GitHub Username: curtisgetz

Mars Explorer

Description

Mars Explorer allows you to explore 'The Red Planet' as a whole, or through the eyes of one of the three NASA rovers currently on the surface.
Learn about each of the three rovers and the experiments they conduct on the surface of Mars.
View the extensive library of pictures the rovers have taken.

Intended User

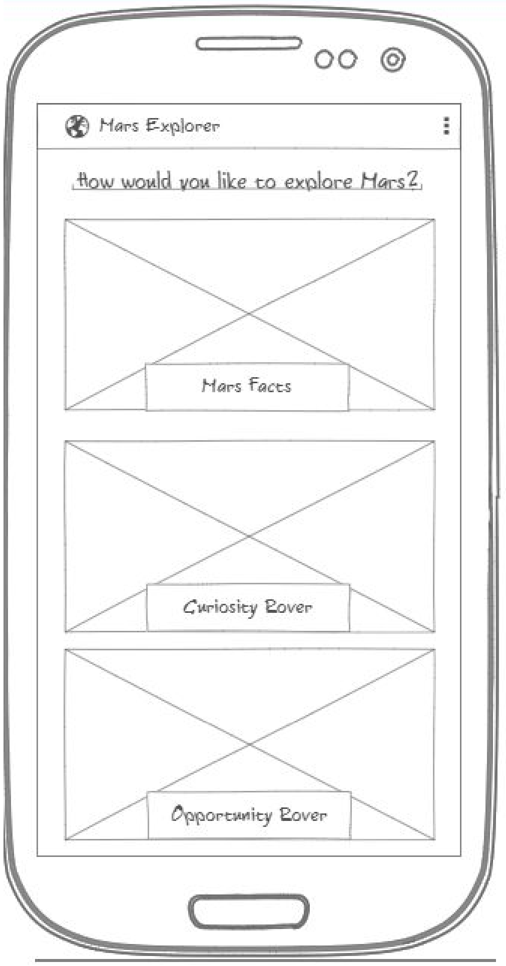
Science students or space enthusiasts.

Features

- Displays pictures
- Save/Share pictures
- Displays information

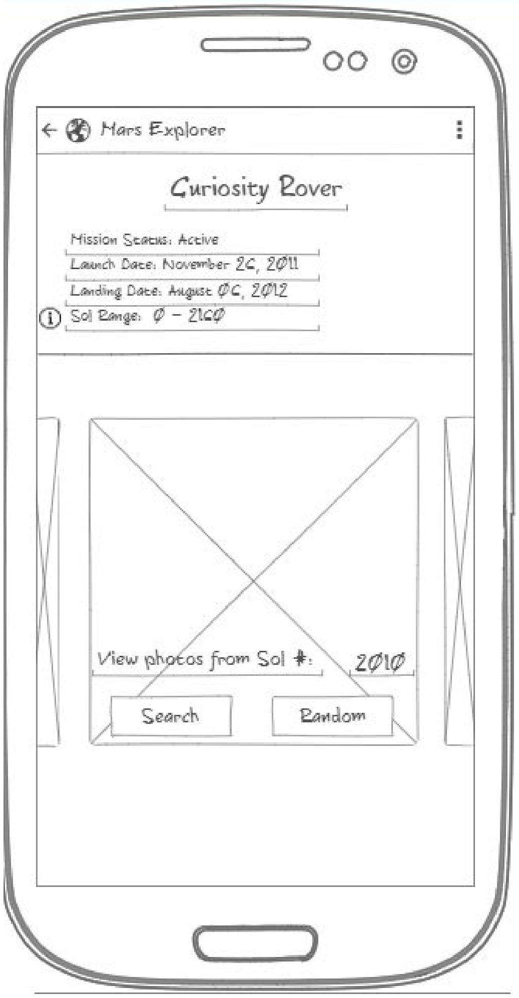
User Interface Mocks

Screen 1



Main screen allows user to select to explore Mars as a whole, or through one of the three rovers.

Screen 2



Curiosity Rover

Mission Status: Active

Launch Date: November 26, 2011

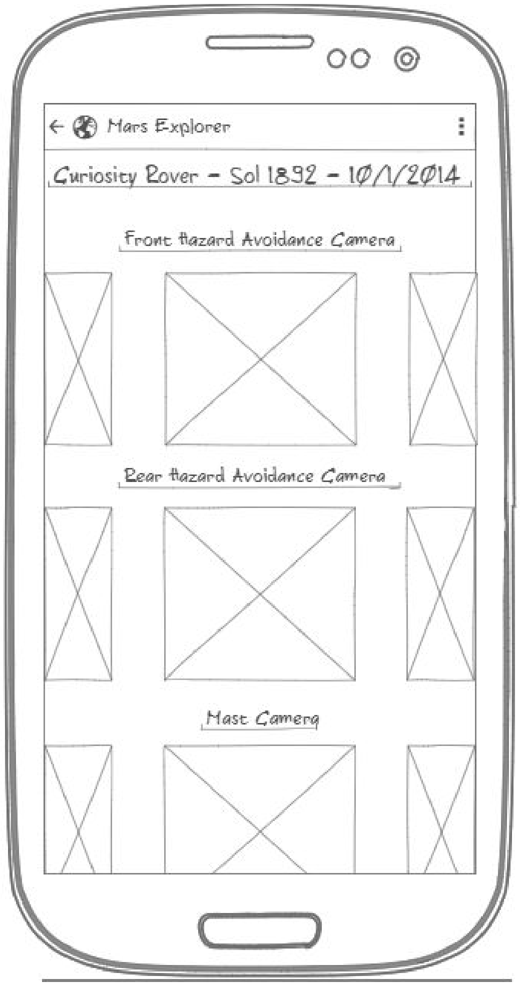
Landing Date: August 06, 2012

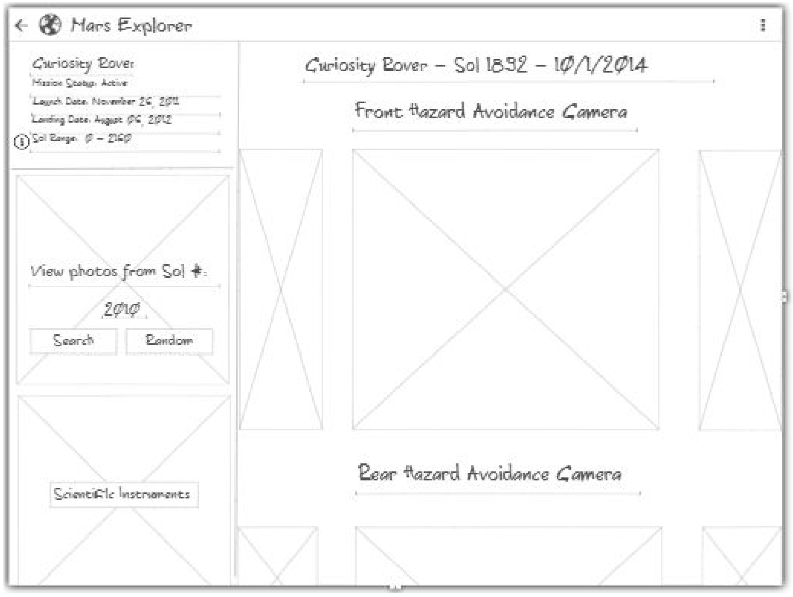
Sol Range: 0 – 2160

View photos from Sol #: 2010

Rover detail will show mission details at the top and allow user to select an option below. Either photos, rover equipment info, or other details about that specific rover.

Screen 3

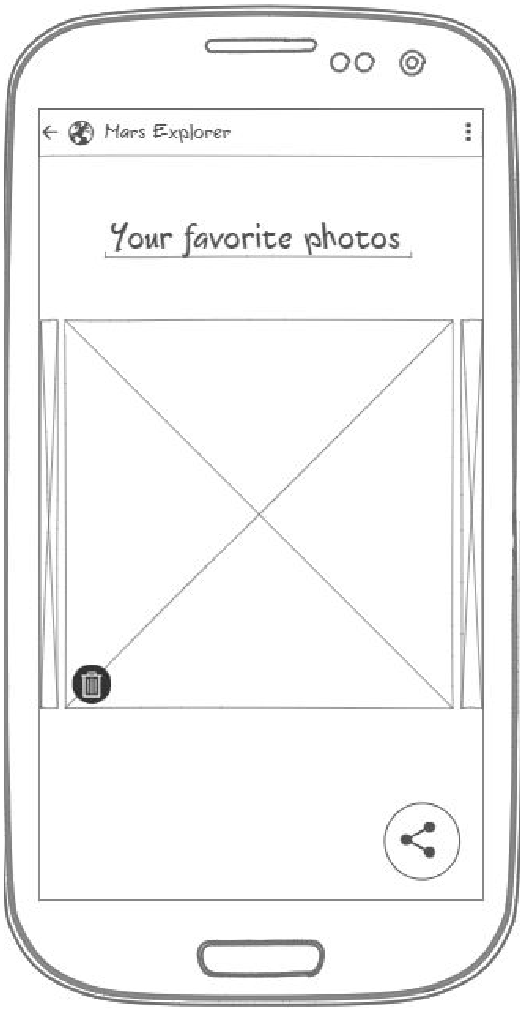

Mars Explorer
Curiosity Rover – Sol 1892 – 1Ø/1/2Ø14
Front Hazard Avoidance Camera
Rear Hazard Avoidance Camera
Mast Camera


Mars Explorer
Curiosity Rover
Mission Status: Active
Launch Date: November 26, 2011
Landing Date: August Ø6, 2012
Sol Range: Ø – 216Ø
View photos from Sol #:
2Ø1Ø
Search
Random
Scientific Instruments
Curiosity Rover – Sol 1892 – 1Ø/1/2Ø14
Front Hazard Avoidance Camera
Rear Hazard Avoidance Camera

(Screen 3 cont)

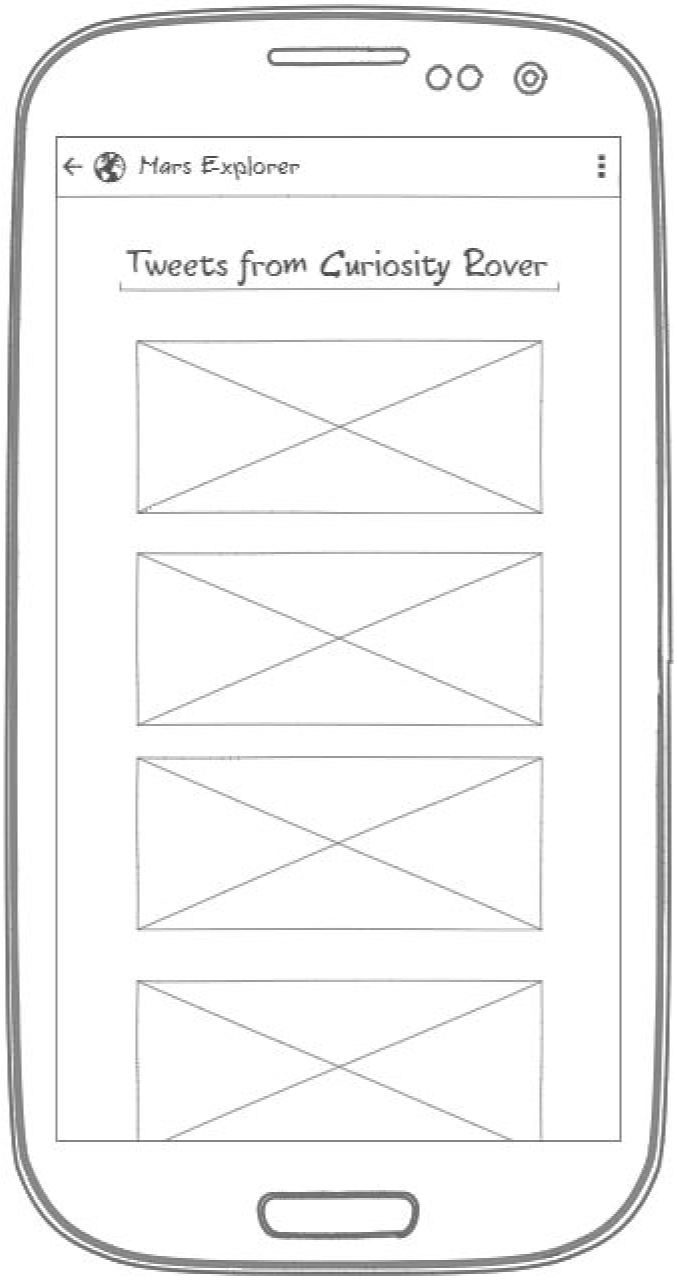
When a sol/date is chosen, UI will display available cameras and photos. Tablets in landscape will use a Master/Detail flow for rover info, including photos.

Screen 4



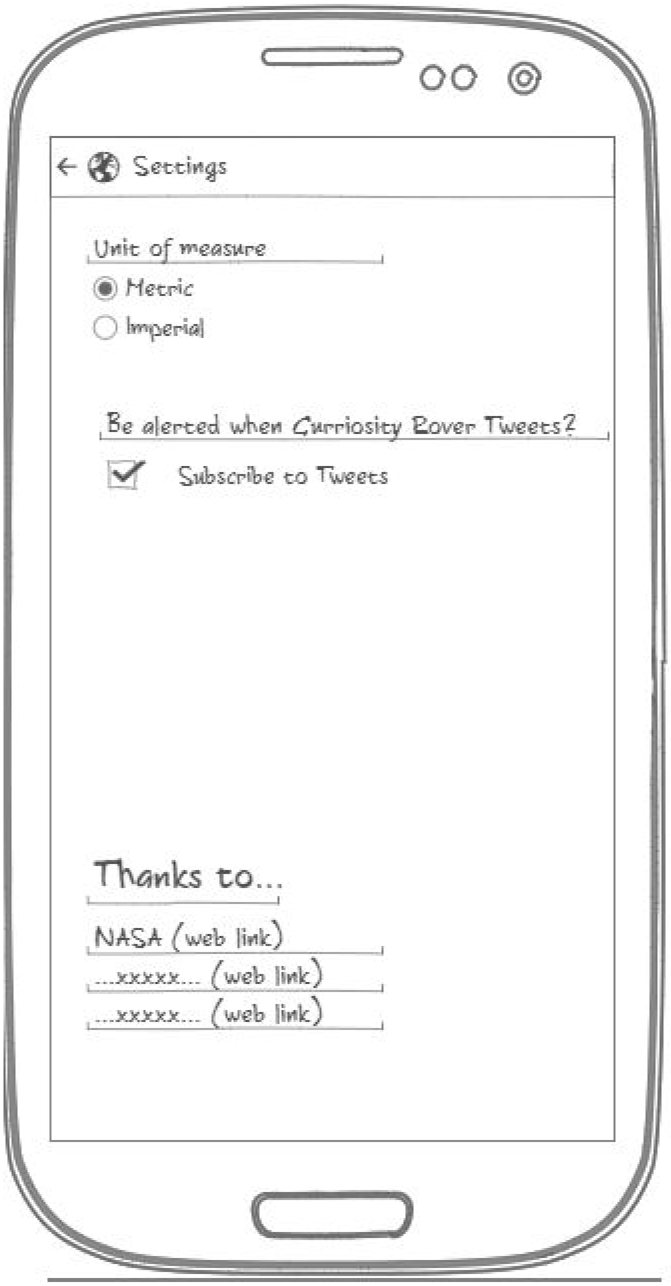
Users can save their favorite photos and view them all on one screen. UI will allow users to share and delete their favorites.

Screen 5



This screen will display the latest Tweets by the Curiosity Rover.

Screen 6



Unit of measure

● Metric

○ Imperial

Be alerted when Curriosity Rover Tweets?

☑ Subscribe to Tweets

Thanks to...

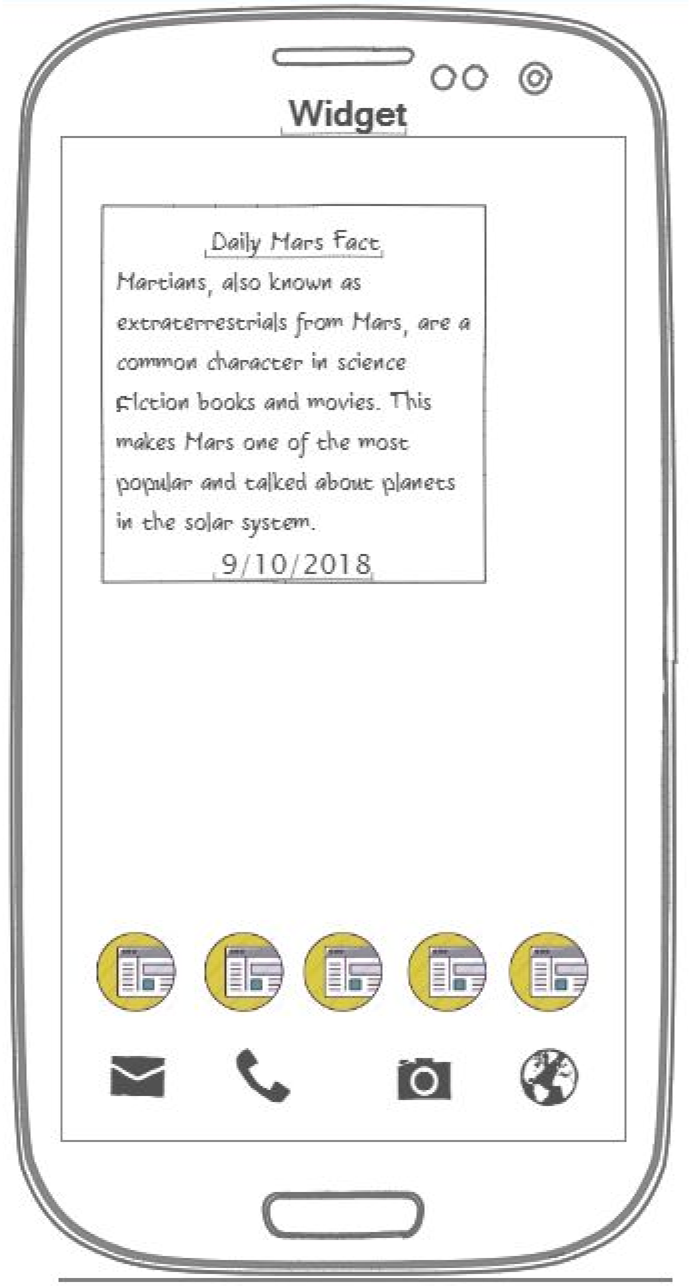
NASA (web link)

...xxxxx... (web link)

...xxxxx... (web link)

A settings screen will have user configurable options and credits.

Screen 7



Daily Mars Fact

Martians, also known as extraterrestrials from Mars, are a common character in science fiction books and movies. This makes Mars one of the most popular and talked about planets in the solar system.

9/10/2018

Widget will display a new fact daily. The Fact class will have a ‘shortFact’ property. This will be a short summary of the fact if it would be too long to be displayed in the widget. If the Fact is short enough the full fact will be displayed in the widget. In either case, a click on the widget will open the app and display the full Fact.

Key Considerations

How will your app handle data persistence?

User favorite photo urls will be stored in a Room database.
Curiosity Tweets will be stored in a Room database
I will use Firebase Realtime DB to store and serve Mars Facts to the app.

Describe any edge or corner cases in the UX.

n/a

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

- Picasso will handle loading and caching of images
- Butterknife will be used for View and Click binding

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

- FirebaseRemoteConfig for API endpoints to handle future changes
- Firebase Cloud Messaging to send message when Curiosity Rover Tweets (Manually for now, would need to develop a backend later for automation)
- Firebase Realtime Database for Mars Facts

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.

Preparation:

- App will be written solely in the Java Programming Language
- All strings will be stored in strings.xml and enables RTL layout switching on all layouts
- App will use an AsyncTask to perform image searches through NASA API
- App may use an IntentService to fetch data & show it on widget
- LiveData & ViewModel are used with Room and no unnecessary DB calls are made
- App will utilize stable release versions of Android Studio, Gradle, and libraries.

Preparation (continued...):

Software versions:

Android Studio	3.1.4	Google-Services Plugin	4.1.0
Gradle	4.4	Firebase Core	16.0.3
Android Plugin Version	3.1.4	Firebase Cloud Messaging	17.3.2
Butterknife	8.8.1	Firebase Realtime DB	16.0.2
Picasso	2.71828	Firebase Remote Config	16.0.0

Task 1: Project Setup

- Create new Android Studio Project
- Edit Gradle to add any 3rd party libraries
- Edit Gradle to add Firebase dependencies
- Edit Gradle to add Room dependencies

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Explore Activity (Main activity)
- Build UI for Explore Main Fragment
- Build UI for Explore Details Fragment
- Build UI for Mars Facts Fragment
- Build UI for Curiosity Weather Fragment
- Build UI for Curiosity Tweets Fragment

Task 3: Implement UI for Settings Screen

- Build Preference Screen UI
- Build Preference Fragment
- Apply Preference Theme
- Import Material Design Theme for Preference UI

Task 4: Create Data Classes

- Create Java Class for Rover object
- Create Java Class for Mars Fact
- Create Java Class for each Camera
- Create Java Class for User Favorite
- Create Java Class for Curiosity Tweets

Task 5: Setup Databases

- Add Firebase to app using Firebase Assistant and ‘Connect to Firebase’
- Add app to Firebase project in Firebase Console
- Add code to read Mars Facts from Realtime Database
- Set up Room database
- Create Entities for user favorite url and Curiosity Tweets
- Implement DAOs

Task 6: Configure Firebase Console

- Set up Firebase Cloud messaging for Curiosity Tweets (will use manual messages from the console for now, but will look into implementing a back end solution at a later date)
- Add code to app to handle incoming Tweets, and save to local Room database to display
- Set up Firebase Remote config to allow quick remote changes to API endpoints
- Add code to app to allow remote config of API endpoints.

Task 7: Finish main app code

- Implement AsyncTask for rover photo search
- Implement JobDispatcher to update Rover Manifest data (daily or 2x a day.)
- Handle and appropriately log any errors

Task 8: Implement Home Screen Widget

- Build widget layout
- Configure widget to update Facts daily
- Configure widget to display shortened version of Fact if full fact is too long
- Confirm clicking on widget opens app and displays the full Fact