# RIDESHARE: CONNECTING TRAVELERS

EE 364: Requirements Specification

Yoni Ben-Meshulam
Garrett Cooper

Team Venture

Dr. Bruce McCann

November 7, 2007

## TABLE OF CONTENTS

## LIST OF FIGURES

## 1.0 Introduction

Our product aims to connect drivers who want passengers with passengers who want drivers. It will do so in an efficient manner, which minimizes travel time and distance, as well as taking into account the preferences of customers.

In a time when energy and fuel efficiency pressuring people to leave their car at home and seek alternatives, our rideshare application can help people coordinate rideshare programs and plan trips efficiently and effectively. People who share rides regularly will significantly reduce their ecological footprint and save on travel expenses.

The target audiences can be divided into three types:

- Travelers who are interested in a regularly-scheduled rideshare to their place of work or school, or for their children.
- Travelers who seek one-time partners for longer trips (i.e. cross-state).
- Individuals interested in sending shipments/cargo via rideshare drivers, essentially substituting our service for standard shipping options (USPS, FedEx, etc.).

Many people travel alone in their cars, with much excess capacity wasted. This waste is compounded by the increasing price of gasoline. People who seek alternatives frequently find mass transit to be unreliable and underdeveloped, and must continue to waste their own and the world's resources.

Our solution will enable individuals to communicate their travel needs and find others who are compatible with those needs. It will eliminate much of the coordinating tasks from the rideshare paradigm, and subsequently increase the likelihood that people will choose rideshare. Our hope is that this product will help facilitate a paradigm shift from single passenger travel to a ride sharing culture.

## 2.0 Needs Analysis

In order to successfully implement our rideshare application, it will be necessary for each facet of our system to adhere to a set of requirements. The following are what we consider to be basic requirements for a number of crucial elements of the system.

When matching riders and drivers, we must implement an algorithm which generates the most efficient results, where efficiency is a measure of a number of parameters. The foremost of these is maximal matching, i.e. the highest number of customers receiving rides. Maximal matching is crucial for the results to be effective and useful for our customers. Route lengths and times must also be minimized by the algorithm, and suggested rideshares should be viable, intelligent solutions. Timeliness of results is also important, since an effective result which doesn't complete in a timely manner is useless. This last requirement can be quantified as an algorithm which runs in polynomial-time, as opposed to exponential-time, which is much slower.

The algorithm should also utilize external inputs efficiently. This can be quantified as an algorithm which minimizes route requests to Google-Maps by reducing the search-space, i.e. the number of possible valid matches. For example, if a driver is going from Austin to Houston, we would refrain from evaluating routes for which the riders' beginning and end points aren't in Texas. Those that are in Texas can further be reduced to riders' who are in central Texas, etc. Once this pre-processing has been completed, we would evaluate only those combinations within the limited search space which has been generated.

Our final product will need a server that can support the projected bandwidth and storage requirements. Bandwidth is a measure of the amount of incoming and outgoing data, and storage is the amount of hard-drive space required for storing the application and database, as well as any run-time storage requirements, i.e. temporary information used by the application. The server must also be an application server which is compatible with our implementation methods. These will most likely include an SQL database, PHP scripts, and Javascripts.

We will need an application development environment such as MS Visual Studio or Eclipse, which supports web application development, including PHP, SQL, and JavaScript. Our choice must be evaluated on its reliability, effectiveness, and usability. Within this environment, we will need to enforce good programming practices, including a secure wiki and file server for collaboration, naming conventions, revision control, and modularization of programming tasks.

The final product will also need to meet certain reliability and availability parameters. Reliability is the measure of how stable (or not "buggy") the program is. An unstable program will crash or generate erroneous results, whereas a reliable one will perform with minimal bugs and interruptions. Availability is the measure of how much down-time, measured as the percentage of time during which our website is unavailable, we can expect to have. These will both ultimately depend on our ability to test for problems early on, and be able to quickly fix any bugs.

In order to be a viable application for people to use, we will need to implement a solid user interface. This includes designing an intuitive menu and a friendly graphical interface with usability in mind. We must minimize the number of menu levels, as well as the number of operations required by users.

It is also imperative that we establish application-level security in order to protect our users' privacy, their personal computer, and the application server from malevolent entities. This requires that we use a secure web server and design all incoming and outgoing communications with security concerns taken into account. We must evaluate the web server and implementation methods based on this requirement.

A good testing platform will be necessary in order to evaluate each module of the application. To accomplish this goal we must design the software with testing in mind, also called design-for-test. Each testing requirement must be quantified in terms of our final software architecture.

We will need to include the following testing methods:

- Functionality-testing: evaluates the program inputs and outputs and assesses their correctness. It also includes navigability of the menu and correctness of links within the website.
- Performance-testing: evaluates the timing behavior of the system under various circumstances.
- Stress-testing: approximates the maximum number of users, server-requests, etc. that our application can handle.
- Security-testing: evaluates server-level security and application level security. This allows us to establish a definitive security framework within which we are operating.
- Unit-testing: testing at the level of individual software modules for part or all of the above testing methods.
- Usability-testing: having users enter and navigate the application, and comment on the menus, look and feel, options, and navigability.

All of these requirements will need to be incorporated into a high-level application architecture and module-level implementation. At each point in the design and implementation phases, we will need to assess our adherence to these requirements.

Assuming these needs are met, we will need a broad user-base in order to ensure that all users actually find ride-share solutions. Given our application, the more people we have in our user base, the better the solution we will be able to provide in terms of cost, distance, and compatibility.

## 3.0 Deliverables

At the culmination of the project, we will provide the following set of deliverables, which will encompass the totality of the project, and allow a peer to completely understand our application and implementation methods.

Foremost, we will provide the functional rideshare application running on a server, which users can utilize to find and offer rideshares. We will offer documentation of the high-level software architecture, as well as the low-level module implementation. The actual code will also document the programming structures and functions used for implementing the low-level modules. We will also provide a comprehensive demonstration of this application.

In addition, we will provide a use-case diagram for the application, defined by Wikipedia as "a type of behavioral diagram...designed to present a graphical overview of the functionality provided by a system in terms of actors, their goals — represented as use cases — and any dependencies between those use cases." [1]

In order to demonstrate the practical functionality of the website, we will provide a site map, including the menu architecture and description, as well as a link graph showing the relationships between different pages of the site.

We will include a maintainability document to ensure the proper operation and maintenance of the site. This document will cover the information necessary to implement the application on a server, including a connectivity graph of application components, bandwidth and storage requirements, compatibility issues, and security requirements.

## 4.0 Principles of Operation

The rideshare project will be implemented as a server-based application, connected via the internet to users and third-party service providers such as Google-Maps. The application will utilize an SQL database to store user data, established rideshares, and other information.

The core operating principle of the application is to take a set of ride requests and ride offers and utilize an optimization algorithm to match drivers with riders in an efficient manner. An efficient solution is one which is maximal with regards to the number of customers actually getting rides, and minimal with regards to distance and travel time. In addition, an efficient solution must meet all user preferences, which may include gender, age, or other criteria.

Possible routes will be compared by the application using different criteria, including route efficiency data derived from Google-Map requests, user preferences, and maximal matching guidelines. These results will then be communicated to all relevant parties via email and the application menu. A given solution may be accepted or rejected by the end users, and new requests added. This process will repeat at regular time intervals.

Policies will be in place to determine possible operations and standard procedures for participation in rideshare. These may include rules for cancellation, alteration, and other operations which may affect other users. In addition, policies may change according to different criteria regarding the rideshare, such as the amount of available time before the appointment.

## 5.0 Inputs and Outputs

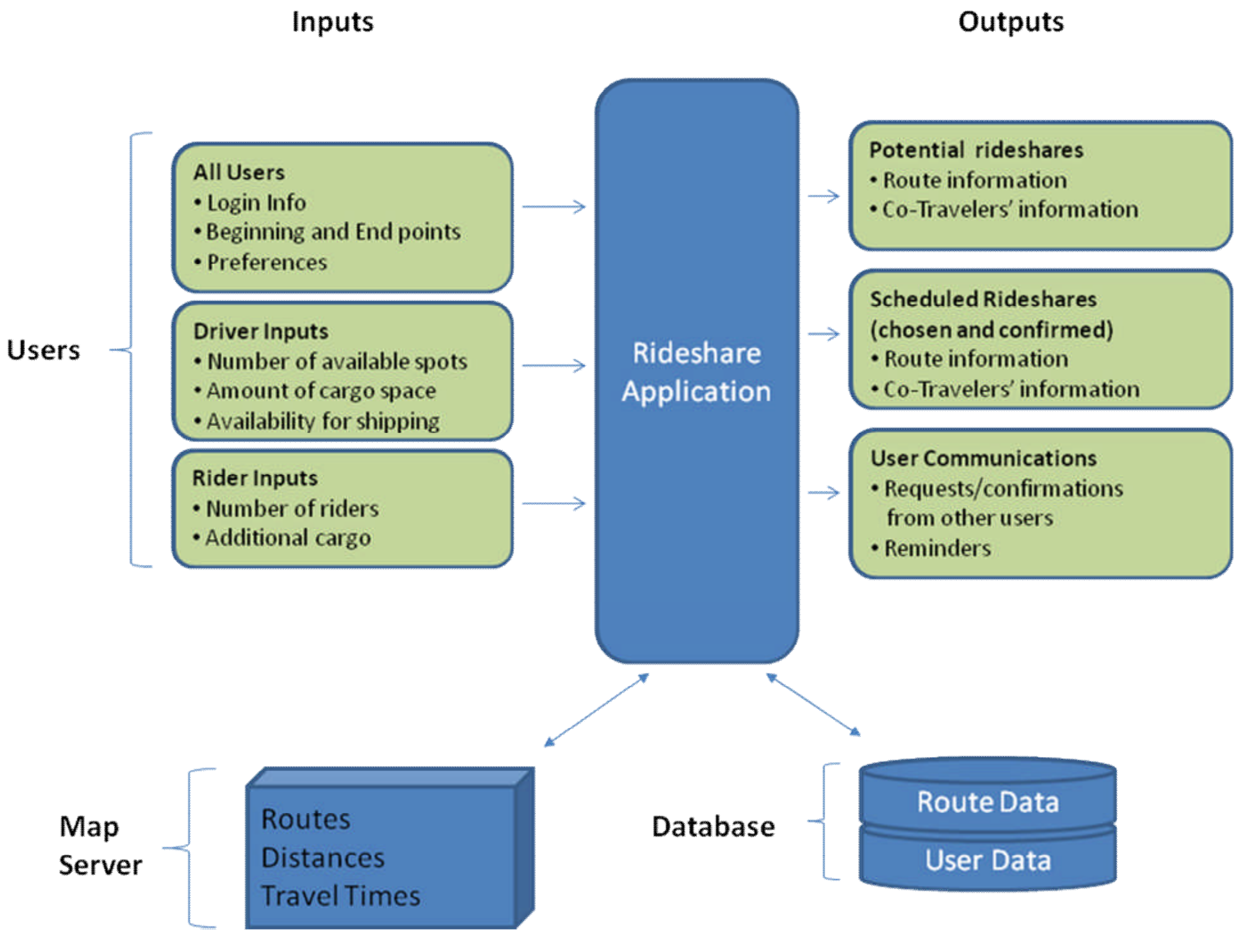


**Figure 5.1: Input/Output Diagram**

Our program will require different inputs from our two types of users, as a driver's needs are different than those of a rider. Both users will input their beginning and end points, along with a time window for the ride. Drivers will input how much space they have in their car, and riders will specify how much space they require, in terms of people and luggage. In addition, both users may include their preferences regarding the age, gender, rating, history, and distance of their counterparts.

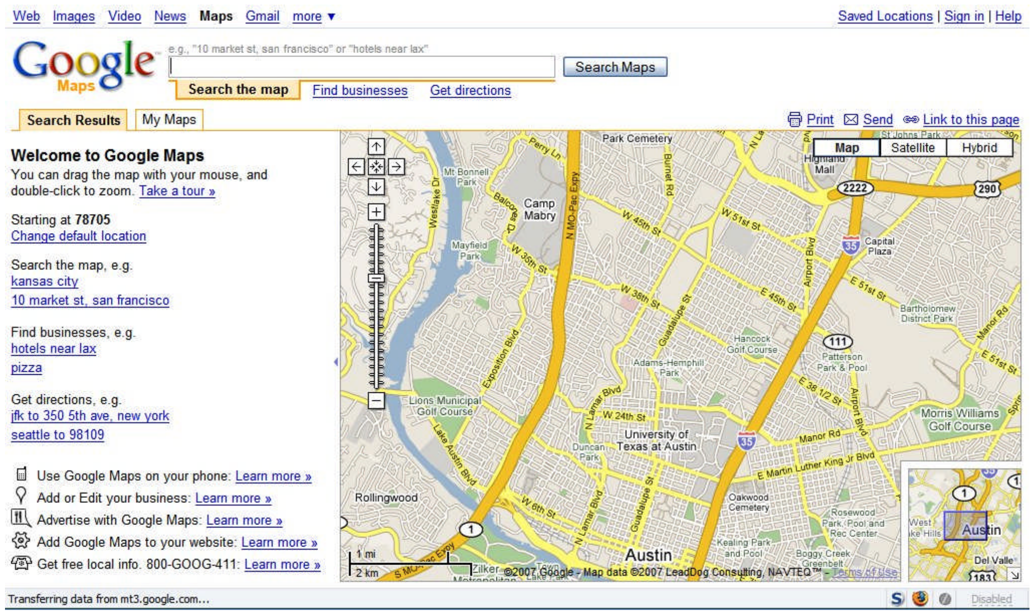
The application will utilize a database to store all user information, as well as route information for established and pending rideshares. In addition, the rideshare program will query a third-party map server for information regarding routes, distances, and travel times. This information will be used to determine optimal matchings and routes for our customers.

Potential matches are then output to the users, along with the route and co-travelers' information. Users can then communicate acceptance or rejection of potential matches and repeat the process if necessary.

Once a rideshare has been established, the details of the trip, along with reminders are sent to all parties.

## 6.0 User Interface Requirements

What sets our project apart from other websites such as eRideShare, [2], is the user interface. By giving the map a central role in the user interactions, we create a unique user experience that will facilitate simple and effective ridesharing. We aim to display routes similarly to Google Transit [3], with menu options available as a side-bar. Features like zooming and turn-by-turn directions will be implemented in the same manner. Figure 6.1 shows a typical map-based application on Google’s own website.


Web Images Video News Maps Gmail more ▾
Saved Locations | Sign in | Help
Google Maps
e.g., "10 market st, san francisco" or "hotels near lax"
Search Maps
Search the map
Find businesses
Get directions
Search Results
My Maps
Print
Send
Link to this page
Map
Satellite
Hybrid
Austin
University of Texas at Austin
Camp Mabry
Rollingwood
Transferring data from mt3.google.com...

Welcome to Google Maps
You can drag the map with your mouse, and double-click to zoom. Take a tour »

Starting at 78705
Change default location

Search the map, e.g.
kansas city
10 market st, san francisco

Find businesses, e.g.
hotels near lax
pizza

Get directions, e.g.
jfk to 350 5th ave, new york
seattle to 98109

Use Google Maps on your phone: Learn more »
Add or Edit your business: Learn more »
Advertise with Google Maps: Learn more »
Add Google Maps to your website: Learn more »
Get free local info. 800-GOOG-411: Learn more »

©2007 Google - Map data ©2007 LeadDog Consulting, NAVTEQ™ - Terms of Use

**Figure 6.1: Google Maps Application Example**

The following paragraphs describe a typical user experience on our rideshare website, though the exact mechanics may change.

The driver or passenger logs in to the secure website. He then proceeds to create a new "looking for ride/passenger" rideshare, or revise a previous rideshare from his history tab. This entails setting the beginning and ending points, as well as designating a window of time for when he plans to leave. He may also designate the rideshare as public or private, indicating if matches may be found only within his "friends" list, or within the general user base. He may set preferences for how far out of the way he is willing to pick people up, as well as for the gender or age of his counterparts.

The user experience will now split into two cases. In the case of a rideshare which will occur on short notice, the user will receive either an immediate match or a message that no match has been found, given the time-frame requested. He can then either alter his rideshare information and search for a new match, or leave a "contact directly" message, so that potential riders can phone or text him to coordinate a pick-up.

In the case of a ride with relatively long notice, the user will receive a confirmation message and email for his request. He will then log out and wait for an email indicating a match, or check back with the service later. When a passenger is added to the driver's route or vice versa, an email or text message will be sent to both parties, and each user can go to the site and review their counterparts. If the user deems the match acceptable, they can accept the rideshare; otherwise they can reject the rideshare and move on to a new search, or review a new possible match if more than one users matched the criteria. This sequence of events is illustrated in Figure 6.2.

Once a rideshare has been accepted by both driver and passengers, a ride confirmation message will be sent to them. This will include all contact information, along with an annotated map of the rideshare, indicating the route and times for all passengers. If a user cancels a rideshare, the website will request they specify a reason to be sent to their counterparts. After a certain number of cancellations or negative reviews, a user's profile may be locked for adding new rideshares, until it has been reviewed by a site administrator.

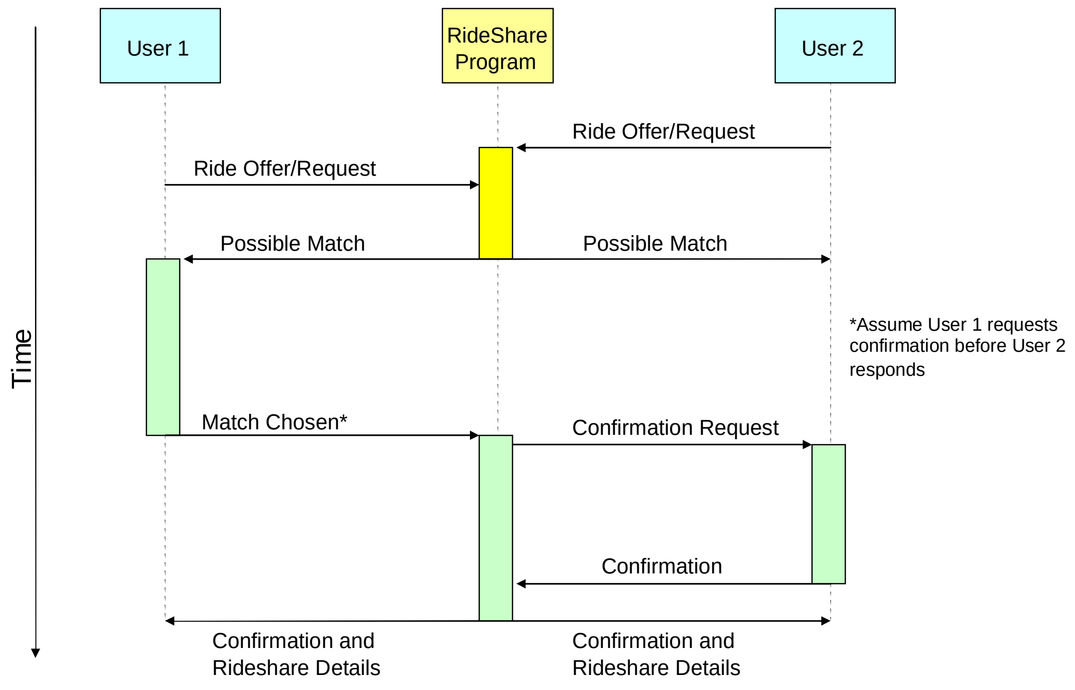


**Figure 6.2: Rideshare Sequencing Diagram**

Users may also manage their profile and review or alter different options. These options may include the following: a user's personal information, such as name, address, gender, and age; general preferences, such as the look and feel of the website, starting page, and communication methods; and rideshare preferences, such as gender, age, and driver ratings. The user may also take advantage of other rideshare tools to alter pending rideshares or set ratings/comments for previous rideshares.

## 7.0 Key Operational Specifications

In order for our program to ultimately work, each of its pieces must perform to a certain standard.

To begin with, our user interface must be secure. Each user's login information and preferences must be saved, along with their route. We can test this using human beta testing as well as attempting to rudimentarily hack user accounts, changing user settings around and verifying that they have been saved.

Additionally, our data server must measure up to several standards. The database it contains must be secure, in that no one outside our program can query it for data, which we can quickly test by setting a password and trying to bypass it. The server's hardware must also perform well; we must be able to add and drop entries from the table quickly, we must be able to query data quickly, and the server overall must have a large data capacity in terms of hard drive space. We can create a simple program that adds and drops many routes faster than a large group of humans could to test the speed of the server, and we can figure out our route capacity by measuring how much space each route takes up in the server. This test program could also test our connection to Google Maps, which needs to be fast as well.

Our algorithm must be efficient. This implies that it should generate a maximal solution in terms of the number of users' served and the number of passengers per car, while providing those users with the shortest possible routes. We also must be sure that our algorithm can intelligently compare routes, mostly in terms of length and time. We can test all of these algorithm related specifications by trying a large number of typical cases, as well as manually entering in specific corner cases which could cause our algorithm to fail. Another very important parameter we must consider and test for is how fast the algorithm completes its task.

Finally, one of the most important functions of our website is its user interface. The interface must be user-friendly and intuitive, which we can measure by timing how long people spend on each page, where their eyes are, and why they chose when to leave the website. Tester feedback will be essential for improving the user interface, as they can inform us exactly where the website's mechanics aren't working, as well as less tangible things like if the user did not like the color scheme.

## 8.0 Key Environmental Specifications

On the hardware level, the environmental requirements for our ride sharing program are the same as any other computer: electricity and a cool, dry room to operate in, as well as a connection to the internet. As this is a software-based project, it will have to stand up to some electronic rigors. Our program will need to adhere to some capacity limitations, probably measured in total number of users and routes saved in our database. Because of hard drive space constraints, our database will only be able to hold so many such users and routes.

Also, the server can only handle some finite number of requests in a short amount of time without causing long waits, because the server's processing power and communication capacity are bounded.

## 9.0 Prototype Cost

We will likely need to purchase several tools to put our first run program together. Initially, we would need an internet-enabled server computer to run our programs, as well as crunch all the numbers. We could rent a server from several different companies for somewhere around $250 per month [4], or at least a virtualized portion of one server on which we could run just what we needed to. Virtualization allows multiple people to have access to the same computer as if they were separate physical machines. Alternately, the University could furnish us with a virtual server for essentially no cost to us.

The server also requires a few programs to run our project. Tomcat, a program which runs a Java web server, is freely downloadable on Apache's website [5]. Additionally, we will require a structured query language, or SQL, database program such as Oracle [6], which is expensive, or MySQL [7], which is free. The database program will allow us easy storage and retrieval of hundreds of thousands of data entries, which is necessary to support a large number of users.

We may also need to acquire a Software Development Environment (SDE), such as Microsoft Visual Studio, which could cost us as much as $300 [8]. This compiler and development environment will allow us to actually put together and publish our code, and it would be very useful in debugging. Alternately, we can likely use University resources such that this cost would not be incurred, or use an open-source SDE such as Eclipse.

To communicate with Google Maps, we will need to use their application programming interface, or API. To obtain this, we will use an API key provided by Google, which is free so long as we do not charge for our service [9]. However, if we do charge for our RideShare program, we will have to purchase an Enterprise API Key from Google for $10,000.

All totaled, our costs could be over ten thousand dollars, or free, depending on the university.

## 10.0 References

[1] Unknown Author, "Use Case Diagram" [Online document], 2007, [cited 2007 Nov 6], Available HTTP: http://en.wikipedia.org/wiki/Use_case_diagram

[2] Unknown Author, "Use Case Diagram" [Online document], 2005, [cited 2007 Nov 6], Available HTTP: http://www.erideshare.com/

[3] Unknown Author, "Google Transit Beta" [Online document], 2007, [cited 2007 Nov 6], Available HTTP: http://www.google.com/transit

[4] Unknown Author, "eApps Hosting Plans," [Online document], 2007, [cited 2007 Nov 4], Available HTTP: http://www.eapps.com/Docs/Documents.jsp

[5] Unknown Author, "Apache Tomcat," [Online document], 2007, [cited 2007 Nov 5], Available HTTP: http://tomcat.apache.org/

[6] Unknown Author, "Oracle, The World's Largest Enterprise Software Company " [Online document], 2007, [cited 2007 Nov 5], Available HTTP: http://www.oracle.com/

[7] Unknown Author, "MySQL, The world's most popular open source database " [Online document], 2007, [cited 2007 Nov 4], Available HTTP: http://www.mysql.com/

[8] Unknown Author, "Visual Studio 2005 Standard Edition" [Online document], 2007, [cited 2007 Nov 4], Available HTTP: http://msdn2.microsoft.com/en-us/vstudio/aa718672.aspx

[9] Unknown Author, "Google Enterprise Website Search Solutions" [Online document], 2007, [cited 2007 Nov 3], Available HTTP: http://www.google.com/enterprise/maps/faq.html