Testing

Carlos

Equivalence classes:

-EC(AcquireLocation):

- Description: Acquire the latitude and longitude from the phone GPS or Network
- Inputs: A call to the network/GPS with the permissions from the user.
- Outputs: A object location with the latitude and longitude from the current phone location, a call to the phone's permission screen.
- Possible input: [Permission location allowed, network]
- Possible output: [location (latitude, longitude)]

-EC(locationCalculation):

- Description: Calculate distance from current position and some event.
- Inputs: The inputs for location are latitude and location, provided by Geocoder and the Network/GPS of the phone. Therefore, no values should be out of bound. Longitude is on the range of [-180, 180), while Latitude is on the range of [-90, 90]. A string specifying "No" is entered too.
- Outputs: distance is shown in either meters or kilometers in the form of an integer. If it is 1000 or more meters, the function converts it to kilometers (rounds to an integer. No decimals), and the string is converted to "Yes"
- Possible input: [origin latitude of 36.5827 and origin longitude of 122.147, destination latitude 34.1975, destination longitude 119.1771, string "No"], [origin latitude of 37.3382 and origin longitude of 121.8863, destination latitude 37.7749, destination longitude 122.4194, string "No"]
- Possible output: [Between 410 and 415. String is "Yes"], [Between 94 and 98. String is "Yes"]

-EC(getLocationFromAddress):

- Description: Computes the LatLng object from the address specified by the user.
- Inputs: A string representing some form of address or location.
- Outputs: A LatLng object corresponding to the closest match of the address that was entered. If no possible address match is found, the function simply returns NULL.
- Possible input: [1156 High St, Santa Cruz, CA 95064], [Mexico City], [San Francisco, CA], [Wayne Manor, Gotham City]
- Possible output: [LatLng (36.9776344,-122.0564944)], [LatLng(19.3911668,-99.4238156)], [LatLng(37.757815,-122.5076406)], [NULL]

Stefano

- When I first posted an event on Firebase, I checked that all the fields were saved correctly, while for those events with missing fields I checked that the application blocked the upload. When I worked on the retrieving part, I first used a Map to check that the fields saved on the database were there, then I created several events in order to test it.
- For the participation feature, I first tested it with just an account, to check if the participant was saved correctly on the database, then I checked with multiple accounts, to simulate a larger population of users. I then checked that the
- For the User and Host information, I first checked that the fields were all saved online, then that the missing important fields were detected, then that the info were visible on the host event page, without the optional empty ones.
- Whenever I had to integrate different parts made by me or the others, I first checked that the new functionalities were working consistently, then that the old ones were working properly.
- When I implemented the search functionality, I checked that the events with the keyword were shown and that the ones without it were not. At the moment they appear twice as one of the other team mates edited improperly the saving functionality.
- Whenever I added UI parts I checked that they appeared properly.
- When I worked on the Categories, I first checked that all the events appear only in the correct tab, then that they were both in ALL and in their category.
- When I linked hosts and events, I first checked that each event was saved with the host email (identifier) on the database, then I checked that each event was properly linked to the correct host profile.

Francesco

- My first and main task during sprint 1 was to create and post locally fake events to test their correct display. The testing in this case was fairly simple, I first made sure that a normal event was saved in the preferences correctly, then I tried with my second equivalence class, uncomplete events, to make sure the system detected missing fields and notified the user.
- The sorting functionality that I implemented has 4 possible modes: distance, upcoming, popularity and recently created. The most interesting one from a testing point of view was the upcoming one, as I had to try a large number of combination of dates and times to test this feature.
- Another feature in sprint 2 was the possibility to add an image linked to the event posted. What I concentrated my testing on was the capability of my compressing algorithm to handle big images, so my tests focused mainly first on small images, and then on relatively big images (in the order of MBs).
- My main task in the third sprint was to let the user decide a radius in which he wanted to look for events, and only query from Firebase the events in this radius to avoid

downloading all of them without reason. Obviously, my test focused on two equivalence classes: inside of the radius and outside of the radius. I also made sure that the difference between metres and kilometres was detected by the system (that is an event 500 metres away from the user was displayed, and not confused with an event 500 kilometres away)

Vanessa

setupGoogleSignIng:

- When someone opens the app, the login activity starts asking if you would like to sign-in with Google. This was done using Google OAuthentication token that is to be returned when the user chooses a Google email. The intent is acquired by called the Google API and Google Sign In options. The output of this is the user's email otherwise the user can not log-in.
- A google OAuth client ID should also be set under the Google console for users to login.

onSignInGooglePressed:

- Intent with Google would either return the user to the correct activity or two different errors: DEVELOPER_ERROR.
- Developer error is due to having each phone need a key acquired by signing the application with a certificate since we are not using a release certificate with the app.

loginWithGoogle:

- For Firebase to authentication Google users, the application had access the token given the Google OAuth. This was done to get provider data such as their displayName to be used later on.
- If a user can not login with the only provider given, there is no way for us to authenticate them.

getImages:

- This adds bitmap images to an ImageView that is used to put selected images into a horizontal gallery.

Case needed to be tested is what position should the image be saved in then on click should fullscreen to the correct position.

Jason

GetFriendlyDate():

- One of my tasks involved writing a function that will take the date and time an event is happening and return a friendly date format (ie. "happening in 5 hours," "happening in 25 minutes," "happening now!").
- There were many corner cases to check for, including: events that already happened, events happening that are that at the exact minute, events that are exactly one hour away. I used a series of if() statements to handle all of the corner cases.

BookmarkEvent()

- Another task was handling how bookmarks would be saved and handled. They get saved into the shared preferences.
- Cases that needed to be tested were: if a user bookmarks an event, will the bookmark icon appear correctly every time the user loads the app. And will bookmarks always appear in the BookmarkActivity list view?