How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: edmundjohnson

Movie Companion

Description

The Movie Companion provides the user with a curated list of movies of the week and DVDs of the week, with a short review of each movie. The user can add each movie to their wishlist, mark it as watched, or mark it as one of their favourites.

The Problem:
When choosing a movie to go and see at the cinema or to watch on Goolge Play, it is time-consuming to go through all the reviews to find out which ones are high-quality and mind-expanding. Even then, it can be hard to remember which ones you have previously decided you would like to see, which ones you have already seen, and those you would like to recommend to your friends.
Movie Companion allows the user to keep track of all this, as well as filtering by genre depending on their mood, to allow them to quickly hone in on something inspirational.

Intended User

Movie lovers.

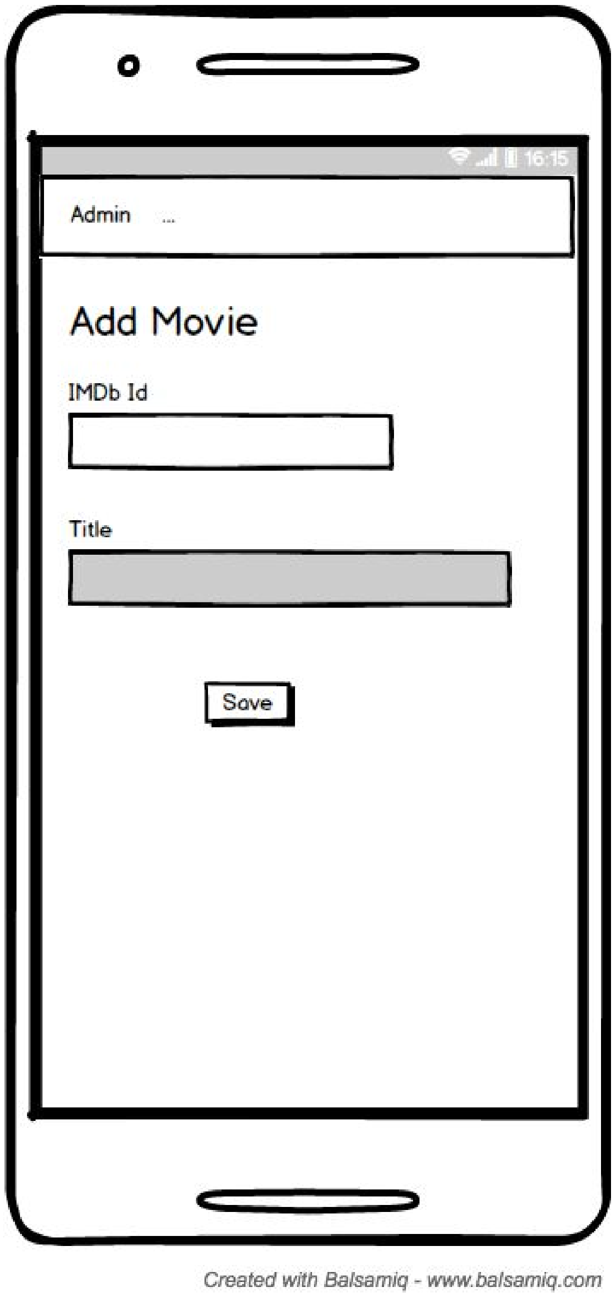
Features

- Recommended latest movies and DVDs, with reviews.
- Personalised wishlist.
- Movies can be marked as watched or favourites.
- Ordering and filtering of movies.
- Link to IMDb page for further info.
- Grid view and list view layouts for the movie list.
- Portrait and landscape layouts for the movie detail screen.
- Widgets which display the latest movies of the week (‘In cinemas now’) and the latest DVDs of the week.
- An admin product flavour which allows reference data to be entered, with some data being fetched remotely via JSON / REST.

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 Photoshop or Balsamiq.

Add Movie (admin)

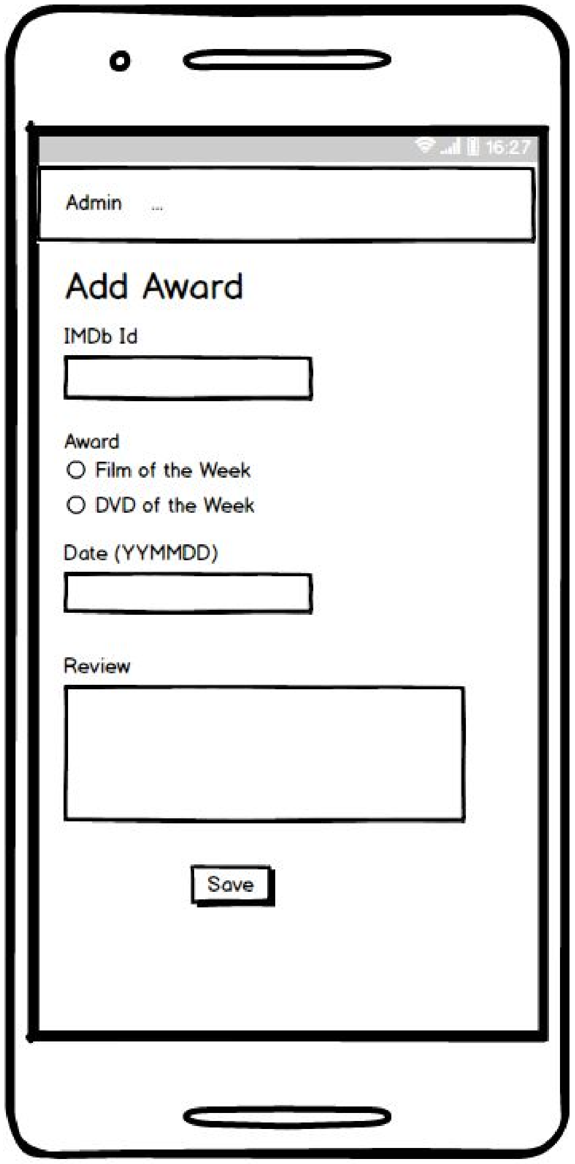


This screen is used to add a movie to the database.
This screen is available in the admin product flavour only. It is not public-facing.
Initially, the Save button is not displayed.

After the IMDb Id is entered:
- If the movie already exists in the local database, an error dialog is displayed.
- Otherwise the movie data is fetched remotely, the title field is populated and the (previously invisible) Save button is displayed

When the Save button is clicked, the movie is saved to the local database.
For now, editing and deletion of movie data is done from the Firebase console.

Add Award (admin)

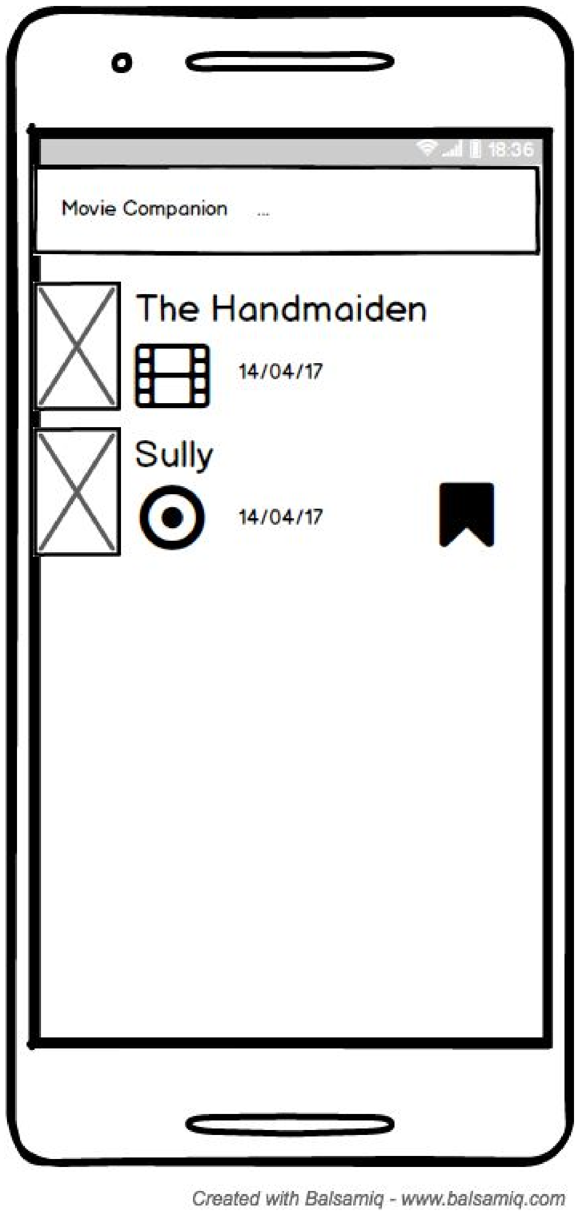


Created with Balsamiq - www.balsamiq.com

This screen is used to add a movie to the database.
This screen is available in the admin product flavour only. It is not public-facing.
When the Save button is clicked, the award is saved to the database, or an error dialog is displayed.

The screen may also contain a ‘priority’ text field, to determine the relative priority of reviews where a movie has more than one award.
For now, editing and deletion of award data is done from the Firebase console.

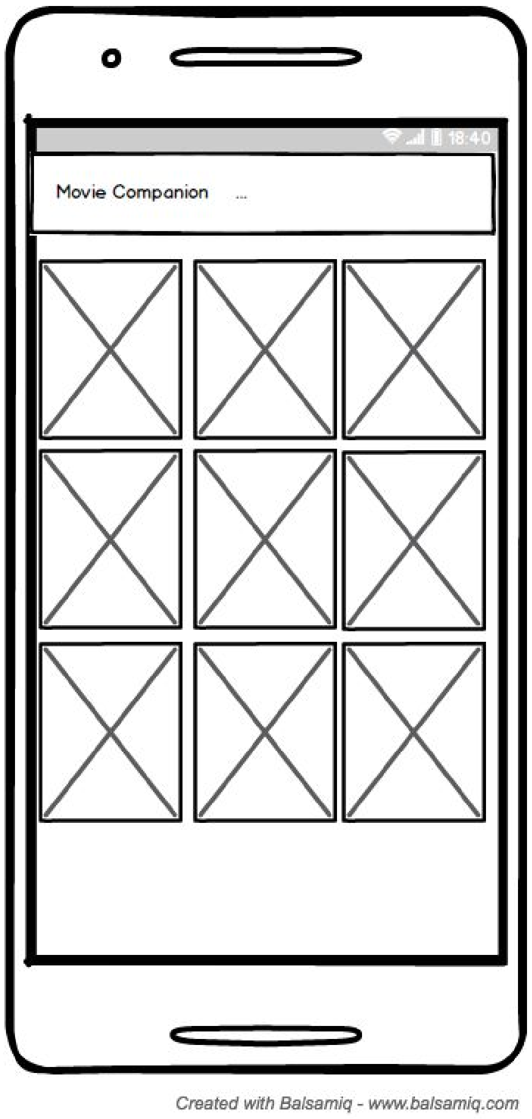
Movie List - List View



This is the ‘home’ activity. Strictly speaking it is a list of awards, rather than a list of movies.
Clicking on a list item takes the user to the corresponding Movie Detail screen.
The action bar contains:
- a grid icon; clicking on it changes the layout to the grid view.
- a sort icon; clicking on it allows the user to modify the list order, via a dialog.
- a filter icon; clicking on it allows the user to filter the list, via a dialog.

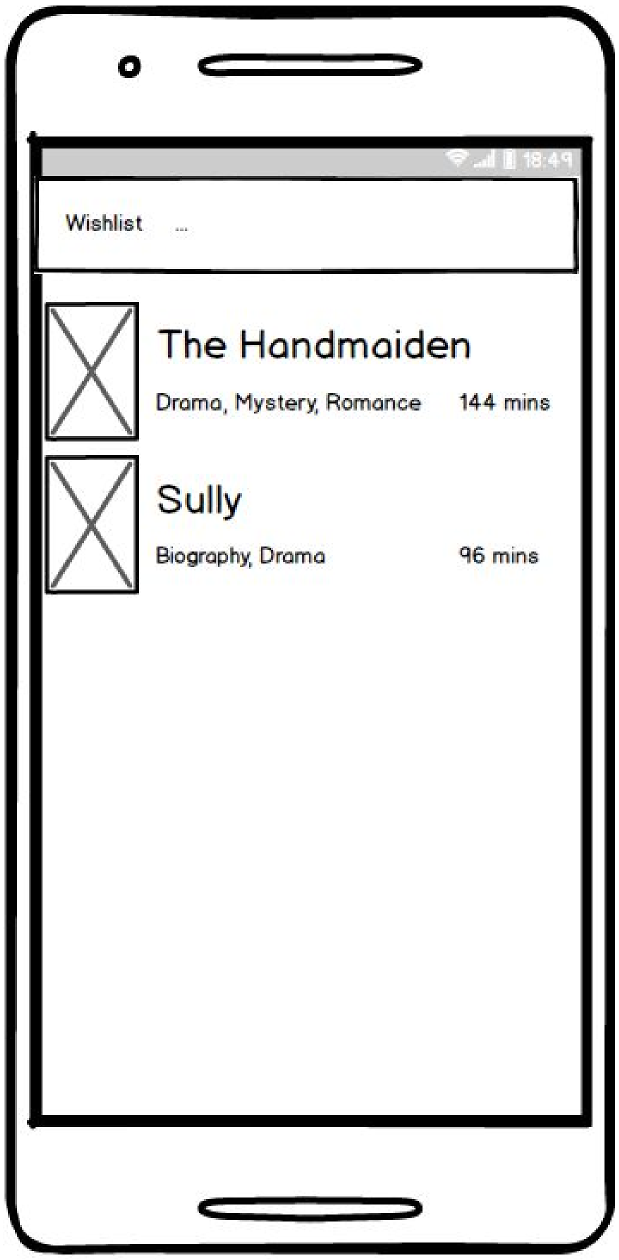
Navigation to the user's wishlist is via an action bar icon or tabs.

Movie List - Grid View



This layout has the same functionality as the List View.
The number of items per row will be determined during development.
The action bar contains a list icon, which changes the layout to the list view.

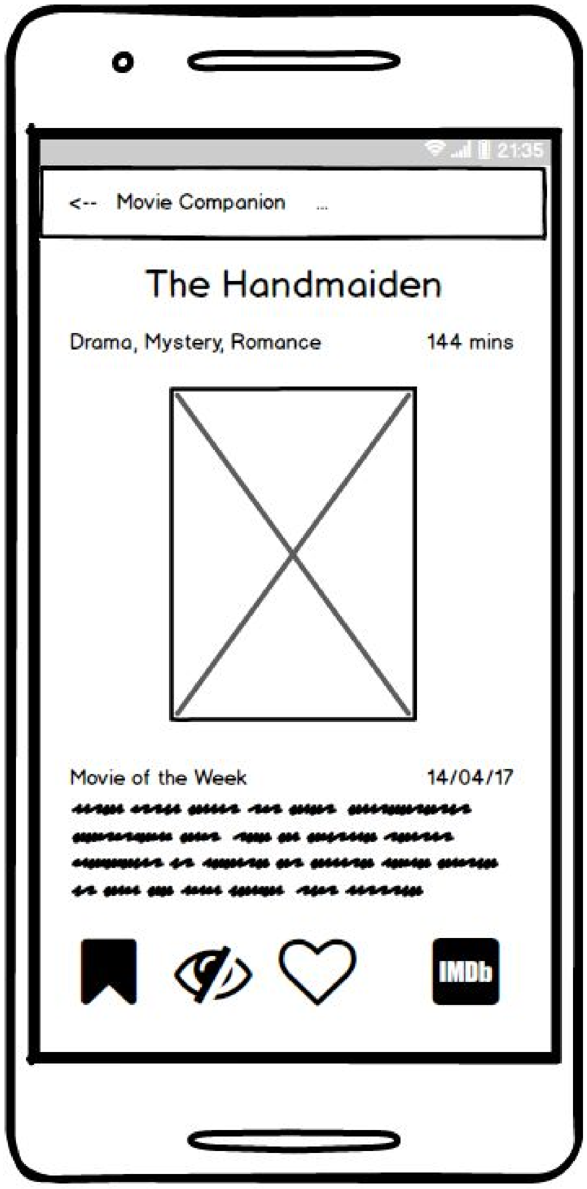
Wishlist



Created with Balsamiq - www.balsamiq.com

Clicking on an item takes the user to the corresponding Movie Detail screen, for the highest priority review.
Navigation to the movie list is via an action bar icon or tabs.

Movie Detail - Portrait

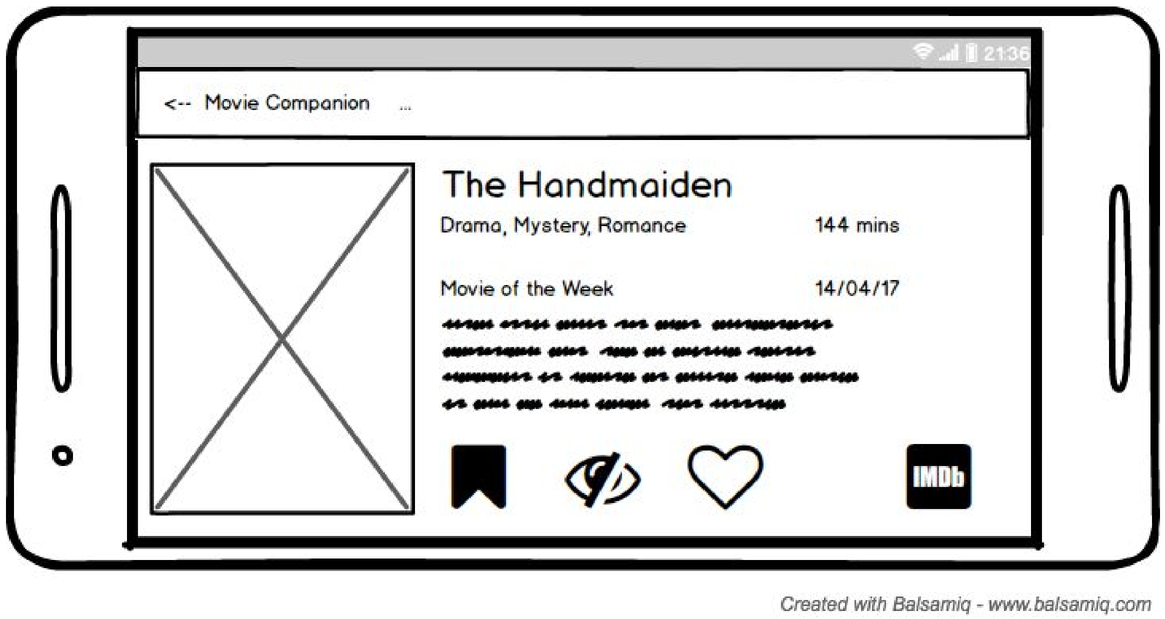


Created with Balsamiq - www.balsamiq.com

It is possible that the movie poster image will be the background for the whole screen.
The text below “Movie of the Week” is the movie review, which is a scrolling area.
The ‘wishlist’, ‘watched’ and ‘favourite’ icons:
- are clickable toggle functions, whose appearance toggles accordingly
- when clicked on, display a toast describing the effect of the click (e.g. “Movie marked as watched”)
- may be displayed in the action bar.

Clicking on the IMDb icon launches an intent to display the corresponding (mobile) IMDb page in an external browser.

Movie Detail - Landscape



This layout has the same functionality as the Portrait layout.

Key Considerations

How will your app handle data persistence?

Movie reference data and personalised user lists will be stored using the Firebase Realtime Database. A content provider will be created to provide access to this data.
Shared Preferences will be used to persist some view preferences (e.g. list view / grid view); however, it is anticipated that there will NOT be a Settings page.

Describe any corner cases in the UX.

It is possible that the list screens will contain no items, due to an empty wishlist or no movies matching a user's filters. An appropriate message will be displayed when this is the case.

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

Picasso will be used to handle the loading and caching of images. Reason: This would very complex to implement myself. I have more experience with Picasso than any other image library.
Retrofit2 will be used to fetch and parse remote JSON/REST data. Reason: This is pretty much an industry standard which I have used before. I have also written code that does the same thing, but it was more lines of code!

Describe how you will implement Google Play Services.

Firebase Authorisation and AuthUI will be used for logging in and out. A user's personalised data, such as their wishlist (but not their security data), will be stored against their unique userid, so that the data can be persisted across multiple devices.
Firebase Analytics will be used to log key events, such as linking to IMDb.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Github:

- Create project. (done)
- Create README. (done)
- Add license info.

Firebase:

- Create project.
- Link Firebase project to app.
- Create Firebase realtime database.
- Configure database rules.
- Configure Firebase Authentication.

Task 2: App Setup

- Create Android project.

- Create data classes.
- Create content provider.
- Implement sign-in and sign-out.

Task 3: Create Admin Product Flavour

- Build UI for AdminMovieActivity and Fragment.
- Create AsyncTask for fetching data from Open Movie Database.
- Create ‘Save’ function.
- Build UI for AdminAwardActivity and Fragment.
- Create ‘Save’ function.
- Add movie and award data.

Task 4: Implement UI for Lists and Detail

Note that loaders must be used to move all data to the views.

- Build UI for AwardListActivity and Fragment (list view)
- Add AwardListActivity list item view for first item on list
- Add AwardListActivity grid view
- Add 'There are no movies matching your filters' message
- Build UI for WishlistActivity and Fragment (list view)
- Add WishlistActivity grid view
- Add 'Your wishlist contains no movies' message
- Build UI for DetailActivity and Fragment
- Add DetailActivity landscape view
- (if time) Add shared element transitions between list and detail screens

Task 5: Implement View Options for Movie List

- Create sort options UI, write options to Shared Preferences
- Create filter options UI, write options to Shared Preferences
- Add clear all filters function to filter options UI

Task 6: Implement Movie Detail Functions

- Add toggle for Wishlist flag.
- Add toggle for Watched flag.
- Add toggle for Favourite flag.
- Add link to IMDb page, in external browser.

Task 7: Add Widgets

- Add ‘In Cinemas Now’ widget.
- Add ‘Latest DVDs’ widget.

Task 8: Prepare and Submit Project

- Accessibility checks (e.g. content descriptions, D-pad navigation).
- Test RTL layouts on all screens.
- Create signed APK.
- Ensure app is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path.

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”