Problem Statement

Many students who attend college live far away from home. Many of those students do not own vehicles on campus. When it comes time to take trips home or to other places, some students need rides or are looking for people to share with to help pay for gas. But sometimes finding people in a student population of 40,000 can be difficult or even impossible.

Background Information

Carpooling first became prominent in the United States as a rationing tactic during World War II. According to the 2009 National Household Travel Survey, carpooling represented 43.5% of all trips in the United States. It is a widely-used money-saving strategy, as it results in less wear and tear on cars and less money spent on gas.

A Purdue Ride-Board group exists on Facebook that students can use to carpool to and from school. Some students are trying to find somebody to take them home, while others are looking to take someone with them to help pay for gas.

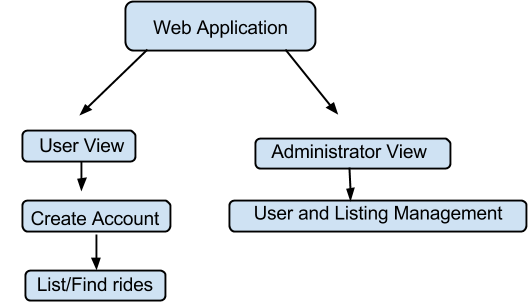
To do this, the users have created a text document that people edit to say “I need a ride going through Ohio” or “I am driving through Chicago this weekend on my way to Wisconsin, anybody need a ride?” This text document’s format is inconsistent and can be hard to read. It does not automatically update, and the page requires manually monitoring the posts to see if a potential ride/driver match has been made.

There are few websites that exist to solve this problem, and none are specifically for college students (and are therefore seen as untrustworthy to many students). Our goal is to create an easy-to-use, automatically-updating, trustworthy, convenient website that students can use to find rides to and from their university with people they see on campus.

Environment and System Models

For our product, we have decided to use Java, because we all are familiar with Java. Java is also a very good language as scalability becomes a concern.

* mySQL: We needed a database to store user data and ride listings.
* Ubuntu Server: Ubuntu Server is world-widely well used and proven server that a lot of businesses and organizations use. It’s also free which is perfect for our product.
* Access API : For ease of management and (I forgot the word), we are going to use API.
* Google Maps API: For map data, Google Maps already is very detailed and robust service, making it very good component of our product.
* SMTP : For security, we use verification e-mails to provide safe environment for users.
* LAMP Server : TO provide web app, we need LAMP server to host a website.



Functional Requirements

* **Website**
  + A website should exist that allows users to easily search for available ride offers or requests
  + Website must be able to allow users to easily search for, edit,
  + The website should hide the exact listings of ride requirements unless the user is already logged in. This is to prevent any unauthorized user from viewing personal information of the users.
* **User to user interaction**
  + Users requesting a ride must be able to browse list of ride offers
  + Users offering a ride must be able to browse list of ride requests
  + There should be an easy way for users to contact each other
* **User to server interaction**
  + Users must be able to create new ride requests detailing their destination, departure time, and any other information they deem pertinent.
  + Users must be able to create new ride offers detailing their destination, departure time, number of open seats, and any other information they deem pertinent.
* **Database to store list of available rides**
  + Database must be able to intuitively list information for users to view
  + Methods should be able to take user input and produce SQL queries to make changes in the database, such as creating, updating, reading, and deleting ride records.
* **Authentication System**
  + Users should have personal accounts in order to track their open ride requests or ride offers. Anonymity should be avoided since it could allow people to abuse the system. Users must sign up with a \*.edu email address since this service will be for college students only.
  + A user email should be displayed to allow users to contact each other. By default, this will be the Purdue email that the account was signed up with, but the user should be able to change this to their preferred personal email address if they wish to do so.

Non-Functional Requirements

* **Reliability**
  + Interactions with our database as well as any map API we will use should not cause the user any errors.
* **Usability**
  + Registration should be easy to do if user provides a university email address of some type. Standard e-mail verification will be responsive and easy to follow and should be a process the user is used to following.
  + Creating listings will simply require you to fill out a short form that will be easy to fill out and should take a limited time before the listing is shown on the website’s listings.
  + Users will use our various search and filtering functions implemented to make it easier to find the listing that will be the most useful for them.
* **Security**
  + Login information will be secured so that no attackers can easily steal user information from our database. Users must take the responsibility of securing their information on their own.
  + E-mail verification will be implemented to prevent people from stealing others’ e-mail accounts to create new user accounts.
  + We will require members who want to register to have a university email address which will help eliminate bots or spam accounts from being created.

Use Cases

* **Account Registration**
  + Potential user registers with a valid Purdue email address, first name and last name
  + Server sends the potential user an email containing a link for verification
  + Potential user successfully registers as a Purdue Ride Mapper user when the link is clicked
* **Logging In**
  + User click register button on main page
  + User enters Purdue email address and his or her Purdue Ride Mapper password
* **Offering a Ride**
  + User select the ‘Create a ride listing’ tab
  + User types the start and destination locations
  + User confirms listing creation through e-mail.
  + Listing will appear on listings tab.
* **Requesting a Ride**
  + User selects the ‘I need a ride!’ tab
  + User types the desired destination
  + A list of results are on the tab, showing users whose journey includes the requester’s destination
  + Requester selects the the desired user and clicks the ‘Request ride!’ button
* **Banning a User (Administrator-level)**
  + Administrator logs in and is brought to an administrative page
  + Admin selects the ‘Users’ tab and is presented with a textbox for an optional search
  + Admin selects the user to ban him or her
* **Deleting a Listing**
  + User clicks ‘My Listing’ button.
  + User clicks ‘Edit’ Option.
  + User can click ‘Delete’ as an option to remove the listing.
* **Edit a Listing**
  + User clicks ‘My Listing’ button.
  + User clicks ‘Edit’ Option.