name of the game and the production year in application bar. Below quick overview of the game is available:

- Rating
- Amount of players
- Complexity
- Duration

If to scroll down some additional information about the game is available, such as description of the game.

Widget

Widget contains last selected game image preview plus all the important information on it: rating, amount of players, avg. playing time, etc.
Clicking on the widget should open screen 2 with game details information.

Key Considerations

How will your app handle data persistence?

Application will use Room database to store information about user favorite board games locally.

Describe any edge or corner cases in the UX.

If trailer or overview video link is available on the backend, it should be shown as a thumbnail. User could click on it to start playing. If changing a device orientation to landscape mode, video should resume playing from the right place in a full screen.

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

- Picasso to load thumbnails asynchronously
- Room to store user favorites
- Retrofit to communicate with the backend
- Android design library to utilize app bar and fab
- AndroidX RecyclerView component to show a list of games
- DataBinder to simplify access to UI elements in activities and reduce boilerplate code
- Firebase Analytics library to gather important information about app usage

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

- Ads banner will be integrated into the application to use AdMobs backend
- Firebase Analytics will be integrated into the application to use Analytics to gather info about application usage
- BoardGameGeek REST XML based API in combination with Retrofit library will be used

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

- Add Picasso, Retrofit and Design library dependencies to a project
- Add dependency to RecyclerView

Task 2: Implement UI for Each Activity and Fragment

Now we need to sketch out our future UI and mock the data:

- Create a layout for MainActivity and bind it to MainActivity. It should contain recycler view to show a list of games.
- Create a layout to main screen list item - basically a thumbnail with game preview
- Create a layout for BoardGameDetailsActivity. It should utilize ScrollableAppBar from Android Design Library.

Task 3: Create POJO for application business logic

Now we need to create appropriate data structures to represent out BL:

- Create appropriate Java classes to represent a game and a list of games

Task 4: Create appropriate backend interfaces

Now we need to prepare the stage to connect BoradGameGeek backend to BL of our application. BoardGameGeek provides REST XML based API:
https://boardgamegeek.com/wiki/page/BGG_XML_API2
Utilize Retrofit library to create appropriate API for backend.

- Familiarize yourself with BGG API
- Define appropriate backend interfaces
- Use Retrofit annotation library to describe backend interfaces: https://square.github.io/retrofit/

Task 5: Use backend interfaces to populate UI

Now we need to bind backend interface with our UI in order to populate it with a real data:

- Fetch a list of games and games descriptions and add them to the main activity's recycler view
- Fetch details of a particular game and populate game details activity screen
- When app needs to pull or send data from a backend itt uses an IntentService to do so

Task 6: Add sort by functionality

Now we will add sort by functionality for main activity:

- Add options button to a toolbar with appropriate options
- Using option button choose which list to fetch from the backend:
    - Most popular
    - Hottest

Task 7: Add search for a game functionality

- Add FAB for search to main activity
- Add functionality to fetch a searched game and display it in details activity. If there are more than one game ask user to pick which one he/she would like to view

Task 8: Save user favorites to local Room database

- Add FAB to add a game to user favorites to game details activity
- Prepare Room database object description (https://developer.android.com/topic/libraries/architecture/room)
- Add functionality to save a game to favorites when user presses FAB or remove it from favorites when user presses FAB again
- Add an option to main screen to show user favorite board games
- Use LiveData and ViewModel to avoid unnecessary calls to a database

Task 9: Add free flavour

Add a new flavour to an app which will show AdMobs banner on main activity screen:

- Add a new flavour to gradle
- Prepare a new layout for free flavour which contains a banner
- Utilize AdMobs backend to show a commercial

Task 10: Integrate Firebase Analytics and add usage reports

Integrate Firebase Analytics library and add usage reports for common application use-cases:

- Open a list of most rated games
- Add a game to favorites
- Open a list of favorite games

Task 11: Add application widget

Add application widget which shows currently selected goard game image preview plus all the important information: rating, amount of players, avg. playing time, etc.

- Add a widget to home screen
- Add a handler to handle click on a widget to open game details activity

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**”