**Let’s Carpool**

Alexander Cihla cs169-fo   
Sui Fung Alex Wong cs169-ba

Stephanie Ku cs169-bc

Peter Lee cs169-dh  
Aime Ngongang cs169-ed

Sam Chi-leong Vu cs169-ar

**Project Abstract**

Let’s Carpool is an Android application that facilitates carpooling. The problem with carpooling is that it is hard to schedule, organize and you want to trust the person you are riding with. By having a carpool finder, it will allow the user to post trips, and/or find potential carpooling schedules that match their destined route. There are two types of users: Drivers and Riders. Drivers are able to list their routes whilst Riders are able to search for possible routes and request a ride. Carpooling provides an economically and environmentally friendly alternative to driving. On an aggregate level, this application will hopefully encourage more people to carpool, which decreases the number of cars on the streets at once.

**Document Revision History**

Rev. 1.0 2013-02-22 - initial version

**Customer**

The customer of our application is the general public who are 18+ and needs a carpool or drivers, with valid license age 21+, who are willing to give a carpool to the people who are traveling in the same direction. Although the customer is the general public, the target audience is people who do not own cars, are unable or unwilling to use public transportation, want to be frugal and more environmentally friendly.

Many people travel long distances every day, and most of them travel alone, but in the same direction. Most of them travel by their own car, yet the others may use different kind of public transports. It would be nice that if the drivers provide carpool services to the other drivers or public transport users, which both of the driver and rider either can save money or time.

**Competitive Landscape**

Potential competitors (list of companies with similar applications):

* Carpool School Edition (Mobile app)
* Karpooler (Mobile app focused on kids)
* Car Pool Party (Mobile app)
* Zimride (Web and mobile interface)
* Looptivity (Mobile app)
* Rideshareonline.com (Web interface)
* Craiglist Ride Share (Web and mobile interface)
* eRideShare (Web interface)

They are many other different companies in this business. However, they do things a little bit different than what we have planned.

Main differences from our competitors:

* Better and friendlier user interface
* Targeted at general public
* Users have chance to bid on price
* Drivers can charge/mile or fix rate
* More on demand rides
* Driver ratings, feedback and reviews
* Better database of current users
* Use of Google map API and map view
* Use of “My Location” functionality
* Use of search engine to find best possible routes

After each ride, each user has a chance to rate the trip and driver. Thus, users are able to read other users’ feedback and reviews on the driver, enabling them to make better decisions.

Many the existing applications have a page showing all the existing trips. We want to create an app with a great search engine, the user will enter his departing and destination points and time, then the app will retrieve a list of cars that will either go from his city or pass that city and that will either stop at his final destination or pass by his final destination.

Overall, from all the competitors, our mobile app will be something more user friendly than all the competitors have. Tools like Google map will enable users to visualize their routes and hence make better decisions. We will have a good database to keep track of our users, compared to other competitors, such asCraigslist, that do not have one. We intend to provide a safe and reliable application for users will be able to know more about their drivers by reading the review.

**User Stories**

|  |  |
| --- | --- |
| **Name** | **Signup** |
| **Actors** | User without account |
| **Triggers** | Click Signup button |
| **Precondition(s)** | User does not have account |
| **Actions** | -Click Signup button -Choose Driver/Rider -Enter Email/Password -If Driver (enter License, Registration, etc.)  -Enter Personal Information (Name, Age, etc.) -Click Submit |
| **Postcondition(s)** | New user is created |
| **Acceptance Test** | User logs in once email is verified through confirmation email |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Login** |
| **Actors** | User with pre-existing account |
| **Triggers** | Input Username (Email)/Password, and clicks ‘Login’ |
| **Precondition(s)** | User has already signed up before |
| **Actions** | -Click Login button -Server checks database to verify email/password combination -If success → welcome activity -Else → Try again error |
| **Postcondition(s)** | User logs in to system |
| **Acceptance Test** | User logs in with username and password |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Forgot Password** |
| **Actors** | User with pre-existing account |
| **Triggers** | Input Username (Email) and clicks “Forgot Password” Button |
| **Precondition(s)** | User has already signed up before |
| **Actions** | -Click “Forgot Password” Button -Re-enters valid email -Click “Request Password” Button |
| **Postcondition(s)** | User receives new email with password |
| **Acceptance Test** | User receives email if he/she requested password |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Logout** |
| **Actors** | User with pre-existing account logged in |
| **Triggers** | Click “Logout” Button |
| **Precondition(s)** | User is already logged in |
| **Actions** | -Click “Logout” Button -Server takes note that user has logged out |
| **Postcondition(s)** | User is logged out and is returned to home activity |
| **Acceptance Test** | Once user logs out, he/she is returned to home activity |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Search Ride** |
| **Actors** | User type Rider |
| **Triggers** | User inputs Routing parameters (destination, time, etc.) and presses “Search Ride” Button |
| **Precondition(s)** | User successfully inputs routing parameters |
| **Actions** | -User clicks “Search Ride” Button -Server filters out possible routes |
| **Postcondition(s)** | User is presented with list of possible routes |
| **Acceptance Test** | List of possible routes presented are suitable for the route requested by user |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Request Ride** |
| **Actors** | User type Rider |
| **Triggers** | User chooses Driver and clicks “Request” button |
| **Precondition(s)** | User has decided route by driver |
| **Actions** | -User chooses route and driver  -Enters payment preferences (cash or electronic payment option) -Clicks “Request” button |
| **Postcondition(s)** | User receives notification that request was sent, and driver is notified by email and on Request bar |
| **Acceptance Test** | Driver receives email only when Request has been made and request displayed on Request bar |
| **Iteration** | 1 |

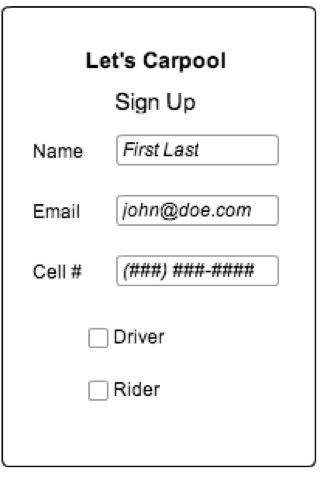
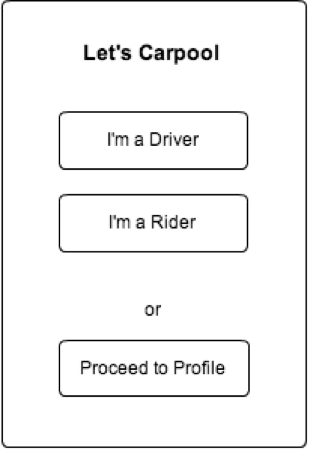
|  |  |
| --- | --- |
| **Name** | **Accept Ride Request** |
| **Actors** | User type Driver |
| **Triggers** | Notification in Requests bar and email confirmation of request |
| **Precondition(s)** | Request has been made to driver for particular route |
| **Actions** | -User opens up Request  -User reviews information shown in request -User clicks “Accept” button (within minimum time span) |
| **Postcondition(s)** | Requestor receives notification of acceptance through email and Accepted Rides bar |
| **Acceptance Test** | Requestor receives email only when Request has been accepted and new ride is shown on Accepted Rides bar |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Find Ride** |
| **Actors** | User type Rider |
| **Triggers** | User clicks “Find Rides” button. |
| **Precondition(s)** | After the user is logged in, the user is given a choice to either find a ride or post a ride. |
| **Actions** | The user is able to see all available rides on our listing. It shows the driver’s departing location, time, destination, driver’s rating and price. |
| **Postcondition(s)** | The user is given a list of rides the drivers posted. |
| **Acceptance Test** | Logged in user is able to see all the available rides. |
| **Iteration** | 1 |

|  |  |
| --- | --- |
| **Name** | **Post Driver Feedback** |
| **Actors** | User type Rider |
| **Triggers** | The user clicks “Give Feedback” on the driver’s profile page. |
| **Precondition(s)** | The user is on the driver’s profile page. |
| **Actions** | -The user will see a textview where he/she can write a feedback and give ratings to the driver.  -The ratings will be based on a five star scale. |
| **Postcondition(s)** | The driver’s profile page is updated with the new feed back. |
| **Acceptance Test** | The driver’s profile page and the rating changed after a user writes a feedback and give ratings. |
| **Iteration** | 2 |

|  |  |
| --- | --- |
| **Name** | **Reject/Change Ride Request** |
| **Actors** | User type Driver |
| **Triggers** | Notification in Requests bar and email confirmation of request |
| **Precondition(s)** | Request has been made to driver for particular route |
| **Actions** | -User opens up Request  -User reviews information shown in request  -User writes in Rejection Reason box (this could be a request for rider to change a part of their request, which means if rider accepts change, he/she needs to submit new request) -User clicks “Reject” button (within minimum time span) |
| **Postcondition(s)** | Requestor receives notification of rejection/request to change through email and Rejected Rides bar |
| **Acceptance Test** | Requestor receives email only when Request has been rejected and rejection with reasoning is shown in Rejected Rides bar |
| **Iteration** | 1 |

**User Interface Requirements**  
We hope to make the UI as intuitive as possible while also giving the user an appropriate feeling of control and understanding as to what our service provides and what functions they are currently carrying out. Submit, back, and exit buttons will be added in appropriate locations in the future. The following wireframes are for explaining the basic functionality and design of our mobile application.



The homepage (left) will pop up automatically if the phone owner has already set up an account with our service. If it is their first time using the app/they haven’t created an account then the user signup page (right) will be shown.

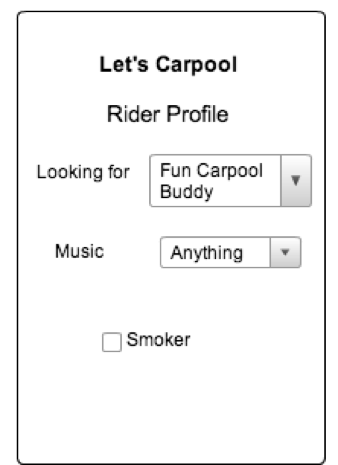
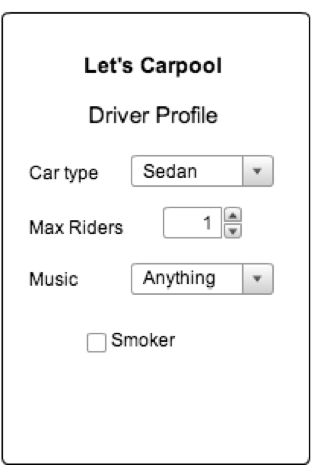
Homepage- Three buttons specify what the registered user would like to do:

1. They are a driver hoping to find riders and would like to post their route specifications (proceed to page 10)

2. They are a rider and would like to find a route in the nearby vicinity (proceed to page 11)

3. They are looking to check in on specifications of past and pending requests along with rendezvous and posted routes (proceed to page 12)

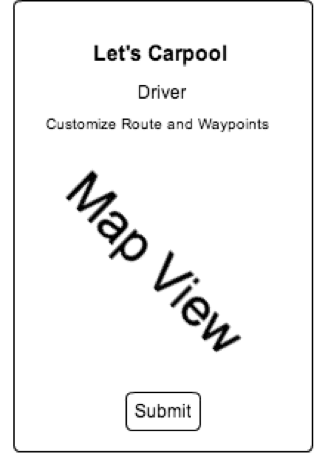
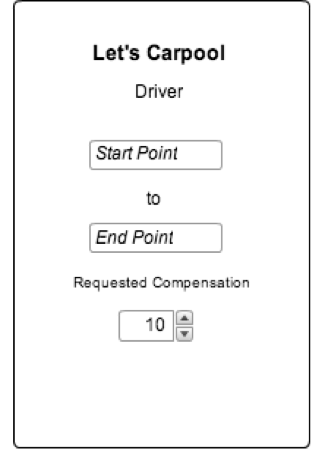
Signup- Collection of key information and checkboxes that will specify as to whether or not the user should then proceed to creation of the Driver and/or Rider profile setups (page 9).



If the user clicked “I’m a Driver” or “I’m a Rider” then they will be redirected to each respective screen above before continuing onto their requested task (page 10 or 11).

The Driver Profile setup will collect various pieces of information that will be publicly displayed to riders looking to find a driver.

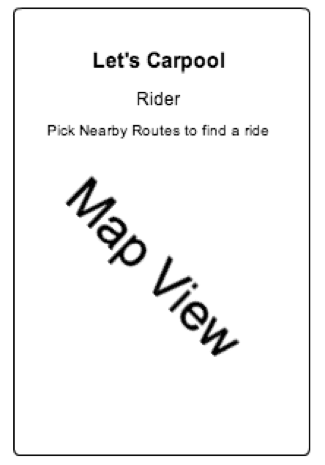
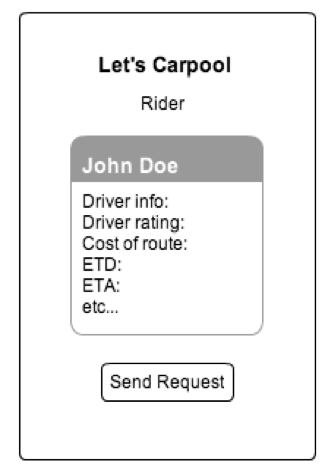
The Rider Profile setup will collect information that can help with drivers to select riders that they would like to pick up. Additional points of interest may be created in the future to better optimize the driver/rider matching.



Clicking on the “I’m a Driver” button will bring you to this sequence of screens which allows a driver to easily enter start point and destination (later we can include waypoints and maximum approved deviation from posted route). The driver can also specify their compensation terms.

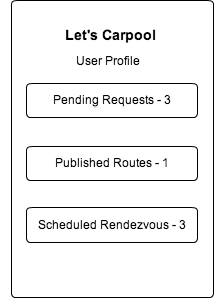
After entering basic route data the user will be able to use our interface (based on Google Maps API) to then drag the highlighted route to use specific roads and further customize their means of travel.

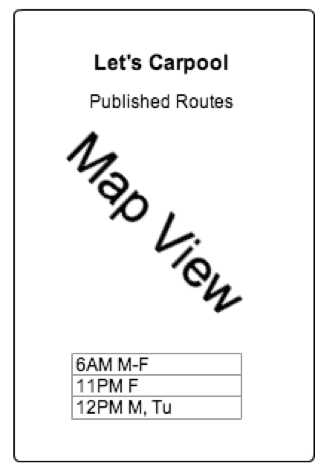
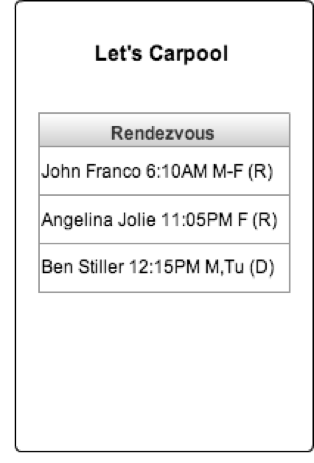
There will be additions to this interface (ex. schedule of specific dates that this route will be traveled... using Google Calendar), but our low-fidelity prototype will probably not include this feature due to scope limitations.



The “I’m a Rider” option will load a map (again using Google Maps) of all local available routes. These routes are updated in real time so that the rider can see where the driver is along their path (and make a request to get picked up if appropriate). Conversely, drivers can see where their passengers are and pick them up if they choose to do so. The map view will include highlighted routes which, when clicked on, will bring up a small tooltip-esk icon that gives destination, ETA, cost, etc. From there the user will be able to send request or continue looking for alternatives on their map.

Clarifying point, the initial map view will bring up local routes by accessing the mobile device’s GPS service and calculating their position.





If the user clicks “Proceed to Profile” then they will be redirected to the above screen (top). From this home screen they can then check the routes that they have posted and any upcoming riders/drivers that they have committed to. Pending requests will also be available, thus giving the user the ability to cancel a request if need be.



Our system will also be able to calculate when the driver and rider have interacted (aka a ride was shared). Pairing rider and drivers by the proximity and distance travelled by their individual mobile devices will allow us to figure when they meet, travel, and split up. Once the journey is complete and both users are on their separate ways, we will send a request to rate their counterpart and leave comments/feedback for public view. This will enable our application to self-regulate while also giving users the experience of expression and voicing of afterthoughts.