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: MarcelBraghetto
Project Owner: Marcel Braghetto

App name: Daily Deviations

Description

The website http://www.deviantart.com/ promotes a community of varied and rich artwork contributed by artists across the world. This app gives quick and easy access to the most popular art submissions as they happen from the 'Daily Deviations' RSS feed.

This app gives quick and easy access to the latest Daily Deviations by automatically keeping up to date with the Deviant Art website and notifying the user when new Daily Deviations are available.

This app will build upon my submission for the Android Nano Degree project 5 and extend it with a number of additional features and improvements.

Intended User

The intended user is someone who appreciates the artwork found on the Deviant Art website and wants to have a mobile channel available to explore and integrate their favourite artworks into their Android devices.

Features

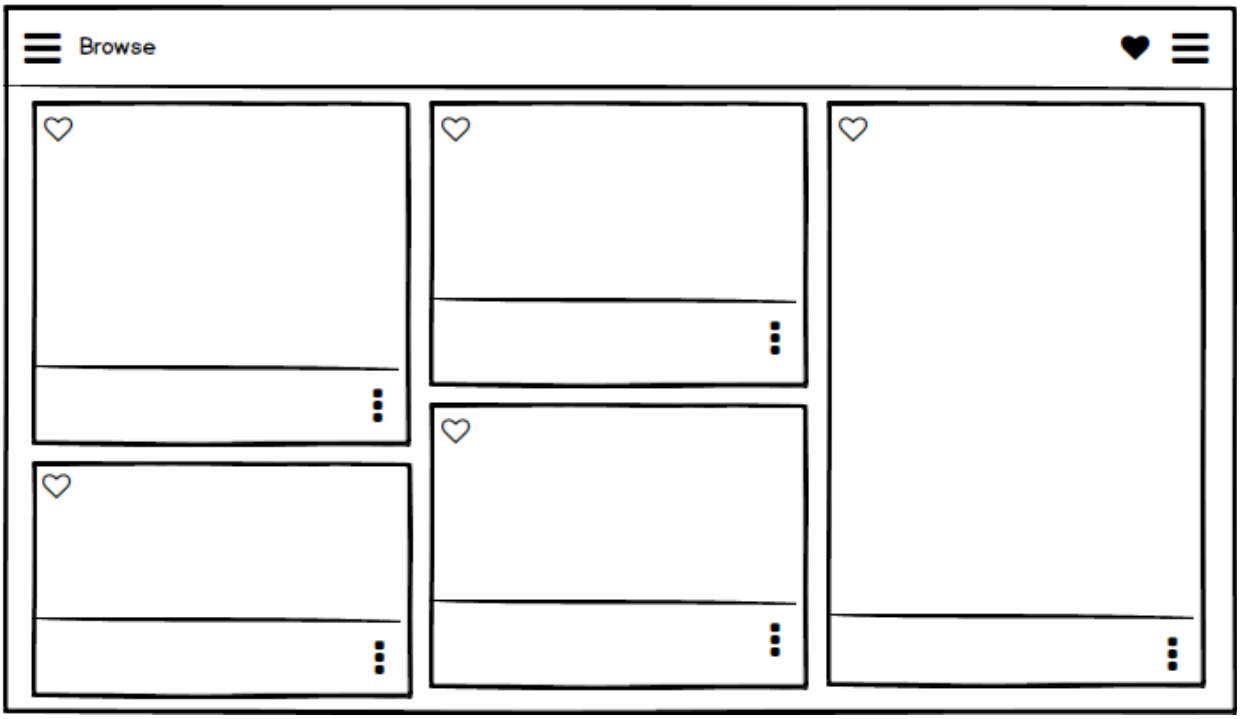
The main features of this app are:

- Automatic notifications when new Daily Deviations are published.
- Ability to explore images using pinch and zoom.
- Ability to mark images as favourites to build a private collection.
- Ability to set an artwork image as your phone background wallpaper.
- Android Widget to always see the latest Daily Deviations on your home screen.

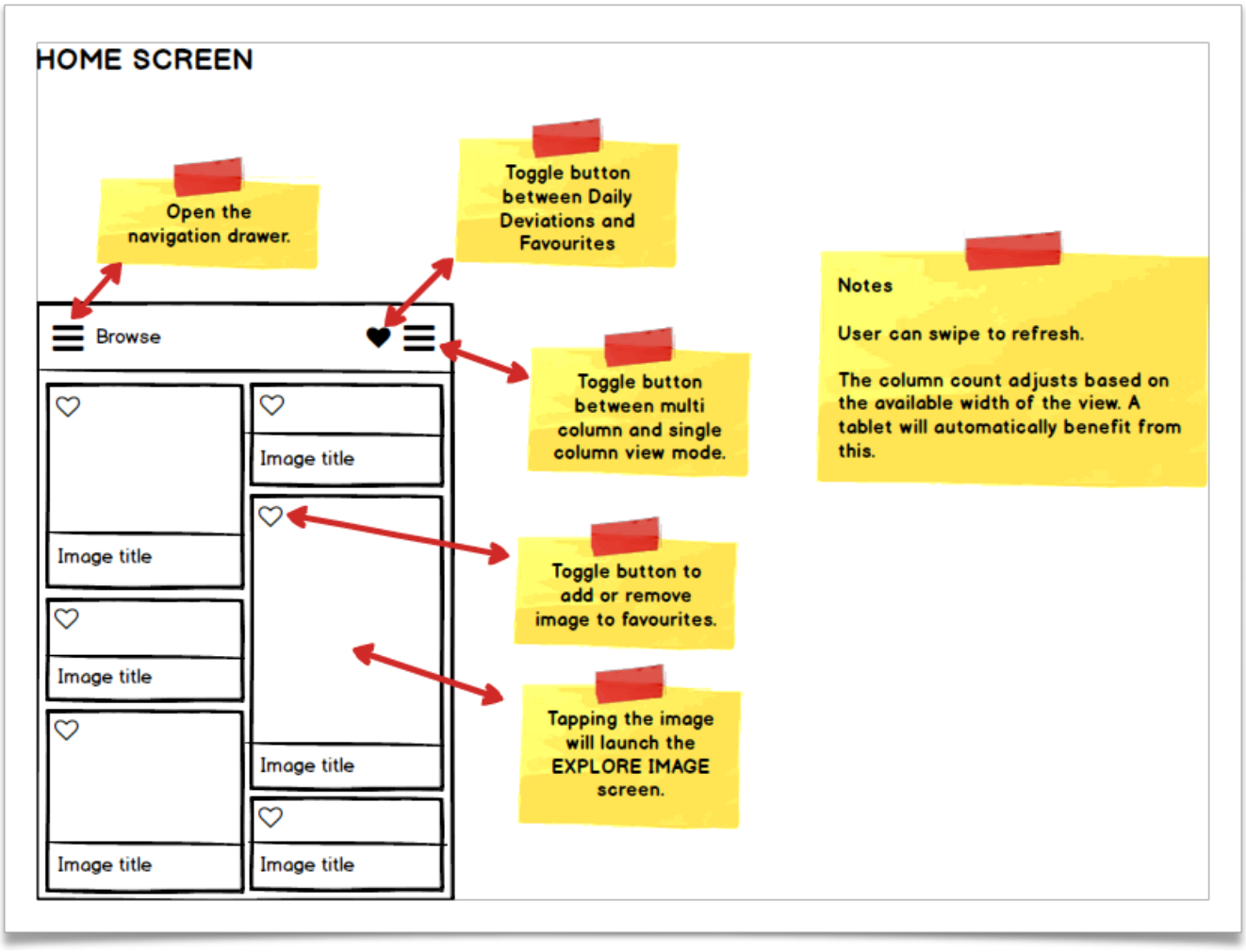
User Interface Mocks

HOME SCREEN - TABLET / WIDE SCREEN

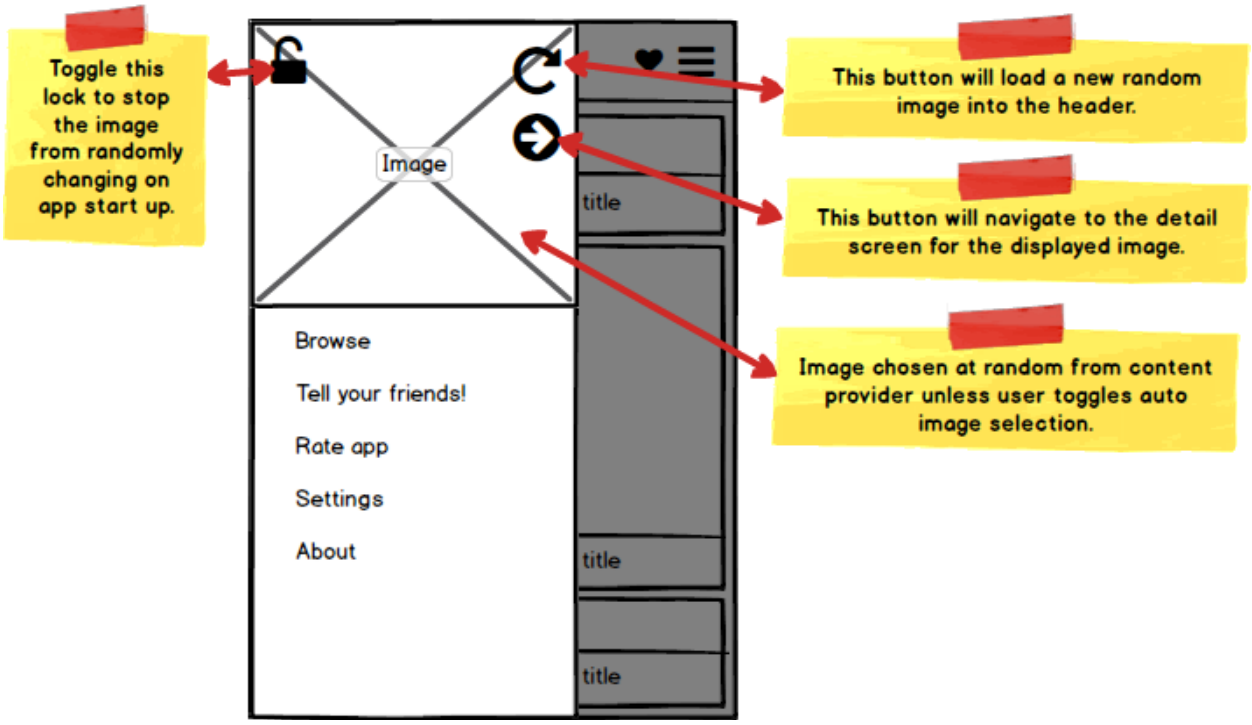
The tablet / wide screen format will operate the same as for phone / narrow screen but with extra and larger sized image columns to gracefully fill the area and still provide a rich user experience.



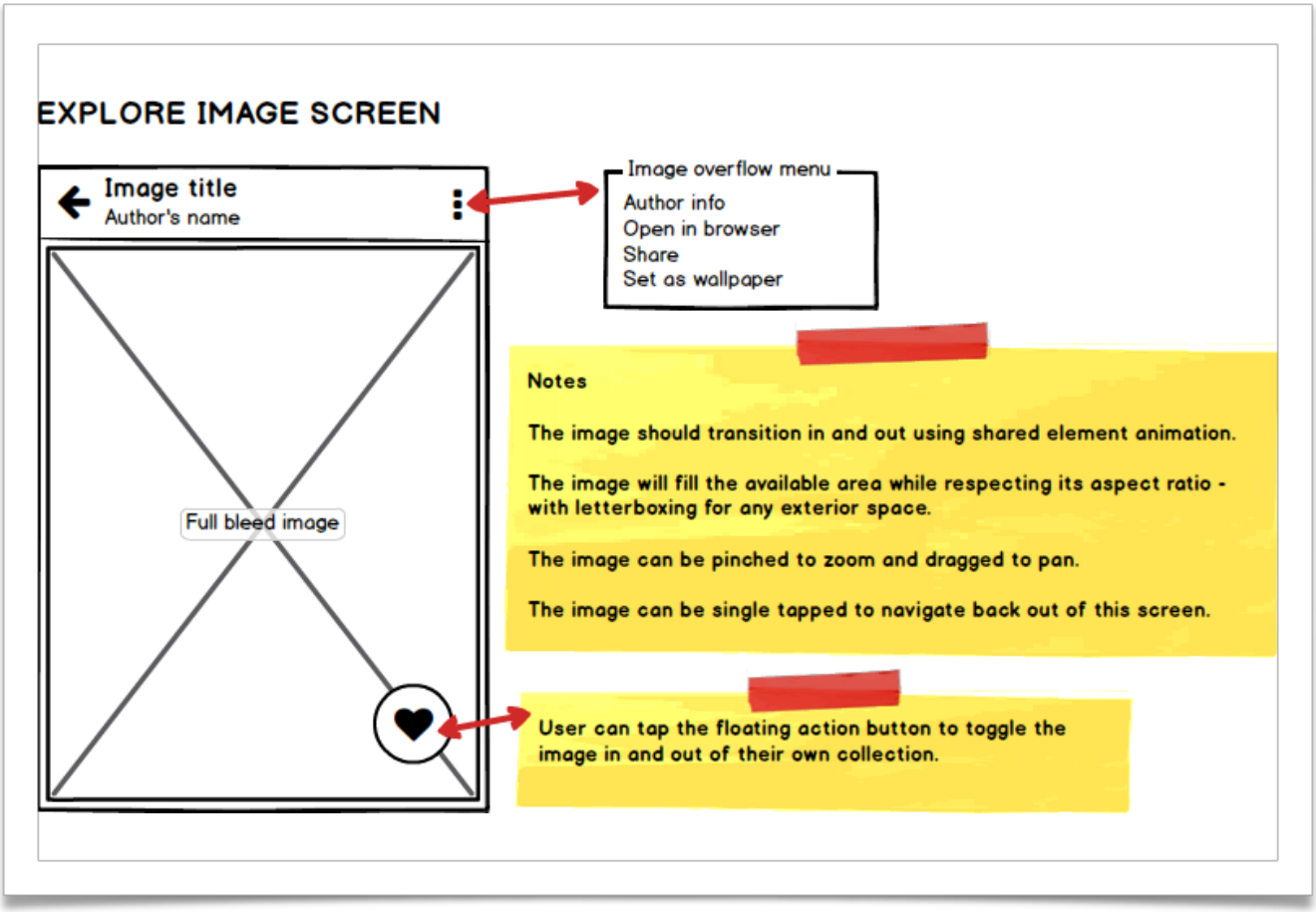
Home screen operation described on following page.


HOME SCREEN
Open the navigation drawer.
Toggle button between Daily Deviations and Favourites
Browse
Toggle button between multi column and single column view mode.
Notes
User can swipe to refresh.
The column count adjusts based on the available width of the view. A tablet will automatically benefit from this.
Image title
Image title
Image title
Image title
Image title
Image title
Toggle button to add or remove image to favourites.
Tapping the image will launch the EXPLORE IMAGE screen.

NAVIGATION DRAWER AND HEADER VIEW

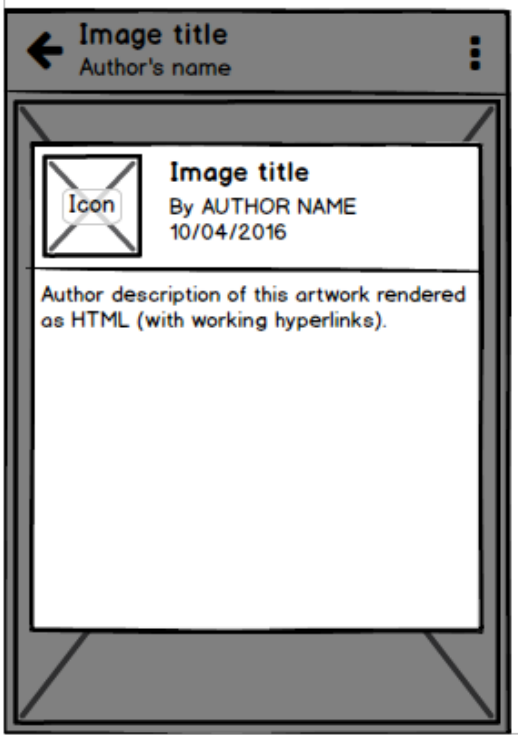


Navigation menu	Action
Browse	Navigates to the main view.
Tell your friends	Initiates the Google Play App Invites flow.
Settings	Opens settings screen / view.
Rate app	Initiates request to Google Play store for app.
About	Opens about screen / view.



Option	Action
Author info	Opens the more info screen / view for the given image.
Open in browser	Opens the current image web url in the browser
Share	Initates the share image flow.
Set as wallpaper	Sets the current image as the device wallpaper.
Floating action button	The floating action button toggles the image in the user's personal collection.

AUTHOR INFO SCREEN



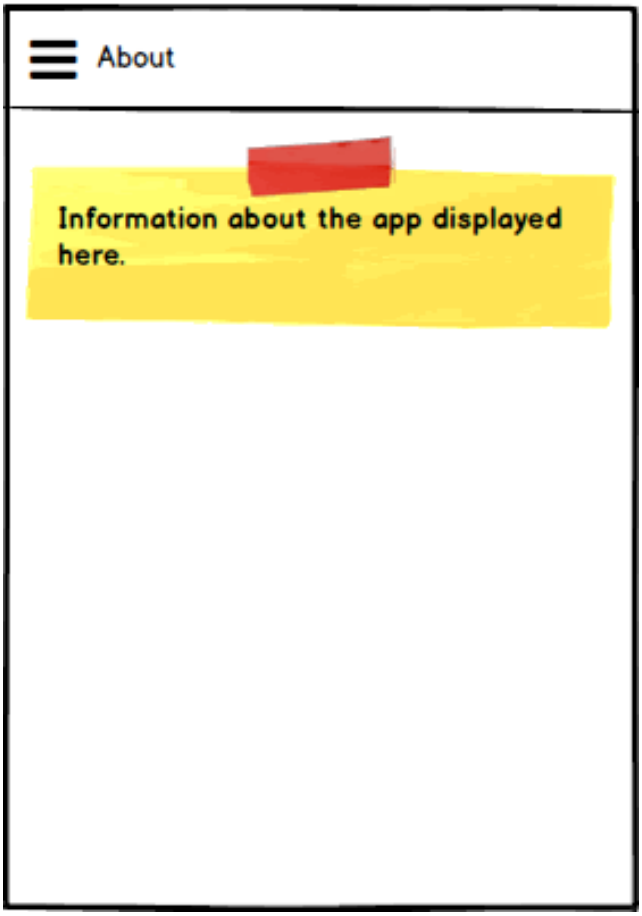
Notes

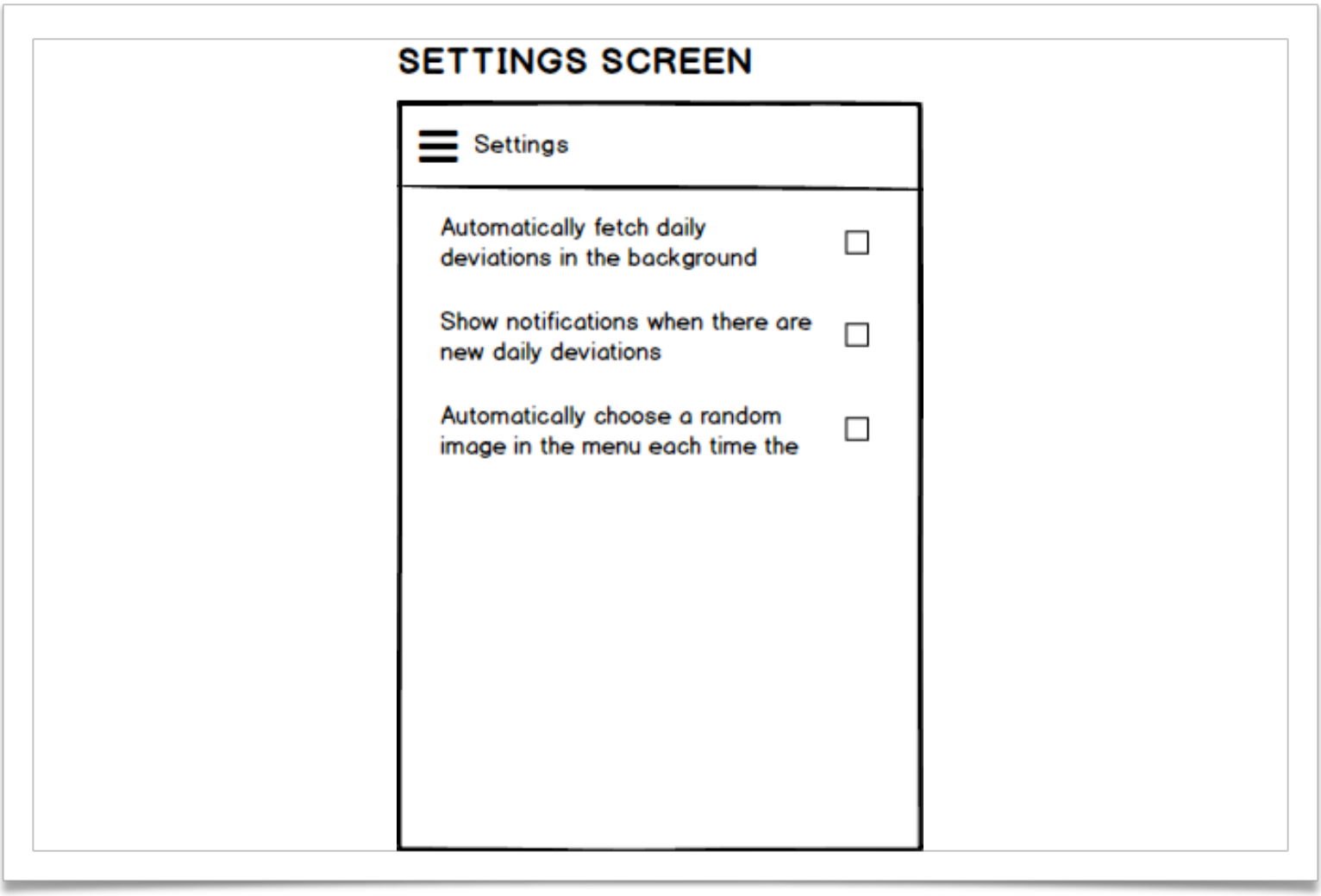
The icon is the author's profile icon fetched from their Deviant Art profile.

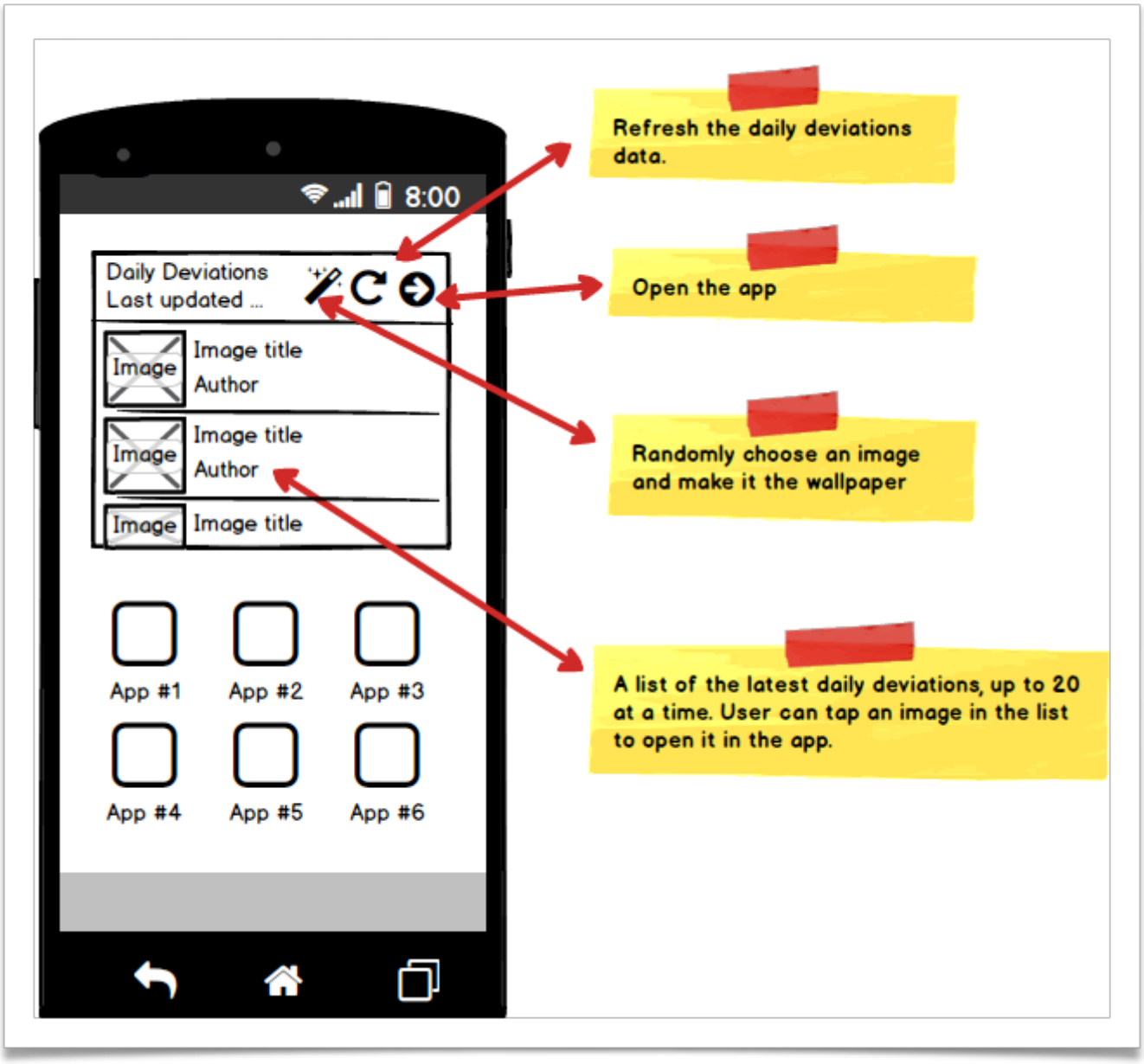
User can exit the window by pressing the back key, or by tapping anywhere outside the content frame.

The content is rendered as HTML, via a WebView that dynamically fits until a maximum size for scrolling.

ABOUT SCREEN




SETTINGS SCREEN
Settings
Automatically fetch daily deviations in the background
Show notifications when there are new daily deviations
Automatically choose a random image in the menu each time the


8:00
Daily Deviations
Last updated ...
Image
Image title
Author
Image
Image title
Author
Image
Image title
App #1
App #2
App #3
App #4
App #5
App #6
Refresh the daily deviations data.
Open the app
Randomly choose an image and make it the wallpaper
A list of the latest daily deviations, up to 20 at a time. User can tap an image in the list to open it in the app.

Key Considerations

How will your app handle data persistence?

Data will persist through a custom content provider which understands the data format for storing and exposing Deviant Art RSS feeds. The content provider will be automatically updated periodically or manually when the user refreshes. It will also track which artworks have been 'favourited' by the user.

Describe any corner cases in the UX.

A user can view an image from the main screen by tapping it, then pinch and zoom to explore. A single tap when exploring will navigate the user back.

A user can view the author's details about a particular image via a popup window. The window can be dismissed via the back key OR by tapping outside the window area.

Error messages due to network connectivity will be handled in the UI.

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

Google libraries that will be used:

Library	Description
Android Support	Android support libraries including: - Appcompat - Design - Cardview - Recyclerview
Google Play Services - Analytics	https://firebase.google.com/docs/analytics/ Firebase analytics via Google Play Services
Google Play Services - App Invites	https://firebase.google.com/docs/invites/ Firebase app invites via Google Play Services

Third party libraries that will be used:

Library	Description
Dagger	http://google.github.io/dagger/ Dagger will be used to provide dependency injection and allow for my preferred architecture pattern for implementing inversion of control.
Android APT	https://bitbucket.org/hvisser/android-apt Libraries such as Dagger require an annotation processor to operate. This gives the ability for annotation processing to run.
OkHttp	http://square.github.io/okhttp/ OkHttp will be used in the network stack to fetch Deviant Art data via the Daily Deviations RSS feed.
EventBus	https://github.com/greenrobot/EventBus EventBus will be used as a replacement for the standard Android Broadcast components. It is very flexible and allows for fine grained event pub / sub interaction.
Glide	https://github.com/bumptech/glide Glide will be used so fetch remote images, cache them locally and to perform any special transformations required.
PhotoView	https://github.com/chrisbanes/PhotoView PhotoView will be used to provide the ability to pinch to zoom and panning to explore ImageView content.
JodaTime	https://github.com/JodaOrg/joda-time JodaTime will be used to assist with date and time calculations.
Parceler	https://github.com/johncarl81/parceler Parceler will be used to remove the standard boilerplate from implementing the Parcelable type.

Library	Description
ShortcutBadger	https://github.com/leolin310148/ShortcutBadger ShortcutBadger will be used to provide the user a visual cue on the application icon as to how many new Daily Deviations are currently waiting to be viewed.
Robolectric	http://robolectric.org/ Robolectric will be used to author suites of unit tests that can run via the Unit Test runner instead of the Instrumentation Test runner and will be applied in writing all the business logic unit tests.
Mockito	https://github.com/mockito/mockito Mockito will be used within unit test suites to provide a way to mock dependencies. This couples with the architecture pattern based on dependency injection (which Dagger gives us), allowing easy mock injections and promoting black box unit testing.

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 a main app module. - Setup and configure the shared framework - wiring Dagger with some basic providers and utility classes.

Task 2: Implement Daily Deviations content provider
- Create a data contract representing collections of daily deviations. - Create a database helper to wrap storage to an SQLite database. - Create a content provider that can take a collection of daily deviations and save them, or fetch the current collection with any applicable filters (such as favourites).

Task 3: Implement Daily Deviations network stack

- Create an intent service to call the Daily Deviations RSS feed via the networking framework.
- Write an RSS parser to extract collections of daily deviations from the feed.
- Connect the extracted collections to the content provider to update the local persistent storage.
- Notify via EventBus the result of the data fetch (possibly resulting in a failed connection etc)

Task 4: Implement automated background connection services

- Implement alarm manager broadcast receiver to periodically trigger background data refresh.
- Implement data refresh receiver to be triggered on application boot or network connectivity changes. This is to catch scenarios where the app might be offline for long periods of time.
- Implement a local notification system to display a notification when data changes in the background. The notification should not be presented if the application is in the foreground.
- Show the number of unseen new Daily Deviations as an application icon badge (if the host operating system supports it).

Task 5: Home screen: Daily Deviations view

- Implement a staggered grid display which renders the collection of deviant artworks stored in the content provider.
- The collections will be displayed in order of newest to oldest.
- Include a toggle button to view the users ‘favourites’ collection.

Task 6: Home screen: Favourites view

- Implement a ‘favourite’ button for each artwork to toggle it being displayed in the user’s ‘favourites’ collection.
- Implement a ‘favourites’ collection view that the user can switch to and from on the home screen.
- Include a toggle button to view the daily deviations view.

Task 7: Home screen: Navigation menu

- Implement an options overflow menu.
- Add an option to share the app.
- Add an option to open the ‘settings’ screen.
- Add an option to open the ‘about’ screen.
- Add an option to ‘rate’ the app.

Task 8: Home screen: Navigation menu header

- Implement an interactive navigation header that randomly displays an image from the content provider when the app starts.
- Add a button to open the current header image details screen.
- Add a button to randomly choose another header image.
- Add a button to toggle whether the image is randomly selected on app startup or not.

Task 9: Explore image screen: Overflow menu

- Implement an overflow menu for the current artwork image in the detail screen.
- Add an option to view the author information about the image.
- Add an option to open the image web url in the system browser.
- Add an option to share an artwork image.
- Add an option to set a given artwork image as the device background wallpaper.

Task 10: Explore image screen

- Implement a ‘explore image screen’ when an image is tapped to allow the user to pinch to zoom and pan to explore the image.
- User can tap a toggle button to add/remove the image into their collection.

Task 11: Author information screen

- Implement a screen or view that displays the author for a given artwork image along with any available description and meta data. The information screen will be triggered using one of the options in the artwork image overflow menu.

Task 12: Settings screen

- Implement a settings screen or view that allows the user to toggle local notifications and whether the app should automatically fetch data in the background.

Task 13: About screen

- Implement an ‘about’ screen or view that allows the user to read about the app.

Task 14: Daily Deviations home screen widget

- Implement a home screen widget that can be added by the user to provide a quick view of the current daily deviations.
- Add the ability to tap on an image in the widget to open it in app.
- Add the ability to tap a button to open the app.
- Add the ability to tap a button to cause a data refresh
- Add the ability to tap a button to randomly set the background wallpaper.

Task 15: Integrate Google Analytics API

- Integrate Google Analytics to track screen views and user triggered events in the app using Google Firebase.

Task 16: Integrate Google App Invites API

- Integrate the Google App Invites API to provide a way for user’s to share the app with their friends and families using Google Firebase.

Task 17: Unit test suite

- Author a suite of unit tests to exercise business logic found in presentation, framework and utility classes using JUnit, Robolectric and Mockito.

Task 18: Release packaging

- Configure Proguard for release build.
- Configure signing key for release build.
- Produce signed release build for submission / publication.

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**"