WolfPool Web Application

Arpita Shekhar
ashekha@ncsu.edu
Computer Science

Aishwarya Sundararajan
asundar2@ncsu.edu
Computer Science

Amulya Varote
avarote@ncsu.edu
Computer Science

Bhavik Patel
bcpatel@ncsu.edu
Computer Science

ABSTRACT

Software Engineering is the process in which the software is developed in terms of versions. The WolfPool web application already has main features like joining the plan,creating the plan, and updating the plan. The software application is improved by adding basic functionalities like updating and deleting the plans, estimating the costs, and integration of Splitwise and PayPal. Almost the same software architecture is used to implement the additional functionalities. There is a small modification in the schema.

1. PREVIOUS WORK

The previous WolfPool version was developed using the ”MEAN” stack. MEAN stack is the JavaScript software stack which is used to develop the dynamic applications. MongoDB was used for the database, Node JS and Express for the implementation of the functionalities. The frontend was developed using HTML and CSS.

The previous work included the following features:

1. Registration and Login: The user has to register for the application to use the services. The email verification functionality was implemented using ”Mailjet”. The user can login to the application after verify the email id.

2.Joining the existing plan: Once the users specify the Pick-up point and the destination, the database is checked to identify if already existing plans solve the needs of the users and have sufficient vacancy as per the number of people the user specified. If there are no similar plans where these users can enroll themselves in, they are prompted to create a new plan.

3. Creating the plan: As previously mentioned, a new plan is created to serve the needs of users who do not find similar plans.

4.Profile updation: The user is provided with a feature to update the profile.

5.Help/FAQ: The Help/FAQ page provides detailed instructions on how to navigate through the application to perform various tasks such as enroll into a plan, check Splitwise balance, get fare estimation, and view old or existing plans.

1.1 MOTIVATION

WolfPool application was built with limited functionalities. We thoroughly studied the future scope of the previous team’s implementation. We observed that the web application like WolfPool should have the features described in the future scope. The survey was conducted to understand the user’s perspective about the application. We can observe from the below results in Figure 1 that all users use Splitwise to track their expenses and use an online payment gateway to make the payment. So we came up with an idea to integrate ”Splitwise” and ”PayPal” with the existing application. Most of the people use Lyft/Uber for carpooling. Therefore, we came up with a functionality to show the estimated costs to the users.

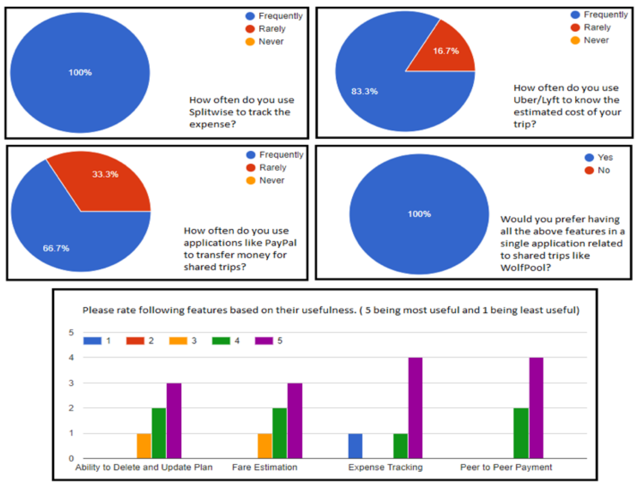


Figure 1: Initial Survey Results

2. SETUP

The readme file written by the previous team provided elaborate steps to setup the working environment.

3. SOFTWARE REQUIREMENT SPECIFICATION

A software requirements specification describes the web application to be developed. It has functional and non-functional requirements of the WolfPool application, including a set of use cases that describe user interactions that the web application must provide.

3.1 FUNCTIONAL REQUIREMENTS

1. The user shall be able to verify his email and register for the application.
2. The user shall be able to update the plan created or joined.
3. The user shall be able to delete the plan created or joined.
4. The user shall be able to leave the plan without affecting the others in the plan.
5. The user shall be able to see the estimates for a particular trip plan.
6. The user shall be able to get redirected to the Splitwise website to track the expenses.
7. The user shall be able to get redirected to the PayPal website to make the peer to peer online payment.

3.2 NON-FUNCTIONAL REQUIREMENTS

1. Security: The security feature has been implemented by sending email to the user to verify the email id. The password of the user is encrypted and is then stored in the database.
2. Usability: The application provides better User Interface to the user to update and delete the created and joined plans.
3. Scalability: The application has been developed by using MongoDB which is very scalable. On an average, at any point in time, there are only ten thousands of students looking for transportation in Raleigh for grocery shopping and many more. Our application can easily support the same.
4. Response Time: The verification email delivery happens almost instantaneously, within 5 seconds of registering through the application.
5. Modifiability: The application is designed in such a way that adding additional functionalities and editing the existing ones is easy and less time consuming.
6. Availability: The application has been deployed on Heroku. The web application is available every time the system is running on heroku.

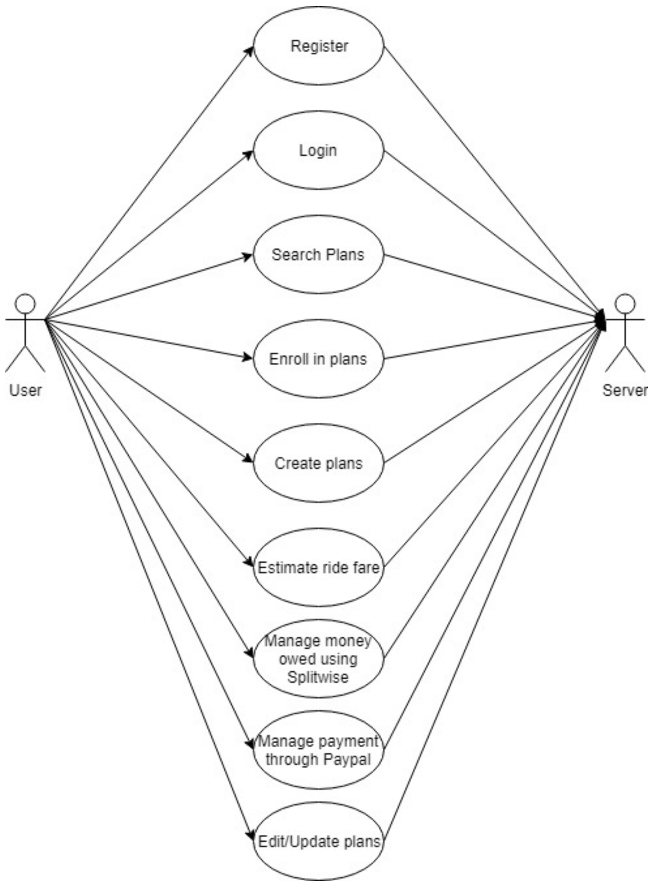


Figure 2: Use case diagram of the application

3.3 USE CASE DESCRIPTION

The use case diagram is the description of the set of functionalities that the user can perform on the system. The Figure 2 shows that the user interacts with the web application. The set of functions that the user performs on the system are registration, Login, join a plan, create a plan, update and delete the plan, view estimated fares, track expenses using Splitwise and pay online using PayPal.

3.4 MODIFIED SYSTEM ARCHITECTURE

The Figure 3 below shows the modifications done to the existing application architecture.

1. MEAN Stack: We have used jQuery instead of Angular.js because the user interface of the application is not too heavy. For our usage, implementation using jQuery was easier as compared to spending time to learn Angular and then proceed with developing the features.
2. Heroku Web Service: Heroku is a cloud platform based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps. Heroku runs the application inside dynos - smart containers on a

reliable, fully managed runtime environment. After deploying the code, build system produces an app that's ready for execution. The system and language stacks are monitored, patched, and upgraded, so it's always ready and up-to-date. The runtime keeps apps running without any manual intervention.
Firstly, it is important to understand the needs of our application while choosing the deployment platform. Email verification is one of the basic features of any application. We migrated from the slow responding Mailjet module (only npm module supported by AWS for email verification) to the much faster Nodemailer module. But we were able to achieve the same only by using Heroku instead of Elastic Beanstalk. The other advantges of using Heroku over AWS Elastic Beanstalk are shown in Figure 4.

3. Database: We are using the existing NOSQL database, MongoDB which was used by the previous team.

4. APPLICATION FLOW AND FEATURES

The users have to register to use the WolfPool web application services. The verification email will be sent and the users have to verify it by clicking on the verification link emailed to them. In order to enroll into a plan, the user enters details like date, time, source address, destination address, and number of people. If the same plan exists then the users may or may not choose to join the existing plan. Otherwise the users are provided with an option to create a plan. The users can see the respective trips. The trips are displayed in the tabular format. The user can update the trip by clicking on "Update" and delete the plan by clicking on "Delete" which are present beside every trip. The user checks the estimated fare for a particular distance by clicking on "Get Estimates". The users can track the expense using Splitwise and make the payment using PayPal which are integrated with the application. The application flow is shown in the Figure 5.

5. IMPLEMENTATION

5.1 Nodemailer

Mailjet SMTP servers have a major disadvantage. The verification email takes 10 to 15 minutes to reach the user's inbox. We replaced the Mailjet email verification module with npm's Nodemailer module. SMTP is the main transport protocol in Nodemailer. The configuration options need to be modified according to the service provider. nodeMailer.createTransport(options,[,defaults])-

1. Options: It is an object that defines connection data.

2. Defaults: It is an object that is going to be merged into every message.

It is a fact that almost every email delivery provider supports SMTP based sending.

Once users signs up, the nodemailer module sends a verification email to the user's registered email address with a hidden randomly generated unique ID appended to the verification URL. Once users click on the verification link from the verification email (as shown in Figure 6), and the randomly generated returned URL matches the sent URL, user's account is verified successfully. After successful verification, the users are redirected to the Login page.

5.2 Fare Estimates

Uber's Price Estimates endpoint returns an estimated price range for each product offered at a given location. The price estimated is provided as a formatted string with the full price range and the localized currency symbol. The response also includes low and high estimates and the ISO currency codes. The ride request estimate function is available in the Uber API. The request estimate has POST parameters like product_id, start_latitude, start_longitude, start_place_id, end_longitude, end_place_id and also accounts for surge pricing.

In addition, the Lyft fare estimator has also been used. The required request parameters for Lyft API are the start latitude and start longitude, all the other parameters are optional. The Lyft API returns the Prime Time Pricing if the destination parameters are not supplied.

The following parameters are displayed through fare estimation as shown in the Figure 7. It lists the prices for all types of Cabs available within Uber and Lyft.

5.3 Edit and Delete Plan:

The previous implementation allowed the users to create plans and join existing plans by entering the source and destination. The plan model had the schema as shown in Figure 9. The email IDs of the users registered for a particular plan was stored in a list. Also, total number of people on that plan was stored as a separate parameter. In order to be able to edit a plan, we also needed to know the number of people travelling per participant. Hence, we changed the list to an object which stored the email ID of the person on plan in addition to the number of people as shown in figure 10.

Update plan feature allows the users to modify the number of participants per user. If the user decides to leave the plan, he can simply update the number of people to 0, which updates his participation and also allows other users to continue using the plan. If there are no other users for the plan, updating the participation to 0, will automatically delete the plan.

In case the plan is no longer active and participation of all the users is 0, user can also choose to delete the plan.

The update/delete view is shown in the Figure 8.

5.4 PayPal Integration

Payment integration is done mainly to ensure that the uses of our application can settle payments owed to others through our website. We faced quite a bit of challenge

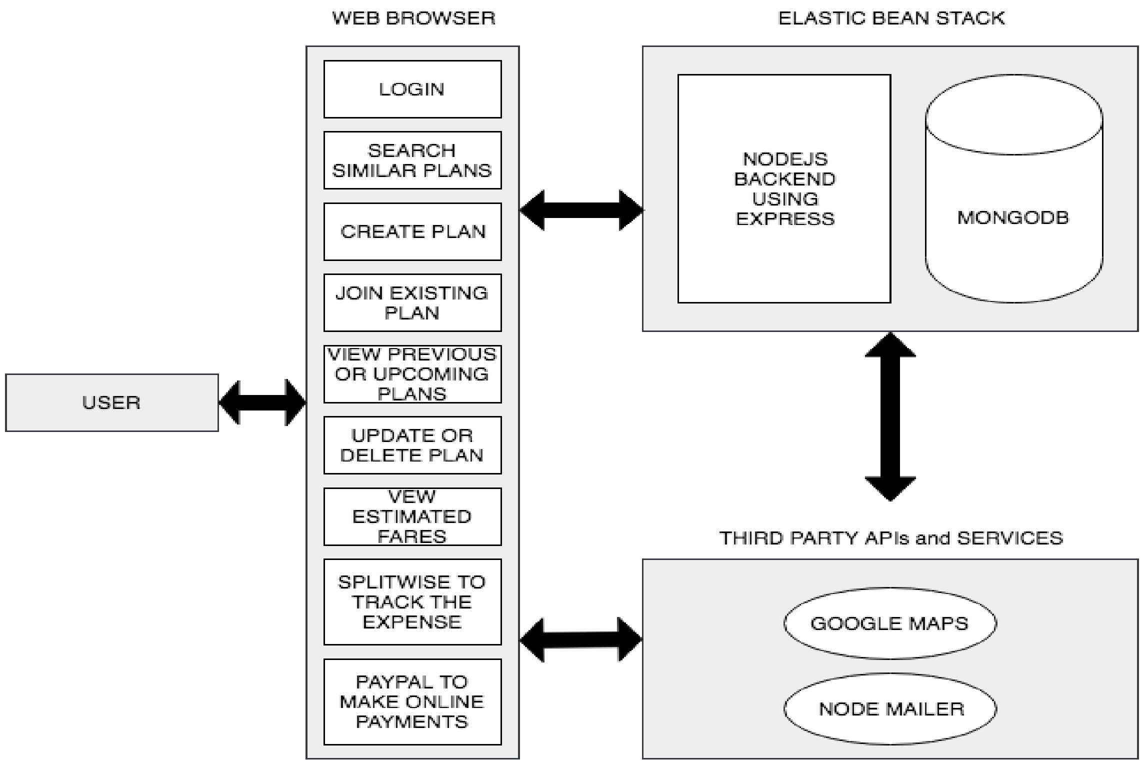


Figure 3: Modified system architecture

looking for right Peer to Peer payment API. Due to lack of time towards the end, we integrated the paypal website login in our application. After completing the payment, users can resume using the application.

5.5 Splitwise Integration

The application is built mainly to track the transportation expenses, the transportation tracking expense feature can be implemented without integrating with the Splitwise. For example, if a person has to pay the other person for both grocery and transportation, it is difficult to track the expense of only the transportation cost shared by both of them.

There is no node npm package that integrates with the latest OAuth version, i.e, version 3.0. Hence, direct integration with the Splitwise API is done using npm's 'request' package.

The following three-way OAuth handshakes are involved in the OAuth implementation of Splitwise as shown in the Figure 11.

1. The Splitwise API provides a consumer key and consumer secret after registering the application.
2. In order to integrate the application with Splitwise, the application requests a temporary token to Splitwise.
3. Splitwise issues the token and the user is redirected to the authorization page from the application.
4. Once the user authorizes the application to access their Splitwise account, Splitwise confirms user authorization and the customer is redirected to the developer customized Callback URL which is appended with the oAuth token and oAuth verifier.
5. The application requests access token to Splitwise and Splitwise issues the same.
6. These tokens are preserved in the database against the corresponding user record.
7. The subsequent handshakes between the userâĂŹs splitwise account and wolfpool happen through the preserved access tokens.

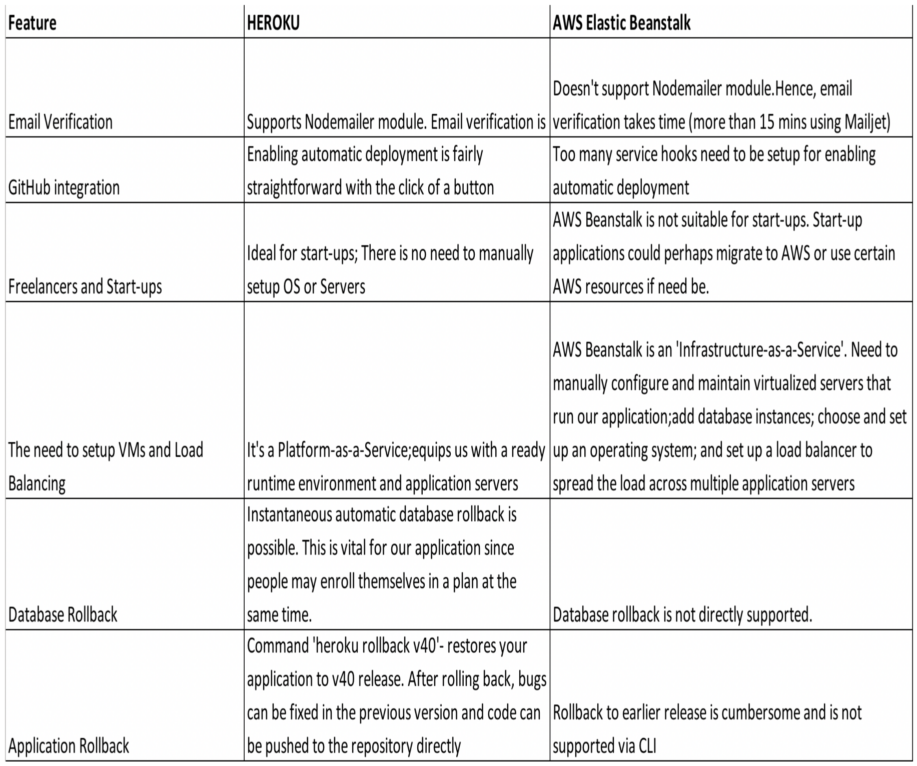
The Splitwise integration with the application is shown in the Figure 12.

6. SOFTWARE EVALUATION

The methods which will be followed to evaluate our proposed software are system Population with known data and beta testing. Both method targets level of satisfaction of the user as a metric for evaluation.

Our target was to add new features to the existing application and make the application ready for use by actual users.

Participants: The participants for our evaluation are graduate students in Computer Science department at NC State University. The rideshare plans were mainly between RDU Airport, Grocery store, SSN office and the residential



Feature	HEROKU	AWS Elastic Beanstalk
Email Verification	Supports Nodemailer module. Email verification is	Doesn't support Nodemailer module.Hence, email verification takes time (more than 15 mins using Mailjet)
GitHub integration	Enabling automatic deployment is fairly straightforward with the click of a button	Too many service hooks need to be setup for enabling automatic deployment
Freelancers and Start-ups	Ideal for start-ups; There is no need to manually setup OS or Servers	AWS Beanstalk is not suitable for start-ups. Start-up applications could perhaps migrate to AWS or use certain AWS resources if need be.
The need to setup VMs and Load Balancing	It's a Platform-as-a-Service;equips us with a ready runtime environment and application servers	AWS Beanstalk is an 'Infrastructure-as-a-Service'. Need to manually configure and maintain virtualized servers that run our application;add database instances; choose and set up an operating system; and set up a load balancer to spread the load across multiple application servers
Database Rollback	Instantaneous automatic database rollback is possible. This is vital for our application since people may enroll themselves in a plan at the same time.	Database rollback is not directly supported.
Application Rollback	Command 'heroku rollback v40'- restores your application to v40 release. After rolling back, bugs can be fixed in the previous version and code can be pushed to the repository directly	Rollback to earlier release is cumbersome and is not supported via CLI

Figure 4: Comparison chart

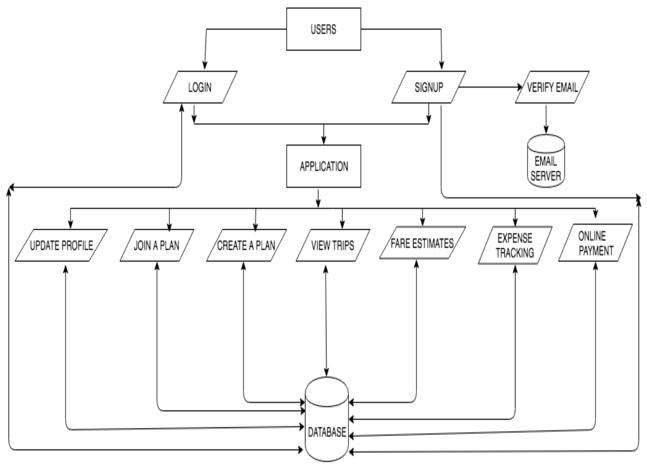


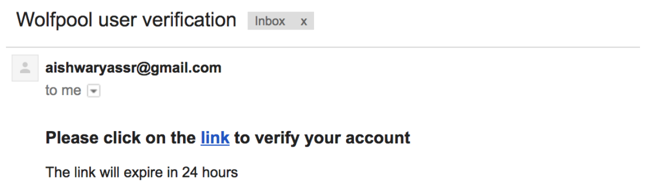
Figure 5: Application flow

areas in and around campus.

6.1 Evaluation Survey Results

Question 1 (Figure 13): Register for the application services. How quickly did you receive the verification email?

We asked this question to find out if the change in email module actually made a difference in the delivery time of the account verification email.



Wolfpool user verification Inbox x

aishwaryassr@gmail.com
to me

Please click on the link to verify your account

The link will expire in 24 hours

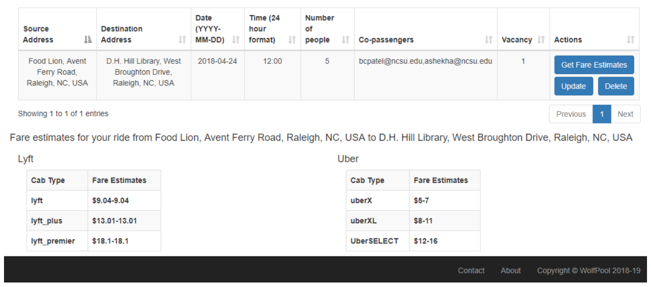
Figure 6: Verification Email

Question 2 (Figure 14): Try updating all 5 plans 'created/joined' above. Track your time for doing this activity.

We asked the users to create or join a total of 5 plans for evaluating the application. The above question was asked to identify if the update functionality was working fine and the response time was not large enough to discourage users from using our application.

Question 3 (Figure 15): Try deleting all 5 plans 'joined/created' above. Track your time for doing this activity. We asked users to check the delete plan functionality using the update plan participation to 0 and by clicking on the 'Delete Plan' button. Recording the time taken in doing so indicates that the functionality works consistently with minimal time.

Question 4 (Figure 16): Click the "Get Fare Estimates" button for any of the plan, are you able to see the estimates?



Source Address	Destination Address	Date (YYYY-MM-DD)	Time (24 hour format)	Number of people	Co-passengers	Vacancy	Actions
Food Lion, Avent Ferry Road, Raleigh, NC, USA	D.H. Hill Library, West Broughton Drive, Raleigh, NC, USA	2018-04-24	12:00	5	bcpatel@ncsu.edu,ashekha@ncsu.edu	1	Get Fare Estimates Update Delete

Showing 1 to 1 of 1 entries

Fare estimates for your ride from Food Lion, Avent Ferry Road, Raleigh, NC, USA to D.H. Hill Library, West Broughton Drive, Raleigh, NC, USA

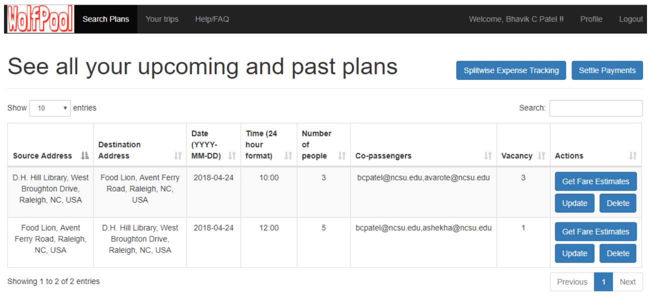
Lyft

Cab Type	Fare Estimates
lyft	$9.04-9.04
lyft_plus	$13.01-13.01
lyft_premier	$18.1-18.1

Uber

Cab Type	Fare Estimates
uberX	$5-7
uberXL	$8-11
UberSELECT	$12-16

Figure 7: Fare Estimates



See all your upcoming and past plans

Source Address	Destination Address	Date (YYYY-MM-DD)	Time (24 hour format)	Number of people	Co-passengers	Vacancy	Actions
D.H. Hill Library, West Broughton Drive, Raleigh, NC, USA	Food Lion, Avent Ferry Road, Raleigh, NC, USA	2018-04-24	10:00	3	bcpatel@ncsu.edu,avarote@ncsu.edu	3	Get Fare Estimates Update Delete
Food Lion, Avent Ferry Road, Raleigh, NC, USA	D.H. Hill Library, West Broughton Drive, Raleigh, NC, USA	2018-04-24	12:00	5	bcpatel@ncsu.edu,ashekha@ncsu.edu	1	Get Fare Estimates Update Delete

Showing 1 to 2 of 2 entries

Figure 8: Trip plans with update and delete option

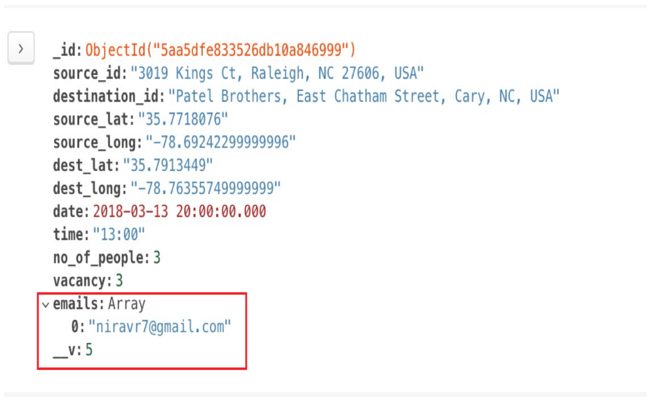
Track time system took to load fare estimates. We asked the users to record the time taken to get fare estimates using our application and the time taken to find out fare estimates using the actual Uber or Lyft application. 55% of the users said it took less than 1-2 seconds to display the fare estimates and it took less than 1 second for 45% of the users, which was very fast as compared to opening the Uber/Lyft application, entering the source/destination and getting the estimates. There were no technical issues while fetching the real-time pricing.

Question 5 (Figure 17): Click the ”Get Fare Estimates” button for all 5 plans one after other. Track cumulative time you took to complete this task. We asked this question to find out how much time can it take at max to plan the whole travel. If the user is able to get fare estimates for all the plans he is enrolled in, it indicates that the response time of fetching the fares is pretty good.

Question 6 (Figure 18): For the same pair of source and destination, validate the fare estimate using the Uber/Lyft application for all 5 plans. How many of 5 plans were having accurate fare estimation?

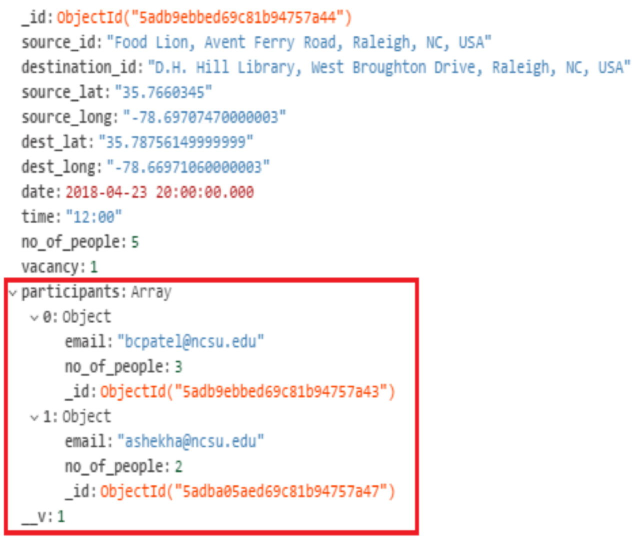
We asked this question to find out how accurate were the fare estimates displayed on our application. Users were asked to get estimates for all the 5 plans they had joined previously and track the accuracy of the displayed fare. 70% of the users got same price range for all the 5 plans they had joined and 30% of the users said they got exact price estimates for 4 plans.

Question 7 (Figure 19): Navigate to ’Your Trips’ page and access Splitwise/Paypal to manage your expenses.



```
_id: ObjectId("5aa5dfe833526db10a846999")
source_id: "3019 Kings Ct, Raleigh, NC 27606, USA"
destination_id: "Patel Brothers, East Chatham Street, Cary, NC, USA"
source_lat: "35.7718076"
source_long: "-78.69242299999996"
dest_lat: "35.7913449"
dest_long: "-78.76355749999999"
date: 2018-03-13 20:00:00.000
time: "13:00"
no_of_people: 3
vacancy: 3
emails: Array
    0: "niravr7@gmail.com"
__v: 5
```

Figure 9: Schema of previous implementation



```
_id: ObjectId("5adb9ebbed69c81b94757a44")
source_id: "Food Lion, Avent Ferry Road, Raleigh, NC, USA"
destination_id: "D.H. Hill Library, West Broughton Drive, Raleigh, NC, USA"
source_lat: "35.7660345"
source_long: "-78.69707470000003"
dest_lat: "35.78756149999999"
dest_long: "-78.66971060000003"
date: 2018-04-23 20:00:00.000
time: "12:00"
no_of_people: 5
vacancy: 1
participants: Array
  0: Object
      email: "bcpatel@ncsu.edu"
      no_of_people: 3
      _id: ObjectId("5adb9ebbed69c81b94757a43")
  1: Object
      email: "ashekha@ncsu.edu"
      no_of_people: 2
      _id: ObjectId("5adba05aed69c81b94757a47")
__v: 1
```

Figure 10: Schema of changed implementation

Were you able to login to your respective accounts?

We asked this question to find out if Splitwise and Paypal was accessible from our application and was working as expected. Everyone who used the application was able to split the expense and manage payment through their respective accounts.

Apart from letting the userâĂŹs evaluate our application we also wanted to evaluate our applicationâĂŹs performance using Google Analytics. Our aim was to record the increase in number of people who use our application as compared to the previous application and also the amount of time a user spends on average using our application.

Our proposed method for calculating the performance was as follows:

WolfPool Engagement Score = 3 * (v2 - v1) + 5 * (p2 - p1)
Performance Score = Number of registrations in WolfPool phase 2 - Number of registrations in WolfPool phase 1

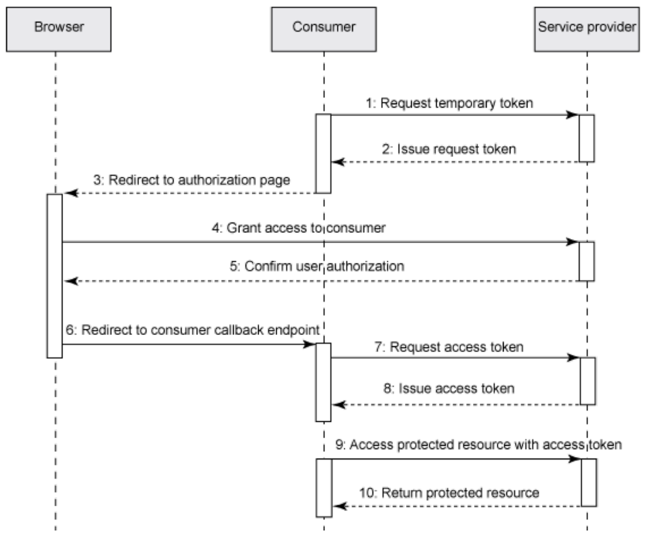
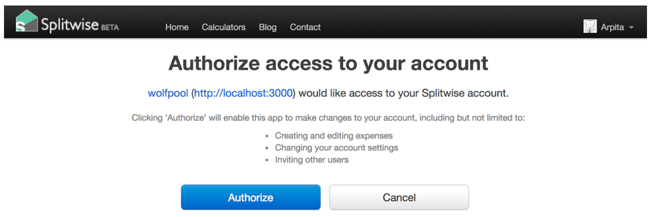

Browser
Consumer
Service provider
1: Request temporary token
2: Issue request token
3: Redirect to authorization page
4: Grant access to consumer
5: Confirm user authorization
6: Redirect to consumer callback endpoint
7: Request access token
8: Issue access token
9: Access protected resource with access token
10: Return protected resource

Figure 11: Splitwise parameters



Authorize access to your account

wolfpool (http://localhost:3000) would like access to your Splitwise account.

Clicking 'Authorize' will enable this app to make changes to your account, including but not limited to:

- Creating and editing expenses
- Changing your account settings
- Inviting other users

Figure 12: Splitwise integration with the application

where:
v1 = Number of visits to WolfPool phase 1
v2 = Number of visits to WolfPool phase 2

p1 = Number of successful plan enrollments with Wolf-Pool phase 1
p2 = Number of successful plan enrollments with WolfPool phase 2

If the Engagement Score turned out to be positive, then our effort to make the application would have been fruitful. A large Performance Score and a small Engagement score would indicate that many people are willing to register for the application but refrain from frequently enrolling in various plans. This would indicate that the effect of integrating these new features did not have a significant impact on the application. Implying that such users usually enroll in plans for which they aren't able to find other users.

However, the evaluation experiment did not turn out as expected. We could not gather substantial data for the old as well as new application to conclude or even compare the performance metrics. The small set of data that we gathered would not prove to be a correct indicator of the application's performance.

1. Register for the application services. How quickly did you receive the verification email?

20 responses

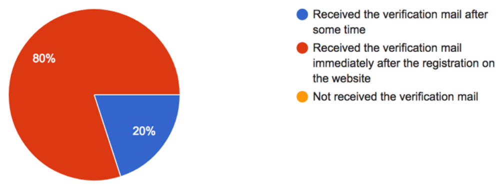


Figure 13: Question 1 results

2. Try updating all 5 plans 'created/joined' above. Track your time for doing this activity.

20 responses

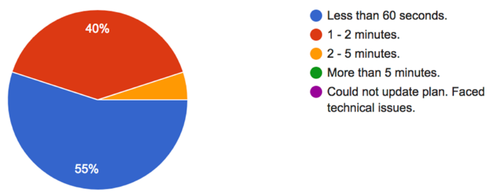


Figure 14: Question 2 results

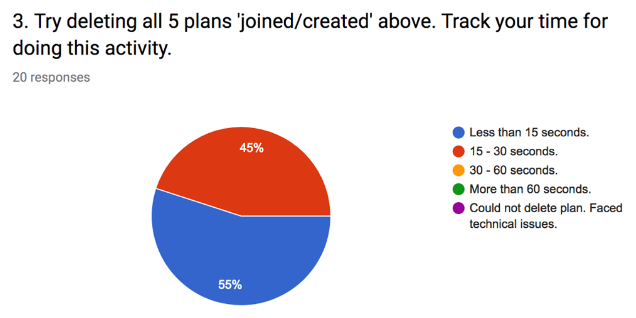
7. CHALLENGES FACED

We faced challenges while integrating Peer to Peer payment with our website to handle expense payoffs after sharing the ride. First on our list was Paypal which is one of the longest-lived payment services.

They support P2P via 'Adaptive Payment' which is a currently a limited release product, restricted to selected partners for approved use cases only and is not available for new integrations.

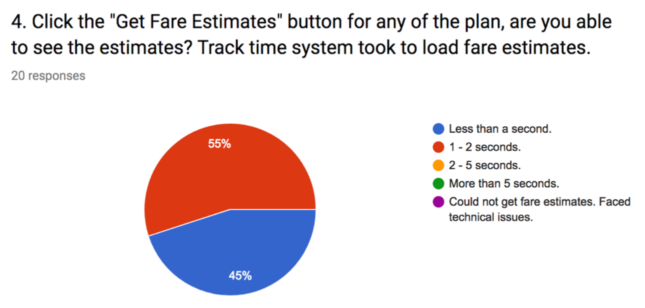
Second we tried looking for Square Cash API but they don't have any public APIs for P2P. However, they do have APIs that allow merchant to process payments which was not suitable for our use case. Venmo's developer API discontinued their support to new developers. Google Wallet has an API for Android Pay which was also not applicable for our particular use case.

8. TIMELINE


3. Try deleting all 5 plans 'joined/created' above. Track your time for doing this activity.

20 responses

Figure 15: Question 3 results


4. Click the "Get Fare Estimates" button for any of the plan, are you able to see the estimates? Track time system took to load fare estimates.

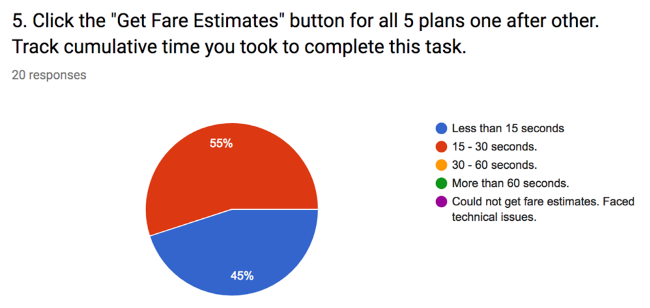
20 responses

Figure 16: Question 4 results

Timeline	Task	Description
3/29/2018	Report 2a Submission	Gathering requirements, surveying and determining the feasibility.
4/1/2018	Study API(PayPal, Splitwise, Uber,Lyft).	Identify the useful modules of API
4/2/2018	User Interface Design	Plan and Decide the User Interface.
4/18/2018	Working Prototype	Develop a working prototype of the system
4/20/2018	Acceptance testing/Evaluation	Draft the testing report.
4/23/2018	Documentation and Readme File	Report of the project including the readme file on GitHub
4/24/2018	5-minute presentation	Give a presentation on cool factor of the functionalities
5/1/2018	Final report submission	Develop detailed developed report

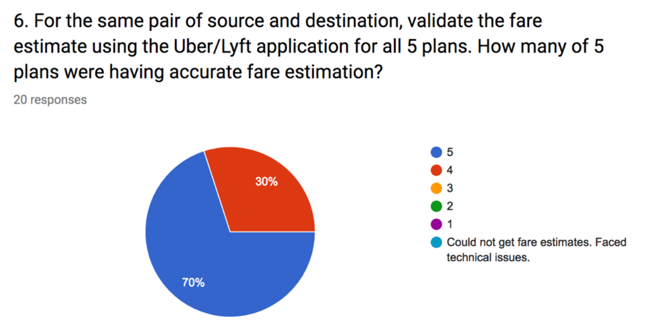
9. CONCLUSION

WolfPool web application is mainly developed for the NC State students after realizing the transportation difficulties faced by international students. Feedback about the application by the NC State students is of at most importance. Based on the feedback given by the students, we decided the most required features in such an application. The ex-


5. Click the "Get Fare Estimates" button for all 5 plans one after other. Track cumulative time you took to complete this task.

20 responses

Figure 17: Question 5 results


6. For the same pair of source and destination, validate the fare estimate using the Uber/Lyft application for all 5 plans. How many of 5 plans were having accurate fare estimation?

20 responses

Figure 18: Question 6 results

isting application which included the features like creating and joining the existing plan is modified to include the features the updating and deleting the plan and integration with Splitwise and PayPal. Software has to be tested thoroughly to gain the user satisfaction. Therefore, the application was tested completely and it was given to the NC State students to use it. Based on the results of the survey, we can conclude that the most of the users really enjoyed using our application.

10. FUTURE SCOPE

1. **Push notification**

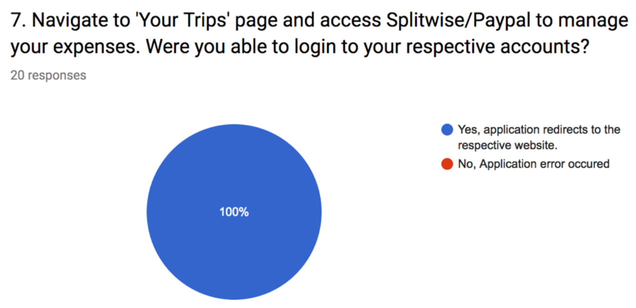
   It is great to have have the notification feature in any application to notify the users about the important tasks and information. For example, users may want to take a ride only if the maximum amount of people (let's say 3) have opted for the plan.

2. **Integration with the social media**

   It is always good to know the people with whom the ride is being shared. Wolfpool offers a platform for strangers to mingle with each other. But users may feel secure if they are paired up with friends opting for the same plan. So, the app could be integrated with Facebook to access users' list of friends and provide priority to pair them up with friends over strangers opting for the same plan.

3. **Implementing the coupon system**

   Any traditional carpooling application like uberPool provides an option for users to refer more people to sign up for the app and provides discounts for gratuity.


7. Navigate to 'Your Trips' page and access Splitwise/Paypal to manage your expenses. Were you able to login to your respective accounts?

20 responses

Figure 19: Question 7 results

Similarly, existing users of the application could be provided a unique referral link with which they can earn discounts by getting more people to sign up for Wolfpool.

4. **Subscription plans**

   Lyft has recently been testing a subscription-based model wherein people pay a monthly price, eg:199 dollars for 30 rides where each ride is worth less than 15 dollars [8]. Such subscription-based models could eventually increase the profit margin of WolfPool and could a feature implemented in WolfPool.

11. REFERENCES

[1] https://github.com/arpitashekhar/WolfPool /blob/master/reports/ WolfPool_March_Final_Report_Group_F.pdf

[2] https://www.splitwise.com/about

[3] https://en.wikipedia.org/wiki/PayPal

[4] https://help.uber.com/h/d2d43bbc-f4bb-4882-b8bb-4bd8acf03a9d

[5] https://s3.amazonaws.com/api.lyft.com/ static/terms.html

[6] https://www.digitaltrends.com/mobile/paypal-vs-google-wallet-vs-venmo-vs-square-cash/

[7] https://www.quora.com/What-is-the-best-bill-splitting-app-for-friends

[8] http://money.cnn.com/2018/03/16/technology/lyft-subscription-plan/index.html

[9] https://developer.paypal.com/docs/integration/direct/express-checkout/integration-jsv4/

12. PROJECT LINKS

Link to our Github repository :
https://github.com/arpitashekhar/WolfPool

Link to our deployed application :
https://wolfpool.herokuapp.com/

13. EVALUATORS TOKEN NUMBERS

1. TZQ
2. SNN
3. ADL
4. BJY
5. ZWK
6. PWU
7. NAS
8. LDH
9. KGM
10. JSH
11. UOZ
12. YCL
13. KXN
14. FSF
15. HVS
16. BEG
17. TPZ
18. ARZ
19. SFL
20. LYH