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: akitikkx

TV Central

Description

With all the shows airing on television it is often difficult to keep track of one's favorite shows as well as discover new ones.

The *TV Central* app aims to aide the TV show enthusiast in keeping track of what's new on television, what's popular and what's currently airing. The user can view more information about each show listed, from synopsis, episode information, similar shows, content ratings, images, acting credits to videos. All TV show data in TV Central will be provided by *TMDB*'s extensive API. The *TV Central* app is a ground-up inspiration of *Upnext TV Series Manager* which I built before exposure to Udacity's Android Developer Nanodegree program. *TV Central aims* to improve on *Upnext*'s lacking elements such as UI (proper implementation of Material Design for both phones and tablets), data, and efficient background syncing.

Intended User

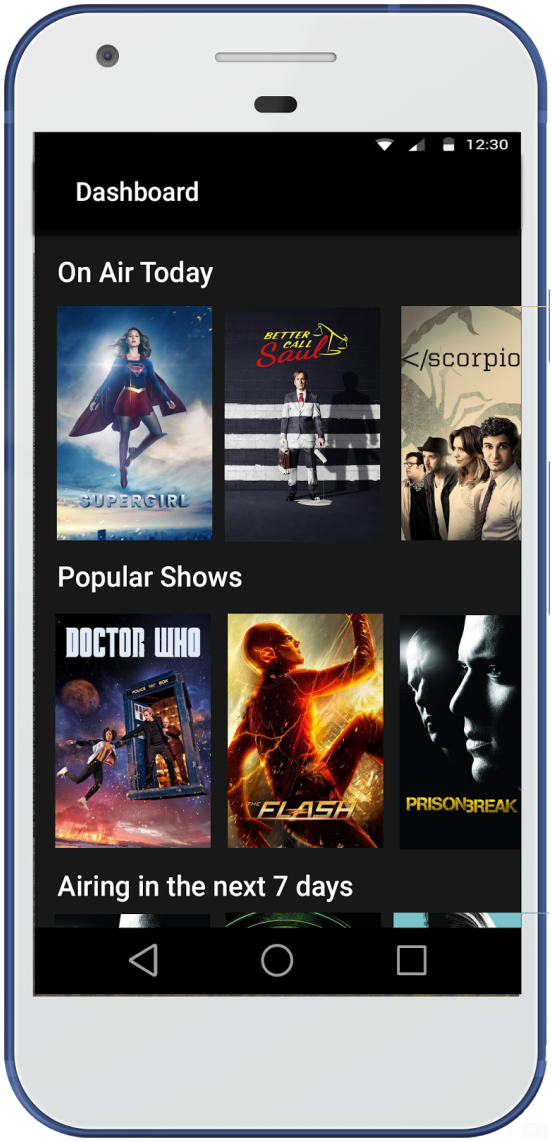
TV Central is for anyone who likes to keep track of their own TV shows as well as discovering new ones.

Features

- Dashboard that contains different lists that are horizontally scrollable to help the user easily discover shows
- Lists are cached on the user device and updated periodically via a background SyncAdapter to allow the user to have a smooth experience as well as save on mobile data (where mobile data usage is a concern), allowing the user to use the app in offline mode. Data will be read from the user's device database via a Content Provider
- Users can add a show to their list of favorites via the detail screen. The show will be stored locally in the device database. Users can easily view these on the *Dashboard* screen
- Each listed show contains synopsis, episode information, similar shows, content ratings, images, acting credits, videos such as season trailers which can be played inside the *TV Central* app.
- Widget which can be added to the user's home screen that displays shows airing on the current day

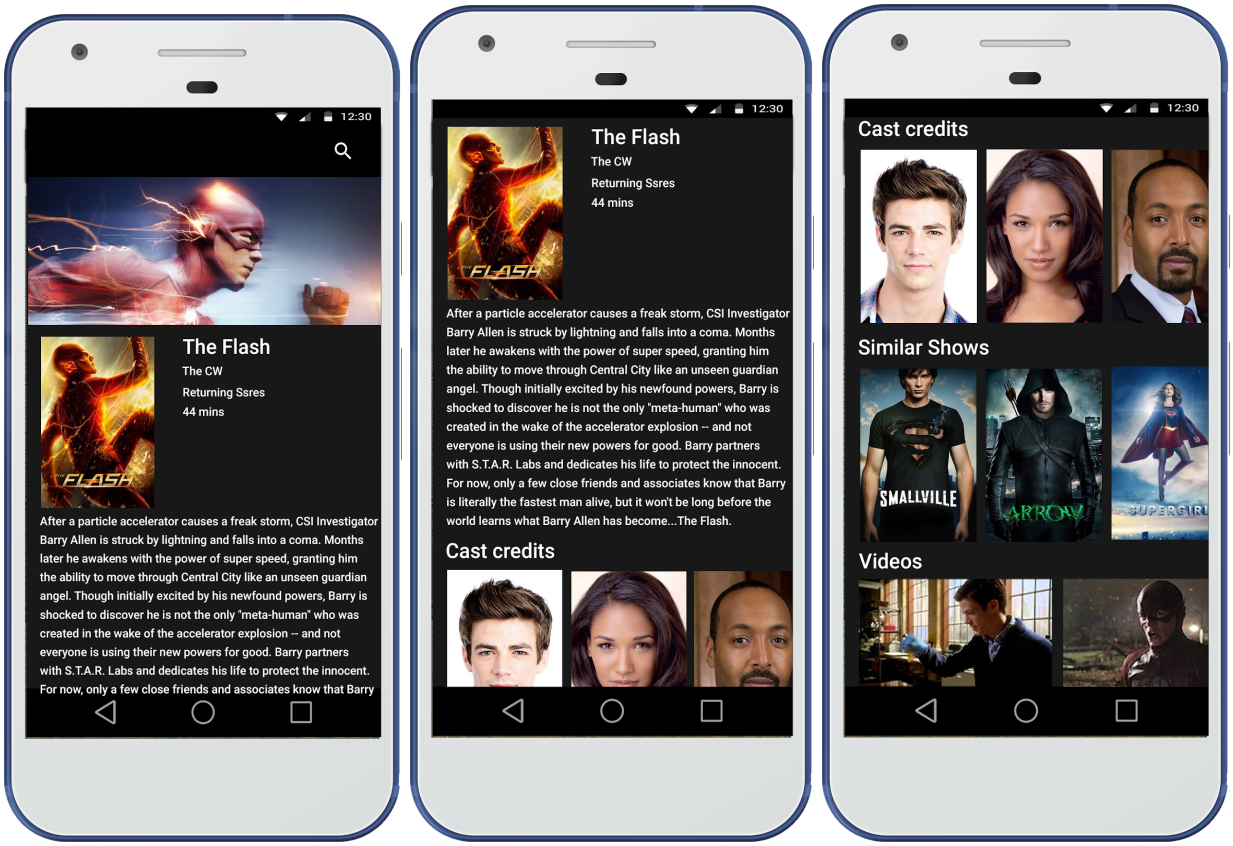
User Interface Mocks

Screen 1



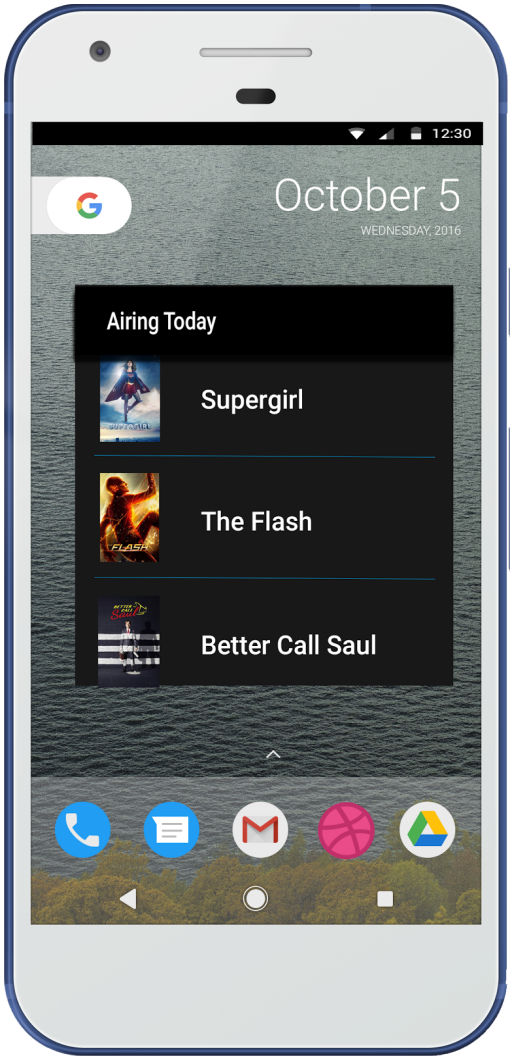
The *Dashboard* screen displays different lists at a glance, the user can view each list by scrolling vertically. Each row, for example "*Popular Shows*" is horizontally scrollable. The user can click on the poster to view more information about the selected show. The information will be retrieved from the device database via the Content Provider.

Screen 2



The *Detail* screen will display information about the selected show. The information will be retrieved from the device database via the Content Provider.

Screen 3



The home screen widget to display shows airing on the current day. The information will be retrieved from the device database via the Content Provider.

Key Considerations

How will your app handle data persistence?

Data will be stored locally on the device sqlite database and retrieved via a Content Provider. The data will be updated in the background at regular intervals via a SyncAdapter.

Describe any corner cases in the UX.

The current instance state will be stored and when the user goes back they will be taken to the exact position they were before they viewed the details of the selected show. A shared element transition animation will be employed to enhance the visual experience for the user. When the user clicks back from the dashboard screen, the app will be closed.

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

- Google Design Library to help leverage the use of Material Design components and patterns.
- *Glide* to handle the loading and caching of images.
- *Retrofit* will be used to connect in the background to the various services such as TMDB's API
- *OkHTTP* to be used as the HTTP client in conjunction with Retrofit
- *Butterknife* for view injection, code will also be neater

Describe how you will implement Google Play Services.

- *Firebase Cloud Messaging (FCM)* will be used to deliver messages in the form of notifications to the user pertaining to show changes or server updates.
- *Firebase Crash Reporting* will be implemented to help address any issues experienced by the users and address them
- *Google/Firebase Analytics* will be used to help determine user behaviour to help determine what users are mostly after when they are using *TV Central*, thereby introducing new features or improving on current features if need be.

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

- Create *TV Central* project base in Android Studio and configure permissions in the AndroidManifest file.
- Create project *TV Central* in Firebase Console and setup google-services.json in app module

- Create theme for *TV Central* that extends *AppCompat*
- Create app singleton for *TV Central* which will be used to initialize *Firebase Analytics* and Firebase Crashlytics
- Add libraries in app build.gradle
- Add packages for *ui*, *data*, *sync*, *adapters*, *api*, *widget*, *fcm*, *services*
- Create custom Content Provider for TV Central with contract and DB Helper models to be stored in the *data* package
- Create SyncAdapter to be stored in the *sync* package to pull data from TMDB at regular intervals and store it in the sqlite database.
- Create Adapter that extends *RecyclerView.Adapter<ViewHolder>* which will be used to display the show lists. Add Callback interface to allow the fragment to handle the onClick event to direct the user to the detail screen.
- Setup API model to be stored in the *api* package that employs *Retrofit* to connect to the *TMDB* API service.

Task 2: Implement UI for Each Activity and Fragment

***DashboardActivity* and *DashboardFragment*:**

- Build UI for the launcher activity *DashboardActivity* and *DashboardFragment* for displaying the lists - *On Air Today*, *Popular Shows*, *Airing in the next 7 days* and *Favorites.*
- UI should include support for accessibility
- *DashboardFragment* should implement *LoaderManager.LoaderCallbacks<Cursor>* to retrieve the lists from the database and display them to the user via RecyclerView Adapter.
- Save the current instance state.
- Handle error cases in *DashboardFragment* such as connectivity or server issues and display relevant empty state screens.

DetailActivity* and *DetailFragment

- Build UI for the *DetailActivity* and *DetailFragment* for displaying the detail for the selected show.
- UI should include support for accessibility
- DetailFragment should implement *LoaderManager.LoaderCallbacks<Cursor>* and load the information from the database. *Glide* is to be employed for loading the images. The *YouTube* API will be used to play the show videos when the user clicks on the trailer thumbnail in the view.
- Setup YouTube API to play the selected thumbnail from within the detail view
- Handle error cases in *DetailFragment* such as connectivity or server issues and display relevant empty state screens.

Task 3: Build

- Configure the release buildType with its signingConfig path configured. App should build and deploy using the *installRelease* Gradle task.
- Path to keystore file should be relative

Task 4: Widget

- Create widget to display the shows airing on the current day
- When a show is selected the user will be redirected to the app

Task 5: Notifications

- Configure Firebase Messaging as per the documentation and store required classes in the *services* package.

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**"