Capstone Stage 1

Table of Contents

GitHub Username: nkrusch

SpaceLaunch 1 - Rocket Launch Schedule

Description

This app allows user to stay informed about upcoming spaceflights by displaying information about upcoming rockets launches.

Intended Users

- people living near launch sites
- astronomy students and enthusiasts

Features

Technical Specs (required)

- ☐ App is written solely in the Java Programming Language
- ☐ App keeps all strings in a strings.xml file
- ☐ App enables RTL layout switching on all layouts
- ☐ All images include content descriptions
- ☐ App is navigable using keyboard and dpad

MVP features (required)

- ☐ List upcoming rocket launches
- ☐ Shows details about each launch event
- ☐ Countdown to next upcoming launch (both in app and in widget)

Nice-to-have

- ☐ Compute and show distance from my location to launch site
- ☐ Ability to filter launches that are within N km radius from my location
- ☐ Filtering launch events based on location, mission, rocket, agency etc.
- ☐ Display notification to user *N* days/hours/min before launch occurs
- ☐ FAB button to share launch event with friends

User Interface Mocks

Main activity

The main activity prominently displays the next upcoming launch and a list of other upcoming launches in chronological order.

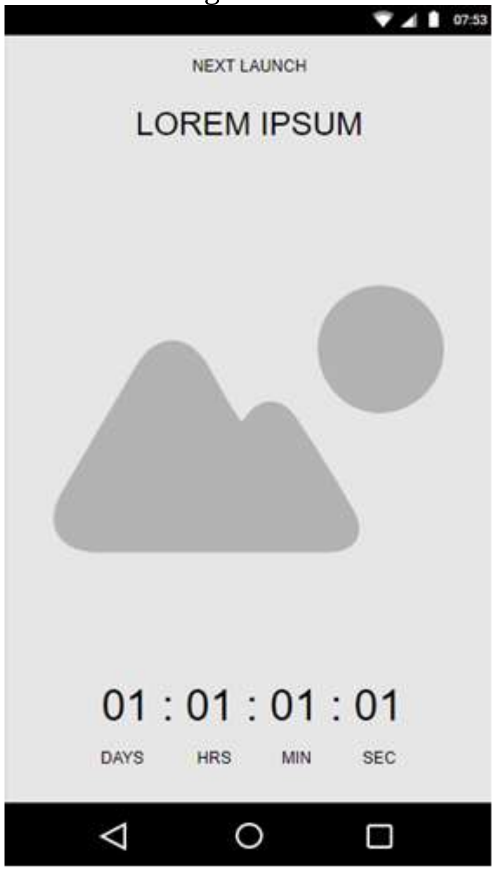
The application will use fragments that can be arranged differently in mobile and tablet layouts.

On mobile device screens 1 and 2 may be arranged vertically so Screen 2 becomes visible when user scrolls down. Screen 1 may be a contained in a collapsing layout or subtle hints may be utilized to indicate that content is scrollable.

On tablet device the two fragments can be arranged side-by-side. The tablet layout could also use an image list for screen 2.

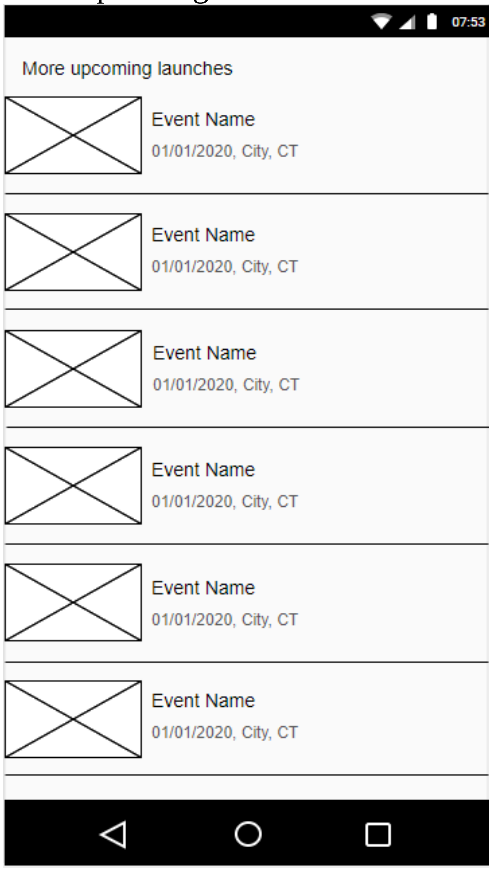
Screen 1

Next launch fragment



Screen 2

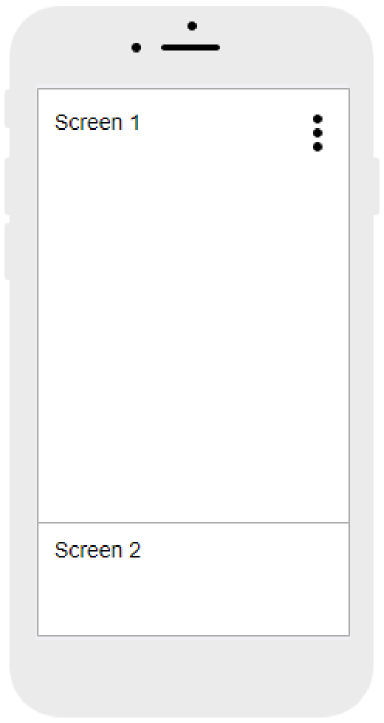
More upcoming launches

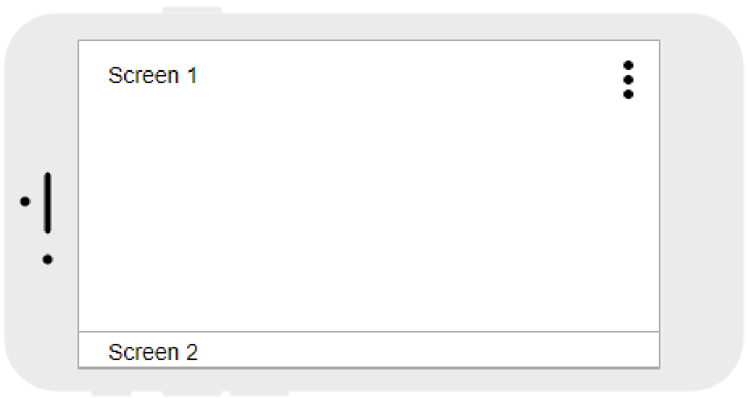


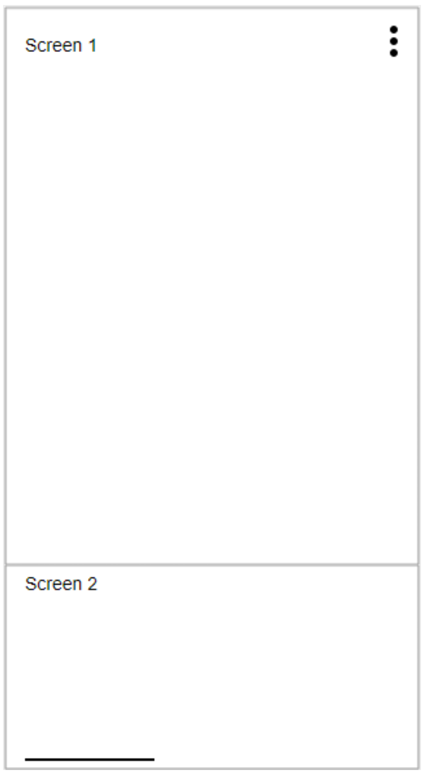
Clicking anywhere on Screen 1 or on a list item will transition user to the details activity showing more information about the selected launch event.

Fragment arrangement

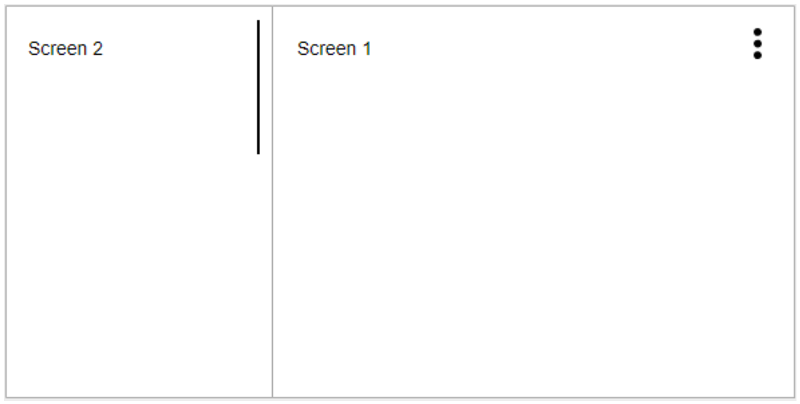
Mobile – Screen 1 should have height slightly less than available viewport height to indicate more content will follow on scroll.







Tablet layout



Details activity

Details activity shows details about a specific launch event and a countdown to that event (Screen 3).

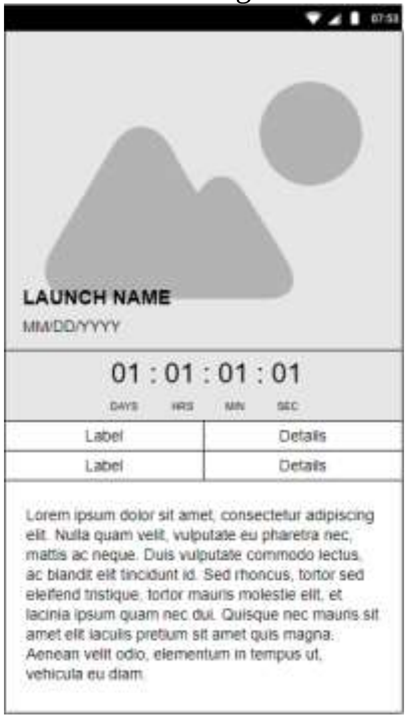
The launch event location (lat/long) is used to generate a map of the launch site (Screen 4). This map can fill the visible space in mobile view and be interactive so user can zoom in and out on the map.

Third fragment (Screen 5) depicts images related to the launch event. These images may include images of the rocket, launch site, or other related images or video returned by the referenced APIs.

The fragment organization should change depending on viewport size:
- on mobile device the fragments may be arranged in a viewpager/tabbed layout.
- on tablet device the fragments may be arranged to take up the available space (no tabs)

The details activity may also include a FAB button to enable sharing the launch event. Details activity should have a toolbar with arrow back to take user back to main activity.

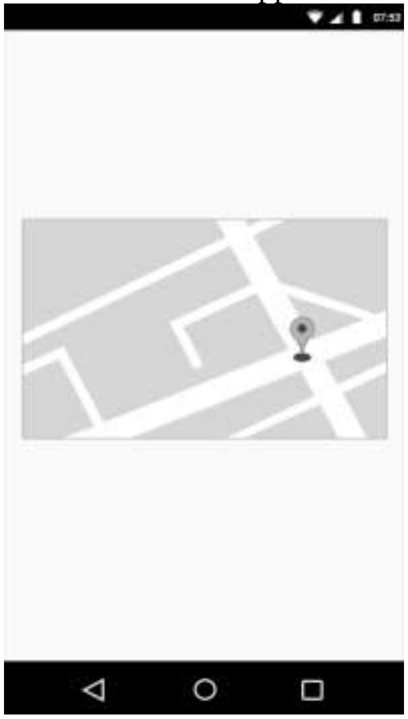
Screen 3
Launch details fragment



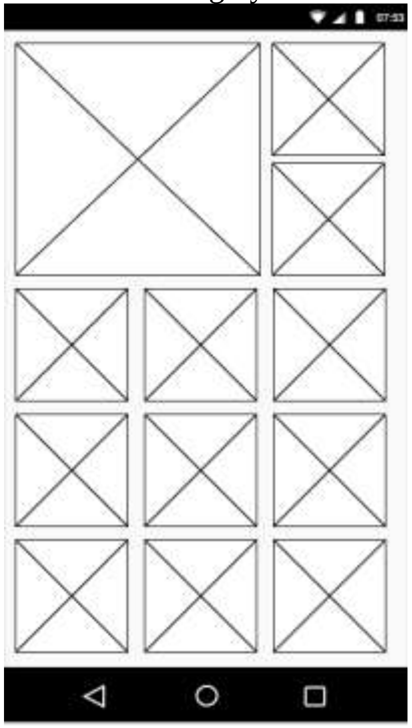
Label	Details
Label	Details

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam.

Screen 4
Launch location mapped

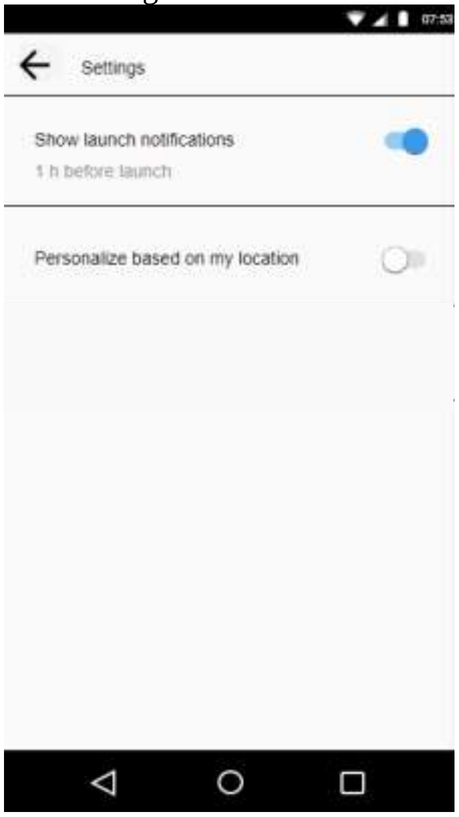


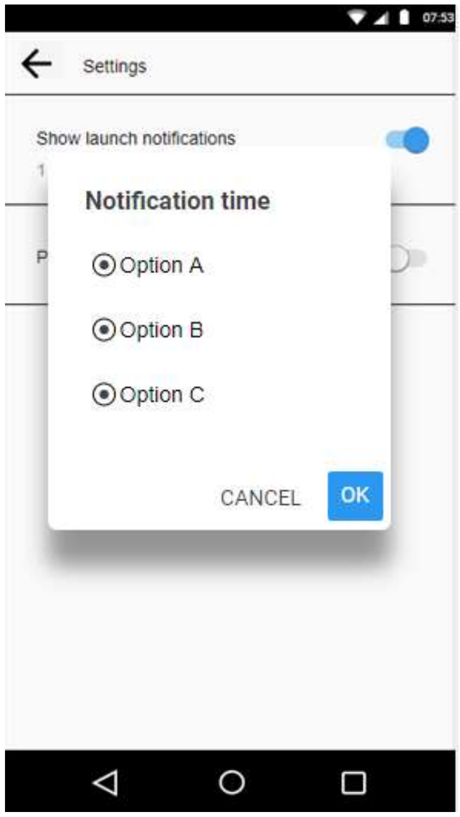
Screen 5
Launch site imagery



Screen 6

User settings



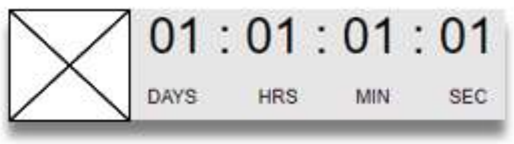


User enters settings by clicking on menu icon in top right corner of main activity. Left arrow takes user back to main activity.

Toggling launch notification will show modal allowing user to pick notification delay.

Location option is a nice-to-have; *if* implemented and enabled by user, the user could see: 1.) computed distance to launch site from their current location and, 2.) ability to filter launch events that are within N km radius from user's location.

Widget



The home screen widget shows a countdown to next launch event.

Key Considerations

How will your app handle data persistence?

While the launch data from the API updates periodically, the projected interval of changes is over several days. There is no need to constantly check the API for updates, therefore app will check for updates on app start and on demand (when viewing launch details), if device is online.

App will read data from a network API using an AsyncTask. These reads include getting a list of launches and details on individual launches. These requests are expected to be short in duration. The app will also store the API response locally using Room to make data available when device is in offline state.

SharedPreferences will be used to save user settings.

Describe any edge or corner cases in the UX.

If there are no upcoming launches or data is not available, app will provide suitable empty state messages to explain this situation to user.

If user is offline, best effort will be made to display content to user. However, the maps and images in details view would be unavailable. Launch event main image and textual information can be saved locally to keep user experience relatively similar between online and offline state. Snackbar or icon should be utilized to indicate to user that some features are unavailable offline.

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

Picasso
This library will be used for loading and caching images. Based on previous experience this library simplifies image handling significantly. This application will use powerful imagery so having a good image handling library is essential.

Retrofit and gson converter
This library will be used for reading data from API. Using this library speeds up implementation and reduces the chance of bugs that might occur from writing a custom json parser.

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

Launch library API
https://launchlibrary.net/docs/1.4/api.html
This is the main application data source where the app will get information about upcoming launches

Google Maps API
https://developers.google.com/maps/documentation/android-sdk/intro
This maps api will be used to create static maps with pins to show launch site to user

Google Places API
https://developers.google.com/places/android-sdk/intro
This API will be used to discover related imagery that is near the launch site (if available).

Next Steps: Required Tasks

Task 1: Project Setup

- ◻ Create project and setup git repo
- ◻ Configure backend dependencies, e.g. retrofit and arch.persistence libraries
- ◻ Create API models
- ◻ Connect to API and ensure reading from API succeeds
- ◻ Create local database and viewmodel(s)

Task 2: Implement UI for Each Activity and Fragment

- ◻ Configure UI libraries: picasso, material design support
- ◻ Create fragment showing next upcoming launch
- ◻ Create fragment listing upcoming launches
- ◻ Create details fragment for a specific launch
- ◻ Create settings for customizing user preferences
- ◻ Implement empty states when no data is available

Task 3: Integrate with Google Maps API

- ◻ Get API key
- ◻ Generate static map with pin to indicate launch site
- ◻ Show this map view in the details fragment

Task 4: Integrate with Google Places API

- ◻ Given known launch site, use places api to discover points of interest nearby using related keywords such as “space”
- ◻ Based on API response, see if any response item contains pictures, then show those pictures to user

Task 5: Add Widget

- ◻ Create home screen widget which shows next launch countdown

Task 6: Add Tests

- ◻ Use espresso to create UI tests to test main app functionality

Task 7: App Signing

- ◻ Create keystore and configure app signing as described in rubric