

Jason Conklin
Bryce Corbitt
Kenneth Desrosiers
Elizabeth Kirschner

CS4518
10/16/2020

Group 3 Final Report

Milestone 1: Proposal

During the first milestone of this project, we proposed a location-based drawing app. This section helped us to set baseline functionality we wanted to implement as well as goals that we wanted to attain. The app's features, motivation for creation, target platform, and market competitors are defined in this section. Our app idea didn't change over time; we were able to implement the app as specified in this milestone.

Introduction

There are a multitude of instant messaging apps available on smart devices. Notable examples include WhatsApp, Snapchat, Messenger, and GroupMe. These applications, though sophisticated and useful, are limited in comparison to real-world forms of communication. These applications require you to know your message's recipient beforehand, using a phone number, username, or other form of identification. In real life (prior to COVID-19), one could simply walk up to someone nearby and strike up a friendly conversation. This is particularly nice in situations where location is a primary unifying factor behind the presence of the individuals, such as a movie theater, high school, convention, or college campus. Many of these locales hastily create their own clunky, error-ridden solutions, which are used only for the associated context, yet remain on the Play Store or App Store unused.

In addition to the problems with these applications and the limits of Instant Messaging, there is also a decline in the artistic value of personal communication. Written communication, and even early typed messages had the advantage of a personal artistic touch, be it through stylistic fonts, handwriting, sketches, or signatures. This historical artistic subtlety to personal communications has been diminished through the use of many of the top instant messaging applications' single fonts and uniform emoji keyboards.

We have identified the need to create a single application to address both problems: instant messaging which is location-based rather than contact-based, and an app that allows for artistic and calligraphic flair in personal messages.

Importance and Motivations

Currently, there are no apps that allow users to leave location-based drawn messages. Our app aims to fulfill this role. Our motivation is to allow users in an area to connect in a fun way with simple drawings. Allowing users to connect locally is the primary goal of this app. The drawings provide a means of interaction not found in other location-based messaging applications. Introducing a messaging app based around drawings is a creative idea that will create a new avenue for location-based messaging. Right now, most social media platforms provide a curated list of media from a user's followers and people they follow. This app will bring together users based solely on location, which will allow users to interact with a wide variety of nearby individuals who also use the app. Users can interact based on content rather than social circles. A key motivation of this app is to create a service that is about interaction through user created content.

Vision and Key Features

We aim to provide a unique platform for local communication that encourages creativity. Posts on the platform will take the form pictures that are drawn by users. Users will be securely identified using the Google Identity Toolkit API. An interface will be created that enables each user to draw by interacting with an on-screen canvas. Upon submission, a rasterized bitmap of the user's canvas will be stored as a urlencoded PNG in a MongoDB database along the user's current coordinates. To browse posts, a feed will be implemented that serves posts filtered by proximity to the user's current coordinates and a user-configured distance radius. A voting system will also be implemented such that posts in the feed may either be sorted by popularity or by relative distance.

Existing Competitors

The market for location-based messaging apps has risen and fallen in popularity over time. In the past, apps like Yik Yak were widely popular. This app allowed users to anonymously share posts with other users who were within a certain location radius. Currently, location-based messaging does not have any leading representatives in the social media app market. Some apps like Snapchat have options within the app that enable users to create location-based community posts (Snap Map/Our Story). There are, however, a few apps that completely fall into this category. One example of a local-based messaging app is [Jodel](#). Jodel allows users to make public community posts, chat with nearby users, and share moments through pictures and videos. Just like Reddit, this app allows you to like posts and collect points called Karma. Another example is [Whisper](#). This app has similar functionality to Jodel; it allows users to make community posts and chat with other users. Unlike Jodel, Whisper allows users to see the latest

and more popular posts no matter where in the world they originate from. In our app, we wish to implement the community feed and direct messaging features that are shown in these two apps.

Target Mobile Platforms

The application will be created for the Android mobile operating system using API level 27 (Android 8.1.0). Communication with the back-end will be implemented as a REST API, such that the service may be accessible to other operating systems in the future.

Estimated Workload Division

A group meeting was held to divide the workload for the project proposal. Responsibility for each section was distributed as follows:

Introduction: Jason Conklin

Importance and Motivations: Elizabeth Kirschner

Vision and Key Features: Bryce Corbitt

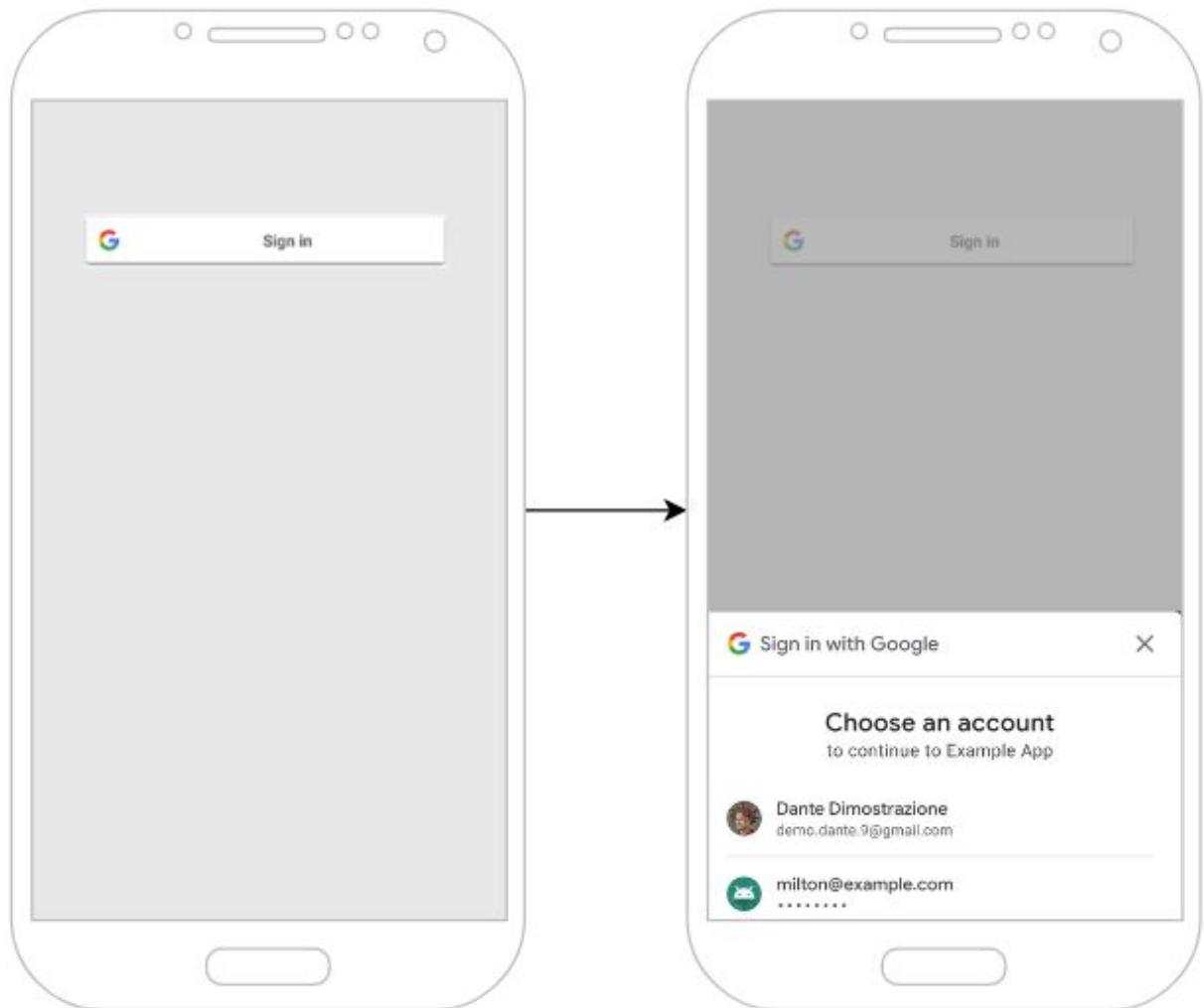
Existing Competitors: Kenneth Desrosiers

Target Mobile Platforms: Bryce Corbitt

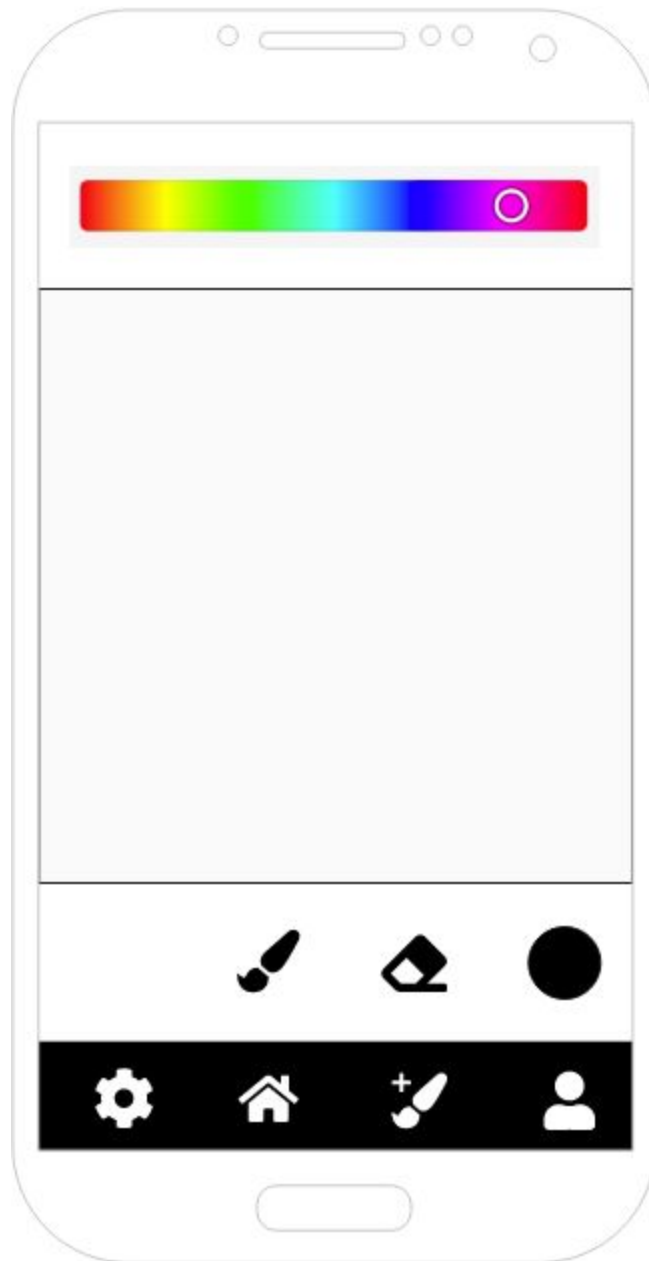
Milestone 2: UI Mock-ups

Our app closely followed the design of the UI mock-ups. We liked these designs because they were simple and effective. A simple design allows the user to use the app without any issues. Creating these UI mock-ups was very helpful when starting the development stage; they allowed us to have a basis to create our UI on. This increased development productivity by wasting less time on design choices.

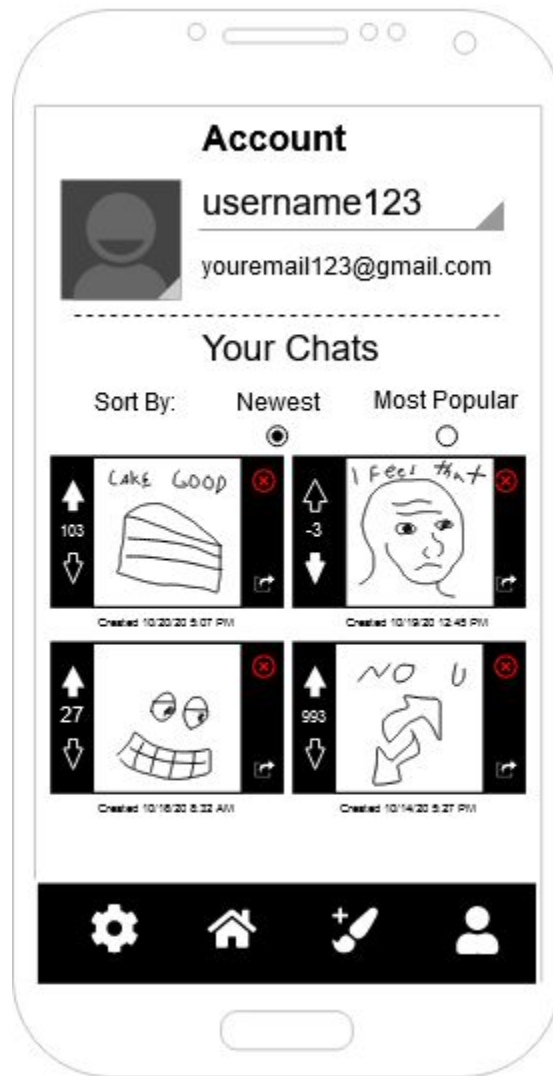
Login



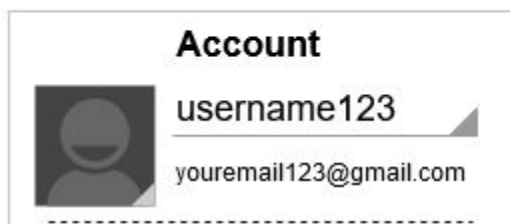
Creation Canvas



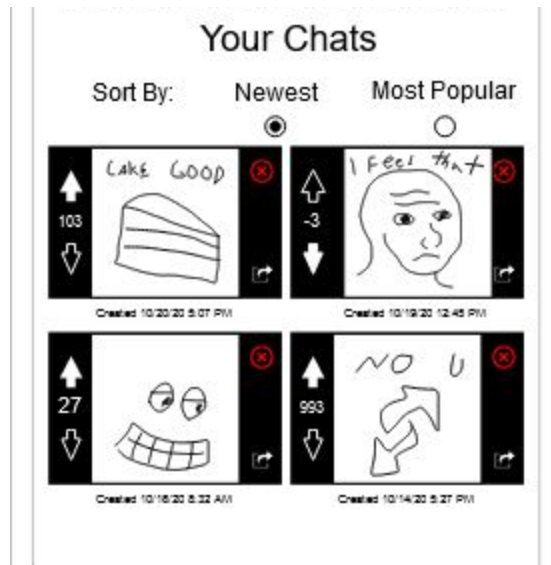
Account



The Account screen allows a user to view their own user account information, including a history of all of their chats and associated likes. The screen has three main parts: the account info section, the account history section, and the navigation bar.



The account info section (displayed as 'Account') provides basic information about the user, including their current username, which is editable, their profile picture, and the email associated with their account.



The chat history section (displayed as ‘Your Chats’) provides basic information about each of the account’s chats. This section displays the likes on each chat, and provides the ‘delete’ and ‘share’ options for each chat. The user can use either of these options by pressing the appropriate symbol on the specific chat that they would like to delete or share. Also the user can sort their chats by number of likes(‘Most Popular) or time (‘Newest’).

Drawing Feed



The Drawing Feed is the home screen of the application for authenticated users, and is used to browse pieces on the platform. The screen is broken into three main parts: the topbar, feed view, and navigation.



The topbar is fixed to the top of the screen, and houses controls used to fine-tune the user's browsing experience. A toggle switch is present on the left side of the bar to switch between

“Local” and “Global” browsing modes. When browsing globally, users may view the feed from the location of their choosing (configurable by pressing the location pin) independent of their physical location. However, users in this mode will be unable to interact with posts that are shown. When browsing locally (which is set by default), users will be able to interact with posts, but the location which the feed is created from is locked to the users’ physical location. A number is also present to the right of the browsing location that represents the browsing radius, or the maximum distance from the current location which posts will be shown. Tapping this view will also direct the user to configure the radius on the preferences screen. A refresh button is located on the right side of the screen, which refreshes the feed to see new posts and updated information.



The feed view displays each nearby post in a scrollable feed. The left side of each post displays the number of votes the post has received. When browsing in “local” mode, users may contribute to evaluating the post by pressing the directional arrows to increase/decrease the vote by one. The upper right corner of each post has an “X” button that may be used to hide posts that a user does not wish to see. A “share” button is also present at the bottom right corner of each post, which creates a permalink for the post for users to share the post with others.



The navigation for the application is fixed to the bottom of the screen. The four icons shown in the navigation represent the Preferences, Feed, Canvas Creation, and Account screens respectively. Pressing any icon will direct the user to its corresponding screen.

Preferences

Select your distance units:

Miles ☒ Kilometers ☐

Select your location radius for posts (mi):

5 10 15 20 25


Choose a location:

Enter a Location





Sort by:

Location ☐ Popularity ☒

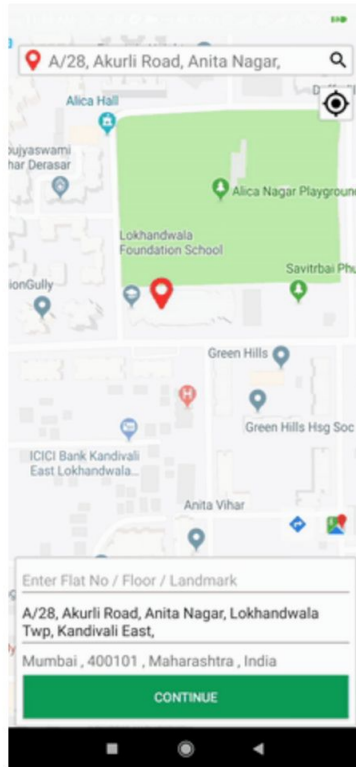
Logged in as:

 xxxxx@gmail.com [Logout](#)

[Delete Account](#)

The preferences screen allows users of this app to configure many different options within the app. The first option allows users to choose between miles and kilometers as their distance unit of choice. They can also choose to minimize or expand their location radius. This allows them to filter the amount of posts they are able to see and respond to. Users of this app will also be able to search any location in the world other than their own to see what posts people are making in those areas.



We will be using a library called [LocationPicker](#). This library will enable users to be able to search for any location in the world.

In global mode, the location radius slider allows them to filter the amount of posts they can see relative to the location (that isn't their own) they entered. By default, posts are sorted by popularity, but users will also be able to sort by the poster's location relative to their own. Further down the screen, users will be able to see what account they are logged in as and log out if they wish to do so. Lastly, users can delete their accounts.

Estimated Workload Division

A group meeting was held to divide the workload for the UI Mockups. Responsibility for each section was distributed as follows:

Login: Elizabeth Kirschner

Creation Canvas: Elizabeth Kirschner

Account: Jason Conklin

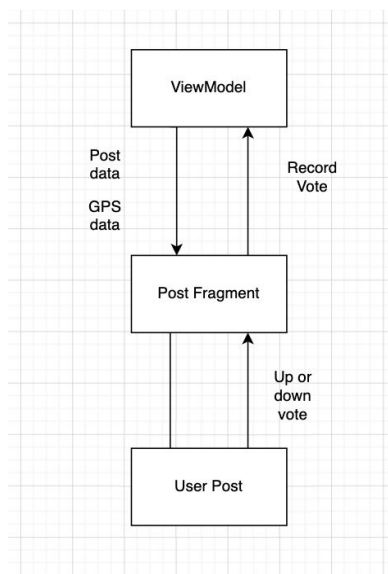
Drawing Feed: Bryce Corbitt

Preferences: Kenneth Desrosiers

Milestone 3: Functional Design

For this milestone, we defined the design patterns we planned to use as well as the various areas of the stack the app will use. The persistence strategy, mobile sensor, network communication, draw functionality, and navigation between fragments was implemented as described in this part. We used a node.js backend server, used OAuth2 via Google for logging in, and the LocationPicker library for choosing a location for Global browsing.

Design Patterns



This pattern represents a user interaction with an individual post. If a user is within appropriate GPS range, they can view a post and either up or down vote. Up votes and down votes will impact where in a user's feed posts appear.

Persistence Strategy

A node.js [express](#) server will be hosted to store various data for the application. Currently, users and posts are the only data models that are planned to be stored in the back-end. A [mongoDB](#) database will be used to store these models. Schemas for model will be defined using the [mongoose](#) Object Document Mapper (ODM).

Authentication will be performed using [OAuth2 via Google](#). Upon a successful Google login from a user within the app, an access token that is generated by the login will be sent to the back-end to retrieve information on the user's Google account. If a user with a corresponding Google ID does not exist, a new account will be created.

Mobile Sensor Usage

For our application, users will be able to send messages that will contribute to a local feed that anyone nearby can also see. In order to do this, our app will request users' locations using the GPS sensor in their smartphones. If users deny the use of their GPS sensor, they will only be able to search for other locations to view posts from, but not be able to make any posts themselves.

Network and Service Communication

To enable the app to communicate with the back-end, a REST API will be developed. The [Retrofit](#) library will then be used within the app to consume the API.

In addition to API communication with the back-end, there are two services that will be used on the application to facilitate location information. The [Fused Location Provider API](#) will be used to retrieve the current location of each user. A user's location will only be used when publishing a new post and interacting with the feed in **Local** browsing mode. The [Google Maps SDK](#) will also be used (in part with a [LocationPicker](#) library) to allow users to select a location when browsing in **Global** mode. When a user browses globally, they are able to see posts from areas outside the vicinity of their current location, but will not be able to upvote/downvote any posts. Users browsing locally may vote on posts that appear in the feed. Location selection will only be used for global browsing, and does not apply to the location used when creating new posts.

Navigation Between fragments

Our app will use single activity architecture and fragments to manage the UI and screen navigation. We will use the standard navigation activity as a starting point to implement our own features. The navigation library will be used to perform the heavy lifting of screen navigation and starting or retrieving fragments as needed. Each fragment will use the MVC architecture to manage the UI and user interactions.

Canvas / Draw Functionality

The application will take advantage of touch screen functionality to allow a user to draw pictures on a virtual ‘canvas’ using their fingers. Android provides several types of touch events that we will use to determine how this part of the application responds to the user. Therefore, we will not use a third-party tool for this functionality. A drawing can be easily encoded into an image file, such as a BMP, for storage on the remote server. This draw functionality will be tightly integrated with the Networking and Communication functionality to allow users to view one another’s creations.

Estimated Workload Division

A group meeting was held to divide the workload for the functional design. Together, the team defined a list of core functional components to be documented under each “functional building block” section where applicable. Responsibility for each functional component was distributed as follows:

Navigation between Fragments / ‘Screens’: Elizabeth Kirschner

Back-end (server, authentication, API, database): Bryce Corbitt

Determining User Location: Kenneth Desrosiers

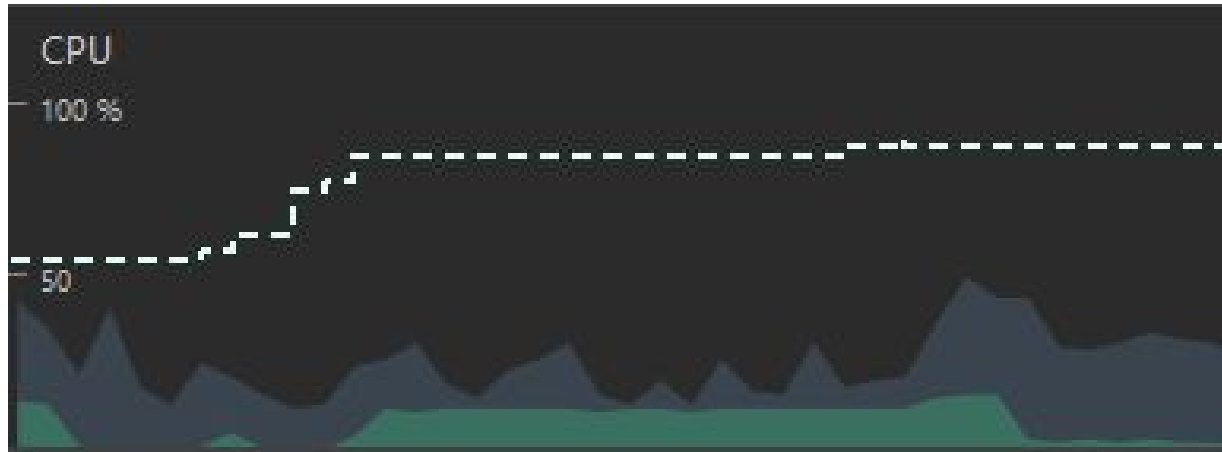
Canvas / Draw Functionality: Jason Conklin

Fetching data from a server: Bryce Corbitt

Milestone 4: Implementation/Evaluation

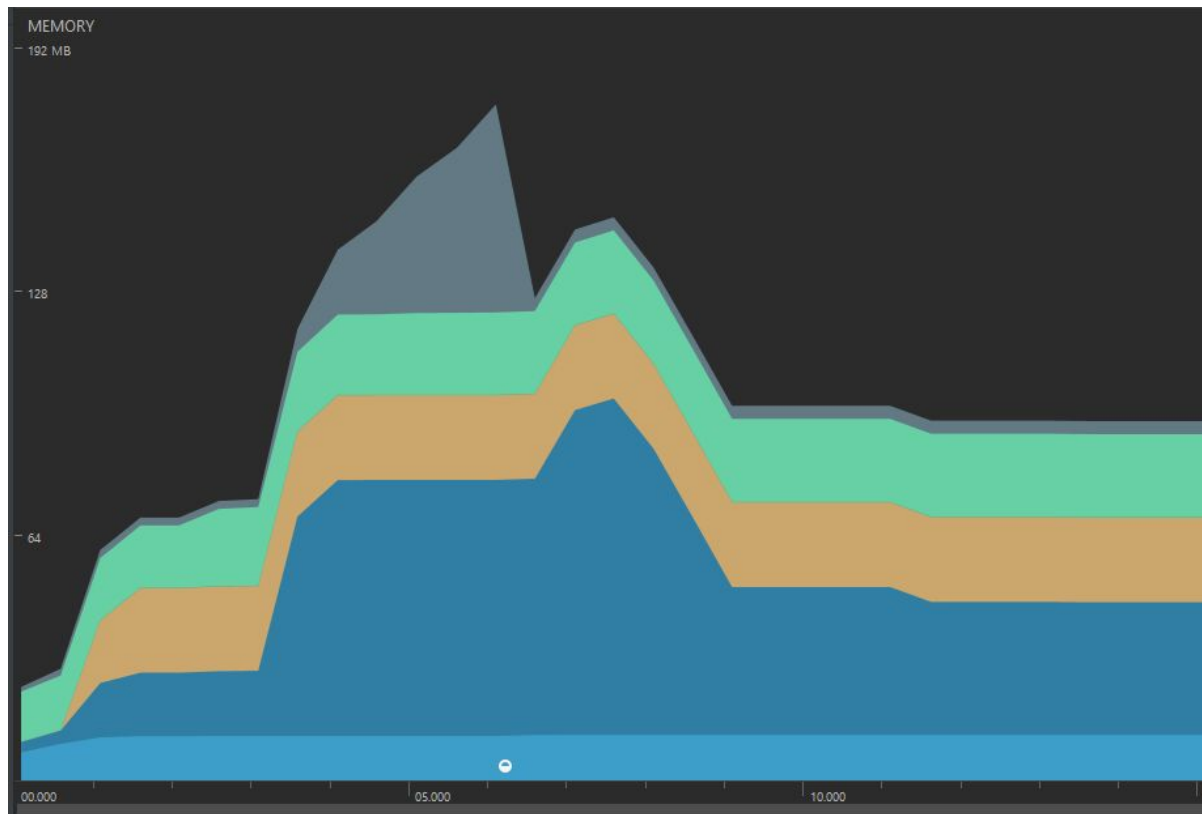
When creating this app, we used a node.js express server to serve as the backend. This server is hosted on a WPI virtual machine. This separation of the client-side app code and the backend server code allows for thorough testing of each one without interfering with the other. There are many separate parts during the implementation process that were brought together. We had to design our application to generate media based on user input, encode that media for upload to a remote server, store data efficiently on the server-backend, and retransmit that data to other users of the app, all for the basic functionality of sharing and communication. Also, the navigational nature of our app created a situation with many asynchronous tasks, often working simultaneously, interfacing and switching between multiple fragments. Our team was able to use our diverse skill set to divide and conquer these many problems. We tackled individual problems on our own, and then merged our solutions together to create the final product. We heavily used version control software and certain Agile methodologies in order to complete the project efficiently and effectively.

After the creation of our mobile application, we wanted to have an accurate measure of its performance compared to our original goals. We used the Android Studio Profiler Tool to determine the resource consumption of our application in four key areas: CPU, Memory, Network, and Energy. All of the below data was gathered while running our application on a real-world, physical Samsung Galaxy S8 running Android 9.



CPU utilization during a 15-second interval of application usage

Our application performed extremely well in terms of CPU utilization when it came to normal use. The utilization of our application (plotted light green in the above chart) averaged about 10 to 12 percent of available CPU resources during normal usage, with brief spikes of 15 or 16 percent during intensive operations. We are pleased with this result, given that this 12 percent amounts to approximately equal to or less than normal background processes, and thus leaves around 65% of total processing resources available to this particular device. This means that our app, like many industry-leading messaging apps, may be kept on in the background, without affecting user experience of their device negatively, and allowing them quick access to a social, creative medium for communication.



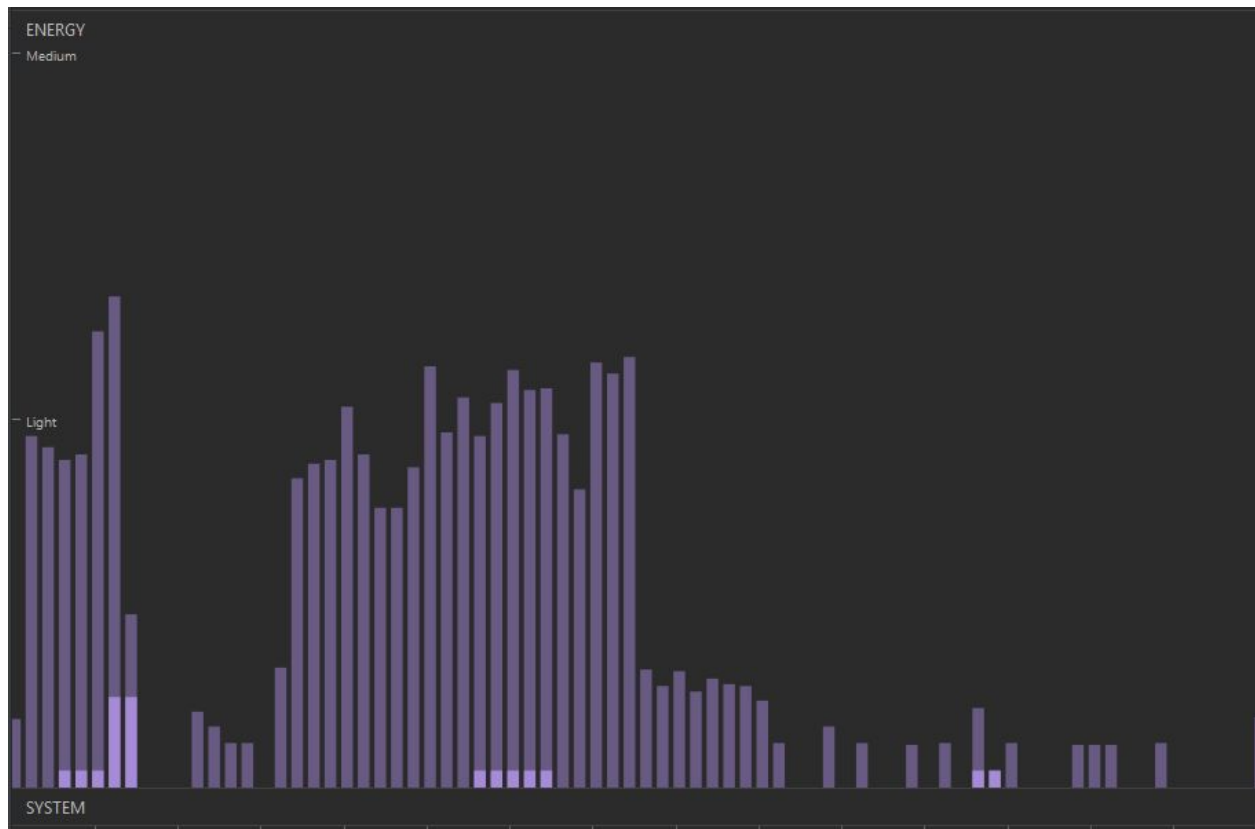
Memory utilization during a 15-second interval of application usage

Our memory utilization over the course of running the app produced satisfactory results. Near the beginning of app launch, the app used a little over 64 MB of memory. This was tested on a Galaxy S8 running android 9. This phone has 4 GB of ram, so 64 MB of memory usage is very efficient. This allows the user to have multiple apps running at the same time with no issues. As the app stayed open and continued to be used, the memory usage peaked at about 180 MB. This is significantly higher than the earlier memory usage, but this is still extremely low when compared to the phone's total available memory. The memory usage lowered over time until it leveled off at about 100 MB. This app shows that it does not require an unreasonable amount of memory to be successfully run.



Network utilization during a 15-second interval of application usage

Our application performed decently well in terms of Network utilization when it came to normal use. The utilization of our application (plotted light blue in the above chart) tended to have zero utilization most of the time, but caused huge spikes in network utilization when retrieving data from the remote server. Despite the ‘spiky’ behavior of our application’s network utilization, it transmits only simple drawn images, which significantly cuts down on the amount of data being exchanged. This means that even when network resource utilization is highest, our application still only uses about a MegaByte at a time, and even then, only in disparate periods. We are pleased with this result, given that this creates a positive user experience, since the user’s personal device offloads much of the networking work to the remote server, and there is less latency experienced by the user.



Network utilization during a 15-second interval of application usage

Our application performed well in terms of Energy consumption. The utilization of our application (plotted purple in the above chart) averaged at or under the Android Studio Profiler's 'Light' categorization. Given that this app is ideal for prolonged periods of location-based gatherings, such as conferences, schools, campuses, and sporting events, the low energy consumption enhances the app's ability to meet the goals we outlined for it earlier in the term. Namely, a longer-lasting battery encourages increased socialization and creativity between users of the application.

```

1  package com.bryceorbitt.artsyapp.ui.canvas
2
3  import ...
4
5
6
7
8
9  class CanvasViewModelTest {
10
11      lateinit var canvasViewModel: CanvasViewModel
12
13      @Before
14      fun setUp() {
15          canvasViewModel = CanvasViewModel()
16      }
17
18      @Test
19      fun testCanvasViewModel() {
20          assertEquals(canvasViewModel.getColor(), Color.HSVToColor(CanvasViewModel.hsv))
21      }
22  }

```

Unit test implementation

Our unit test implementation tests the getColor functionality of our canvasViewModel. Unit tests prove to us the implementation of our app works. Our unit test model creates an instance of the CanvasViewModel class. This class holds critical data about our canvas fragment and is therefore critical to ensure it works. The test proves that the ViewModel is being initialized correctly and that its methods function as expected.

Estimated Workload Division

A group meeting was held to divide the workload for the implementation and evaluation. Together, the team defined a list of the tasks and deliverables needed to implement all of the parts of our application and remote server backend. We then also divided the evaluation steps between the team. Responsibility for each part of the implementation and evaluation was distributed as follows:

Implementation

‘Feed’ Fragment : Elizabeth Kirschner

Back-end (server, authentication, API, database): Bryce Corbitt

‘Preferences’ Fragment : Kenneth Desrosiers

Client API (Neofetch + GSON): Kenneth Desrosiers and Bryce Corbitt

Canvas / Draw Functionality: Jason Conklin

Account Fragment: Jason Conklin

UI Design: Elizabeth Kirschner

Evaluation

JUnit Testing: Elizabeth Kirschner

Demo Video: Bryce Corbitt

Report Drafting and Editing: Kenneth Desroisers

Android Resource Profiling: Jason Conklin

Report Writing: All

Backend Repository: <https://github.com/CS4518-Group-3/back-end>

App Repository: <https://github.com/CS4518-Group-3/ArtsyApp>