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: ASheler

SpaceXploration

Description

Keep up with SpaceX news, go through past launches and track closest upcoming launch. All in one simple to use app.
App provides detailed info and statistics of SpaceX rocket launches, SpaceX itself and even launchpad Data.

Intended User

From SpaceX hardcore fans to everyone who is interested in SpaceX launches, info and details. There are no age, religion, nationality or race restrictions in using of the app. App will be created with English localization as per Api sources, with extracted Strings for possible future localizations.

Features

- Gets newest data of SpaceX launches
- Provides detailed information about SpaceX vehicles, sites, missions
- Provides images and videos of rocket launches

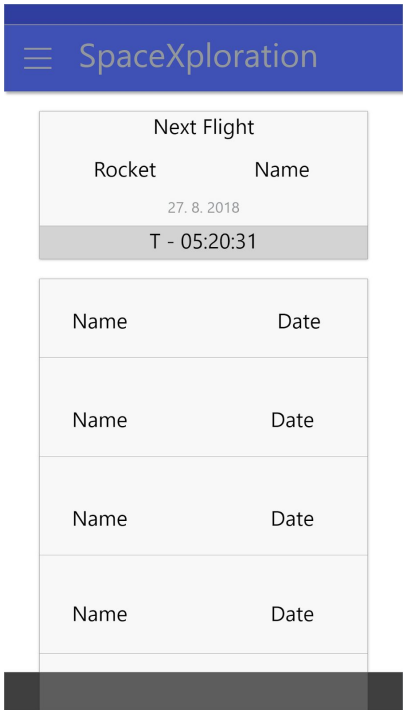
App Building

- App will be written solely in Java Programming Language
- App will be written in Android Studio 3.1.3, using Gradle v. 4.8.1

User Interface Mocks

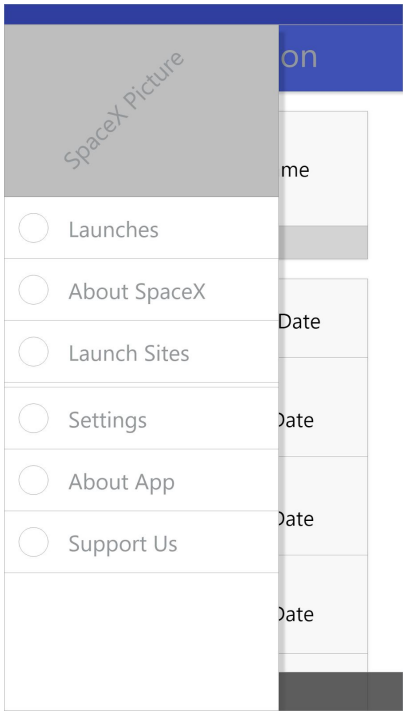
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



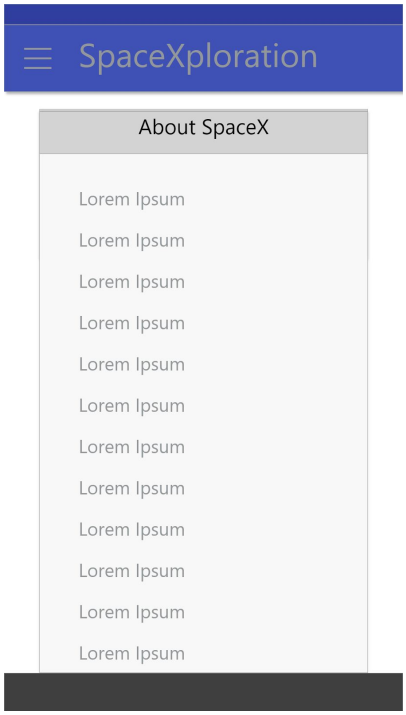
Default screen upon loading app. Screen shows RV with launches (first view is different with info about upcoming flight). On left side of AppBar is Hamburger icon for Navigation Drawer. Clicking each flight brings brand new activity with detail of the flight

Screen 2



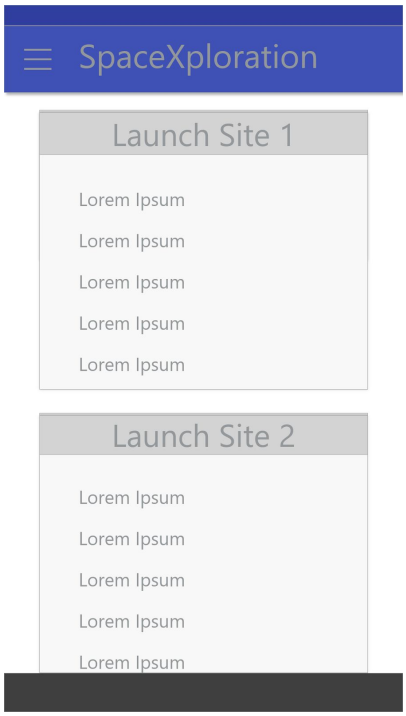
Navigation drawer with SpaceX picture on top, 2 menu selection groups. First group opens new Fragment and closes the drawer, second group opens brand new Activity.

Screen 3



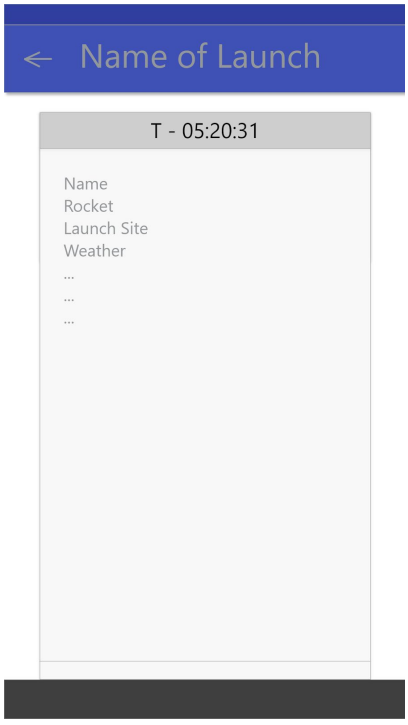
About SpaceX fragment, showing cardView with All info about Spacex

Screen 4



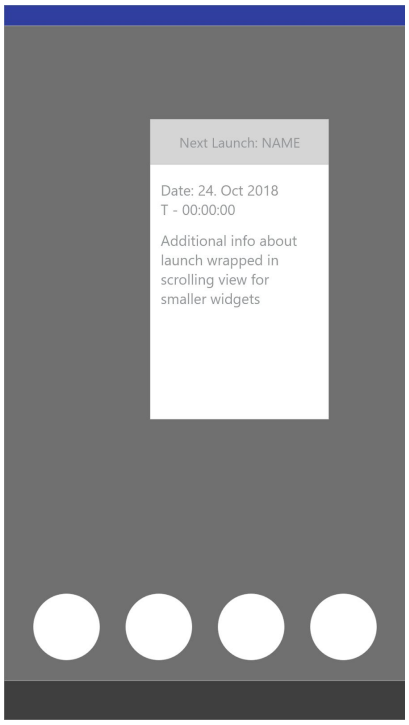
Launch Sites info fragment - RecyclerView with info cardViews of each launch site

Screen 5



Flight Detail Activity. Simple Activity with info about flight.
Top left AppBar has back button for smooth navigation

Screen 5



App Widget, providing info about next Launch. On click on widget the user is shown Launch Detail Activity

Add as many screens as you need to portray your app's UI flow.

Key Considerations

How will your app handle data persistence?

Data will be acquired from r-SpaceX api and stored locally via room for offline access. I will save the data to room DB via AsyncTask after receiving from API call so it won't block UI thread. Data will be accessed via LiveData and ViewModels for persistence and continuity.

Describe any edge or corner cases in the UX.

Detail activities will have parent activity set up properly, so hitting back button or back arrow will lead user back to expected activity and state.
In No Internet connection state, the App will try to get data from local db - if user have data stored, they will be shown with proper snackbar with info about connection. If user have no data locally, there will be message saying there is no connection.

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

Android support library - backward compatibility (v. 27.1.1)
Android design library - material design components & other design futures (v. 27.1.1)
Retrofit - api work (v. 2.4.0)
ButterKnife - faster and easier views and resources lookups (v. 8.8.1)
Gson - creating TypeConverters (v. 2.8.5)
Room - store data from api to db for offline access (v. 1.1.1)
Picasso - loading images from web (v. 2.71828)
Google analytics - user analytics (v. 16.0.1)
Google mobile ads - monetization of the app (v. 15.0.1)
LifeCycle - data persistence, work with room data (v. 1.1.1)

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

Google Analytics for usage analytics
Google Mobile Ads for displaying ads

Resources management

There will be no hardcoded strings. All strings, colours etc will be stored in their equivalent resource xml files (e.g. strings.xml)

Accessibility

- RTL layout will be enabled and the usable in all cases
- All images will have Content description
- Text will use SP instead of DP
- App will adhere to accessibility standards as per *https://material.io/design/usability/accessibility.html*

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

- Set up project, set name, build tools version, min and target sdk versions
- Implement all external libraries
- Setup permissions
- Setup vcs for project

Task 2: Implement UI for Each Activity and Fragment

- Build main UI for MainActivity - create AppNav drawer, create menu
- Build UI for all fragments (launches, history, launch sites, …)
- Build UI for other activities - launch detail, settings, about...

Task 3: Connect with world

- Make Retrofit api methods
- Create Room db and classes
- Create LiveData calls where needed
- Connect Retrofit with Room

Task 4: Create RecyclerView Adapters

- Create adapters for RVs, taking data from Retrofit and/or Room
- Set Adapters for RVs

Task 5: Polish UI, Implement Ads

- Polish UI, implement Material design guidelines
- Implement Ads to desired Activities

- Create possible different layouts for different sizes

Task 6: Implement Analytics

- Implement Google Analytics

Task 7: Test

- Test UI, connection to api, db
- Test on various screens and devices

Task 8: Polish code, release

- Go through code, make sure everything is readable and commented
- Optimize inputs, delete unused resources, methods, …
- Release, Submit for review

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**"