GitHub Username: kae-udacity

BGDb

Description

BoardGameGeek is a board game database of over 84,000 games. This app is to provide an easy, mobile-friendly and aesthetically-pleasing way to search for games and save favourites.

Displays the latest top board games on BoardGameGeek. Users can search BoardGameGeek for their favourites and save them.

Intended User

Board game lovers.

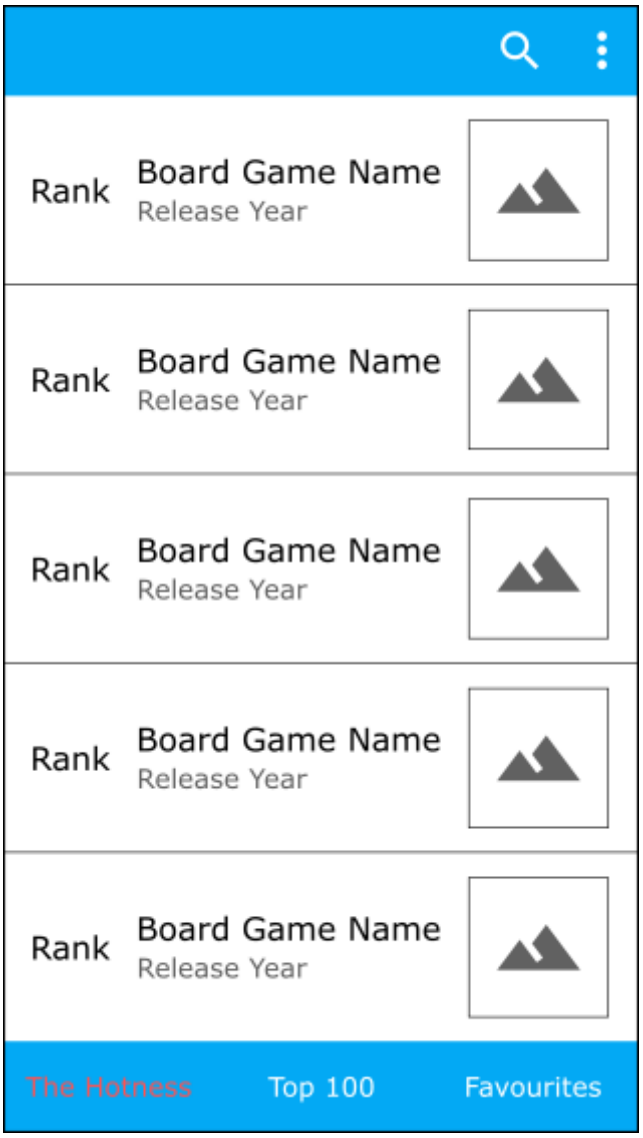
Features

- Displays the top board games using BoardGameGeek.
- Users can search for board games on BoardGameGeek.
- Users can save their favourites.

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.

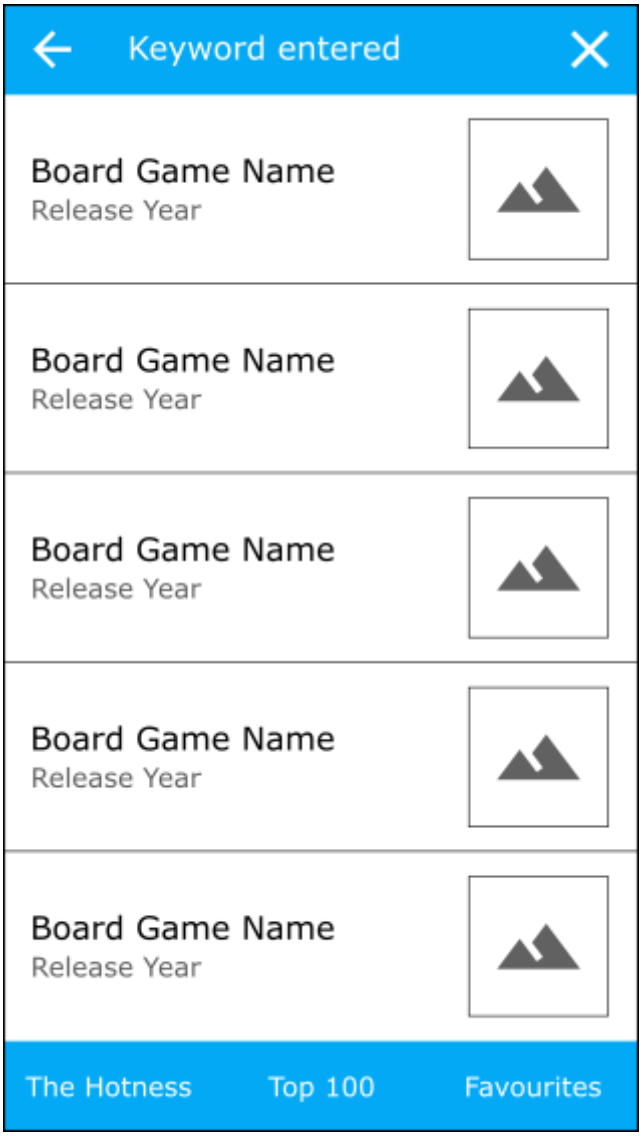
Screen 1



The main screen of the app displaying the list of most popular games (called The Hotness on BoardGameGeek). Tabs are along the bottom for easy navigation. Tapping on a tab will

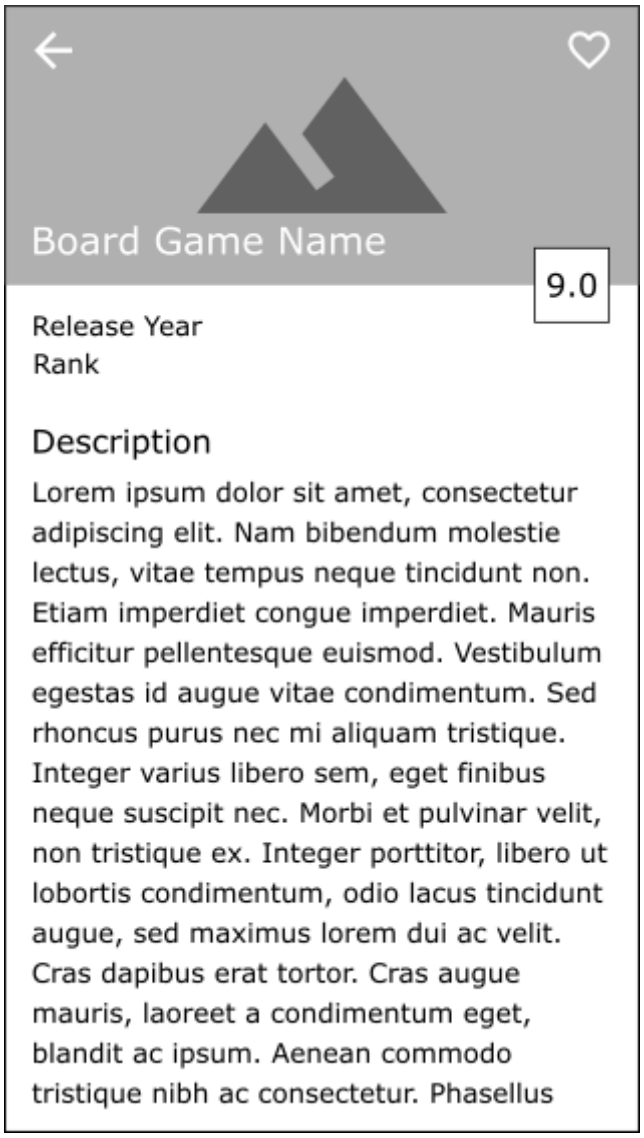
display the list of board games for that tab in a RecyclerView. The top action bar includes a search icon.

Screen 2



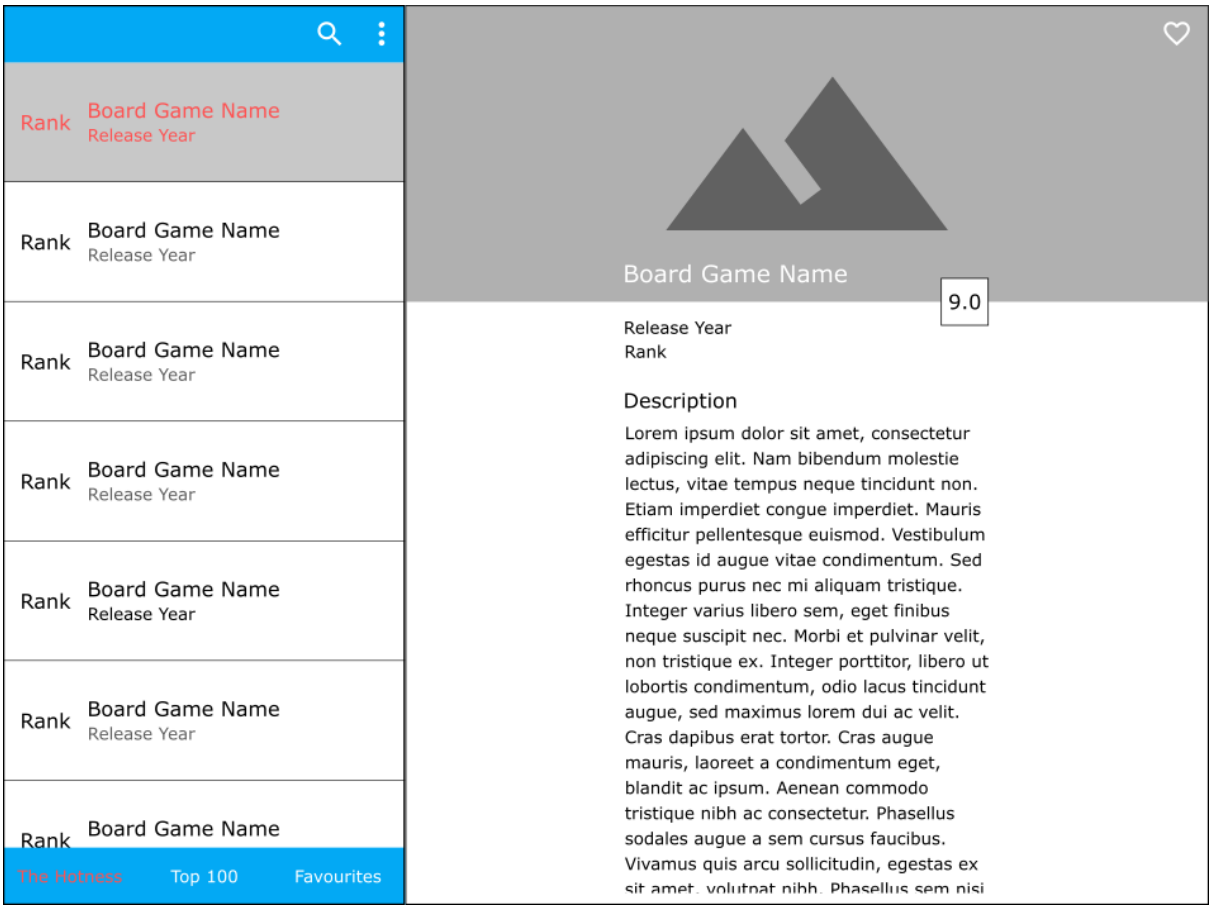
The search results screen. When the search icon is tapped, a text field is displayed in the action bar where a user can enter a keyword. Includes a back arrow and close icon in the action bar. Search results are displayed in a RecyclerView.

Screen 3



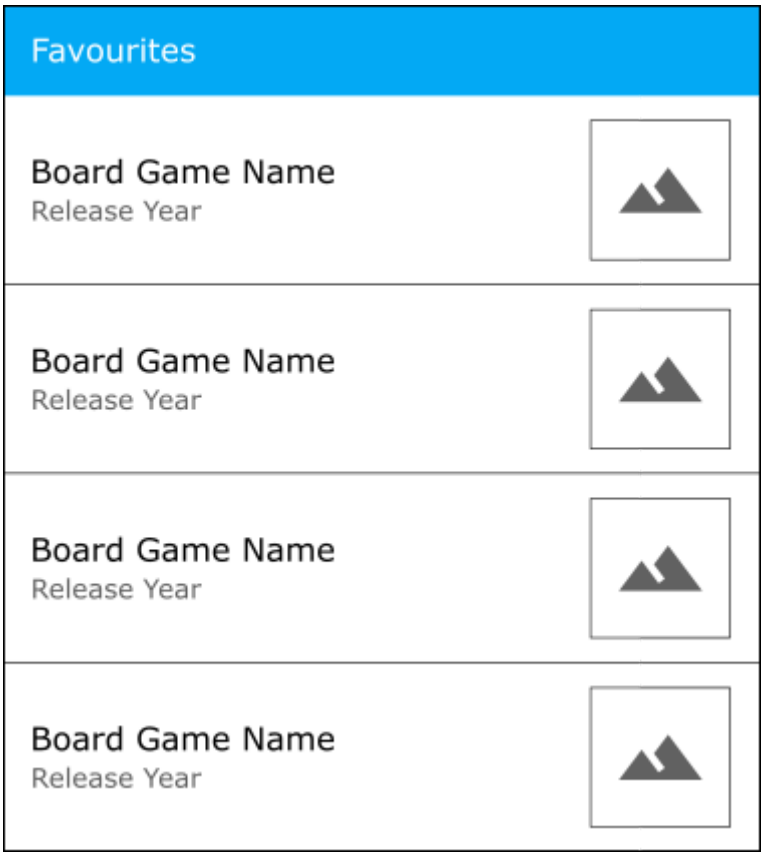
The board game details screen. Includes an up arrow to return to the list and a heart icon to favourite and unfavourite board games. The image will collapse when scrolling and display the action bar.

Screen 4



The main screen on a tablet. It will follow the master detail flow with the list of board games on the left. The tabs will be placed below the list. The right side will be used to display the details of the selected board game. The ImageView will collapse as the details view is scrolled.

Screen 5



The widget will show a list of the user's favourite board games.

Key Considerations

How will your app handle data persistence?

Favourite board games will be saved using a ContentProvider and SQLite.

Describe any edge or corner cases in the UX.

User exits the app or presses the back button while performing a search, request should stop without error.
User exits the app or presses the back button while updating the database.
User performs a search when there is no network connection.
User performs a search and then rotates the device.

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

Glide because it handles the loading and caching of images.
Butterknife because it handles the binding of views and removes the need to use findViewById multiple times.

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

Firebase Invites: to easily allow users to refer others to app.
Firebase Analytics: used to log when the app crashes and when the user first opens the app.

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 dependencies to build.gradle.
- Configure Firebase services.
- Set correct target and compile SDK versions.

Task 2: Implement UI for Each Activity and Fragment

- Build UI of main activity - single pane for phone and dual pane for tablets.
- Build UI of fragment containing pager with tabs.
- Build UI of detail fragment containing photo and info of board game.
- Build UI of detail activity to hold detail fragment in single pane.

Task 3: Implement search function

- Implement AsyncTask to handle board game search requests.
- Add class to send and retrieve requests, as well as extract info from JSON.

Task 4: Implement favourites database

- Implement database contract.
- Implement ContentProvider.
- Implement Loader for CRUD operations.

Task 5: Implement Firebase services

- Implement Firebase Invites.
- Implement Firebase Analytics.

Task 6: Implement widget

- Implement widget config and layout.