# Parking Recommendation System

### Group Members

| Name | Group ID |
|---|---|
| Jay Dhammendra Solanki | jds797 |
| Aniket Krishna Bhandarkar | akb501 |
| Yash Devshibhai Balar | ydb219 |

### Introduction

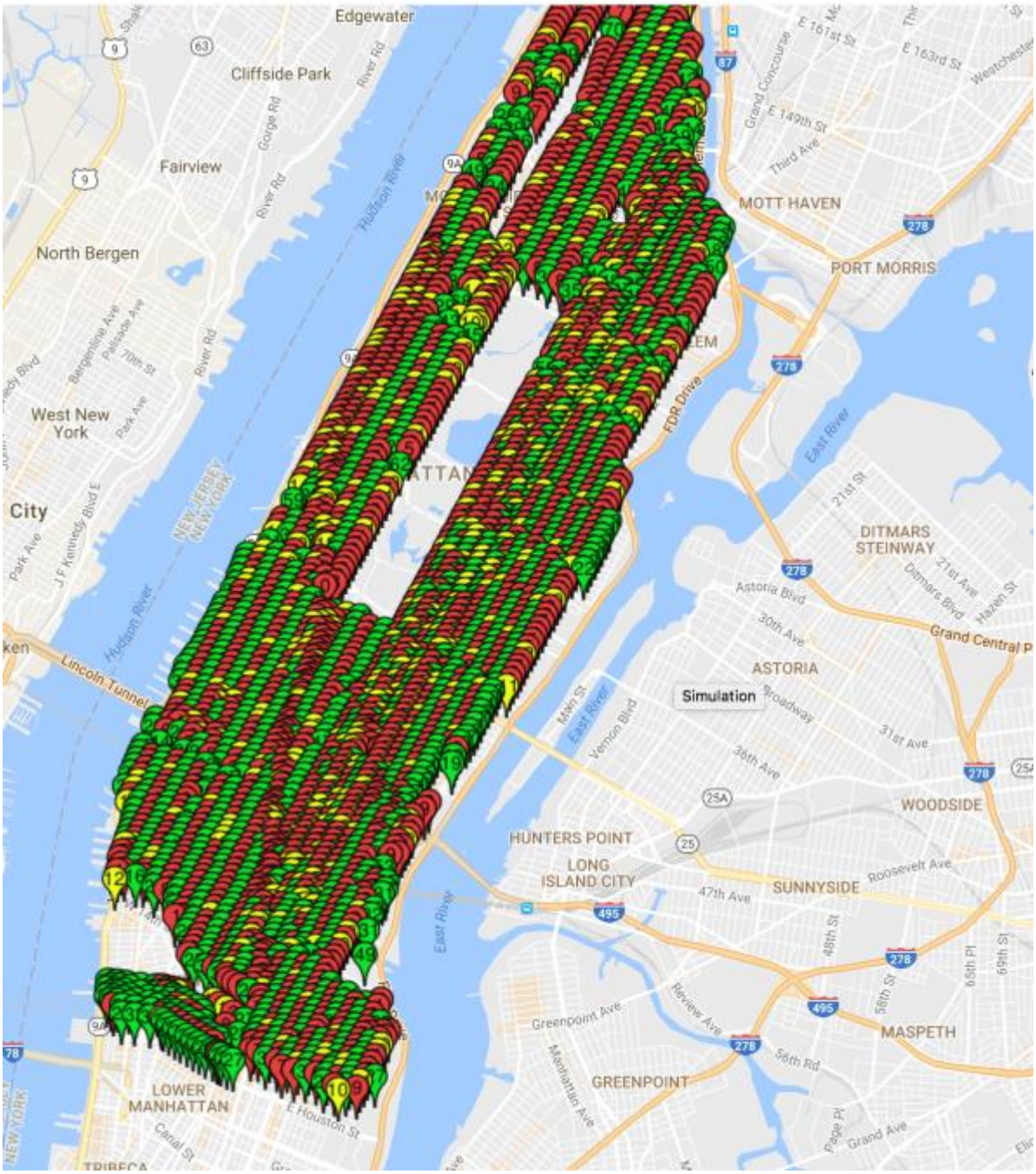
Our goal was to build a Parking Recommendation System which helps to find and notify about available parking spots in Manhattan. We created an Android application and used Amazon Web Services for deployment purpose.

#### Links

- Github: https://github.com/jaydsolanki/CloudProject
- Video: https://www.youtube.com/watch?v=l9raqRGotEg

## Data

- We wanted a dataset in which we get mid points of all the availale blocks in Manhattan.
- To do so we created a small API to mark points in a Manhattan map using Google Maps API to find (latitude, longitude) coordinates and give number of parking spots in that block.
- Using this approach we created a complete dataset of Parking Spots in Manhattan.

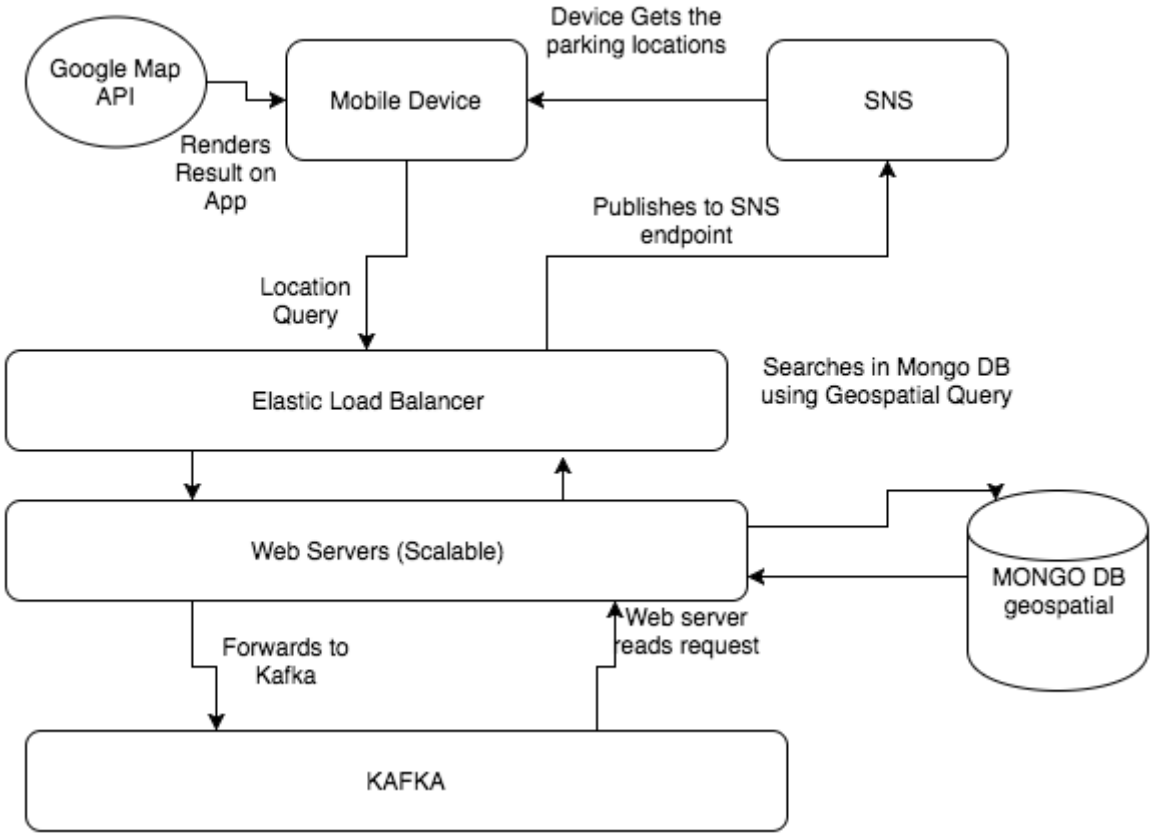



**Technology**

- The UI was made built for Android devices.
- The backend was written in Python facilitated by Django framework hosted using AWS Elastic Beanstalk.
- We use KAFKA queue to handle incoming user requests.

- The notification service for the system was created using AWS's SNS.

## Architecture




The above diagram explains the system architecture

**Deployment**

- The Web Framework in Django was deployed using Elastic Beanstalk which is a scaling infrastructure.
- The Mongo Instance was deployed using AWS EC2 instance.
- We decoupled system components by implementing a Apache Kafka queue. This queue was hosted by an AWS EC2 instance.
- We used Google Cloud Messaging service (GCM).
- For publishing the push notifications we used AWS Simple Notification service (SNS).

**Flow**

- The user would make a request for parking. The request would contain the (latitude, longitude) of the user location or the destination location with the GCM token which would be used later on for push notification.
- The request would be taken up by the Web Server and would pass that request to the Kafka Queue.
- This enabled decoupling of requests and hence made a Non-Blocking call to the web server.
- Then the Server would read the request from Kafka Queue and find the locations using the Geo Spatial Index in MongoDB.
- This information would be an array of results having Coordinates and number of available parking spots.
- We would then use the GCM code to see if the user is already a SNS subscriber. This was stored in database.
- If he/she wasn't a registered SNS user we would register the device and if he/she is already a member then we would move forward for response.
- The response would be then pushed to the end point device using AWS SNS service.
- On the mobile we wrote the logic to get the response and parse the results and show the locations using a marker in Google Maps API on the device.

**Functionalities**

Registration:

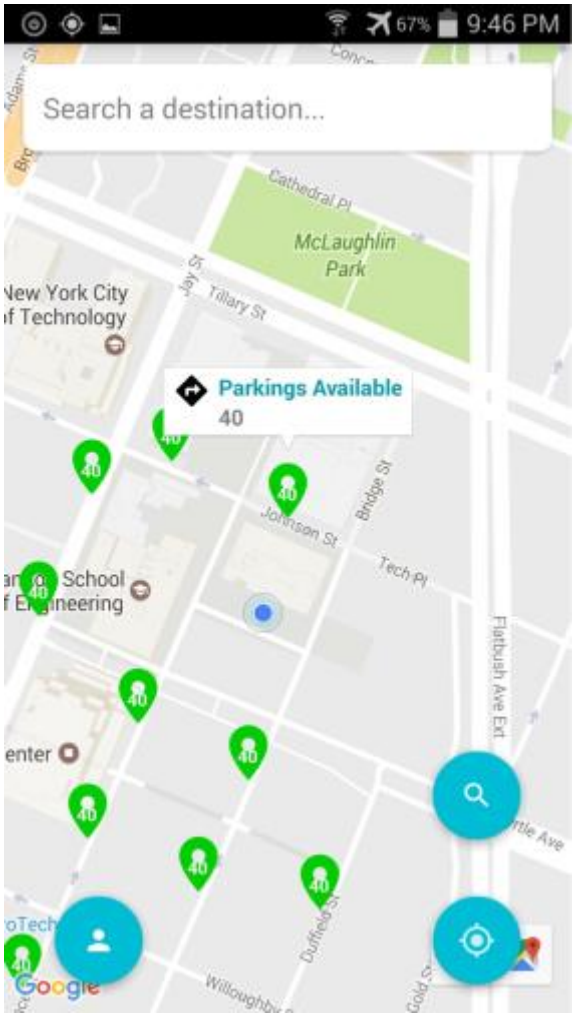
- The user could register to the application using `Full Name`, `Email` and `Password`.
- If the username has already been registered that would be displayed on the App.
- In response to this we would generate a token and send to the deivce for further requests as authentication was done by those tokens for each request.
- This token was refreshed every time user logs out and logs in.

Log in

- Once the user is a registered he/she can log in to the app to use functionalities.
- With every request the token first generated is used. If the user has previously logged out, with this login we would generate a new token and send to the user device.

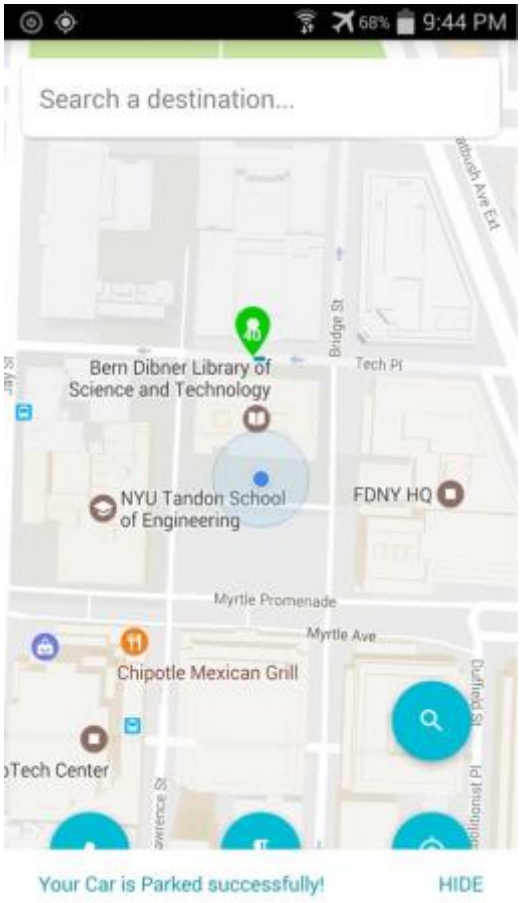
Search For Parking

- This functionality had 2 parts
    - The user could search parking spots near by which would find in a radius of a quarter mile.
    - The user can also search parking on the destination where he/she is going.



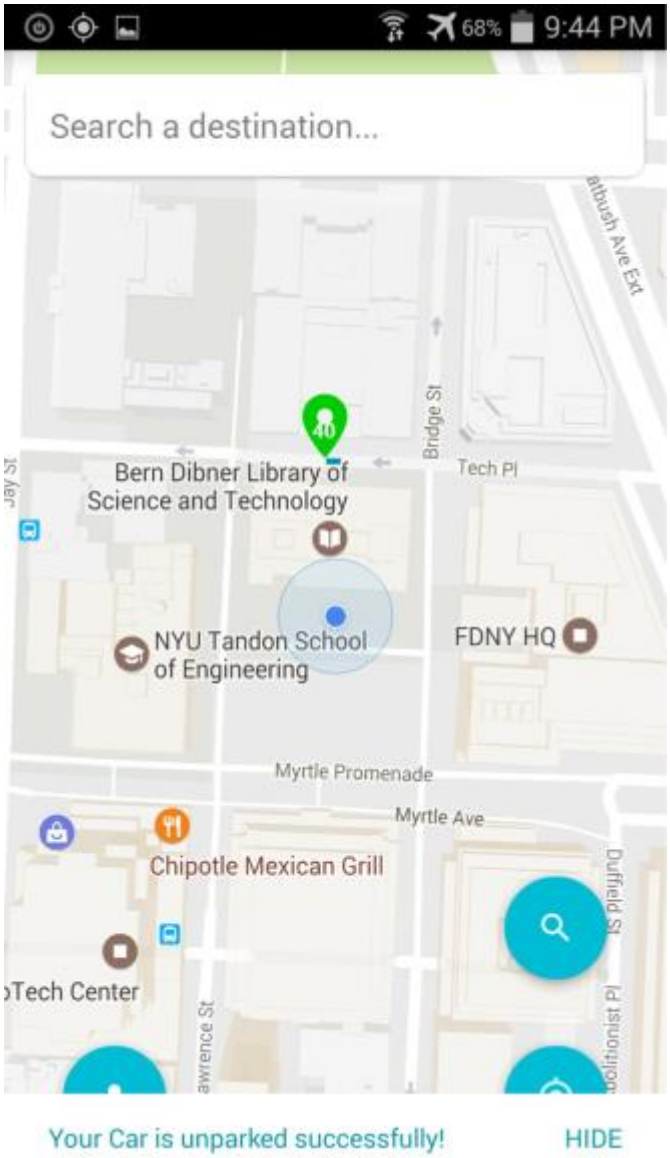

Park Vehicle

- When the user would park the vehicle he/she can press the parking button to notify the app.




Unpark Vehicle

- Once the user returns to the car he/she can unpark the vehicle so that the system can show that place as available for other users.




Authentication

- We used a token based authentication system in which we would create unique tokens every time for the user. So no one can ask for parking spots without registration by making requests to the server.
- For every login we created a new token.
- After selecting a particular parking the android app would route to that place so the user knows where to go.

Web UI

- This was especially created for the developers/management where we can visualize current map with availalbe parking.
- We used this to run simulation for checking what would happen if people would actually use our app for parking. (Check the video for this).
- We also used this UI to collect data and test results quickly. This process helped us to develop logic quickly as it saved deploying the app and using Android device to check it.

**Code structure**

Django

- All the requests from the App were taken as HTTP POST requests with JSON Parameters.
- The response to the APP using the SNS push notifications where the data was embedded as JSON.

Mongo DB

- We created a Wrapper Class in Python to manage all the Mongo Requests.
- This Class handled the client connection, querying and termination.
- This helped us to manage our code better and it was easier to add new functions.

AWS Services

- We created a Wrapper Class for handling requests to AWS services.
- This included creation of Notification Device Endpoint, Adding requests to KAFKA queue, Reading Requests from KAFKA Queue.

**Future Work**

This app can be modified through

- User feedback on the parking slots availability. At times there is no parking available on particular spots where the user ca notify the app.
- We could include a point system that would motivate the user to use the app.

**Conclusion**

We successfully created the Android Application and now plan to work on the aforementioned functionalities we wish to add.