Final Application Report: Wut’sUp

Danny Nsouli, 20531407, dmansouli@connect.ust.hk
Ivan Bardarov, 20501426, iebardarov@connect.ust.hk

1. Introduction:

Wut’sUp is a general meeting app that will include different types of events that users can set up and share with their friends on an interactive map. The goal of the app is to make event planning more convenient, especially for more spontaneous event ideas. The interactive map is available to help people easily find new events to enroll in around their area, while also encourage them to possibly meet new people at more general happenings. The pins on the map also allow users to see a detailed description of the event before deciding whether or not to enroll.

2.1 Requirement Analysis:

- Interactive map to display all events around the user (location services)
    - Google Maps API v2 integration
    - Detecting the current user location
    - Interactive event pins that lead to full event description pages
- List view of the available events
- Event creator with multiple options to help describe the event organized
    - Event title
    - Time/Date
        - Dropdown calendar to select the date
        - 24h format clock to select the time
    - Capacity
    - Location
        - Press on the screen to select location
        - Search by address to detect location
    - Description
    - Type
        - Dropdown list to select from already existing types
- Individual profiles
    - Log in by
        - Creating new profile
        - Existing Google profile
        - Existing Facebook profile

    - Remember the user so that he/she does not have to log in every time
    - Customizable profile photo
        - Take a photo with the camera to change the profile photo
        - Select a photo from the gallery
    - Owned/Created events list
        - Navigate to the details of the event
        - See them on the map
    - Events enrolled in list
        - Navigate to the details of the event
        - See them on the map
- Support landscape configuration for using the app in landscape mode

2.2 Design:

The project consists of two packages, one for the model that are used around for easier data handling and another one for the activity controlling classes.

Model classes

The main in the project is the **Event class** which contains all the information for a specific event and is used to pass the information around more easily. A custom **Observer class** because of its simpler nature than the one included in the Java packages. Its purpose is to update the users view with events every time an event is added or removed and prevent corrupted data to be shown to the user. The Observable class updates the custom **MyEventRecyclerViewAdapter** which is the model for the way events are going to be shown as a list to the user. **CustomInfoWindowAdapter** takes care of the proper visualisation of the event information on the map and allows the user to click on it and go to the details of the events. Also, it can be future expanded by adding the option to add custom event images and the info window displaying it. **PlaceAutocompleteAdapter** is the google maps implementation for being able to search for places by street name or other distinctive information. There are also two helper methods, **Activities** and **Resources**. **Activities** works as an enum class for easier navigation between different activities. **Resources** only contains the profile photo at this point, whenever the user visits the Profile activity the photo is temporarily saved so that it does not have to be downloaded next time the user goes to the activity.

Activity classes

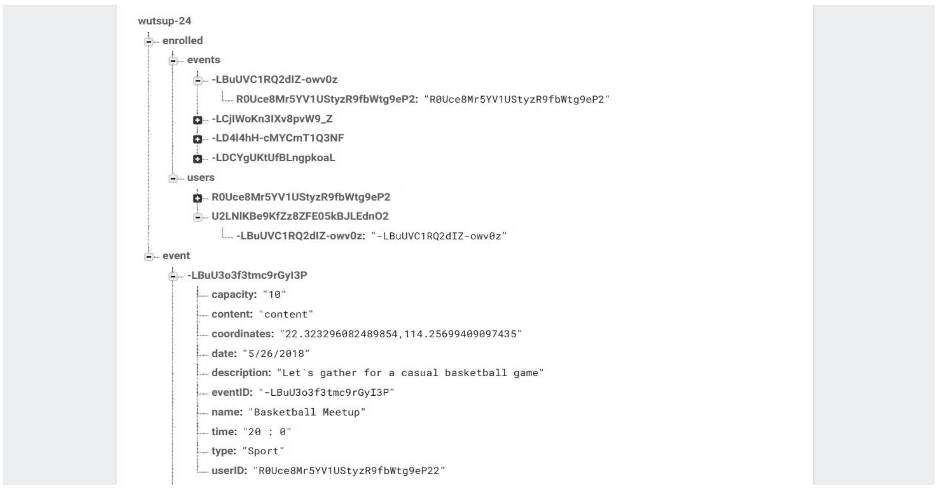
AddEventActivity is used from the user to add a new event to the database. It contains all the fields that need to be filled and also navigates to the **MapActivity** after certain checks are performed for the permissions from the user. The **MapActivity** contain all the logic that is connected to maps and is different depending on which activity it is called from, finding the current location and moving the camera to it, putting markers for the events, selecting coordinates for the event location, searching by text. The communication with FireBase Realtime Database is executed through the **DatabaseCommunicatior** class, which contains static methods for inputing data into the database or getting data from the database. It also stores the events locally for faster retrieval following times. It is an Observable class since it sends updates to the UI if there are changes to the database. **EventDetailsActivity** is the activity that gets called when an event has been selected and allows the user to enroll in different events. It creates the **EventDetailsFragment** which is responsible for visualizing the information to the user. **EventListActivity** contains the recycle view which shows a list of the event and allows the user to select one of them. If a device with larger screen is detected then it makes sure it adds the **EventDetailsFragment** to the side to make use of the larger space on the screen and simultaneously show the list and the details of one of the events. The **MainActivity** is responsible for calling the Firebase Auth services and navigate to the **OptionsActivity.** The **OptionsActivity** contains the buttons to navigate to the respective activities to add event, explore events or check the user profile. The last activity is the **ProfileActivity** which displays the user profile photo and gives them the option to change it, and also navigates to **EventListActivity** for events created by the user and events that the user has enrolled in.

2.3 Implementation:

AddEventActivity

- Participates in event creation
  - Initializes all event object attributes
  - Sets all of the user input data from the event creator into the different variables of the new event object
  - If an attribute is left unaltered then there will be an error (event cannot be successfully added)
  - Use Date/TimePickerDialog to allow user to pick date/time and Calendar class to get current date/time
  - For “event type” drop down menu use spinner with attached ArrayAdapter<String> for possible dropdown choices

DataBaseCommunicator



```
wutsup-24
 ⊟ enrolled
    ⊟ events
       ⊟ -LBuUVC1RQ2dIZ-owv0z
          └ R0Uce8Mr5YV1UStyzR9fbWtg9eP2: "R0Uce8Mr5YV1UStyzR9fbWtg9eP2"
       ⊞ -LCjlWoKn3IXv8pvW9_Z
       ⊞ -LD4l4hH-cMYCmT1Q3NF
       ⊞ -LDCYgUKtUfBLngpkoaL
    ⊟ users
       ⊞ R0Uce8Mr5YV1UStyzR9fbWtg9eP2
       ⊟ U2LNlKBe9KfZz8ZFE05kBJLEdnO2
          └ -LBuUVC1RQ2dIZ-owv0z: "-LBuUVC1RQ2dIZ-owv0z"
 ⊟ event
    ⊟ -LBuU3o3f3tmc9rGyI3P
       ├ capacity: "10"
       ├ content: "content"
       ├ coordinates: "22.323296082489854,114.25699409097435"
       ├ date: "5/26/2018"
       ├ description: "Let`s gather for a casual basketball game"
       ├ eventID: "-LBuU3o3f3tmc9rGyI3P"
       ├ name: "Basketball Meetup"
       ├ time: "20 : 0"
       ├ type: "Sport"
       └ userID: "R0Uce8Mr5YV1UStyzR9fbWtg9eP22"
```

- Makes use of Firebase Realtime Storage and stores the data in the way shown above. Relates eventIDs with userIDs to store information about enrollment and relates event details with eventID to store the details of the information
- Contains a list with all the events in the database and a list with the events that the user is involved in. Sends updates to the activities to update their information once it is downloaded.
- Adds events to the database
- Adds enrollment of users in different events

EventDetailActivity

- Enroll Button with a listener to add the event through the database communicator
- Sets the event detail fragment that shows the information

EventDetailFragment

- TextView fields are created in the layout of the activity in order to show all the accompanying data of the event from its record in the event database

EventListActivity

- Outputs the correct list of events depending on what intent led to the current activity
- Uses if then statements to get the correct list between 3 choices
    - Main page (options activity) = all events from database
    - Profile (show enrolled events button) = only those from enrolled events database
    - Profile (show hosted/user created events button) = those from events database that have the user ID in their record
- Shows a progress bar if the data to be displayed is not yet loaded, which gets dismissed when it receives an update from the Observable

MainActivity

- Checks if the user is already signed in from previous session
- Starts a new activity for login in using the Firebase SDK to construct the intent.
- When logged in successfully navigates to the options activity

MapsActivity

- Checks if the required permissions are granted and asks if they are not
- Implements OnMapReadyCallback interface provided by the google maps api
- When the map is read, tries to get the user location with the FusedLocationProvider and moves the camera to the coordinates provided by it.
- Checks what is the calling activity using the Activities enum class and hides certain elements depending on the needs for each of them
- Makes use of Geocoder to find a place by the address that the user gives using the search text AutoCompleteTextView.

- In case the calling activity is an activity to show events on the map then markers are put for each of the events taking their coordinates as location
- Custom info window is added to the marker to show information about the event to the user
- A listener which waits for user clicks is added so that it navigates to the event details if the custom info window is pressed

OptionsActivity

- Main purpose of that class is navigation to three other activities.

ProfileActivity

- Navigates to EventListActivity to show events that were created by the user or the user has enrolled by providing two buttons.
- On starting the activity it sets up the lists with the events which are passed to the EventListActivity after
- Shows the user profile photo in an Image view by connecting to the database with an AsyncTask
- Displays the username of the user which it gets from the firebase auth instance
- In case the user wants to change the profile photo a new alertdialog is shown letting them to choose between taking a photo or selecting a photo from the gallery.
- Sends the respective intents depending on the users preference and uploads the photo to Firebase Storage
- Updates the image view after the photo has been successfully uploaded using AsyncTask again
- Contains logic for flipping images if they are flipped to the wrong side

3. Testing and Evaluation:

Integration Testing

- Is able to handle interrupts from OS (sent deliberate messages from other applications)
- Integrates well with other core device features (back and home buttons work well with the interface)
- Suspends and resumes as expected (user can stay logged in)

Usability Testing

- Power consumption rate normal, full day of testing didn't not consume significant battery life
- Easy to navigate, buttons large enough to quickly tap and images made for quicker navigation through visual comprehension

Conformance Testing

- Google Maps API v2 and Firebase utilities readily accessible, no policy or licensing issues with image

Edge Case Testing

- Events are unable to be added unless all their required description fields filled
    - Otherwise will see red highlight over neglected fields
- Unofficial locations not found by google maps API are unselectable

Screen Orientation Change Testing

- All layouts are able to seamlessly transition between landscape and portrait mode on device (used separate layout files for each position for most activities)

Configuration Changes Testing

- Keyboard easily accessible for all required input activities

External Resources Dependence Testing

- Reliable availability of network access in order to receive information from Firebase database server and google maps data
- GPS availability also reliable in terms of functionality, application was always able to find user location

4. Conclusion:

Our team was able to deliver on most of the features that we planned out. The application is able to open a map view successfully in order to both attach locations to events while also being able to check out other events in the surrounding area marked by clickable pins. There is a login screen to allow for individual profiles with customizable profile pictures and the ability to see events that that user created/owns and enrolled in. The event creator system also was achieved beyond what we expected with multiple attributes that a user is able to assign to an event including a set of predetermined types (i.e. sport, party), time, and date. A shortcoming would be that the application doesn't have a friend system, meaning that a user cannot friend other users in order to create more private events (no invitation system). All events are available to be enrolled in by anyone using the service, which wouldn't be preferable if the app gains a substantial user base in the future. As well, there is no direct messaging system as a result. There is also no event search by category, the events will only be able to be selected via a list view ranked by newest creation by a user or the map.

5. References:

https://play.google.com/store/apps/details?id=com.meetup

https://play.google.com/store/apps/details?id=com.eventbrite.attendee
https://play.google.com/store/apps/details?id=com.hellow_app.hellow
https://play.google.com/store/apps/details?id=com.mapmymeet.act
https://play.google.com/store/apps/details?id=com.ushtestudios.invit
https://developers.google.com/maps/documentation/android-sdk/start

Final Application Report: Wut'sUp

Danny Nsouli, 20531407, dmansouli@connect.ust.hk
Ivan Bardarov, 20501426, iebardarov@connect.ust.hk

1. Introduction:

Wut'sUp is a general meeting app that will include different types of events that users can set up and share with their friends on an interactive map. The goal of the app is to make event planning more convenient, especially for more spontaneous event ideas. The interactive map is available to help people easily find new events to enroll in around their area, while also encourage them to possibly meet new people at more general happenings. The pins on the map also allow users to see a detailed description of the event before deciding whether or not to enroll.

2.1 Requirement Analysis:

- Interactive map to display all events around the user (location services)
  - Google Maps API v2 integration
  - Detecting the current user location
  - Interactive event pins that lead to full event description pages
- List view of the available events
- Event creator with multiple options to help describe the event organized
  - Event title
  - Time/Date
    - Dropdown calendar to select the date
    - 24h format clock to select the time
  - Capacity
  - Location
    - Press on the screen to select location

        - Search by address to detect location
    - Description
    - Type
        - Dropdown list to select from already existing types
- Individual profiles
    - Log in by
        - Creating new profile
        - Existing Google profile
        - Existing Facebook profile
    - Remember the user so that he/she does not have to log in every time
    - Customizable profile photo
        - Take a photo with the camera to change the profile photo
        - Select a photo from the gallery
    - Owned/Created events list
        - Navigate to the details of the event
        - See them on the map
    - Events enrolled in list
        - Navigate to the details of the event
        - See them on the map
- Support landscape configuration for using the app in landscape mode

2.2 Design:

The project consists of two packages, one for the model that are used around for easier data handling and another one for the activity controlling classes.

Model classes

The main in the project is the **Event class** which contains all the information for a specific event and is used to pass the information around more easily. A custom **Observer class** because of its simpler nature than the one included in the Java packages. Its purpose is to update the users view with events every time an event is added or removed and prevent corrupted data to be shown to the user. The Observable class updates the custom **MyEventRecyclerViewAdapter** which is the model for the way events are going to be shown as a list to the user. **CustomInfoWindowAdapter** takes care of the proper visualisation of the event information on the map and allows the user to click on it and go to the details of the events. Also, it can be future expanded by adding the option to add custom event images and the info window displaying it. **PlaceAutocompleteAdapter** is the google maps implementation for being able to search for places by street name or other distinctive information. There are also two helper methods, **Activities** and **Resources**. **Activities** works as an enum class for easier navigation between different activities. **Resources** only contains the profile photo at this point, whenever the user visits the Profile activity the photo is temporarily saved so that it does not have to be downloaded next time the user goes to the activity.

Activity classes

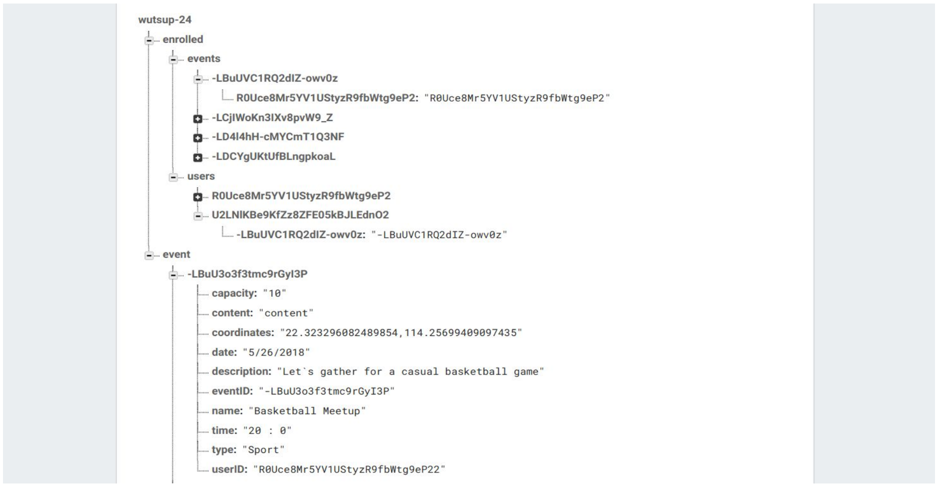
AddEventActivity is used from the user to add a new event to the database. It contains all the fields that need to be filled and also navigates to the **MapActivity** after certain checks are performed for the permissions from the user. The **MapActivity** contain all the logic that is connected to maps and is different depending on which activity it is called from, finding the current location and moving the camera to it, putting markers for the events, selecting coordinates for the event location, searching by text. The communication with FireBase Realtime Database is executed through the **DatabaseCommunicatior** class, which contains static methods for inputing data into the database or getting data from the database. It also stores the events locally for faster retrieval following times. It is an Observable class since it sends updates to the UI if there are changes to the database. **EventDetailsActivity** is the activity that gets called when an event has been selected and allows the user to enroll in different events. It creates the **EventDetailsFragment** which is responsible for visualizing the information to the user. **EventListActivity** contains the recycle view which shows a list of the event and allows the user to select one of them. If a device with larger screen is detected then it makes sure it adds the **EventDetailsFragment** to the side to make use of the larger space on the screen and simultaneously show the list and the details of one of the events. The **MainActivity** is responsible for calling the Firabase Auth services and navigate to the **OptionsActivity.** The **OptionsActivity** contains the buttons to navigate to the respective activities to add event, explore events or check the user profile. The last activity is the **ProfileActivity** which displays the user profile photo and gives them the option to change it, and also navigates to **EventListActivity** for events created by the user and events that the user has enrolled in.

2.3 Implementation:

AddEventActivity

- Participates in event creation
    - Initializes all event object attributes
    - Sets all of the user input data from the event creator into the different variables of the new event object
    - If an attribute is left unaltered then there will be an error (event cannot be successfully added)
    - Use Date/TimePickerDialog to allow user to pick date/time and Calendar class to get current date/time
    - For “event type” drop down menu use spinner with attached ArrayAdapter<String> for possible dropdown choices

DataBaseCommunicator



```
wutsup-24
 enrolled
   events
     -LBuUVC1RQ2dIZ-owv0z
       R0Uce8Mr5YV1UStyzR9fbWtg9eP2: "R0Uce8Mr5YV1UStyzR9fbWtg9eP2"
     -LCjIWoKn3IXv8pvW9_Z
     -LD4l4hH-cMYCmT1Q3NF
     -LDCYgUKtUfBLngpkoaL
   users
     R0Uce8Mr5YV1UStyzR9fbWtg9eP2
     U2LNlKBe9KfZz8ZFE05kBJLEdnO2
       -LBuUVC1RQ2dIZ-owv0z: "-LBuUVC1RQ2dIZ-owv0z"
 event
   -LBuU3o3f3tmc9rGyI3P
     capacity: "10"
     content: "content"
     coordinates: "22.323296082489854,114.25699409097435"
     date: "5/26/2018"
     description: "Let`s gather for a casual basketball game"
     eventID: "-LBuU3o3f3tmc9rGyI3P"
     name: "Basketball Meetup"
     time: "20 : 0"
     type: "Sport"
     userID: "R0Uce8Mr5YV1UStyzR9fbWtg9eP22"
```

- Makes use of Firebase Realtime Storage and stores the data in the way shown above. Relates eventIDs with userIDs to store information about enrollment and relates event details with eventID to store the details of the information
- Contains a list with all the events in the database and a list with the events that the user is involved in. Sends updates to the activities to update their information once it is downloaded.
- Adds events to the database
- Adds enrollment of users in different events

EventDetailActivity

- Enroll Button with a listener to add the event through the database communicator
- Sets the event detail fragment that shows the information

EventDetailFragment

- TextView fields are created in the layout of the activity in order to show all the accompanying data of the event from its record in the event database

EventListActivity

- Outputs the correct list of events depending on what intent led to the current activity
- Uses if then statements to get the correct list between 3 choices
    - Main page (options activity) = all events from database
    - Profile (show enrolled events button) = only those from enrolled events database
    - Profile (show hosted/user created events button) = those from events database that have the user ID in their record
- Shows a progress bar if the data to be displayed is not yet loaded, which gets dismissed when it receives an update from the Observable

MainActivity

- Checks if the user is already signed in from previous session

- Starts a new activity for login in using the Firebase SDK to construct the intent.
- When logged in successfully navigates to the options activity

MapsActivity

- Checks if the required permissions are granted and asks if they are not
- Implements OnMapReadyCallback interface provided by the google maps api
- When the map is read, tries to get the user location with the FusedLocationProvider and moves the camera to the coordinates provided by it.
- Checks what is the calling activity using the Activities enum class and hides certain elements depending on the needs for each of them
- Makes use of Geocoder to find a place by the address that the user gives using the search text AutoCompleteTextView.
- In case the calling activity is an activity to show events on the map then markers are put for each of the events taking their coordinates as location
- Custom info window is added to the marker to show information about the event to the user
- A listener which waits for user clicks is added so that it navigates to the event details if the custom info window is pressed

OptionsActivity

- Main purpose of that class is navigation to three other activities.

ProfileActivity

- Navigates to EventListActivity to show events that were created by the user or the user has enrolled by providing two buttons.
- On starting the activity it sets up the lists with the events which are passed to the EventListActivity after
- Shows the user profile photo in an Image view by connecting to the database with an AsyncTask
- Displays the username of the user which it gets from the firebase auth instance
- In case the user wants to change the profile photo a new alertdialog is shown letting them to choose between taking a photo or selecting a photo from the gallery.
- Sends the respective intents depending on the users preference and uploads the photo to Firebase Storage
- Updates the image view after the photo has been successfully uploaded using AsyncTask again
- Contains logic for flipping images if they are flipped to the wrong side

<u>3. Testing and Evaluation:</u>

Integration Testing

- Is able to handle interrupts from OS (sent deliberate messages from other applications)
- Integrates well with other core device features (back and home buttons work well with the interface)
- Suspends and resumes as expected (user can stay logged in)

Usability Testing

- Power consumption rate normal, full day of testing didn't not consume significant battery life
- Easy to navigate, buttons large enough to quickly tap and images made for quicker navigation through visual comprehension

Conformance Testing

- Google Maps API v2 and Firebase utilities readily accessible, no policy or licensing issues with image

Edge Case Testing

- Events are unable to be added unless all their required description fields filled
    - Otherwise will see red highlight over neglected fields
- Unofficial locations not found by google maps API are unselectable

Screen Orientation Change Testing

- All layouts are able to seamlessly transition between landscape and portrait mode on device (used separate layout files for each position for most activities)

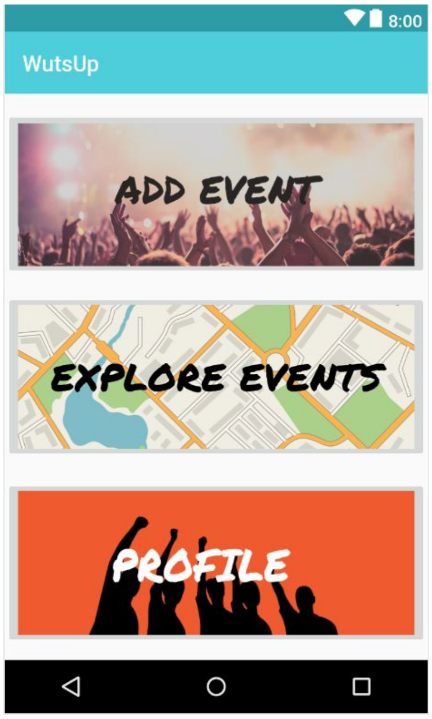
Configuration Changes Testing

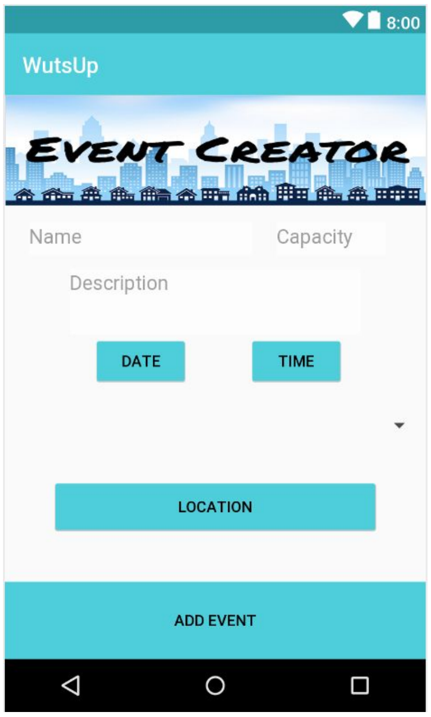
- Keyboard easily accessible for all required input activities

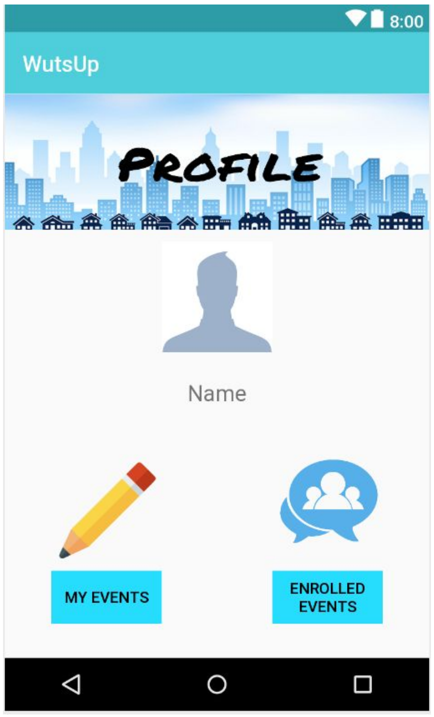
External Resources Dependence Testing

- Reliable availability of network access in order to receive information from Firebase database server and google maps data
- GPS availability also reliable in terms of functionality, application was always able to find user location

4. Screenshots:







5. Conclusion:

Our team was able to deliver on most of the features that we planned out. The application is able to open a map view successfully in order to both attach locations to events while also being able to check out other events in the surrounding area marked by clickable pins. There is a login screen to allow for individual profiles with customizable profile pictures and the ability to see events that that user created/owns and enrolled in. The event creator system also was achieved beyond what we expected with multiple attributes that a user is able to assign to an event including a set of predetermined types (i.e. sport, party), time, and date. A shortcoming would be that the application doesn’t have a friend system, meaning that a user cannot friend other users in order to create more private events (no invitation system). All events are available to be enrolled in by anyone using the service, which wouldn’t be preferable if the app gains a substantial user base in the future. As well, there is no direct messaging system as a result. There is also no event search by category, the events will only be able to be selected via a list view ranked by newest creation by a user or the map.

6. References:

https://play.google.com/store/apps/details?id=com.meetup
https://play.google.com/store/apps/details?id=com.eventbrite.attendee
https://play.google.com/store/apps/details?id=com.hellow_app.hellow
https://play.google.com/store/apps/details?id=com.mapmymeet.act
https://play.google.com/store/apps/details?id=com.ushtestudios.invit
https://developers.google.com/maps/documentation/android-sdk/start