NUSTown

Milestone III

[NUSTown.apk]

[https://github.com/thatseant/NUSTown]

Aim

Something New Everyday

Learn something new
Various CCAs hold open classes to, for example, teach students the basics of rock-climbing or Contemporary Dance or Basic Life-Saving. NUS Hackers organise workshops and talks to expose students to new technologies and applications. Our app parses data from various sources and presents them in a consistent and integrated manner.

Improve Existing Events
While our app already has existing data, CCA Heads can add additional updates and communicate directly with interested participants. Besides adding photos and videos, CCA heads can survey and request RSVP from interested participants. Participants can chat with each other to build rapport and share details.

Organise Something New
In NUS, there is a huge list of martial arts or performing arts CCAs but not ones revolving around hobbies or recreational sports.

In our app, pick-up games can be easily organised as students can indicate the numbers they need. You can choose to organise meet-ups, teach new hobbyists or organise field trips. We hope that through such ad-hoc events, eventually more diverse interest groups can take off.

User Stories

Events and Clubs

Collating Events

1. As a student bombarded by emails, Telegram and Instagram pages, I want an app that gathers all NUS activities so I do not have to check so many platforms.
2. As a freshman, I would like to know past activities so I can get a feel of what campus life will be like after COVID-19.
3. As an interested participant, I would like to chat about the event with fellow participants to get to know them better and exchange information.

Event Updates

1. As an event organiser/CCA head, I want to update existing info for my event so that interested participants are kept up to date.
2. As an event organiser, I want to know who is interested and confirm the number attending to aid planning.
3. As an event organiser, I want to survey participants on their preferences so that my event interests them more.

Collating Clubs

1. As a freshie, I want to know more about all the interest groups and CCAs NUS has to offer so that I can be more involved in campus life.

Jios and Groups

Organising New Activities

1. As a participant, I want to easily chat with the event organiser and other participants so we can all better prepare for the event and build rapport.
2. As someone looking to play Futsal, I want to start a casual event where people can join me and my friends as we do not have the numbers.
3. As someone interested in improv comedy, I want to host a casual meetup/interest group to find a community that shares this interest.

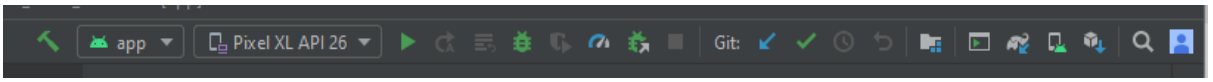
Setup

Android App or 3rd Party Emulators:

1. Download APK: NUSTown.apk
2. Install the app.
    a. Follow your device instructions.
    b. You might have to allow “Install unknown apps” in settings.
3. On the login page, type any email that ends in .com. You can create multiple test accounts.
    a. If you don’t see a login page, log out first from the menu bar.

Build in Android Studio:

1. Download Android Studio
2. Clone from: https://github.com/thatseant/NUSTown.git
    a. Zip Version: https://github.com/thatseant/NUSTown/archive/master.zip
3. In Android Studio’s start page, click “open an existing project” and select NUSTown/Android folder.
4. Setup and run Android Emulator
    a. Run Android Virtual Device (green play button) on the top right hand. Or create an AVD device if there isn’t one.



    b. Recommended Virtual Device: Android 10.
    c. Minimum: Android Oreo.

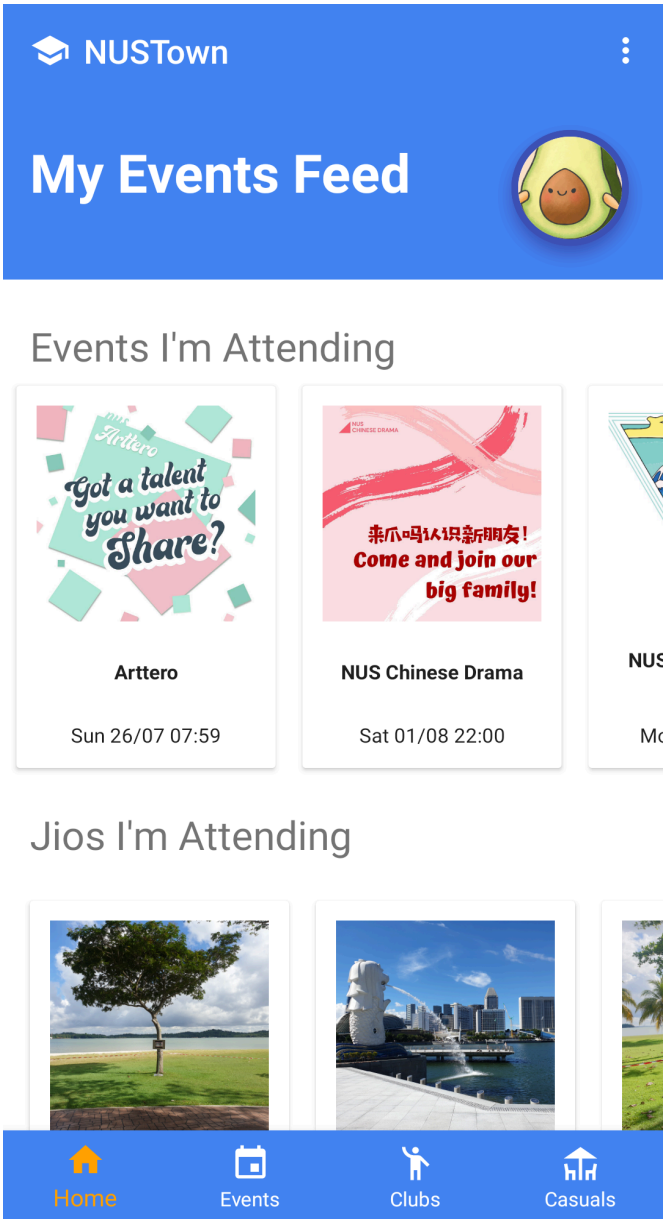
For evaluation purposes, you can try out the core features below.

Core Features

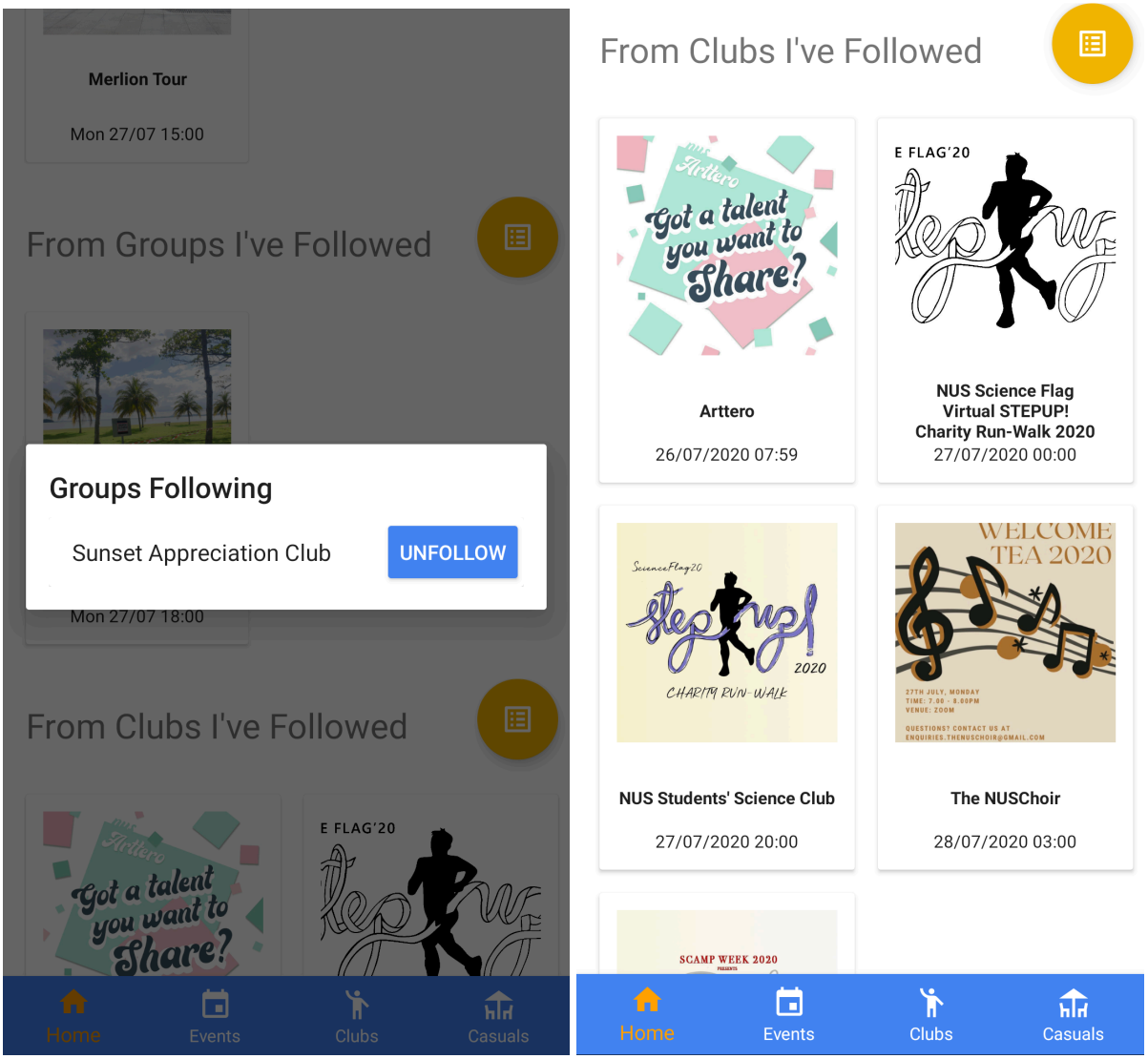
Login

- The user keys in his email, name, and password during sign up. The app directs the user to login if his email is an existing one.

#1: Homepage

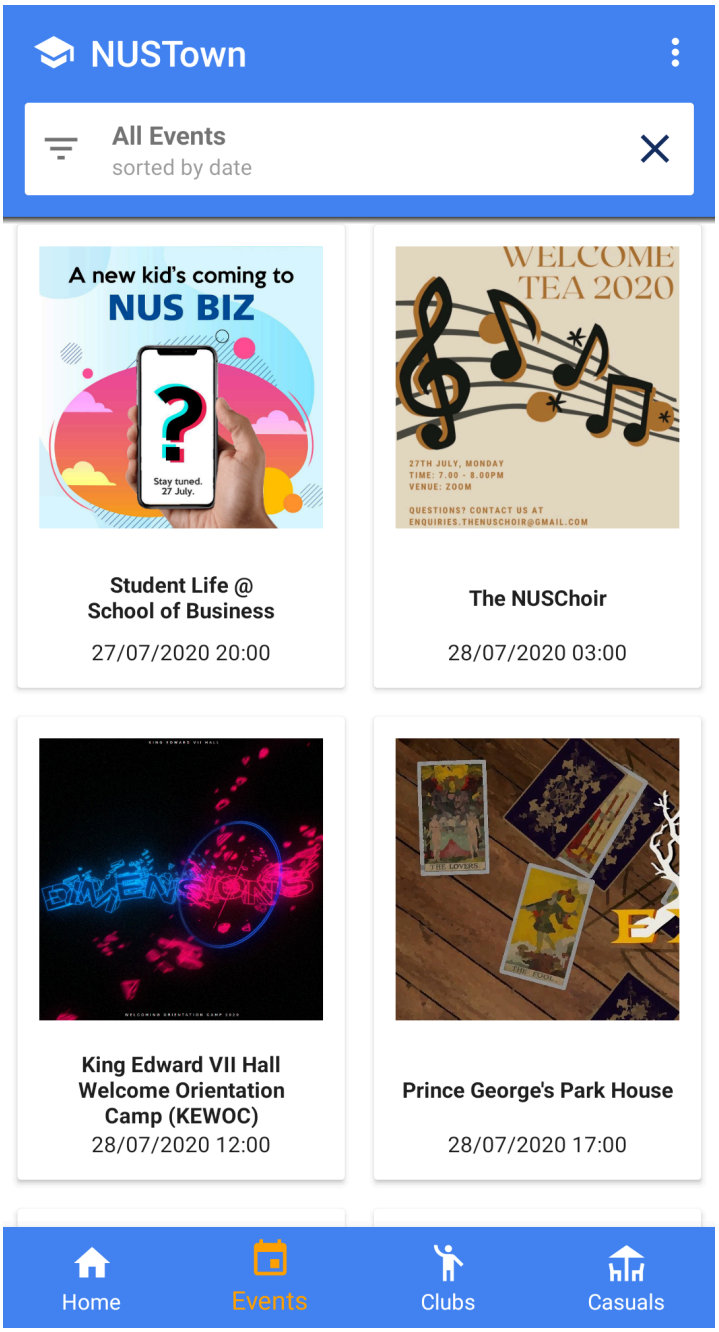


- Events/Jios Attending:
    - Once logged in, users will see the future events they're attending, both from official sources and NUSTown users (known as "casual jios").

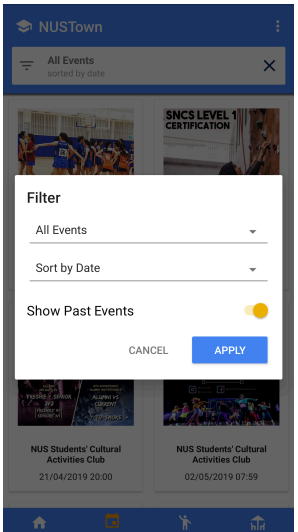


- From Clubs/Groups I've followed
    - Below the events they attend, they can see all future events from **official clubs** and **casual groups** created by NUSTown users they have followed within the app.
    - Clicking on the yellow club button on the right shows all the clubs/groups the user currently follows.
    - The user can unfollow them by clicking on the blue button on the right.
- All events are sorted in chronological order.

#2: Events List

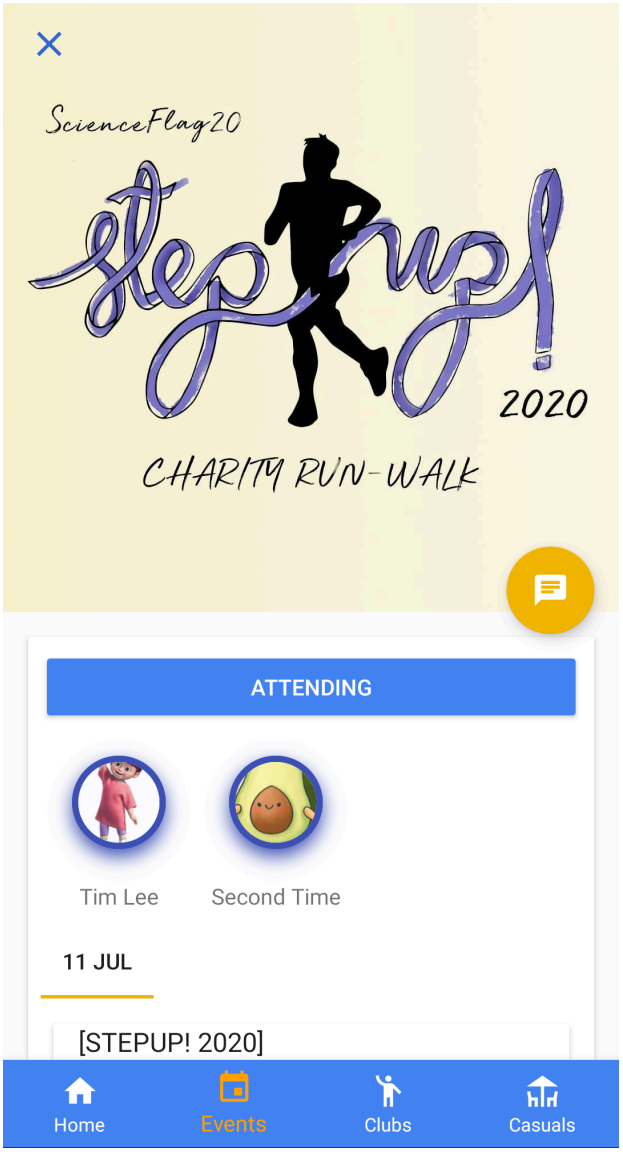


- The second tab in the bottom nav bar displays all events we have gathered from our sources.
- Events are extracted from NUSync, Telegram and the Instagram accounts of NUS' 215 official clubs.
    - **Our algorithm parses the Instagram captions and Telegram messages from these accounts and only sends event posts to our database.**

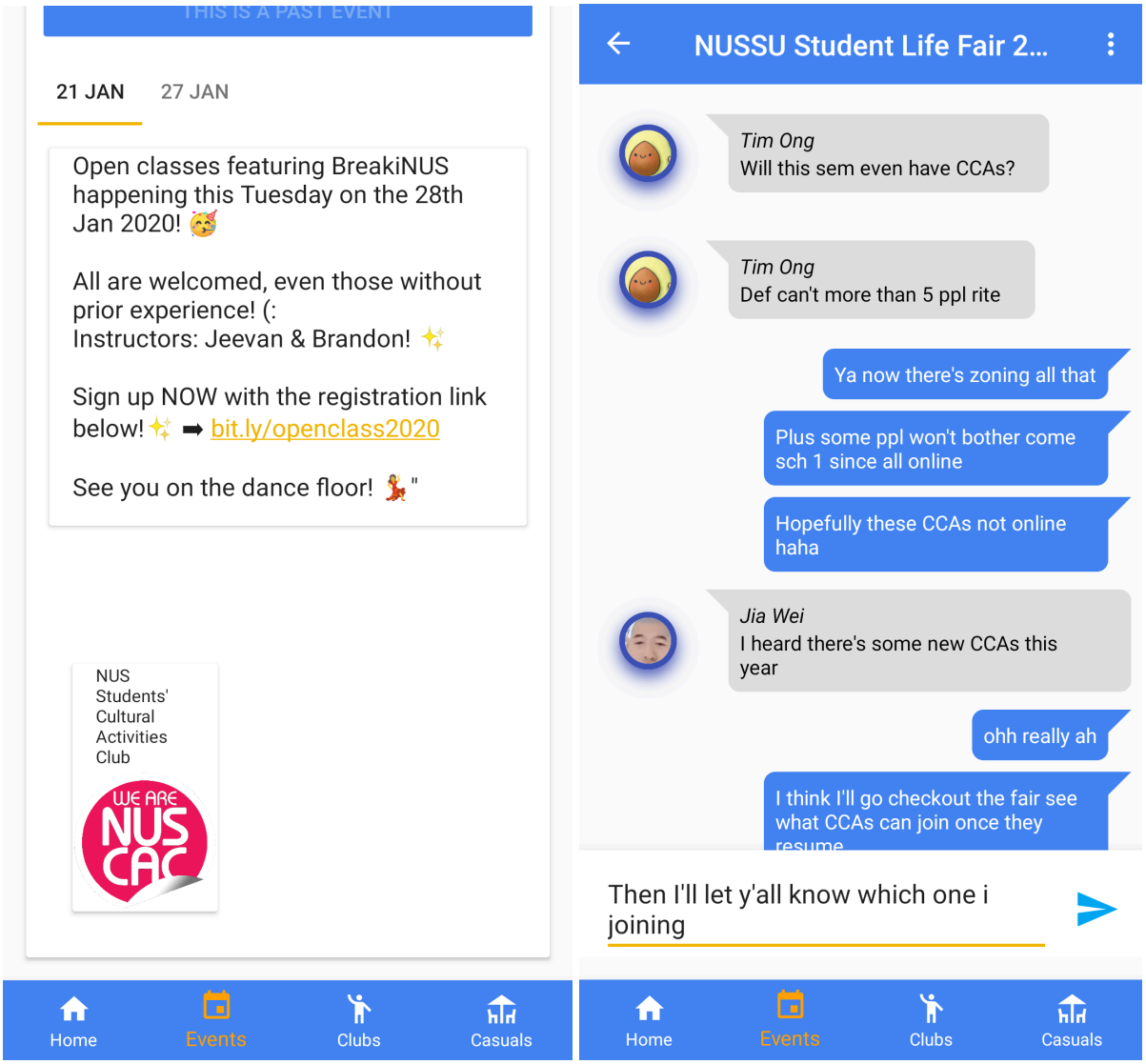


    - **Date and Time is extracted from these captions.**
- Filter Bar
    - The filter bar at the top allows users to filter events by categories and sort them chronologically, either by event date or by date of latest post.
    - Users can toggle display past events to get a feel of NUS events.

#2: Individual Event View

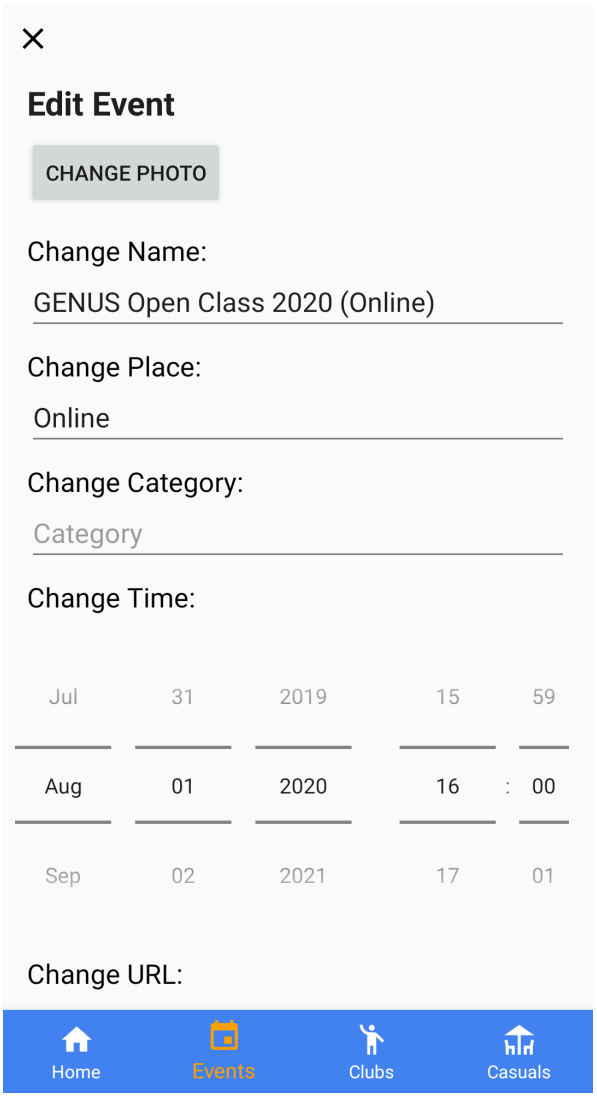


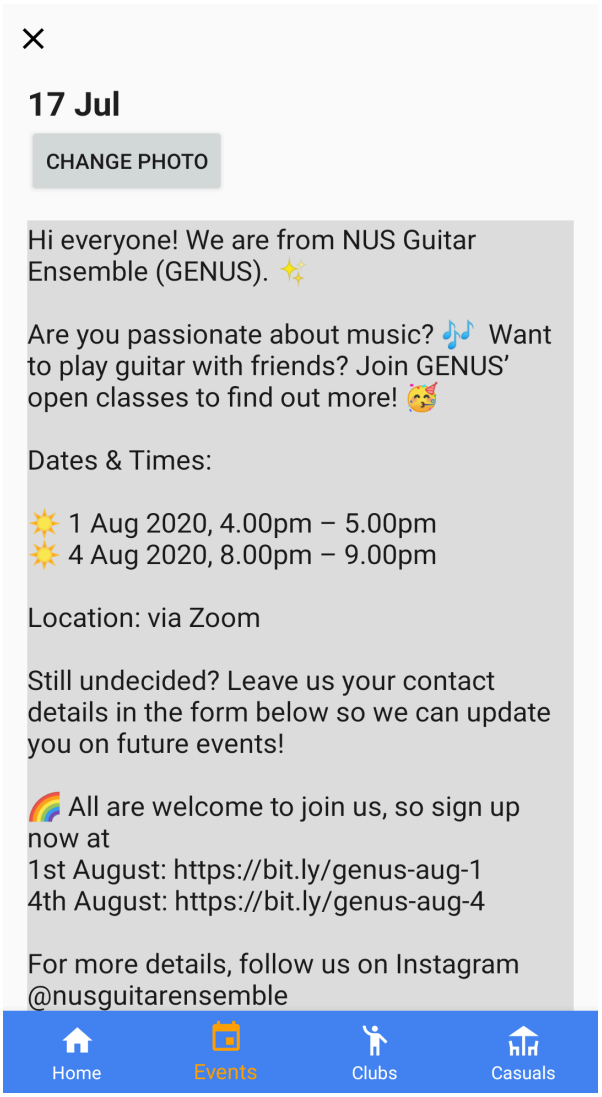
- Clicking on an individual event opens a view that shows more information about the event.
- **Duplicates Posts from our sources on the same event are recognised by our algorithm and consolidated together.**
    - Users can swipe or click on the tabs to see the latest updates.
    - Users can click on url links in the posts.



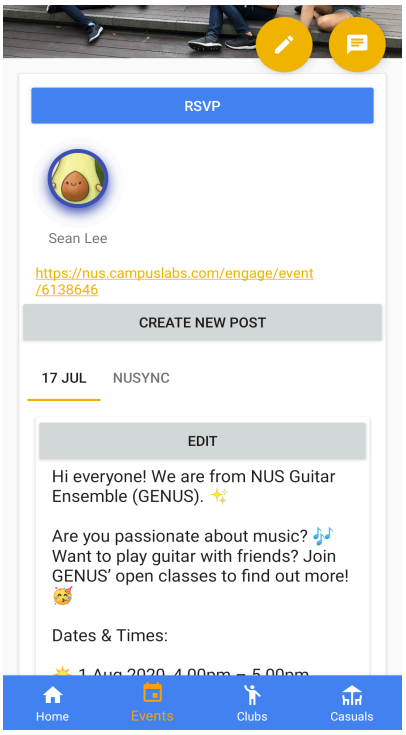
- Users can RSVP to publicly tell other members that they would be attending.
    - They can also see other members that are attending.
    - Users can also chat with other users including organisers about the event.
- The organising club of the event is shown at the bottom. Users can click on it to go to the club page.

#2: Edit Event View

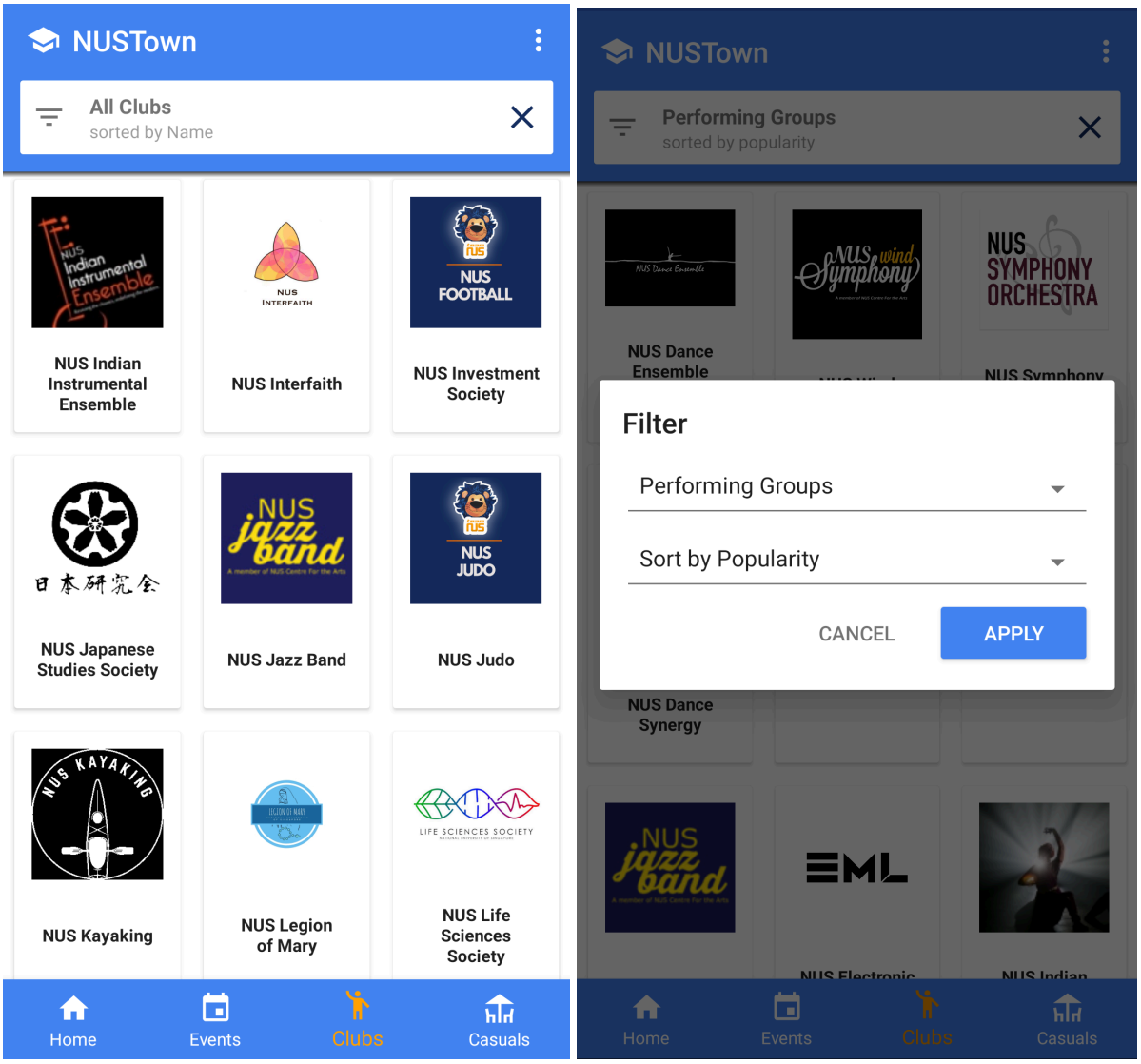




- Event organisers and users that have approved can edit the event on the app directly.
    - They can also edit and add updates to the event including photos.

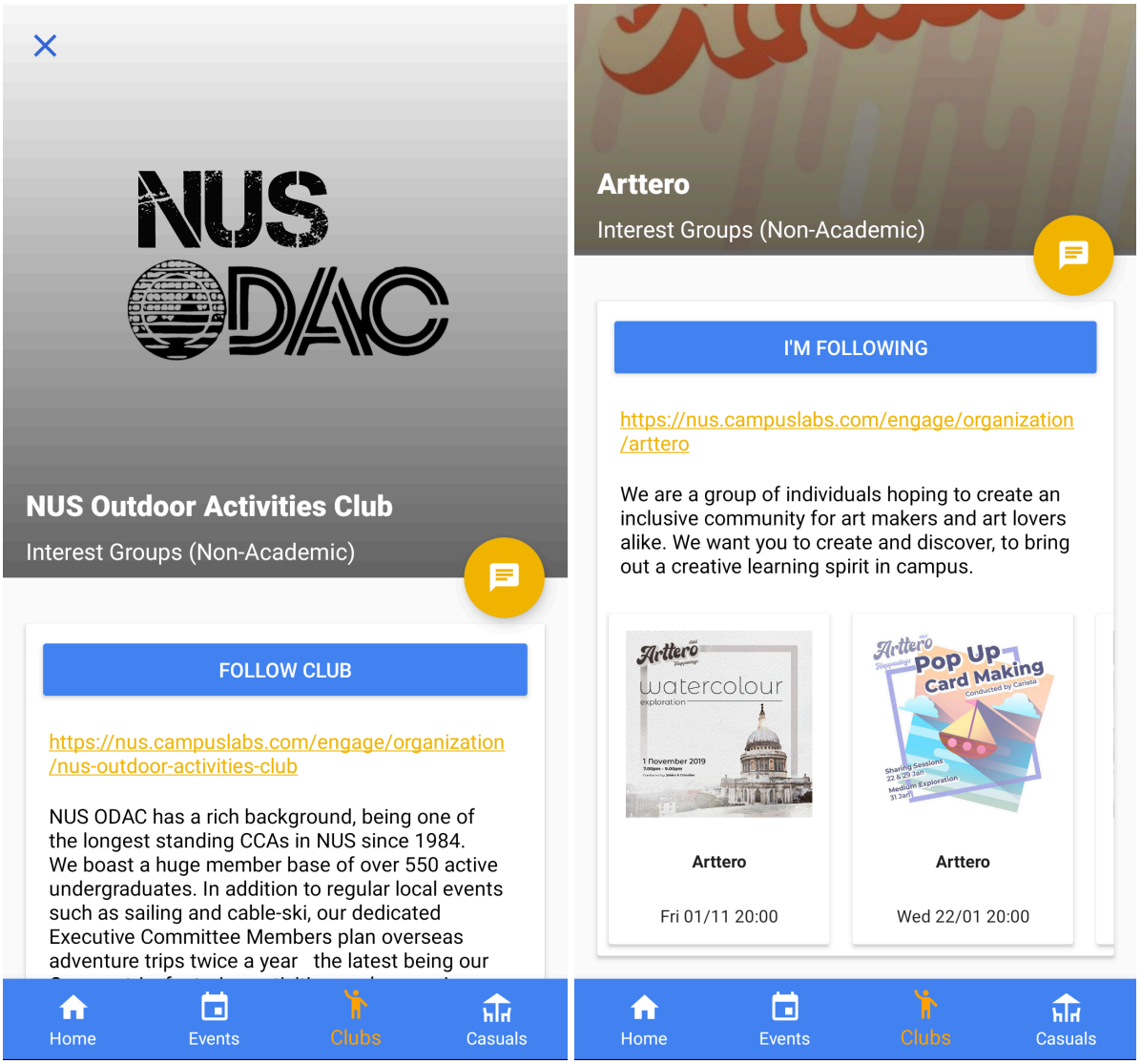


#3: Clubs View



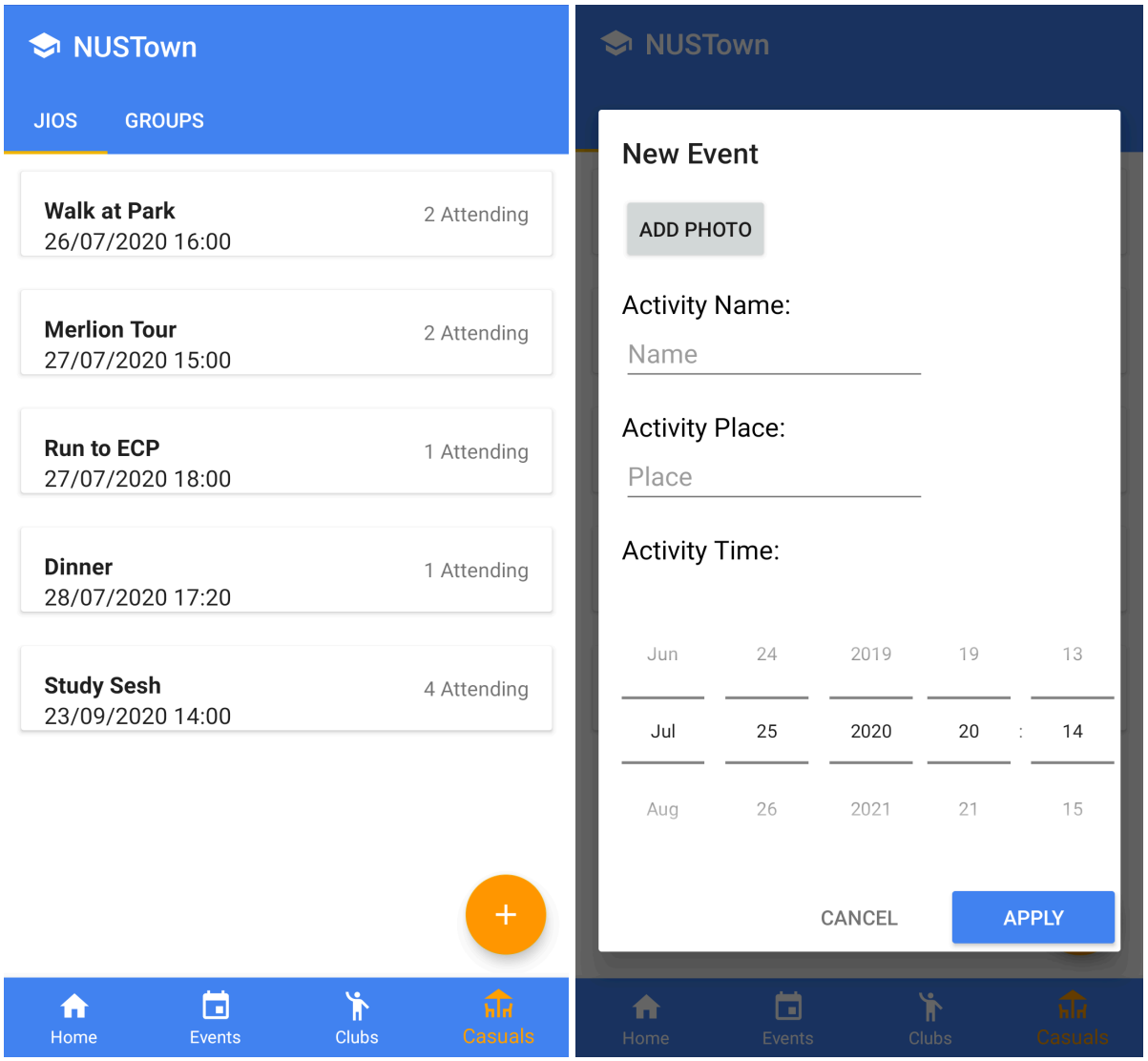
- All 215 clubs are displayed in alphabetical order.
- The clubs' info are parsed from NUSync.
- Filter Bar
    - The filter bar at the top allows users to filter clubs by categories and sort them either by name or by popularity which is determined by the number of Instagram followers they have.

#3: Individual Club View



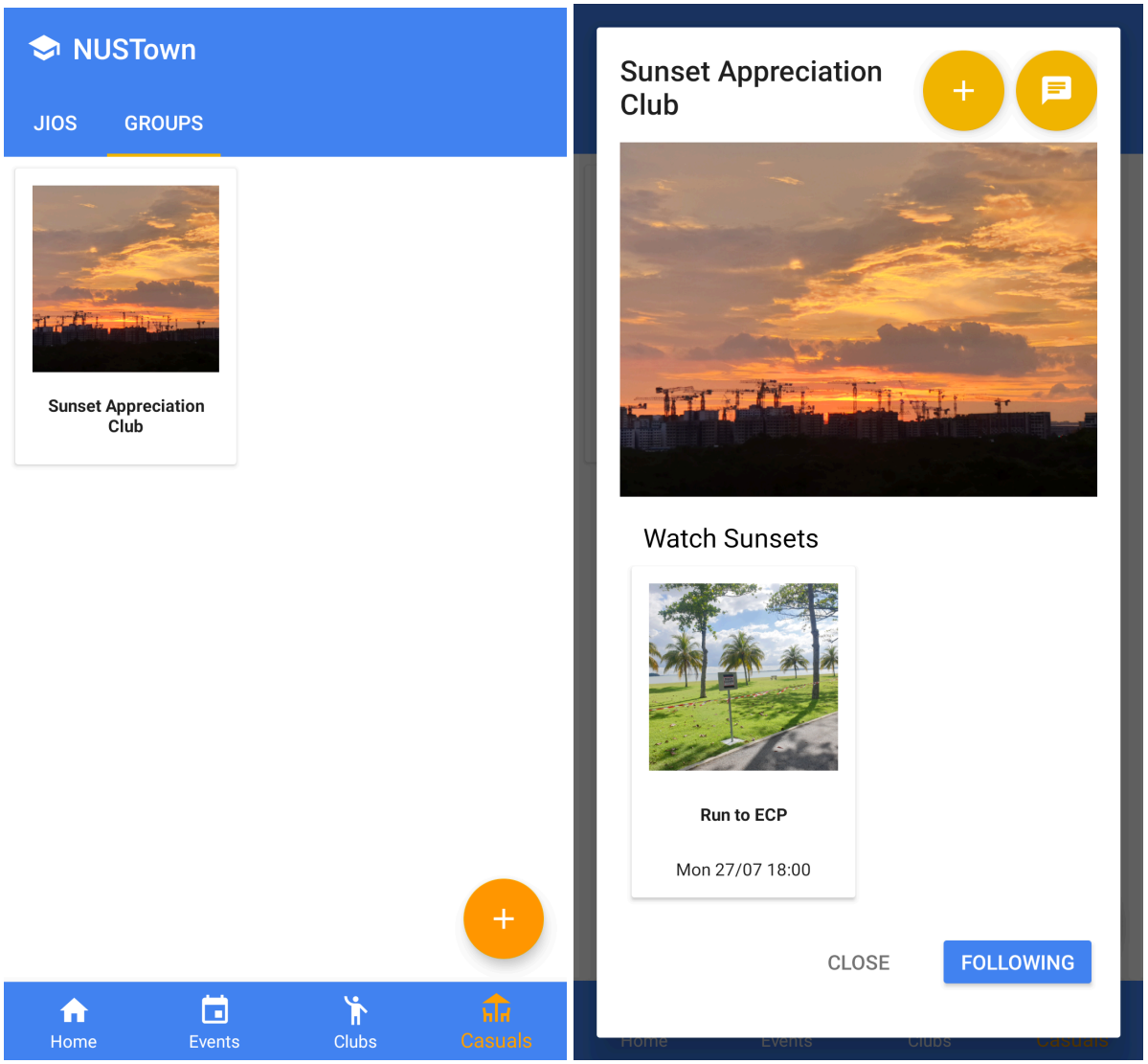
- Besides information about the club, users can see a swipeable list of events the club has organised. Clicking on them brings up the event view.
- Users can also follow the club and have upcoming events show up on their homepage.
- Interested viewers can also chat with each other by clicking on the yellow chat button.

#4: Casual Jios View



- Displays all gatherings and activities created by users in-app.
- Clicking on a “Jio” brings up a Dialog with info about the event and shows other users who are attending.
- Add Jios by clicking on Floating Action Button
- Jios can be edited by organisers.
- Interested attendees can RSVP, see more info and chat with each other by clicking on the Jio.

#4: Casual Groups View



- Groups are informal clubs started by NUSTown users.
- Jios can be added to groups by clicking on the yellow “add” icon.
- Just like clubs, users can see the groups’ jios and chat with each other by clicking on the yellow “chat” icon.

Software Development

Tech Stack

Frontend Application Development : Android Studio (Java and Kotlin)

Database : Firebase Firestore and Firebase Storage (Images)

Cloud Functions : Firebase Functions (Node.js), Google Cloud Functions (Python) and Google Cloud Scheduler

Data Crawling: Instagram GraphQL Endpoints, NUSsync API, Telegram Bot API (Javascript and Python), Axios HTTP Library

Date Extraction: Chronos Node.js library (with enhancements)

Authentication : Firebase Authentication

Version Control : Git and Github

Frontend Overview

NUSTown's frontend is developed entirely on Android Studio using Java and Kotlin. Android Studio integrates well with Firebase which we used extensively for our backend. Our application fetches data from the database, Firebase Firestore, and images from Firebase Storage to display to the users.

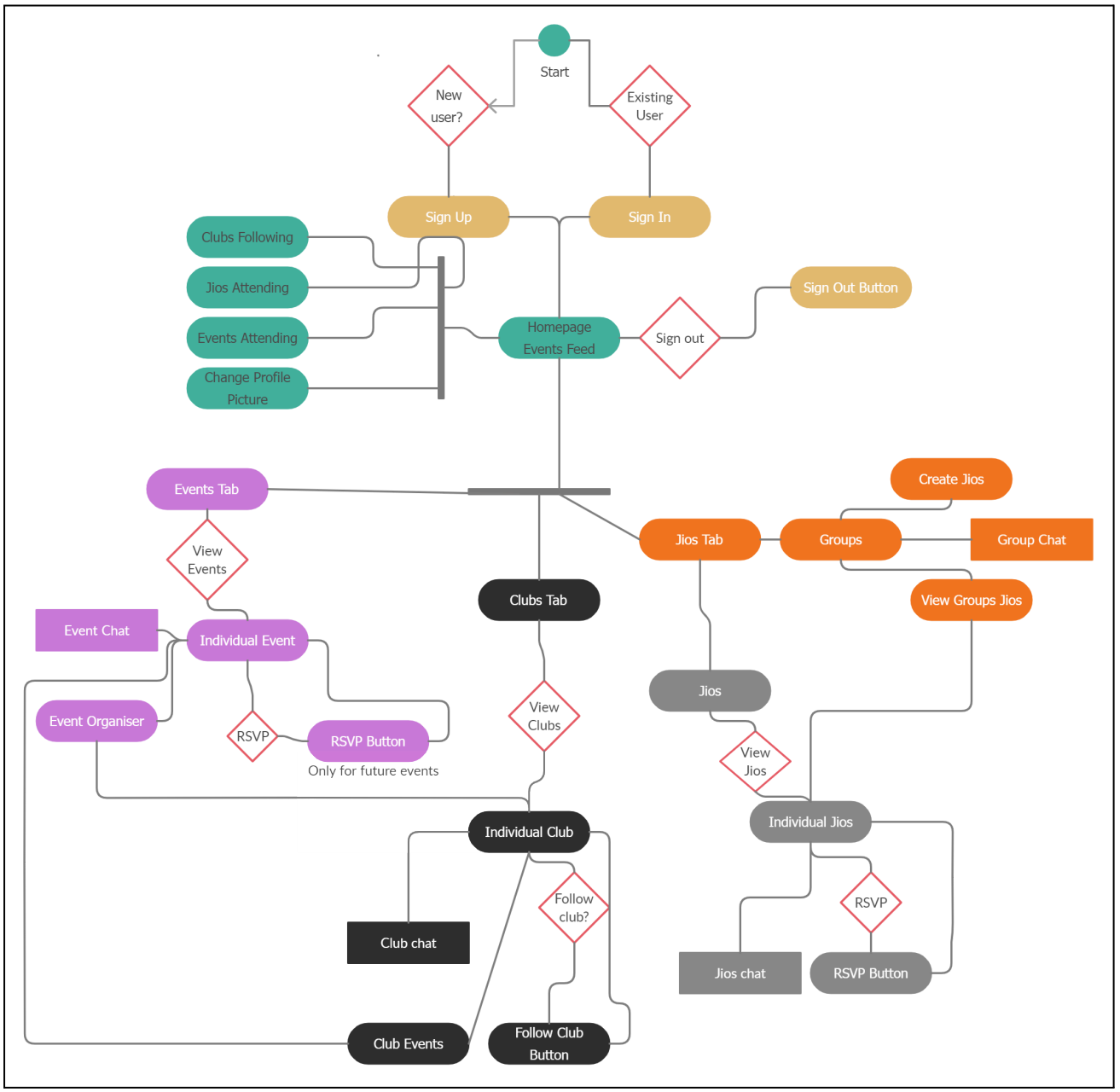
Backend Overview

As for NUSTown's backend development, a variety of APIs and SDKs are involved. Firebase authentication service is used for signing in and signing out. To populate our database, we implemented crawlers via Cloud Functions that parses events and clubs data from Instagram, NUSync and Telegram in a Node.js environment. (See Documentation section) In addition, some of the clickables in NUSTown, including our RSVP button, call Cloud Functions that interact with our database directly.

Software Engineering (SWE)

Activity Diagram

The diagram below is our application activity diagram that shows a possible path that NUSTown users will go through. Each of the core features are colour coded differently. NUSTown allows the user to go from one activity to another without having to return to the homepage. More details of each core feature are mentioned under “Core Features”.



Project Management

We decided to adopt Scrum, one of the Agile methodologies so that we could remain flexible as the requirements and expectations of the project could change as we move to later stages of development. This was important as we were new to software development.

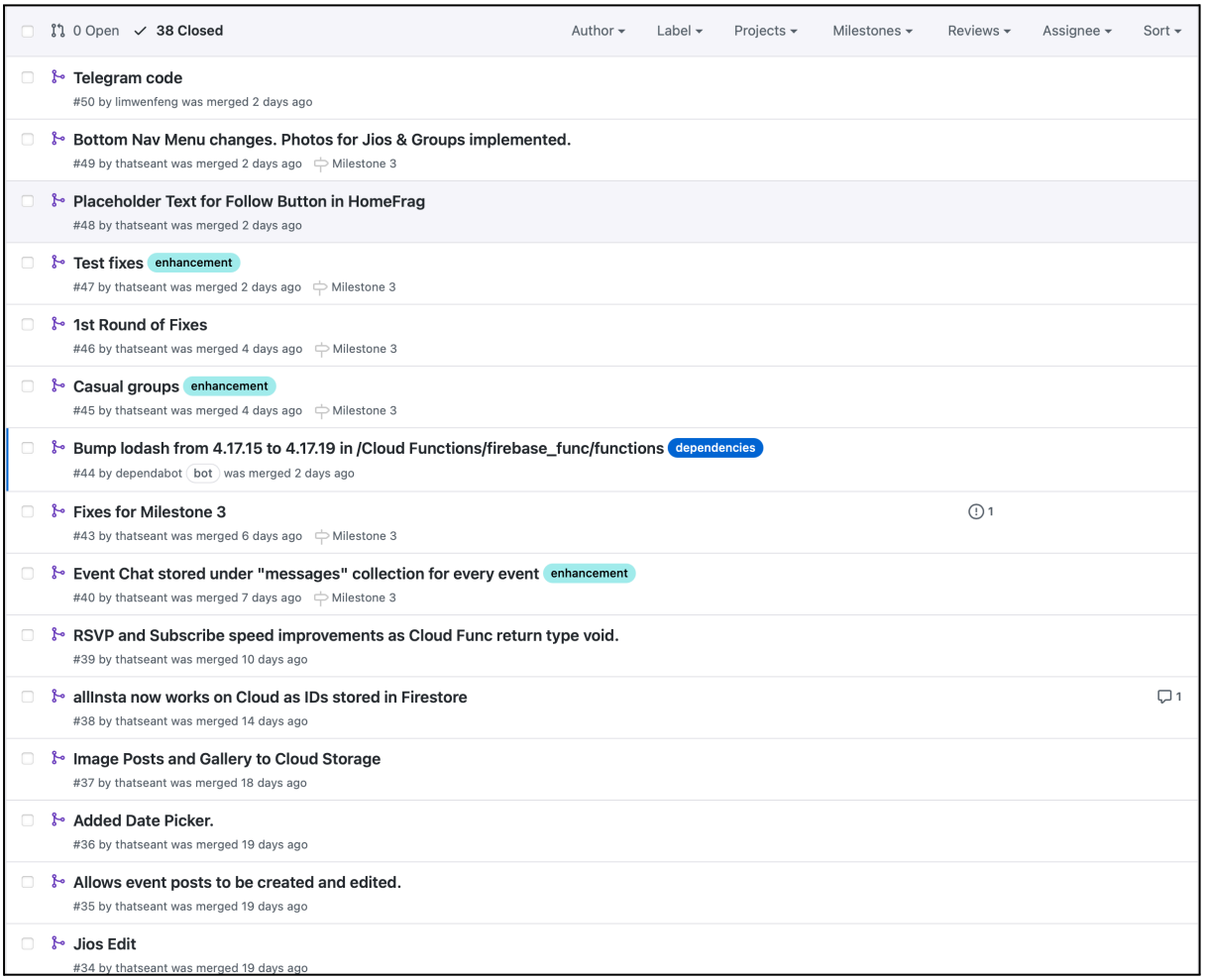
Each of our features were broken down into sprints and a deadline would be assigned. The team would then work on these features individually. At the end of each sprint or when the sprint's deadline is met, we would have a team meeting to present the completed features or the obstacles faced if the feature is not completed within the sprint. This, together with Git Pull Requests workflow, ensures transparency which allows for the project to be followed by both members. At the end of the meeting, we would also decide the deadline and application features for the next sprint and the cycle would repeat sprint after sprint.

As the agile approach consists of regulating releasing updates to consumers and receiving feedback, it was the same for us as we regularly update our advisor and mentor while their feedback is immediately considered during the planning of the following sprint.

Using Scrum approach allows us to maintain the quality of our work and ensures that we prioritize the sprints correctly, as after all, we have to achieve a Minimum Viable Product first.

On the other hand, this approach does not define our team member role as developer clearly. We do not have a distinct role as back-end developer and front-end developer however, this ensures that the team members get equal exposure to different aspects of application development and the team is cross-functional.

Pull Requests



From Milestone 2 onwards, we have adopted the Git Pull Request workflow where we checked out feature branches to work on new features. Once finished, we submitted pull requests, discussed with each other and finally merged the branch into master.

Testing

Feature Testing

As for application testing, we did feature testing with different test scenarios for each of our features. We came up with different test cases and ran these tests for the results. The test cases that passed are highlighted in green, while those that pass partially are highlighted in yellow. Uncompleted features and failed test cases are coloured in white and red respectively. We would be continuing to work on the yellow, red and whites test cases.

Test Feature	Test Case	Test Data	Expected Result	Pass/Fail
Login/Sign In				
Login/ Sign in	Keying in existing email	Existing email: gameshelpdesk.wilfred72@gmail.com	Keying in existing email would open “key in password” activity	Pass
		New email: newemail@gmail.com	Keying in new email would open “Sign up” activity	Pass
Home Tab				
Changing Profile Picture	Clicking profile picture and selecting photo	With Different Kind of Aspect Ratios	New Profile Picture displayed with appropriate cropping	Pass
Events Attending	Display of event thumbnails and event details	Test account must be attending some event	Event name formatted correctly	Pass
			Picture displayed correctly	Pass
			Date formatted correctly	Pass
			Only future events displayed	Pass
Casual Jios	Display of event thumbnails and event details	Test account must be attending some casual jios	Event name formatted correctly	Pass
			Date formatted correctly	Pass
			Only future events displayed	Pass
From Clubs you Follow	Clicking on Club Following list and clicking on unfollow	NA	Displays Followed Clubs	Pass
			Unfollowed Clubs no longer displayed	Pass
			Display all future events of clubs	Pass
Events Tab				
Events List	Display of event thumbnails and event details	NA	Only future events displayed as default	To be Done
			Event Date and Time Displayed correctly	Some Events Fail
			Events in Chronological Order	Pass
			Non-Events not displayed	Some Events Fail
Events Detail	Edit rights for organiser	With organiser email	Add, Edit and delete posts buttons and edit events button shown.	Pass
	Edit rights for non-organiser	With normal email	Add, Edit and delete posts buttons and edit events button not shown.	Pass
	Layout		Photos are displayed well.	Pass
			Correct organising club displayed	Club - Pass
			Description does not have html text	Pass

	Post Updates		Post Updates Picture displayed	Pass
	High traffic volume on events chat	To be done with multiple users on the same event. Pending.	Fetches documents by batches to reduce Firestore cost.	To be done
	RSVP Button	Different users click on RSVP Button	RSVP Button Text reflects attending status	Pass
			Username and Picture properly shown for RSVPed users.	Pass
			Event shown on homepage.	Pass
Edit Events				
Edit Posts	Changing Photos	With Different Kind of Aspect Ratios	Photo displayed correctly	Pass
Edit Posts	Changing Description	With different length of text	Full Length of Text Displayed URL Displayed	Pass
Clubs Tab				
Clubs List	Layout	To test with at least 30 clubs	All clubs have the right categories	Pass
Clubs Filter	Applying different Categories		Only clubs of selected category displayed	Pass
	Clear Filter		All Clubs displayed	Pass
Clubs Detail	Following or unfollowing clubs	To test with at least 30 clubs	Description does not have html text	Pass
			NUSync Link displayed	Pass
	Follow Button		Reflects followed status	Pass
			Club Events displayed on Home Tab	Pass
Casual Jios Tab				
Jios List	Number Attending	Jio has max number	Max number and number attending displayed	Pass
		Jio does not have max number	Only number attending displayed	Pass
	Layout		Only future jios displayed as default	Pass
Jios Dialog	RSVP function of participant		RSVP Button Text reflects attending status	Pass
			Username and Picture properly shown for RSVPed users.	Pass
			Event shown on homepage.	Pass
	Edit rights for organiser	Have to create an event first	Edit Button shown	Pass
	Edit rights for non-organiser	Test on an event created by someone else	Edit Button not shown	Pass
	High traffic volume on events chat			

Adding Jios	Input Fields	Empty Name/Place or Time	Android Toast displayed: "Empty Field"	To be done
		Time is in the past	Android Toast displayed: "Date is in the Past"	To be done
		Name, Place and Time inputted	Jio shown in refreshed list with correct details	Pass

Expert Testing

We consulted with our mentor, Samuel Lim, extensively on UI and UX design. He helped to test our app and gave very good suggestions on aiding user navigation and also consistency of UI elements. This instilled in us an end-user perspective throughout development for our app.

User Testing

In addition to feature testing, we also did user testing to evaluate our UI and UX. We were able to test the user-friendliness of our app by asking testers to try to fulfil certain tasks in the application. We did both in-person and survey testing.

For in-person testing, we started by handing them the phone with an app running and observed how they interacted with it. Following which, we interviewed them about the app's UI and UX. We then asked them to perform certain specific tasks and again observed them.

We also did survey testing in order to reach a wider pool of testers including CCA heads. We felt that letting them install the app on their devices and using it over a longer period of time would also allow for more real-world testing on a variety of devices. Two rounds of user testing was done: 8-12 July and 22-26 July. Below is how we did the survey testing.

Following is a list of instruction given out to the participants:

1. Registration
2. Navigate pages
    a. Follow clubs from specified category (e.g religion club)
    b. RSVP for one of the club's event
    c. Change profile picture
    d. Create one jio Informal jio event

After which, a set of questionnaires was given to them to collect their opinions and experiences with our application. The questions and results are attached in the Appendix A of the report.

After analyzing the results, we concluded that the application did fairly well in terms of functionalities and keeping things simple. Many of the users indicated that the system is not too complex while still able to achieve the intended purpose. The system was consistent throughout the application such that users are able to quickly pick up the navigation and functionalities. Lastly, the majority of the users also felt that the core features of the application are useful and they intend to use it regularly. Still, there were several areas where user-friendliness could be improved.

UI

The user testing result showed that the application UI is lacking. Therefore, we did an investigation to find out the details and following are the findings. Our improvements are also detailed below.

Features	Improvement needed
Homepage	New users will not have any events attending or clubs following on the homepage and this causes the homepage to be relatively empty with just plain text and leaves the users disoriented. We improved on the UI by including feedback messages when there aren't any events attending or clubs following.
Throughout the Application	Inconsistent spacing between the UI objects such as buttons, images and text boxes.
Event Page and Club Page	The images used for events and clubs have inconsistent resolution and sizes, this causes our application interface to be bombarded with awkward spaces. Improvement: Centre cropping of all images to ensure consistent size for user familiarity of event cards. Placeholder icons for cards without images.
Jios and Groups	Jios and Groups did not have images, making their card dull and visually stark when placed alongside event images on the home page. Improvement: Added images for jios and groups, making them as appealing as official events and clubs.
Bottom Nav Menu	Not obvious. Many users did not realise this was central to the app. Improvement: Highlighted selected tab and gave all tabs text below icon.
Spinner for RSVP button	Our RSVP button was not able to update the attendance immediately as it requires sometimes to execute cloud functions. While the function is loading, it misleads the user that the click wasn’t registered and the user would end up clicking on it multiple

	times. We added a spinner to inform the users that the function is being executed.

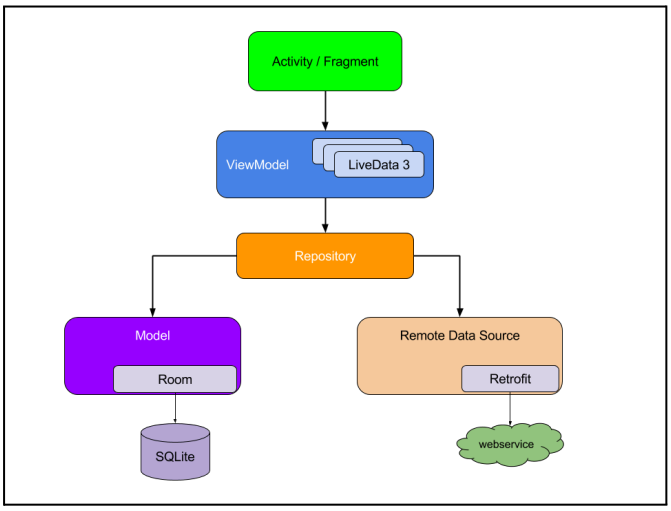
Documentation

The source code for our app is on Github repository (Link at the start of this document).

Frontend Development: Android Studio

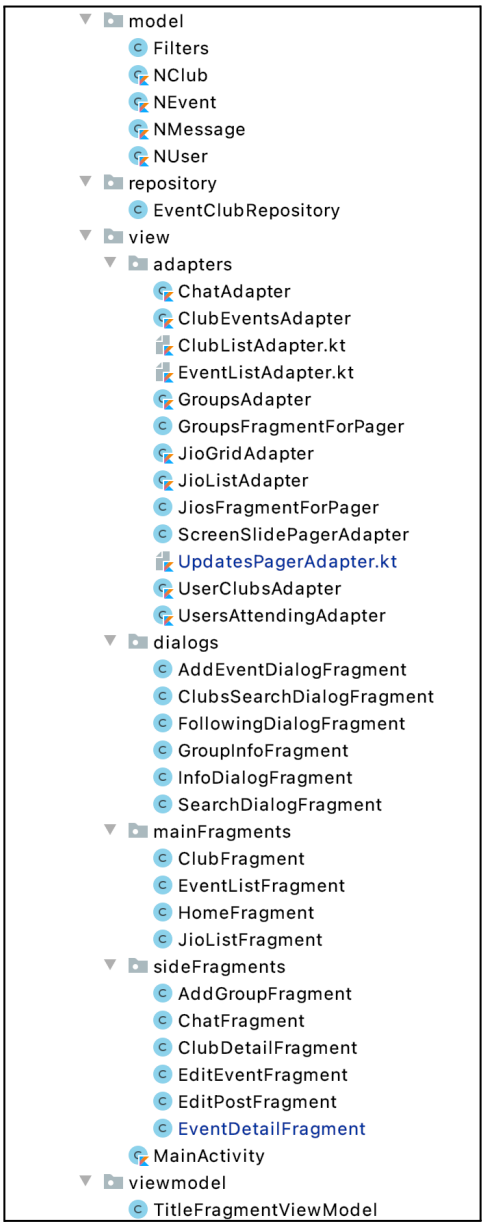
We decided to focus exclusively on Android development using Android Studio. We felt that a native app provides a smoother experience for users. In addition, Android Studio integrates well with Firebase which we used extensively for our backend.

MVVM

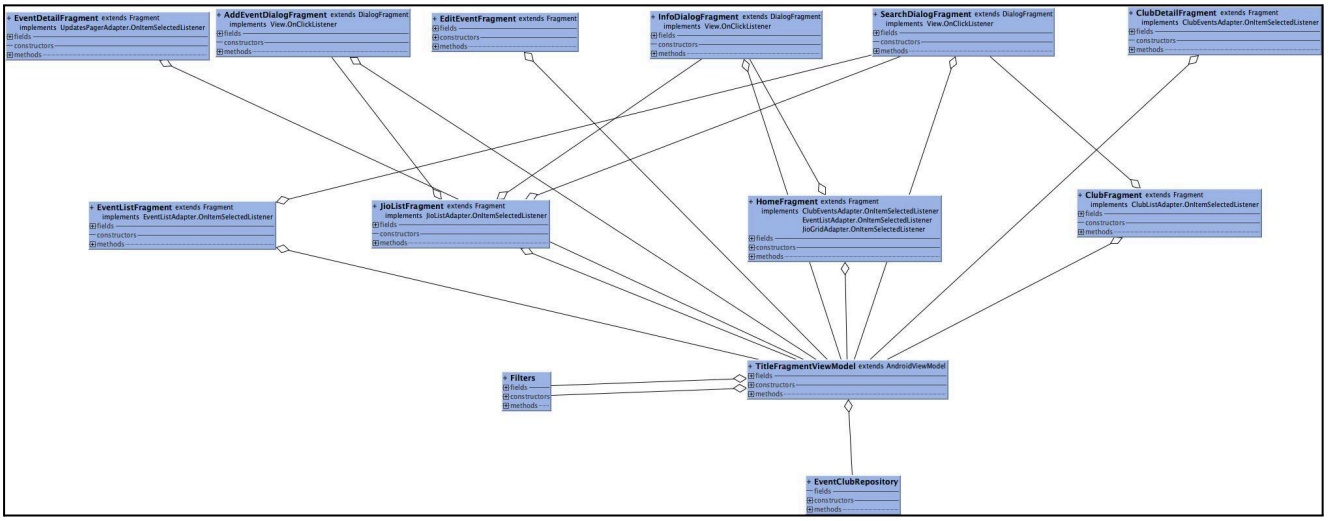


For NUSTown, we have adopted the Model-View-Viewmodel (MVVM) UI design pattern. Like other UI design patterns, MVVM ensures modularity through separation of concerns.

For MVVM, separation of concerns separates the **View** (representing UI), **Model** (representing Object Classes) and **ViewModel** (Handling Business Logic) as much as possible, allowing for better abstraction and reducing coupling. In our app, code for the **View** rests primarily in Fragments, the **Model** is implemented with object classes and the **ViewModel** supplies data from our database (via a repository) to the view.

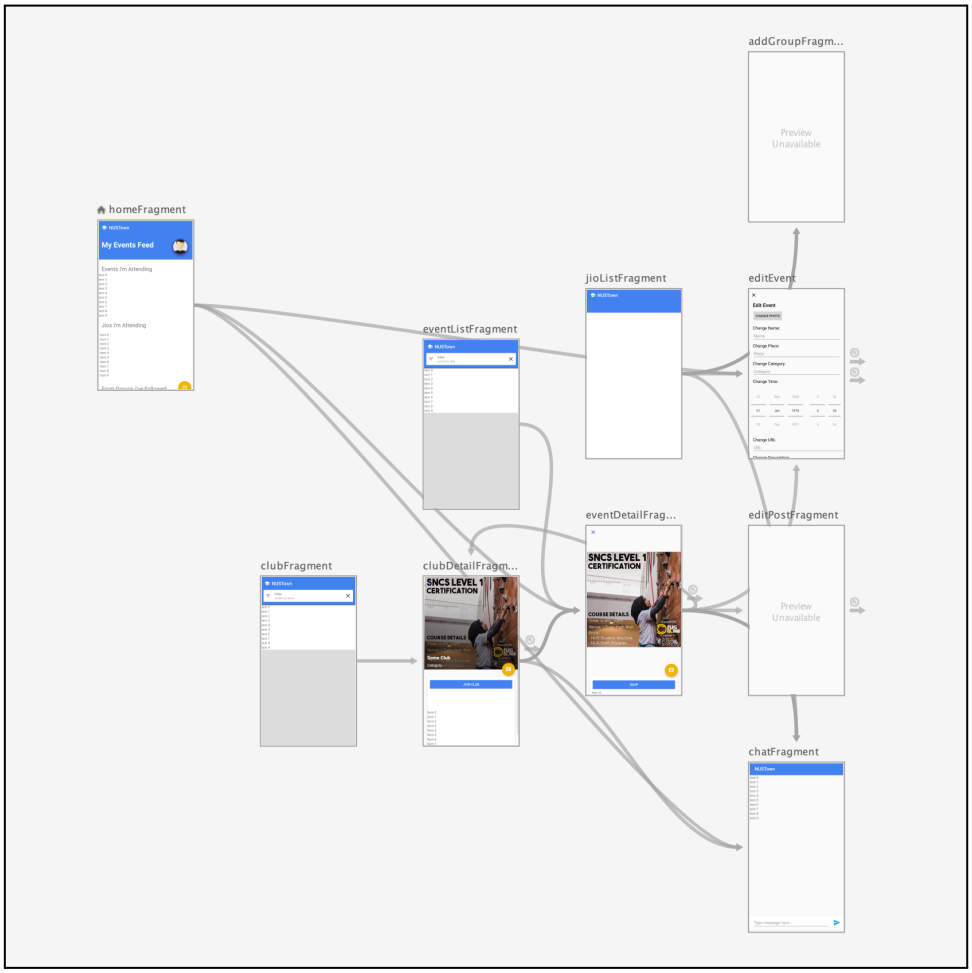


NUSTown Classes (MVVM Model)



Expanded Diagram:
https://drive.google.com/file/d/11tgju0pivbSEGa3FwsIQQLL-xdldRLXp/view?usp=sharing

Navigation Component (Navigate between Views)



Our views (as fragments) are linked together in a navigation graph (See Above). Navigation Components simplify the interaction between views.

Recycler Views and Adapters (View)

All lists of data from our database are displayed in RecyclerViews. RecyclerView is an enhanced list view that dynamically loads elements as users scroll through the list. Combined with pagination of Firestore data where results from database are retrieved in batches of 15, memory usage and database queries are significantly reduced.

An adapter handles the logic for the RecyclerView.

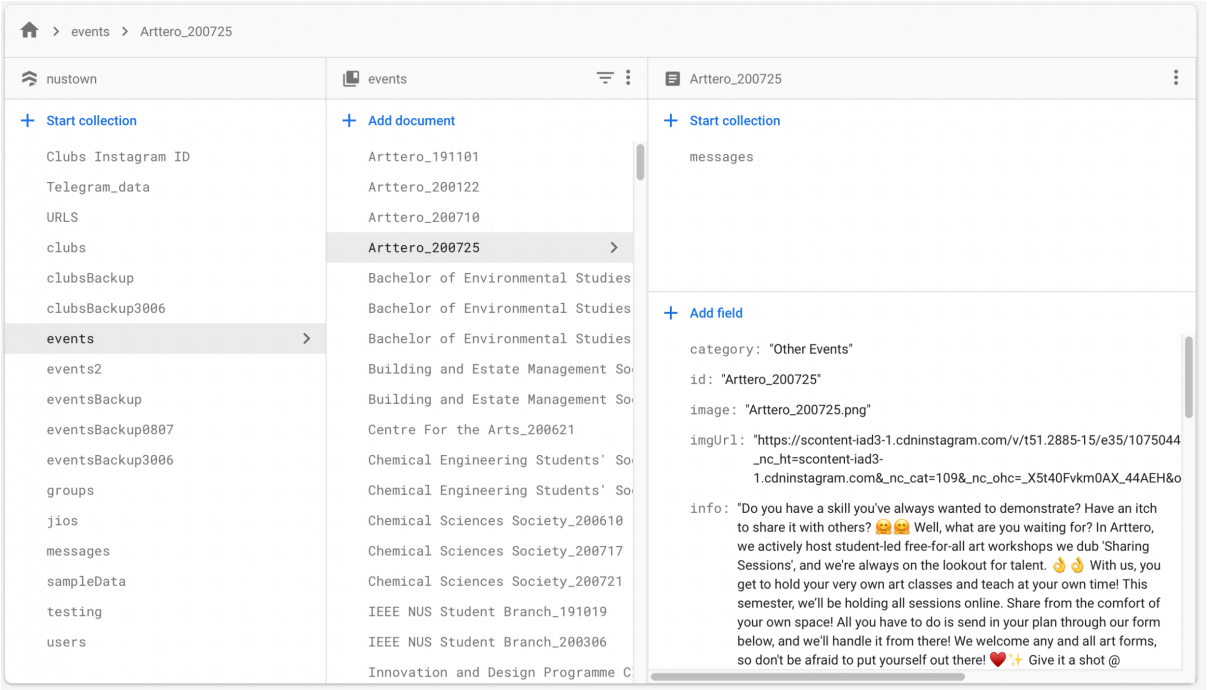
ViewModel
The ViewModel is the intermediary between the View and the Repository. ViewModel functions updates and returns LiveData. After asynchronous calls to the database are finished and results from the database successfully retrieved, the LiveData changes. The adapter for the RecyclerViews (see below) is then updated with the new data.

Repository
The repository abstracts database calls with generic functions that handle queries of different types. Some functions make calls to Firestore with .get() while others receive Firebase's QuerySnapshotListener that listens for changes. Some LiveData in ViewModel are attached to these SnapshotListeners and are automatically updated when a Firestore document is modified. For example, this allows the UsersAttending RecyclerView to be updated when a user clicks on a RSVP button in an event.

Backend Development: Firestore

As a Backend-as-a-Service (BaaS) platform, Firebase provides a unified SDK that links frontend applications with its various services. NUSTown uses Firebase Auth for managing users and security, Cloud Firestore as its database, Firebase functions to run periodical background functions and Cloud Storage to store media content.

Cloud Firestore: Introduction

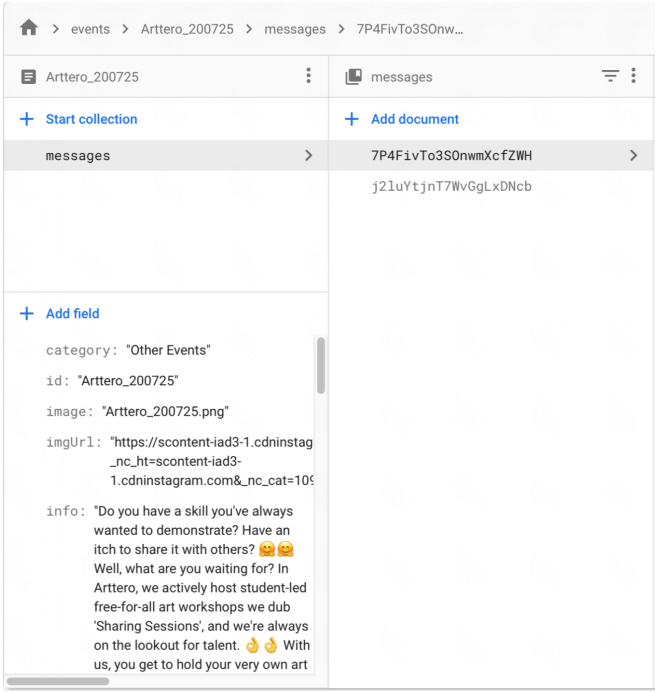


Cloud Firestore is a schemaless NoSQL database that stores data in documents which are organised into collections. Documents contain key-value pairs similar to JSON files. We choose Cloud Firestore as our database because of its scalability and efficiency. Cloud Firestore ensures near-constant querying time as it uses a hashmap type structure.

Firestore uses a Collection -> Document -> Collection -> Document (and so on) structure. Each document contains fields that can be strings, numbers, timestamp, arrays or maps among others. Documents can be sorted or filtered by these fields.

Cloud Firestore: Structure

Our top-level collections are events, jios, clubs, groups and users. Events and jios have the same data structure, likewise for clubs and groups.



Individual events, jios, clubs, groups are stored as documents in their respective collections. Chat messages are stored as documents in a messages collection under the events/jios/clubs/groups document.

Cloud Firestore: Events/Jios Structure

id	"NUS Climbing_190323"
imgUrl	"https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/e35/54510804_776889522676121_3010906346639395756_n.jpg?_nc_ht=scontent-iad3-1.cdninstagram.com&_nc_cat=105&_nc_ohc=64CvKSPWtl4AX8E7a7g&oh=baaa053be0268931e0013503293bee55&oe=5F3EF527"
info	"It's us back with SNCS Level 1 Certification! If you're an NUS student or staff, and are interested to learn how to belay, look no further as we bring you the Singapore National Climbing Standard (SNCS) Level 1 course! Dates open for sign-ups: either 23rd or 24th March 2019 (it is a one-day course) For more information, please visit http://nusclimb.com/courses or sign up now at https://orgsync.com/140428/events/2626620/occurrences/6445811 Hurry! Slots are limited and will be allocated on a first-come-first-served basis. Kindly note that confirmation of the course is subject to the availability of instructors and a sufficient number of participants. An email will be sent out upon confirmation of the course date."
isPastEvent	TRUE
lastUpdate	14 March 2019 at 10:52:05 UTC+8
name	"NUS Climbing"
org	"NUS Climbing"
postDate	"14 Mar"
time	23 March 2019 at 12:00:00 UTC+8

Updates -> 14 Mar[0]	"It’s us back with SNCS Level 1 Certification! If you’re an NUS student or staff, and are interested to learn how to belay, look no further as we bring you the Singapore National Climbing Standard (SNCS) Level 1 course! Dates open for sign-ups: either 23rd or 24th March 2019 (it is a one-day course) For more information, please visit http://nusclimb.com/courses or sign up now at https://orgsync.com/140428/events/2626620/occurrences/6445811 Hurry! Slots are limited and will be allocated on a first-come-first-served basis. Kindly note that confirmation of the course is subject to the availability of instructors and a sufficient number of participants. An email will be sent out upon confirmation of the course date."
Updates -> 14 Mar[1]	https://scontent-iad3-1.cdninstagram.com/v/t51.2885-15/e35/54510804_776889522676121_3010906346639395756_n.jpg?_nc_ht=scontent-iad3-1.cdninstagram.com&_nc_cat=105&_nc_ohc=64CvKSPWtl4AX8E7a7g&oh=baaa053be0268931e0013503293bee55&oe=5F3EF527
Updates -> 14 Mar[2]	"NUS Climbing_190323_14 Mar"
UsersAttending[0]	Sean Lee_jnhYXyEZpvYBHgEAXjymGWWt1Ca2

Above is a table showing the fields in an event document.

All posts of the same event are stored in the Updates map. This is a map of *post date:array[post caption, image url and Cloud Storage image name]*. This is used by UpdatesPagerAdapter in the app to display posts.

Cloud Firestore: Clubs/Groups Structure

cat	2
followers	462
imgUrl	"https://se-infra-imageserver2.azureedge.net/clink/images/97e4aea8-032e-462e-b30f-e4ff2daaba832f87f727-518e-4bfa-8723-74fd89621773.jpg"
info	"We are a group of individuals hoping to create an inclusive community for art makers and art lovers alike. We want you to create and discover, to bring out a creative learning spirit in campus."
name	"Arttero"
url	"https://nus.campuslabs.com/engage/organization/arttero"

Above is a table showing the fields in a club document.

Cloud Firestore: Users Structure

Field	Index (For Arrays)	Item

clubsSubscribedTo	0	"Arttero"
	1	"NUS Students' Science Club"
email		"tim@lee.com"
eventAttending	0	"Arttero_200725"
	1	"Society of Social Work Students_200725"
	2	"NUS Students' Science Club_200727
groupsSubscribedTo	0	"Sunset Appreciation Club"
jioEventAttending	0	"zIWf4tmx8mzcHOjCBL0P"

Above is a table showing the fields in a user document.

Backend Development: Crawler and Parser

Our Instagram and Telegram Crawler retrieves all posts by NUS Clubs. It sends these posts to our event parser which filters out non-event posts. Our NUSync crawler retrieves all events on NUSync (a limited number compared to Instagram/Telegram). Events documents, of the structure shown in the previous section, are sent to our database.

Instagram Posts Crawler

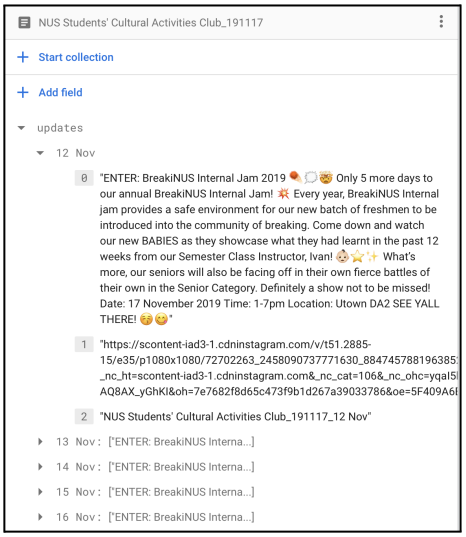
We used an Instagram HTTP endpoint to find the Instagram IDs of clubs given their username. In the Cloud Function ***alllnsta***, we then make a HTTP request to the GraphQL endpoint for the profile. This returns a JSON file (see above) that contains all the profiles' posts, including image links. We send the post captions to our parser to filter posts for events and store the events and their photos to Firestore and Cloud Storage (photos separately saved via another Cloud Function) respectively.

Events Parser

Instagram captions and Telegram messages are not organised into fields. Thus, a parser is needed to know if a post refers to a future event and to extract out key details like date and time. We forked ***Chrono,*** the open source natural language date parser, and customised it for our use case. Our modifications can be seen here. Specifically, we improved accuracy by detecting additional words commonly used in Instagram captions that related to time.

On top of modifications directly to the library, we wrote an algorithm that further filters the posts to make sure only future events are extracted, reduced sensitivity to more casual posts, and improved time detection. This can be seen in the function ***InstaToDatabase,*** called by allInsta.

Posts by an organisation of the same event date are treated as duplicate posts. Thus, if an event document already exists, additional posts are saved as updates (see right).



Telegram Crawler

We have built a telegram bot using a python library that interacts with telegram bot APIs. The bot polls for events and promotions by the members in the group at regular interval. The telegram bot codes are deployed on google cloud functions and it makes use of firebase SDK to directly update the firestore.

NUSync Events Crawler

In the cloud function ***nusSync***, we use NUSync's API to extract future events and send them to our database. If the event document already exists in our database (from other sources), it creates an update, labelled "NUSync" in the event document.

NUSync Clubs Crawler

In the cloud function ***nusClubs***, we use NUSync API to extract future events and send them to our database.

Backend Development: Cloud Functions

Firebase Cloud Functions is a serverless framework that lets us to automatically run backend code in response to events triggered by Firebase features and HTTPS requests. These functions are written in JavaScript or TypeScript, are stored in Google's cloud and run on their own according to the trigger set by us.

Introduction

In the previous section, you have already seen the crawler and parser which run on Cloud Functions. In our app, we have more than a dozen other Cloud Functions and they can be called directly from our application, triggered by a specific event or scheduled to run at a regular interval.

RSVP Function

Our event's and jio's RSVP function are called directly from our app when the user clicks on the buttons. Our app passes in the event's and user's information as parameters into the cloud functions. These parameters help the firebase function pinpoint the part of the database that these functions need to interact with. For example, the RSVP function would modify the list of users attending a particular event and the list of events the user attends.

New User Creation

Our trigger functions also ensure that the sign up procedure of a new user is taken care of. Whenever a new user registers their email with NUSTown for the first time, functions would be triggered to create a document under the user collections and initializes the clubs following and events attending array fields.

Trigger functions are also deployed when we have new data from our Telegram sources. For example, upon a Telegram message, a function is triggered to determine whether it's a valid event and whether there's a duplicate event. Further details on parsing have been elaborated in the previous section.

Cloud Scheduler

Google Cloud Scheduler, Google Cloud Functions (GCF) and Firebase Functions ensure that we get the latest updates of events happening in NUS. Our Instagram and NUSync Parser is written in Javascript in Firebase Functions while our Telegram Parser code is written in Python in GCF. To ensure that we keep our database up-to-date, we used the Google Cloud Scheduler (GCS) to run the parsers periodically.

For example, we have used GCS to periodically execute the Telegram bot polling function, where it will poll for new messages and update the firestore accordingly. GCS would make a http request to the Cloud Function to execute the polling function and it can be configured to run the code every minute to once every week.

Backend Development: Other Firebase Services

Firebase Authentication

Firebase Authentication offers a simple way to authenticate users and manage them. Through its tight integration with Cloud Firestore, for some areas of our database, we can restrict read and write-access. For example, only event organisers can be allowed to edit particular events or see the list of all participants who have RSVPed. Through FirebaseUI, Firebase supplies its own login UI for our app.

Firebase Cloud Storage

As Firebase Cloud Storage is integrated with Cloud Firestore, we store images we parse from our sources in Cloud Storage, and their location path in corresponding documents in Cloud Firestore. In our app, we are able to retrieve these images by looking up the location in these documents.

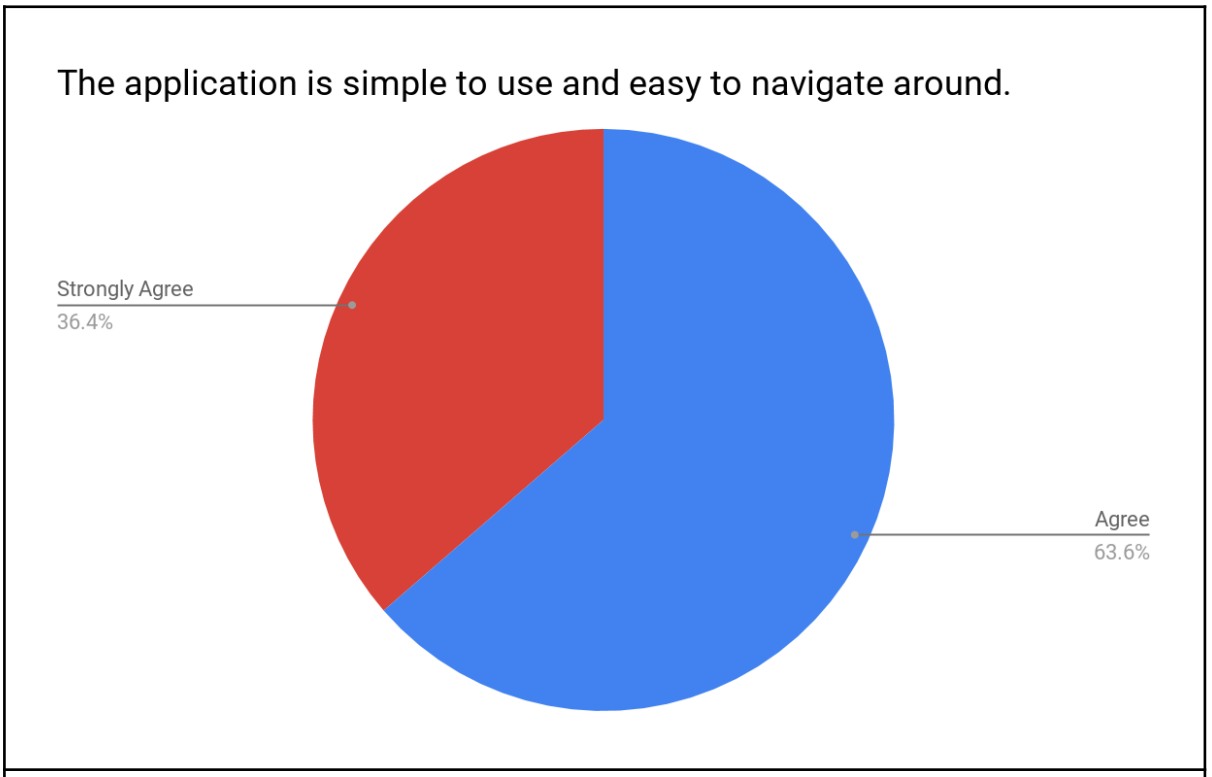
Appendix

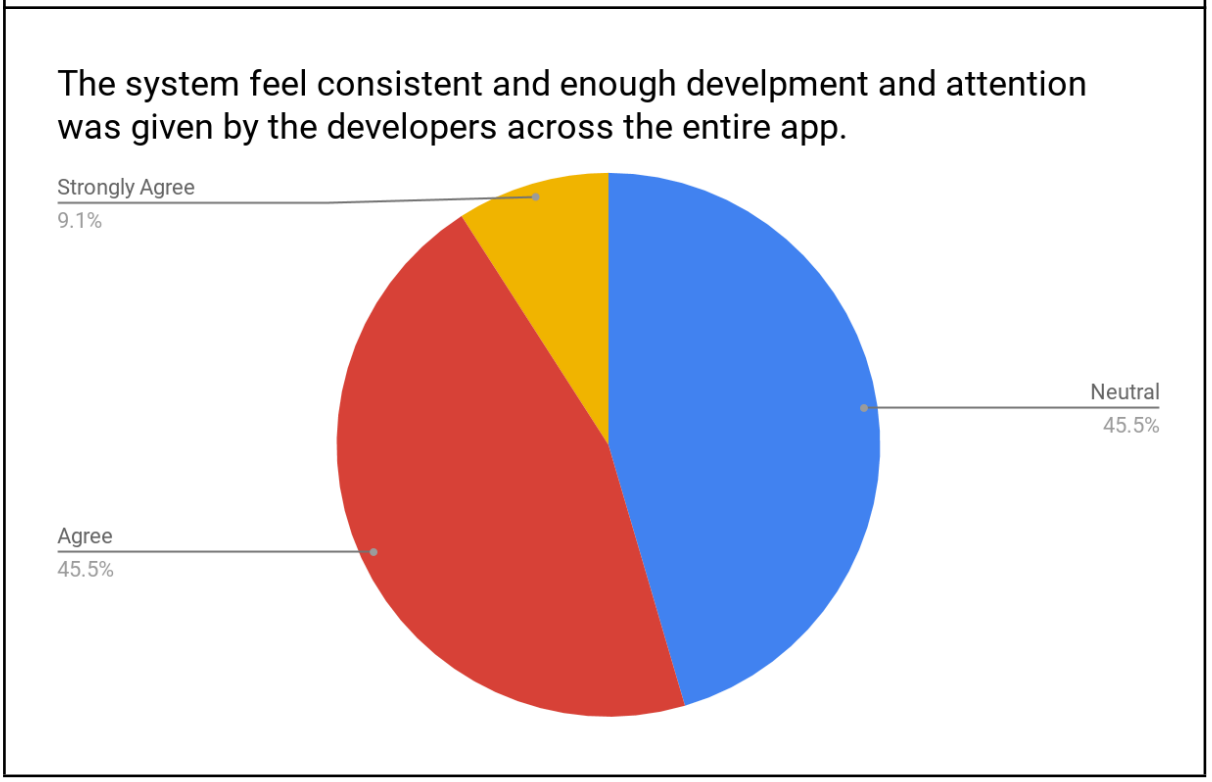
Appendix A

Testing for user friendliness (Instructions provided)	Questionnaire 1. Did the user take any extra steps? 2. Does the system feel consistent? 3. Was it easy to use the application?
Usefulness	Questionnaire 1. What do you think is the purpose of the feature? 2. How often would you use this feature? 3. Is there any alternative app or platform to serve the purpose of this feature? 4. Out of all the features, which is the most useful and which is the least useful feature?
UI experience	Questionnaire 1. What was the first impression of the app? 2. Is the interface simple and straightforward or is it too complicated and hard to navigate? 3. Does the visual help you with navigation or makes it harder to navigate the application?

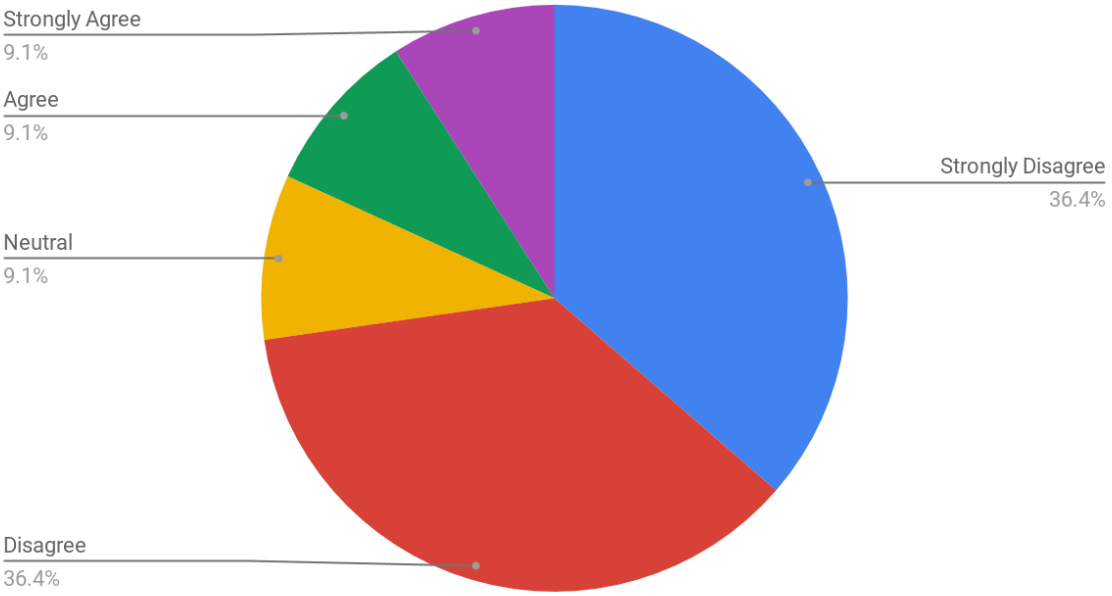
Appendix B

User testing survey results

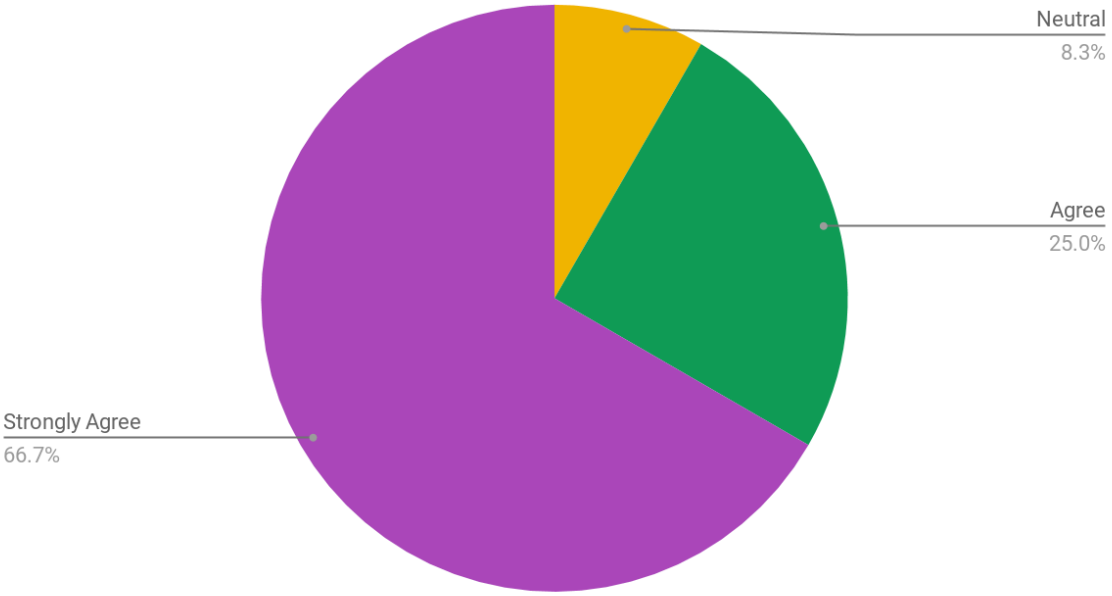




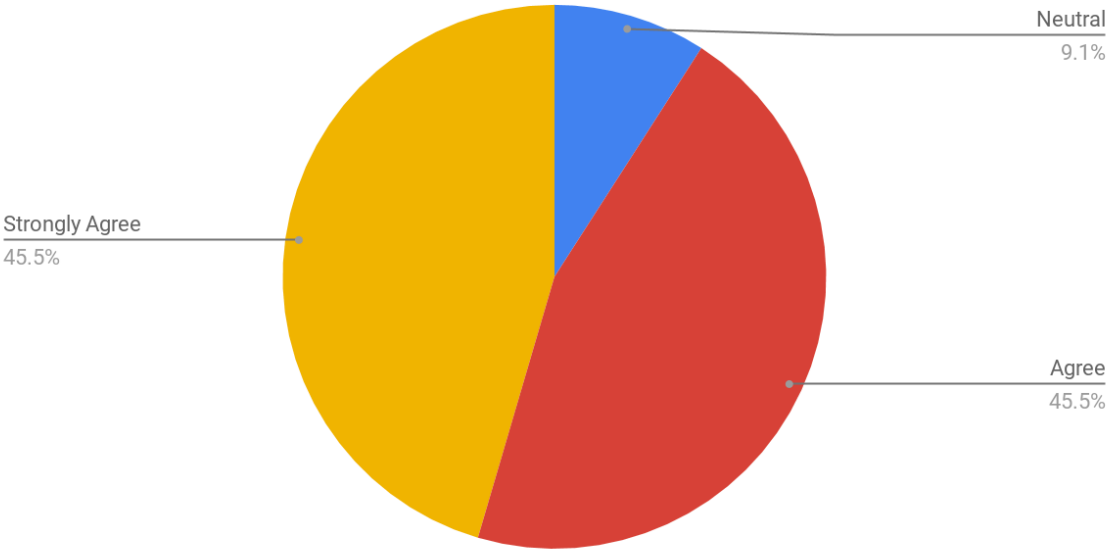
I feel that some parts of the application are unnecessarily complex.



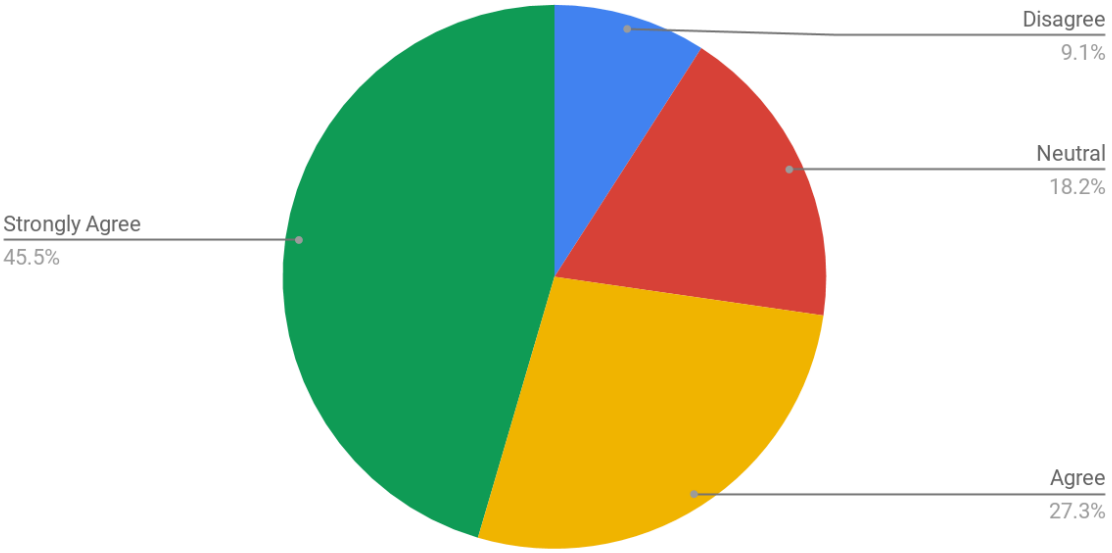
I think the various features are well integrated.



I would imagine that most people would learn to use this application very quickly.



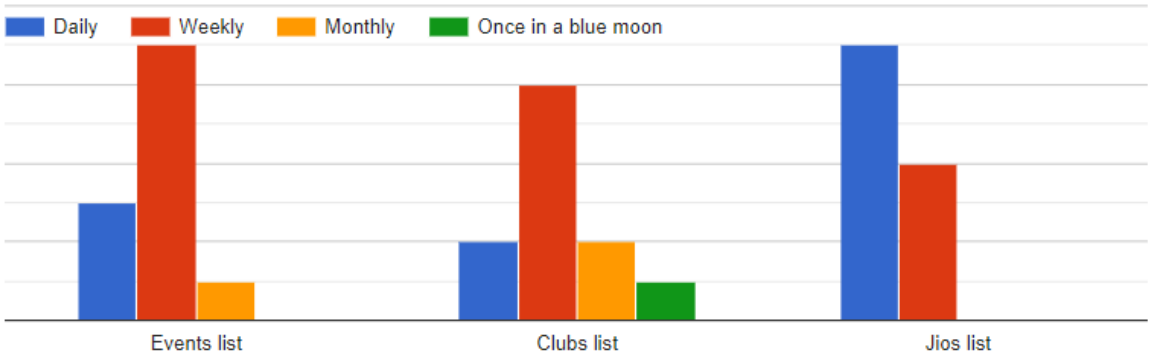
I don't need to learn a lot of things before I could get going with this system.



Does the feature serve its intended purpose well? If not, please state the feature below.

- All responses were Yes.

How often would you use each of the features? Events list to check for events and indicate attendance etc.



Out of all the features, which is the most useful feature for you and least useful feature for you?

What was the first impression of the app in terms of colors schemes and visual interface?

Dull

straightforward and clear

very simple but works fine

no colours

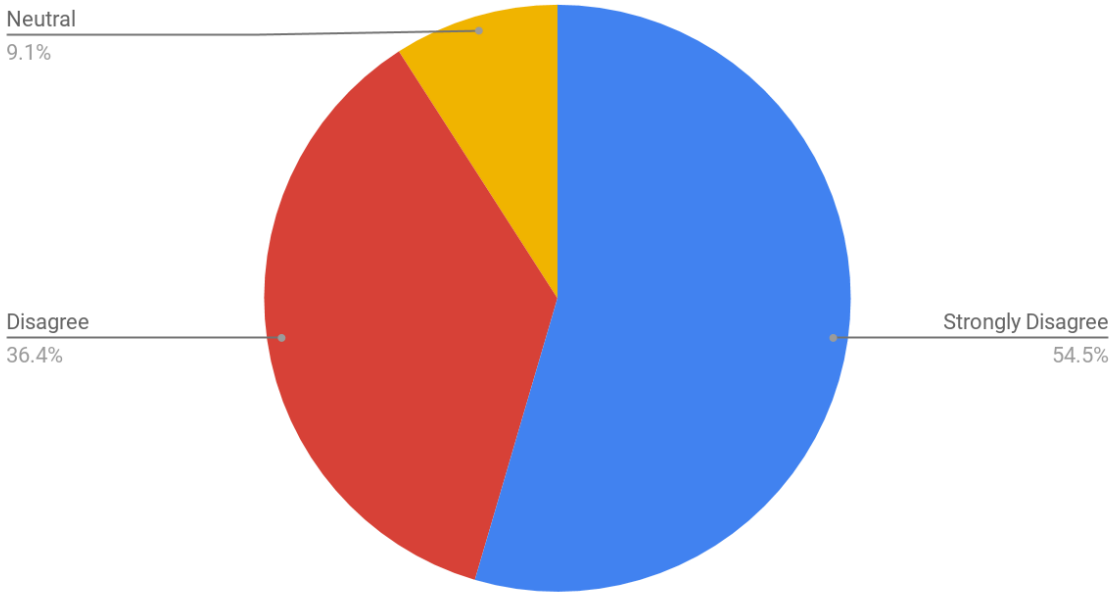
Boring

Minimum however serves its functionalities.

Plain

Minimalistic

Is the interface too complicated and hard to navigate?



Does the visual help in terms of making it easier to navigate?