CPSC 312 Mobile App Development (Android)

Gonzaga University

Gina Sprint

PA6 RecyclerViews (100 points)

Individual, non-collaborative assignment

Learner Objectives

At the conclusion of this programming assignment, participants should be able to:

- Display a list of items using a `RecyclerView` and `CardViews`
- Use an `Intent` to start another activity
    - Pass information between activities using an `Intent`
- Utilize various input controls such as `Spinner` and `CheckBox`
- Interact with the user via `AlertDialogs`

Prerequisites

Before starting this programming assignment, participants should be able to:

- Utilize common Android views such as `TextView`, `EditText`, `Button`, and `ImageView`
- Organize views/widgets in a `ConstraintLayout`
- Display `Toast` messages

Acknowledgments

Content used in this assignment is based upon information in the following sources:

- None to report

Github Classroom Setup

For this assignment, you will use GitHub Classroom to create a private code repository to track code changes and submit your assignment. Open this PA6 link to accept the assignment and create a private repository for your assignment in Github classroom: https://classroom.github.com/a/tZqS-0gZ

Your repo, for example, will be named GonzagaCPSC312/pa6-yourusername (where yourusername is your GitHub username). I highly recommend committing/pushing regularly so

your work is always backed up. We will grade your most recent commit, even if that commit is after the due date (your work will be marked late if this is the case).

Overview and Requirements

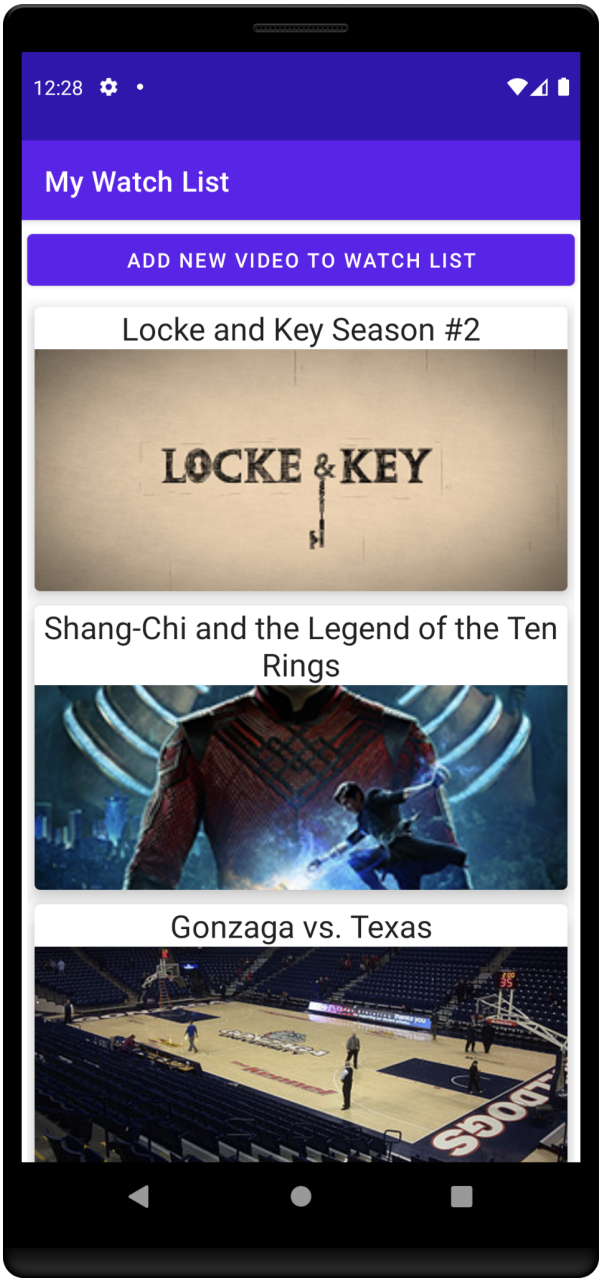
We are going to create an app called “My Watch List” for keeping track of all the awesome videos you want to watch!! A video consists of a title (string), a type (string such as “Movie” or “Series” or “Other”), a status of whether it has been watched or not (boolean), and a preview image (drawable resource ID). For example, suppose your user wants to watch the new season of Locke and Key (I know I do!!). Logically, here is how the video is represented:

Title: Locke and Key Season #2
Type: Series
Wathed: false
Preview image: id for an image like:

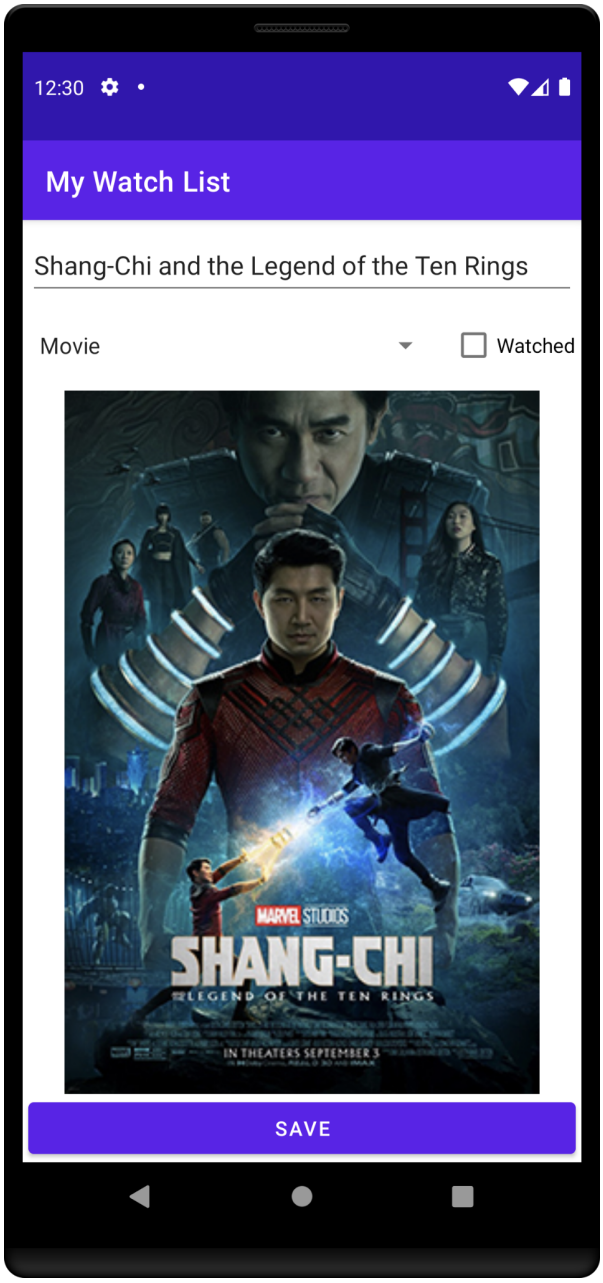
(image source: https://en.wikipedia.org/wiki/Locke_%26_Key_(TV_series)#/media/File:Locke_&_Key_(TV_series)_Title_Card.png)

Our app has two screens (activities), the `MainActivity` and the `VideoDetailActivity`. Both activities use a `ConstraintLayout` as their root layout.

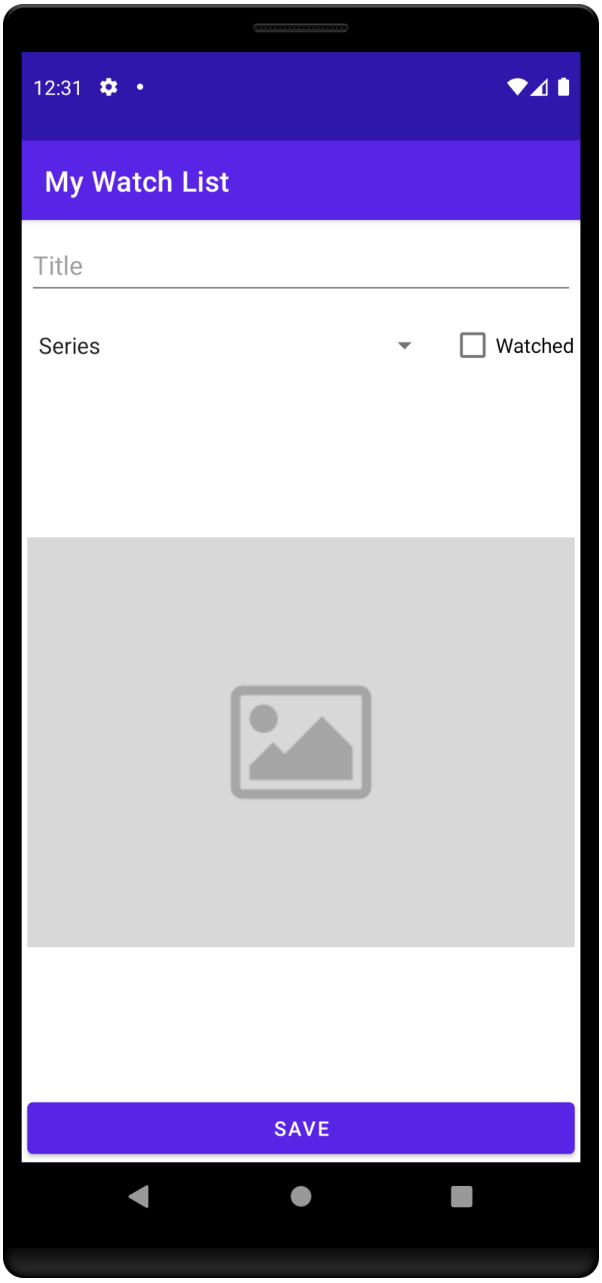
1. `MainActivity`: shows all of the user’s videos in their watch list. Includes a button to add a new video that advances the user to `VideoDetailActivity`.
2. `VideoDetailActivity`: shows the video details editor. Includes a button to stop editing the video and return to `MainActivity`.



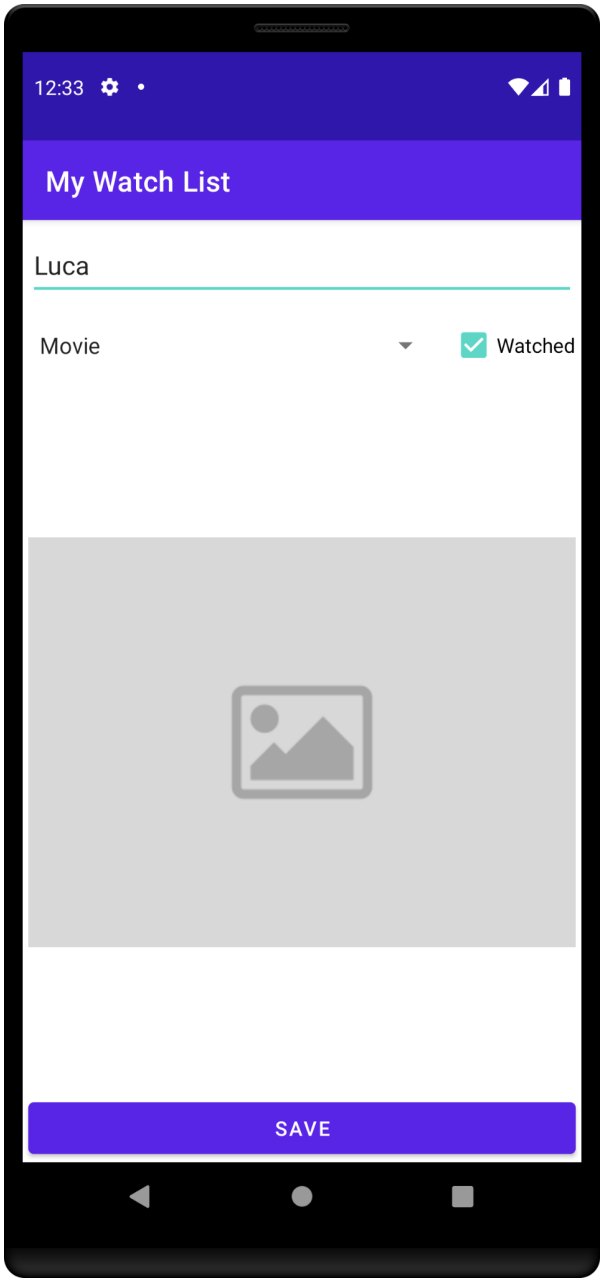
`MainActivity` with a few videos.



`VideoDetailsActivity` showing editable details for a selected video.



`VideoDetailActivity` when the user is creating a new video to watch.



`VideoDetailActivity` after the user has entered information for a new video or the user is editing an existing video.

Data Model

Remember model-view-controller (MVC) architecture? The model for this app includes a class called `Video` (in Video.java). `Video` contains at least four private fields to represent a video (see example above). We also need a `List` of `Video` objects for our `RecyclerView.Adapter<CustomAdapter.ViewHolder>`'s data source. A `List<Video>` will do the trick, but it would be great to have a class called `VideosList` (in VideosList.java) to wrap

our data structure. If you choose to implement `VideosList` (it's optional, but encouraged), it would contain at least one private field that is a `List` of `Video` objects. Regardless of your approach, pre-populate your `List<Video>` with at least three videos with unique preview images. Be sure to cite your sources!

Finally, we are going to practice an OOP design principle called data encapsulation. Only `MainActivity` has access to the `List` of `Video` objects. `VideoDetailActivity` should not access this `List` of `Video` objects. `VideoDetailActivity` is only concerned with creating/editing a single video. Thus, any relevant video data is passed between the two activities via `Intent` objects.

MainActivity

`MainActivity` is the first `Activity` your user interacts with. `MainActivity` has the following GUI components (at a minimum):

- A `Button` for the user to press to add a new video. Clicking this button advances to the next screen.
- A `RecyclerView` to display the user's videos. A video should be displayed in the `RecyclerView` using a `CardView` containing the video's title and preview image.

Additional details

- Clicking on an item in the `RecyclerView` should open `VideoDetailActivity` to edit the video's details.
- Long clicking on an item in the `RecyclerView` should open an `AlertDialog` prompting the user to delete the video or not. Handle the user's selection accordingly.

VideoDetailActivity

`VideoDetailActivity` is the `Activity` where your user creates and/or edits a video. `VideoDetailActivity` has the following GUI components (at a minimum):

- An `EditText` for the video title
- A `Spinner` for the video type
    - Video types are fixed to at least "Series", "Movie", or "Other"
    - Note: feel free to add more if you'd like!
- A `CheckBox` for whether the video has been watched or not
- An `ImageView` for displaying the video's preview image
    - Note: the user won't be able to add an image to a new video, that's okay
    - Use a placeholder image (here is the one I used)
- A `Button` for the user to press when they are done creating/editing the video. Clicking this button returns to the previous screen.

Additional details

- Allow the video preview `ImageView` to be as large as possible
- Validate the user enters a non-empty title. Inform the user accordingly with a `Toast`

Bonus (5 pts)

- (3 pts) Ensure the user enters unique video titles. Implementing this feature should not violate the data encapsulation design we have set up for our activities.
- (2 pts) Allow the user to search for streams for a given video using a web search app. To do this, add a neutral button for searching to the delete video `AlertDialog`.

Submitting Assignments

1. Use Github classroom to submit your assignment via a Github repo. See the "Github Classroom Setup" section at the beginning of this document for details on how to do this. You must commit your solution by the due date and time.
2. Before your final commit, go to Build->Clean Project. Your repo should contain your entire Android Studio project. Double check that this is the case by cloning (or downloading a zip) your submission repo and opening your project in Android Studio and running your code.

Grading Guidelines

This assignment is worth 100 points + 5 points bonus. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 15 pts for correct data model
- 5 pts for implementing a design that supports data encapsulation
- 40 pts for `MainActivity`
    - 5 pts for ConstraintLayout with no warnings
    - 5 pts for correct video GUI components
    - 5 pts for add video button passing appropriate data and starting `VideoDetailActivity`
    - 10 pts for supporting video editing on `RecyclerView` item click
    - 15 pts for supporting video deleting via an `AlertDialog` on `RecyclerView` item long click
- 30 pts for `VideoDetailActivity`
    - 5 pts for ConstraintLayout with no warnings
    - 10 pts for correct video details GUI components
    - 10 pts for save/done button passing appropriate data and returning to `MainActivity`
    - 5 pts for validating video title is not empty and informing the user accordingly
- 10 pts for adherence to proper programming style and comments established for the class
    - Note: Throughout the semester I am going to update the style guidelines document as we learn more about Android conventions. Always take a look at it before you submit.