WolfPool: An easy way to plan your rides

Ankitkumar Jain
Masters of Computer Science
NC State University
Raleigh, North Carolina
aajain2@ncsu.edu

Chirag Jain
Masters of Computer Science
NC State University
Raleigh, North Carolina
csjain@ncsu.edu

Nirav Jain
Masters of Computer Science
NC State University
Raleigh, North Carolina
najain@ncsu.edu

Pratik Kumar Jain
Masters of Computer Science
NC State University
Raleigh, North Carolina
pjain22@ncsu.edu

Rishabh Jain
Masters of Computer Science
NC State University
Raleigh, North Carolina
rjain10@ncsu.edu

ABSTRACT

UberPOOL and LyftLine are very efficient and affordable for traveling short distances. However, for suburban areas and a majority of metropolitan cities this option is not available. This is mainly due to low number of people traveling on the same route. The connectivity and frequency of public transportation is limited in these areas. In addition to these issues, there are a lot of difficulties to communicate and plan the rides using social media platforms such as Facebook and Whatsapp. To tackle the problems mentioned above, we have devised WolfPool as a service that will specifically target to enhance communication and convenience in planning of rides.

Keywords

UberPOOL, LyftLine, Ride sharing, Google Geo-coding, Campus pooling, Planning, Car pooling, Rentals, Cost saving, Time saving, Safety, Convenience

1. INTRODUCTION

Every fall, students from all over the world apply to North Carolina State University to get admitted into their dream course to fast-track their career progression. Majority of the students that are admitted are International students who not only have to overcome what is known as a "cultural shock" and adapt to different cultural and social norms but additionally also have to face other challenges like figuring out the most cost-effective mode of transportation for their daily commute. Since, renting a personal car, or using any ride-hailing taxi application add to the expense of living, it is not a feasible option for many.

Another alternative is public transportation, more popularly known as GoRaleigh, but the frequency of the service is limited, and the travel time also significantly increases based on the distance between the pick-up point and the destination, and the number of stops between the same. Moreover, there is an absence of more economical means of transportation in Raleigh like ride-sharing application for example, "UberPOOL". Looking at the plight of students based on the results of our survey, we came upon a realization that an application, that lets students who are traveling to the same destination, or are heading the same way directly and mutually co-ordinate with each other, so as to share their ride would be more favorable, as it would help combat travel costs. Taking this into consideration, we have developed a solution, in the form of an application that will overcome these challenges.

2. CASE STUDY

In Raleigh, if there is a bus available for a particular destination outside campus, the frequency of that bus is not so great. Furthermore, the routing is also not optimized. For example it takes more than 50 minutes to reach Crabtree Mall from NCSU main campus compared to less than 15 minutes taken by a car (as shown in figure 1). This means a lot of time is wasted in the journey and waiting for the bus. Other great alternatives to public transportation are car rentals and private services like Uber or Lyft.

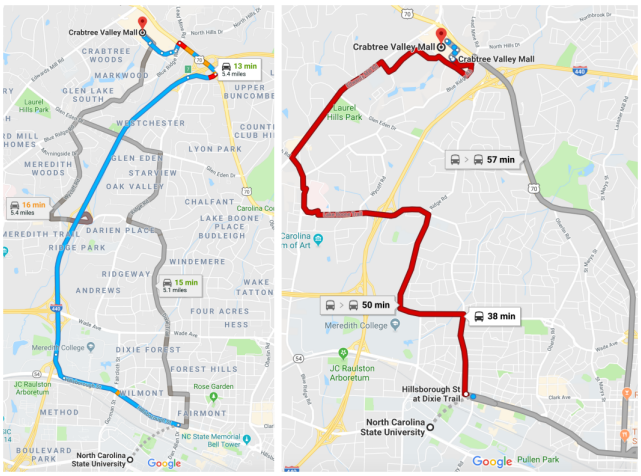

Figure 1: Map showing the time taken by public transportation [right] versus time taken by car [left] between same source and destination

They overcome the aforementioned disadvantages of public transportation but at a greater cost. To reduce the cost factor, Uber has the pool service in a few metropolitan areas in United States and across the globe. Lyft on the other-hand has restricted their car pooling service (LyftLine) to

only United States. The existing service by Uber that is UberX provides about 40% less cost fare to users than normal taxi services. By launching UberPOOL services the prices were reduced even more. For example, UberX would cost about $10 from one place to another; UberPOOL will only cost $6 [13].

This also has a great benefit to the environment. UberPOOL in San Francisco after a year and a half after the launch made up nearly half of Uber's trips in the city. In one month alone, the service saved 120 tons of carbon dioxide emissions when compared with non-pooled trips, the equivalent to the output of over 128,000 pounds of coal [14]. LyftLine, which matches drivers with one or more passengers for a less expensive fare, is currently available in San Francisco, Los Angeles, New York City, Boston, DC, Austin, Chicago, Atlanta, and Miami, Denver, Philadelphia, San Diego, San Jose, Seattle, and Newark (as shown in figure 2). It's an unsurprising expansion, given that Lyft Line represents 40% of the rides the company does in the cities where it's currently available. Launched in 2014, Uber said in April that it had passed 100 million Pool rides [12].

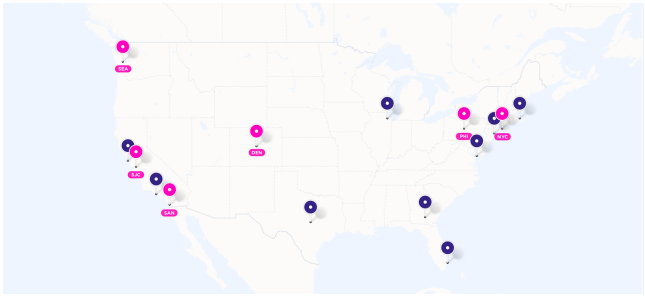

Figure 2: Map showing the cities in United States where UberPOOL and LyftLine are available in black and pink respectively [as of Apr 5, 2016]

Unfortunately a lot of college towns and campuses currently do not have these services. One of the major reason for the success of the car pool model was the heavy demand of traveling on a common route by people. This means that even if UberPOOL or LyftLine would be available in these places it would not be as efficient in college towns, as the frequency and the number of people making trips on a common route is very less, effectively beating the purpose of the pooling service. Also, in few places where the pool service is available, it is not operational during certain hours in night. Thus, we observed and personally experienced in Raleigh that a majority of the students used WhatsApp/Facebook or any other social media platforms to reach out to people to plan and share their rides. This required a lot of coordination and often led to a lot of confusion. There was no way to check if a particular ride sharing plan is still available. Confusions were also caused in deciding what the source and destination of the ride should be. To solve this sometimes special groups are created on the same social media platform but this is just a temporary solution. Also, a majority of the people were comfortable to plan a trip with someone they know. Few of the instances where a large number of students performed such planning are as follows.

- Social Security Office
- Airport Pickup/Drops
- Indian Stores (Patel Brothers, Around the World)
- Walmart/Best Buy
- Crabtree Mall
- PNC Arena (Athletic events)
- College Surplus Sale

3. INITIAL SURVEY

We conducted a survey prior to the implementation to understand the feasibility and adaptability of the idea.

Question : If Public Transportation or Personal Vehicle is not available would you like to use an application that helps you share your Lyft/Uber? (Pool option is not available in Raleigh)

We asked this question so that we could evaluate the usability and adaptability of the application in the areas where there is no option of pooling their rides through Uber and Lyft.

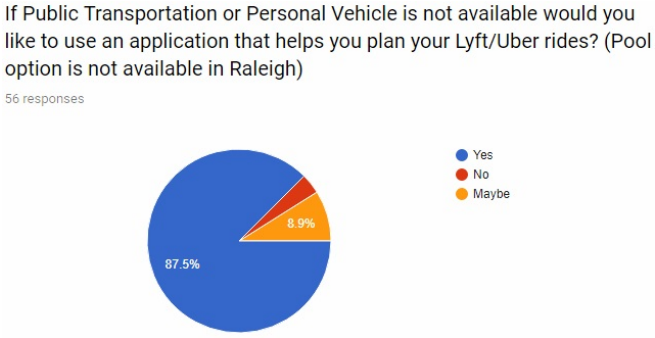

Figure 3: Responses for Question

3.1 Survey Conclusion

Based on the 56 responses we got from the survey we concluded that there was a large number of people who use private cab service like Uber/Lyft and/or rental cars for their personal transportation and would like to decrease the burden of cost and time associated with them. Also the responses for Question 2 indicate that the Uber/Lyft pooling model won't be effective and profitable as it is based on the idea of high number of people traveling on common routes but majority of people have a travel frequency of just 1-5 rides per month. So there is a need of an application that would help the people plan and share their rides in a convenient and safe way and be assured about it well in advance, especially in the cities where there is no Uber/Lyft pool available.

4. IMPLEMENTATION

4.1 System architecture overview

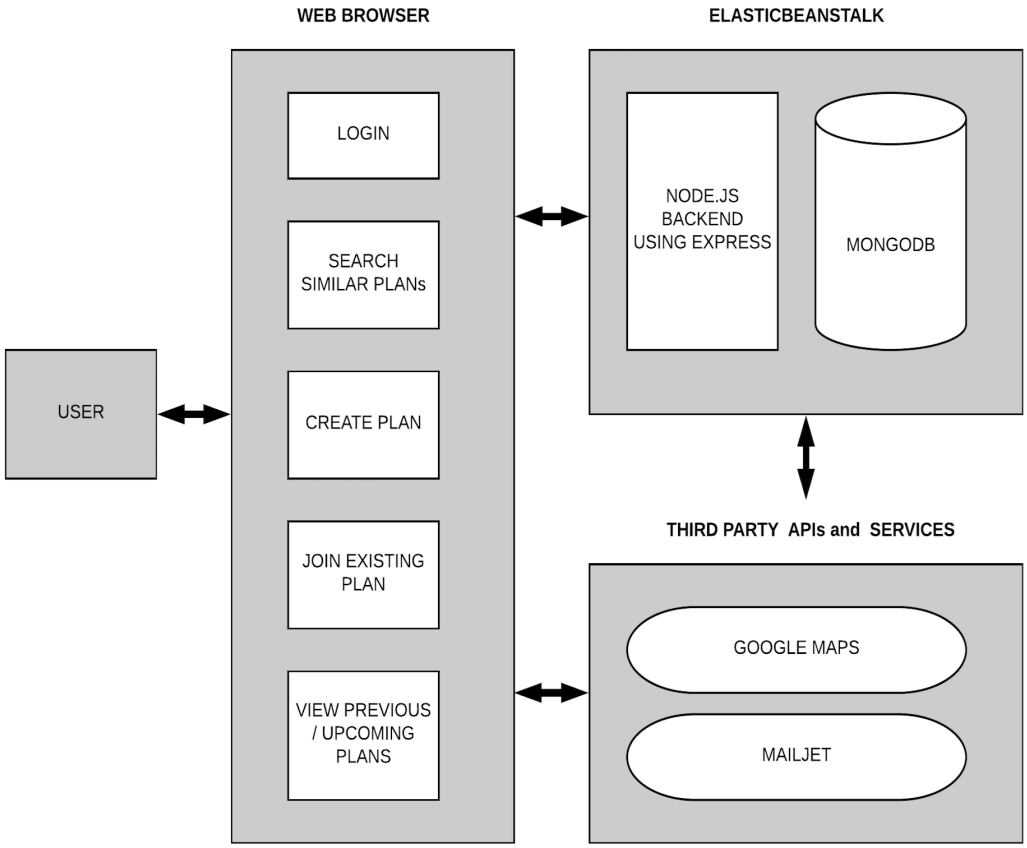

Figure 4: System architecture overview

4.1.1 MEAN stack

MEAN[9] (as shown in figure 5) is a JavaScript software stack which is used for building dynamic web sites and web applications. The major reason why we decided to use this tech stack is due to it's following advantages:
1) It is open source
2) Applications can be written in one language for both server-side and client-side execution environments
3) Cloud Compatible
4) It supports MVC architecture
5) Uses json to transfer data

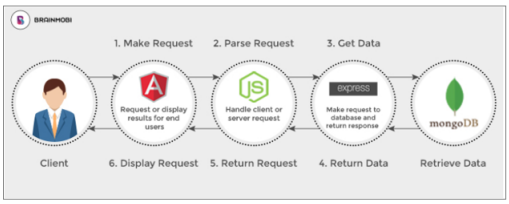

Figure 5: MEAN Stack overview

Following components of MEAN stack are incorporated in our application:

MongoDB: It is a NoSQL database. It uses JSON format for data representation. One of the major reasons for choosing MongoDB is that it gives the ability to use JavaScript as a single language throughout, which fits perfectly in the MEAN stack.

ExpressJS: It is a HTTP server framework for web applications that gives useful modules and components to work upon the common task for the website. It provides the MVC framework to the application. It also gives a simple interface to make request endpoints. Apart from that it is good at enabling the simple REST routes, handling automated HTTP header and supporting Connect middleware to plug in synchronous functions in order to manage the requests and responses.

Node.js: It is a concurrent JavaScript environment used for building scalable and fast web applications. It compiles the JavaScript code to native machine code before the execution. It is lightweight and perfect for the real time applications.

AngularJS: It is a frontend JS framework to develop complex client side applications with modular code and data binding UI. It is used to develop the single page applications with the use of the MVC architecture. It improves the structure of the code and makes the testing easier with the

dependency injection.

We have used jQuery in our project instead of Angular.js. This is due to the fact that the UI for our application is not too heavy, therefore implementing jquery was much easier. Another reason was the time constraint, learning Angular.js would have taken too much time which we utilized in developing more features.

4.1.2 Database

Amongst the various database products, we are using MongoDB for having flexible schema, fast performance, high availability, security and compatibility. MongoDB is an open-source document-oriented NoSQL database. Instead of storing the data in tables made out of individual rows, like a relational database does, it stores the data in collections made out of individual documents. In MongoDB, a document is a big JSON blob with no particular format or schema. MongoDB's document data model maps naturally to objects in application code, making it simple for developers to learn and use. Documents give you the ability to represent hierarchical relationships to store arrays and other more complex structures easily. In the system a plan can have variable number of people (email ids) associated with it without having to specify that while creating the plan and MongoDb makes it possible to have that flexibility. MongoDB has a great support to store and retrieve location data. Moreover MongoDB is part of the MEAN stack and is highly compatible with node.js.

NPM (Node Package Manager) also provides Mongoose package which is a MongoDB object modeling tool designed to work in an asynchronous environment. It can be used to access the model using objects from nodejs. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box. MongoDB provides high availability by providing support for replication and sharding. We have installed MongoDB database on elastic beanstalk on the underlying EC2 instance that is running the node.js server. The instance type is db.t2.micro with General Purpose (SSD) Storage capacity of 20 GB.

4.1.3 Amazon Web Service

We are using Amazon's Elastic Beanstalk[2] which is an easy-to-use service for deploying and scaling web applications and services. Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. For the backend framework we are using Node.js[11] v8.9.3 running on 64bit Amazon Linux 2017.09 v4.4.4. The Instance type is t1.micro and availabilty is in all zones provided by AWS. Our application's primary hosted region is the North East US region in Virginia but it is load balanced and can instantiate new hosts in multiple regions

4.2 Features

4.2.1 Geolocation for addresses

The system uses multiple APIs to perform geospatial operations. To get user's current location we are relying on the browser's native support for HTML5 gelocation API[8]. This API requires a secure connection which is handled by the SSL certificate. This leverages any GPS sensor available on the device and results in high accuracy for mobile device having dedicated GPS sensor.

We have used the Google Maps API for JavaScript to get an auto-complete input field like Google Maps and perform reverse geocoding[6] of user's current location (as shown in figure 6). This enhances the user experience and removes the burden of typing the entire address in the form. To improve the accuracy of the address search results, we have also displayed a map besides the form that displays the user's intended source and destination address. The Google Maps API is highly scalable and can support high volume of geocoding and reverse geocoding requests. Also, the Google Maps are platform independent are render efficiently across all browsers and devices (as shown in figure 7).

4.2.2 Email Notifications

After a user registers on WolfPool, the user receives an email for account verification. This email consists of a link that verifies the user and it redirects the user to the application. If the user forgets to verify the account within a predefined time then the account will be deleted. The application runs a batch file every 24 hour at 4:00 AM and after this if any user is not verified then the user account is deleted. After redirecting, the application prompts the user to log in again to use the application and search for plans.

The user receives an email after creating or joining a plan (as shown in figure 8). The user receives an email every time any other user joins the plan of which the user is a part. This email consists of the details of the trip. The details consists of the source and destination location along with the date and time of the plan. This email will have the email-ids of all the user that are a part of the plan. Using these details the users can contact with other users and co-ordinate accordingly.

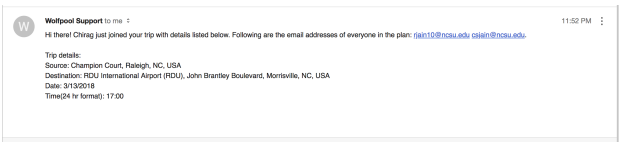

Figure 8: Sample email received after joining a plan

The users can then book a car pooling or rental service from a source to destination according to everyone's convenience and in this way a solution to their problem will be provided. To send the email, we have used a third party service called Mailjet. In Mailjet, we can also monitor number of emails that were sent directly to the spam folders and to the inbox.

4.2.3 Creating and sharing travel plan

The application provides the flexibility to create and share your own travel plan. The application asks details like source location, destination location and time slot for the trip before making it public. This feature makes it very convenient for the users to create their own plan if it is required. Currently, we do not allow to have a plan with more than 6 people assuming that it will be difficult to arrange for transport of more than 6 people in a car.

4.2.4 Join existing plan

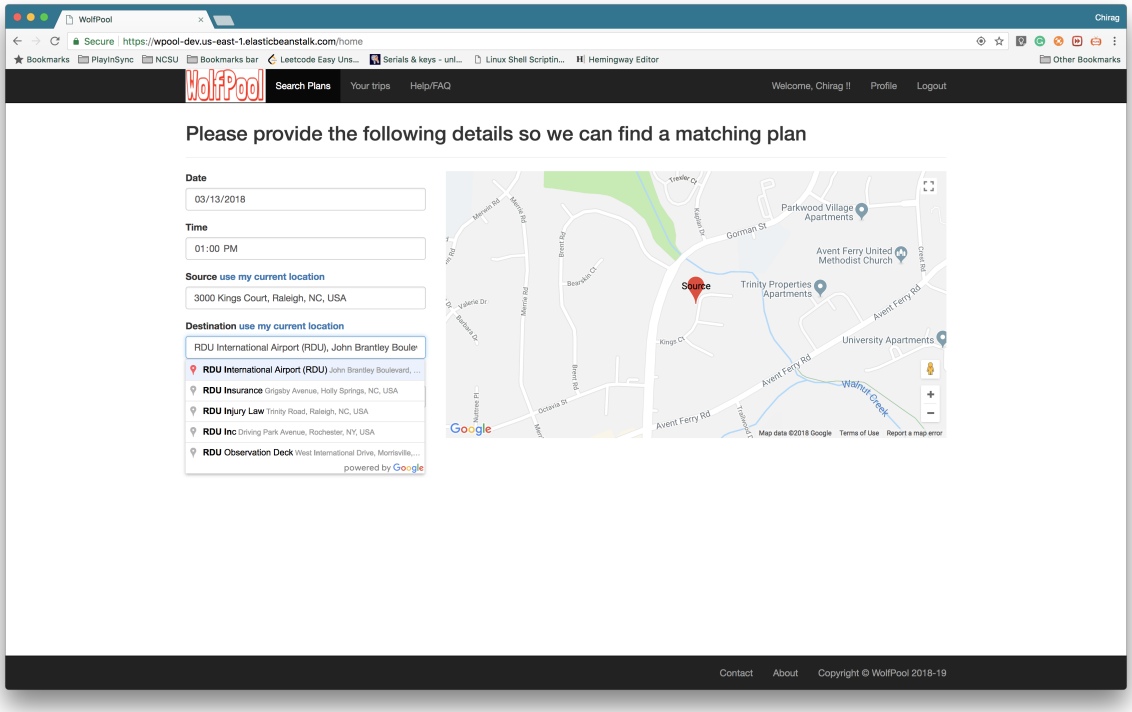

Figure 6: Website rendered on a Desktop browser

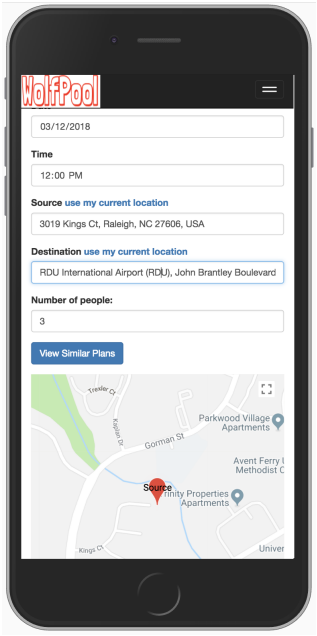

Figure 7: Website rendered on a Mobile browser

Along with creating plans, WolfPool also provides users with an option to join an existing plan. The user can directly search for the plans according to his / her requirements and if there is a best possible option that matches the requirements then the user can join that plan. The user can also sort the search results based on distance, date, time as per their convenience. Joining a plan adds that user to the list of users sharing that trip, and everyone receives an email notification that the user has joined the plan, with the corresponding trip details.

4.2.5 Trip matching

WolfPool uses a robust algorithm to efficiently match available plans. Various factors are taken into consideration during this process including source address, destination address, departure date and time and the number of people.

The application provides multiple matches, if exists, giving users the flexibility to choose between multiple options (as shown in figure 9). Whenever user enters his travel details, he is always recommended to join similar plans instead of creating new plan. Whenever a new plan is created, the source and destination latitudes and longitudes are stored in the database. When a new user requests to view similar plans, the systems queries the database to find plans that have source and destination coordinates within 2 kilometer radius from the current user's location of source and destination and date and time of travel greater than the current user's preferences. If the user is currently a part of the plan then he won't be shown that plan in the search result.

To avoid hitting the daily limit of 2500 requests imposed by Google Maps API, we have used mathematical equation called Haversine distance formula to calculate the distance from geolocation data. The matched plans are returned on the screen and displayed in a jQuery datatable which pro-

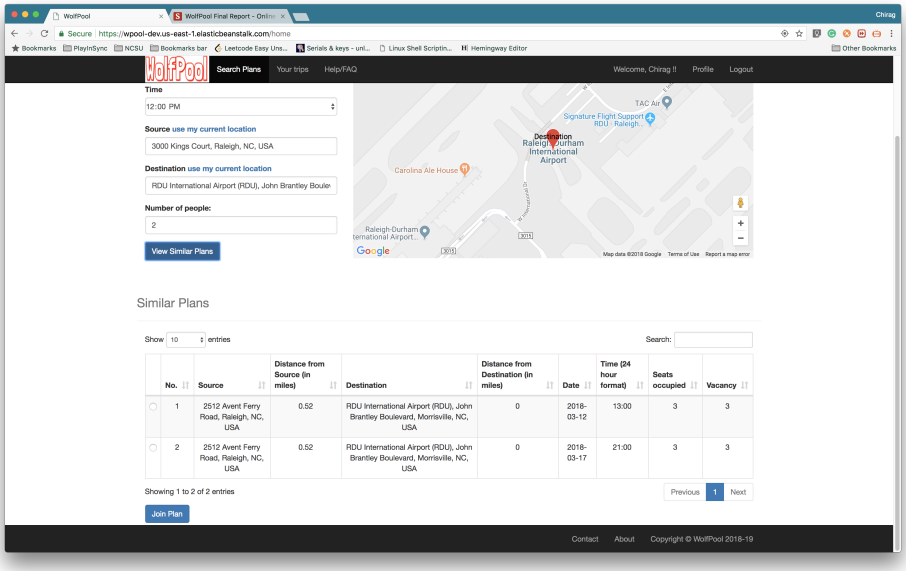

Figure 9: The webpage showing search results for similar plans

vides flexibility to user to view and sort the plans by distance from source / destination or date of travel as per his preferences. If the similar plans do not match user's preferences, he / she can create a new plan. Corner scenarios like not showing the plans in which the current logged in user is already part of is handled.

4.2.6 Security

WolfPool ensures complete safety of travellers by limiting the access to the application to only university affiliated personnel. This will be performed by parsing the email address of registrants to look for '.edu' domain. Due to this, the system is accessible to members of the university whose identities are verified and trusted by the university itself. As a result of this, intruders are restricted to use our application. This ensures the legitimacy of the users and thereby, ensuring safety. Our web application is also secure technically.

The application uses latest SSL encryption certificate standards that secures the data transferred over the network. This is important as we are collecting confidential details from the user like their university email address, name, residential address, etc. Also, we do not save passwords as plaintext in our database. We have encrypted the passwords using hashing on the actual password. Along with this, to make the user sessions more secure we have used a server side secret key.

There are no API keys hardcoded and these are read as environment variables from the Elasticbeanstalk configuration. Similarly, the default root for the EC2 instance is deleted and IAM users with specific roles and accesses are created to ensure use of only intended services when using third party tools like TravisCI.

4.2.7 Tracking Trips

The application provides users the feature to check their previous and upcoming trips. Details such as source location, destination location, date, time and details of co-passengers are shown in the 'My Trips' page (as shown in figure 10), which enables users to contact their co-passengers. The user can also sort the trip list based on date, location, etc. This feature makes it easier for the users to get details of any trip they are / were a part of.

4.3 Software Engineering

4.3.1 Agile Methodology

We decided to use Agile methodology to develop our project, since the project was a time critical application as we just had around two months to work on it. This helped us develop small incremental features, with each release building on top of the previous one. We used the storyboard provided by github, in order to create around 40 user stories and issues. Dividing the user stories this way also helped us prioritize these tasks relatively, just as we do in planning poker technique. We learned the importance of effort estimation, as our stories often moved around from 'done' to 'in progress' sections in the storyboard. At this point, we did a sprint retrospective and realized that multiple people were working on same files because of overlapping user stories. Thus, we decided to do pair programming, and and as a result we were able to improve on the quality of the sprints. This enabled us to deliver on all the features as promised in the initial report.

4.3.2 Pair Programming

Soon after we started working on the project, we realized that the effort distribution was not uniform. As a result, we resorted to pair programming in order to work on the features and resolve bugs. Developers working on overlapping features worked together, which helped us close our tasks faster and as a result have successful sprints.

4.3.3 Code Review

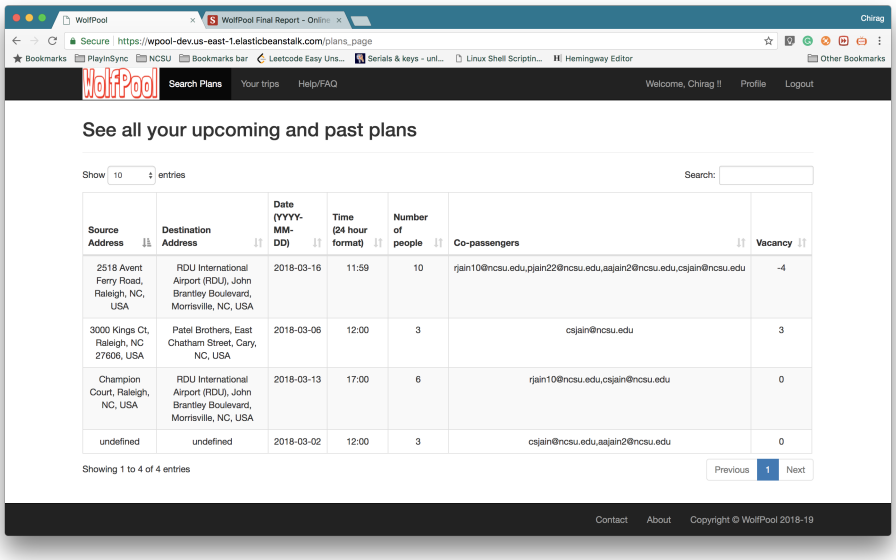

Figure 10: The webpage showing the upcoming and old plan

Code reviews practices were followed throughout the project thoroughly. We disabled direct commits to the 'master' branch, and had a mandatory review system in place in order to merge pull requests. This helped us reduce the number of bugs in production. We also did pair programming, which is another well known technique for code review.

4.3.4 *Proof Of Concepts*

AWS vs Heroku vs Google Cloud.

We compared the services offered by Google Cloud Platform and Heroku[4][3][7]. We decided to finally go with AWS because it provides a lot of configurability and control over the platform we use to host our application. Due to the extensive documentation, support and experience of working with AWS it was the obvious choice. The options for scalability in terms of processing power and availability by means of multiple server zones are varying and can be selected at a very fine level based on the cost and efficiency.

The comparsion between AWS and Heroku is shown in figure 11.

	Amazon Web Services	Heroku
Owner	Amazon.com	Salesforce.com
Hosted on	Proprietary servers	Amazon's data centers
Service type	IaaS (Amazon EC2) PaaS (Amazon Elastic Beanstalk)	PaaS
Languages	.NET, Ruby, NodeJS, Go, Docker, PHP, Python	Node.js, Java, Ruby, PHP, Python, Go, Scala, Clojure
Geographic regions	USA, Canada, South America, Europe, Asia-Pacific, China	Europe, USA, Australia, Japan
Key features	- Various deployment options and ability to roll back to the previous version; - Full control over app health or servers changes with email notifications; - Quick restart of all app servers with a single command; - Adaptive load balancing spreads out the incoming app traffic across instances automatically; - Automatic scaling of your web application based on its particular needs and defined conditions;	- Fully manageable runtime environment with smart containers (dynos) system; - Manual horizontal and vertical scaling; - Ability to roll back your database or code in no time; - On-board app monitoring system to keep track of metrics, like response time, throughput, memory, etc. - Consistent GitHub integration;
Categories of clients	Startups, Medium Businesses, Large Enterprises;	Freelancers, Startups, Medium Businesses, Large Enterprises (Heroku Enterprise);
Used by	3M Health Information Systems, BMW Group, Airbnb, Coursera, Atlassian, Discovery Communications and others.	Product Hunt, Macy's, Toyota, Citrix, Westfield, Carbon Five, Lutron, Yesware, Greystone and many others.

Figure 11: Feature Comparison - AWS Vs Heroku [3]

MongoDB vs MySQL.

MongoDB enables us to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale. MongoDB can also be scaled within and across multiple distributed data centers, providing new levels of availability and scalability previously unachievable with relational databases like MySQL (as shown in figure 12). As the deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing the application. In contrast, to achieve scale with MySQL often requires significant, custom engineering work.

MongoDB has a rich query language, highly-functional secondary indexes (including text search and geospatial), a powerful aggregation framework for data analysis, faceted search, graph processing and more.

MEAN stack vs LAMP stack.

One differentiating feature of the MEAN stack is that JavaScript code can be reused for both client and server, and makes life easier for developing teams to switch between the two. It makes the workflow homogeneous and also enables two groups of full stack developers to work well together on the application as a whole. On the other hand, LAMP uses PHP/python for server and JavaScript for client which helps maintain security if the client and server code bases are kept

	MySQL	MongoDB
ACID Tranactions	Yes	Yes
Flexible, rich data model	No	Yes
Expressive joins, faceted search, graphs queries, powerful aggregations	Yes	Yes
Idiomatic, native language drivers	No	Yes
Horizontal scale-out with data locality controls	No	Yes
Analytics and BI ready	No	Yes

Figure 12: Feature Comparison - MongoDB vs MySQL [10]

separate.

Another feature of MEAN is that it uses a Non-Relational database while LAMP uses Relational database which provides development team with the benefit of enhanced speed for data retrieval.

MailJet vs other email services.

MailJet is a Paris-based, all-in-one Email Service Provider (ESP) that allows businesses to send both marketing and transaction email. Initially we tried sending email from a Google Gmail account but it was not supported in AWS so we had to change to Amazon Simple Email Service (SES) but it had a limit on the number of emails that can be sent using that service. To increase the limit, we would have to increase the quota which would incur additional charges. So we decided to use Mailjet, which is a free service that is heavily used in our application.

5. TESTING

Following table (figure 14) consists of all the tests which are carried out on the system.

6. EVALUATION

6.1 Comparison with similar platforms

We performed an initial assessment of WolfPool by comparing it with the existing systems/tools (figure 14) that are currently available and used to plan and share trips

6.2 Participants

The participants of both, our initial and final evaluation survey are graduate students in the Computer Science department at North Carolina State University. The places where NCSU students often need pooling service are India grocery store (Patel Brother), SSN office and RDU airport, and therefore, we saw many plans were created to go to these places. Note that while the evaluation results are limited to NCSU students, the application itself is not. Any student with a valid '.edu' email can participate and use our system.

6.3 Evaluation Survey Results

Question 1 : How appealing is the idea to you?
We asked this question to understand the usability of the application and whether the idea of such an application will help the user in the future to plans their trips.

Test	Desciption	Result
Registration	Is user able to register	Pass
Email Verification	After registering, user received a verification email	Pass
New User Login	If user is not registered then it should be redirected to register page	Pass
User Login	If user tries to login after verification	Pass
Session Management	When user tries to access a web page without login, the user should not be able to access that page	Pass
Form validation	Is front end validation implemented	Pass
Search Plan (Based on Time)	Search plans shows the plans where date and time is greater or equal to date and time present in the filter and source and destination is within 2000 meters radius	Pass
Search Plan (Based on User)	Search results should not show the plans that the current user is already part of	Pass
Search Plan	If user enters vacancy more than possible then it should not show plans	Pass
Email Notification	Are all people in trip receiving email notification when a new user is added to the plan	Pass
Create Plan	Is user able to create a new plan	Pass
Trip Details	User is able to see his/her previous and future trip details	Pass
Join Plan	If user select a plan to join, it will add the user to the plan	Pass

Figure 13: Testing along with the result

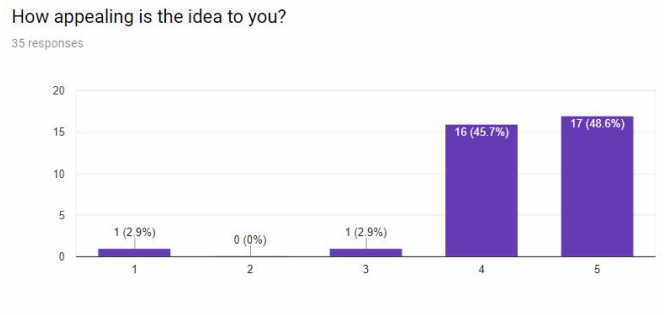

Figure 15: Responses for Question 1

Question 2 : How would you rate the overall system on scale of 5?
We asked this question to understand the acceptance of our system and the scope for improvement.

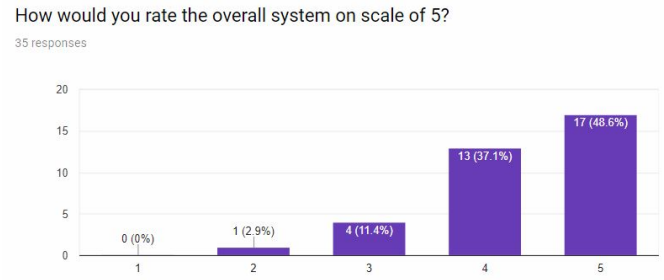

Figure 16: Responses for Question 2

Question 3 : Did you find it easy to search for similar plans?
We asked this question to understand whether the user could understand the searching feature of the application which is

	Security	Adaptability	Coordinability	Trip Experience	Availability	Cost Saving	Time Saving	Convenience
uberPool	-	++	-	-	-	++	-	++
LyftLine	-	+	-	-	-	++	-	++
Carpool	++	+	+	++	+	-	++	++
Car Rental	++	+	+	+	+	-	+	+
Public Transport	+	++	+	+	+	++	--	-
Facebook Post	+	-	-	-	-	+	--	-
WhatsApp Group	++	+	-	+	+	+	-	-
WolfPool	++	+	+	++	+	++	+	+

Figure 14: Assessment of various products/services

the main functionality of the application.

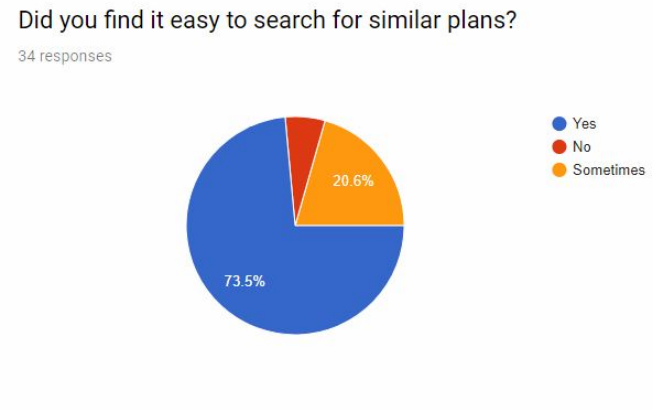

Figure 17: Responses for Question 3

Question 4 : Was the flow of the application intuitive?
We asked this question to understand whether the flow of the application was proper as the application is self-explanatory.

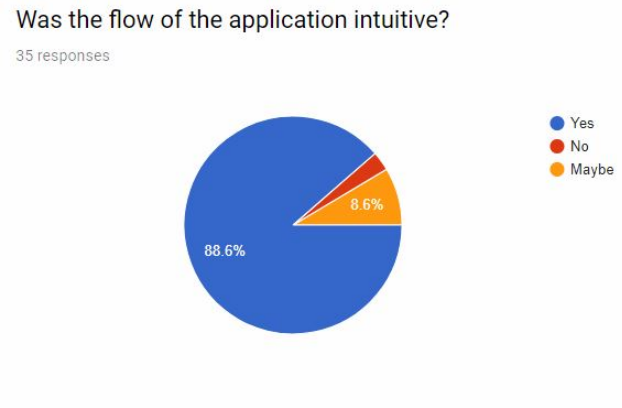

Figure 18: Responses for Question 4

Question 5 : Rate the user experience on scale of 5?
We asked this question to understand whether the UI of the application was easy to understand and use.

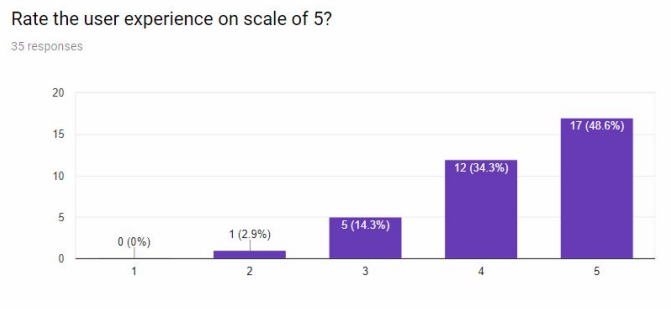

Figure 19: Responses for Question 5

Question 6 : Will you use this web application in future?
We asked this question to get the user retention rate

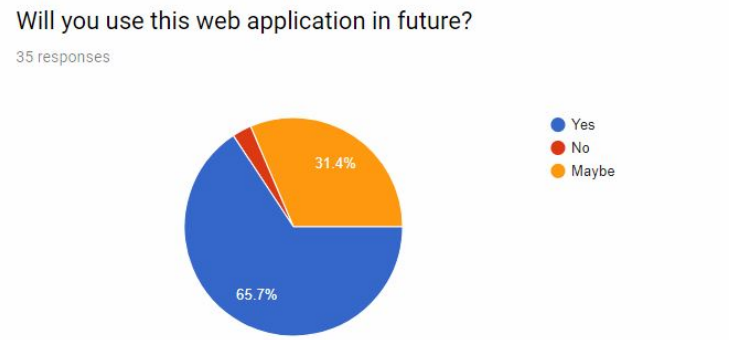

Figure 20: Responses for Question 6

6.4 Observation

Based on the 35 evaluations we can observe that the majority of people found the idea appealing and intend to use the application for day to day ride sharing as the application is user friendly and intuitive. As per the survey the most popular features are auto-completing the address fields, the search results are sortable on all columns and receiving real-time emails when someone joins their plan. Some of the features that would like to have are the ability to edit existing plans and have a way to communicate with other members in the plan integrated in the application. This would be a good start point for the team that takes over this project to work-on.

6.4.1 Google Analytic

Note: The following inferences are based on the 700+ page views on our web application since we enabled analytic.

The intended platform for the web application was mobile devices. However, the reports from Google indicated otherwise as 82.35% of our current users accessed the website from desktop. This shows that our efforts should be directed towards enhancing the user experience more on the desktop version.

The report also showed that Chrome is the primary browser on which the website is accessed, thus we can restrict our efforts on adding support features for less modern browsers.

The home page (search plan page) was the one where users spent most of the time, averaging just 1 minute. This is a good indicator that the detail collection process for the plan is short and simple.

7. FUTURE WORK

7.1 Edit or Cancel trips

Currently the system does not support editing or canceling a trip. However, a user should be able to make changes to his/her upcoming trips. After an update is done, a notification should be sent to everyone involved in the plan.

7.2 Built-in chat

A group chat interface enabling the matched travellers to communicate with each other to figure out the logistics of the travel. Travellers can meet on a common spot, or adjust the travel route to pick fellow travellers on the way. Instant messaging chats are also quicker and hassle-free, compared to email communication.

7.3 Travel Cost estimation

With the help of APIs provided by Uber[5] and Lyft[1] we can provide an estimate cost of the trip based on the type of cars available and the geospatial center for both source and destination location. This can be extended to other car rental services based on the availability of data/API.

7.4 Rating and feedback

Users can be given an option to rate their ride, and even fellow travellers. Thereby, every user will have a rating that will be visible to other users. Users can use this information to decide whether they want to join an existing trip, or create their own.

7.5 Expense tracking

Expense tracking apps such as Splitwise can be integrated with our system, for splitting bills such as gas, tolls or any other expenses incurred during the trip.

7.6 Payments integration

Existing digital wallets such as PayPal, Venmo or Circle can be integrated for quick and easy settlements of money.

8. CONCLUSION

Currently Uber and Lyft are leading the ride sharing market. However, the focus of their operations is mainly in a few selected metropolitan cities. Social media channels such as Facebook and Whatsapp provide a communication platform for planning a ride but they are not specifically build for this because they are not time efficient and are inconvenient to use. WolfPool service focuses on ride planning service primarily in the suburban areas and other cities with features to create and view listing of available rides. Also, WolfPool is designed keeping campuses and students in mind for security and safety concerns. Considering the disadvantages of other platforms, the features provided by WolfPool would be an ideal way to plan rides.

9. REFERENCES

[1] Availability - Ride Estimates. https://developer.lyft.com/v1/reference#availability-ride-estimates. [Online; accessed 1-February-2018].

[2] AWS Elastic Beanstalk. https://aws.amazon.com/elasticbeanstalk/. [Online; accessed 31-January-2018].

[3] AWS vs Heroku: Cloud Platform Comparison for 2017. https://hackernoon.com/aws-vs-heroku-cloud-platform-comparison-for-2017-5f2194c0673e. [Online; accessed 1-February-2018].

[4] Cloud platforms compared: Heroku, AWS, Azure, Digital Ocean, Google App Engine. http://selbielabs.com/cloud-platforms-compared/. [Online; accessed 1-February-2018].

[5] Developers | Uber. https://developer.uber.com/docs/riders/references/api/. [Online; accessed 1-February-2018].

[6] Getting Started | Google Maps Geocoding API. https://goo.gl/akCtYL. [Online; accessed 5-March-2018].

[7] Heroku vs. Google App Engine vs. AWS Elastic Beanstalk. https://stackshare.io/stackups/aws-elastic-beanstalk-vs-google-app-engine-vs-heroku. [Online; accessed 1-February-2018].

[8] HTML5 Geolocation. https://www.w3schools.com/html/html5_geolocation.asp. [Online; accessed 6-March-2018].

[9] Mean Stack. https://www.brainvire.com/basic-guide-mean-stack. [Online; accessed 5-March-2018].

[10] MongoDB vs MySQL. https://www.mongodb.com/compare/mongodb-mysql. [Online; accessed 1-February-2018].

[11] Node.js. https://nodejs.org. [Online; accessed 31-January-2018].

[12] I. Lunden. Disrupt NY 2016. https://techcrunch.com/2016/05/10/uber-says-that-20-of-its-rides-globally-are-now-on-uber-pool/, 2016. [Online; accessed 25-January-2018].

[13] P. Pham. UberPool Marketing Plan. https://www.slideshare.net/PhuongPham59/uberpool-mkt-plan-final-paper, 2014. [Online; accessed 25-January-2018].

[14] W. Ting. Let us help with the commute: introducing uberPOOL in the East Bay. https://newsroom.uber.com/us-california/uberpool-in-oakland-and-other-parts-of-the-east-bay/, 2016. [Online; accessed 25-January-2018].

10. CHITS

XBO
AQW
LZC
WKJ
SQO
BKK
ZFR
FTU
RSL
NUH
QWG
HHX
ROD
QJW
JHC
HDF
TYL
ZAC
ANS
ANF
HTB
KND
KXG
TKX
DQC
JPA
OTD
YVL
KDC
WVQ
OIX
LAA
XRR
EQL
KTJ