GitHub Username: kdrosado

TrendyArt

Description

TrendyArt is an android application that allows you to view beautiful art pieces from many places.

Intended User

TrendyArt is intended for the following users:

- Artists
- Art Lovers
- Anyone who enjoys a free android app

Features

Here are the main features of TrendyArt:

- App conforms to common standards found in the Android Nanodegree General Project Guidelines
- App is written solely in the Java Programming Language
- App utilizes stable release versions of all libraries, Gradle, and Android Studio.
- App integrates a third-party library, in this case Artsy API.
- App validates all input from servers and users. If data does not exist or is in the wrong format, the app logs this fact and does not crash.
- App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.

Google Play Services:

- App integrates two or more Google services, in this case AdMob & Google Analytics.
- Each service imported in the `build.gradle` is used in the app.
- `Admob` is used to displays test ads.
- `Analytics` is used to creates only one analytics instance.

Material Design:

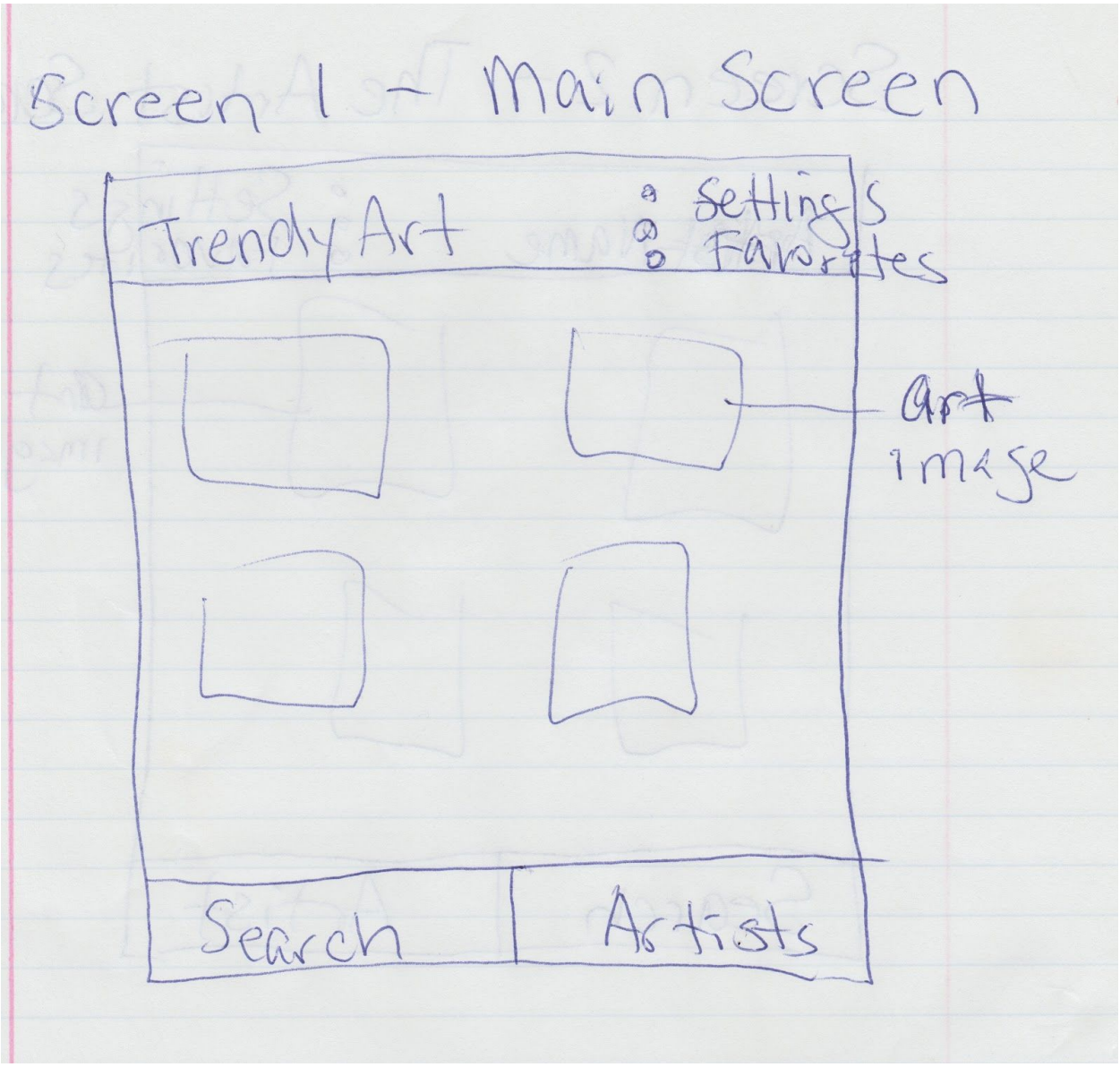
- App theme extends `AppCompat`.
- App uses an app bar and associated toolbars.
- App uses standard and simple transitions between activities.

Building:

- App builds from a clean repository checkout with no additional configuration.
- App builds and deploys using the `installRelease` `Gradle` task.
- App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path.
- All app dependencies are managed by `Gradle`.
- App keeps all strings in a `strings.xml` file and enables RTL layout switching on all layouts.
- App provides a widget to provide relevant information to the user on the home screen.

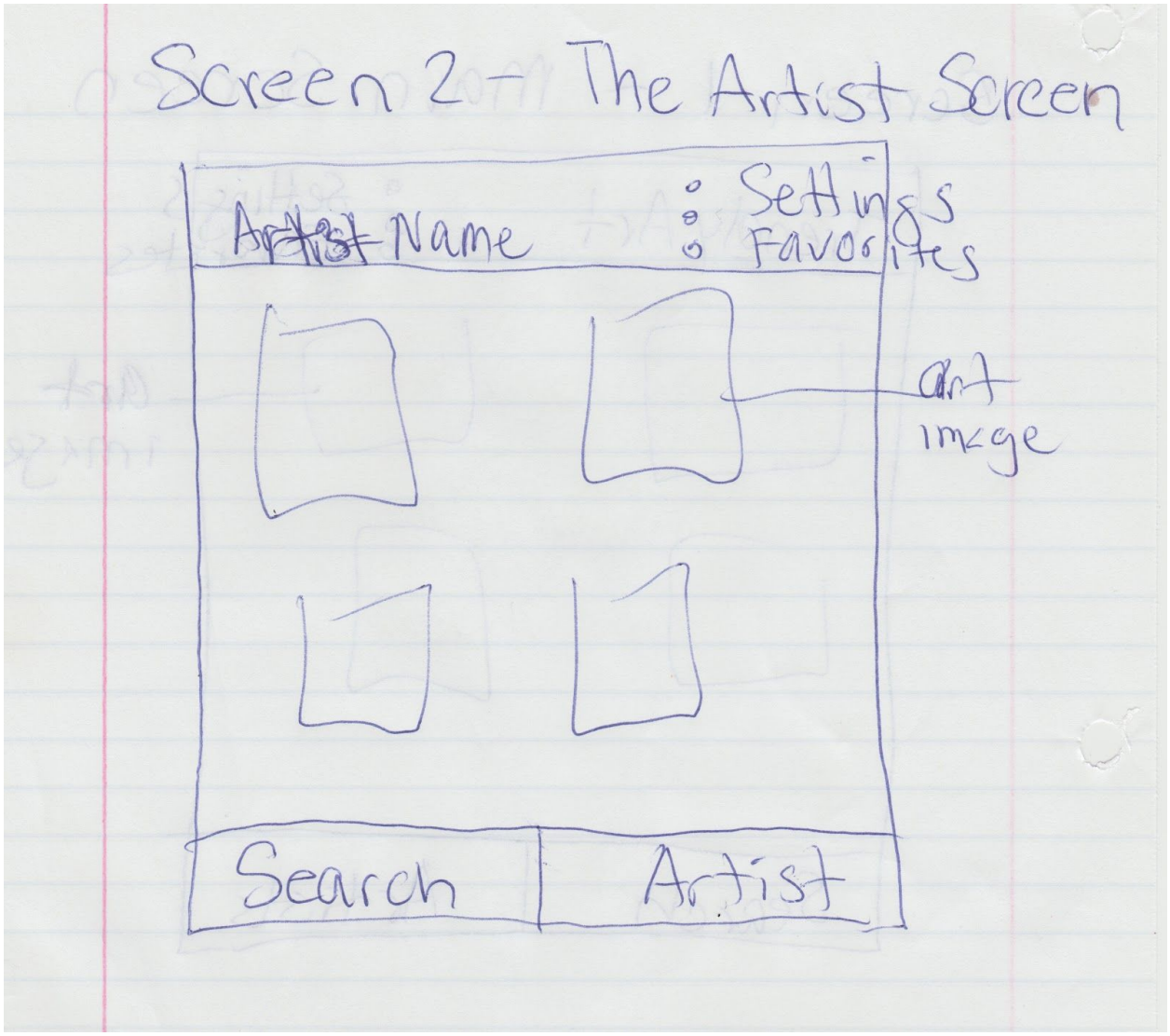
User Interface Mocks

Screen 1 - Main Screen



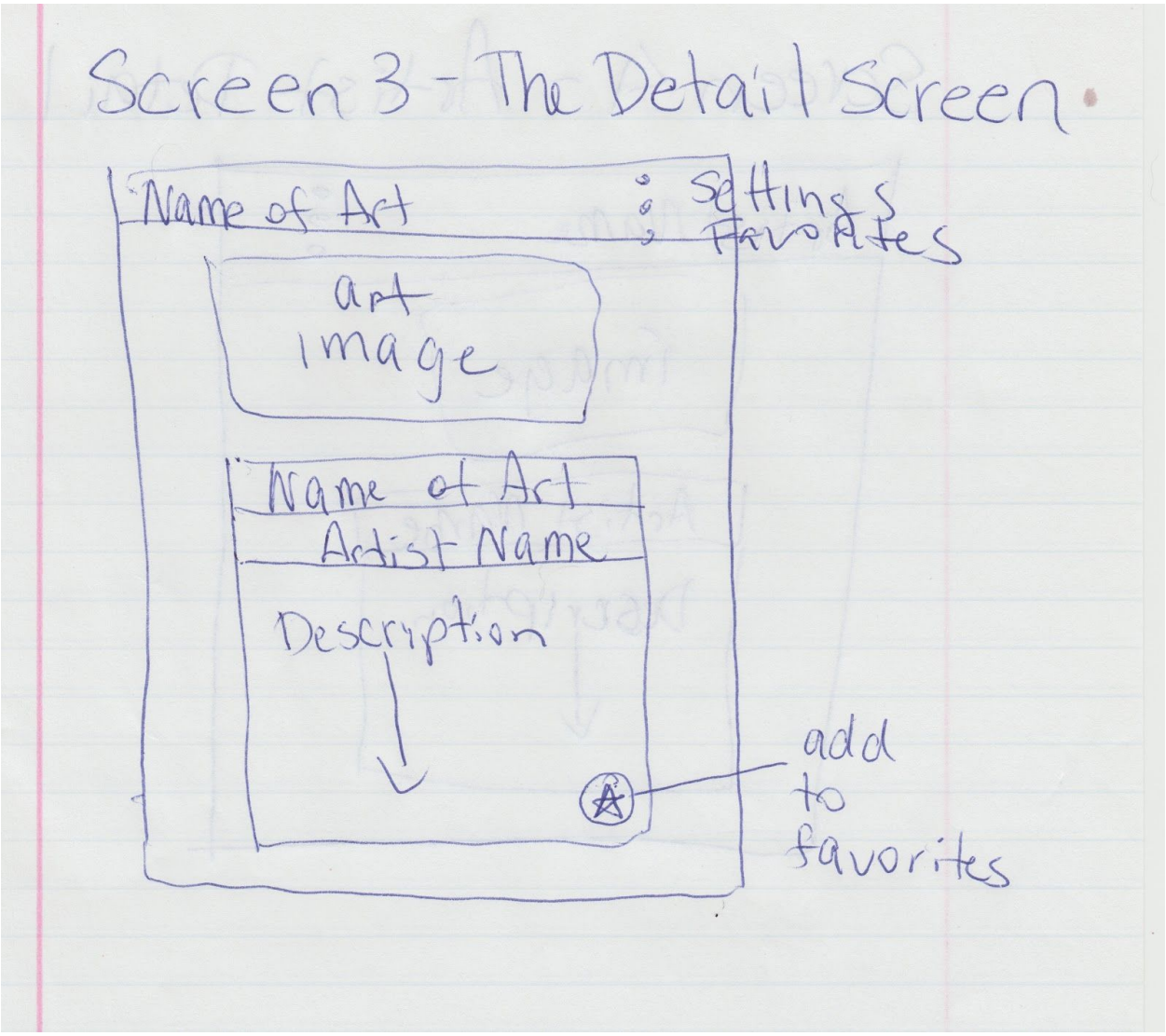
The main screen will display a top navigation bar with the application's name, access to settings, and favorites. It will also display the art images. The bottom navigation bar will allow the user to do a general search, or search by artist.

Screen 2 - The Artist Screen



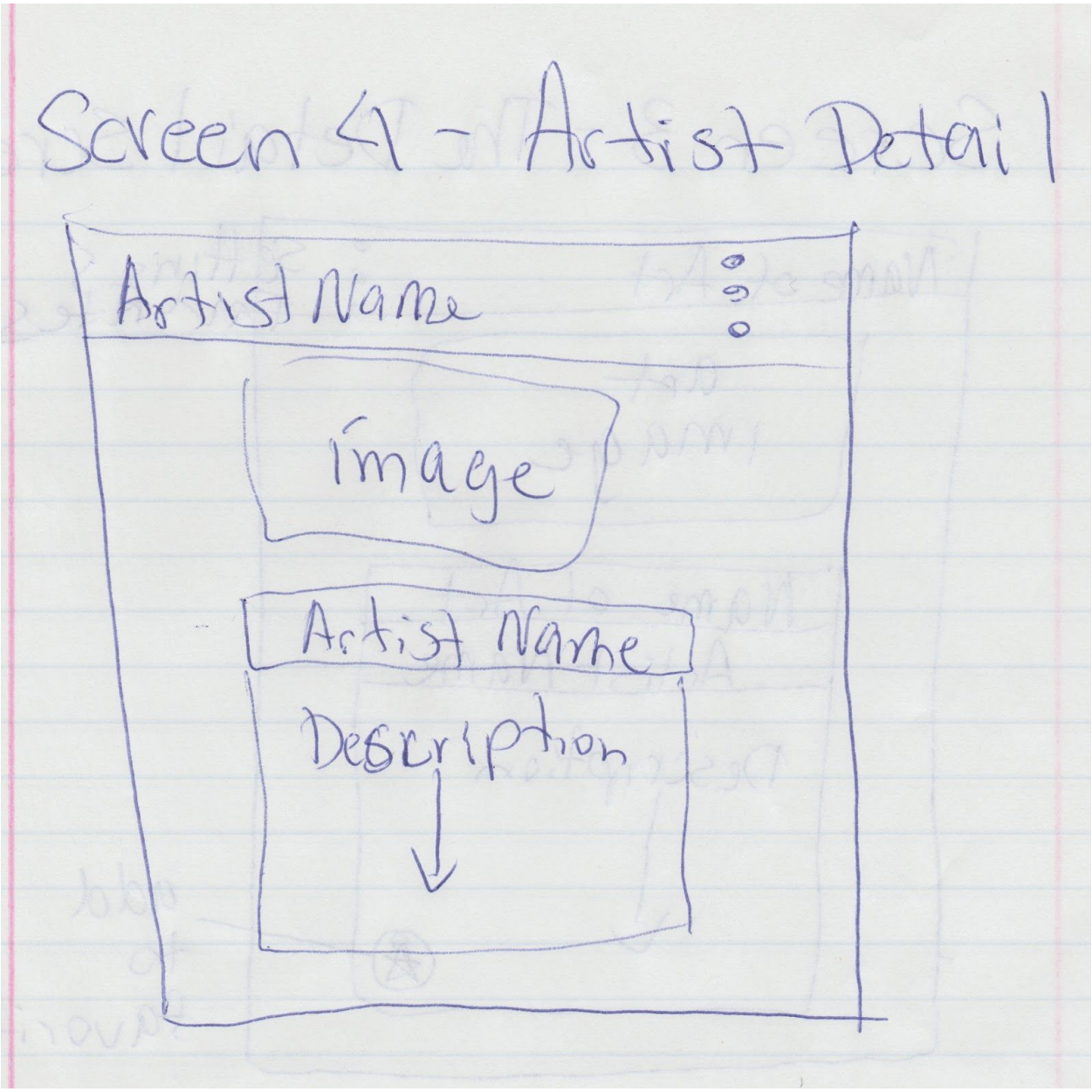
The artist screen is very similar to the main screen, except the images shown are specific to the artist the user has selected. So you will see a top navigation bar with the artist name, access the settings, and favorites. The bottom navigation will show the general search and artist search options for the user.

Screen 3 - The Detail Screen



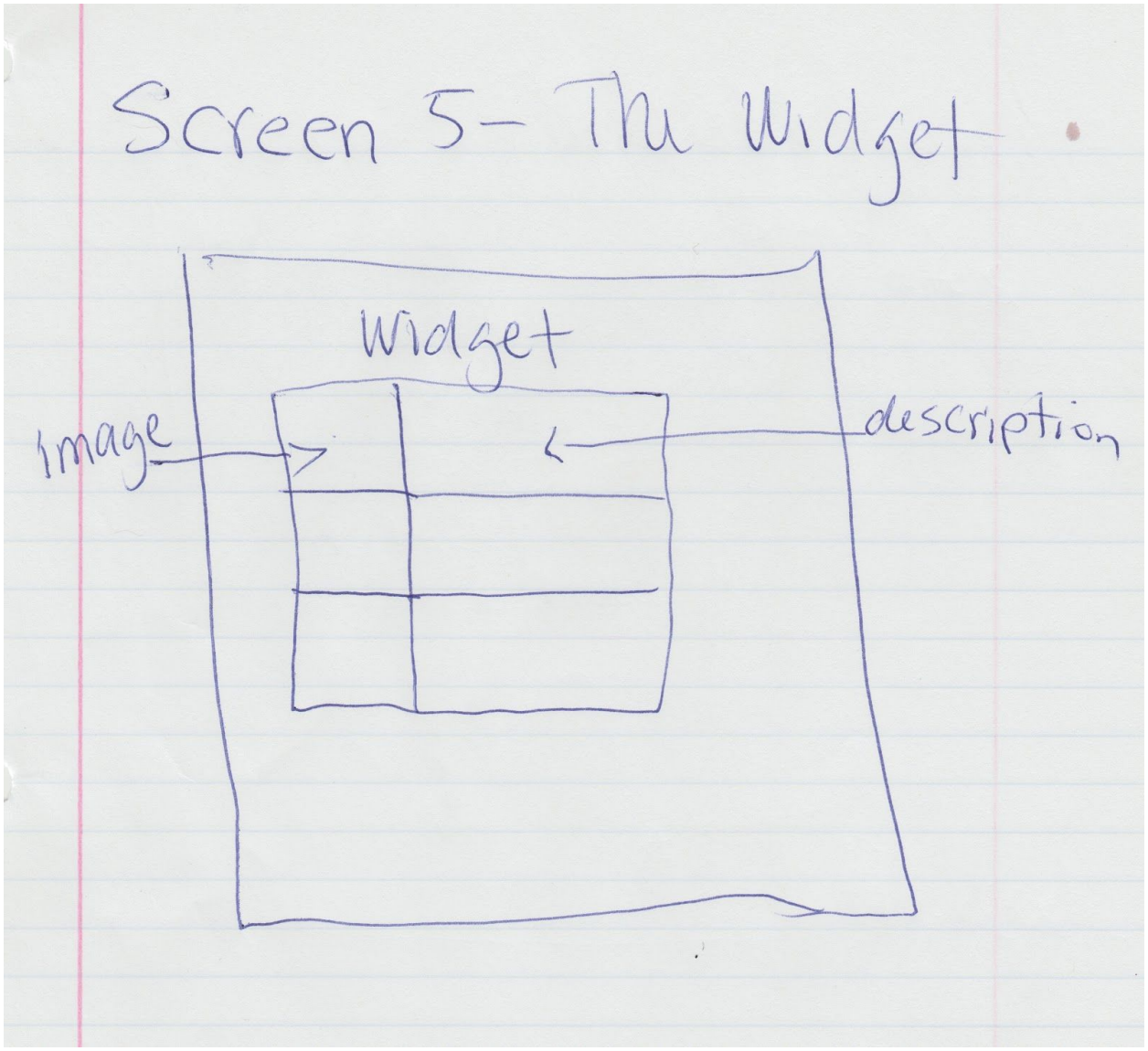
The detail screen will show the details of the art image that was selected by the user. It will display the image, name of the art piece, the artist name, and general description of the art piece. The user will also have the option to add this to their favorites.

Screen 4 - The Artist Detail Screen



The artist detail screen will be very similar to detail screen. However, the focus will be on displaying information on the artist.

Screen 5 - The Widget Screen



The widget screen will display the users favorites.

Key Considerations

How will your app handle data persistence?

- Room will be used to store the users favorites

Describe any edge or corner cases in the UX.

- Back button will allow user to return to previous screen

- Forward/next button will allow user to move on to the next screen
- User will also be able to navigate the app by selecting an item on the screen

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

- **Picasso:** to handle loading and caching of images
- **Retrofit:** to create readable Java interfaces and classes that match the API we are accessing, as well as allowing us to interact with simple Java objects instead of having to parse network responses manually
- **Butterknife** - for data binding
- **Room** - for database access

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

- **Artsy API:** To provide the data that will be displayed
- **Admob:** To display test ads
- **Google Analytics:** To detect the most used features

Next Steps: Required Tasks

Task 1: Project Setup

- Create a new project in Android Studio
- Set up access to Artsy API, AdMob, and Google Analytics
- Include all necessary libraries in build.gradle

Task 2: Implement UI for Each Activity and Fragment

- Build UI for the MainActivity
- Build UI for the artist screen
- Build UI for the detail screen
- Build UI for the artist detail screen

Task 3: Set-Up the Backend

- Make a network call using Retrofit
- Implement android architecture components
- Implement the Room persistence library for data storage
- Implement Google Play Services, Admob & Analytics

- Handle error cases by displaying an empty view when no connection is available, and testing for app errors
- Use an AsyncTask to check the internet connectivity

Task 4: Implement the Widget

- Build the UI for the widget
- Create an adapter
- Create a widget provider

Task 5: Set-up The Keystore

- App is equipped with a signing configuration, and the keystore and passwords are included in the repository. Keystore is referred to by a relative path.

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**"