Brainpick

You pick our money, we pick your brain

Group 2

Kwok Hin Kwan, Billy (20188664)
hkkwokaa@connect.ust.hk

Shin Wai Ching, Martin (20191104)
wcshin@connect.ust.hk

Introduction

Problem

Researchers and data scientists in both the commercial world and academia have wasted a ridiculously huge amount of time, money and manpower on data collection, cleansing and processing. In fact, a survey [1] reveals that data scientists spend 80% of their time on these tedious tasks instead of the value-adding parts of their duty, like modeling, algorithm design, test and evaluation, etc.

At the same time, urban dwellers have also wasted countless hours on unproductive tasks like commuting and queuing. When they need to wait, most of them are just killing time by playing mobile games, watching junk videos or browsing social media newsfeed. Such phenomenon manifests the inefficiency of modern urban lifestyle.

Approach

What if everyday people can help researchers and data scientists to perform short tasks on their smartphone in exchange for micro-payment?

Brainpick makes it possible by bringing these two groups of people together in a single platform. It is an intelligence crowdsourcing app that minimizes researchers time and effort in data collection and processing. Both commercial and academic researchers can submit tasks on the platform and the system will automatically distribute micro-tasks to all users, for example, photo tagging, price estimation, questionnaire, sentiment of words, and so on. They can then receive monetary rewards, either provided by submitters directly or sponsors of the request. By doing so, even average people can contribute to state-of-the-art research projects or help companies develop products that solves million people's problems. This is also the reason why we call the app Brainpick - picking the brain of the crowd.

Goals

- Go through design thinking cycles from low-fidelity prototypes to MVP
- Gain experience in writing iOS app with React Native
- Learn to connect React Native to native iOS modules if certain functionalities cannot be achieved in pure React Native
- Develop a functioning iOS app with intuitive UI/UX design
- Support app services with Firebase as backend
- Demonstrate that our solution can help machines become more intelligent by crowdsourcing human insights

Design and Implementation

Requirement Analysis

The table below shows a list of requirements in our project.

Requirement	Result
It should take less than 1 second to load from splash screen to main app	Achieved
Butter smooth animation and interaction that is comparable to truly native apps	Achieved
The app should consist of 3 main sections	Built ‘Discover’, ‘Contributions’ and ‘Settings’ page
Each story should not be more than 4 layers	Only two layers from Discover to Task completion page
Data fetching and update should be close to real time	Adopted Firebase Firestore and Redux to support super fast real-time data update and fetching
Data structure should be well maintained and extendable	Used Firestore hierarchical structured collections as data storage and Redux for app state management and data propagation
The app should be cross-platform	Built with cross-platform components in React Native so that both iOS and Android can be compiled with the same JavaScript codebase

Design

Visual Design

Before coding the user interface, we have drafted a storyboard that indicates the basic flow of the app using Sketch. We then asked our friends for feedback and further improved the design. This saved us from repeatedly modifying codes to tweak the design of user interface. A draft of the storyboard is shown below.

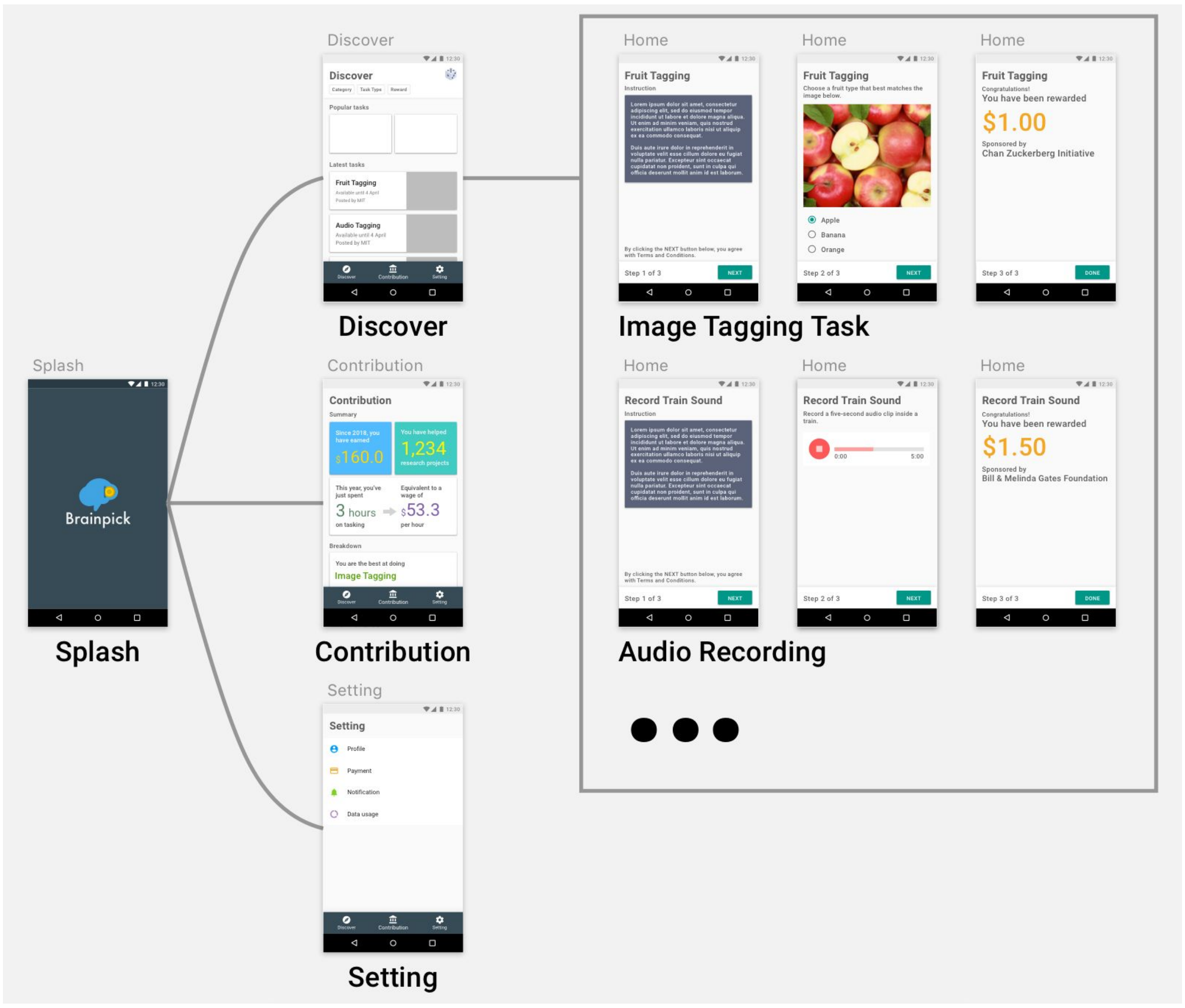


Figure 1. A draft of storyboard

System Design

We decided to write a React Native app instead of a fully native one because it minimizes the development cost so that we can focus on the functionalities instead of pure software engineering. Our team members have already built several native Android apps in Java and Kotlin so we think that we would learn much more using React Native than native codes. Also, our team have solid experience in JavaScript and the web version of React, which flattens the learning curve of React Native.

Implementation

The app is implemented with a series of components. Each of the major components and their relations with each other will be discussed below. For details, please refer to our Project Github Development branch[1].

React Navigation

We use the library React Navigation to handle the navigation of Brainpick. The entire navigation structure of the app consists of a tree of navigator objects and React component. Navigator object is a special type of React components that manages the navigation among a group of other React components. For example, switch navigator manages a group of mutual exclusive components, while stack navigator manages a group of components stacking on top of each other. Figure 2 illustrates the navigation structure of the app (grey lines indicate the navigation hierarchy and red lines indicate the flow of navigation)[2].

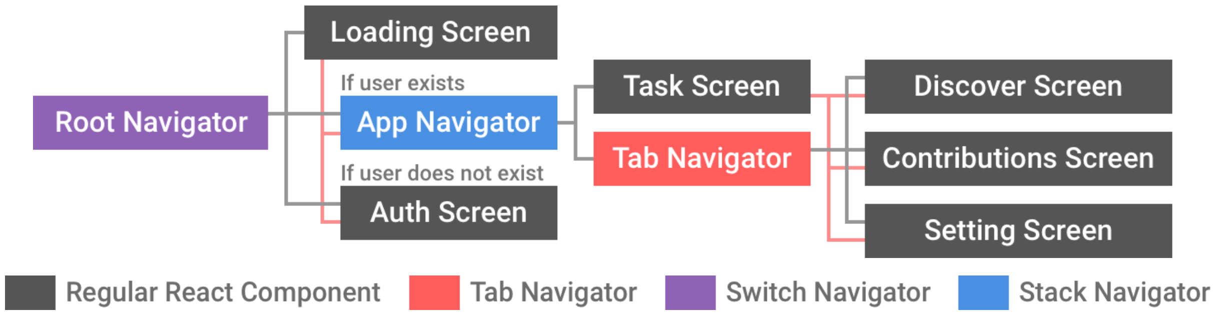


Figure 2. React Navigation

Bottom Tab Navigator

The tab navigator is associated with a bottom navigation bar. The bar lets users switch between major app functionalities. It appears on the bottom of the screen so users can easily reach it with their fingers.

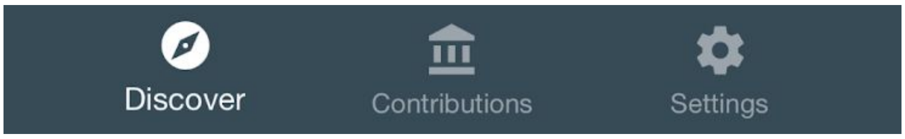


Figure 3. Bottom navigation

[1] https://github.com/MarShin/Brainpick/tree/development
[2] For more details please refer to code in config/routes.js, where SwitchNavigator, StackNavigator and TabNavigator are the APIs used to stack screen components in screens/.

The tab navigator manages the transition among following three screens.

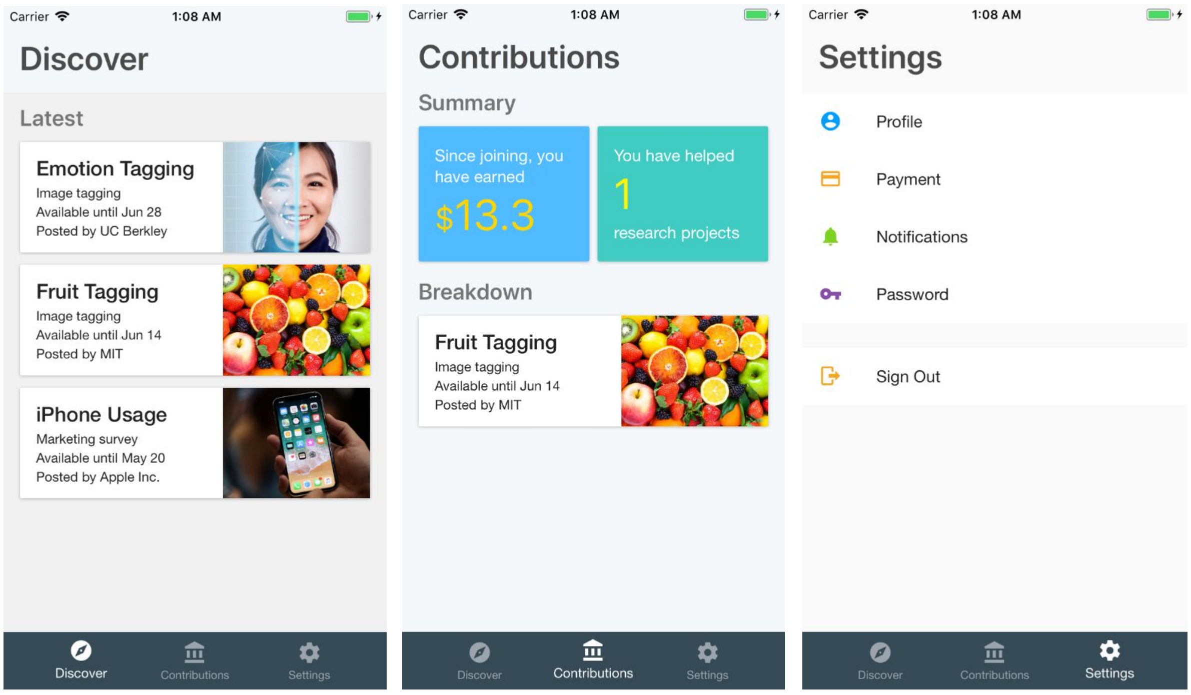


Figure 4. Home screen

Authentication Screen

Firebase Authentication is used to handle user management system related logic, which abstract away the complexity of building a secure OAuth system shipped with token management, generation, and expiration handling. At the same time, it can connect with third-party OAuth provider, for example, Facebook, Google, and Github.

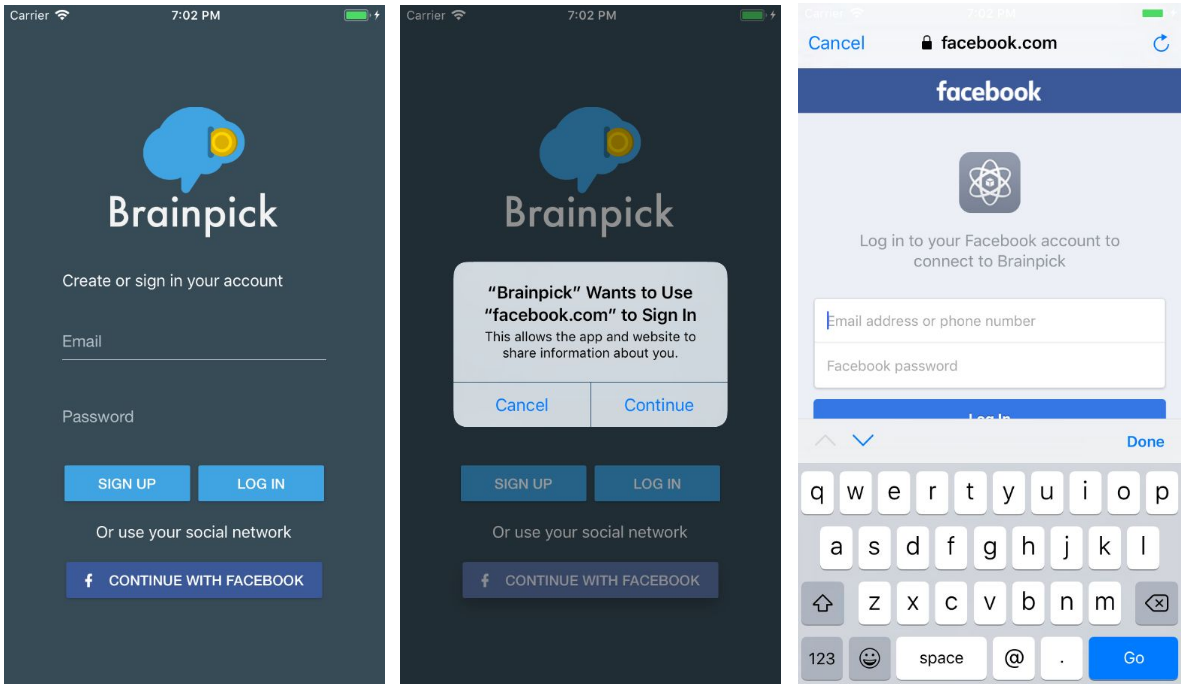
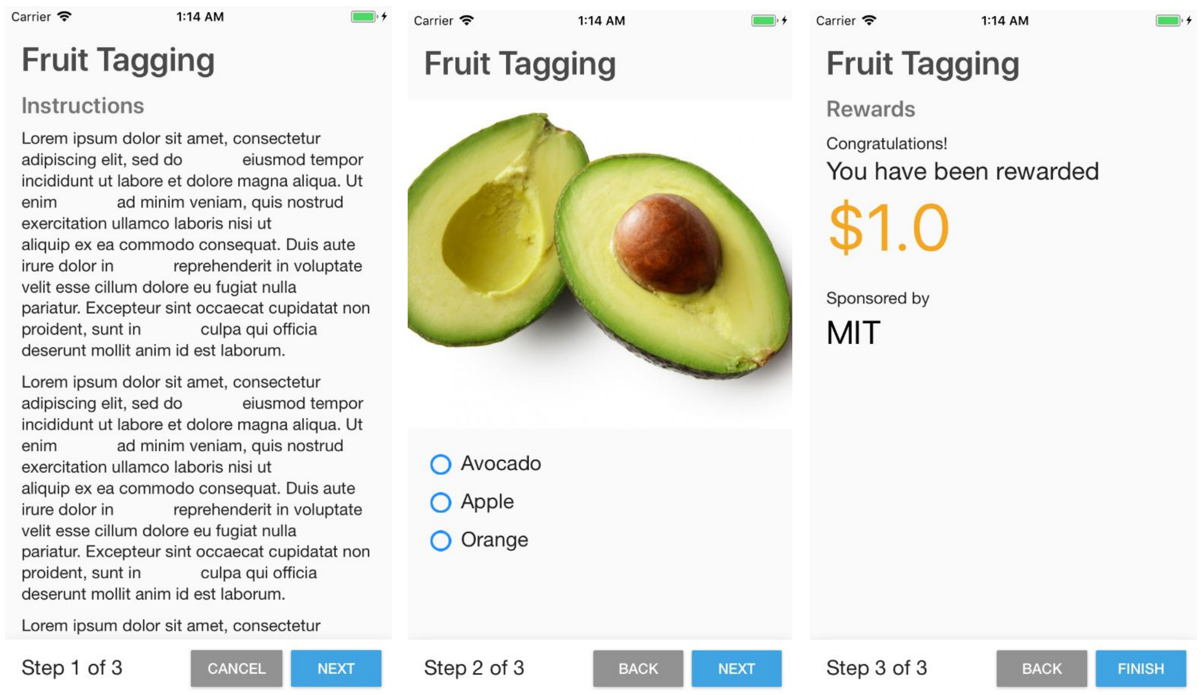

Brainpick
Create or sign in your account
Email
Password
SIGN UP
LOG IN
Or use your social network
CONTINUE WITH FACEBOOK
"Brainpick" Wants to Use "facebook.com" to Sign In
This allows the app and website to share information about you.
Cancel
Continue
facebook.com
facebook
Log in to your Facebook account to connect to Brainpick
Email address or phone number
Facebook password
Done

Figure 5. Authentication flow

Task Screen

The task screen has a step-based navigation. All types of tasks share the same layout, initial step (instruction) and final step (reward), but the intermediate steps are with different implementations. For example, image tagging tasks show an image for each subtask, while marketing survey show a MC question for each subtask. Such design makes future expansion of task type and change in task format very easy.



Carrier 1:14 AM

Fruit Tagging

Instructions

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur

Step 1 of 3 CANCEL NEXT

Carrier 1:14 AM

Fruit Tagging

○ Avocado
○ Apple
○ Orange

Step 2 of 3 BACK NEXT

Carrier 1:14 AM

Fruit Tagging

Rewards

Congratulations!

You have been rewarded

$1.0

Sponsored by

MIT

Step 3 of 3 BACK FINISH

Figure 6. Task navigation

Store, Reducers and Actions Dispatchers

The data in our app is managed by Redux with a couple of third-party libraries. The store centralizes data storage to achieve a single source of truth. Each top-level key of the store is associated with a reducer that transforms current states into new ones given a specific action. Each action is a data update request that describes the mutation of data and is dispatched to reducers after its creation.

```
{
  "currentTask": { "id": "aK1f23", "step": 0, "response": [ ... ] },
  "firebase": {
    "auth": { /* Authentication details */, ... },
    "profile": { "balance": 1.0, "history": [ /* Task Ids */ ] },
  },
  "firestore": {
    "data": { "tasks": { "aK1f23": /* Task Object */, ... } },
    ...
  }
}
```

Figure 7. Store structure

The data structure of task objects is as follows.

```
{
  "id": "aK1f23",
  "name": "Fruit Tagging",
  "type": "IMAGE_TAGGING",
  "rewardPerItem": 0.7,
  "dueDate": "25-05-2018",
  "instructions": [ "paragraph 1", "paragraph 2" ],
  "issuer": "MIT",
  "preview": "https://img.com/abc",
  "payload": {
    "data": [ /* Subtask Page Information */ ],
    ...
  }
}
```

Figure 8. Task structure

Firebase Integration

We use firebase as our backend instead of building a dedicated web server and database, which simplifies the system architecture designs a lot, thus allowing us to focus more on the mobile app development. The library React-native-firebase is used to integrate Firebase in our React Native project. The purpose of using this library is that although firebase web SDK work with react native, it is mainly built for the web. But having this module provides a JS bridge to native firebase sdk (iOS and Android), enabling Firebase to run on native thread while the rest of the application runs on the JS thread, which leads to boost in performance and avoid issues like lagging or frame drops in animations in the app. Two major Firebase services are incorporated in this project, namely Authentication and Firestore, which will be describe in more details other sections of the report.

The firebase framework for React Native uses a NoSQL-style data update and fetching mechanism. It does not share the same API with Redux. Even worse, the data is not merged into the Redux store, introducing a separate source of truth and defeating the purpose of using Redux. To solve this problem, we used the library React Redux Firebase and its complementary library Redux Firestore to synchronize the remote data into Redux store and dispatch Redux action.

Alert

A simple alert module is written using react-native-dropdownalert which is a simple way to provide in app notification by specifying the notification type (e.g. error, warning, info etc.) and notification message.

UI Libraries and Styling

A couple of libraries are used for UI engineering and styling.

React Native Paper

This is an external package for rendering common and generic components in a style following the Material Design. This is a handy module for quick coding, but for more advanced styling, custom made components coding in plain react native is recommended.

EStyleSheet

This module functions much like scss for css styling in web development, which allows media query, variables to share values such as color code, width size etc.

Code Style, Linting and Formatting

To ensure the consistency of our codes, we used linter and autoformatter to maintain a single coding style.

Style guide

The style guide that we are following is the Airbnb JavaScript Style, which is one of the most popular, readable and error-proof JavaScript style guide nowadays. This style guide also receives tremendous support from the open-source community across different text editors, linters and formatter.

ESLint

ESLint is the defacto linter of JavaScript. It greatly reduces syntax and runtime errors by enforcing a safe coding standard.

Prettier

Prettier is the defacto formatter of various programming languages in the recent years. After installing the plugin in text editors, all files are auto-formatted when saved. Every contributor to the project now shares the same code format.

Testing and Evaluation

We have conducted a series of test and

Unit Test

Ideally, we wish to use Test-driven Development throughout the project. Unfortunately, due to the need for rapid prototyping and time limitation, we only wrote a test case to make sure the App is able to render in react without crashing.

Integration Test

The integration test of our app focuses on the communication with Firebase backend. The firebase backend connection consist of mainly 2 services: Authentication and Firestore. Since firebase offer all of their services in a modular approach, we have to activate each component separately after setting up the packages and libraries and using correct API key. For Authentication, we use 1) email and password and 2) Sign in with Facebook, so additional configurations are needed to connect with 3rd party OAuth provider. Once the setup is done, we make sure that only accounts providing the correct information can have access to our main app, and the log in history of users are all recorded in Firebase Authentication.

For Firestore, data are stored in the 'Tasks' collection with a few dummy 'Task' document within the collection. Once the dummy data have been setup, we first grant read/write access to the configured Brainpick app, then verify by fetching the data with a correct path structure and submit user answers to make sure both read and write data flow are coherent on the firebase console and our app.

User Acceptance Test

User acceptance test was conducted to determine if the whole application meet the requirements from users' perspective. Criteria such as mobile app power consumption, response time, ease of use, invasiveness towards daily life were assessed. To conduct the test, we invited our friends to use our app on our testing devices instead of the simulator to obtain a more realistic experience and feedback. They provided valuable feedback in the process.

Conclusions

In conclusion, we have built a working front-end mobile application that demonstrates the potential of an intelligence crowdsourcing platform. Although the app is mainly written in JavaScript, it already has descent performance and feels exactly like a truly native app. Also, our app includes two types of tasks for users to complete and has a basic account management system with Facebook authentication. The app is able to communicate with the backend Firebase server smoothly under normal Internet connection.

Shortcomings

Currently, we only support two types of tasks - marketing survey and image tagging. It is because other types of tasks require more advance access to the native API. Due to React Native's nature of directly executing JavaScript at runtime, setting up native modules and connecting them into JavaScript codes require considerable effort. This significantly hinders the development of our app.

Also, the tasks are currently manually created in the backend for demonstration purpose. For the entire intelligence crowdsourcing platform to be a complete product, we need to build a web interface for researchers and companies to upload tasks to the app.

Takeaway

In the process of this project, we learnt a lot about the trade-off between developing a JavaScript powered native app (React Native/Flutter) and its truly native alternative (Java/Kotlin/Swift). The experience that we gained from this project, together with the knowledge we learned in class, has given us a satisfying answer of this trade-off question.

In the future, we wish to further explore the idea of intelligence crowdsourcing platform and continue the development of the app to make it a real product.We hope to extend the app with more functionalities such as adding more types of tasks thus able to collect new types of data, provide an interface for task issuer to allow researchers/ companies issue new tasks, connect with the payment gateway for reward redemption.

We believe our app has great potential in helping the machine learning community to explore new class of data therefore train smarter models as we are addressing a pain point which not many have thought of.

References

1. Airbnb React Style Guide
   https://github.com/airbnb/javascript/tree/master/react
2. Project Organization Guide
   https://medium.com/the-react-native-log/organizing-a-react-native-project-9514dfadaa0
3. 11 mistakes when writing RN and Redux Project 11 mistakes making RN Redux app
   https://medium.com/dailyjs/11-mistakes-ive-made-during-react-native-redux-app-development-8544e2be9a9
4. React Native camera app with live preview saturation and brightness filters
   https://medium.com/react-native-development/react-native-camera-app-with-live-preview-saturation-and-brightness-filters-d34535cc6d14
5. React-native-camera
   https://github.com/react-native-community/react-native-camera
6. Navigation prop reference
   https://reactnavigation.org/docs/navigation-prop.html
7. react-native-video
   https://github.com/react-native-community/react-native-video
8. react-native-audio
   https://github.com/jsierles/react-native-audio