

CS 361 Software Engineering (Section 400 - SP17)

Group 20—Project A

Members:

Joshua Huff (Josh)	<huffj@oregonstate.edu>
Rebecca Kuensting (Rebecca)	<kuenstir@oregonstate.edu>
Alexander Li (Alex)	<liale@oregonstate.edu>
Ya Lien (Vicky)	<lieny@oregonstate.edu>
Elizabeth May (Liz)	<mayel@oregonstate.edu>

Contents

Introduction.....	2
Functional Requirements.....	2
Use Cases.....	4
Non-Functional Requirements.....	5
Supporting Diagrams.....	6
Prototype.....	9
System Design & Architecture.....	11
High Level Overview.....	11
Lower Level Data Flow Diagrams.....	11
Fault Tree.....	13
Architecture Analysis.....	13

Introduction

Traffic congestion is an issue facing commuters in every city nationwide. Ride-share applications like Uber and Lyft are far too expensive for most people to use on a twice-daily basis, and mass transit is only an option—albeit a suboptimal one—for larger cities like New York and Los Angeles. Carpooling is easier on the environment and the pocketbook than driving alone, and it offers timing flexibility that mass transit—with its often unreliable and inconvenient time schedules—just can't match. Rather than face a commute that takes longer, costs more, and doesn't offer door-to-door service, many people would happily give carpooling serious consideration, but existing platforms for finding suitable poolmates (e.g. Craigslist) are unreliable and sometimes unsafe. Enter Oonter. Oonter is a web-based social media tool designed to counter the effects of traffic congestion by helping users conveniently and easily arrange a carpool.

Functional Requirements

Profile Creation

Oonter works by matching users based on detailed personal profiles, much like an internet dating platform. Though users may browse carpools in their area without signing up, all users must create a profile in order to join or create a carpool. Profile creation can be done by linking directly to Facebook or Google account info, or users may create a standalone account that is only used to log into Oonter. During setup, the basic profile requires an active (verified) email address, a username, and a zipcode.

Updating Profile Information

After creating their profile, users will be asked to provide the following additional information: driver's license or state ID number, beginning and endpoints for commute, intended departure time, and intended time of arrival. Users will also optionally be asked to save their phone number and ride preferences for use when a suitable carpool arrangement is found. This information can be changed and updated at any time.

Browsing/Searching Existing Commutes

Users will be able to search through the database of existing carpools that other users are participating in. The search function should provide various search criteria, including locations, destinations, departure times, and intended arrival times. The search algorithm should determine the best routes between commuters and display the results in terms of route efficiency, commute time, and projected arrival time for all parties. The algorithm will take into account the time of day and the typical traffic on the route. Oonter will prioritize matches based on physical address and general time of arrival. If there are plenty of available seats to choose from within active carpools, and a user wishes to narrow the results, additional search parameters include less-important factors like preferred genre of music for

the car ride, a preference for vehicle type, and whether the driver allows smoking in the car.

Create New Commutes

In the event that a user is unable to find a suitable existing carpool group for their commute, users can create a new commute that will be suggested to other users whose destinations and departure/arrival times are similar. When a user creates a new commute, their destination, departure time, arrival time, vehicle type, and current passenger count is required as part of the commute details. Other users that are interested in joining the commute will be able to contact the original poster via private message for more details.

User Messaging

A dedicated Instant Messenger will be included in the basic Oonter user interface, so users are not required to give additional personal information to stay in contact with the members of their carpool. Given what's at stake (being late to work can be devastating for most jobs), push notifications are a clear must. The system will anonymize its users when exposing their schedules to the general public, but relax those anonymity standards when a user accepts another's request to join a pool. For example: Alice wants to join a carpool, so she submits her details to the system. Oonter retains her address for the purpose of comparing distances to other user's home and workplaces, but only shows the latter (in miles). She can choose to remove herself from a carpool at any time, in which case her details become anonymized again.

Account Clearance

Naturally, user safety is of paramount importance. As with any social networking tool that brings users together in real life, there exists a small chance of meeting unpleasant strangers. Oonter gives users something in the way of risk assessment: there are two levels of "clearance" for user accounts: Basic User and Verified User. A Basic User has not previously participated in an Oonter pool, and has taken no steps to prove the safety of their vehicle or their driving. A Verified User has had successful carpool experiences in the past; other users have positively rated him for punctuality, safe driving, etc. The instant messaging system provides users with a way of gauging the personalities of potential matches before they meet in person. Because Oonter prioritizes local matches and cannot be used for financial gain, the odds of abusive behavior seem acceptably low.

Ranking and Reporting Users

In order to maximize the chance that a user will have positive experiences with her carpool, Oonter will provide the ability to "block" users and will operate with a rating system. Giving another user a thumbs-down indicates a negative experience (e.g. the person was frequently tardy). A thumbs-up indicates a neutral-to-positive experience. The system will accumulate these ratings and show the user's favorability in qualitative terms: Not Recommended (for ratings below 60% positive), Recommended (for ratings from 61% to 89%), and Highly Recommended (for ratings 90% and up). Users who receive consistently low ratings will be considered for permanent bans on a case-by-case basis.

In the event of a serious breach of the software's terms of service (which includes showing common courtesy to other users regardless of sex, race, color, religion, creed, or national origin), users have the option of reporting an offender. If a user accumulates enough reports of inappropriate behavior, their account is flagged and they are no longer permitted to use the service. To avoid someone simply creating a new account and repeating the process, the user's email address, driver's license (or ID) number, and IP address will all be logged in the system. If they signed up using Facebook or Google, that account name will be logged as well.

Banned users may appeal a ban by emailing Oonter admins directly, but Oonter admins will reserve the right to uphold bans indefinitely at their discretion.

Use Cases

The following three use cases define typical user interactions with the Oonter carpool application, focusing on the three most important functions of the application: profile creation, carpool search, and carpool creation.

Case 1 - Create a Profile

Actor: New user

Preconditions: User does not have an existing profile or Google/Facebook account

Postconditions: User has a profile containing carpool-relevant information

Flow of events:

1. User enters basic personal information to initiate sign-up
2. User confirms email address
3. User fills remainder of profile, including state ID or driver's license number
4. User enters departure and destination addresses
5. Oonter encapsulates those addresses and gives them only as a distance in miles to other users
6. User enters a specific desired arrival time for both trips
7. Oonter encapsulates those times and gives them as ranges of time to other users. A filter will consider a user's exact time (For example: If a user must be to work by 7:05 a.m, he will be included in the search results with the criteria of 6:00-7:00 a.m., but he will not be included in other users' search for 7:00-7:30 a.m. arrivals)
8. User selects the days of the week he will participate in the pool (default is Monday-Friday)
9. **(Optional)** User provides additional information, such as preferences for vehicle type; smoking or non-smoking; music, talk radio, or a "quiet" ride

Case 2 - Searching For Existing Carpools

Actor: Non-driving user

Preconditions: User has an existing profile

Postconditions: User is given a list of all other users matching search criteria

Flow of events:

1. After login, user chooses "Search for Carpool" option
2. User enters departure and destination addresses (default: addresses on profile)
3. User enters a specific desired arrival time for both trips (default: times on profile)

4. User selects the days of the week he will participate in the pool (default: days on profile)
5. **(Optional)** User specifies additional preferences: vehicle type; smoking or non-smoking; music, talk radio, or a “quiet” ride
6. Oonter returns table of all carpools with availability that match user’s preferences, in order from closest to most distant (If user’s search is too restrictive, the table may be empty, or provide suggestions for an alternate time)
7. User has the option of viewing each user profile in the search result and sending a message
8. Oonter sends an email or push notification to the contacted user to ensure efficient communication (carpool arrangements can be considered time-sensitive)

Case 3 - Start a New Carpool

Actor: Driving user

Preconditions: User has an existing profile; user is not already involved in an existing pool

Postconditions: User's profile appears in searches as an available carpool

Flow of events:

1. User chooses "Create a New Carpool" option
2. User enters departure and destination addresses (default: addresses on profile)
3. User enters an arrival time for both trips (default: times on profile)
4. User selects the days of the week the pool will operate (default: days on profile)
5. User enters make, model and production year information about the vehicle
6. User places limits on how far the pool will operate beyond the given addresses and times
7. User specifies whether smoking is permitted and whether the car plays music, talk radio, or is a "quiet" ride
8. User specifies how many seats are available to other Oonter users

Non-Functional Requirements

Efficiency

The application will not produce duplicate search results, nor will a simple search take longer than 15 seconds on a standard desktop computer on cable internet speeds. Returned search results will be accurate, and will reflect the user's stated preferences in a way that is predictable and transparent.

Integrity

User privacy is essential. The application will never reveal a user's last name, contact information, or home or work address. With no exception, users are in complete control over who else gets their personal information. Likewise, the application will be carefully designed to avoid allowing a carpool to become "overbooked" with occupants. If the specifications of a user's carpool change (it becomes a "quiet" carpool, or the car owner moves to an apartment several blocks away), users will be immediately notified of the change and given the option to keep their pool or switch to a new one.

Usability

The application's minimalist UI and straight-forward design will make setting up a carpool trivially easy. The UI will make the most of user's familiarity with similar applications, using fields and buttons reminiscent of online dating or shopping applications to help guide the user through the tasks of profile creation, searching, and contacting other users. Given a reliable internet connection and ordinary hardware, a user should be able to set up an account and begin searching for a new way to work within five minutes. The user will be able to check the schedule of their weekly carpool at-a-glance using a simple summary page, and push notifications will help remind all users of their carpool's schedule.

Maintainability

Like all software projects, the application will require maintenance as technologies evolve. The user-created portions of the site (the profile database and active carpool database) will need monitoring to ensure that their information remains up-

to-date, useful, and acceptable within the scope of the terms and conditions of the application.

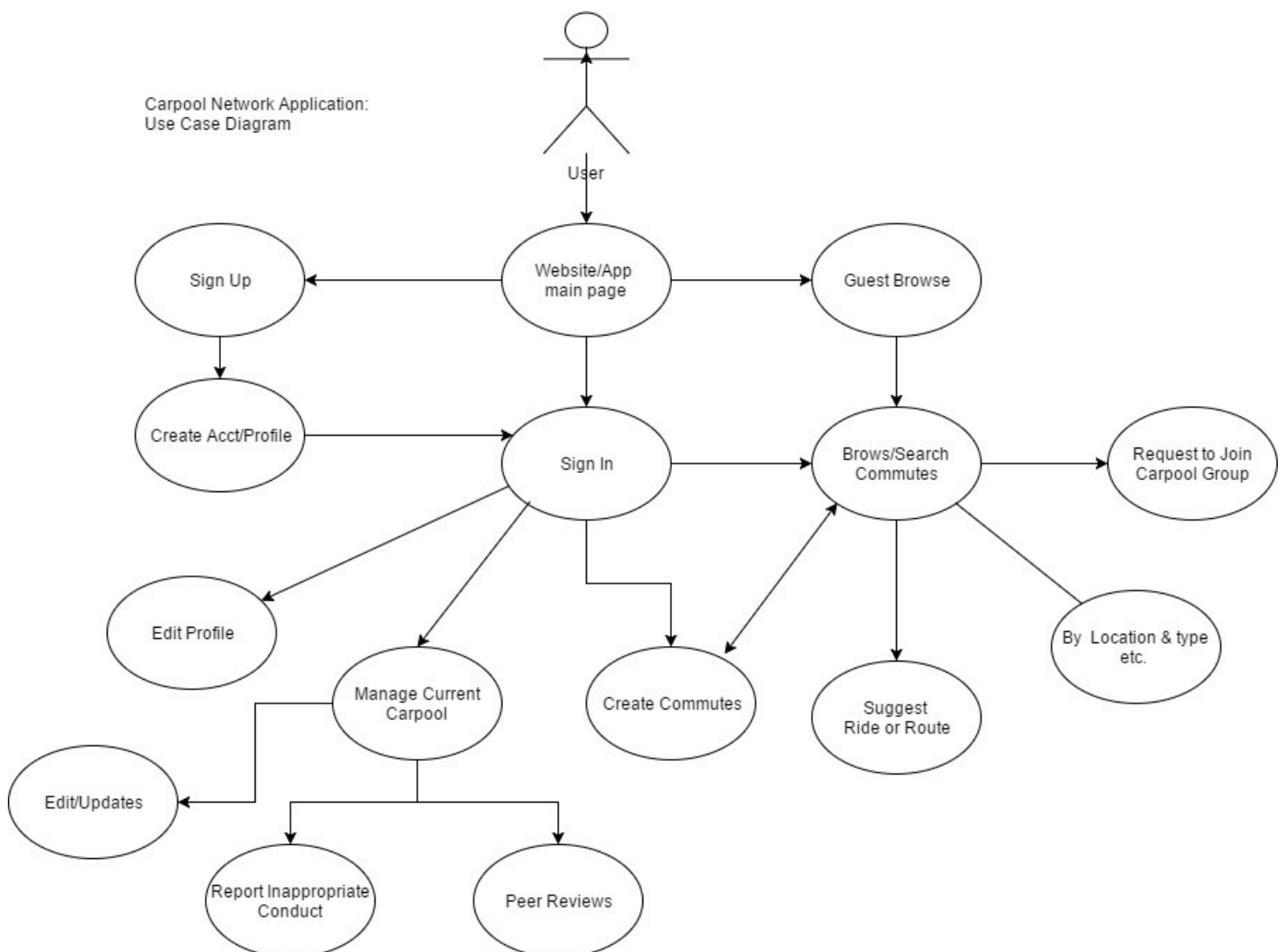
The source code will be written and commented in a consistent fashion for ease of future development and revision. Though Oonter is a complex system when taken as a whole, it will be easily modularized into several discrete sections so that it can be easily scaled as the userbase grows. If the userbase becomes very large, the user and carpool databases could be easily subdivided by region, since users must be geographically near each other to make practical use of the application.

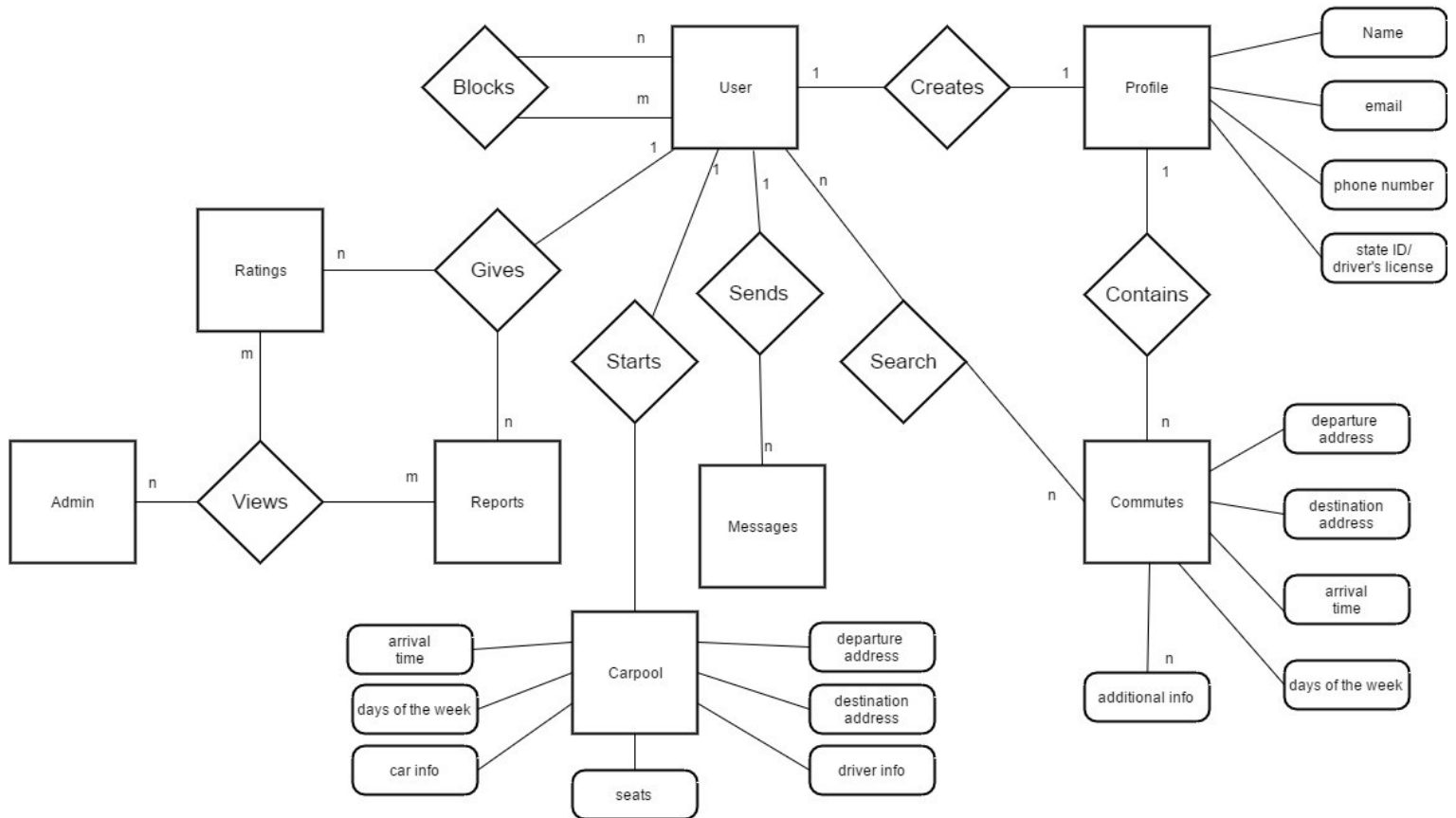
Portability

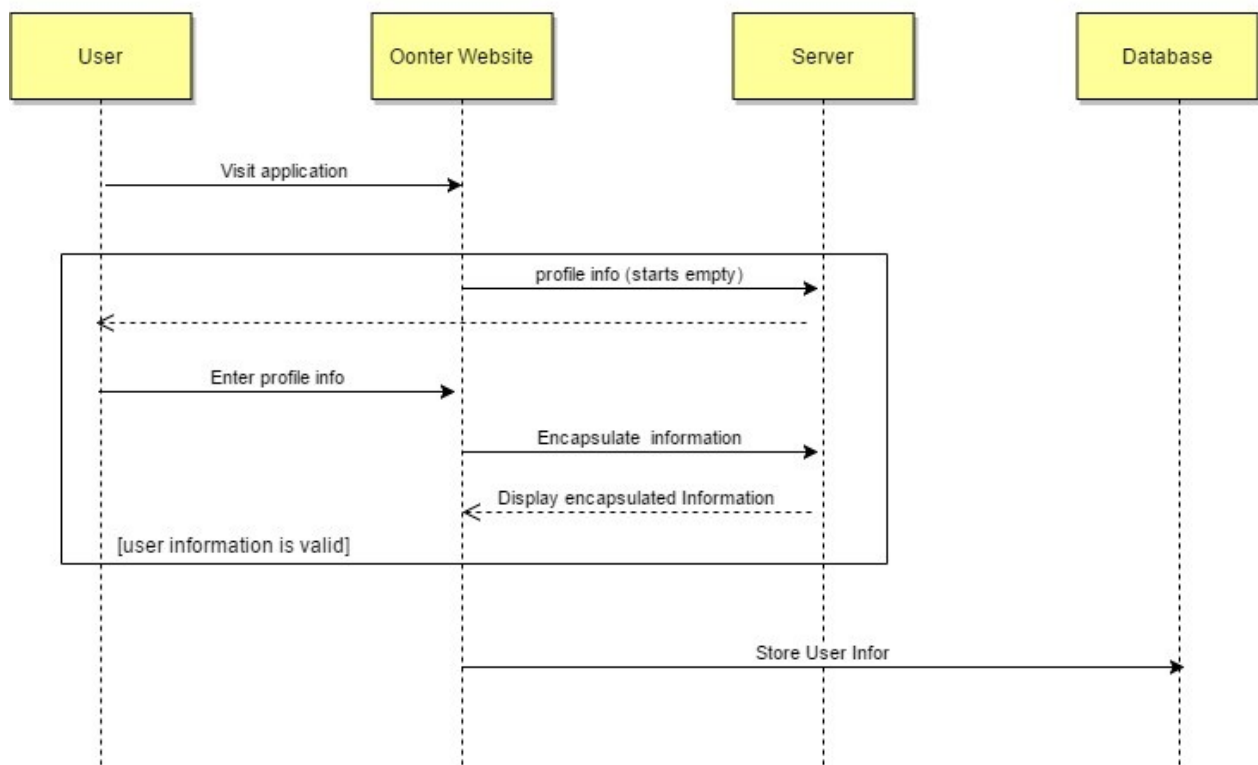
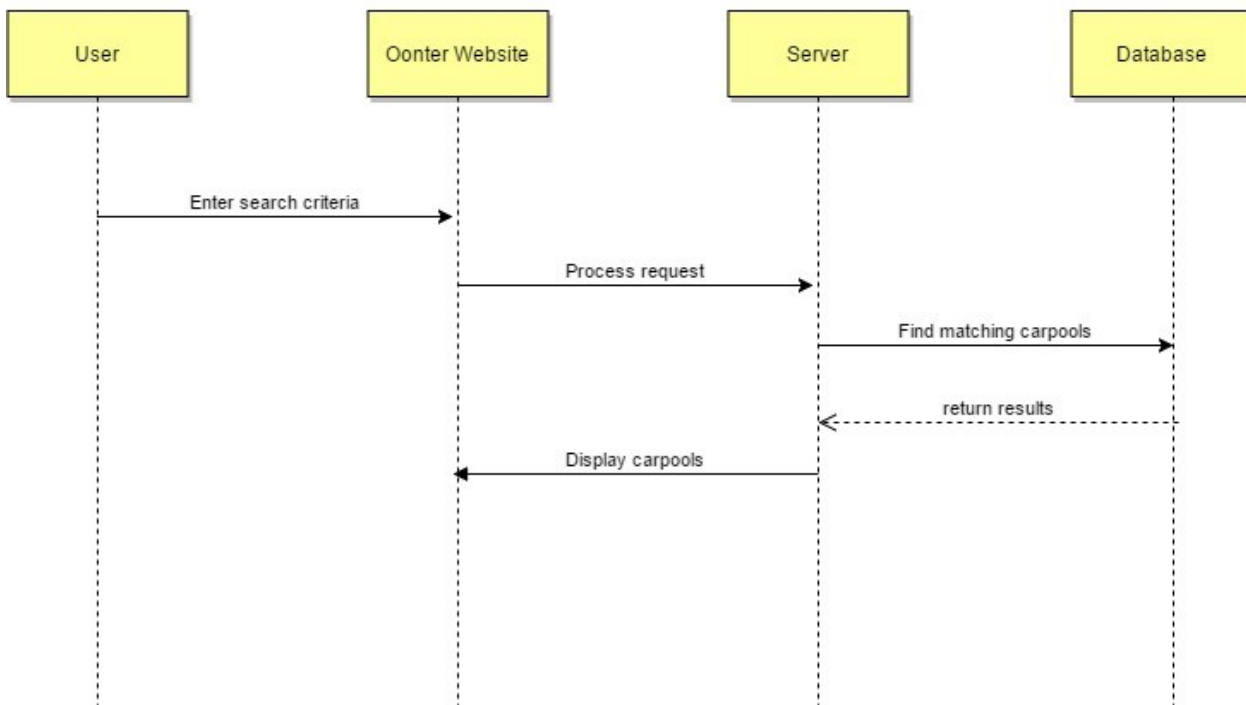
The application will be designed with modern browsers and smartphones in mind, which means it must be compatible with many different operating systems and control schemes. Keeping the UI minimalistic will help keep the application visually consistent across platforms

Supporting Diagrams

Use Case Diagram



ERD Diagram

Message Sequence Diagram: Create Profile**Message Sequence Chart: Search for Carpools**

Prototype

To ensure that we fully understand the customer's concepts, we created the following low fidelity prototype. In working through the potential screens we realized that while there are many different screens, the majority are simply different representations of the same information. With that in mind, our prototype focus on how to gather that information, by users creating a profile and providing desired route information. Again, we emphasize that this is a low fidelity prototype that would be adjusted and refined as the project advanced. The screens were created to be neutral to application for a web site or mobile app and the first screen, a "theme" page was created as a more refined visual to generate ideas from and give a tone to the prototype.

For detailed, higher resolution images please see the separately submitted PowerPoint file, so titled Prototype.ppt.

Theme page to generate ideas & style concept

The theme page for the Carpool Network features a background image of a busy highway with many cars. At the top, there is a header with the text "Carpool Network" and a "Log In" button. Below the header, there are three main buttons: "Sign Up", "Search", and "Pool". Under the "Search" button, there is a form with fields for "City/Area", "From:", and "To:", followed by a "Find" button. At the bottom, there is a small URL "www.CarpoolNetwork.com".

Oonter  Carpool

Log In

Sign Up

Search as Guest

This is the info needed to sign up. For app interface questions would be divided up between multiple screens depending on UI testing.

Oonter  Carpool

Sign Up

*First Name:

*Last Name:

*Email:

Address:

City: STATE

ZIP:

Are you willing to be the driver? ☐ YES ☐ No

ID or Drivers License #:

How many passenger can you comfortably fit?

Your Vehicle:

Make: Year:

Model: Color:

License Plate #:

/* insert legal disclaimers and info ... */

Next

Oonter  Carpool

Route Info

DEPART From:

ZIP: Time:

Departure Flexibility:

Sun Mon Tue Wed Thur Fri Sat

Do you have the same schedule every week?

☐ YES ☐ No

ARRIVE At:


ZIP: Time:

Departure Flexibility:

Sun Mon Tue Wed Thur Fri Sat

Do you have the same schedule every week?

☐ YES ☐ No

Oonter  Carpool

Carpool Preferences

Vehicle Type ?

☐ SUV

☐ Sedan

☐ Hybrid

☐ None

Sound?

☐ Music ☐ Quiet ☐ Either

Music Preference Type 1 TYPE OF MUSIC 2

Smoke Preference ?


☐ Yes, please.

☐ Okay

☐ Don't care

☐ Absolutely Not

Special Requests or Other Concerns:

Oonter  Carpool


Email or UserName:


Password:


Log In


[Forgot password ?](#)


Screen:


Oonter  Carpool


Search 


Create Or Manage Carpool 

Messenger 



Profile 

User Review 

Oonter  Carpool

/* Google Style Search Results */

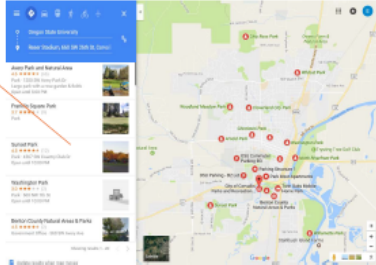
Carpool SOMENAME
From Suburbs to Downtown


Leave Suburbs @	Leave Downtown @
07:00 AM	5:00 PM

Open Seats : 2

Meeting Location: Outlet Mall


/* APP Style Search would be similar to Yelp */



Oonter  Carpool

Carpool SOMENAME

Route

From Suburbs 

To Downtown

Round trip ? Yes

Meeting Location: **Outlet Mall**

Open Seats: 2

Seat Position(s) Available:

- right rear behind driver

/* other info listed will be music & smoking */

[Request to Join](#)

Schedule:

From:


Leave Suburbs @	Arrive Downtown @
07:00 AM	8:00 PM

To:

Leave Downtown @	Arrive Suburbs @
05:00 AM	6:00 PM

Carpool Members


D	User: UserName *Driver
1	User: UserName Rating info if any
2	User: UserName Rating info if any

Oonter  Carpool

REQUEST TO JOIN

Do you have any questions or special requests we should include in your request to join?


[SUBMIT](#)

Oonter  Carpool


Create Carpool Screen


Simply a editable form version of the carpool details screen.

- Route
- Members if any
- Open Seats & positions
- Preferences
- Special Notes

Oonter  Carpool

Messenger Screen

Today 12pm - Hey guys I can't make it 

12/22/17 - Can we leave 15 minute late? 

[Start New Message](#)

Create New:

Send To:

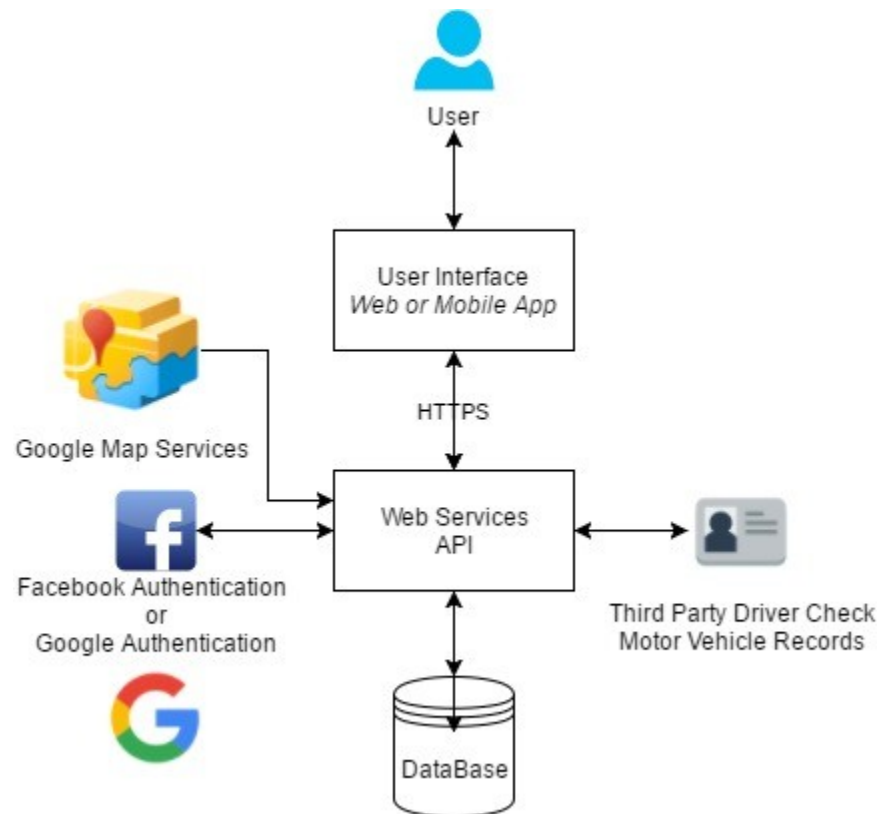
Text entry

[Send](#)

System Design & Architecture

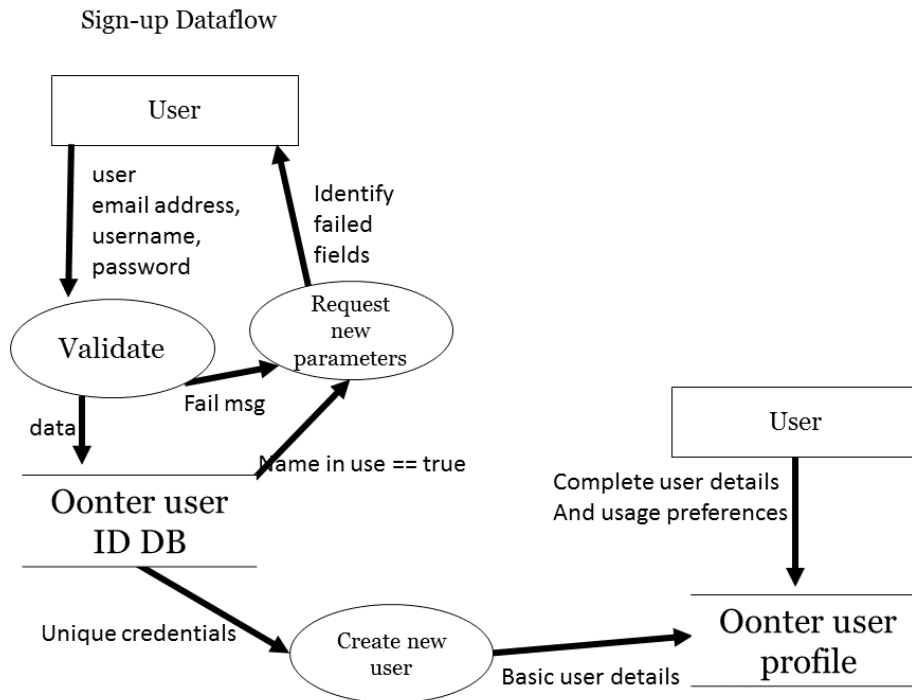
High Level Overview

The carpool network would need many different technologies to be most useful, but in order to execute correctly we have started with the basics as outlined in the diagram below. The technologies referenced would enable us to meet the requirements outlined previously.

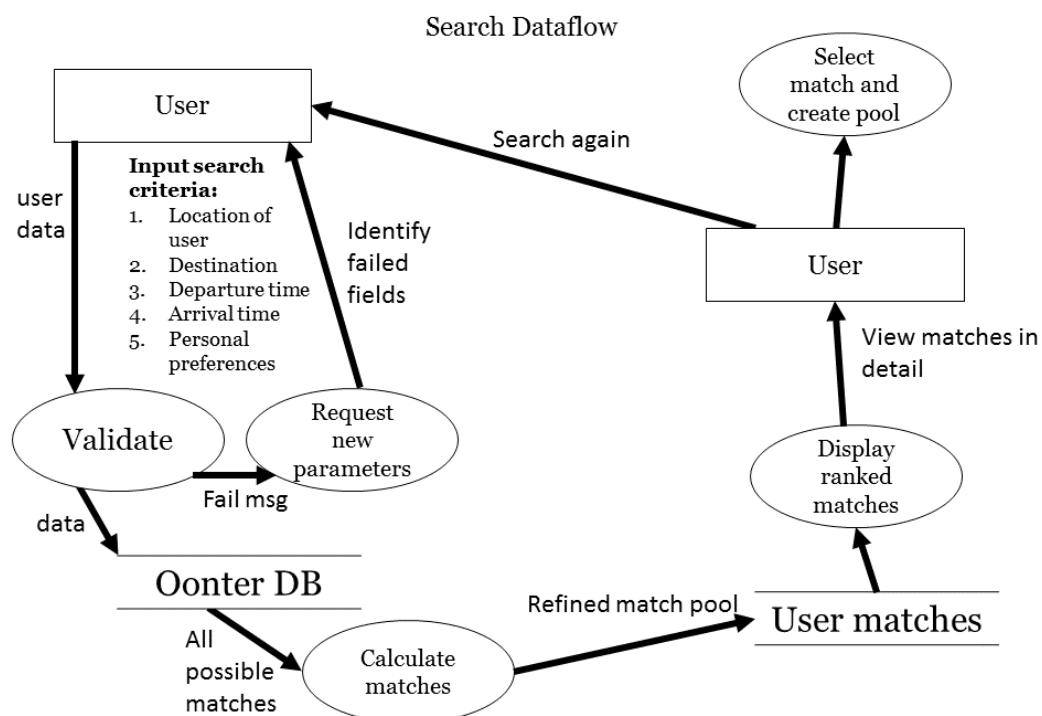


Lower Level Data Flow Diagrams

The following figure shows the decomposed dataflow of the sign-up process. As depicted, the user must first pass data through a validation loop (to verify that their email address and other account credentials are valid), then their chosen username and other details will be compared against the database of current users to ensure that they are not creating a duplicate or invalid account. Their input will then be used to initiate a new profile, which is subsequently editable by the user as they alter their personal information over time.



The next figure shows the dataflow of the search procedure, which is executed when the user queries the database of potential carpool matches. As depicted, the data, originating in the form of the user's query, is validated by the system, then passed to the Oonter user database, which returns all possible matches to a function that refines matches according to user location and preferences. The resulting matches are compiled as a temporary dataset and displayed to the user, who can investigate the returned data (by interacting with links), or initiate a new search.

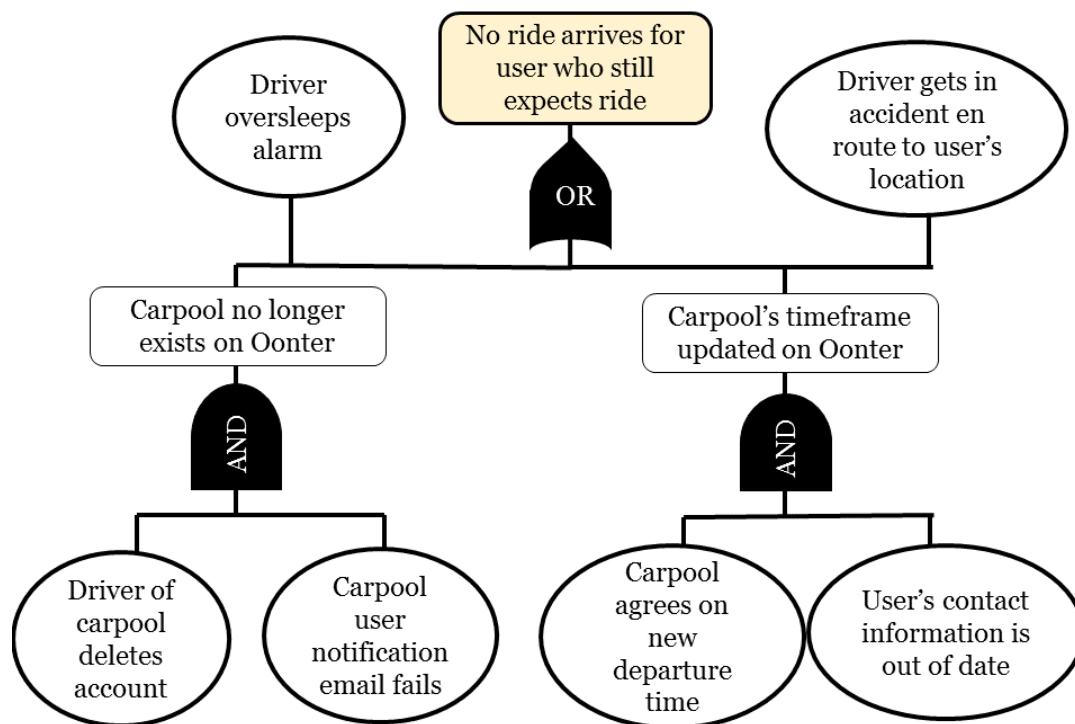


Fault Tree

In the following tree, we trace the possible system states that could lead to the user's carpool not arriving as expected. This represents a serious failure of the system, as it would result in Oonter's basic functionality not being delivered to the user.

Fault tree

Case: User's ride does not arrive:



Architecture Analysis

The architecture described above fully satisfies the requirements stipulated by the customer. The specific requirements were that Oonter should be a web-based platform, that it should function based on user profiles and a matching search engine, and that the focus should be on locating and maintaining a group of people for a carpool, not on the logistics of driving or getting a ride. As stated by the customer in an email to the design group: "I do not envision a ride hailing service... (Oonter) gives you the information to arrange your own carpool with other people."

After our initial design meeting, the customer specified further feature requirements, all of which have been accounted for in our architecture. These features include the ability to rank users and flag them for inappropriate conduct, the ability to message users within the app without revealing personal contact info, and the addition of carpool size and the number of available vehicle seats to the search results.

The primary method through which we are evaluating and ensuring the functionality of the customer's requirements is walking through our use cases and identifying opportunities in the architecture for improvement based on these cases

Case 1: Creating a profile

The first use case and a vital aspect of the site functionality is creating a profile to use the service. When a user accesses the web client or mobile application interface for the first time, they must create a profile that ultimately stores their personal information in the back-end user database. The web client or mobile application interface communicates to the web services API server via HTTPS, thus ensuring the user reasonable protection from information and security breaches, such as a man-in-the-middle attack. With that being said, integrity is a vital quality attribute for this platform, and therefore stealing information directly from the database is also a grave concern. The web services API server must be secured such that the servlet cannot trivially be compromised, and of course the data stored in the database must be encrypted. There are various ways we can implement database encryption to ensure the integrity of the architecture, such as symmetric database encryption, in which a private key exchange must occur between the client and the database before the data can be decrypted.

Case 2: Searching carpools

In the second use case, users must be able to search for existing carpool groups whose details reasonably match the user's preferences, such as departure and destination locations, arrival time, the days of the week the user will be participating in the carpool, as well as optional preferences, like music or radio preferences or smoking/non-smoking passengers(?). The user begins by interacting with the user interface, filling in the desired search criteria for existing commutes. Once the form is sent via HTTPS to the web services API server, the server must work in conjunction with the database to return relevant results to users efficiently and quickly. Since efficiency plays a large role in our quality attributes (non-functional requirements) goals, the web server must exchange the user's specific private key with the database in order to decrypt relevant data, process data returned from the database after encryption, sort the results from closest to most distant while taking into account additional user preferences, and return a set of existing users whose carpools match the user's search query in under 15 seconds.

Case 3: Starting new carpools

The third use case allows users with existing profiles to start a new carpool group. The user interacts with the web client or mobile application interface, entering commute information like departure and arrival addresses, arrival times, vehicle information, etc. After the required information is provided, the new carpool information should be sent over HTTPS to the web services API server, at which point the pool information is securely encrypted on the database, and the server should add the user's profile and new carpool group to the list of currently active carpools.

The fault diagram in the previous section provided us an opportunity to think about issues in the system that might prevent Oonter from delivering its most basic functionality (rides to and from work) to our users. In creating that diagram, we

realized that though our architecture provides several failsafes that should keep catastrophic failures in check, the Oonter application is heavily dependent on the dependability of its users in order to function properly. There are many external, user-initiated variables that may cause the reliability of the app to suffer. Carpools are, by nature, a cooperation between different people working towards a common goal. Oonter encourages users to cooperate with strangers, and to trust them with their schedules (and, to some degree, their safety). Our architecture anticipates some degree of user behavioral variability by including extensive validation, location and time-restricted search results, an emphasis on user privacy, and push notifications for timeliness, but the userbase itself is the unknown ingredient that will make individual carpools succeed or fail.

Databases

As shown in our low-level and high-level diagrams, the Oonter architecture is dependent on databases. The first database a user will interact with is the user profile database, which is the primary way that the matching algorithm will select carpool options for the user's commute. An overview of the user profile database clarifies how the information in the database conforms to the customer's requirements. Each user's entry in the user database will contain at least the following information, but the list is not exhaustive and is summary in nature.

<u>Data ID/Title</u>	<u>Description</u>
Name	First & Last Name of user
UserName	User-created, must be unique
Email	Must be validated at time of profile creation, encrypted within database
Password	Encrypted within database
Address - Street	Encrypted within database
Address - City	Encrypted within database
Address - State	Encrypted within database
Address - ZIP	Encrypted within database
Phone number	Optional. Encrypted within database
Drivers license #	(Stored locally only as an encrypted key -- actual driver's license data can be stored externally, possibly with a third-party service)
TripID	Each individual trip gets a unique ID, so a morning commute would be one and a evening (return) would be another. Every trip a user takes, and its metadata, are stored in their profile.
Start Address	User primary commute route. Location where user wants to commute from.
End Address	User primary commute route. Location user wants to get to.

Leave Time	User defined time of commute.
Desired Arrival Time	User defined time when user needs to be at end address.
Schedule Days	(Su,M,T,W,T,F,S) User provided days of the week when commute route applies.
Round Trip	T/F question.
Additional Preferences	User will have the option to set a variety of different preferences (represented as booleans), including smoking/nonsmoking, quiet ride/conversation, etc.
+ many more	This list is not exhaustive but a summary in nature. As the project progresses we acknowledge that there will likely be many more items to keep track of.