Requirements document for a car pool system

27th February, 2013

1 Contents

1 Introduction

1.1 Purpose

This SRS document describes the functional and non-functional requirements of the carpool system. This document is intended for the designer, developer and the maintainer of the carpool system.

1.2 Scope

The aim of this system is to facilitate commute of LUMS students, staff and faculty to and from campus by providing an online platform where individuals interested in car sharing can meet. It will allow car owners with space in their cars to offer a ride to those who need it, provided their routes are similar.

This initiative will help both lift-owners and lift-takers to economize on travel expenses by sharing the fuel cost. Not only this, but it will also reduce the vehicle fleet on-campus as users would be able to find partners for regular car-sharing using our system.

1.3 Definitions

- **User**: This is a super category which covers

  Driver: A car owner who is willing to provide a lift to someone.

  Passenger: A person requiring a lift

- **Start location**: The location from where a driver is willing to give the lift and the passenger requires the lift.
- **End location**: The destination of both driver and the passenger
- **Lift-share match**: The case where the source and destination of both a driver and a passenger match.

- **Trip ID**: When a lift-share match is found a Trip-ID is sent to both driver and passenger via email, so the trip can be identified and reconfirmed later.
- **Central Database**: A database storing all the lift-share matches and lifts still to be serviced. The database will store three categories: *drivers* that have not been matched with anyone yet, *passengers* that have been not been matched yet, and the *lift-share matches*. The database will also store user-id and passwords.

2 Requirements and Interface

2.1 Functional Requirements

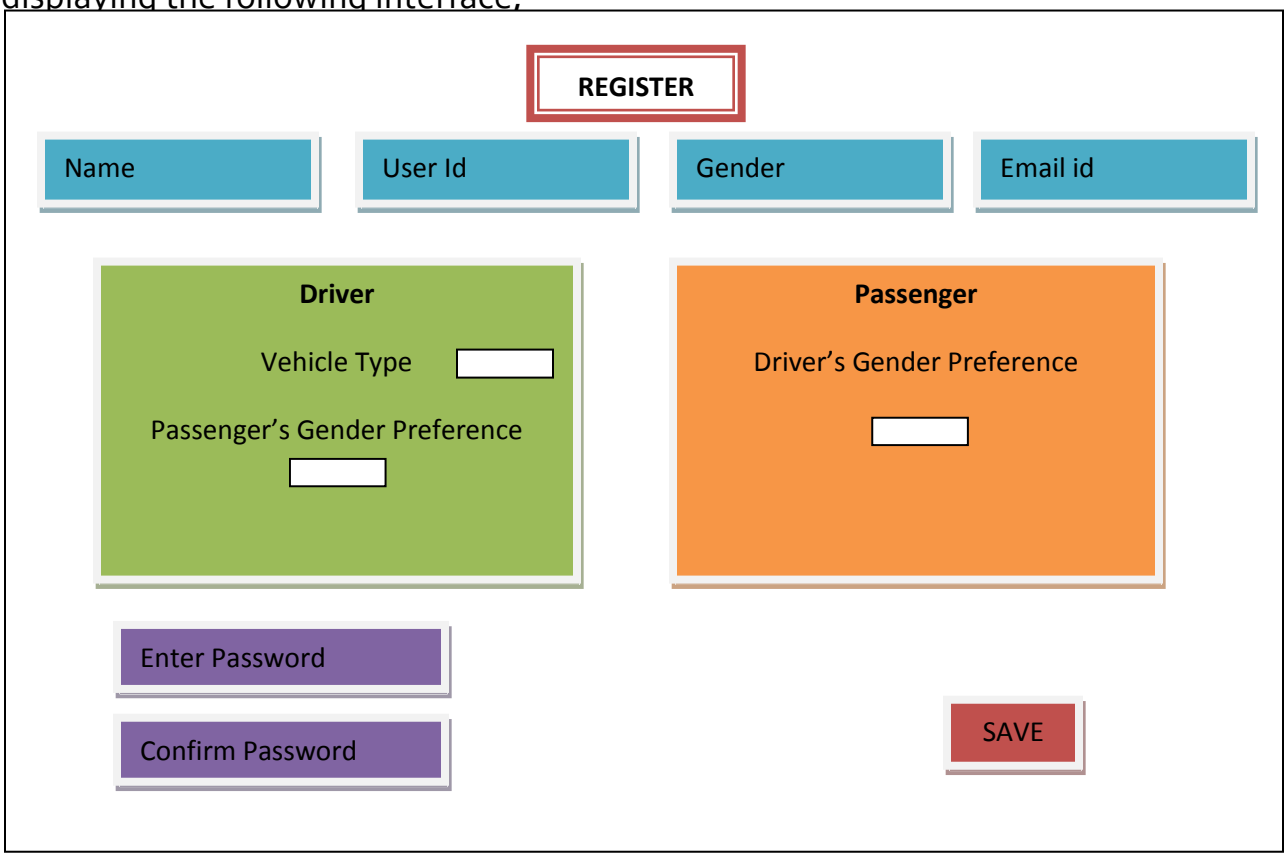
Functional requirement 1

Description

Register users onto the system.

Input

Allow the user to register to save user's permanent information by displaying the following interface;



Processing
Save user information onto the database.

Output
Display a registration confirmation message to the user.

Functional requirement 2

- *Description*
  User login id and authorization mechanism.

- *Input*
  The user will enter his/her id and password.

- *Processing*
  User id and password will be verified against our database

- *Output*
  If the id is valid, display an interface that takes the user to functional requirement 3.

  If the id is not valid, display an error message and prompt the user to try again.

Functional requirement 3

- *Description*
  Display an interface that prompts the user to select from one of the following options;
  i) New Ride
  ii) Reconfirm an Existing Ride

- *Input*
  User selects his/her status.

- *Processing*

Take user to the required interface, according to the option he/she selects.

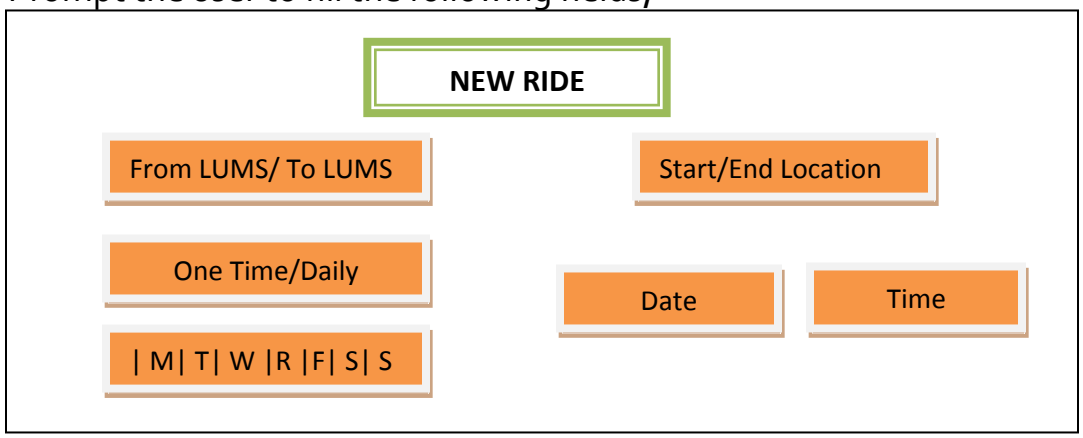
- *Output*
  If the user selects Reconfirm, display an interface that takes user to functional requirement 8.

  If the user chooses New Ride, display an interface that takes the user to functional requirement 4.

Functional Requirement 4

Description
If the user opts for a new ride, get the required information from the user

Input
Prompt the user to fill the following fields;



Processing
Save this information in the database.

Output
Redirect the user to the next page, specified by requirement 5.

Functional Requirement 5

Description
Get user category input

i.) Driver
ii.) Passenger

Input
Allow the user to select his/her category.

Processing
Matching user type to the redirection page.

Output
If the user is a passenger, display a 'Thank you' message and move on to functional requirement 7.

If the user is a driver, redirect the user to another interface specified by requirement 6.

Functional Requirement 6

Description
Get driver input for;

i.) Number of seats available.
ii.) Approximate fuel cost per km.

Input
Allow the user to fill in the required fields.

Processing
Sanity check for fuel cost. Cost should not exceed Rs.15.
Add this information to the database.

Output
Display 'Thank you for keeping the LUMS environment green' message to the user and move on to functional requirement 7.

Functional Requirement 7

Description
Determine lift-share match.

Input
Extracted from database.

Processing
The database maintains three kinds of records; a list of drivers, passengers, and lift-share matches. Every time a user entry is made, the lists are updated. As soon as an entry is added to drivers or passengers, lift-match share is called.

Three cases exist;

- If a match is found, add it to *lift-share matches* list. Update the other two lists by deleting the passenger whose match is found from the passenger list and decrement the seat availability of the driver in the drivers list. If the driver's seats are full, delete it from the driver's list.

- If the trip is cancelled in reconfirmation phase, move the passengers and drivers from lift-shares back to their respective lists.

- If the trip time of passenger or driver has passed, delete the entries from the database.

Output
If lift-share match found, a trip id is generated and automatically emailed to both the driver and the passenger.

If no match is found, do not return anything.

Functional requirement 8

Description
Reconfirmation of lift. If a lift-share match is found, the participants of the ride, i.e., the driver and passenger should reconfirm their availability until before half an hour before the time of the lift-share.

Input
Trip ID

Processing
Check if valid trip ID

Output
If the trip id is not valid, display 'Trip Id invalid 'error message

If the trip id is valid, display new interface asking to confirm or reject the lift-share match.

If the trip id is valid, but the driver or passenger reconfirm after half an hour time limit, display 'Confirmation Time Expired. Trip cancelled' error message and update the database by deleting the driver and passenger from *lift-share matches* and adding them back to the dr

Functional requirement 9

Description
If the user confirms the lift-share match, the status is set to reconfirmed.

Input

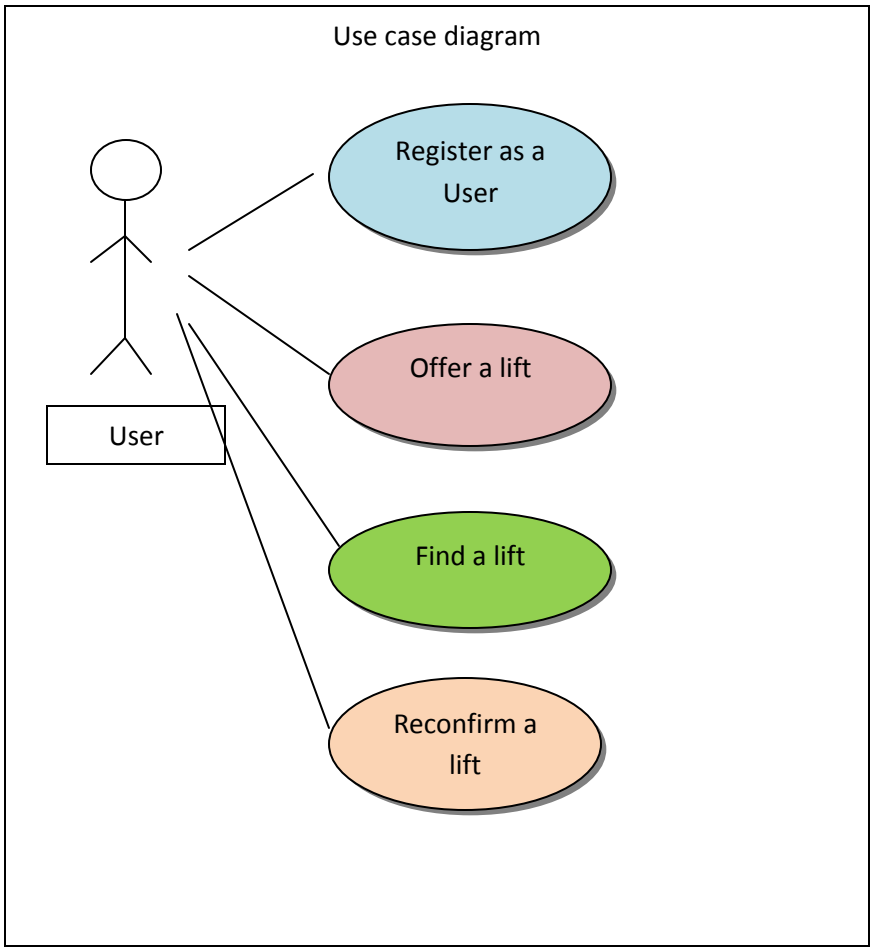
Confirm button.

Processing

Set the status to confirmed by driver or passenger or both.

Output

If the trip id is valid display new interface asking to confirm or reject the lift-share match



2.2 Performance requirements

- **Overlapping Time**
  The system allows user to make only one lift reservation at one time.

- **Multiple Bookings**
  Database consistency should be maintained efficiently so that clash of lift-matches does not occur for the same seat. One seat cannot be assigned to multiple passengers.

- **Match Time**
  System should return a lift-match within 30 seconds or one minute.

- **Window Close**
  If the user closes the window in the middle, the lift should not count.

2.3 System Attributes

- **Availability**
  Ensure that the system is up and working 24 hours.

- **Consistency**
  Ensure that updates to the system are consistent.

- **Security**
  Ensure privacy of information provided by the user.

- **Scalability**
  System has the potential to develop and expand in order to cater to a bigger network of community.

2.4 Additional requirements

Several other features can be incorporated into the system once a working car-pool system has been established.

- **Integration with LUMS Database**

  Initially we will work on synthetic data to create a model of a working car-pool system, but if approved by IST, it will be integrated with the LUMS database to instantiate a fully functional system.

- **Partial Route Matching**

  To start off, our lift-share match algorithm will work for end-to-end matching from start to end location but later we plan to improve it by allowing partial route matches in order to cater to a larger audience and increase mobility through our software.

- **Driver and Passenger Ratings**

  A feature of driver and passenger ratings may also be included, if time permits. This will allow both the driver and the passenger to give feedback related to each other, that can help other users make their choice of ride or riders.

3 Conclusion

If implemented properly, such a system could prove to be invaluable as it would not only be a source of convenience for many but would

also help save resources. There is also a lot of potential for optimization in such a system, which can be considered once the framework has been established.