Testing Units of SnakeCatcher

Google Signin:(Jiayuan Liu)
To test this module, I have to create a project in google developer console, get the SHA1 certificate fingerprint from android studio signing report to enable google sign in on developer console. Then get the json configuration file from developer console and download it under app folder of our project. Add dependency to our build.gradle file. I also need to make a UI for login page. A private section for user profile section. And a sign in section. Implement google signin API. The testings I have done to see if Google Signin works would be:
-Login to my gmail account: it would show my id and my photo.
-Sign out my gmail account: it return to the login page and it can also see the account I logged in before.

Motion Detect:(Jiayuan Liu)
To implement this function, I have to create a background service running in its own thread. The background service remembers the time T0 when it has been started, and listen to the acceleration. The service has a method, that can be called from the activity. The method is used to check if someone moved the phone at least 30 seconds ago, to give you time to pick the phone up and check. Calls the Service, which calls it in the Service Task. This tells the activity if the phone was moved; update UI accordingly. To test this feature I've done:
-Install the APK to my android device and start it.
-Put it down for without moving it then wait 30s and check the message show on the UI
-Start it again and move it after put it down

Android Camera and Recording:(Jinxuan Jiang)
To test this module, I need to make sure every functions work correctly. For the functions to take picture, I need to test *takePicture()* in *Camera* class. After I call *takePicture(), onPictureTaken()* will be triggered, then I can get the target file by calling *getOutputMediaFile().* When *onPictureTaken()* is triggered, preview function will stop immediately and we need to add *camera.startPreview()* to keep the camera previewing. For the functions to record, I need to instantiate *MediaRecorder*, and then *startRecording()* will call *prepareVideoRecorder()* to test whether the function have access to *hardware.camera* (if not, quit). If connecting to the camera correctly, function *start()* will begin to record. *stopRecording()* will call *stop()* to stop recording, and then

close-out by calling *releaseMediaRecorder()*. Function *isRecording()* is to determine whether the device is recording or not.
To test this feature I have done:

1. Install the APK to my android device and then start it.
2. Click on Recording button, Camera button, Start button, Stop Button and Preview Button to check each function.

Firebase Database: (Rishita Roy and Ky Kham Tiet)
Storage -
To test the storage I had to make sure I was able to create new data, in this case the time and date of when the user signs in and have to show up under their UID in a proper format. Testing this was a lot through the firebase console, storing dummy data and checking if it shows up properly in the console. Once it did and the hierarchy of the tables were satisfactory the storage function deemed done.
Pulling the data to RecyclerView -
The process of writing this module had a lot of small tests in between. We traced where the data would be passed through and put multiple logs to see if the data has been passed correctly or at all sometimes. To fully test each function, we had an account designated to be tested which was cleared off data at first and then we started populating (see storing tests) the database with each log in. Once we saw data in the console, we checked if we were able to pull this data into a query and using the query objects organizing them according to our *HistoryItem.class* and setting it to options so that we can create the *FirebaseRecyclerAdapter*. We tested some functions in the adapter such as *onDataChanged()* to see when each function gets called. We also checked the *bind* function in the same way (using logs). After everything ran properly we logged in with a new account and followed the log files closely and watched for each new item to pop up in the database console and then in the view as well.

General Testing of Various functions while error handling (Rishita Roy and Ky Kham Tiet)
When we were fixing up the code and making the app runnable, we had to test various modules individually as well as within the app. To do so the general process was to run the app whenever there was a change, know the expected outcome that we want to see either directly in the app (frontend), in the log (testing) or in the consoles (database), and check the log for errors. If there were no error messages that we could build of off and the outcome was unexpected, we would go back and look at how data would be passed around as that was the biggest concern. We would then proceed to add log functions in each module tested and check where the data gets lost and work from

there. The modules were deemed “done” once we were able to randomly input data and the outcome would still be as expected.