Contents

GitHub Username: ***iamareebjamal***

Feddup

Description

Feddup is an app for any fed up person providing anonymous image and content posting service. Users can open the app and see the already posted images and articles associated with them with posts ranked in descending order of downvotes.

User can view, downvote or favorite any post without any authentication. Second use case involves user posting an article. In order to do so, he/she needs to provide a post title, an article image, article content and a username (no requirement of registration, username will be anonymous) he/she wants the post to be associated with. The data is sent to the server which responds to the success or failure of the post creation with associated message with them.

The user can save posts in drafts and access them afterwards to edit or complete before posting. User can also save his/her favorite posts in the favorites section for convenience. The downvote cast on any post syncs in real time across all applications and thus the most downvoted post sinks to bottom and the newest post is automatically at the top.

Intended User

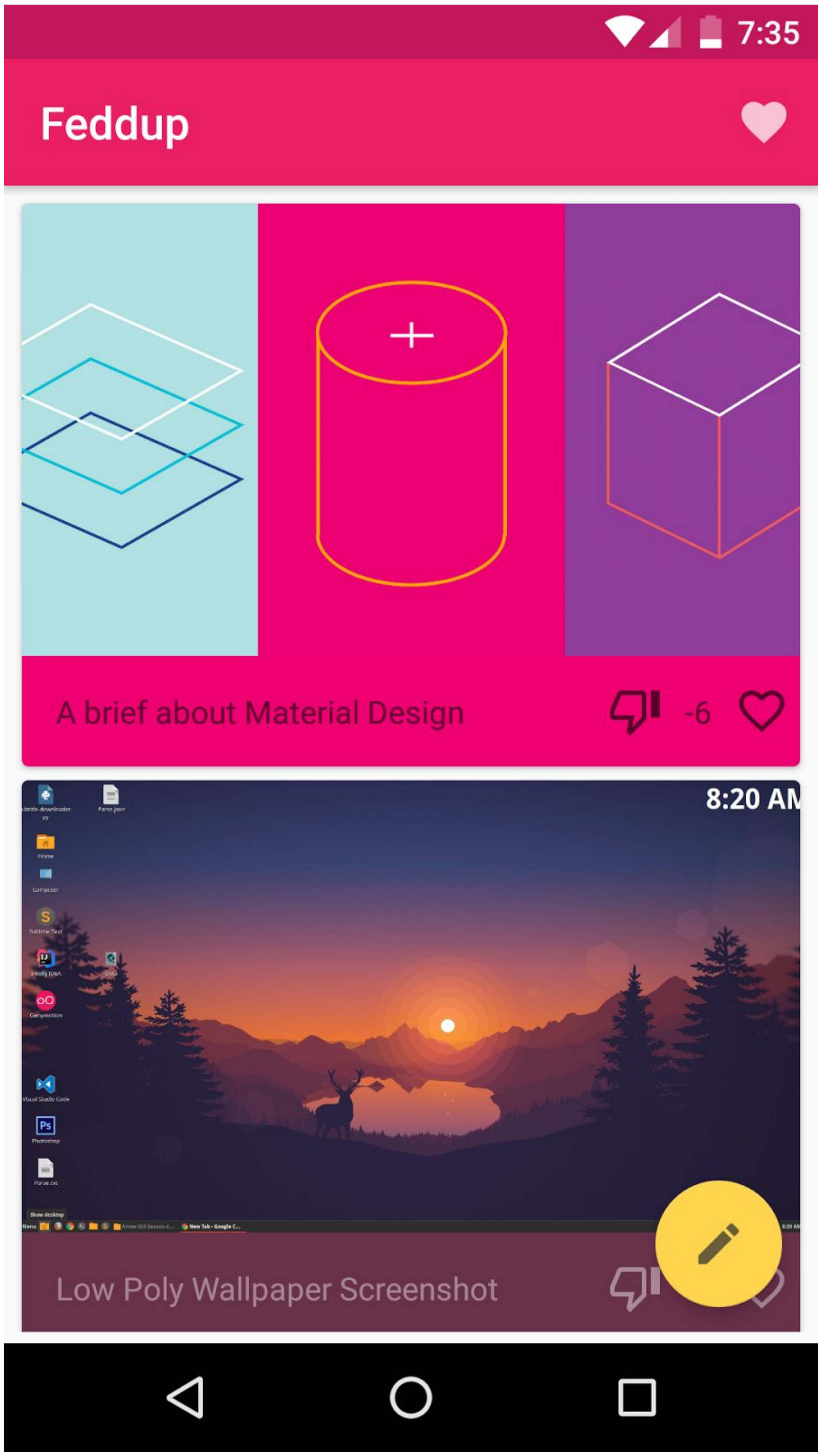
Intended for anyone who is bored and wants to browse or post photos with content articles with a downvotes based rating system anonymously.

Features

- Browse and View Articles
- Downvote Articles
- Favorite Articles
- View Favorite Articles
- Create Posts
- Save incomplete posts in Drafts

User Interface Mocks

Main Screen



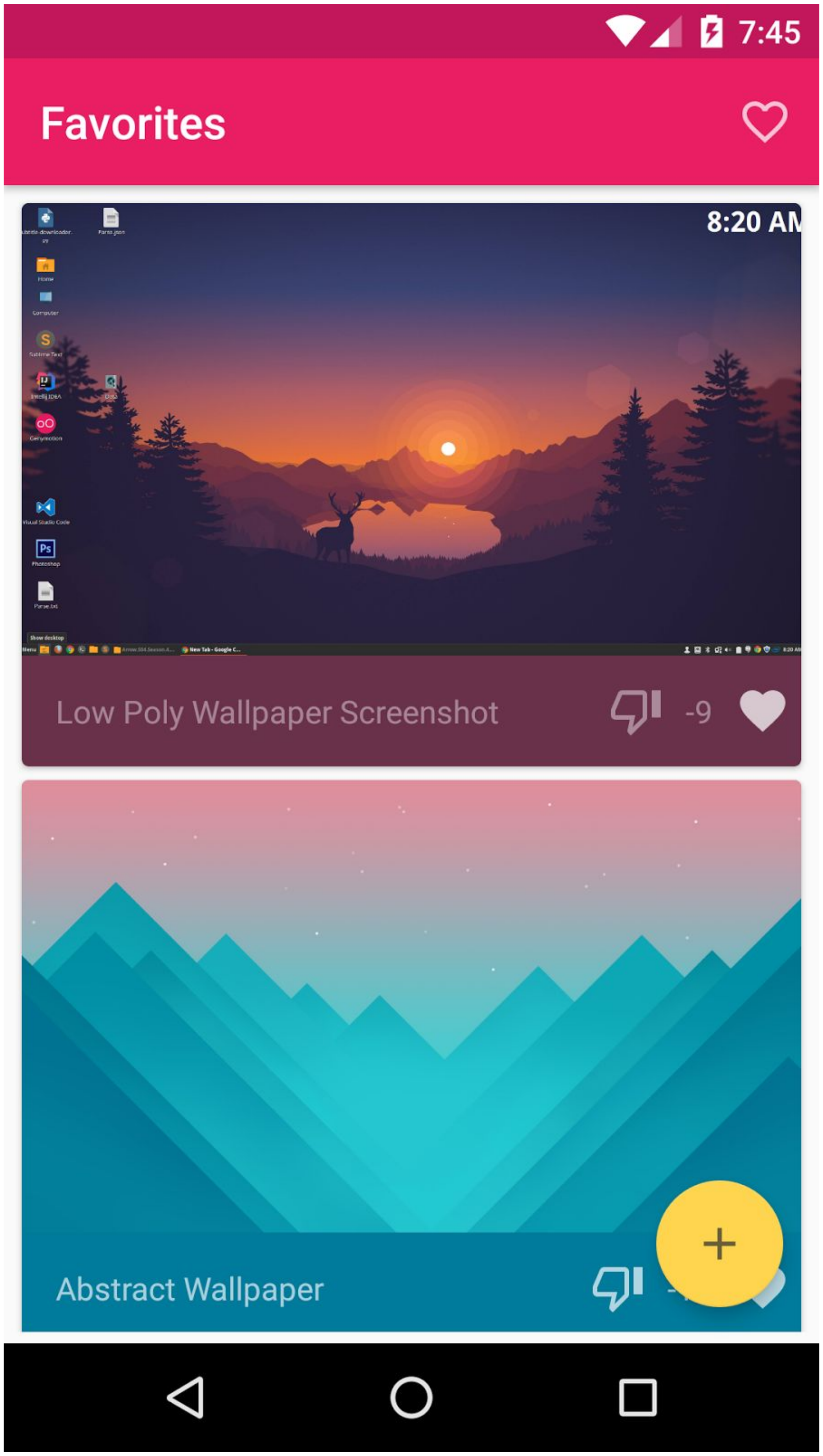
Main Screen of Feddup shows all the posts sorted in increasing order of downvotes. Each post shows the image associated with it,its title, the option to downvote it, the amount of downvotes it has, and an option for favoriting it.

Each card container for the post is clickable which leads to the Details Screen where we can see more details about the post.

The heart menu option in Toolbar leads to favorited posts.

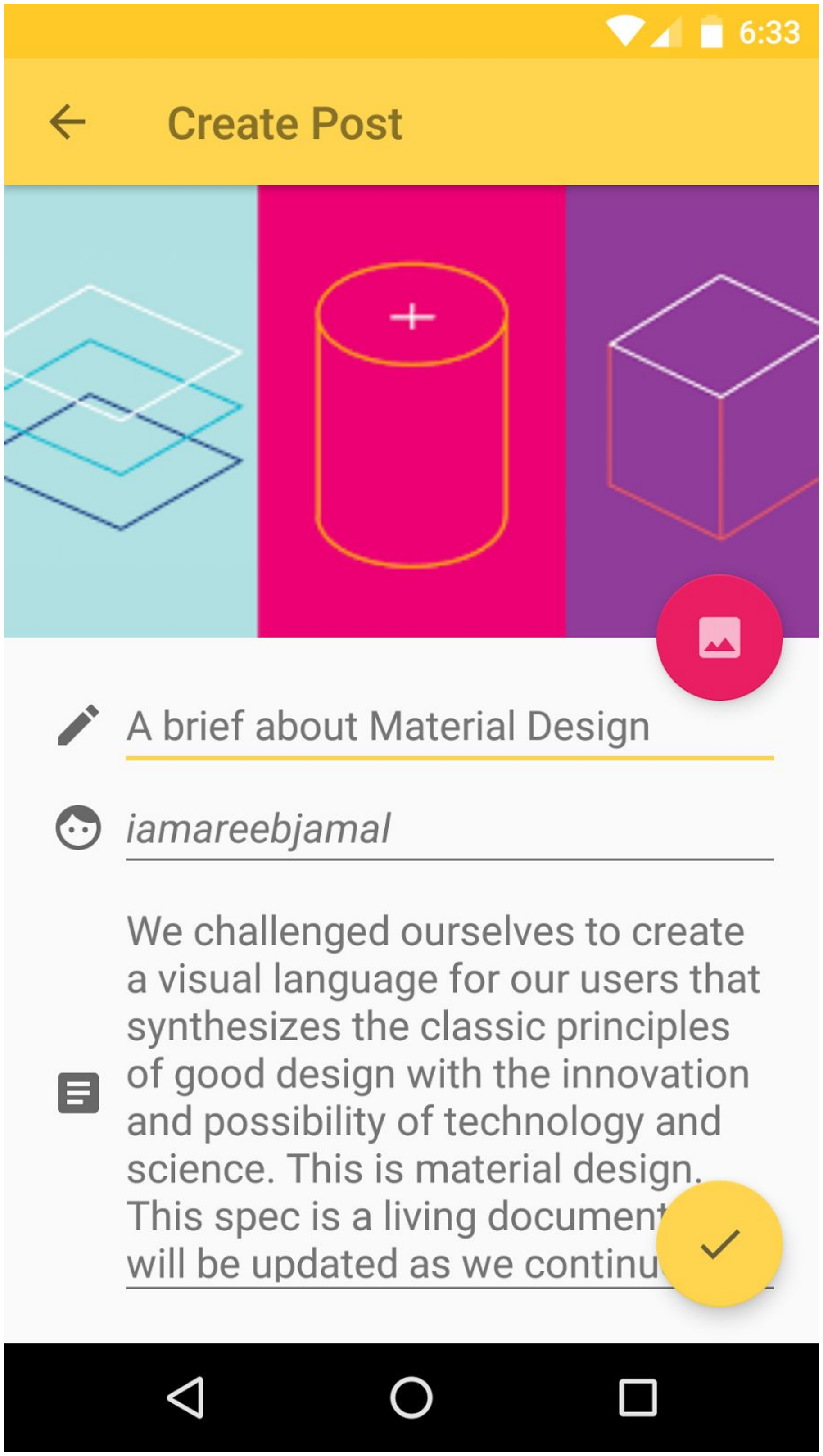
The Floating Action Button below leads to Post Creation Screen, where the user can create new posts.

Favorites Screen



Favorites Screen shows all posts the user has favorited. The card container of posts are same as they were on Main Screen. The empty heart menu option in Toolbar takes user back to the main screen.

Post Creation Screen



A brief about Material Design

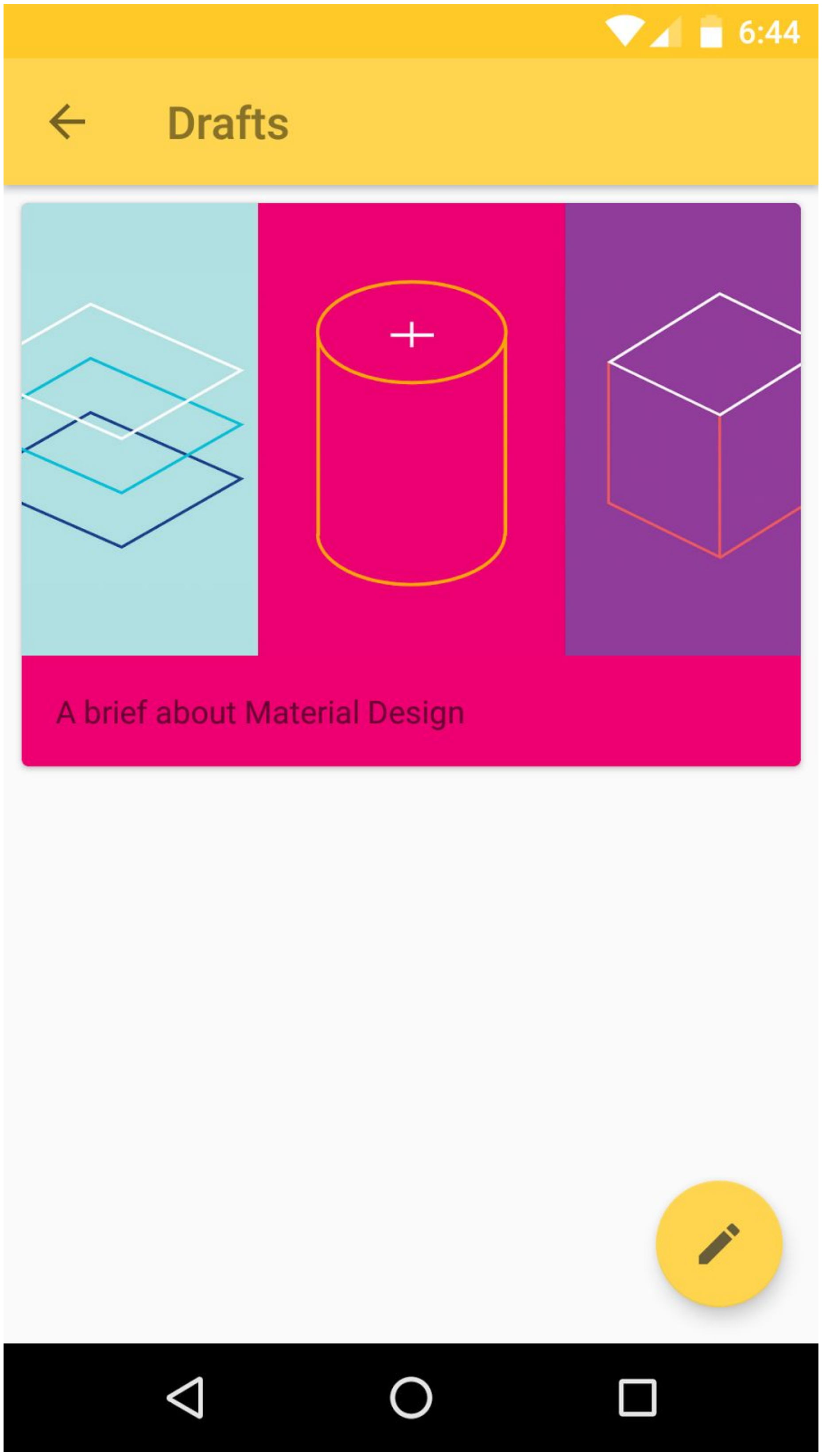
iamareebjamal

We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document will be updated as we continu

Provides option for the user to create a new post. Contains fields for Article Title, Username, Content and a FAB for attaching image to the post. The image below Toolbar shows the attached photo.

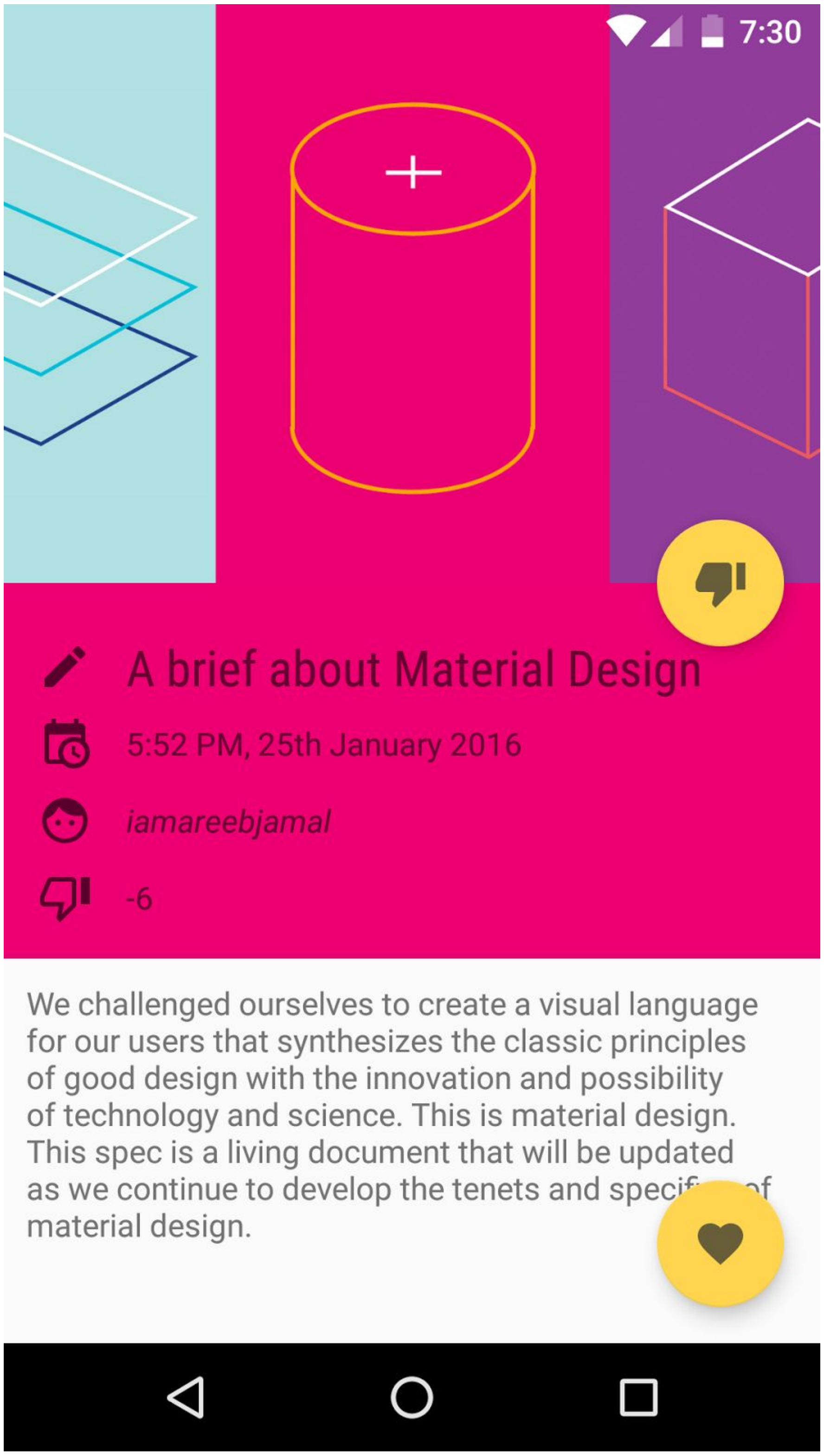
The Floating Action Button with check sign below posts the content and user is returned to main screen with his/her new post on top. If the user exits before posting or posting fails, the post is saved to Drafts.

Drafts Screen



Contains the unsent posts with card container showing the title and image if attached. Clicking on FAB opens Post Creation Screen where user can complete the post and send it to the server.
The FAB at the bottom of the screen allows to create new post.

Detail Screen



A brief about Material Design

5:52 PM, 25th January 2016

iamareebjamal

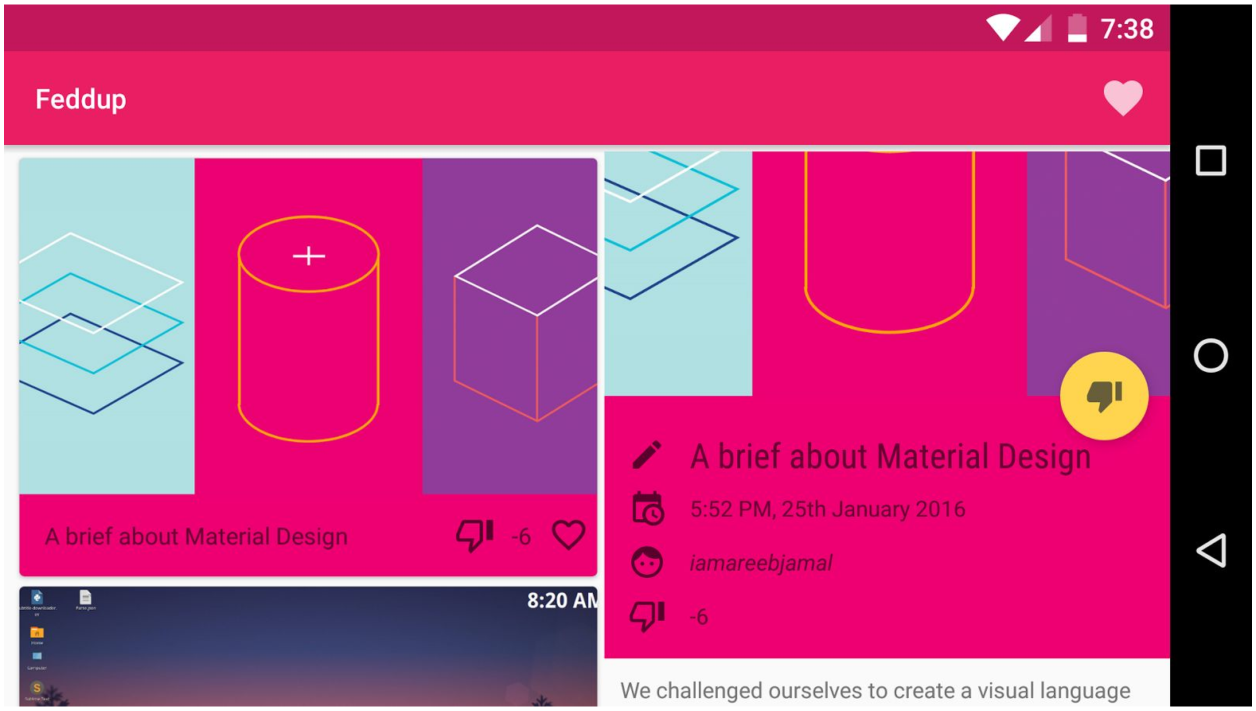
-6

We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document that will be updated as we continue to develop the tenets and specifics of material design.

Shows complete information about posted article with title, time posted, username of poster, downvotes, the content and the image attached to it.

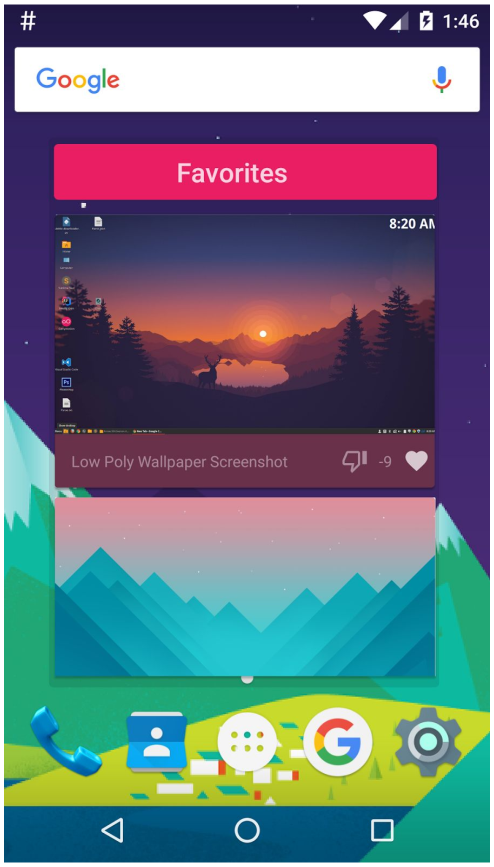
The top FAB allows to downvote the post and the one at bottom allows to favorite the post.

Tablet Screen


7:38
Feddup
A brief about Material Design
-6
8:20 AM
A brief about Material Design
5:52 PM, 25th January 2016
iamareebjamal
-6
We challenged ourselves to create a visual language

Tablet Layout of Main Screen with Detail Screen in form of Master/Slave layout.

Widget Screen



Widget Screen showing Favorite Posts with ability to downvote or remove them from favorites.

Key Considerations

How will your app handle data persistence?

Main Screen shows the up to date posts from a Firebase Realtime Database. The posts created by users are sent to a backend server which adds them to the Firebase Database. This will provide the automatic syncing when a downvote happens. The index in the Firebase Database node is defined on downvotes and the reference is used to sort them accordingly.

The local SQLite database with Content Provider front-end will store the favorite posts and the unsent drafts. In order to store favorites, only the firebase post keys will be stored in order to preserve the syncability of data in the Firebase Database. For drafts, all the information provided by the user will be stored and deleted when the draft is send or discarded.

Describe any corner cases in the UX.

- The full heart on Main Screen takes to Favorite Screen
- The empty heart on Favorite Screen takes to Main Screen
- The FAB on Main Screen or Favorite Screen takes to Post Creation Screen
- The image FAB on Post Creation Screen is used to attach image to post
- The check FAB on Post Creation Screen is used to send post to server
- The back button press or the back arrow in Toolbar on Post Creation Screen takes back to Main/Favorite Screen based on the UI stack
- A drafts menu on Main/Favorite Screen will be provided to access drafts
- Clicking on card of Drafts Screen takes to Post Creation Screen with provided information filled in
- Clicking on card of Main/Favorite Screen takes to Detail Screen showing details of Post

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

- ButterKnife for View Binding
- Retrolambda for Lambda Support
- Retrofit for network requests
- Picasso for image loading and caching
- Schematic for Content Provided Creation
- RxJava, RxAndroid for managing Asynchronous Tasks
- Android Support Library
- Android Design Library
- Google Services (Firebase)

Describe how you will implement Google Play Services.

- Firebase Database will be used for loading and synchronising posts
- Firebase Messaging Service will be used to implement user notifications

Next Steps: Required Tasks

Task 1: Project Setup

- Create new Android Project with Minimum SDK to be 16
- Configure build.gradle to include all libraries
- Gradle sync in order to check all libraries are included in project
- Add signing config and other required configuration in build.gradle

Task 2: Implement UI for Each Activity, Fragment and Widget

- Implement the Post Card Layout with required operations
- Implement RecyclerView with dummy data
- Implement Main Activity :
    - Implement Main Screen Fragment
    - Implement Favorite Screen Fragment
- Implement the Detail Fragment to show the post details
- Implement the Post Creation Activity with required form fields
- Implement Drafts Activity
- Implement the UI for tablet layout using Main, Favorite, and Detail Fragment
- Implement Widget UI for showing favorite posts

Use ButterKnife to bind views
Use Picasso to load images

Task 3: Implement backend connection for post request with Retrofit

The backend server endpoint for creating a new post is located at ***https://iamfeddup.appspot.com/new_post*** with required fields as follows:

- ***title*** : The title of new post *(String)*
- ***user :*** The user new post is made under *(String)*
- ***content :*** The article content of the new post *(String)*
- ***image :*** The image to be attached to the post. *(Image File with correct mime type)*

*Note that the request type must be **POST**, NOT **GET** !*

When an unsuccessful request is made, a JSON response with appropriate HTTP response code and error message is returned in this format :

```
Status : 400 Bad Request
{
   "message": "Missing Data : ['title', 'content', 'user']",
   "error": true,
   "user": ""
}
```

When a successful request is made, a JSON response like this is received :

```
Status : 201 Created
{
   "url":
"https://res.cloudinary.com/iamfeddup/image/upload/v1485657359/iamareebjamal/a_brief_about
_material_design.png",
   "user": "iamareebjamal",
   "key": "-Kbc43TZuCm4RerVuEyt",
   "error": false,
   "message": "Post created successfully"
}
```

The subsequent keys have following meaning :

- ***url*** : The url of posted image
- ***user*** : The username of author of post
- ***key*** : The firebase database item key
- ***error*** : Boolean representing error during post creation (false on successful creation)
- ***message*** : Error or Success message explaining the state of request

Set up Retrofit to make POST request to backend

Task 4: Implement Firebase Database and Database UI

Get the ***google_services.json*** from Firebase Console and add it to ***app/*** directory in your Project and set it up accordingly.

The firebase reference for posts is ***posts/*** and contains following keys in each items :

- **title** : Title of post
- **url** : The url of attached image
- **user** : The username of author
- **time** : The timestamp of when post was created
- **content** : The article associated with the post
- **downvotes** : The downvotes a particular post has

*The index is defined on **downvotes** and it the only writable field*
*Make reference to **posts/** and define **orderBy** attribute to **downvotes***

Task 5: Create SQL Database and Content Provider

- Use Schematic to create basic SQL Database
- Create table to store favorite posts
- Create table to store drafts
- Create table to store disliked posts to ensure one user only dislikes a post once
- Implement appropriate Content Provider methods

Task 6: Connect implemented business logic to UI

Replace dummy data with :

- Firebase Realtime Database Reference
- Favorite Content Provider
- Draft Content Provider

*Use **CursorLoader** to load favorites and drafts into their respective activities.*
*Use **AsyncTask** to make network request to the backend server to post new content using Retrofit*