**Project Summary:**

We are proposing to create an Android application for locating parks and recreation areas based on query by location. After parks are found, users may refine the search by user ratings or by park features (e.g. playground, basketball hoop, etc.). Data is created by the end user. There will be a quick report function to add a park to the database and an option for users to rate the park, report additional features, add comments or share via social media.

As parks are not businesses, there has been little incentive for an app to be developed of this nature (one may exist, I haven’t done exhaustive research). Information about local parks across the country is not consolidated, and the end goal is to enable people to find these free community resources in a global format. I foresee this being of benefit to local municipalities and to non-profit youth fitness programs.

**Project Requirements:**

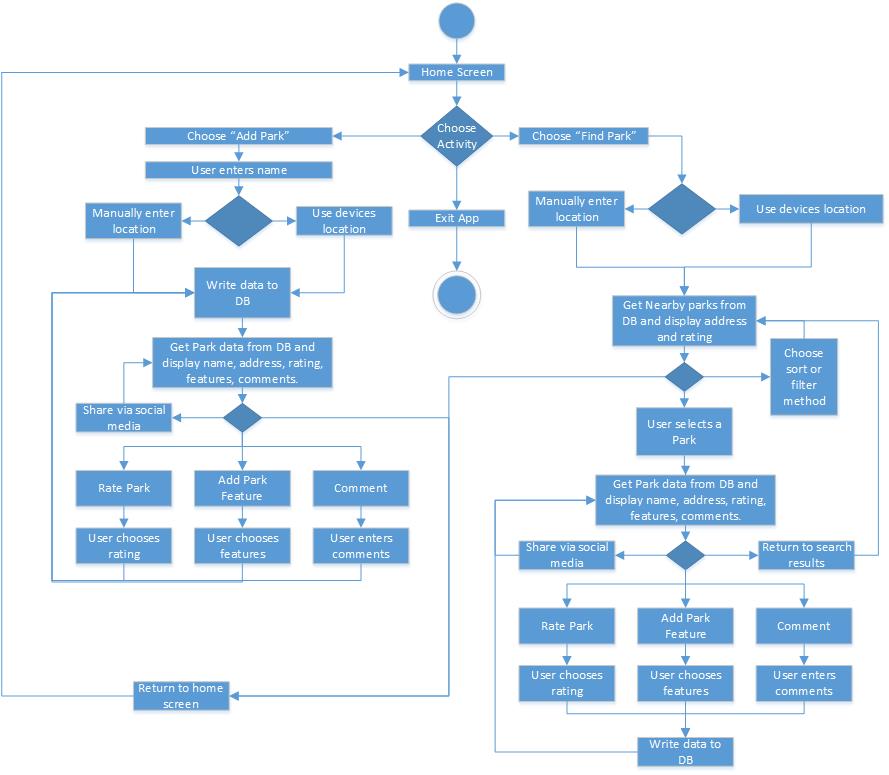
Main goals and associated responsibilities:

1. Create independent database of local park resources.
   1. Gather park details from the end user
   2. Find location of the end users device via cell network or GPS.
   3. Connect to persistent storage.
   4. Write data to storage
2. Provide public means to query this database.
   1. Connect to database
   2. Retrieve a list of parks based on proximity to a specified location
   3. Sort and/or filter this data
   4. Display park details of the query results to the end user
3. Allow user to easily share the data. Provide simple methods to send the park details via common communication methods.
   1. Provide email interface
   2. Provide Facebook interface
   3. Provide other interfaces as necessary.

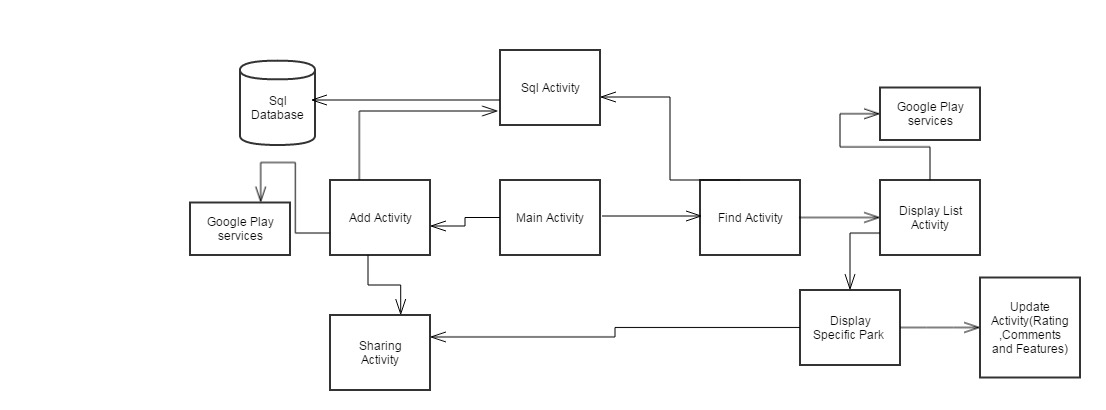
**Users and Tasks:**

* A First user searching for a park using his location, he can select find park activity and select the address with the option of current location or enter the address which will give him the details of all the parks near his location. The Main Activity will give the option of the finding the activity and the find activity will look for the database based in the location if it finds any matches it will return the data, user can select for a specific park and look for the features, rating and comments. When the user clicks on the specific park the Display specific park activity will display all the data of that particular park.
* (Users and Tasks cont’d) Another user will be the one who finds a new park and add it FindAPark database. He can select the add activity and then he will be able to enter the name, Location, rating, comments and Features. He can also share the park on the social media this is done by the sharing Activity. In Add Activity once the user clicks on the Add park the data will be saved in the database for further usage.

**Activity Diagram**:

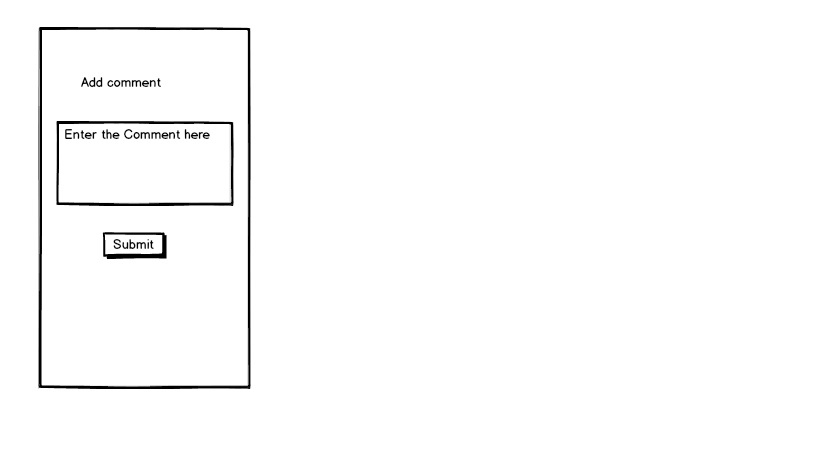
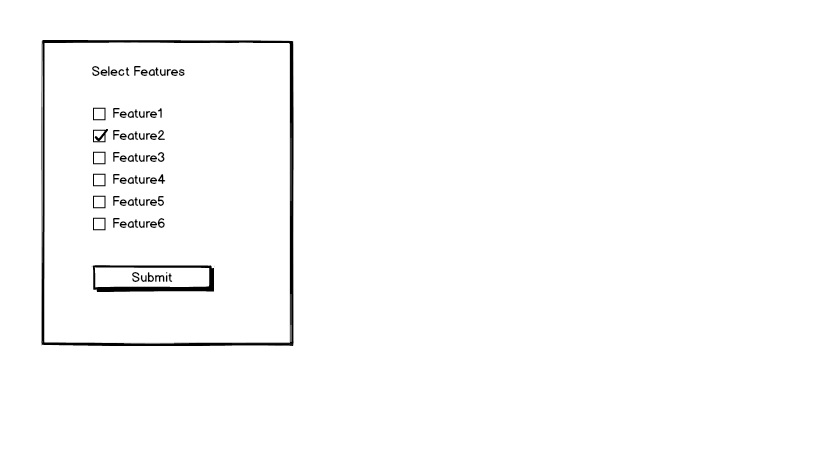
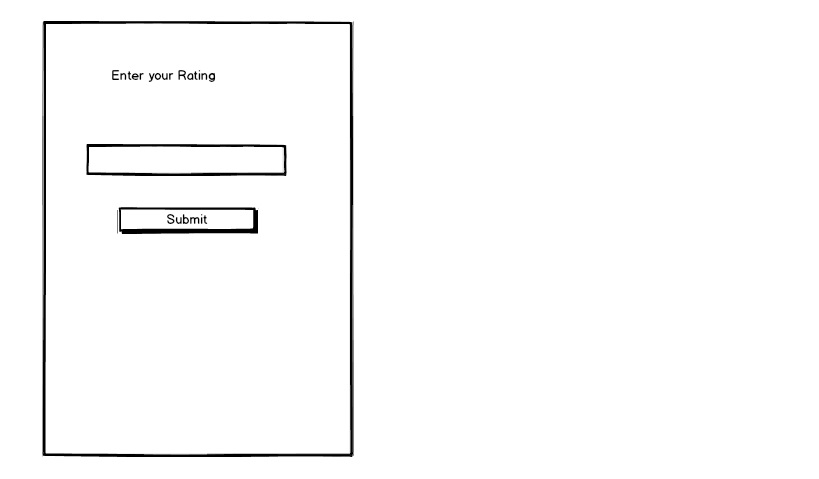


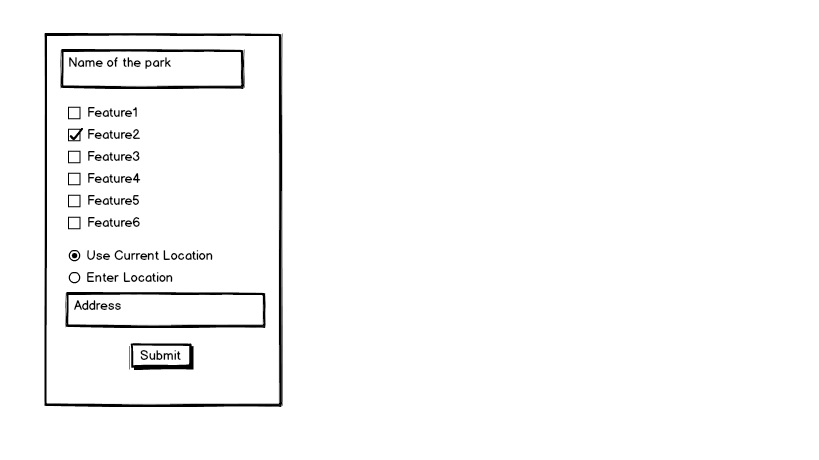
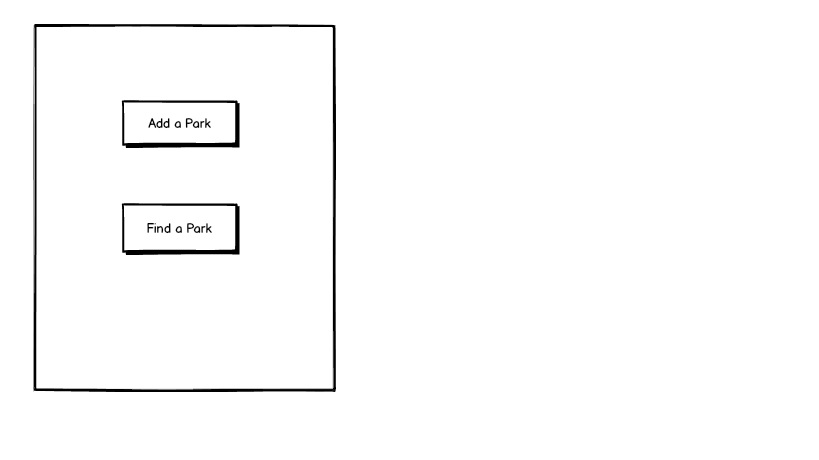
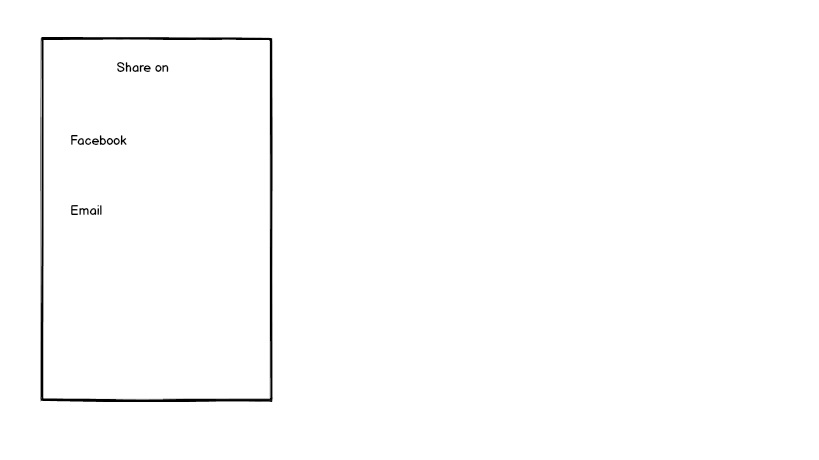
**Architecture Diagram**:



**Data Storage:**

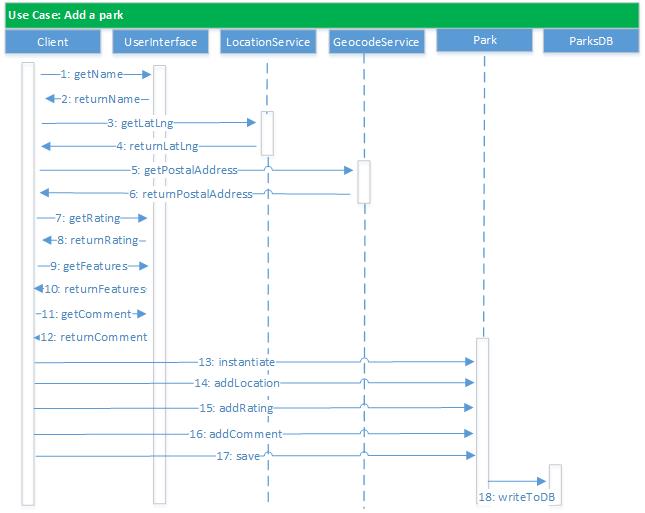
* The application uses the Android in build SQLite database to store the data and to retrieve the data up on the user request. The data will be stored in the tables and it is retrieved using general database queries. We use the SQl Activity to access the database, Add activity, and find park activity will request the sql activity for data.

**UI Mockups:**



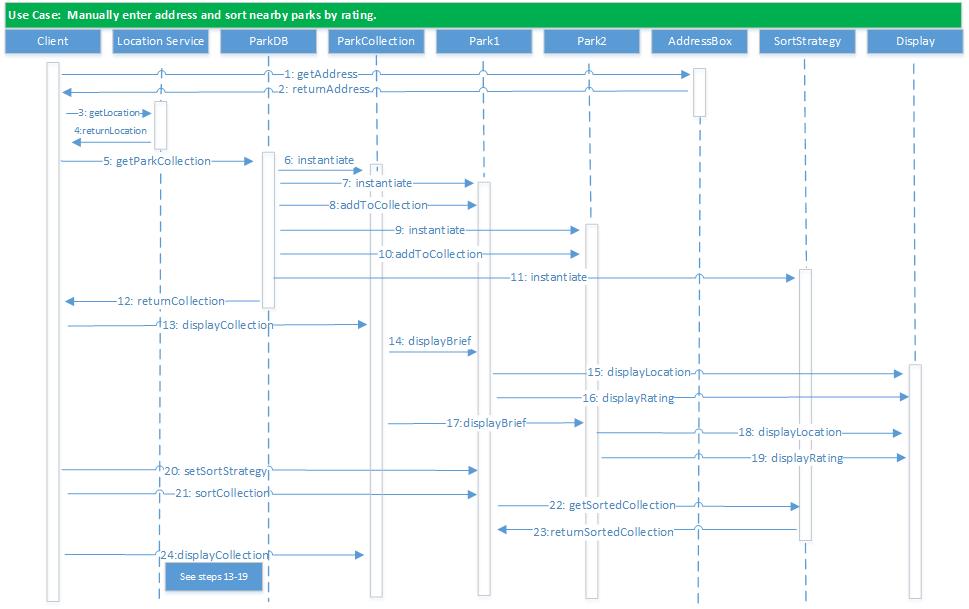
**User Interactions:**

1. **Adding a park** - The user will navigate to the “Add a park” screen from the Home screen. There will be a series of prompts, Name and Location will be required and the rest may be left blank if desired. If the user uses the option for “Use device location”, then the client will use Android Location Services to retrieve the Latitude and Longitude, and then query the Google Maps Geocoding service to discover the Postal Address. When the user hits “Save”, a Park object will be created using the criteria provided. In the use case below, the end user opted to use the device location and to leave features blank. After the park object is created, it will be sent to the database interface, which will upload the required data using SQLite.



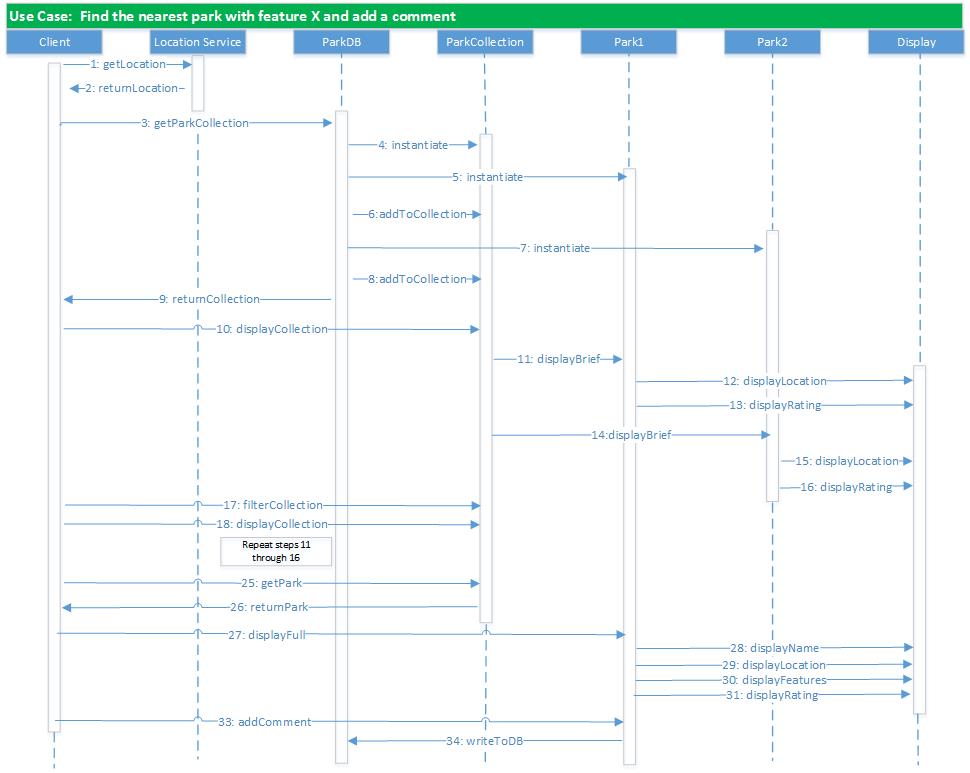
**2. User searches by manually entering address and sorts by rating.**

From the Home screen, the user will select Find a park. They will be prompted for a postal address. After entering the address and hitting submit, the latitude and longitude will found using Google Maps Geocode. The client will then query the SQLite database using latitude and longitude. Some default square shape will be used for the query (i.e. return a park if the latitude and longitude are both within +/- 5 miles of the queried LatLng). The database interface will create a ParkCollection and substantiate it with Parks based on the results of the SQLite query. The client will display the parks on the Android screen as a list (later the display method will be replaced with a GoogleMapsActivity). The user will then select Sort by Rating, which is sent to a SortStrategy object. The SortStrategy object will return a sorted collection and display it.



**3. Find a park with a specified feature and add a comment**

The client will get the current LatLng from Android Location Services. The client will ask the database interface to run a SQLite query based on the current LatLng. The DB interface will create a ParkCollection and substantiate it. The client will ask the ParkCollection to display itself. The user will select a filter method and the filtered collection will be displayed. The user will select a park, which will tell the client to display the full details for that park. The user will select “Add a comment” and provide the comment. The client will update the Park object and send it to the DB interface, which will write it to the database.



**Class Diagram:** 