RUThere

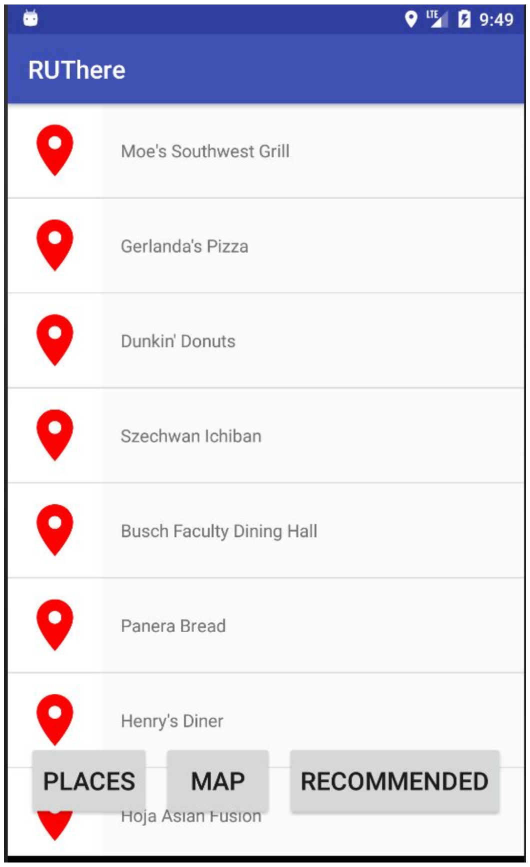
Rutgers University
Electrical and Computer Engineer
Mobile Apps for Engineers, Fall 2018
Final Project
Leonardo Roman (project manager and developer)
Zhaoxiang Liu (developer)
Christopher Salandra (software tester and administrator)
Zachary Joseph

Introduction

Our final project was designed to use multiple RESTful web services such as Google, Yelp and one of our own. The application uses device current location to find nearby restaurants around the area and then display them on a map by using google places API. Our designed web-service application will do computation that can not be hold in a regular mobile device, such as querying/cleaning data requested from Yelp web-services. Finally, our application intends to record user's activity by checking in users near the restaurants displayed by the application and constantly updating the database to have an actual count of people at each location.

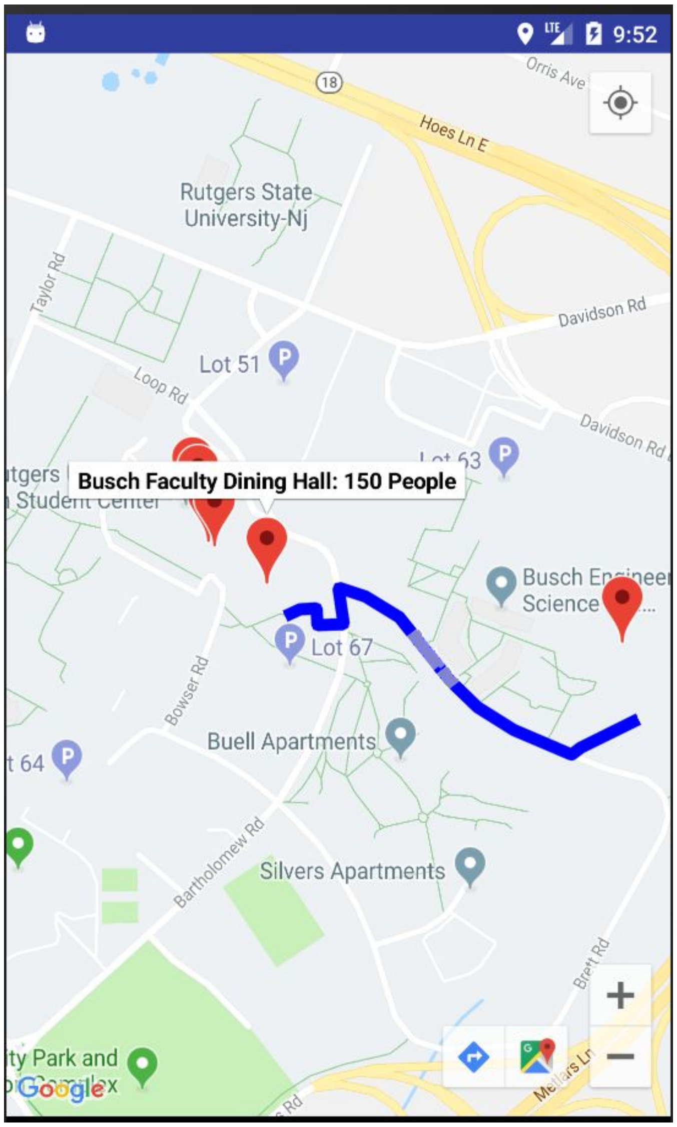
Motivation

The motivation behind this project is to make use of the mobile tracking mechanisms. Data is big part of market growth and generating data is no simple task. The idea is to use a tradeoff between application usage and device permissions. A user wants to use our application and we want the data that this generates. We will do this, simply by recording user's activity in a database system, using their device ID in order to determine which restaurants are frequently or even repeatedly visited the most. We believe that by showing a number of current people in a restaurant, at a particular time, might influence the user's decision to visit that location. For example, a user might go to a restaurant that is less crowded at current time or might still visit the restaurant even if is crowded which makes data analysis more interesting.



Our Solution

RUThere gives the user a friendly and easy UI which displays nearby restaurants, a map with marked locations and a recommendation of high rated restaurants around the area. Our application communicates with three different RESTFul services. First, we request a map and places by using the Google maps/places API. The API gives us the data we need in order to display the nearby restaurants to the user. Second, we use the Yelp API in order to find the ratings and reviews of the restaurants, if exists. However, we make this API call using a "backend API" in order to reduce overhead in our mobile application. Yelp does not return reviews of any business just by giving a name. First we need to get the type of business data, in our case restaurants, and then retrieve the location ID in order to perform another query to get the reviews. This seemed to much work for our mobile application so we decided to create a web server application using python and the flask library as our back end server. Once all data from the Yelp service is clean and formatted, our python webserver returns this data to our mobile device. Finally, we record the user's location in our database if the user is within 5 meters of any displayed location, in our application. We are constantly checking if the user still within the radius and once they are not we update our database. The database stores name of location, device id, coordinates, time, date and a boolean flag to be updated. We use device id, time and date as primary keys since we are also interested if the same user goes back to the same restaurant at any other given time. Once a user is checked in, the boolean flag is set to true and once we detect the user left, the boolean flag is set to false. This simple database gives us the power to query a current count of people, using our application, for each restaurant and display it on the map. As mentioned before, we want to influence the user's decision by displaying a current count for each restaurant.



Architecture

As you can see in the diagram, the application works in a distributed manner by requesting service to three different RESTful APIs. The data is process performing asynchronous tasks, using Retrofit library, in order to not block the main UI thread. The application requests data to the RESTFul API, this returns a Json format of the data and then Retrofit parses this into Java readable data. Retrofit does heavy work that is required to run in the background, such work includes creating instance of classes and assigning their fields with the data gathered from the RESTFul service.

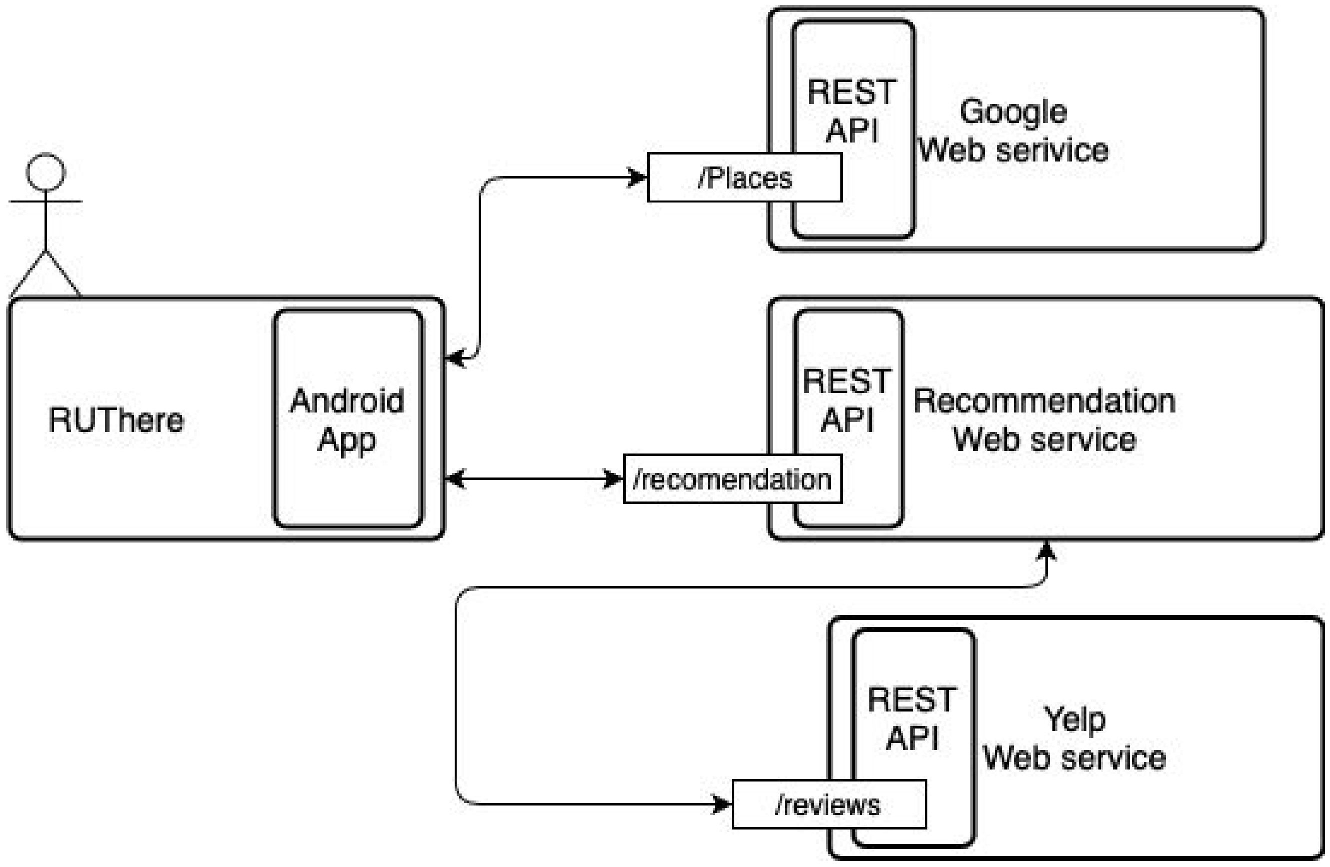


Figure 1. Application architecture

The sequence diagram shows a more detailed timeline of how the application requests for service to the different APIs. Here we can see how every system works together in order to perform a task.

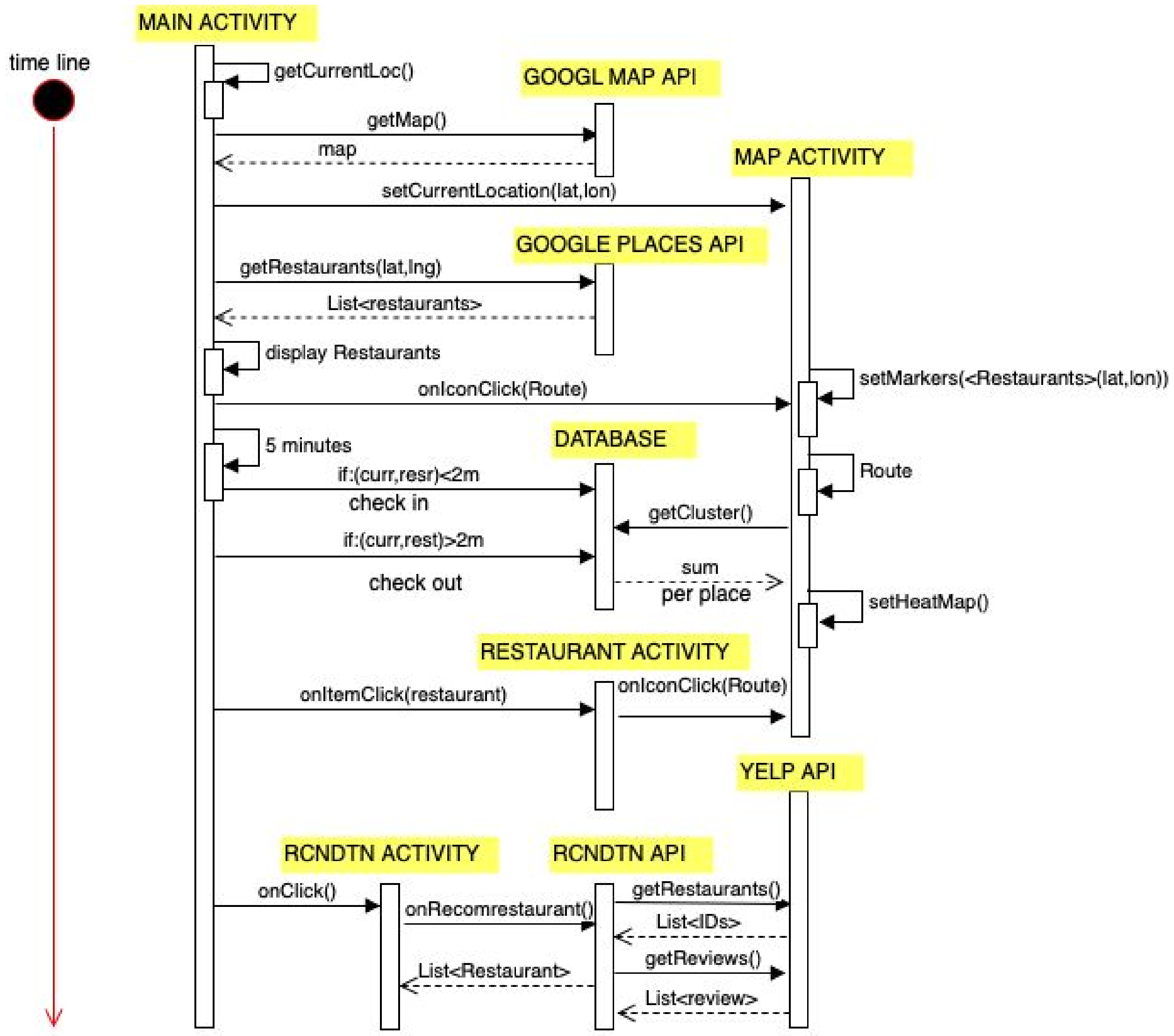


Figure 2. Flow diagram.

System Requirements

1. Android API 23
2. Google Play services 16.0.0
3. Gson 2.8.5
4. Retrofit2: Retrofit 2.5.0 and converter-gson 2.5.0
5. Google API key
6. Yelp API key
7. Python API server running

UI requirements

Activity 1

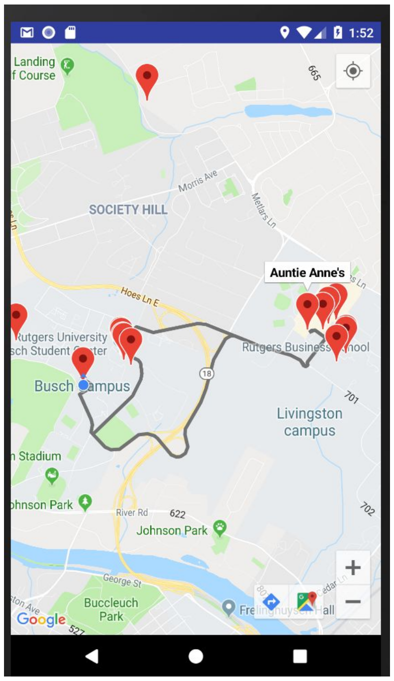
1. Display nearest places given current location (Places Button)
2. Display a map with nearest places marked with location markers (Map button)
3. Give a route/direction for the restaurant as requested (Red Icon Button)
4. Display a daily recommended restaurant (recommended Button)
5. Automatically check in users after some time (5-min)
6. Save check in a database using some sort of mobile id and a boolean flag that states if user still at location (phone number, device ID, etc if possible)
7. Set the boolean flag to false after user leaves place (using some function check every minute for user current location)

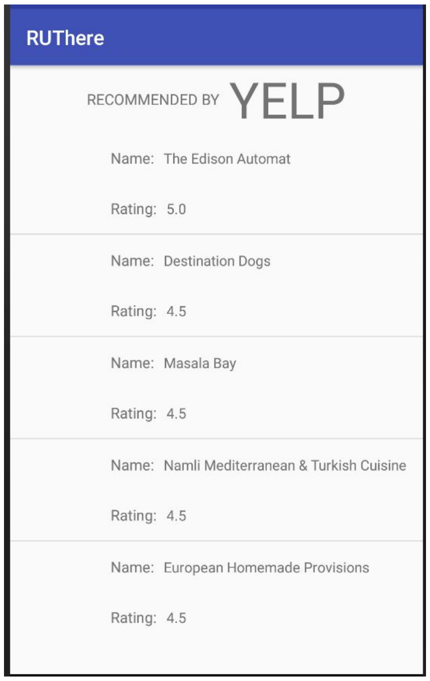
Activity 2

1. Display a heat map of places with respect to place crowd volume by using database queries (e.g. red most crowded, blue less crowded or number of people)
2. Route directions on marker click (should be using same logic as in activity 1)

Activity 3

1. Route direction of location
2. Display location information (name, number, address, reviews)
3. Display number of location and be able to call by clicking the number
4. Display reviews by making API call to Yelp webservice
5. Make reservations
6. Calculate a total review





RUThere

RECOMMENDED BY YELP

Name: The Edison Automat

Rating: 5.0

Name: Destination Dogs

Rating: 4.5

Name: Masala Bay

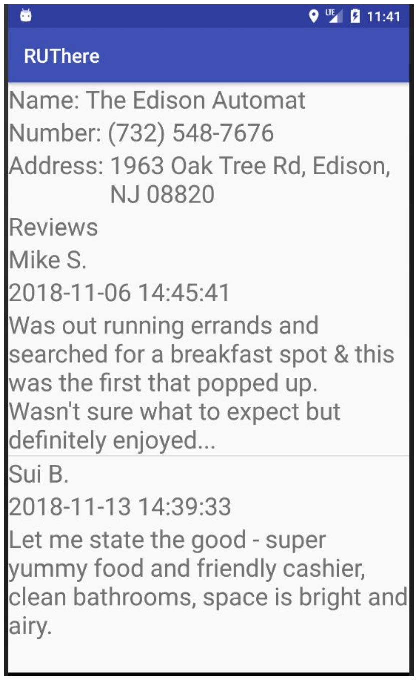
Rating: 4.5

Name: Namli Mediterranean & Turkish Cuisine

Rating: 4.5

Name: European Homemade Provisions

Rating: 4.5



11:41

RUThere

Name: The Edison Automat
Number: (732) 548-7676
Address: 1963 Oak Tree Rd, Edison, NJ 08820
Reviews
Mike S.
2018-11-06 14:45:41
Was out running errands and searched for a breakfast spot & this was the first that popped up. Wasn't sure what to expect but definitely enjoyed...
Sui B.
2018-11-13 14:39:33
Let me state the good - super yummy food and friendly cashier, clean bathrooms, space is bright and airy.

Project Management

The development of the project was conducted by applying every concept learned during the period of this course and some other concepts learned in other courses such as Distributed Systems, Software Engineer and Cloud Computer. We first designed a paper lo-fi prototype in order to get an idea of what the application would look like. Then we constructed a skeleton of the project by creating all required activities and model classes. And finally shared the prototype to the group members for contribution. Timing was an issue and we hoped to accomplish more. We originally had planned to display a heat map instead of just a plain count of people. We also planned to use machine learning to further analyse Yelp data and make recommendations. The team constantly communicated via emails. There were two developer and one software tester. As the developers pushed a new commit the tester was in charge to find bugs so the developers can fix it. We all played a important role in the development of the application. The commits were done using google docs instead of any web-based hosting service for version control such as Github.

Work Distribution (please refer to the index to identify the libraries)

Leonardo Roman 14/23 files + python RESTful API
Zhaoxiang Liu 5/23 files
Christopher Salandra 4/23 files + powerpoint slides + software testing
Zachary Joseph final report

Appendix

Packages and java files (6 packs and 23 files)

1. adapter **(Java classes by Leonardo Roman)**
   1.1. RestaurantListAdapter
   1.2. ReviewAdapter
   1.3. YelpAdapter
2. api **(Interfaces by Leonardo Roman)**
   2.1. PlacesAPI
   2.2. YelpAPI
3. database **(Java class by Leonardo Roman)**
   3.1. DataBaseManager
4. directionhelper **(Java classes by Zhaoxiang Liu)**
   4.1. DataParser
   4.2. FetchURL
   4.3. PointsParser
   4.4. TaskLoaderCallBack
   4.5. PolyLineData
5. pojo **(Java classes by Leonardo Roman)**
   5.1. Coordinates
   5.2. Geometry
   5.3. Location
   5.4. Place
   5.5. Places
   5.6. Restaurants
   5.7. Restaurant
   5.8. Reviews
6. activities **(activities by Christopher Salandra)**
   6.1. MainActivity
   6.2. MapActivity
   6.3. RestaurantActivity
   6.4. YelpRecomendationActivity