Project Final Report - COMS E6156

Team Batman Versus Superman
Thiyagesh Viswanathan (tv2219) and Anush Ramsurat (ag3630)

Abstract

At most of the universities, especially private schools, 64 percent of full time students live on campus, 19 percent live off campus. Only 17 percent live with their parents. The large student base living in dorms and off campus often live with their peers who can significantly benefit from an application that could save their time by reminding them whenever their friends or roommates are near a location at which they would have wanted to be, For example, if someone wants a pizza for dinner the application would remind him/her whenever his/her roommate is near a pizza store. We conduct various surveys and make important decisions relating to the architecture in the process of building the application. In short we try to mimic a health software engineering practices followed by a corporate development team. The project involves as much study to be performed as the development work involved. We try to prove that such combinations can reap fruitful results in the long run.

1 Introduction

The project spans over multiple stages. In the Survey phase we take a survey on the features to be included in the application and other important factors like front end platform, backend platform, deployment model, which IaaS provider to use and so on. There are two different categories of surveys targeted at consumers (about features and design aspects) and open source developers (about platform specifications, testing processes and other technical suggestions. In the Backend development phase, based on surveys, we confirm over the backend platform to use (among NodeJS, Ruby on Rails, Django) and the cloud vendor(AWS or Google Firebase or Heroku). We then start developing models using right databases.
In the Front end development phase, we create Android or IOS app according to user's suggestions. In the Testing phase, we figure out ways to do location based testing and perform alpha, beta testing.
The first phase involving user surveys and the related responses are mentioned in Section 1. Similarly section 2 is dedicated to developer survey results and the conclusions drawn from it. Section 3 contains detailed description of the backend implementation. Section 3 contains in depth analysis of the location simulation tool and its benefits. In section 5 we discuss in detail about the various challenges encountered while building the application. Details on the tools used and development environment are recorded in Section 6 with a link to the Github repository in subsequent Deliverables section. Section 8 and 9

discusses about the individual contributions of Batman and Superman to the application apart from saving Gotham and Metrocity. Possibilities of future work are included in Section 10.

2 User and Developer Surveys

As part of the project, we carried out surveys among the target consumers (friends and peers from Columbia University) of apps and certain open source developers, in order to gather their opinions on a variety of factors that would contribute to the potential success of the app including desired features, design and tools used in development. Since ours is a location-based app with minimalistic design, we polled users on a variety of factors pertaining to usability and what features they wanted to see in the app. We also quizzed developers on what sort of platform they would prefer to use to build such an app and then modified our own development techniques to match popular opinion.

2.1 User Survey

Coming to the users, we decided to ask questions that related to many key points that decide the popularity of an app and ease of use. Here is a sample of a few questions the users were asked:

1. Do you prefer the app to be on Android or iOS?
2. How helpful do you think this app will be for you?
3. What features do you think we should include in the app?
4. Do you think a 'Do Not Disturb' feature should be added to the app (in order to avoid unnecessary notifications)?
5. Do you think the app should be extended to potentially include an option for group formation?
6. Do you think integration with a payment platform such as PayPal/Venmo/Splitwise would be a good add-on feature?
7. What do you feel about the power consumption of the app?
8. Do you think the simple, minimalistic note-taking kind of design is prudent for this app, in terms of User Interface?
9. How secure and robust are the backend frameworks under consideration (Django, Ruby, Node.js)?

The target consumers were asked these questions with multiple options where they had to choose from one. They were also asked to give additional comments where we had received several suggestions. We received about 19 responses from various students.
We just stuck to two major platforms(Android and IOS) due to time constraints and lack of knowledge for other platforms. While the votes for IOS and Android remained close, Android emerged the winner. Also users wanted minimalist designs with less power consumption and features like Do not disturb, Vemo and

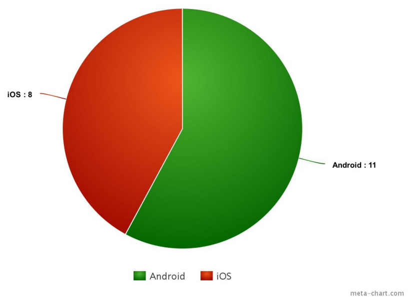

Figure 1: User responses

Splitwise integration, ability for the app to work only when users are walking and not when they are using other means of transport. All the users unanimously agreed on the application being useful for them. Many users wanted the application to be secure and smooth (as smooth as a simple notes application). They don't mind being frequently notified as long as there is a "Do not disturb option" for the application.
Having noted the features that users suggested, we then proceeded to take opinions of a 12 open source developers(students, interns and employees) to zero in on the most suitable platform to satisfy user's requirements. Why we are not considering the simplistic model?
Since we are using API from YELP, we don't need a server as YELP provides libraries for android. The only other thing needed is a common database to sync requests of all users in the group. However since users preferred a minimalistic lightweight app with less power consumption, we realized the only solution would be to move most of computation to a backend server with the apps having to only send REST requests to get results and store data.
Also, since number of responses for Android and IOS were almost the same, we though a backend server can also make developing an IOS application easier as the application is very simple now. The scalability factor has therefore been accounted for. (While YELP might have had a scalable server with proper load balancers, it is a good software engineering practice not to rely on assumptions).

2.2 Developer Survey

Developers of apps were asked questions that were related to ease of development and frameworks/tools that could be used. Here are a few sample questions:

1. What backend platform is easier to work with Android, out of Node.js, Ruby on Rails and Django? Most Votes: NodeJS
2. Which backend platform is the most secure? **Most votes:** Django and Ruby on Rails
3. Which of the backend platforms have the best documentation and support for developers? **Most votes:** Django

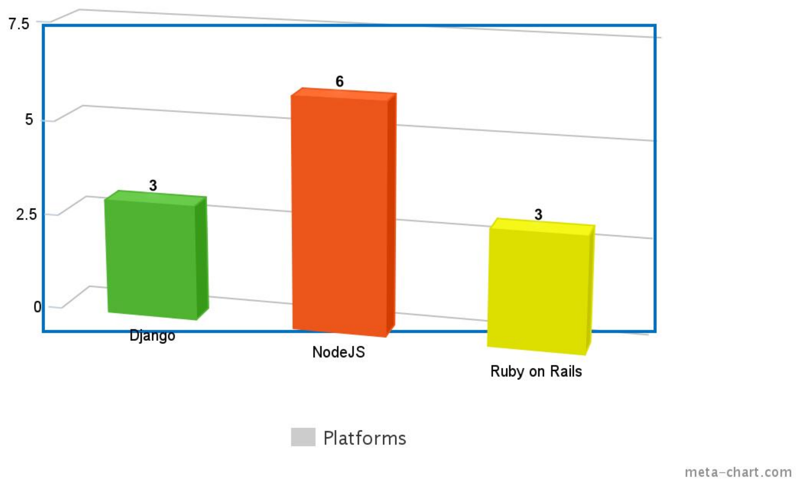

Figure 2: Developer responses

4. Which cloud platform is best suited for the app, in terms of cost and ease of integration for IaaS? Most votes: AWS (also took into consideration Thiyagesh Viswanathan's midterm paper survey)

5. Which model to choose: the moving user updating his/her location coordinates to a database frequently with the stationary user at home polling the server frequently for updates or the stationary user updating his/her requirements(pizza, milk..) to the database and let moving users poll the server frequently and get updates for requirements of all users in the group? **Most votes:** Model 1

6. Which model consumes the least power? Most votes: Model 1 as the moving user needs battery power the most by polling the server less frequently compared to stationary user. (comment of a developer)

7. Will Relational Database be suitable for the project or NoSQL is needed? **Most votes:** NoSQL

2.3 Summary of the Surveys

Users and developers were thus sent these surveys and asked to complete them, and their responses were considered for finalizing our design and implementation for the application.
Their responses indicated that apps in Android was preferred over iOS, and users preferred a lightweight notes-app like design with simplistic UI. Developers preferred to use secure frameworks which can support a heavy duty backend (which is a tradeoff for the lightweight UI). Developers also indicated that they preferred cheaper, easy to use cloud platforms such as Google Firebase and

Amazon AWS to build these kind of apps. Users indicated they would prefer an app that does not drain too much of their mobile phone battery. They also indicated that they would prefer to have a 'Do Not Disturb' option in the app to avoid unnecessary disturbances.

3 Backend Development

Based on the survey results, we chose to use Django as the Backend platform, and AWS and DynamoDB and NoSQL as part of the IaaS. RESTful architecture was adopted to build and Android is chosen for the front end.

We assign each user a UserID and each group a GroupID. All users in a group have the same GroupID. We have a relational schema table for every group. Each group table is named by the GroupID and contains the UserID's of users in that particular group. There is a global table with details of users and their location coordinates. Retrieval from this table is facilitated by a key value pair scheme(DynamoDB). The stationary user stays at home and waits for some alert from a friend near the right kind of store. The mobile user keeps moving and updates location to the DynamoDB at periodic intervals (say 5 minutes since he would be walking and location coordinates wouldn't have changed much). The stationary user will have to frequently poll the server passing his/her search terms (pizza, groceries and so on). The server will then retrieve current location coordinates of all users in the stationary user's group and using YELP API, it computes the pizza shops or grocery shops accordingly. The stationary user will get all these results in a JSON array. The results would state which user of his/her group is near which store. If the result is not satisfactory then the stationary user keeps polling frequently (2 minutes) until a satisfactory result is achieved.
The YELP API takes location coordinates of a mobile user from the database and corresponding search term from stationary user's GET request parameters and returns the right results to the Django server.

4 Location Simulation and Testing Phase

Since the project in its entirety dependent on proper functioning of location coordinate generation, detection and parsing it is important that we do rigorous testing before moving ahead further with alpha and beta phases.

In the first two phases of the project, we had completed the initial customer survey, a developer survey, decided on the architecture and tools to use and also completed the basic backend development. The third phase involved completing the backend development by making it scale for multiple users and multiple groups (In our earlier progress report we had maintained a simple two case scenario. Now the backend can support as many users or groups.

Testing phase or Android development phase: While ideally the Android development phase should have preceded the testing phase, we decided to take a detour after extensive study on few factors. As mentioned earlier, location

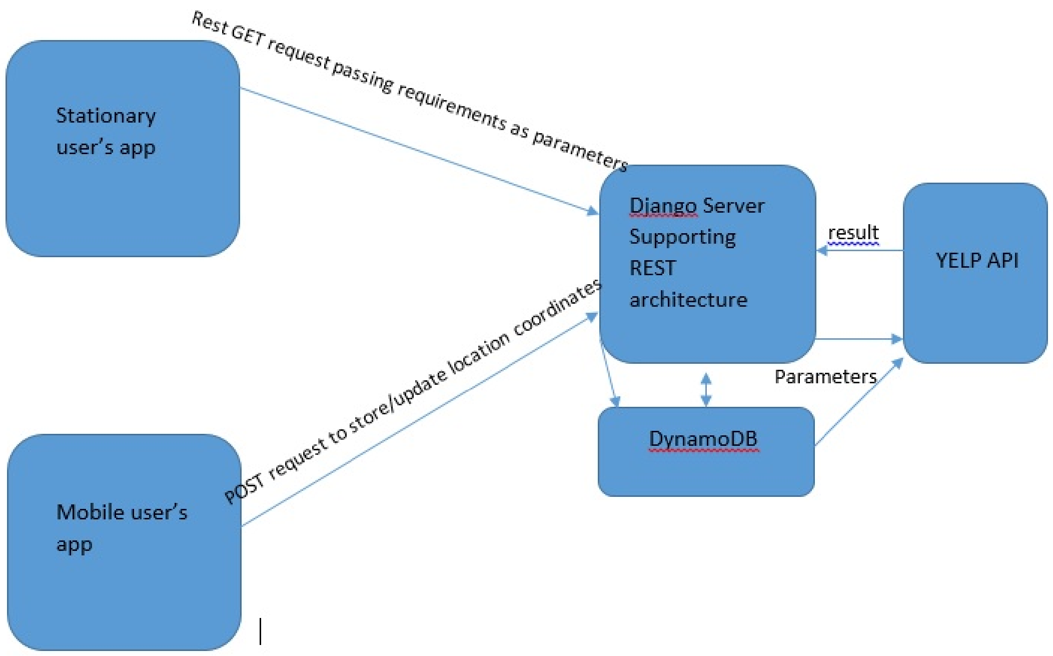
Rest GET request passing requirements as parameters
Stationary user's app
Django Server Supporting REST architecture
result
YELP API
Parameters
DynamoDB
POST request to store/update location coordinates
Mobile user's app

Figure 3: Block Diagram showing Backend.

functioning is very important for the project. In the end it is the location coordinates that are fed to the YELP API. Hence, we work on simulating multiple location coordinates for multiple users in various scenarios, modify the backend accordingly by plugging the loopholes in the software. While location testing can be done/simulated from an android device too, we had few reasons to why a web based simulation application or tool would be better.

1. The Android SDK does not provide an inbuilt location simulator unlike Xcode for IOS. The developers have to implement their own mock location generators which become tedious when we need to simulate over a large number of devices. Also, testing for location would mean that we would be disturbing the Android code which we don't want to.

2. While a web based application would act as an effective intermediate by keeping both the frontend and backend intact (in our case just the backend) it can also be scalable for applications like ours that involves testing location coordinates. Hence this would be an application by itself and could be a valid addition to Android or IOS development company. We have decided to base our Web based application on REST architecture to make it devoid of dependency with a particular backend structure so that it could be scalable across multiple platforms.

3. This can also be of use for future developments of the application. Say for example after two months of deployment considering the number of

downloads we decide have an IOS application, we can directly proceed with front end development without much ado since we had performed enough tests on the backend and correctness of the application. All the developer now needs to do is to develop a suitable interface that would go well with the backend. We have thus save time and costs.

One important advantage of having the simulation performed before the Android development phase is that we were able to roll back several bad pieces of code that made computations inefficient or frequently showed wrong results. On trying for a lot of different scenarios we were able to identify the various possibilities of a request getting failed. Like, user who is not in a group can still find answers from all the members of the group. POST and PUT methods cannot be combined into a single methods as we realized that PUT involves just partial parameters in many cases (name is excluded often). POST requires all the parameters to insert unless there is a default value. But since we are excluding the name as it is not of much use, we cannot have default values for it and therefore it became necessary to have separate methods for POST and PUT requests.

Since the backend has been refined much, Android development can take place smoothly correlating with background. There are no more major changes to JSON structure or structure of a model. Hand we had the Android phase before simulation phase, then we might have ended up committing changes to both Android and Backend parts of the application. Taking into account the lack of experience with Android, this seemed to be a decisive trade off.

5 Challenges

One important challenge faced was to complete the various phases in a span of 45 days or before. There were lots of technical challenges we faced while building the REST interface for Django. The backend is exposed to three different endpoints with GET, PUT and POST access for each of the endpoints. The requests had to be carefully crafted discussing as much boundary and failure conditions possible. Normally, these requests are vulnerable for hackers can take into advantage a poor request structure and gain access into the server. This was the biggest challenge that we encountered.
We had to create a REST framework for a model less object. That is we had to make possible a user to access list of restaurants matching the search term directly from YELP API without storing it in a database. We had to do this to save space since the data is of no use to us. Having GET, PUT, POST methods for a non database object comes with lot of complications. We had to build a local, in memory hash table to store the userid and corresponding search results and keep flushing it repeatedly after a request is processed and response is sent.
The next big challenge we faced was while building the location simulation tool. There were instances where we had to update the database by sending back to back requests. To make this possible in Jquery one cannot simply use a loop to achieve it. A callback mechanism is implemented to achieve synchronous PUT requesting.
Further challenges involve, background location tracking and android app development. We list it as a challenge since we do not have much prior experience

with Android app development.

6 Tools and Technologies Used

1. **Backend:** Django REST Framework 2.3.8: Python 2.7. APIs and Libraries used: YELP API and a corsheaders middleware package for CORS access.
2. **Location Simulation Tool:** Language - Javascript, APIs and libraries used - Google Maps API, Jquery
3. System used for Development: Mac OS X El Capitan 10.11.5
4. **Text editor:** Textmate and Vim
5. **Other packages:** Latex for documentation, Github for development, Excel for plotting graphs and Google forms for survey.

7 Deliverables

Github: https://github.com/Thiyageshv/PizzaReminder

8 Individual Contribution and What I learned from the Project – Anush Ramsurat

My individual contribution to the project was to design the features of the application, conduct user/consumer surveys and to build the location simulator and the web overlay for the web application and handle the testing phase for the product. I have had limited prior experience using the Google Maps Javascript API or the JSON parsing process for web projects and for this project, I had to learn how to use this API in addition to learning how to use the JQuery library to build out the web overlay and show the functionality of the application. I also learned how to conduct a proper user survey with a lot of parameter supervision, and I used my insights on conducting and following up with users from another project I worked on with Prof. Kaiser, to broaden the scope of this survey and obtain results sooner and more efficiently by targeting the right channels of distribution.

When you're building a product that espouses a new idea and addresses a significant pain point, it comes with its own design and product decision challenges. We discovered that this app would solve a problem at scale for potentially a huge number of users, because people do have this pain point of not being able to or not wanting to step out themselves to buy food/groceries but tap their friends who might be near a store to ask them for the favour. Essentially, this follows the 'Uber model' of delivery – the app merely acts as an interface and facilitates interaction between people who want to have things delivered and people who are willing to deliver them. I learned a lot while debating and designing the features that we implemented, particularly the ‘Groups’ feature.

An interesting caveat to this is that group selection in the app was initially designed by us to be based on location/proximity radius and groups would be formed based on how close the users lived to each other – as this had more likelihood of them having to tap a particular group for say, pizza (which could be within a two mile radius) rather than for vegetable shopping, which could vary upto a five mile radius. What Prof Kaiser suggested was that we should have a provision to allow selection and monitoring of locations of users in multiple groups at the same time, and these groups need not necessarily be based on radius; they could be based on relationships such as friends, family, etc. too.

Hence, a potential redesign of this feature is warranted. I learned a lot during the course of this project in terms of design decisions and of course, programming and implementation using web technologies (HTML5, Javascript, JSON, jQuery, etc.) and testing the functionality by using both pre-formatted input and randomized input.
I had a lot of fun doing the project and learned quite a lot within a short span of time, and we managed to create a functional product and study the various software engineering processes that went into the building of the project.

9 Individual Contribution and What I learned from the Project – Thiyagesh Viswanathan

The project is about building a complete application which involves designing the backend architecture, formulating user experience designs and also perform rigorous testing. Therefore, we put ourselves all stages of software development cycle. While I had prior experience in Django and Python, I did not have experience with the Django REST Framework. Apart my reinforcing my experience with Django, I also got familiar with the Django REST Framework. Also, working on the tool reinforced my strengths with JQUERY Ajax.

The location simulation tool is lightweight to a great extent. It is flexible to try any combinations of users and groups. Designing such a tool involves careful crafting of the JQUERY objects. Like synchronously transmitting continuous PUT requests rather than the traditional asynchronous method used by JQUERY. Also, I got to learn about various security aspects. The advantages of Django is that it is very adamant on forcing the user to integrate security mechanism like token passing using cookie variable. I had never done this while developing through PHP using Apache.

One important takeaway from this project is that, one must have a careful survey of tools to use and features to include. Also, important studies need to be performed which can be of great use in long run. For example, the decision to create a tool for simulations before developing Android front end is something I would not have considered if not for this course. It would have cost me a lot in time and performance otherwise. Therefore, any software that is developed for real world must take lot of decisions into considerations. Intelligently developing

the application throwing all the skills alone cannot achieve success. My decision making skills have really improved in the process.
In short, this course and this project particularly thought me to make careful decisions on every phase. With having limited experience in the industry, I can claim that I can confidently release an application or start up in future by employing healthy practices.

10 Potential Future Work and Applications

The functionality and proof of concept of the application has been demonstrated to its full potential. The app is able to allow users to track other users in real time and form groups on the app as per their preference and gain location data and determine who best to contact. Possible extensions in terms of features could include the ability to select groups that the user wants to request location data from, instead of polling the entire user set or just a particular group. Also, group selection could be extended beyond proximity-based to other parameters. In addition, the front-end would be built in Android and the app would need to be ported on to the mobile platform – and would need to be equipped to handle large scale requests from a huge number of users. This app has the potential to be a large-scale product.
Also, Android/location testing(just to check if location coordinates frequently poll the server and are stored)should be done. We have already check for the correctness of the application in our previous phase. Alpha and Beta testing with users should be completed prior to full release. Users generally tend to have a lot of feedback regarding features, and we should modify the app to suit user needs and address any glaring vulnerabilities before a full-scale release. Finally, we push it to app store and based on demands, extend the model for an IOS development.

11 References

1. Google Javascript API: https:developers.google.commapsdocumentationjavascript
2. Google Android SDK Toolkit - http:developer.android.comtrainingindex.html
3. YELP API -https:www.yelp.comdevelopersdocumentationv2overview
4. JQuery - https:jqueryui.com
5. Django REST Framework - http:www.django-rest-framework.org
6. DynamoDB - https:aws.amazon.comdynamodb
7. sqlite3 -https:docs.python.org2librarysqlite3.html
8. Backend comparative study - https:www.quora.comDjango-vs-Rails-vs-Node-js-vs-Code-Igniter-YII-Which-is-better
9. Django design philosophies- https:docs.djangoproject.comen1.7miscdesign-philosophies

10. Rails vs Django Bernardo Pires 2014- https:bernardopires.com201403rails-vs-django-an-in-depth-technical-comparison

11. Django vs Node JS Suchan Lee 2014- https:suchanlee.svbtle.complaying-nice-with-django-backbonejs

12. Android Location tracking Mobisoft- http:mobisoftinfotech.comresourcesblogandroid/3-ways-to-implement-efficient-location-tracking-in-android-applications

13. IOS location tracking https:developer.apple.comlibraryiosdocumentationUserExperienceConceptualLoca

14. Model_less view Marcel Chastain- https:github.commarcelchastaindrf-demo

15. CORS- https:spring.ioguidesgsrest-service-cors