GitHub Username: Exerosis

Meme Stream

Description

MemeStream gives you access a stream of of top notch hilarious posts filtered by a cutting edge AI that learns from you!

Quickly check your stream, or a stream of top posts to cure boredom, have a good laugh with friends, or to find something good to share with a friend.

Intended User

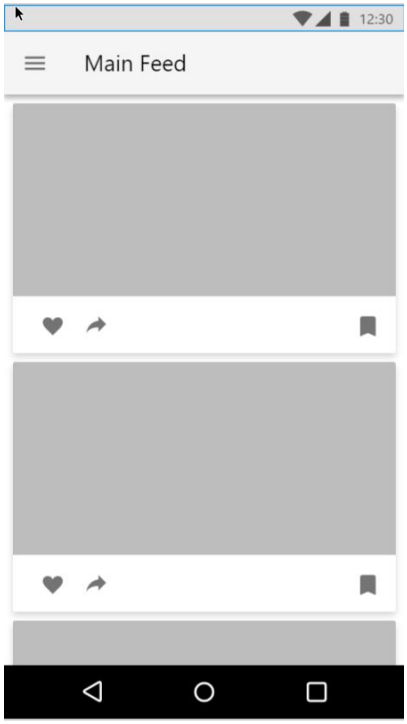
The average user this app targets is 10-35 years old, and likely currently uses apps like 9GAG, iFunny, and or Reddit.

Features

- Displays posts best fit to an account from backend.
- Sends new posts to backend.

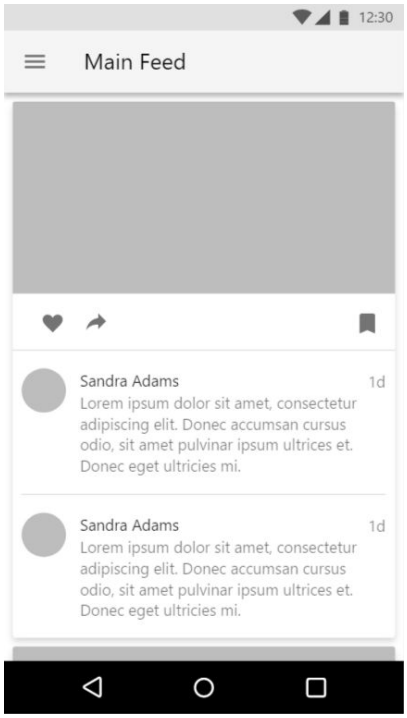
User Interface Mocks

Screen 1



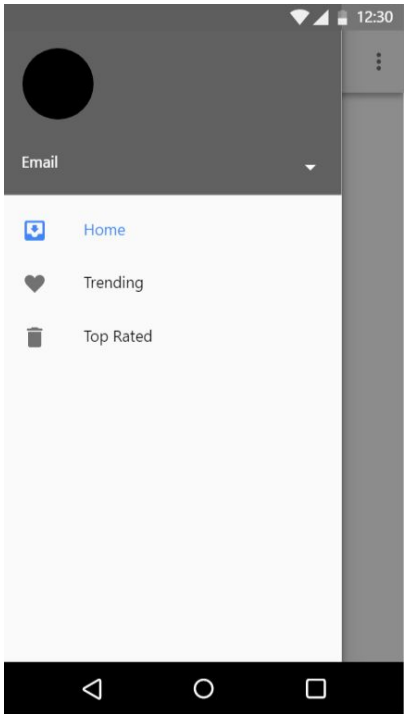
The main feed for the user currently signed in, they are faced with a list of posts that are most likely to be funny or entertain them. Each one can be liked, shared, or expanded to show comments(See Screen 2). Bookmark icon will be replaced by a collapse/expand carret. Ads will replace actual posts once in awhile. Long looks will aid machine learning to find the posts that best fit the individual.

Screen 2



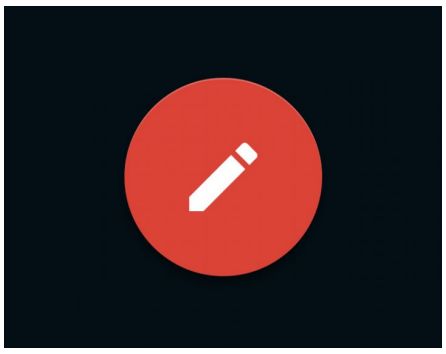
When expanded the comments section will be revealed to show each commenter's name and comment, if too long they will be elipsed with a ‘show more’ button. Comments will likely support 2 deep commenting.

Screen 3



The navigation drawer is the primary way to get around it will allow the user to switch from their home to a currently trending or top rated sections(Might change or be removed going forward). As well as login and logout of their accounts.

Widget



The icon will be different, a click on it will bring up an intent to pick a photo file and once selected it will be posted and the app opened. As simple as it is, I feel a small light widget is a better fit for my app.

Key Considerations

How will your app handle data persistence?

The app persists offline data to files using store and posts are posted to a webservice where they are saved to a backend database. As well as firebase to store comments instead of a contentprovider and loader. Any other data will be loaded using a loader alone.
A call to /Posts/UserID will return a JSON Array containing UUID, number of likes, and maybe name, author, or post date. A call to /Thumbnails/UUID will return a thumbnail for the image with the specified UUID. Which will be displayed in a blurred fashion with a progress bar until the full image is loaded from /Images/UUID.

Describe any edge or corner cases in the UX.

The primary corner case I can think of is loading old posts as the bottom is reached. I know this can be a little tricky from past experience.

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

NYTimes Store - Handles caching, persistent, and fetching data in the best way I have found so far.
RxJava - It has powerful threading abilities and makes io a breeze.
Retrofit - Simple RxJava and Gson supported web service interaction.
Firebase - Makes storage easier.
Rx2Firebase - Makes firebase flow more nicely with Reactive X.

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

Google, Facebook and Twitter OAuth will be used to authenticate users. AdMob will be used to generate revenue on each post. Firebase will be used to store and persist comments. Firebase job dispatcher will be used for on demand data loads instead of a SyncAdapter.

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

- Create BISP Android Architecture
    - Perform some tests

- Make plans for backend web service
    - Mockup Data Objects
    - Figure out libraries for access.
- Setup project in studio.
    - Import depends.
    - Setup play services.
    - Register app with OAuth providers.

Task 2: Setup Each Controller

- StreamContainerController
- StreamContainer
- ProfileController
- SettingsController
- CommentsController

Task 3: Remote Connections

- Setup Retrofit
- Create Stores
- Connect to Firebase

Task 4: Create Layouts

- Create layout files for each controller.

Task 5: Debug & Test App

- Phone
- Tablet
- Other Emulated Sizes