Final Report: App Opener

Mobile App Engineering and User Interface

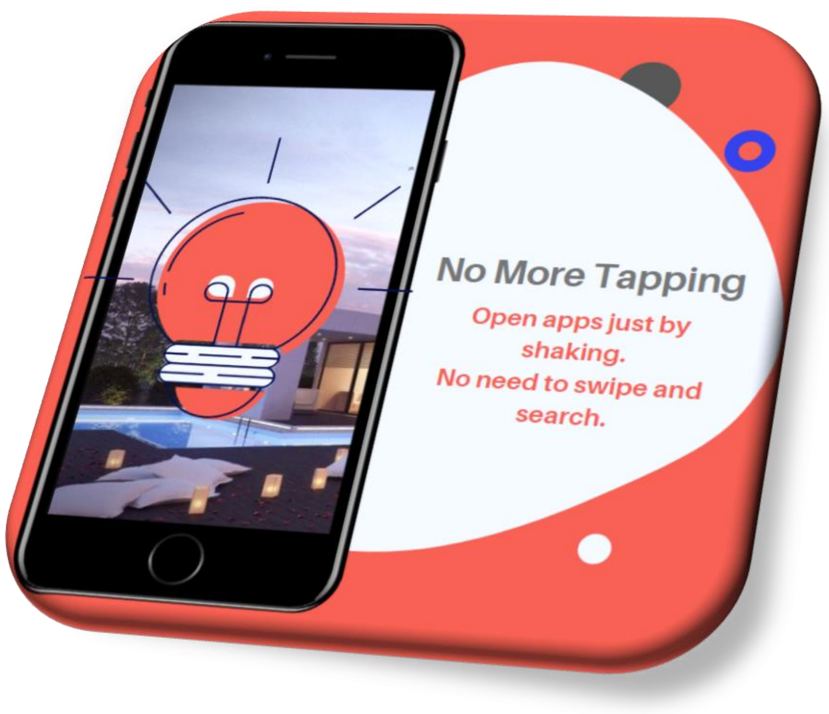
Students:

Andy Lee - ahl88@scarletmail.rutgers.edu

Parthkumar Patel - parth.14@rutgers.edu

Robert Riso - rar267@scarletmail.rutgers.edu

Aryeh Ness - akn45@scarletmail.rutgers.edu



Motivation – What is the project good for?

- The main motivation to create this project was to make it easier for the user to open apps without swiping through screens and searching for a particular app.
- Many shops and fast food restaurants give discounts if you use their app instead of ordering through the cashier. This results in way too many apps on our phones and searching for them takes about 20 seconds.
- Since there is no swiping or searching required this app can also save a lot of time.

Our Solution to solve the problem

- We implemented the main objective by using sensors on our phones that can detect the shake and open a selected app right away.
- The sensor that we chose is the acceleration sensor which will measure the shake of the phone and then open the desired application for the user.
- The location-based part allows user to choose an application and then assign two locations that they frequently visit. This allows our app to open the user selected app when the user is in close proximity the selected location.
- Our features implemented allow us to solve most of the features we wanted to create using our motivation.

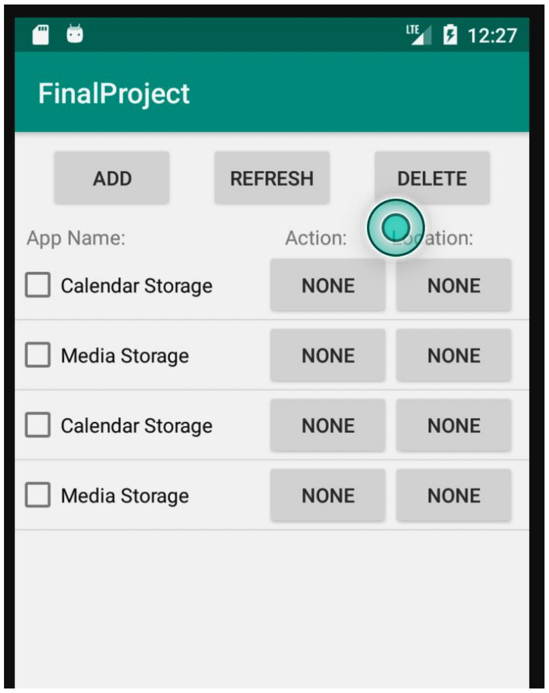
How does it work – key components and their interaction

Key Components of our Project:

- Shaking to open the app.
- Rotating the phone to open the app.
- Location based app opening for the user.
- Database that handles all the information entered by the user and used for the app.

Describe the user interface of your system and how it makes the most common tasks easy and quick to accomplish:

- Our app consists of the following buttons, better displayed in the screenshot below.
  - ADD
    - This allows the user to add the application they desire to add to the app.
  - REFRESH
    - Refreshes the list of events after an app is added.
  - DELETE
    - Allows the user to delete an application with the action and location related to it.
- Once a user adds an application to the list there is a list view for the apps that are added and this list view has a check box and two buttons next to each app
  - CHECKBOX
    - Select and delete the app
  - ACTION (button listed under action column)
    - Allows the user to select from the three following options
      - Shake
      - Rotate
      - None
  - LOCATION (button listed under location column)
    - When pressed takes the use to a google maps API to select the location they desire on the map.
      - The user could also search the location using the search bar that is provide. Example – Name of the store such as Starbucks or BestBuy.



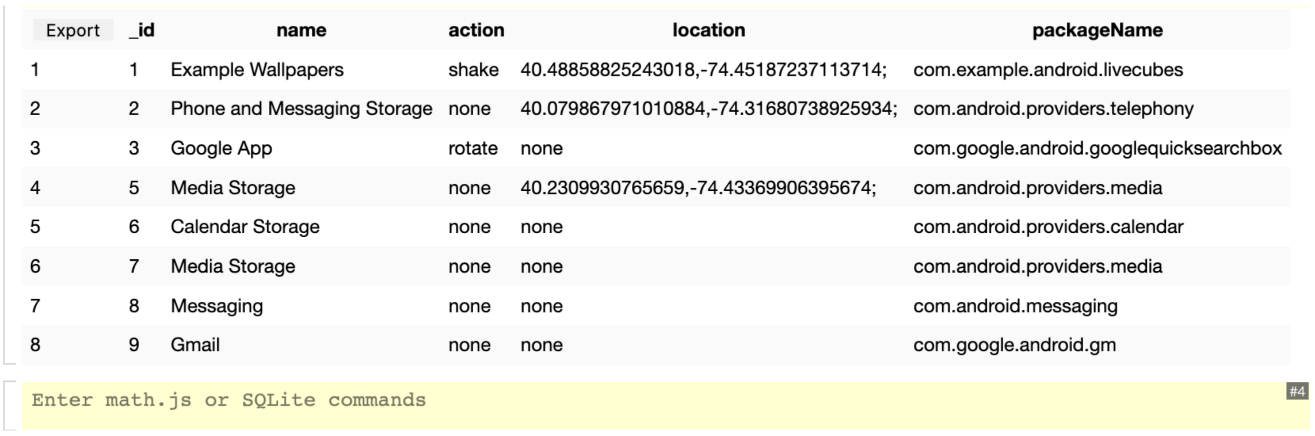
All the buttons are showed in action in this screenshot.

Describe the key components of your systems (e.g., activities, services, remote database) and how they interact:

<u>Key Components of our Project:</u>

- Shaking to open the app.
  - This function is implemented using the hardware libraries for the sensors. Our app uses Accelerometer to measure the shake by the user. Detecting the shake is completed by ShakeDetector.java this is then connected to ShakeService to link it with the interface and the database for the app.
- Rotating the phone to open the app.
  - Rotation to open the app is implemented in ShakeService.java this is also connected to the database eventDBHelper.java

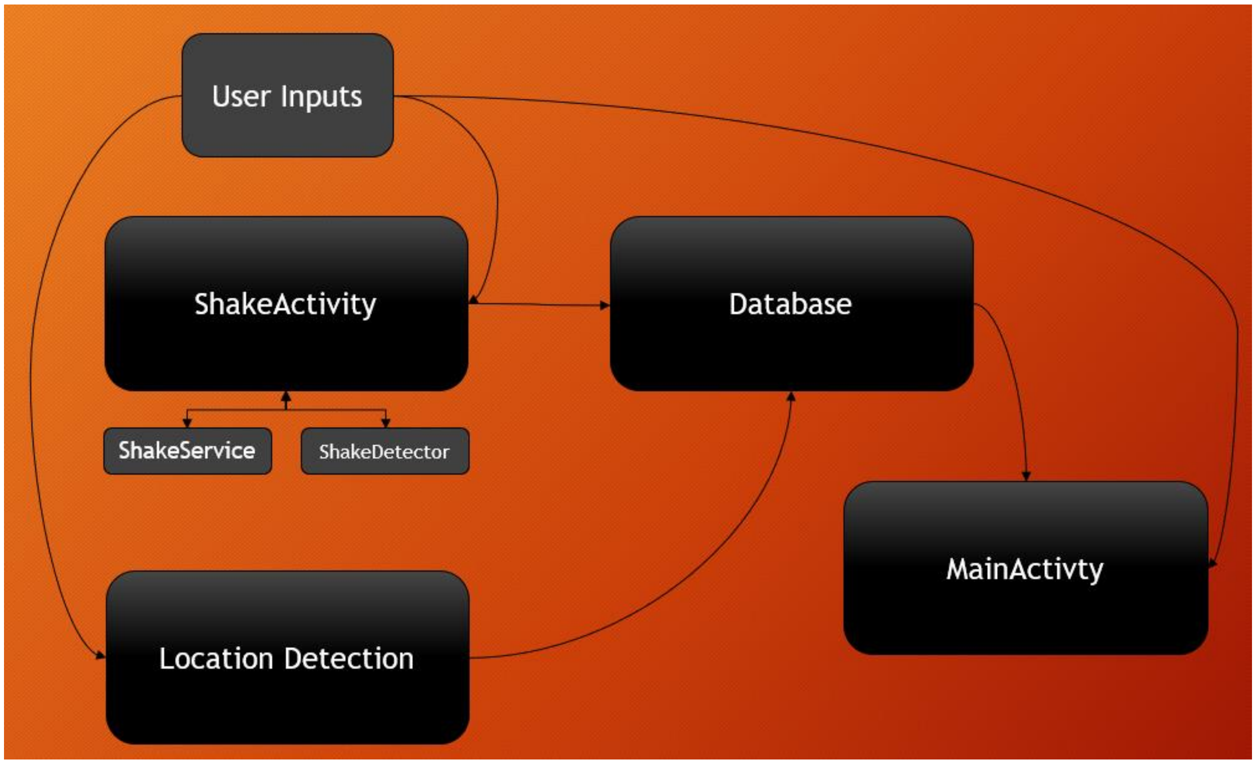
- Location based app opening for the user.
    - For getting the location we have used Google Maps API. When user selects location to be set it takes the user to a map fragment and a search bar that allows the user to select a location to add to the database.
    - The database takes locations from the user and then we check that location to the current location. Once the user is close the app opens.
- Database set up:
    - Our database is set up in the following way as shown below.



Export	_id	name	action	location	packageName
1	1	Example Wallpapers	shake	40.48858825243018,-74.45187237113714;	com.example.android.livecubes
2	2	Phone and Messaging Storage	none	40.079867971010884,-74.31680738925934;	com.android.providers.telephony
3	3	Google App	rotate	none	com.google.android.googlequicksearchbox
4	5	Media Storage	none	40.2309930765659,-74.43369906395674;	com.android.providers.media
5	6	Calendar Storage	none	none	com.android.providers.calendar
6	7	Media Storage	none	none	com.android.providers.media
7	8	Messaging	none	none	com.android.messaging
8	9	Gmail	none	none	com.google.android.gm

```
Enter math.js or SQLite commands
```

    - It consists of the following columns:
        - ID – Index for the apps
        - Name – Name of the app
        - Action – Has three options as mentioned before
        - Location – Location user inputs for the app to open.
        - packageName – To extract the app packages.



Describe the implementation of selected key components so that a knowledgeable app engineer could create a similar app with the provided information:

- Shaking to open the app.
  - **FILL THIS**
- Rotating the phone to open the app.
  - **FILL THIS**
- Location based app opening for the user.
  - **FILL THIS**
- Database that handles all the information entered by the user and used for the app.
  - **FILL THIS**

How did you organize work in your team? What concepts from class did you apply?

The work in our group was divided in the following way:

- Andy Lee was responsible for Database set up.
- Aryeh Ness was responsible for Shaking and Rotation.
- Robert Riso was responsible for getting the location from the user and saving it.
- Me (Parthkumar Patel) was responsible for getting the current location and checking if the user was close by the user entered location and also for the report.

Concepts applied from class:

- Setting up the database, we had hard time calling the functions, but the class did get us the basics of the database.
- Location and using Google API was something that helped us in the project, but we still faced complication. Specially me (Parth) getting the locations to compare to the database.

Authors for the Java Class

- Andy Lee set up the Database using;
  - EventContract.java
  - eventDBHelper.java
- Aryeh Ness set up Shaking and Rotation using;
  - ShakeDetector.java
  - ShakeService.java
  - ShakeActivity.java
- Robert Riso got the location from the user and saving it using.
  - MapsActivity2.kt

- Me (Parthkumar Patel) was responsible for getting the current location and checking if the user was close by the user entered location.
    - My complied code had errors as I was only able to get the users present location and was having difficulties comparing it with the database values. The app would crash with my code other than the current location part.
    - Robert Riso was then able to help me and implement my functions straight into his Kotlin code which got the app to work.
    - Functions implemented in MapsActivity2.kt