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: bapspatil

WallCards

Description

One of the most powerful features of buying an Android phone is its customizability. Wallpapers are one of the most important aspects of customizability in Android, and are almost solely responsible for the look and feel of an Android phone. They affect the user's mood on a very subconscious level and in some cases, like live wallpapers, enhance the overall user experience by providing certain extra features. But most of the wallpaper apps available on the Play Store are filled with numerous gimmicks and have many UX design flaws.

WallCards aims to solve this problem by being a great wallpaper app that encapsulates minimal elegant design along with great functionality. The main design principle followed here is what I

like to call the WallCards principle, which is an iteration of the Material Design principles by Google. The app shows wallpapers as if they were “cards”, not any different that playing cards, and then allows users to set them as their phone’s wallpaper. The motion of these “cards” is very similar to those of playing cards being thrown (as seen in magic tricks).
As for choice in wallpapers, I don’t want to bog down our users with a ton of choices. I want to keep it minimal and elegant, and will offer the top wallpapers from the Unsplash community from a couple of sections.

Intended User

Users that wish to have a unique wallpaper app that is easy to use and has a great UX to set a wallpaper in just a few taps.

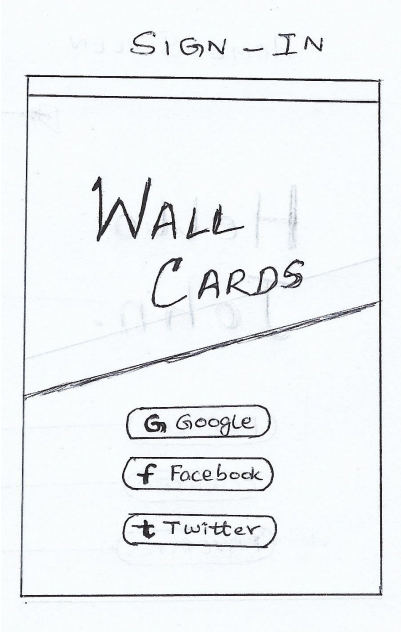
Features

Main features of WallCards:

- Shows a grid of “WallCards”
- Has a few sections for showing different kinds of wallpapers
- All WallCards have edge-to-edge full-bleed wallpapers inside them
- Smooth animations and beautiful transitions make the app a pleasure to use
- Sets wallpaper for home & lock screen
- Users can add a WallCard to their Favorites
- Widget for the app shows the user’s favorite WallCards

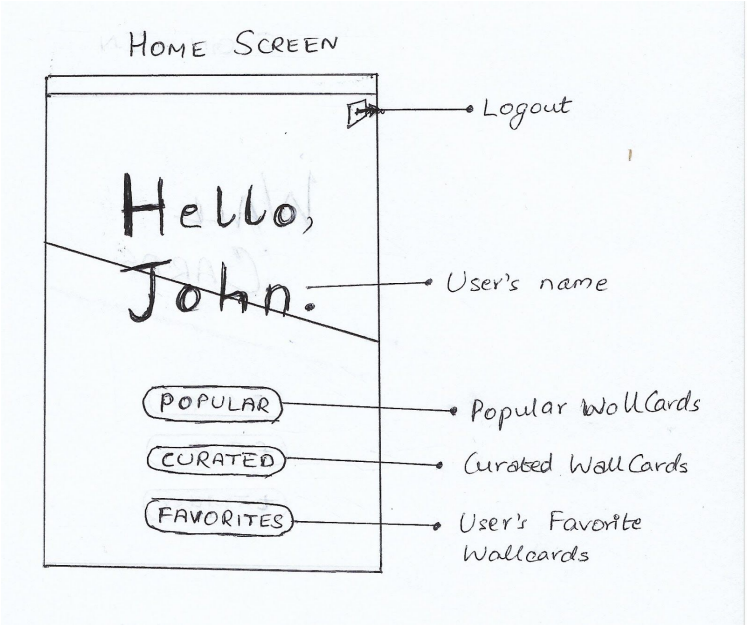
User Interface Mocks

Login Screen



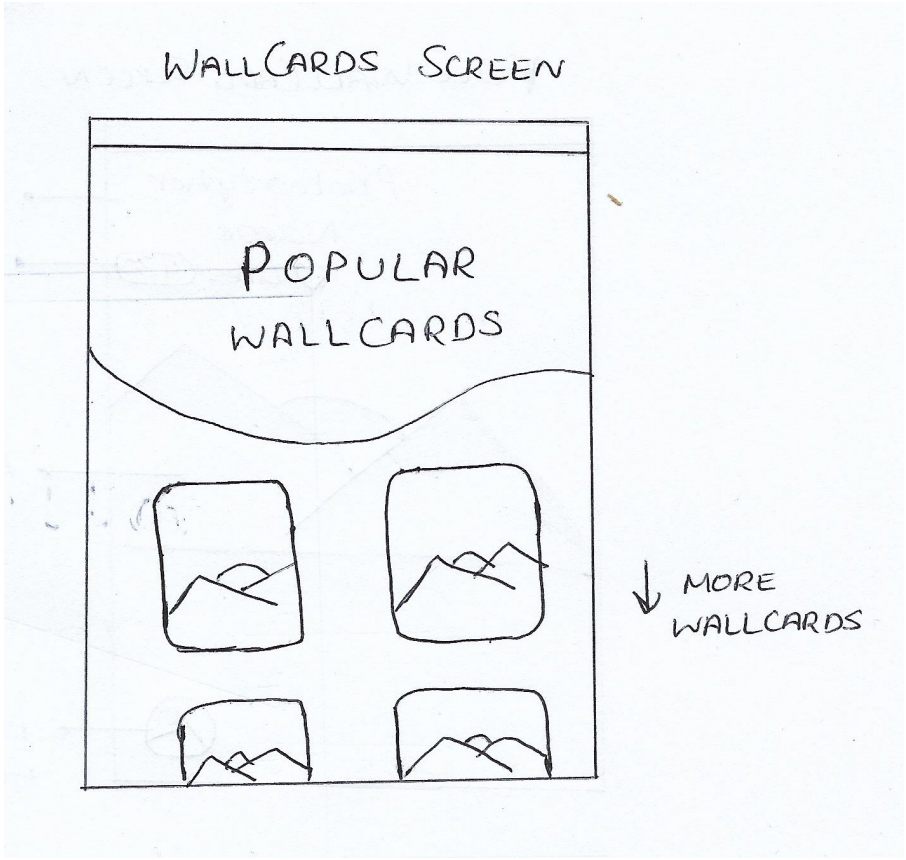
This is the Login screen where the user can login.

Home Screen



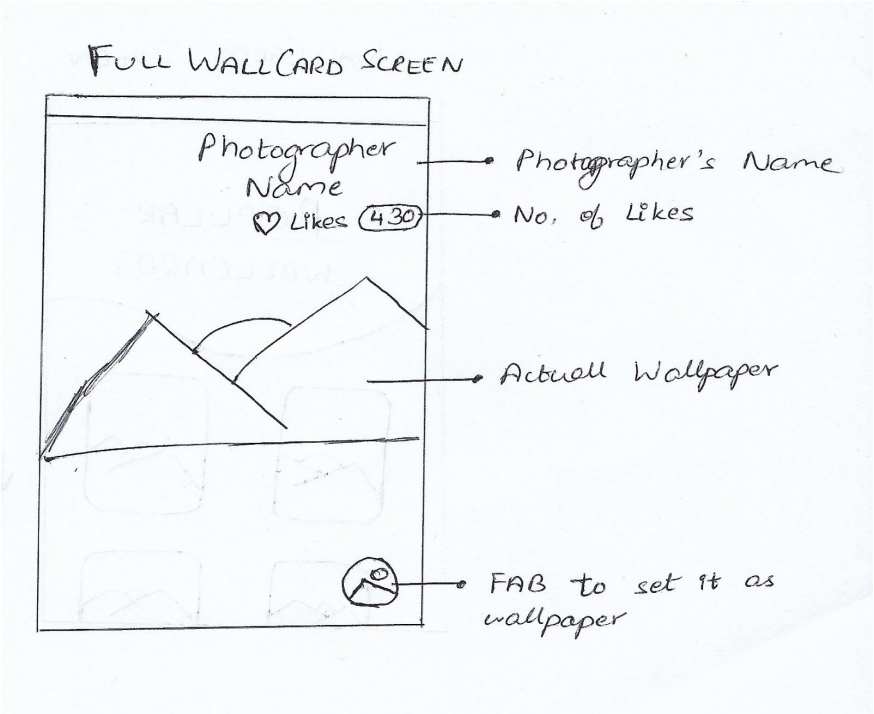
This is the Home screen where the user can see Popular, Curated or his/her Favorite WallCards. He/She can also log out from this screen.

WallCards Screen



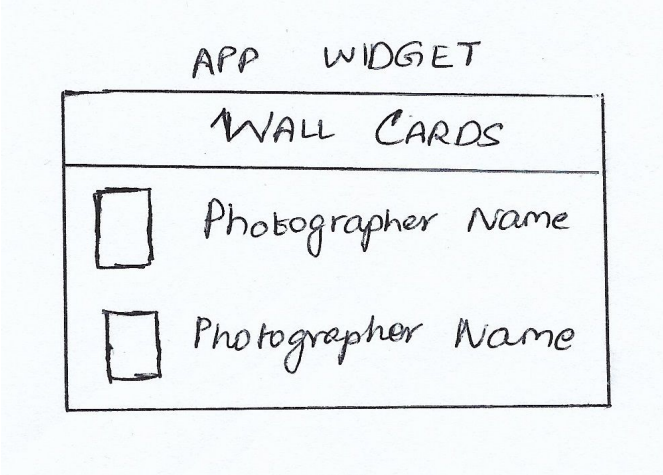
This is the WallCards screen which shows a grid of wallpapers in the form of WallCards for Popular, Curated & Favorite WallCards.

Full WallCard Screen



This is the Full WallCard screen from where you can set the selected image from the previous WallCards screen as the wallpaper for your phone.

App Widget



This is a preview of the widget for the WallCards app that shows a list of the user's Favorite WallCards.

Key Considerations

How will your app handle data persistence?

The app will use a ContentProvider, Contract and SQLiteDbHelper that stores the user's favorite WallCards. It will use Glide to cache and store all qualities of the wallpapers so that can be viewed offline too.

Describe any edge or corner cases in the UX.

There will be a main Home screen from where the user can go to Favorites screen or the new Wallpapers screen to get the latest wallpapers. He/She can logout from the Home screen with one tap.

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

As mentioned above, I'll be using Contracts, Content Providers and SQLiteDbHelpers with the help of SQLite for data persistence, because it's fast and has a great Annotation-based support for data operations.
Apart from that, I'll be using Glide, Retrofit, and many more animation and design libraries to enhance the quality of my app.

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

I'll be using Firebase Crashlytics, Firebase Analytics and Firebase Cloud Messaging services to notify the user of any new app updates or anything else if required, handle crash reporting and see how my app is being used so I can improve on it in further iterations.
I'll also implement Firebase Authentication in the beginning to get the user to login so that the app can make a friendly greeting on the Home screen that is different for every user.

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

The subtasks for the Project Setup include:

- Deciding on the minimum API the app should support, preferably API 19 or API 21 at this point.
- Adding the Gradle dependencies for the app
- Using ProGuard to keep the APK size small

Task 2: Implement UI for Each Activity and Fragment

The subtasks for the UI implementation include:

- Build UI for MainActivity
- Build UI for all fragments that represent different screens of the app
- Build UI for the details activity of a wallpaper

Task 3: Firebase Implementation

Implementing Firebase Crashlytics, Firebase Analytics and Firebase Cloud Messaging.

- Setting up the app and adding it to a Firebase project
- Setting up Firebase Analytics to listen to custom events in the app
- Setting up the Firebase Cloud Messaging to listen to notifications sent to the app via the Firebase Console
- Setting up Firebase Authentication for user login

Task 4: Setting up Data Persistence

- Adding all entities to the database
- Setting up SQLite database operations
- Auto mapping these entities with the help of Retrofit and GSON

Task 5: Implementing Retrofit to access APIs

There will be 2 APIs used in the app for it to provide the features mentioned above.

- Adding GSON serialization/deserialization to the Wallpaper objects
- Making GET requests with Retrofit
- Auto-mapping the results of the GET requests made by Retrofit by creating Response data classes for all GET responses and embedding Wallpaper objects in them

Task 6: Building the adapters, widgets and writing logic for core app functionality of app

This step includes writing the adapters, widgets and connecting the logic to these views of the app.
It also includes fetching Favorites and displaying it in the Favorites section.

Task 7: Adding visual polish to the app

This last step consists of making the app look just fantastic and adding those subtle but really cool-looking animations and transitions between all the elements in the app, and activities.

Conclusion:

RecyclerView will be used to show any grid of wallpapers in the app.
Retrofit and Loaders will be used to run most of the asynchronous long-running tasks in the app.
App will have lots of animations and visual treats, and will follow most of the Material Design guidelines with also having implemented a design of its own, called WallCard Design.

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**"