RidePAD: Car Pool Android Application

Abstract:

Fuel is limited and valuable which makes fuel saving an important aspect to consider. It is expensive and beginning to be unaffordable day by day with recession and global inflation. In addition burning fuel is in no way helping our dying environment. Even if global warming is just a theory the fossil fuel is getting used up fast. The need of the hour is fuel consumption aware technology. Carpooling brings to us such a solution to all our fuel consumption problems. It can help you share the cost of your daily commute, help save your car from wear and tear and also helps the environment. Every big city has dedicated high occupancy vehicle lanes nowadays. On the long run savings from such practices can be significant. In this report we present an easy to share and easy to use carpool android application called RidePAD (Ride Pick-up and Drop). RidePAD aims in providing users with an easy and effective way to share their commuting expenses.

Aziz Nanthaamornphong
Phd Candidate
Software Engineering Group
Department of Computer Science

Debarshi Chatterji
Phd Candidate
Software Engineering Group
Department of Computer Science

Prateek Bahri
Phd Student
Software Engineering Group
Department of Computer Science

1. Introduction.

In an effort to reduce traffic on streets, most of the countries are encouraging car pool systems by having dedicated HOV lanes and reserved parking. In 2009, 10% of the commute travel in the United States was represented by carpooling. The growth of internet and the soaring prices of fuel have recently triggered a sharp rise in the percentage of commuters using some kind of carpooling systems.

1.1 Motivation.

The primary reason for carpool which is a direct effect on a commuter is financial benefits. If it costs $40 to fill your car's fuel tank and you do this every week, multiply that by 50 weeks to total $2,000 a year just in gas. Sharing that cost with other individuals will have a significant impact on your discretionary income.

Figure 1

Studies have proved that annually an average car emits about 10,000 lbs of carbon dioxide and over 600 lbs of carbon monoxide. It speaks a lot about how much pollution is pushed into the atmosphere without even considering the repercussions of such irresponsible waste of natural resources. According to Rideshare.com, "The US could save 33 million gallons of gas-each day-if the average commuting vehicle carried one additional person."

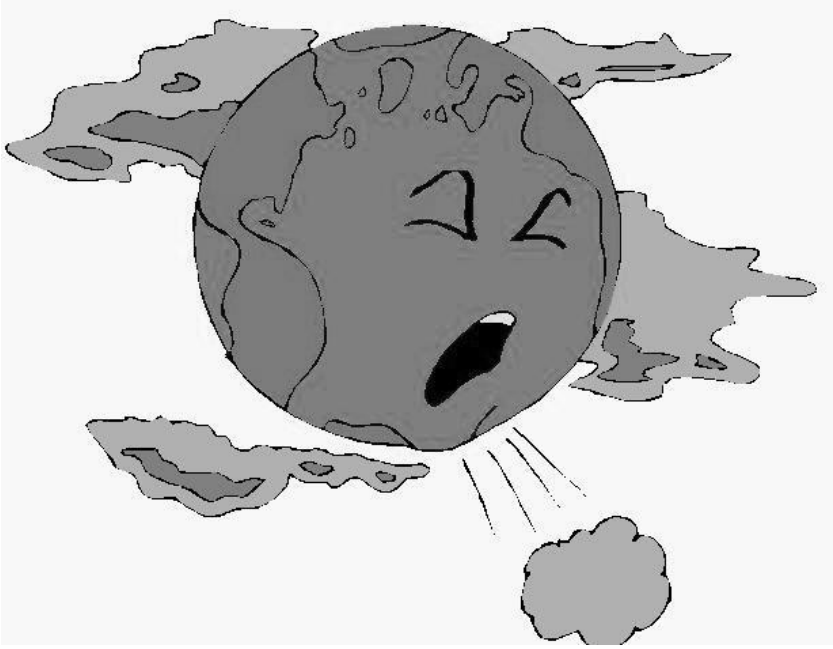


Figure 2

1.2 Our Idea of a successful carpool system.

We wanted our application to be robust and flexible because in practice when we commute more often than not we change the routes frequently. Sometime we change the destinations. So there was a certain amount of flexibility that we wanted to provide with our application. On these grounds we wanted to provide the following features:

- Flexible destinations
- Flexible groups of riders
- Flexibility in terms of gas prices and vehicle used.

We will talk about more about the features of RidePAD in the following sections.

2. Design and Implementation.

The project had two different phases of design. The front end *User Interface* and the backend *Server Architecture*. On a very high level the two phases supported the following functionalities:

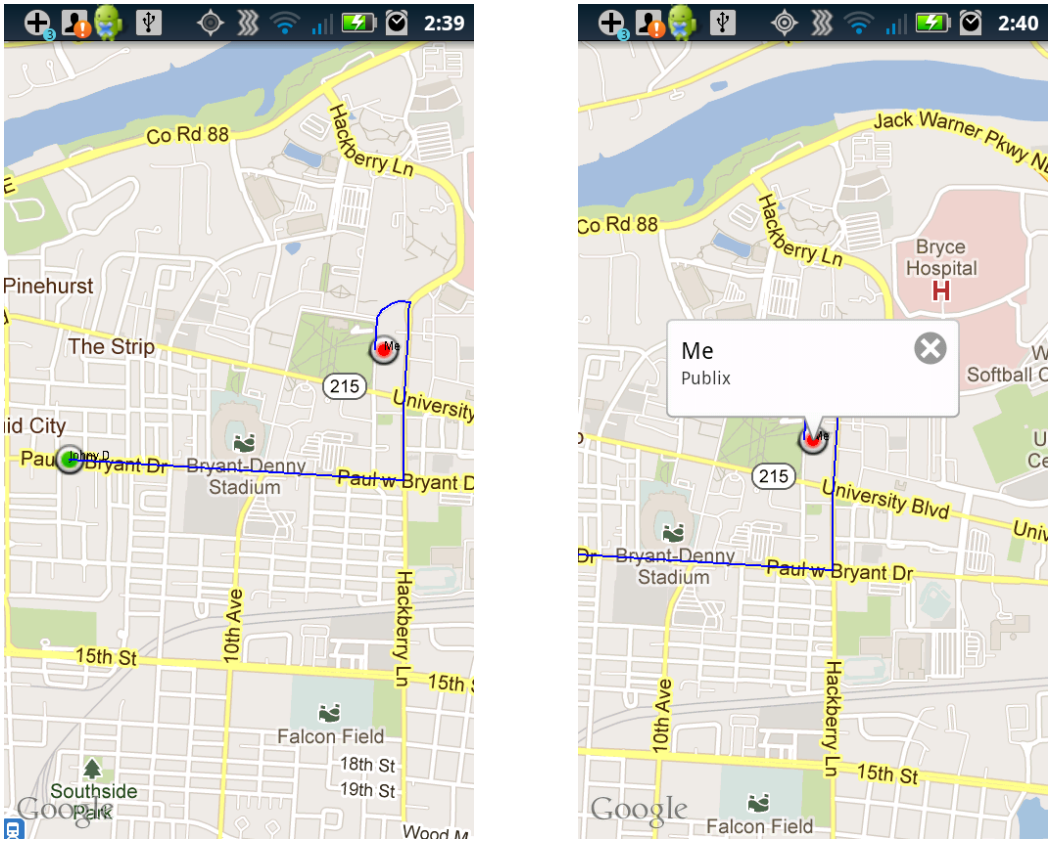
- Flexible routes: Our main aim was not to constrain the amount of travelling by having pre decided destinations. We designed the system such that for the system to work the users do not have to pre decide on the number of miles they are going to travel. So we gave the user the flexibility to take detours.
- Total distance travelled: Total distance was calculated with the help of GPS on the actual number of miles travelled between the START and the END of the tour.

- Participating members: Only the members that accept the request are considered when calculating the trip cost per head.
- User Account: Every user will have a registered account which can be edited at any time.
- Friends list: each user will have a group of friends with whom he usually carpools.
- Offer/Request: Functionality to offer a ride or request a ride.
- Accept/Decline: Functionality to accept or decline a ride.
- Cost of ride: the cost will be calculated as per the parameter miles per gallon for the vehicle in use and cost will be distributed over the number of participating members.

2.1: Specifications.

Our application has the following specifications.

2.1.1. Map functions on Android: We used the Google maps API to track the path of the vehicles using GPS location markers. Once the user accepts a request or another friend selects his request or offer, the user can see the location of his friend(s) on the map as GPS pin. The users can see the place as a popup balloon where a member sent the request from. This maps also tracks the number of miles travelled while a trip.



(a) **(b)**

Figure 3

2.1.2. Web Services: In order to exchange the information among the RidePAD applications, we built the web services that are deployed on the application server. These web services have main features for providing information to users in real-time and storing information into the database. Each RidePAD application needs to communicate to the server over the Internet. Below is the list of service names along with their brief descriptions.

- getLocation – This service provides the current latitude and longitude information of the specified user.
- isUserExist – This service is used to check whether the e-mail address exists and whether it is valid or not.
- setLocation – The application has to call this service for updating the current location of the user.
- Login – This service is used for authorizing the users. The users need to enter the email and password.
- acceptFriend, rejectFriends, and acceptFriends – These services are used for manipulating the acceptance or rejection of the friend invitations.
- register and addUser – The system uses these services for adding a new user.
- getFriends and getFriendsMap – These services are used to manipulate the list of friends that each user has.
- getLatestRequest, getRequest, acceptOffer, rejectOffer, getAccepts, addOffer, addReqeusts – These services are used for handling the request or offer from the users.

2.1.3. Database Support: We used database for maintaining the data including the location of users, user information, authentication information, and transactions. The database is designed to support the web services in the online transaction-processing manner. The system does not allow RidePAD to access to the database directly, which provides a layer of security.

2.1.4. Architecture: We designed the system based on the client – server architecture as shown in figure 4. All the mobile devices will be connected through a common account on the server by using web services. The communication between a device and the server relies on the TCP/IP that is supported by Google android. To send or receive data, the device needs to use either the Wi-Fi or cellular wireless (like 3g, 4g, etc.). The mobile devices do not communicate with each other directly.

2.2. User Interface.

While designing the user interface for RidePad, we tried to closely follow Shneiderman's "Eight Golden Rules of Interface Design". During the design phase we felt that the golden rules proved to be very helpful in designing the best possible user interface. Let us discuss the eight golden rules compared to our design.

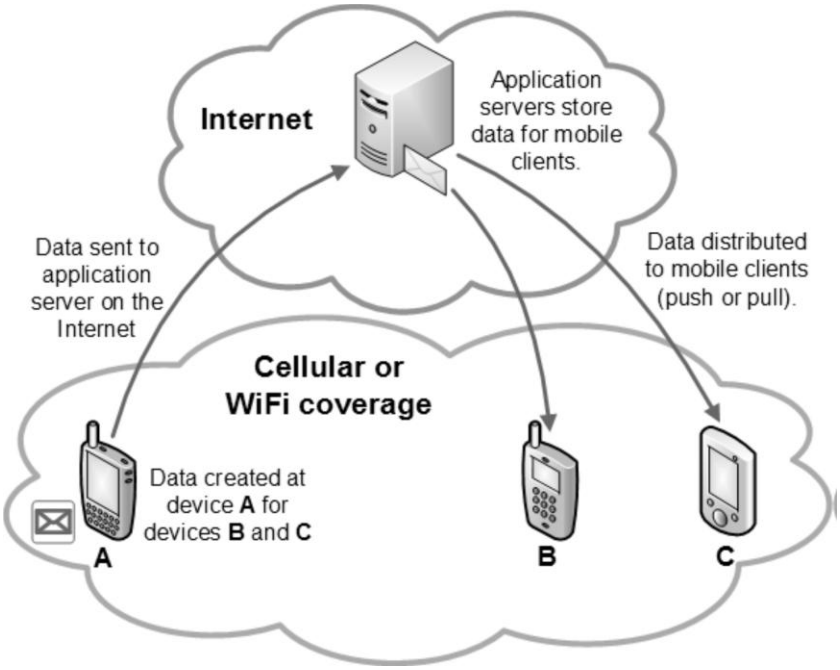


Figure 4

2.2.1. Strive for consistency: Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

Clearly from figure 5 (a, b, c, d, e and f) we can see that all the various pages had a faily similar and consistent design and pattern throughout the application.

(a) **(b)** **(c)**

(d) **(e)** **(f)**

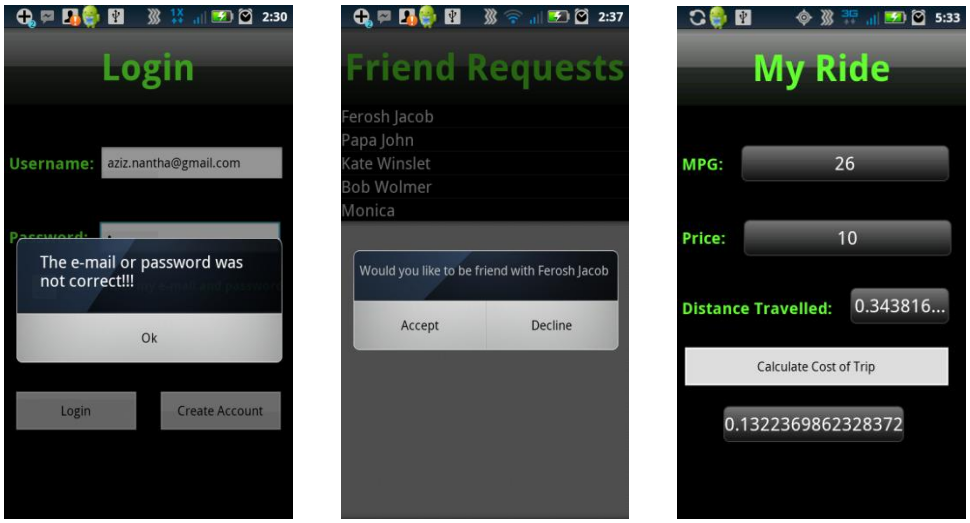
Figure 5

2.2.2. Enable frequent users to use shortcuts: As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction.

While using RidePAD, the user logs into a customized account which makes sure that all the settings are according to the user's last log in. Unless and until there is no reason to change the account settings everything remains same as the last login. Practically speaking there is not much to customize, however there is no need to enter information like miles per gallon unless and until there a need to change them.

2.2.3. Offer informative feedback: For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

System feedback is provided wherever possible, as can be seen in figure 6.



(a) **(b)** **(c)**

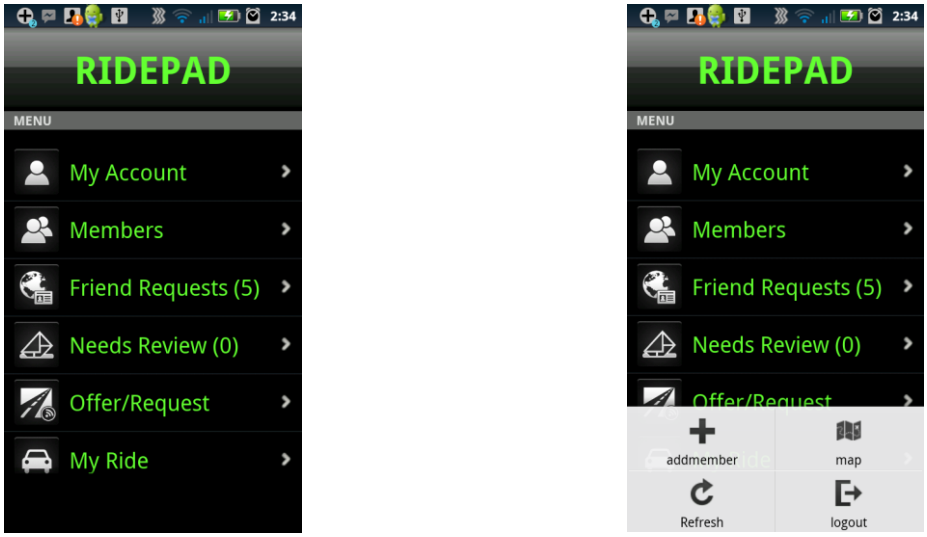
Figure 6

2.2.4. Design dialog to yield closure: Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

As shown in figure 7, the home page navigation provides a very distinct modular control over the whole application. Any process starts from the home page and on completion returns to the home page for the next process.

2.2.5. Offer simple error handling: As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

In case of some error a simple close and restarting the application resets the settings to previous state, which is pretty standard in all mobile apps.



(a) Home Page **(b) Additional Controls**

Figure 7

2.2.6. Permit easy reversal of actions: This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

RidePAD in its present version does not handle any reversal of actions. Once you accept or decline a request, there is no way to undo that action. This is one rule that has not yet been implemented with RidePAD, however, in future versions of it we will be adding features to handle reversal of actions.

2.2.7. Support internal locus of control: Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

In RidePAD, user always has the feeling of the initiator. He always has the control over his part of the process when he offers/requests or accepts/declines.

2.2.8. Reduce short-term memory load: The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be

consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

The application has the capability to remember user id, passwords and all the previously stored entries of what cars to be used, or the address of the user.

2.3. Formal Design.

This section displays the high level design and implementation of the application. We have made available a bunch of design documents that can be found online in our library on the server.

2.3.1. Class Diagrams: As shown in figure 8 we have three class diagrams (a) RidePAD library, (b) RidePAD Applications and (c) RidePAD Webservices that can be found on our server at the following link: http://cs.ua.edu/~aziz/607/ridepad.htm.

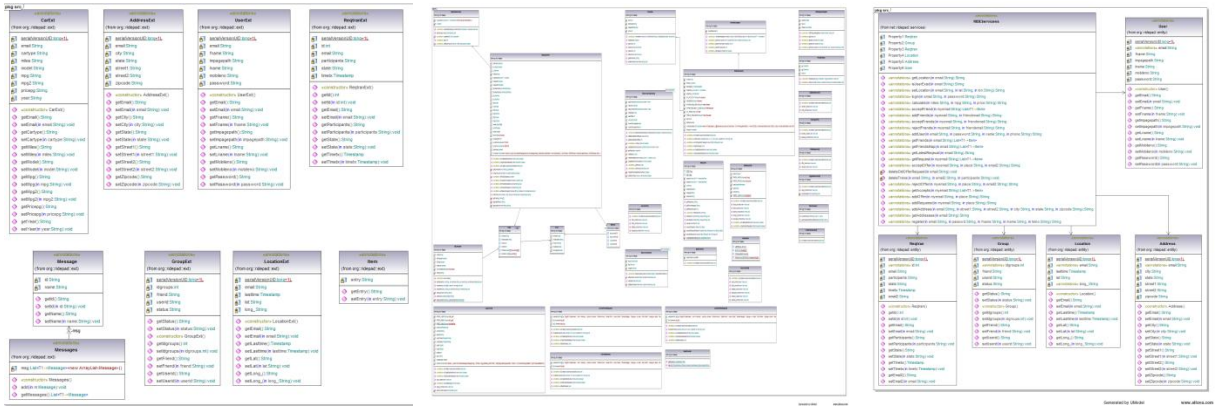


Figure 8: Class Diagrams

2.3.2. Use Case Diagrams: The use case diagram as shown in figure 9 is available on our server at http://cs.ua.edu/~aziz/607/ridepad.htm.

Figure 9: Use Case Diagrams

3. User Study.

Ideally we wanted to run an expansive user study where we could actually perform a number of mock carpools and some try to break the application. Until and unless we do that we cannot have the total confidence on our application if it is going to be practically working and error free. However due to time constraints and the fact that this class was about designing a UI which would appeal to a user we planned to perform a satisfaction survey.

3.1. Satisfaction Survey.

The survey was done and analyzed in a way so that there is complete anonymity as far as the participants are concerned. Figure 10 shows our survey form. The users were expected to fill it after getting them acquainted with the application.

3.1.1. Participant pool: We randomly selected 10 participants between the age group of 24 to 33. All the participants are graduate students from various departments of the University of Alabama. All the participants knew one of the project members in one way or other, which raises the biggest threat to validity to the results of the survey. However, that was the best we can do due to time constraints. The eligibility for participation in the survey was that they should own a smart phone.

3.1.2. The task: Every participant was given 15 minutes to get acquainted with the User interface.

- They were asked to register to RidePAD;
- Login;
- Send a few friend requests to pre listed mock users which one of the project member accepted.
- Then they were asked to spend the rest of the time cursing through the application and its functions.

Once they are well acquainted with the application they were provided the observation questionnaire which took about another 5 minutes to complete.

3.2. Results.

Out of the 10 participants nobody reported to have used a carpool application in the past. So we can say that we had a fairly unbiased subject pool. Usually with these tasks, people who have used a similar tool previously could have walked in with some degree of expectation and bias because of the familiarity with the previously used application.

3.2.1. Familiarity with Android: The next thing considered was the familiarity of the users with Android devices. Except for 2 participants all the other participants had some sort of Android familiarity. In figure 11 '*novice'* users represent participants who had no experience with Android devices, they use other smart phones. '*Acquainted'* represent the participants who have been using Android since less than 6 months. '*Experts'* represent participants who have used Android

for over a year. '*Developer'* represents the participants with some Android development experience.

RidePAD Observation Questionnaire

Name _____

Have you ever used or are using a carpool application?
__ Yes.
__ No.

What is your previous experience with Android devices? (Check the bottom-most item that applies.)
__ I have never used Android devices.
__ I have started using Android devices very recently in less than 6 months.
__ I have been using Android devices for over 6 months.
__ I have developed software as an Android developer.

Experience with smart phone applications.
Please rate your experience in this section with respect to the following 5-point scale:
(Please include any relevant comments below each section)
1 = Poor;
2 = Below Average;
3 = Average;
4 = Pretty good but needs some improvement;
5 = Very good.

As compared to other major smart phone apps how would you rate RidePAD based on the following criteria?

- Color contrast and clarity 1 2 3 4 5
- Simplicity and ease of use 1 2 3 4 5
- Usability/Efficacy 1 2 3 4 5
- Quick and easy navigation 1 2 3 4 5

Comments: _____
_____
_____

On a 1-5 scale, where 1 means "not at all" and 5 means "most likely":

Is carpooling in today's life a useful idea and something to think about? 1 2 3 4 5
Comments:
_____
_____
_____

If you happen to use a carpool application then would you use RidePAD? 1 2 3 4 5
Comments:
_____
_____
_____

Figure 10: RidePAD Observation Questionnaire

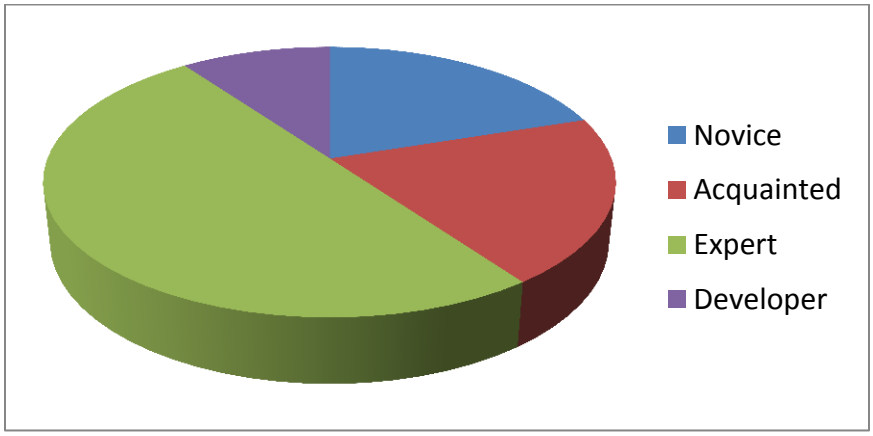


Figure 11: Familiarity with Android devices

3.2.2. User Interface Satisfaction: From the survey we tried to get user feedback from the participants regarding the user interface. Specifically we asked the participants to judge the UI on four grounds, namely clarity, simplicity, usability and navigability. The user response was received on a 5 point Likert scale with 1 being poor and 5 being very good. Figure 12 represents the data in a Kiviat diagram. In Kiviat diagrams, number of edges represents the number of subjects. We can clearly see that the radars are spread outwardly. Since distance from the center is directly proportional to degree of satisfaction, hence we can clearly tell that the participants were fairly satisfied with the user interface of RidePAD.

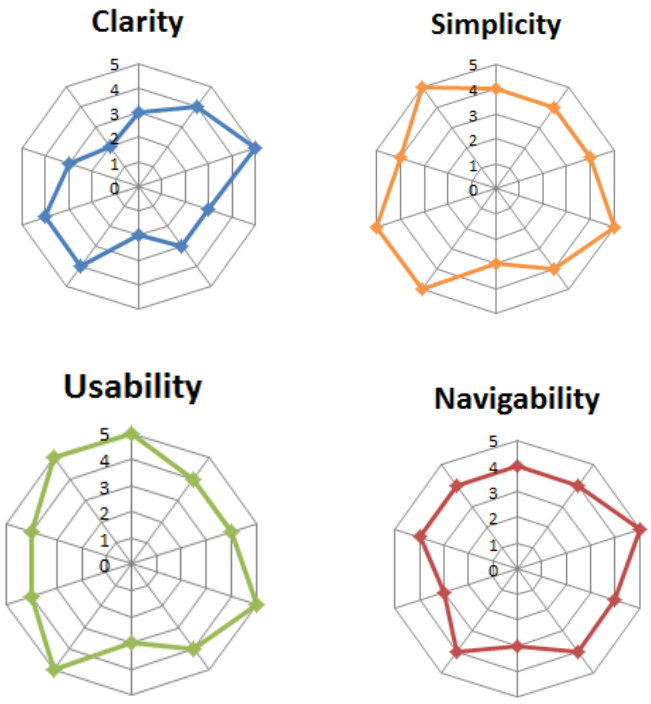


Figure 12: UI Satisfaction

3.2.3. Future of RidePAD: Most of the participants thought that carpooling is a useful idea in today's world. The mean of the responses out of 5 was 3.5 which shows a positive trend. Similarly on asking the users, if they will use RidePAD as a tool the response was even more positive. The mean out of 5 was a staggering 4.6., which clearly indicates that, if and whenever they use a carpool application, they would not have issues with choosing RidePAD.

4. Conclusion.

To conclude, we would say that RidePAD did not fail to satisfy users and surely pointed out the necessity towards thinking about saving fuel not just for the personal reasons. However, having said that, we would like to point out that RidePAD is in its first version and not yet ready for commercial use. A lot of thought process has already gone into making RidePAD able to deliver something extra from the other available carpool applications and a lot more is needed. We seriously hope that if ever we can make RidePAD available commercially as a free software, it would be able to solve a lot of fuel issues.

5. Acknowledgement.

We take this opportunity to thank our participants for giving us their valuable time to provide us with invaluable feedback. We would also like to thank Dr. Randy Smith for his guidance and mentoring and giving us the opportunity to work on this project, which was nothing less than fun. It gave us a chance of trying something different, something that is not mundane as the regular coursework. We learnt to see things differently, a simple button is no more so simple to us, we understand the many altercations that went behind for it to be put at that place saying that particular thing. Last but not the least we would like to thank our friends and families for their endless support.

The source code and other design resources are available online at http://cs.ua.edu/~aziz/607/ridepad.htm