University of Toronto

CSC309 Phase 4 Documentation

Online Carpooling Reputation System

Team 4: Smart Share

Min Woo Lee, Europa Shang, Teresa Shih, Kobe Sun

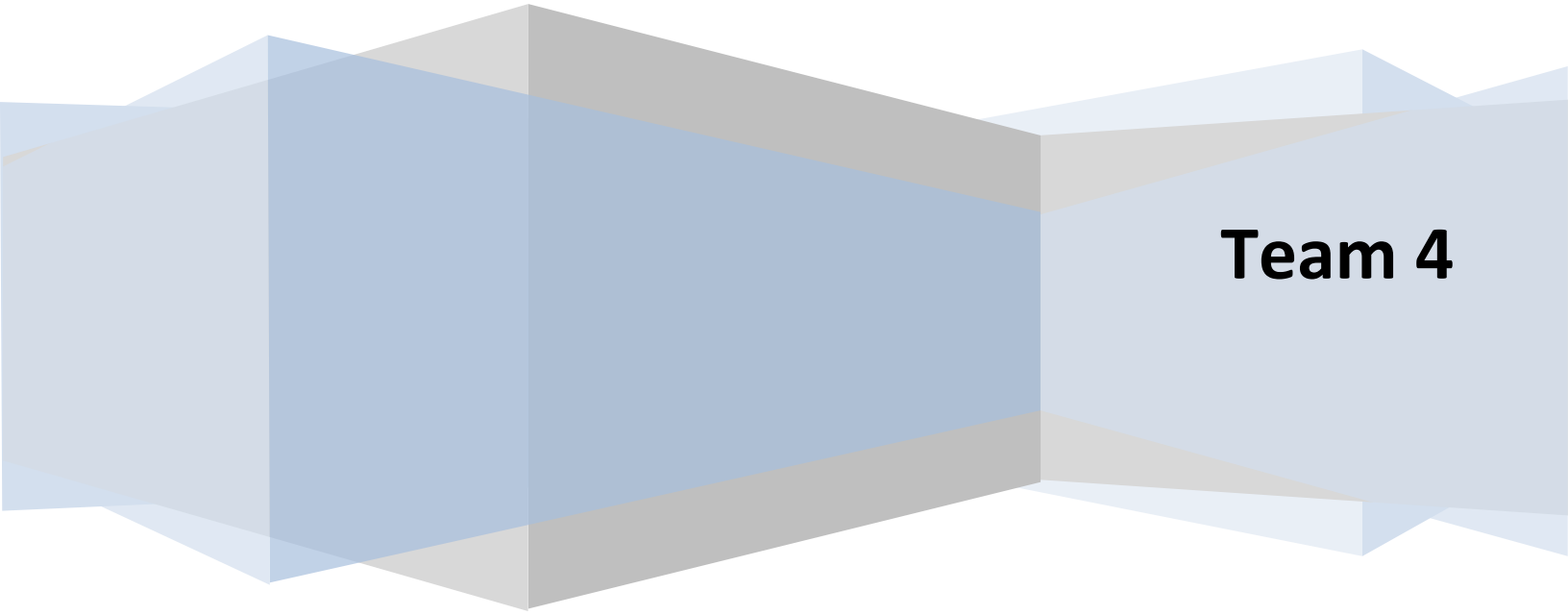


Table of Contents

2.0 Summary of Changes

- Section 2.1.1.1: Account Registration and Activation
  - Deleted information regarding account deactivation
- Section 2.1.1.3: Change Password
  - Removed e-mail notification to user for password change
- Section 2.1.1.4 Forgot Password
  - Add new feature
- Section 2.1.2.2 View Friends
  - Renamed and updated this section
- Section 2.1.3 General Trip Management
  - Removed the section regarding embedded chatting system
- Section 2.1.3.5 Demote from Passenger to Candidate
  - Add new feature
- Section 2.1.3.6 Leave a Trip
  - Updated, passengers will not be allowed to leave trip before the start time
- Section 2.1.5 Search Option
  - Removed the section regarding advance search
- Section 2.1.6 Administrative View
  - Updated, admin can update gas price and promote users to admin
  - Removed the section regarding sending mass e-mails and blocking users
- Section 2.1.7 Feature Enhancements
  - Removed section 2.1.7.3 Real-Time Gas Price
  - Removed section 2.1.7.5 Instant Messenger
- Section 2.2.2 Personnel Roles
  - Regular update
- Section 2.3.1 Interaction with Environment
  - Removed section regarding Instant Messenger
- Section 2.3.7.3 Overall Interaction
  - Removed section regarding account deletion
  - Updated section regarding Friends
- Section 2.4.1
  - Graph change due to database structure change
- Section 2.4.2
  - Data Schema update
- Figure 1.4
  - Data Schema graph update

2.1 Feature and Functionality Specification

(Refer to Appendix A: Figure 1.1)

2.1.1 Accounts

2.1.1.1 Account Registration and Activation

Accounts are the only method through which a user can access the entire system. Account registration will need to be made available in multiple areas of the user interface for easy access. It should be a fast and simple process, which asks the user to fill in 3-4 text fields, including username, password, email and captcha. These fields will allow for future access and retrieval of necessary information of this account used for login. After account creation, a confirmation email will be sent to the provided email. The user must confirm in order to activate the account; otherwise, login will not be possible.

2.1.1.2 Login and Logout

An account is clearly not of much use unless a user activates it and logs into the system. With an account, a user can create/join a trip, communicate with trip friends, and access the reputation system. A user can also access all trips under that account and advanced carpool searching based on reputations, user relations, and other preferences. Alternatively, a user can explicitly log out the system through multiple areas of the user interface. Aside from the regular login process, we will also implement the auto login option by using Cookie. Once the user check “Keep me logged in” box, username and encrypted password will be stored in browser Cookie, and next time when he or she access the website, username and password will be fetched from Cookie to enable the auto login process.

2.1.1.3 Change Password

To aid with account security, a user has the option to modify the account's password. The user must input the old password first, then the new password (twice, for confirmation) before the change is committed.

2.1.1.4 Forgot Password

To help recover the password, a user can utilize this option to reset his/her password. Once the user enter the correct username, an email including a new initial password will be sent to the email address that he/she provided when register. The user will be encouraged to change the password in the email.

2.1.2 Social Network

2.1.2.1 Update Profile

A user’s profile is a collection of personal data, which includes user identity, reputation, friends list, and information about the user’s previous trips. A user can fill/update profile at any time.

2.1.2.2 View Friends

Users can become friends through their trips. All users within the same trip will automatically become friends at the day the trip takes place. Under 'Friends' page, the user will be able to view their friend's profile page and trips created by their friends.

2.1.3 General Trip Management

(Refer to Appendix A: Figure 1.2)

Trip management is the kernel feature of a carpool system. Generally, the drivers' operations on a trip include selecting passengers who would like to travel to the destination in addition to creating, modifying and cancelling a trip entry. Passengers, on the other hand, may join and quit a trip.

2.1.3.1 Create a Trip

This function allows the driver to create a trip thread in which he or she will look for passengers to share the ride. A few mandatory fields, such as trip name, start date, one or multiple destinations, price, number of passengers, car model etc., must be filled out on the "create trip" form. Once the driver submits the form, a trip thread will be created in the database and displayed on the website with public access.

2.1.3.2 Modify a Trip

This function allows the driver to add or modify the details of a trip he created. A notification will be sent to all passengers to notify this change. All trip details can be changed by the driver.

2.1.3.3 Cancel a Trip

This function allows the driver to cancel a trip and a notification will be sent to passengers who have joined the trip. This action may result in bad rating against the driver.

2.1.3.4 Select Passengers

This function allows the driver to select passengers who would like to travel to the destination from a candidate list, possibly based on candidates' reputation. The candidate list is ordered by descending reputation with the exception that candidates at the top of the list should be people from driver's friend list. Besides reputation, driver can view candidates' public personal information as well. Different email notifications will be separately sent to the passengers who have been picked and unsuccessful candidates if the maximum number of passengers is reached. The driver is free to alter the selection at any time. However, again, the driver may receive bad rating if he or she changes the selection too often.

2.1.3.5 Demote from Passenger to Candidate

It is possible that the trip driver accidentally promotes a wrong user from candidate to passenger and the trip driver may want to undo the operation. In this case, the trip driver can use this option to demote the user from passenger to candidate.

2.1.3.6 Join a Trip

This function allows the passenger to apply to join a trip. After the request is sent, the user will be pushed into the candidate list of the trip. A notification for application result will be sent to passengers after the driver makes the selection.

2.1.3.7 Leave a Trip

This function allows the passenger to quit the trip in which an notification will be sent to the driver. However, passengers are not allowed to quit a trip before the start date of the trip. Similar to cancelling a trip, this function may possibly result in bad rating against the passenger.

2.1.3.8 Trip List

This function allows the user to view all the past and upcoming trips that he or she has been or will be involved. For each trip, some useful information, such as money saved, will be displayed in addition to all the basic information. They will also be requested to rate each other.

2.1.4 Rating System

On each trip, a driver and passengers will be able to rate each other, from a scale of 1 - 10, on their driving skill, behaviour, and overall friendliness. The rating system will encourage both driver and passenger to attend a trip they have arranged, and be friendly during the trip.

Our rating system will keep track of rating of the users in two separate pool; a driver rating and a passenger rating. We need two separate ratings because it is often the case where a good passenger may not be a good driver, and vice-versa.

2.1.5 Search Option

In order to quickly find trips available to the user, the user may use the search option to find currently available trips. The search will show the most relevant result on top of the search, such as a trip arranged by one of the user's friends. The user may specify dates, destination and location of the trip for more accurate search result.

2.1.6 Administrative View

The administrative view is exclusively available to the carpool system administrator. Aside from performing regular operations, the administrator will be able to publish global site information such as traffic news, update daily gas price, and promote other users to admin. In addition, the overall statistics

such as total trips, average cost saved, and total number of users will be collected from the database and shown to the administrator in graphic model under this view.

2.1.7 Feature Enhancements

2.1.7.1 Cross-Browser Compatibility

This pertains to an extra feature of our system as a whole. Cross-browser compatibility can often be found an issue with the use of CSS. Extra code and method calls will be used to bypass this otherwise inefficient viewing issue. We plan to allow the user to interact on one browser (i.e. Firefox) with another user on a different browser (i.e. Internet Explorer) as well.

2.1.7.2 News Feed

This pertains to an extra feature of our main page. The expandable widget, displayed after login, will have a list of news for friend updates, including profile updates and trip updates. It will also include any number of alerts about the changes of their upcoming trips. Traffic news will be displayed as well.

2.1.7.3 Google Map

This pertains to an extra feature in order to visualize the trip information, based on the distance, route, and destination of the trip. Estimated total distance and travelling time will also be displayed along with the route map to help interested users find an optimal choice.

2.1.7.4 Notification

This pertains to an extra feature to achieve some functional goals. An important use case of it is when user finishes submitting the sign-up form, he or she will receive a confirmation email which requests to activate the account. In addition, any changes of upcoming trips will be notified through email. Some use cases will also be illustrated in other sections

2.2 Project Plan

2.2.1 Team Organization

Our team is using a variation of agile software development life cycle (SDLC) as opposed to the heavyweight methods of traditional SDLC. Although not set on a specific agile software development method, we are promoting the agile methodology of teamwork and collaboration, sharing of information and being adaptive to change.

The team has decided on weekly meetings where team members get together in person or via videoconference to discuss collaboratively, update on progress and split up tasks. For each meeting, notes are taken which form the basis of the status report. In terms of product development, we are using the agile methodology where the team break tasks into small increments and short deadlines before each deliverable (phases) instead of long-term planning.

2.2.2 Personnel Roles

Technical Publicist	Kobe Sun Teresa Shih Min Woo Lee	Responsible for the technical documentation of the system. Also responsible for updating, proofreading and editing the documentation before finalizing and submission for each deliverable.
Database Programmer	Kobe Sun Min Woo Lee	Set up database for the storage of user profile and trip information. Responsible for creating a logical and efficient database layout for the system
Web Programmer	Kobe Sun Min Woo Lee Europa Shang Teresa Shih	Frontend programmer responsible for the aesthetics of the website layout and logo design. Working collaboratively with database programmers to ensure that the website works cohesively with the database.

2.2.3 Project Milestones

Milestone	Deadline
Phase 1	06/07/2011
- Team formation	
- Project selection	
Status report	06/17/2011
Phase 2	06/24/2011
- Specification	
- Requirements	
- Features	
Status report	07/08/2011
Phase 3	07/12/2011
- Prototype	
- Report	

- Source code	
- Access URL	
Status report	08/05/2011
Phase 4	08/09/2011
- Final report	
- Project URL	
- Source code	

2.2.4 Work Breakdown Structure

ID	Task Name	Start Date	End Date
1	Phase 2	06/07/2011	06/24/2011
2	--- Plan database structure	06/12/2011	06/15/2011
3	--- Documentation	06/12/2011	06/24/2011
4	Phase 3	06/24/2011	07/15/2011
5	--- Set up database	06/24/2011	06/30/2011
	--- Populate database	07/03/2011	07/04/2011
	--- Website layout	07/05/2011	07/15/2011
	--- Prototype	07/04/2011	07/15/2011
	------ Login	07/04/2011	07/07/2011
	------ Register	07/04/2011	07/07/2011
	------ Create trip	07/04/2011	07/08/2011
	------ Search function	07/08/2011	07/15/2011
	------ Rating system	07/08/2011	07/15/2011
	------ Update report	06/30/2011	07/15/2011
	Phase 4	07/12/2011	08/09/2011
	--- Add friend	07/12/2011	07/25/2011
	--- User profile	07/12/2011	07/25/2011
	--- Instant messenger	07/12/2011	07/25/2011
	--- Notification system	07/12/2011	07/25/2011
	--- Google API	07/12/2011	08/01/2011
	--- Final report	07/12/2011	08/09/2011

2.3 Software Architecture and High-level Design

2.3.1 Interaction with Environment

In the case of interaction with browsers, our carpool system aims to be compatible with the browsers we think to be of use in the mainstream public, including Mozilla Firefox 4.0, Google Chrome 12, Internet Explorer 6/8/9, and Opera 11. We've considered supporting mobile browsers, but currently we are focused more on internet capabilities. As three mainstream mobile operating systems, iOS, android and windows phone 7 have very excellent support on website browsing, we believe it will not be a problem to run our carpool system on smartphone or tablet.

Catering to the notion that we aim to have a marketable product by the end of this project, our service will properly operate on Windows, Ubuntu and Mac OS, the three leading operating systems in the market right now.

As one of the main enhancement in our carpool system, the Notification System will be developed based on an open source product PHP Extension PEAR-Mail. The PEAR-Mail package will enable us to use Gmail account to send out the notification email as SMTP port 25 is commonly blocked by the ISP which prevents us from building our own email server.

As for our choice of web server, the Apache HTTP Server is the most popular web server in current use and because we are familiar in its use, the choice was clear. Apache Version 2.2.17 is the one running on our server and we plan to use it in tandem with PHP scripts. In addition, for security purpose, we will enable the SSL channel to protect the connection, because account information may be processed in our system.

2.3.2 High Level System Architecture

2.3.2.1 System Architecture

At the high level, the system will follow the standard model-view-controller architecture in the manner described below. In addition, the technology/language selected for each implementation is provided.

Model - (Database - MySQL)

Our database will be the backend of the system whose sole responsibility is that of data storage and retrieval. MySQL will provide us the required database management system, and will run as a server, allowing multi-user access.

View - (Webpage - HTML, PHP, CSS)

In essence, the user interface of our project. The webpage will be responsible for laying out information in a visually pleasing manner, and providing information to the controller and model as required. A subset of the provided technologies will be used.

Controller - (Editor/Webpage – AJAX, PHP, Mootools)

The controller works as a liaison between view and model aspects of the system. It takes required inputs, performing whatever manipulations are necessary and stores that information as required. Mootools is a JavaScript framework for building interactive web applications, including a set of useful utility libraries.

2.3.3 System Decomposition

There are two major components to the system: the webpage and the database server. Each can be focused on independently in development and will need to be integrated for the final product to be delivered.

2.3.3.1 Webpage

The webpage will be the user's portal to the system, where each user will be able to search and edit trips and interact with friends.

2.3.3.2 Database

The database is simply a storehouse for all required information for the site to operate. It is anticipated to include the flowing information: miscellaneous user information, trips, friend lists, and possibly other information which has yet to be discovered useful.

2.3.4 System Design

The webpage will interact with the database directly. It will perform queries on the database to retrieve relevant information as indicated by the user, and it will also store desired information. The main page will load trip and user account information and then be responsible for saving them to the database upon modification.

2.3.5 Exception Handling

We are going to use 3 level protections to handle any exceptions. At the top level, most of the user input values in HTML forms will be validated by using AJAX. At the time that user finishes a field, JavaScript will validate the input value at client side and, if necessary, send a backend connection to the server to ensure that the value doesn't violate any database rule. After the user click the submit button, some input data, like captcha, will be validated at the middle level on the server. Error messages will be returned to the user as part of the response. Finally, at the bottom level, we have defined the relationships among tables, the primary keys, foreign keys and writing operation constraints (except CHECK constraint as MySQL doesn't support) in the database. Therefore, most of invalid values will be rejected at this level even if they passed the validations from above two phases for some reason.

2.3.6 Password Encryption Design

Network security has become a more and more important factor to determine the success of a website application. In our carpool system, we need to store the password into the database and possibly in browser Cookie (when "Keep me logged in" is checked). We spend a large amount of time on thinking of a good way to protect password from being hacked. The regular and probably the most popular way is to use MD5, because this is an one-way encryption which hasn't been cracked so far. However, we have found that there already exists a huge MD5 results dictionary created by some hacker organization, in other words, MD5 encryption is not as safe as before. Nevertheless, MD5 is still the best way; especially it supports multiple level encryptions. Therefore, instead of doing one level encryption, we use MD5 to encrypt the password 5 times. This is supposed to be complex enough to protect the password. We actually tried to look up some sample, mostly weak, encrypted passwords in the MD5 results dictionary and none of them can be translated. In the future plan, we will add a "hash" field in the user table in database, which can be considered as a verification code generated by MD5(session_id()) and altered each time the user logged in. The "hash" value stored in the database will be compared with that stored in the user Cookie in order to reduce the risk of Cookie hacking. If they are not the same, the user will be forced to log in via the regular way even if the username and encrypted password are matched. Looking from another perspective, we are actually comparing the user session history stored in both database and cookie to ensure the cookie validation.

2.3.7 Class Allocation

(Refer to Appendix A: Figure 1.3)

Each of the classes presented in the class diagram may be associated with a certain component of the system. That class is then considered to be implemented by that specific component. The details about these implementations are provided below.

2.3.7.1 Webpage

All classes will be manipulated or created at the webpage level. This includes the User Interface boundary class and all entity classes. The Search and Login control classes will be manipulated at this level as well.

2.3.7.2 Database

The database is only used to store information and will interact with nearly all classes. However, it is not, in fact, directly responsible for the creation of these classes. More information are provided in Section 2.4: Information Representation.

2.3.7.3 Overall Interaction

Every visitor of the website can use the Search control to find Trips by specific criteria. The information of a Trip is accessible through the User Interface. A visitor can use the Login control to create/login an account, which is represented by a User.

Once a User is logged in, a User can start to arrange a Trip. On creation of a Trip, Destinations will be created automatically with specific order recorded. Also, a TripUser will be created automatically for the User and the new Trip with a Driver status. As other Users apply to join the trip, a TripUser with Candidate status will be created for each request. A TripUser status will change from Candidate to Passenger if the Driver accepts the request, and a TripUser is deleted if the Driver rejects the request of a Candidate. If some TripUser decides to leave a Trip, the TripUser status will be set to Cancelled. If this TripUser was a Driver, then the Trip will be cancelled and the Trip date will be set to null. If not, the TripUser will no longer show on the passengers list. In either case, this TripUser cannot rate anyone on this Trip anymore, but other TripUsers can still rate on him/her.

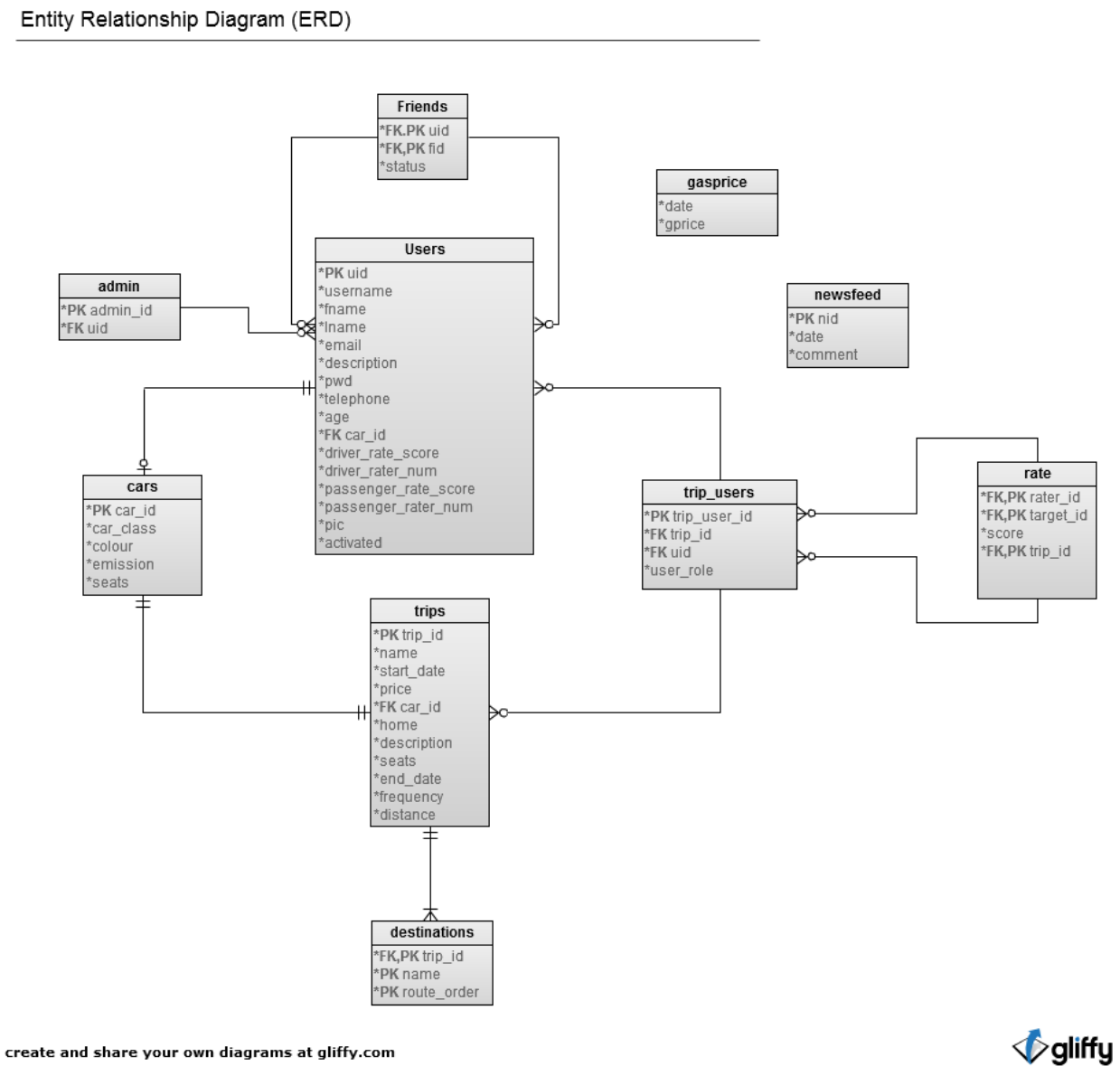
A User can view their friend list which contains an array of Friends (empty on creation of an account). Friends will be automatically created between TripUsers of the same Trip on the date of the Trip happens.

A Vehicle is created when a User updates its profile. In this case, the Vehicle will be the default Vehicle of all trips the user creates. As another option, a User can create a Vehicle on creation of a new Trip and this Vehicle will be used by this single Trip only.

2.4 Information Representation

The Database Management System (DBMS) that will be used is MySQL as it is among the most widely used DBMSs as well as the most popular open source DBMS because of its ease of use, fast performance and reliability. We also use phpMyAdmin as our frontend tool to handle the administration of MySQL for convenience.

2.4.1 Data Model (ER diagram)



2.4.2 Data Schema

(Refer to Appendix A: Figure 1.4)

Admin

Field	Type	Null	Default	Extra	Links to
admin_id	int(11)	No		auto_increment	
uid	int(11)	No			users.uid

Cars

Field	Type	Null	Default	Extra	Links to
car_id	int(11)	No		auto_increment	
car_class	varchar(16)	No			
emission	int(11)	No			
colour	varchar(16)	No			
seats	int(11)	No			

Destinations

Field	Type	Null	Default	Extra	Links to
trip_id	int(11)	No			trips.trip_id
name	varchar(128)	No			
route_order	int(11)	No			

Friends

Field	Type	Null	Default	Extra	Links to
uid	int(11)	No			users.uid
fid	int(11)	No			users.uid
status	int(11)	No	0		

Gasprice

Field	Type	Null	Default	Extra	Links to
date	date	Yes	NULL		
gprice	double	Yes	NULL		

Newsfeed

Field	Type	Null	Default	Extra	Links to
nid	int(11)	No		auto_increment	
date	datetime	No			
comment	text	No	0		

Rate

Field	Type	Null	Default	Extra	Links to
rater_id	int(11)	No			trips_users.uid
target_id	int(11)	No			trips_users.uid
score	double	No			
trip_id	int(11)	No			trips_users.trip_id

Trip_users

Field	Type	Null	Default	Extra	Links to
trip_user_id	int(11)	No		auto_increment	
trip_id	int(11)	No			trips.trip_id
uid	int(11)	No			users.uid
user_role	varchar(16)	No	0		

Trips

Field	Type	Null	Default	Extra	Links to
trip_id	int(11)	No		auto_increment	
name	varchar(64)	No			
start_date	datetime	Yes			
price	int(11)	Yes	NULL		
car_id	int(11)	No			cars.car_id
home	varchar(128)	No			
description	text	Yes	NULL		
seats	int(11)	No			
end_date	datetime	Yes	NULL		
frequency	varchar(128)	Yes	NULL		
distance	double	Yes	NULL		

Users

Field	Type	Null	Default	Extra	Links to
uid	int(11)	No		auto_increment	
username	varchar(32)	No			
fname	varchar(16)	No			
lname	varchar(16)	No			
email	varchar(64)	No			
description	text	Yes	NULL		
pwd	varchar(255)	No			
telephone	varchar(16)	Yes	NULL		
age	smallint(6)	Yes	NULL		
car_id	int(11)	Yes	NULL		cars.car_id
driver_rate_score	float	No	0		
driver_rater_num	int(11)	No	0		
passenger_rate_score	float	No	0		
passenger_rater_num	int(11)	No	0		
pic	varchar(255)	No			
activated	int(11)	No	0		

2.5 Test Strategy and Test Plan

2.5.1 Test Strategy and Staff Responsibilities

2.5.1.1 Strategy

We will use Trac, an open source, web-based project management and bug tracking tool, to deliver wiki knowledge, track the status of roadmap, monitor the svn commit including "diff the changes" and open tickets for reporting bugs, implementing enhancements and assigning tasks.

There are several sections of this project which will require heavy testing from both server side and client side. PHP will be one of the major programming languages we use to implement the project, and due to the reason that it is impossible to perform any test on PHP from the client side, we will use Eclipse PDT + XDebug to test our coding from the server side. We have configured our Apache server to enable XDebug to interfere with project running. Therefore, we can set breakpoint anywhere to help us monitor run-time stack changes. Javascript, CSS and HTML are all running on the client side browser so that we will mainly test them from the client side by using Firefox with Firebug. For the database testing, we will import a script to call SQL query to test the DDL constraints, determining whether it can prevents user from inserting non-sense records. Besides the regular testing described above, the website system will need to be loaded on a selection of browsers to ensure compatibility, including IE 6\8\9, Chrome 12, Firefox 4, Safari 5, Opera 11 and Dolphin Mobile Browser. Apache server log will possibly be monitored for some special purpose such as tracking mod_rewrite and mod_proxy status.

2.5.1.2 Responsibility

Database programmers will be responsible for creating the test script and fix any design flaws. As described in the above section, the database should have the ability to protect itself even without PHP layer. Web programmers will address most of the issues in the script running on the server, including PHP, JavaScript, HTML, and CSS etc. It will also be the responsibility of each programmer to properly document their code to ensure that a general explanation at the function level is provided along with an overview as well.

2.5.2 Test Plan

2.5.2.1 User Interface (View)

The user interface consists of the web page from which users will access their account and trip searching. Basic social networking features such as accounts, friends list, and trip management will be accessed directly using this avenue. Each feature will need to be tested individually, and the questions of cross browser compatibility will be addressed.

2.5.2.2 Database (Model)

The database will house all account and account related information. The integrity of the database will need to be analyzed before it is integrated into the entire project system as a whole, then will need to be rechecked to ensure proper cohesion with other project units later.

2.5.2.3 Data Processing and Transmission (Control)

This is how we utilize PHP, AJAX to handle dynamic data shown on the website and therefore will undergo the most testing. Besides those basic testing methods described in section 2.5.1.1, we will also simulate the actual user to play within the site, producing any situation that may cause the error. When PHP interacts with backend database, we will compare the data sent and retrieved on both sides to ensure the transmission accuracy. Regarding to the test of data processing, some test cases will be written to compare the actual output with the expected ones. Any mismatches will be reported. Assertion may also be used as inline testing while developing.

Appendix A

Figure 1.1: High-Level Diagram

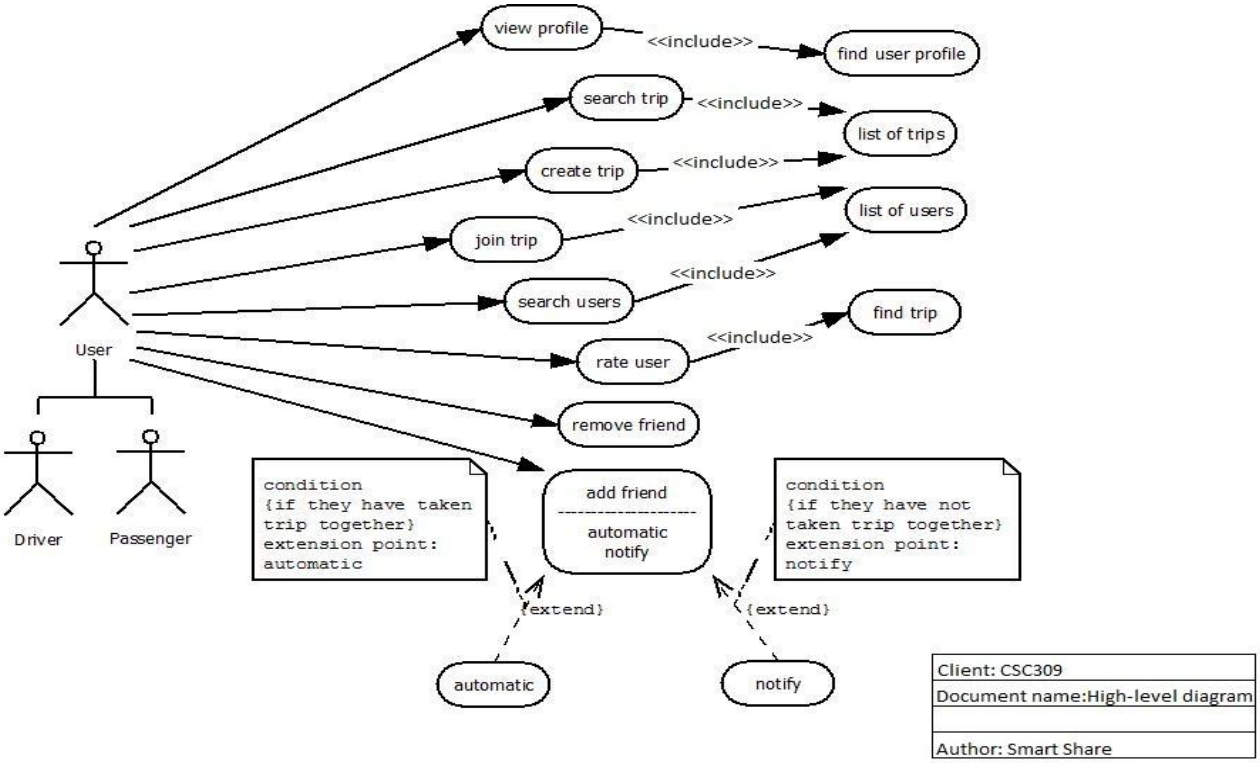


Figure 1.2: Create Trip Use Case

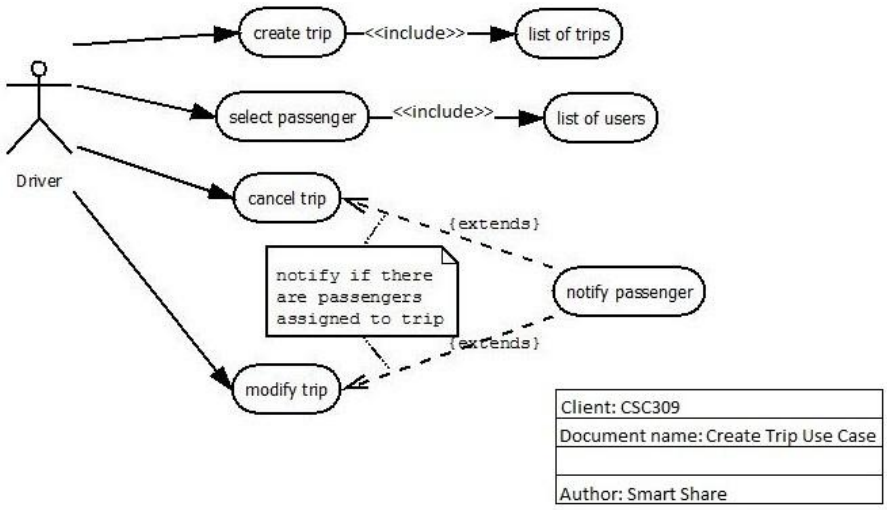


Figure 1.3: Class Diagram

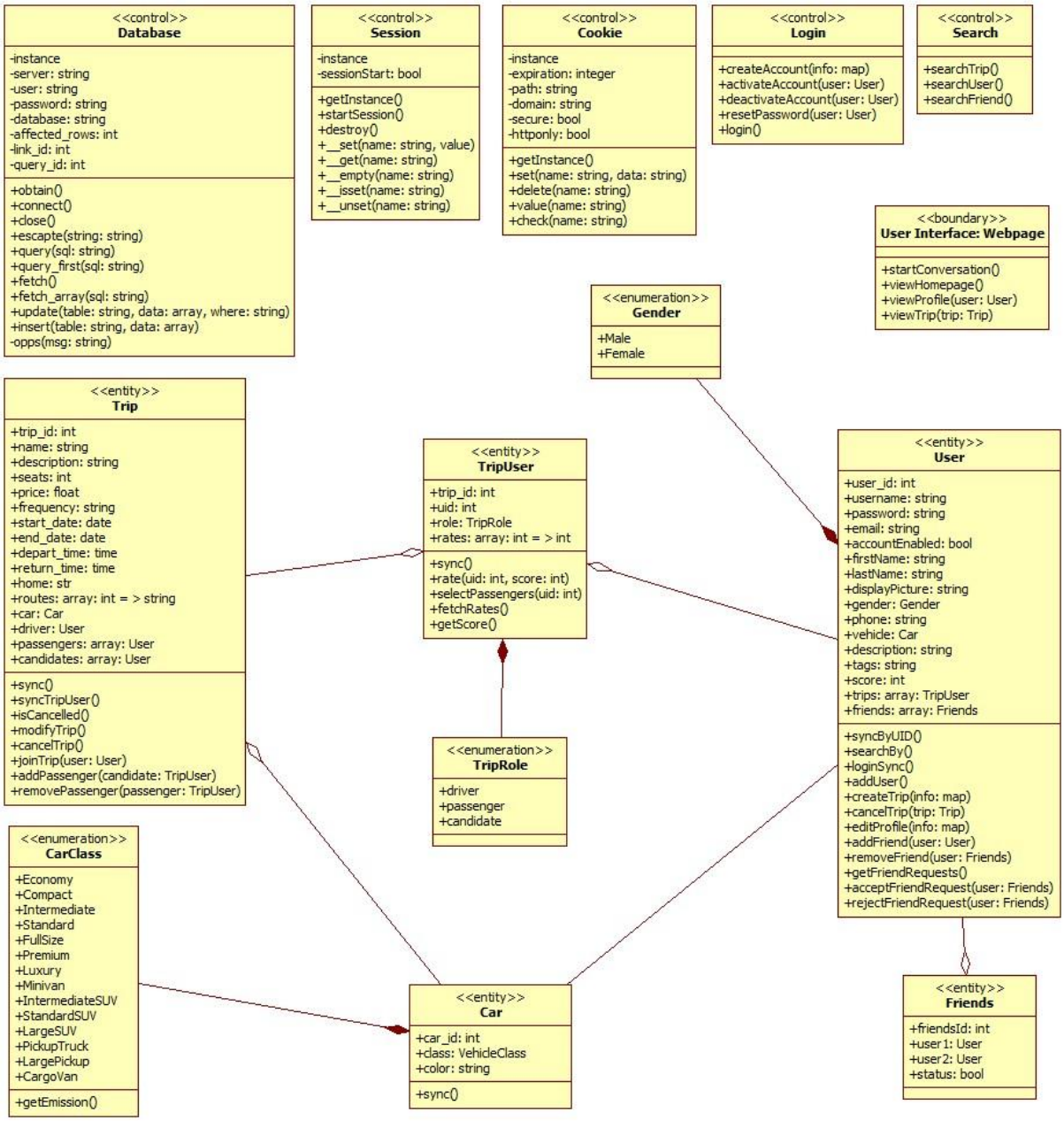


Figure 1.4: Data Schema