SYSC 4907
Forth Year Engineering Project
Final Report

Carleton University
Carpool System

Supervisor:
Professor Gabriel Wainer

Group #87, Members:
Guangjie Joey Deng…100298750
Andrew Lyn…….100299121

Document Status:	Final
Version:	1.0
Issue Date:	April 4, 2005
Department:	Systems and Computer Engineering Facility of Engineering Carleton university

Table of Contents:

List of Figures

List of Tables

Abstract:

This is a final report for the 4th year engineering project of SYSC 4907 entitled "Carleton University Carpool System." The group members are Guangjie Joey Deng (100298750), a 4th year software engineering undergraduate, and Andrew Lyn (100299121), a 4th year computer systems engineering undergraduate. We have proceeded with this project under the supervision of Professor Gabriel Wainer.

1.0 Introduction

Air pollution, traffic and road repair are a big problem within cities. As the population increases pollution and traffic increases and health problems are created that could physically and mentally stress individuals. One solution to these problems is carpooling. A carpool system allows people to travel in groups; therefore fewer cars are required on the road. Having fewer cars on the road is a benefit because it will reduce road use therefore lower repairing cost of the road, and decreases the amount of car emissions therefore improving air quality. In today's society solutions to decrease the large amounts of green house gases are in high demand. A carpool system at Carleton University will help decrease the green house gases from cars in Ottawa and also save students/employees money through shared gas expenses. Another advantage of carpooling is the social networking of students and employees. Carpooling is best described as a mutual agreement between drivers and passengers forming an alternative transportation method that conserves energy, while reducing traveling cost, traffic, air-pollution and road-repairs.

The Carleton University carpool system connects students/employees of Carleton University so that they can carpool together to Carleton University and to other cities. The software helps reduce air pollution and road damage, by providing a cost effective way for students to travel to Carleton University. Our goal is to complete the "Carleton University Carpool System" software and have it setup on a web server to provide service to students and faculty according to our plan. The completed system allows users to share rides within Ottawa and to other cities.

This is the second year that this project was continued. The previous team who work on this project has completed the basic structure of the system; it met the minimal requirement for this project. The “old system” has left many unfinished tasks to be complete and are required to make this system secure for public use. In this project, our goal is to set up a mail server, html server, and improve the carpool system. The main software programming will be shared among the group members in this project. Our Goal is to create a web application that can securely handle large amounts of Internet traffic. **Figure 1** below describes how we will be developing the Carleton University Carpool System by focusing on the following: easy to maintain, easy to upgrade, portable, efficient processing of a user’s request, and secure.

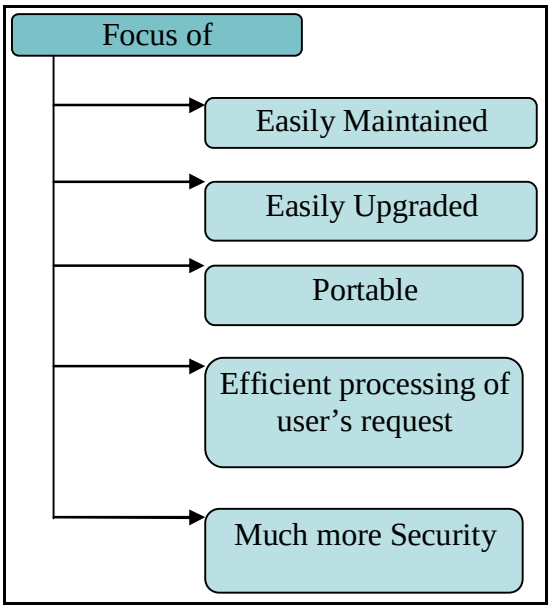


Figure 1 - Focus of Development

Our solution involves java programming, upgrading a MySQL rational database system, and improving the Tomcat Apache Web server configurations. These technologies allow us to create a web based application solution programmed entirely in java that was built with future

expansion in mind. As a group we plan to learn web application development, use of email technology within a web application (javamail), programming an application with greater than ten thousands lines of code and gain valuable team work experience.

This final report contains information that compares the new system with the old system to give the reader a good overview of the work performed and then the report details out all the new features of the Carleton University Carpool System. Also included in the body of this report are future designs. Finally attached to the end of the report are the setup, configuration, maintenance, and user manuals of the Carleton University Carpool web application.

2.0 Body

2.1 System Design

First part of our project is to design the new features that we want to add to the system. After a thorough analysis of functional and non-functional requirements we have created two diagrams to help us to see a clearer image of what the system will be. Those two diagrams are use a case diagram, and a class diagram. **Figure 2** shows the revised use case diagram of the system; it show behaviour of the carpool system from a user's standpoint.

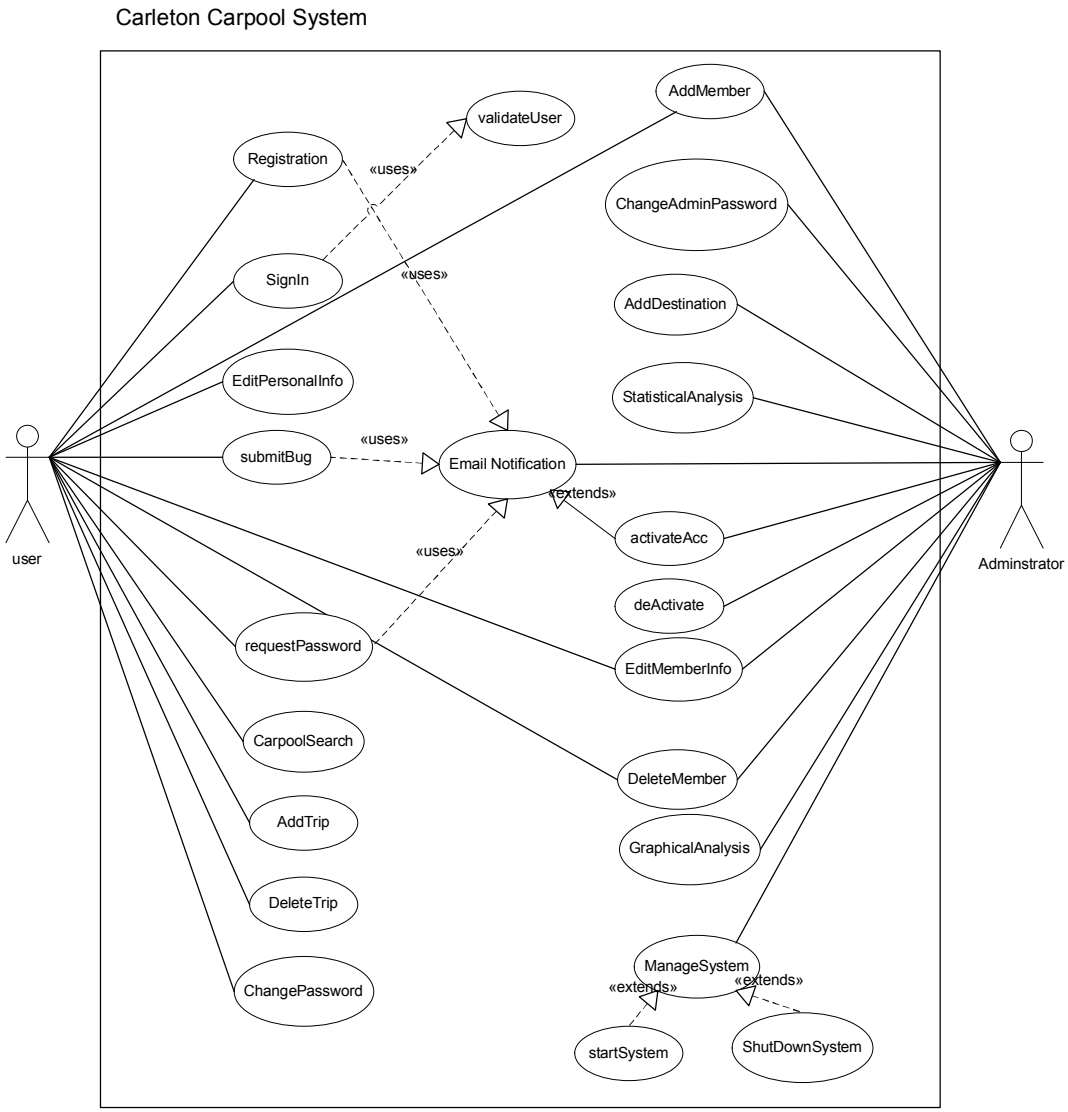


Figure 2

Figure 3 show the class diagram of the system. It was revised to accommodate the changes we made to the system. The Figure 3 shows the types of objects in a system and their relationships. There are three main class types: boundary, control, and entity.

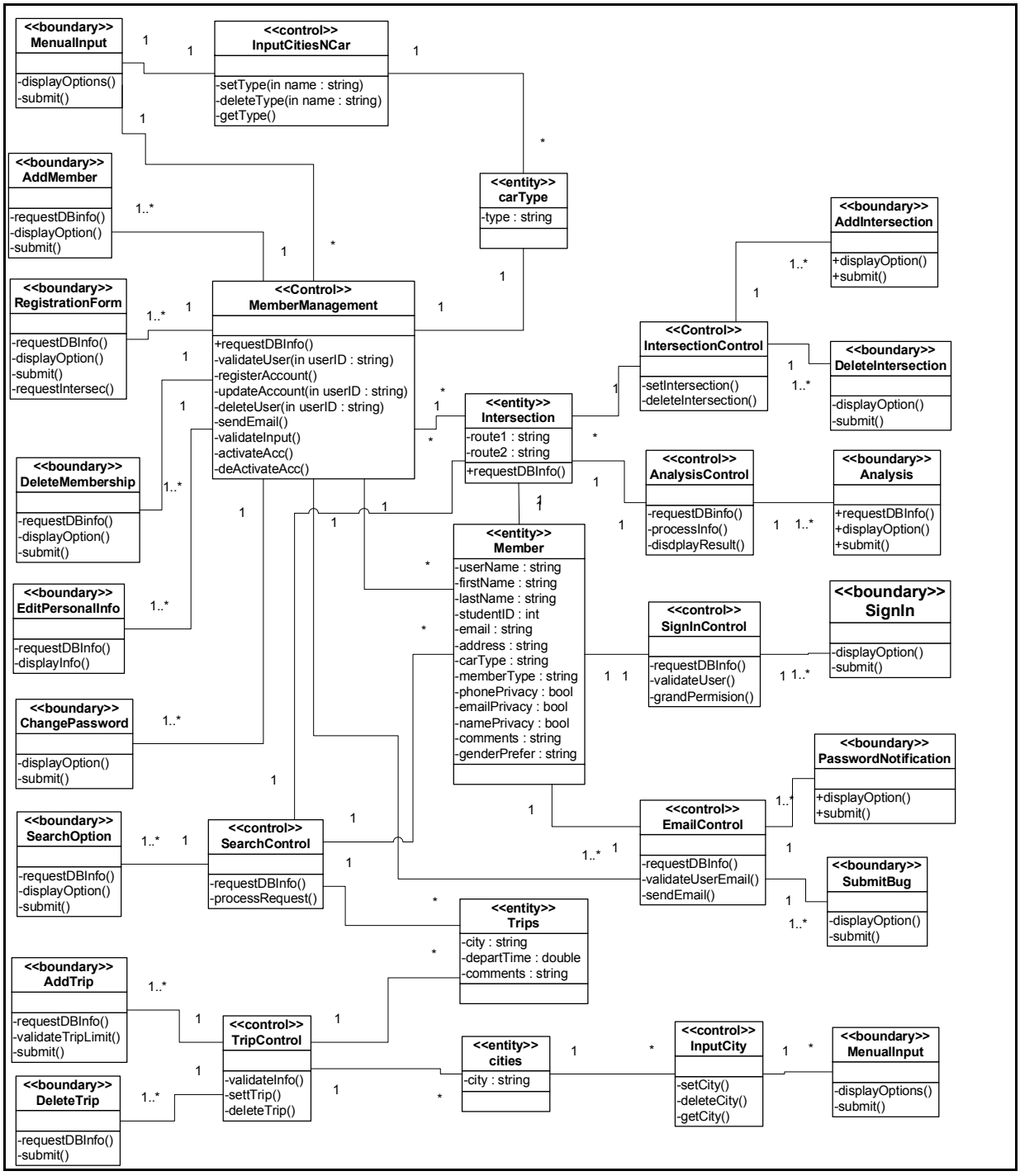


Figure 3

2.2 Comparison Between Old System & New System

The "old system" runs using Apache web server, Java, and MySQL. It provides basic functionalities to the members, however, it was not insecure to use. **Figure 4** shows connections between the components in the system. In the "old system" design users connect to the carpool system through the internet. The apache web server takes the requests to and from the user and reroute them to the java program. When the program completes the requests it sends the data back to the user. Table 1 shows the features available in the old system, and the system's backbone detail.

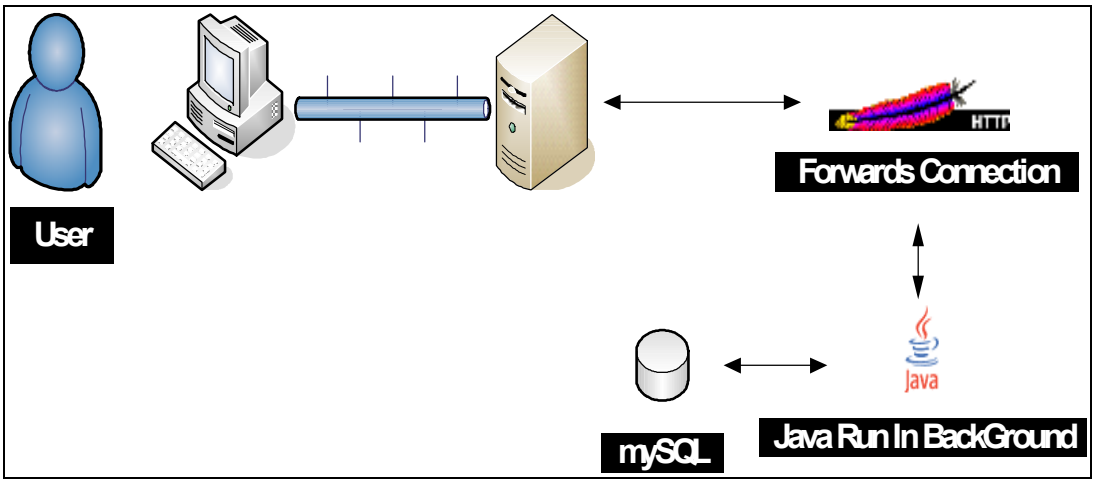


Figure 4

System Details:	Member features:
- 26 java servlets and 8 html files - No Declaring necessary for servlet (Invoker used) - Secure Sockets Layer SSL connections available for sending some data - Username and password tracking from URL - Welcome email on Registration - outdated GUI - no protection - member password displays on the address bar - No protection on website: hacker can hack the website if the hack knows the direct address to the protected pages. - Passwords are not encrypted in the database; it allows the whoever has access to the database to see the passwords.	- Change password - Edit personal information - Forgot password - Login - Basic Register Function - Search (No options) Admin features: - Add intersection - Change password - Delete intersection - Delete member - Graphical analysis - Login - Register user - Statistical analysis

Table 1

"Old System" Database Diagram:

Figure 5 shows the old design of the relational database. ADMIN holds the administrators login information. POSTAL holds all the postal codes for the Ottawa area. MEMBERS contains general information on a member. INTERSECTIONS contains intersections, ROUTES contains predicted routes through intersections. ROUTES is used in the administrator's statistical and graphical analysis options. JOV to K4M Postal code tables used to hold member information.

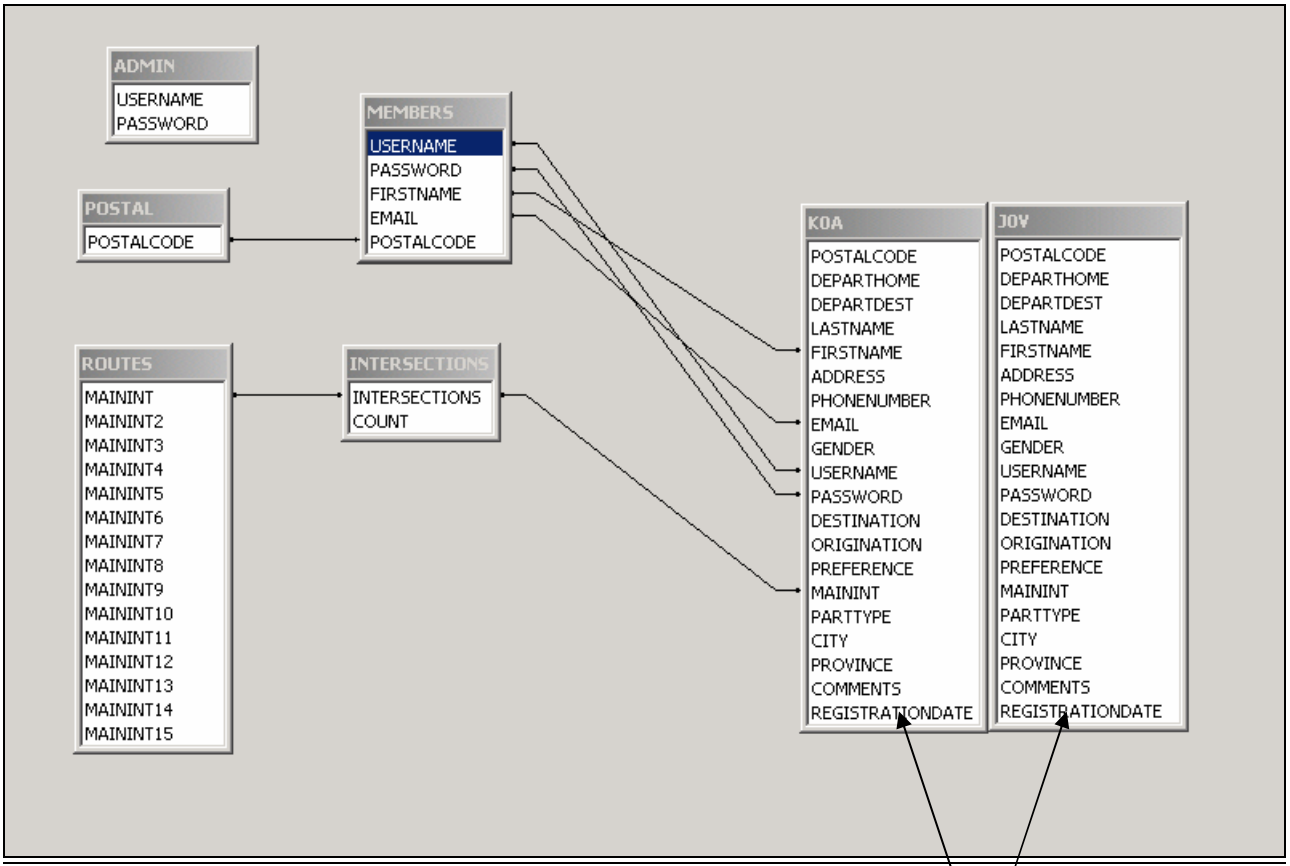


Figure 5

J0V to K4M - User Information is separate by Postal Code

“New System” Overview:

Figure 6 below shows the updated design of how each component in the system interact with each other, it is similar to the old design but there is more security. “Cookies” provide protection to prevent intruders from seeing private website content. Connection Pool Manager manages the connections to the database. Table 2 shows the features available to members and administrator in the new system, and the system’s backbone detail.

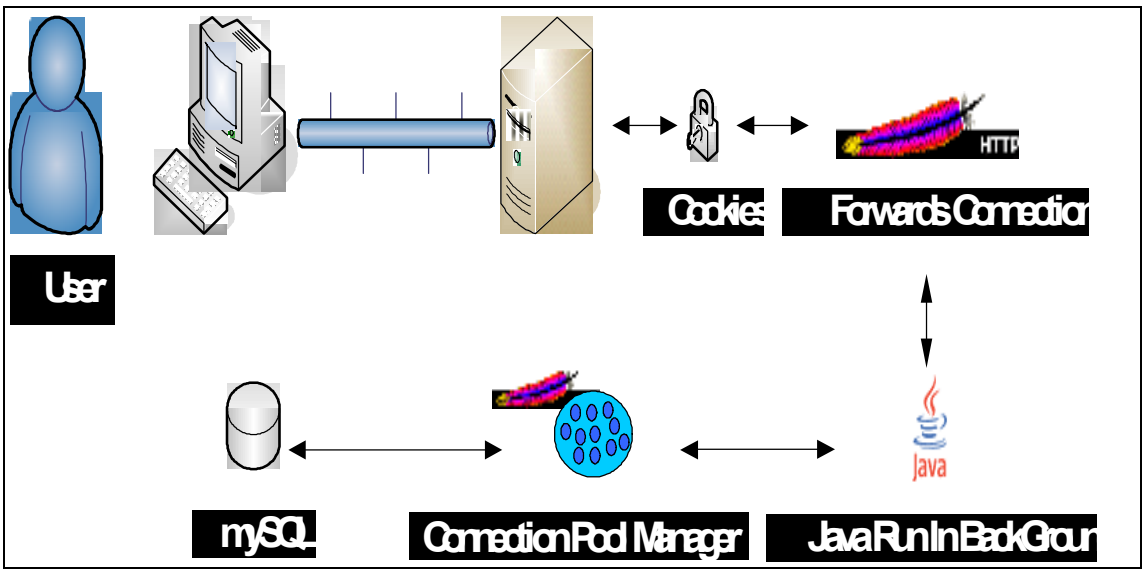


Figure 6

System Details:	User Features:
- 46 java servlets, 5 java classes - Activation of user account tracked - Declared all servlets in a web.xml file (Invoker disabled) - Connection Pool Manager used between mysql and java servlets - Modulized interface into Carleton Branding with Modulized URL's throughout code - Modulized registration information for quick edits - Password encryption (in both database and browser's address bar) - Secure Sockets Layer (SSL) connections available for sending all logged-in user information - Session Management Created to handle servlet web application security and user tracking with cookies. - Username and password tracked by cookie - Welcome email and Activation email on Registration - Updated GUI, now it is used Carleton University's "look and feel"	- Add Trips (to other cities) - Change password - Delete Trips - Delete Membership - Edit personal information (match to the new options added to the registration) - Forgot password (Reset password feature) - Login (Session Management) - Register (New Registration information) - Advanced Search with options - Submit Bugs Admin Options: - Activation of user's account - Add intersection - Change password - Delete intersection - Delete member - Edit user's personal information - Graphical analysis - Login (Session Management) - Register user - Statistical analysis

Table 2

“New System” Database Diagram:

Below is **Figure 7**, it shows the new relational database design for the new system. It has all the basic structure as the old system as mention in the previous page, but it also has new tables to accommodate the requirement. First there is a “trips” table to store trips to other cities and there is a “cities” table to store cities where the members can travel to.

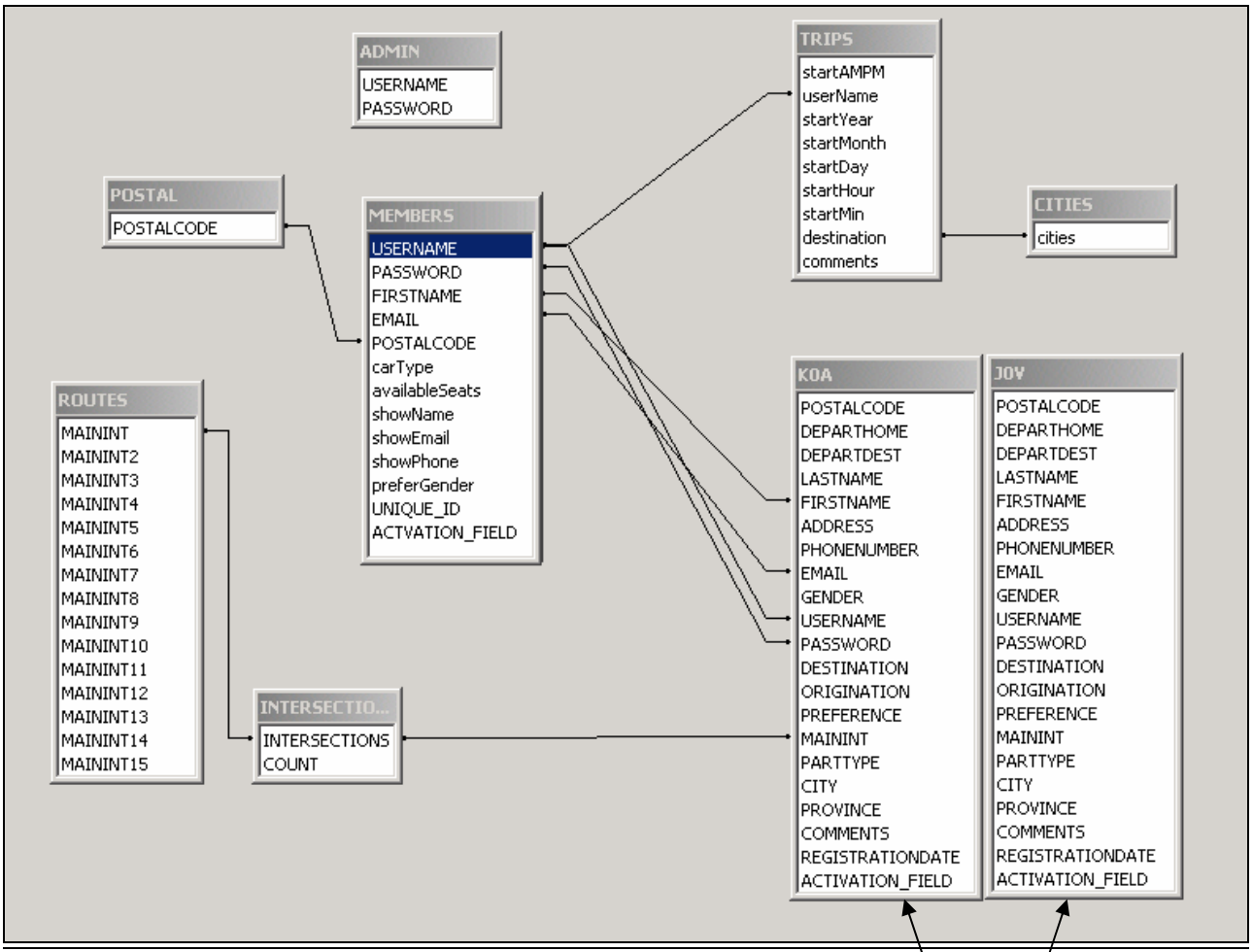


Figure 7

J0V to K4M - User Information is separate by Postal Code

Database Descriptions:

ADMIN	- Holds the administrators login information.
CITIES	- Holds cities that are possible destination.
POSTAL	- Holds all the postal codes for the Ottawa area.

MEMBERS	- Contains general information on a member.
INTERSECTIONS	- Contains an intersection.
ROUTES	- Contains predicted routes through intersections. Used in administrator's statistical and graphical analysis options.
TRIPS	- Holds information on trips planned by members.
JOV to K4M	- Postal code tables used to hold member information.

2.3 New Features

This is the second year of the project, when the project first start by the previous team, it was plan to offer students and staffs a carpooling service. However, there was much more to be done. There are features that were required for to the system, and real security implemented before the releasing the system to the public. This year all the required features have been implemented. They make the system more usable and secure. Those new features are listed below.

2.3.1 Membership Activation

For a user to use there account it must be activated after registration, through an emailing process to confirm a user's identity and email account. Including an activation link URL in an email sent at the time of registration prevents false registration. False registration is done through using someone else's email account or a fake email account. This security feature has become a common function in many web applications that involve open registration.

After a user registers using the online registration form of the Carleton University Carpool System, the user is not granted immediate access to the system instead an email containing an activation URL is sent to the user's email account used in the registration. All functions like the search function of the system ignore member's information of an account that has not been activated. Once a user receives the email to activate their account, they must click on the activation link included to start the account activation process. During the account activation process, the system will check the 32 character string that has been describe as a password in the URL (deceive hackers from guessing the activation code). The 32 character string is composed of a unique field of the user's information and a secret constant that has been

encrypted using MD5 for a one-way encryption use. Once the activation code is matched in the system the account activation process will be completed by setting the activation field to a global constant value. The user will now be capable of using all features of a membership including search and trip creation.

Membership activation feature has been successfully implemented within the Carleton University Carpool System. The general use described above has also been complimented within the system by an administrative interface to set the activation field to the active state or the inactive state. If the administrator disables the account the user cannot manual activate the account anymore though the URL that was sent at the time of registration.

The membership activation feature is a security measure that was implemented last for the registration process. This feature brings the Carleton University Carpool System up to the general standards of a web application's registration process. Web applications with online registration have used many ways to confirm a user's identity but the email membership activation process has been proven successful by the many instances of its use, as seen in a simple search of Google (1).

Designs - Finite State Chart:

Figure 8 shown on the next page is the finite state chart that shows the flow of states in the event of Membership Activation.

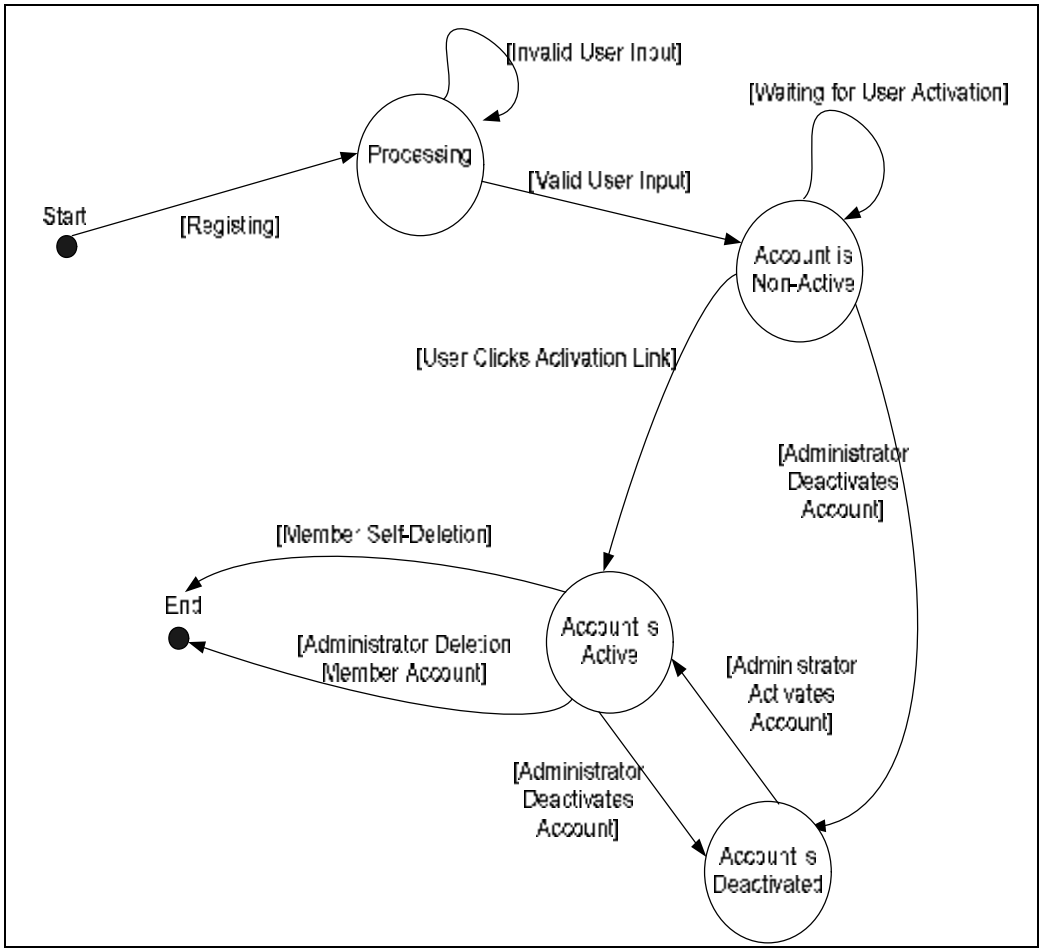


Figure 8

Scenarios:

User Login

1. User access the sign-in page.
2. User enters username and password.
3. System checks if account is active.
4. If the account is active the user gains access to the website

Search

1. User who has an active account access the member's search feature.
2. User select search options.
3. System checks for users with the search criteria and also checks to make sure those results only contain users with active accounts.

2.3.2 Password Reset

To login into the system as a member the user requires his/her username and password information to be entered in the fields of the sign-in interface. If a user forgets his/her password the user would not be able to login to the Carleton University Carpool System. The password reset feature uses an emailing system to allow the user to reset his/her password. By showing that the user has access to the registered email account the user's identity can be confirmed. An email containing a special URL is sent to the user. The URL contains a special 32 character string that is used to start the password reset process. The password reset feature is used in the event of a lost password instead of sending the password to the user's email account for extra security reasons. In the event that a user compromises access to their email account to another person, the system will not disclose a password which maybe a common password used by the rightful user. Therefore the hacker will not be able to comprise other systems used by that rightful user.

On the sign-in interface the user has an option if they forget their password. The user is asked for his/her email address and if the address is found in the database the user is shown a message stating that an email has been sent to the user's email account. The user must login to their emailing system and access the sent email. Once access to the email has been achieved the user can reset their password to their postal code by clicking the URL sent with the email. In the event that the user did not issue a forgot password process, the email informs the user to ignore the email. Since the user did not ask for the password reset, the user's password is still valid and no changes have been made to the system. If the user did forget his/her password, once the user clicks the URL in the email the password will be reset to the user's postal code. After the user resets his/her Carleton University Carpool System password examples are shown to the user describing the format of the password as their postal code.

Reset password has been successfully implemented in the Carleton University Carpool System. Currently the system uses a personal emailing address for the systems outgoing emails. This address should be changed to a local Carleton address in the future. Changing the address can be achieved by editing the constant variables of CarpoolRegister.java, sendEmail.java, EmailAdmin.java, and ChangePassword.java (See Appendix D).

Since this is a common event when a user forgets his/her password the password, the system offers the user reset password to aid the administrator from having to manually change the user's information. A forgot password process is implemented in most web applications where the users have to use a username and password to access the system. This feature makes the Carleton University Carpool System a much more user friendly application to meet the need of the users who commonly forget their passwords, as evident in most large systems (2).

Designs - Finite State Machine:

Figure 9 shown below is the finite state chart that shows the flow of states in the event of Password Reset.

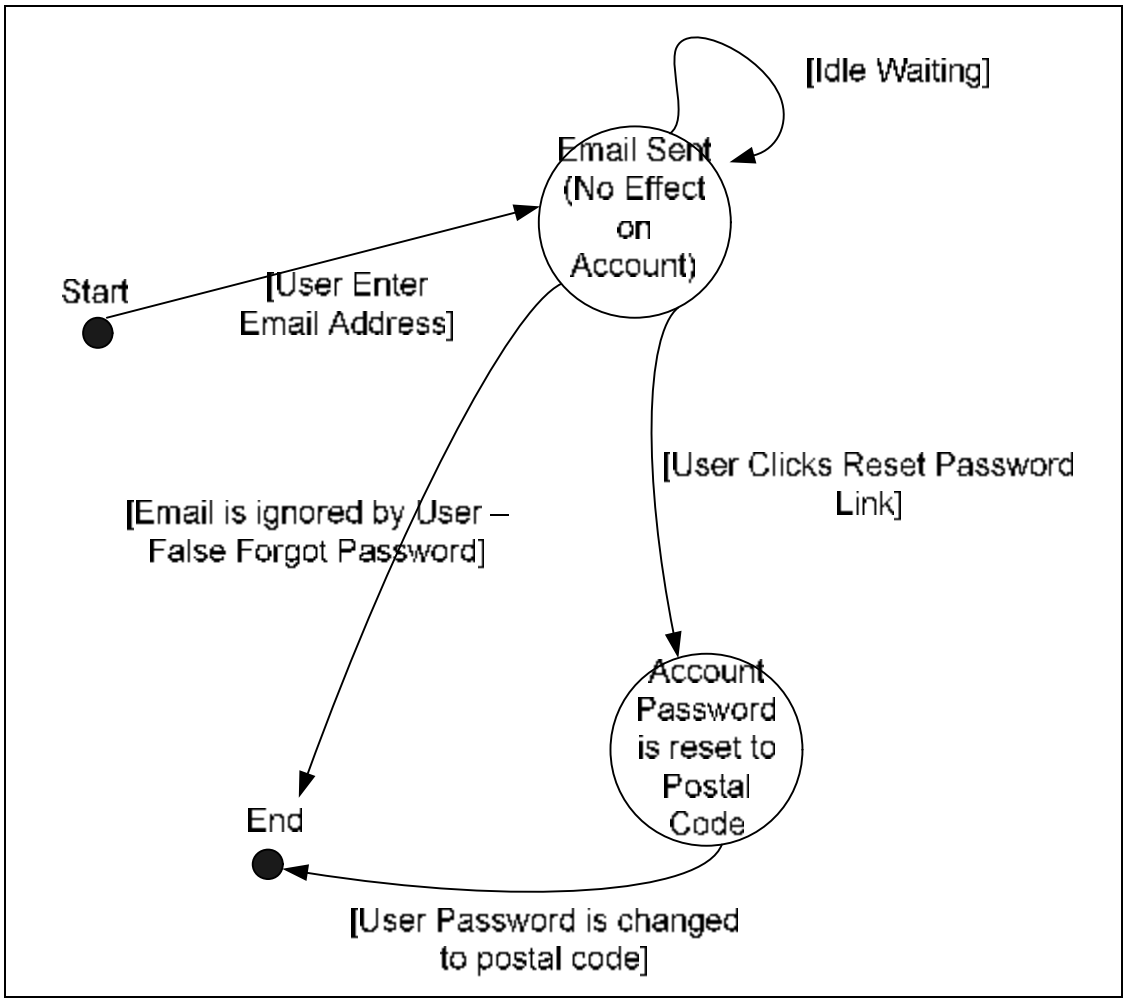


Figure 9

Scenarios:

User Forgets Password Process

1. User enters email address.
2. System checks for a valid email address and then sends an email to the user's email address.
3. User receives email from system and clicks the reset password URL
4. System checks if URL has a valid 32 character string to confirm the reset.
5. User's password is reset to his/her postal code.

2.3.3 Session Management

When a web application restricts access to only user's who are registered in the system it is important for the system to track users logged-in and only give access to the member's interface if the user's identity can be confirmed. Session management is the term used to track a user's access to a web server's content. A session uses a cookie to identify every single user that accesses any web content of the web application. The cookie is server-side, this means that information associated with a user is stored on the server and not in the cookie stored on the client's machine. This cookie stores the member/administrator's password and username which can be used to validate the user access rights to the requested interface of the web application. Session management describes the way that the sessions (cookies) are used to maintain proper logins for user tracking. A system security beyond the server-level can be found in the Carleton University Carpool Systems' Session Management.

Protected interfaces of the application are the administrator's interfaces and the member's interfaces. These interfaces are protected using session management in the web application. The Carleton University Carpool Systems' protected interfaces checks every time they are accessed to determine if the current user has the right privileges to be given access to the interface. All public interfaces (sign-in interface) do not check for specific information in the cookie of the current user but the cookie does exist for tracking purposes. This is not considered an excess since the cookie is empty for user's not logged in, and does not put excess load on the server. The cookie will log the user out if the user is inactive after a given period, which can be declared at the server level in the web.xml file (See Appendix C).

Session management has been successfully implemented in the system. Any unauthorized attempts to access a protected interface will results in the user being redirected to the sign-in

interface. The system's session management properly handles browsers that do not take any cookies by testing for browsers who reject cookies. The system could be design to handle a complete server side cookie based system by passing the cookies unique id in the user's URL, but is currently not implemented in the system. All user tracking systems use some kind of cookie and/or session management system to maintain proper access control over the system.

The Carleton University Carpool System is ready to handle random visits by users on the world-wide-web with ease, knowing that the protected interface have been implemented using an access control session management system. Since the system has session management implemented the system can easily be upgraded to track any other information on the user's behaviour on the web application. The system tracks the user's information using a cookie class called CarpoolCookie.

Design - New Sign-in - Sequence Diagrams:

Figure 10 shown below is the sequence diagram that shows the flow of states in the event of checking member's account when users sign in.

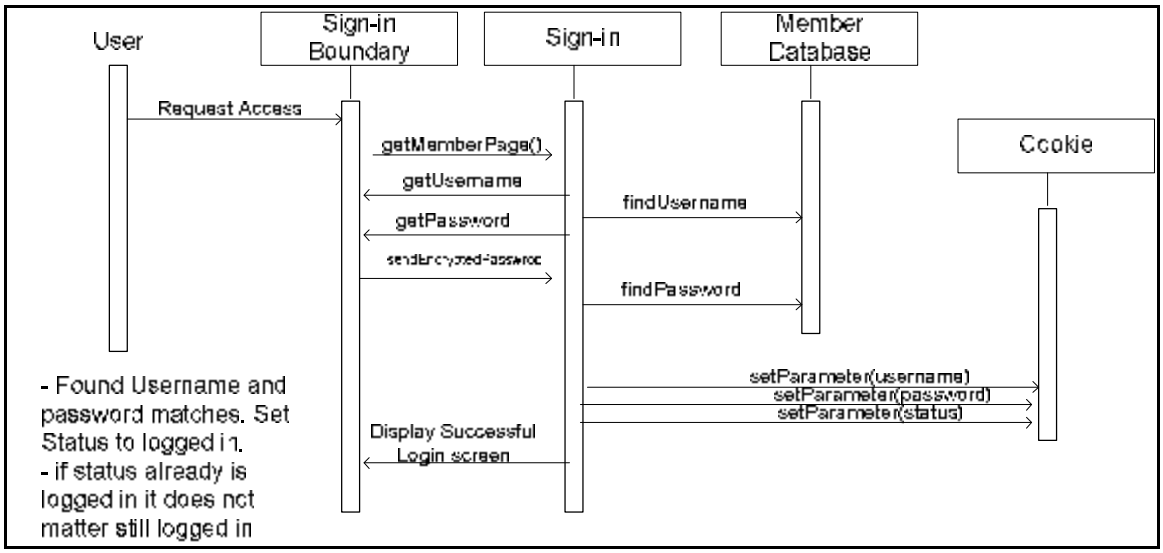


Figure 10

New Sign-out - Sequence Diagrams

Figure 11 shown below is the sequence diagram that shows the flow of states in the event of checking member's account when users sign out.

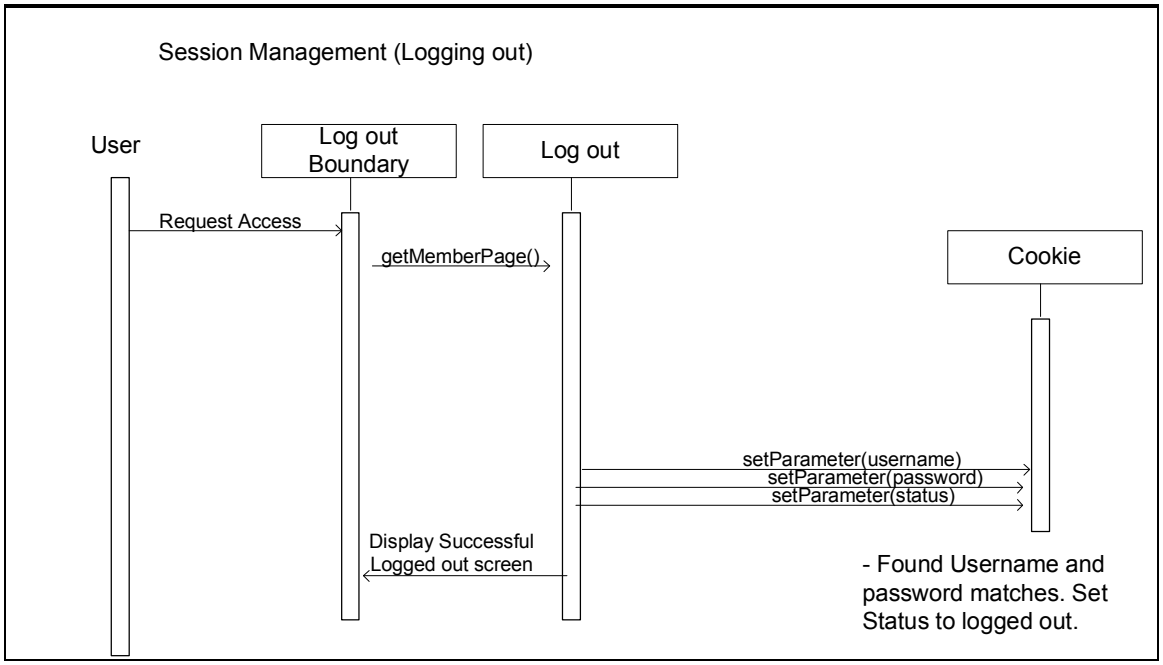


Figure 11

New Member/Admin Page Access – Sequence Diagrams

The following sequence diagrams are used to illustration sign in and sign out processes. **Figure 12** shows the scenario that grants access to protect pages after sign in. **Figure 13** shows the scenario that denies access to private pages when illegal member status is presented to the system. **Figure 14** shows how cookies are generated in the system. As soon as the user access a webpage a cookie is created.

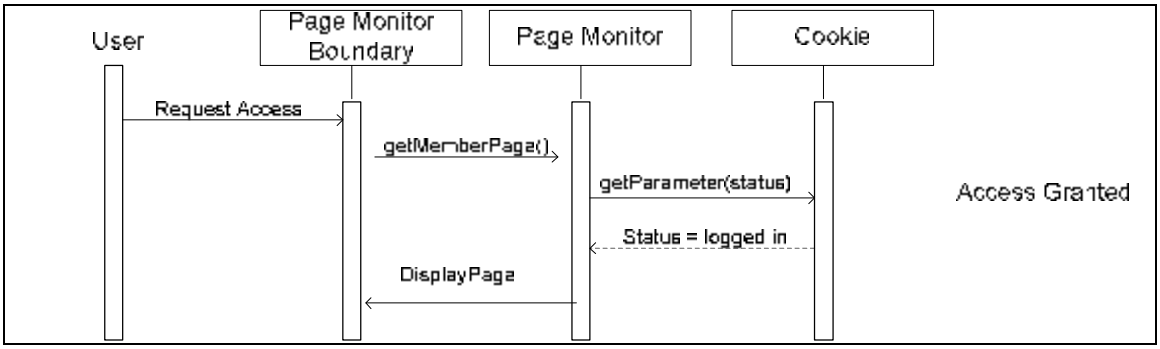


Figure 12

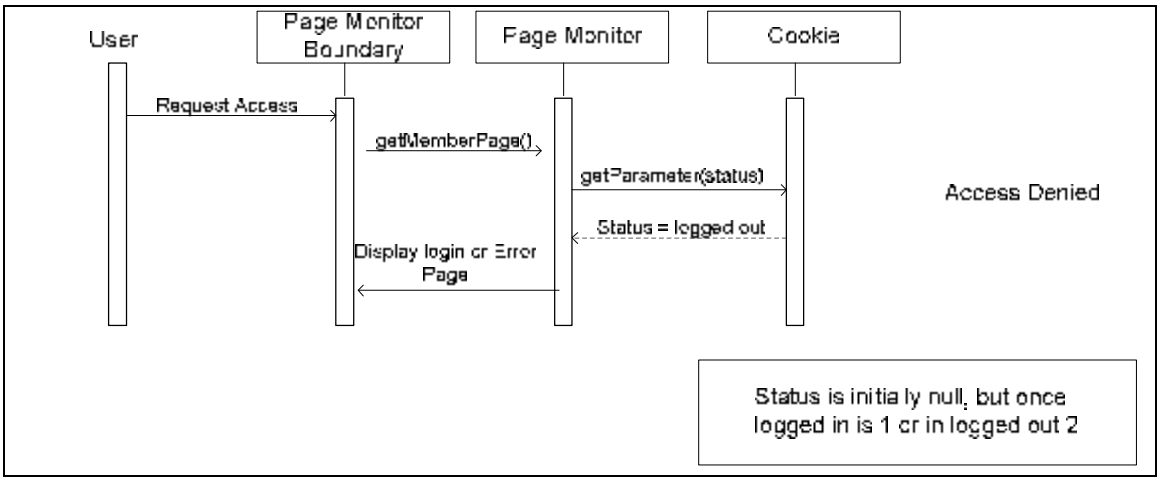


Figure 13

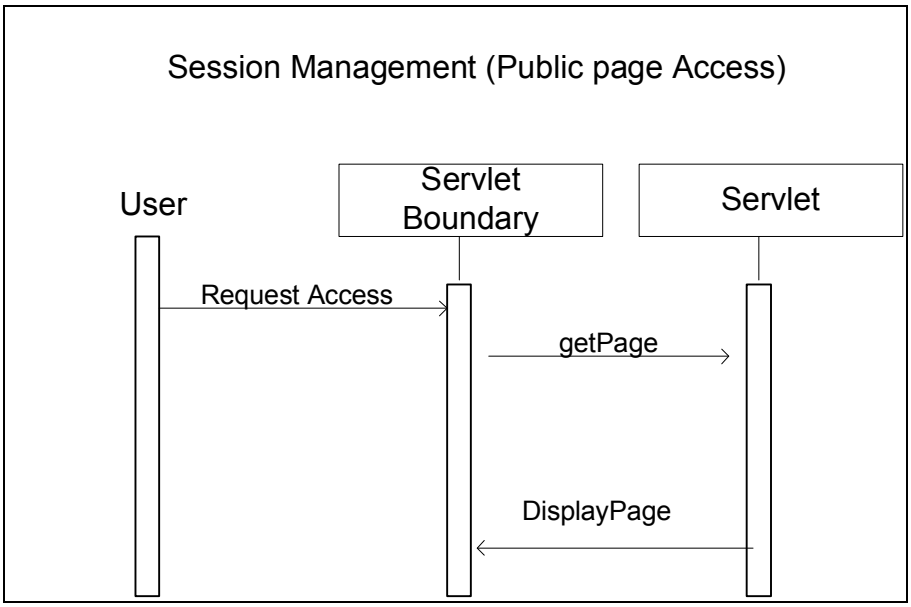


Figure 14

2.3.4 Modulizing Error

Applications that have thousands of lines of code there will exist hundreds of possible outcomes situations. When errors occur in the system, debugging needs to be handle in a manner that can be planned and organized. The end users (clients) need to understand that an error has occurred in a familiar manner so that user's confidence in the system's image as a professional application is maintained. Modulizing errors in the system allows the developer to control the application behaviour quickly and easily, so that in the event of upgrades or new features the system remains consistent in the way it handles errors and behaviour can therefore be predicted in the error of common errors.

Normal operation of the system some user errors when interacting with the system can be predicted for and handled in the system. Errors that the user has no control over are major errors that need to handle with common interface for the user to understand the situation. These major errors can be seen as common or critical in certain features of the system. All exceptions of the system are handled through a common error layout and logging system in the application. These errors can then be sent to the administrator or the user may just ignore and continue to try to use the system in the case of a random error that was due to some odd situation and normal use can continue.

Currently the Carleton University Carpool System uses a common error interface for exceptions in the servlets (application code) and for some common predictable errors in the system. This has allowed us to get the users of the system involved in the debugging of the system. The error is displayed to the user with information that will be helpful in debugging the system of errors. The user can then sent this information to the administration through the bug submit form detailing out the process that caused the error. The administrator or developers can

then use this information to find the source of the error. This method of debugging has already showed value in multiple situation where the end user found a bug and submitted it to the developers for debugging purposes.

The system can be seen as a system that can be released to the public since the error system is intact. The modulizing of errors in system will maintain user's confidence in the system to handle all error situations in a controlled manner. General use of the system will show that the system acts in a controlled manner at all times because the error system fills in all the situations that cannot be planned for in advance.

Designs - Sequence Diagrams:

- Specific Error Message shown on figure 15. These are errors that only occur in one area of the system and occur as not often as common errors.
- Common Error Message shown on figure 16. These are errors that occur in most of the servlets and are assumed to occur during normal use of the system.

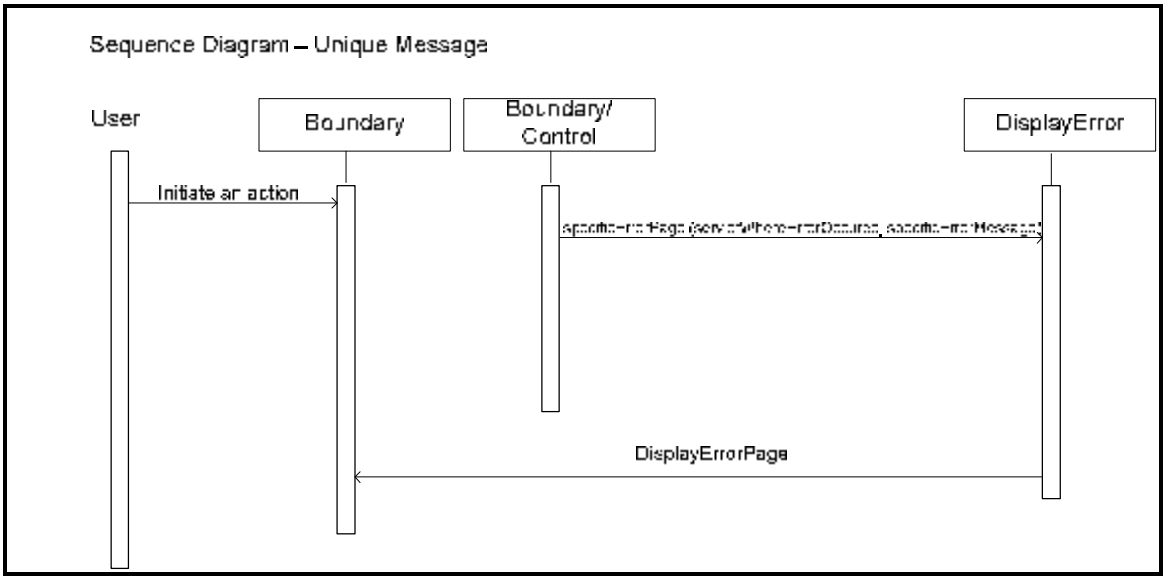


Figure 15

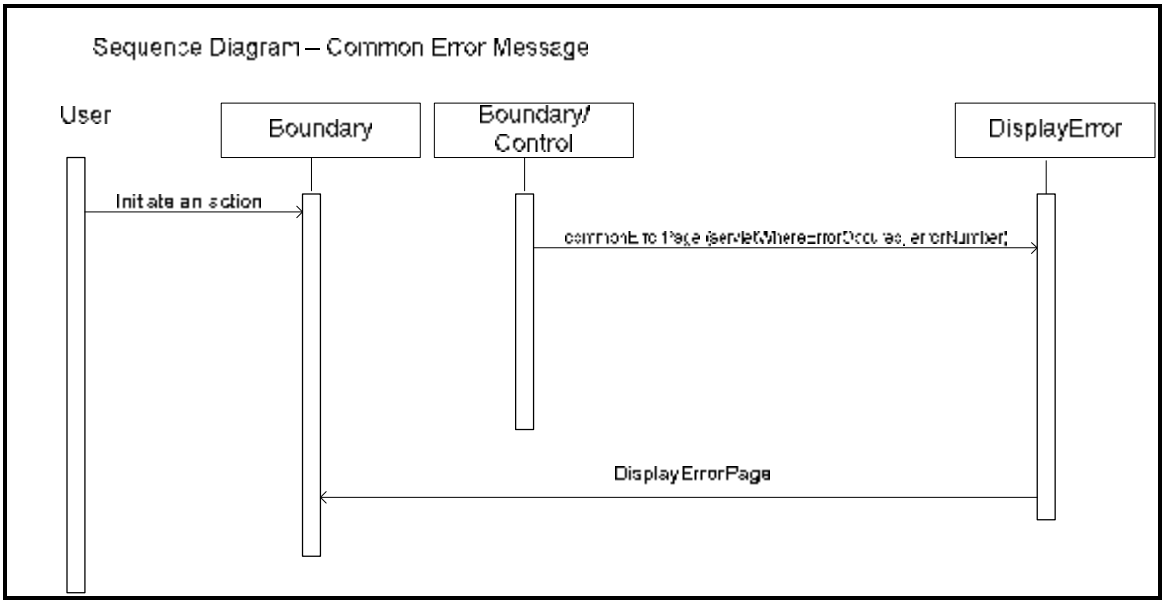


Figure 16

2.3.5 Registration Code Modulation

Information in the database can be manipulated in many interfaces of the application. The user has access to the information, the administrator has access to the information and the general public can produce new information to be inserted into the database. The information common to the registration process is affected by all users the members, administrator and the general public adding new information to the system. When a user registers new information into the database, when a member edits his/her information or when an administrator enters new or edits registration information in the database, the system will be handling the same fields of information. Since this information is common to all the situations just mentioned, the integrity of the system relies on the fact that all fields are included and handled in a common manner. To have the Carleton University Carpool System, handle the information in a modulated fashion portion of the application will allow the system to easily and quickly be upgraded or debugged.

When a user request the registration fields to either register for the first time or as member for editing their information in the database, the system gives the user actually the same interface with minor changes unique only to each situation. When the administrator manually registers a user or edits a user's information the database he/she is given the same interface as the users registering or editing their information with minor changes unique to each situation. The majority of the registration code is common to all these situations, allowing the system to maintain a consistency between these interfaces so that the changes in one section affect all the other interfaces. Adding new fields to the system is made simple with this setup, because all other interfaces are upgraded at the same time adding both the initial insertion of information and editing behaviour.

Modulizing the registration information has been completed and is now up and running with the same quality as before with maintenance in upgrading and debugging a simpler process in the fact that changes need only be made at one location. As more changes has had been required the changes have quickly been made without worrying as much about the integrity of the system and data in the database because of the new modulizing of registration information.

The system development has quickened after registration information was modulized because new information required from the user could be quickly added to the structure of the database and application, so that more focus could be made on the new feature and its requirements in the system. This feature reflects the focus of development of our team: easily maintained, easily upgraded, portable, efficient processing of user's request, and security. The portability of the system can be seen in modulizing of the system because the fact that the system can now be quickly changed for a new purpose/focus besides Carleton University because changes would be more easily made. Modulizing the registration information makes for efficient in processing of the user's request because the processing of the information is the same for each interface that uses it. Since the process is the same the efficiency can be kept at a high level. Security is seen in this feature for the same reason that the process is the same for every interface that uses the modulized code this means that the security can be easily tracked in one location. Modulizing of the registration information is seen as a mandatory requirement of good systems, because it follows a more object-orientated approach for which java servlets function well in this fashion.

Designs - Class Diagram:

Below is **Figure 17**, it shows the new design for old sequential codes. The diagram shows how we generalization help the programmers to reduce complexity in the code and provide easy way to reuse the codes.

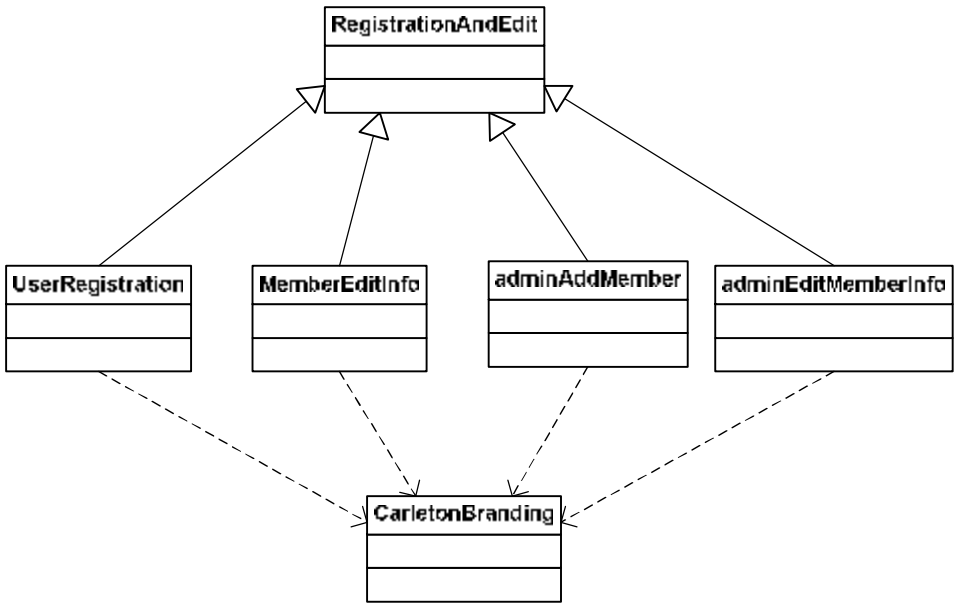


Figure 17

2.3.6 Carleton Branding

Every interface in the system should have a common look and feel so that the user can recognize the layout of the system and identify it as the Carleton University Carpool System. A common layout will allow the user to easily navigate the website and will aid the user in understanding the location of all functions that could be used by the user. Every request of a user to display an interface should go through this common layout. The Carleton University Carpool System uses the Carleton University Branding developed for websites of Carleton University. This is composed of pictures, titles, menus, navigation, contact information, search feature, and main content.

As we developed the software we had edited the original branding to be able to maintain the look with the new functionality created. When we decided to use the new Carleton University Branding all interface were changed to use this layout, so that the web application allowed the user to connect the system by common look and feel to Carleton University. When a user request a interface the system creates a Carleton Branding object that is used to display the systems information in an orderly fashion.

Carleton Branding has been successfully implemented into the system and is used by all the code to display an interface with specific content. The menus developed for the Carleton Branding are public, administrative and member menus. The menu displayed is dependent on the interface being accessed and user's access rights outlined in the session management section. Carleton Branding is used in all forms of interfaces including errors interfaces, administrative interfaces, member interfaces and public interfaces.

The Carleton University Carpool System gives the user access to information pertaining to students and employees of Carleton University. The system therefore should generate a user's response to the system as to be able to recognize that the system is affiliated with Carleton University by the common look and feel created by the system. From general response from students at the poster fair, students recognized the potential positive impact that the system would have on students and employees at Carleton University. This was evident by their positive interest in the system as to when the system will be in place and active. A large part of student being able to trust a web application with their personal information is in the interface's layouts that can either positive associate the web application with trusted sources or negative ones. The Carleton Branding since it is in use with the Carleton University main website, allows user to realized and quickly associate the Carleton University Carpool System with Carleton University.

Designs

Figure 18 shows the Sections of the Carleton Branding Interface:

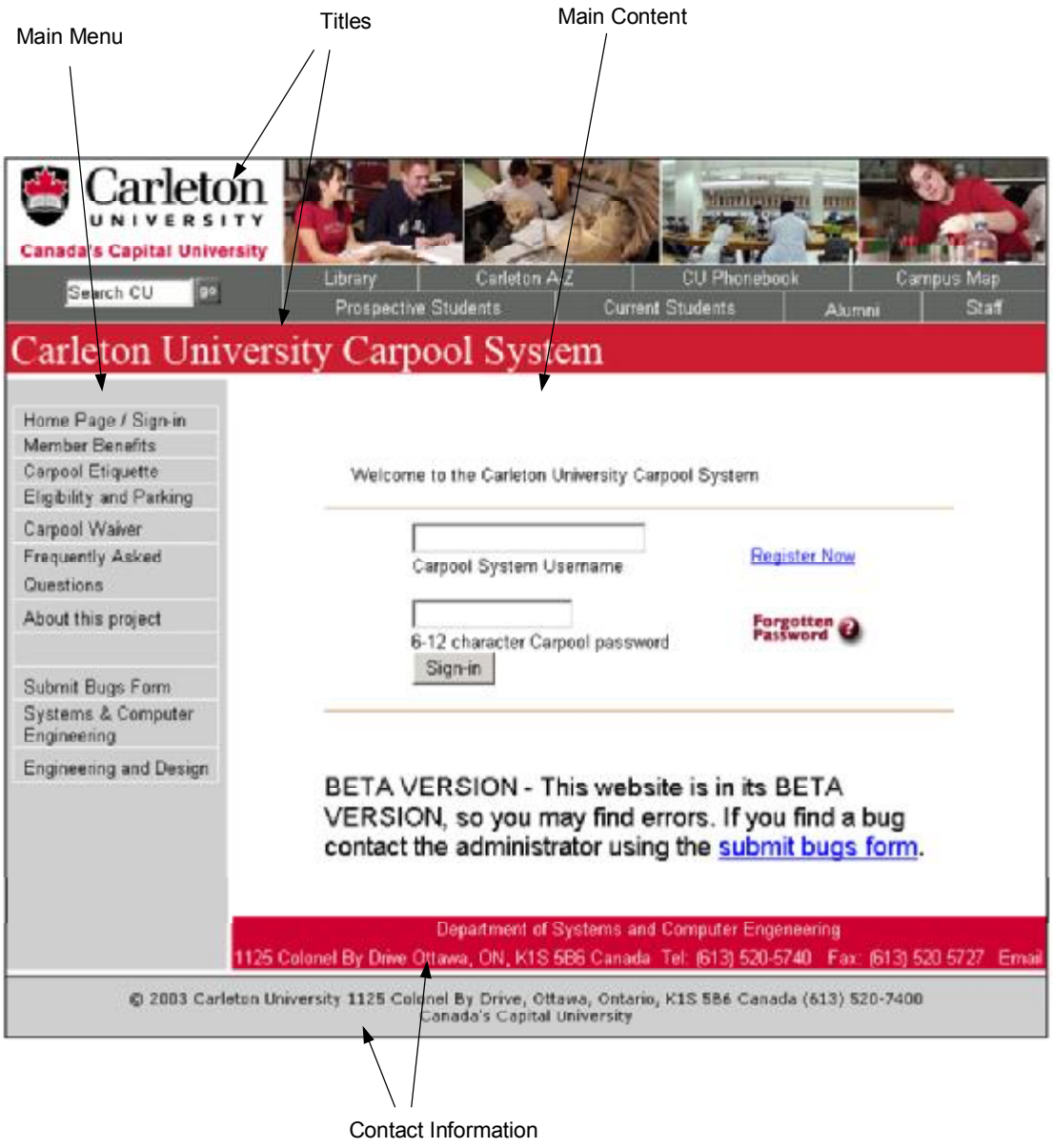


Figure 18

Main Menu – Pubic, Administrative or Member

Main Content – The specific content being accessed by the user, where the main information is displayed to the user. This section is directly associated with a servlet and that servlet will use the Carleton Branding to create an interface.

2.3.7 Submit Bugs

In beta Version software many common errors that the system will encounter in normal use will arise. Most bugs can be created more easily by the users who will behave like any other user on the system, since a developer cannot predict the behaviour of a user so bug information should be passed from the user to the developer. A simple solution is to have an interface that handles this process built into the system. Submit bugs form was develop for this purpose with security and ease of use in mind for both the user and administrator/developers.

The Carleton Branding main menu has a Submit Bugs link in the menu for when a user finds a bug he/she can quickly pass this information on to the administrator and developer using this form. The form is comprised of an email field and a comment field. The email is used for communication between administrator and user, and the comment field is used to describe how the bug was created.

Submit bugs form has been successfully implemented in the system using stmp and allows users to send feedback to the administrator without have to login to an email account or disclose the administrators email address to the user. The email address is hidden in the system and the Carleton University Carpool System will send the message to the administrators email address directly.

Much of the systems future designs and success will be in how well the system meets the needs of students and employees at Carleton University as an efficient Carpool System. The future users will benefit greatly by how quickly the system gets up to version release state and out of beta version.

Designs – Sequence Diagram

Figure 19 shown below, shows the sequence diagram for user to submit bugs to the administrator

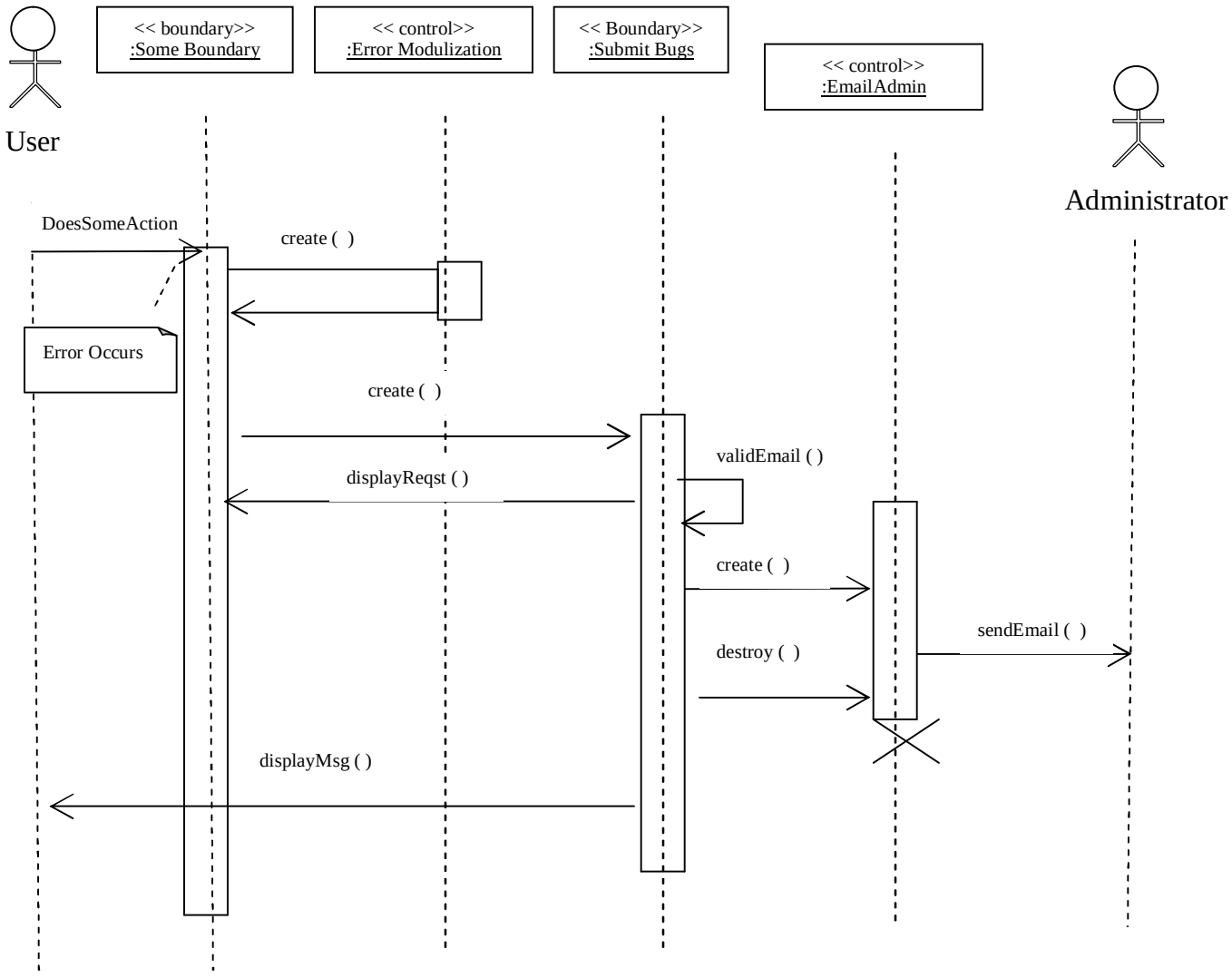


Figure 19

2.3.8 Modulize URLs

The Carleton University Carpool System uses two kinds of connection normal regular HTTP and SSL to communicate to a client's browser. SSL is Secure Socket Layer is used to encrypt communication of sensitive information from the client's machine to the server and from the server to the client. This two kinds of communication use different URL that includes either http or https and difference port values (8080 or 8443). These values are used throughout the Carleton University Carpool System and should be consistent in the system, through proper modulizing of the values.

The system stores three constant values: the regular address that uses http and port 8080, and SSL that uses https and port 8080 for secure connections and the web application's folder's name in case the web server serves multiple web applications at once. All links in the system referring to internal content have been created using these constants declared in the Carleton Branding java Class. The web application's folder name that is part of the modulizing of the URL allows the Carleton University Carpool System to run any directory and not just the ROOT directory of the server in the situation where the server has multiple web applications running.

The modulizing of URLs has been successfully implemented in for all links to use the constant values. The system can be quickly moved to a new address without changing many lines of code and is capable of working on a server with multiple web applications running.

The system is more portable with modulized URL's, because the system's URL's can easily be changed at one location. If the ports change or SSL is not be to used anymore then this can simply be implemented in the system because all the variables that refer to code that implements the ports and SSL are modulated in one location.

Design - Overview

Figure 20, 21 below show the design reason for different URLs

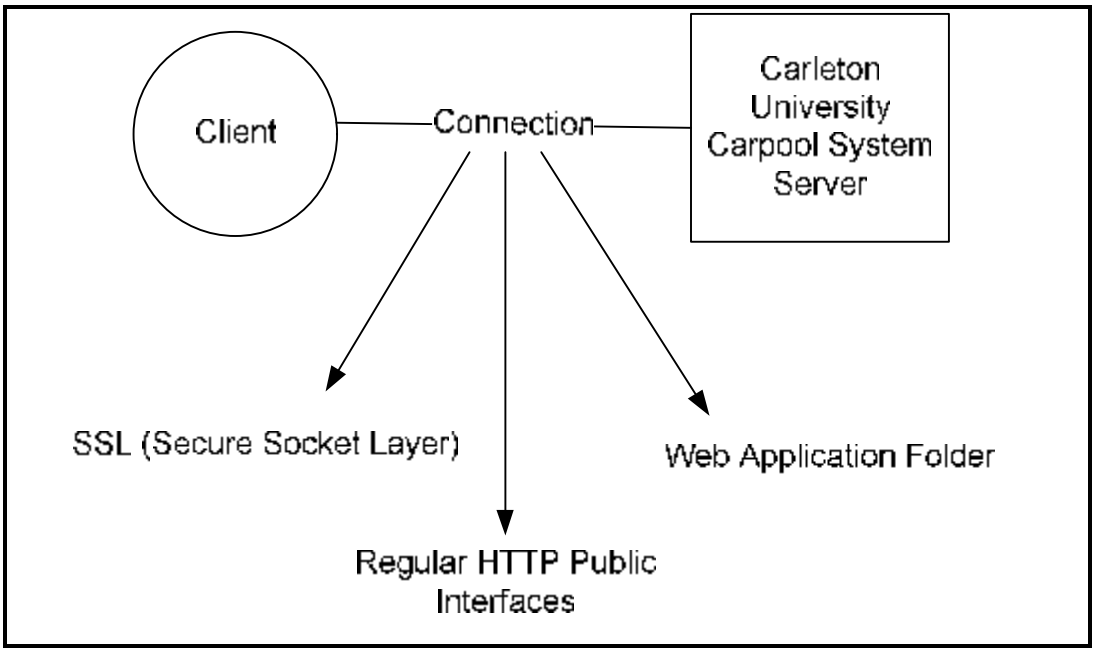


Figure 20

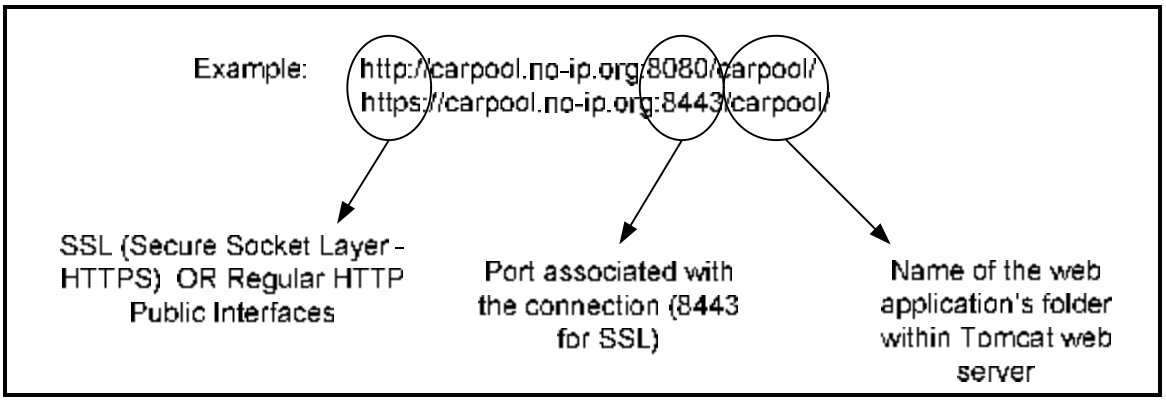


Figure 21

2.3.9 Search Options

In a Carpool System the user need to be able to find other user to carpool with to their destination, so there needs to exist some sort of function within the system that provides a feature to find other users. The Carleton University Carpool System fulfills this requirement through its searching feature that includes search options.

In the web application there are three kinds of searches: normal searches, trip searches and advance searches. Normal searches will search the database based on the main intersection criteria to the destination of Carleton University. This means that if another user has the same main intersection they will be apart of the search results. Trips searches just involve a certain city destination outside of Ottawa. All search results will be based on the city selected by the user. Advance searches will search the database based on the follow criteria and are used for the destination of Carleton University: main intersection, direction user wishes to go (one-way or both ways), postal code, departure times from home and Carleton with a range, gender preference, carpool preference (driver, passenger, drive or passenger) and car preferences. All three kinds of searches have there benefits the normal search will give the user the most hits. The advance search that uses a specific postal code will be the fastest processed search request, since the database's member information is separated by postal codes.

The search has a default search that searches intersections, but the design of the database is not used optimally unless the user uses the advance search options. Future plans will have a search mechanism that will not display any personally identifiable information (See the new feature section). This new design is described in the new feature section.

This feature meets system primary functional requirements as a system to be able to match users for a carpool. Users can are able to try to match themselves with other users in the system with the search function so that communication in starting a carpool can be arranged.

Designs – Diagram

Figure 22 Describes the Search Criteria and Current Search Filters. The client creates a search using search criteria and the system gets the results from the database.

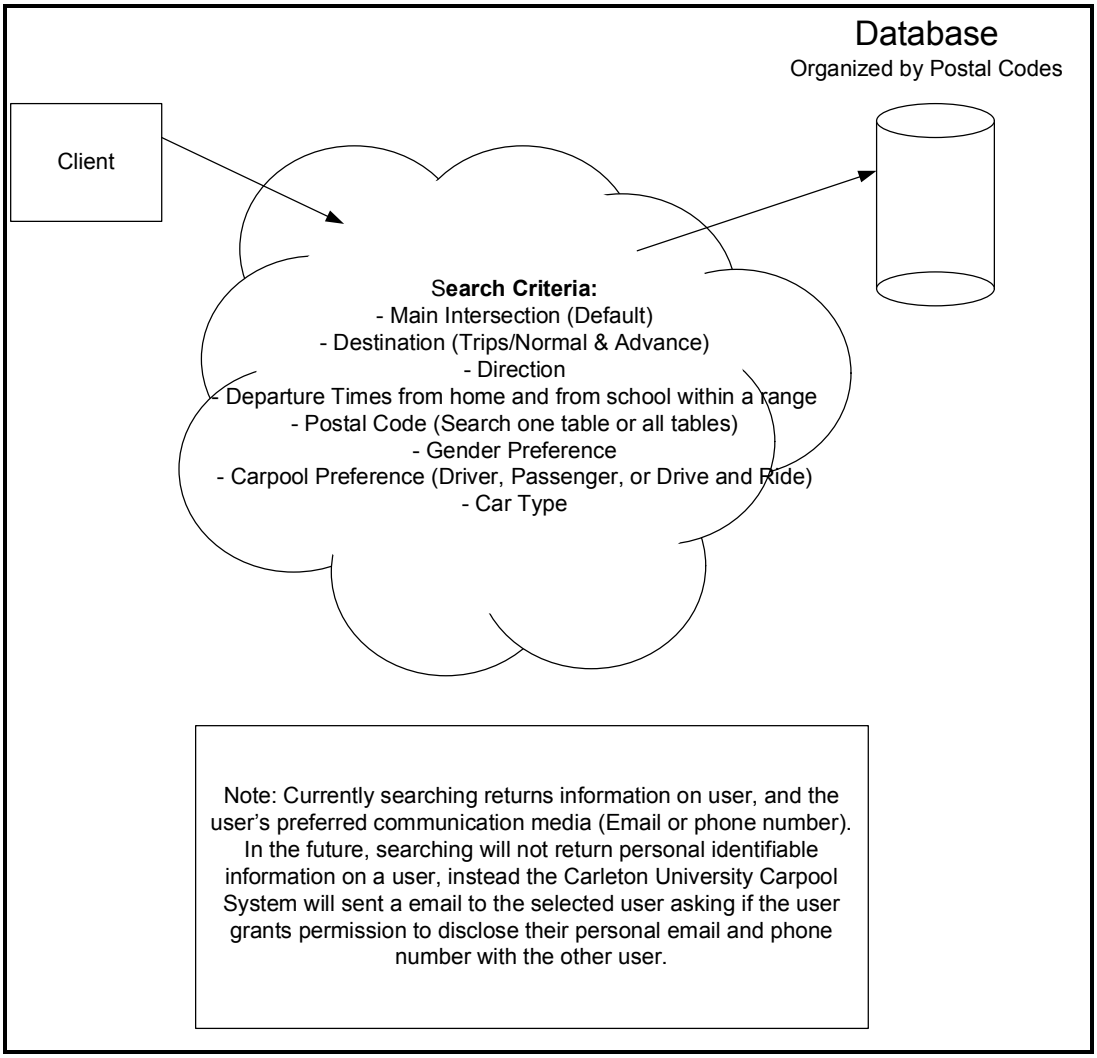


Figure 22

2.3.10 New Registration Fields Password Encryption

The method by which the Carleton University Carpool System matches uses for carpooling is based on information provided during the registration process. The registration information collected may have its requirements changed as the method of matching users change and as new functionalities are added to the system.

The Carleton University Carpool System registration process has been changed to adding the following fields: Student/Employee Number, Display Email, Display Phone Number, Display Name, Car Type, and number of passengers.

The new fields have been successfully implemented in the system. The information is used in the following new features: the trips to other cities, new search options, and for password encryption. Password Encryption has added another level of security to the system for the end-user.

The user can be confident that their personal password is protected at many levels from being stolen, even at the administrator level.

2.3.11 Membership Deletion

A user should feel that the Carleton University Carpool System gives the user opportunities as a member, but the user should feel in control of their personal information. If a user feels that they have complete control they can put more trust in the system. The trust allows the system to gain the confidence in the system to meet their needs.

When a user registers in the Carleton University Carpool System their information they enter is immediately stored in a MySQL database on the server. The information resides on the system until: the administrator deletes it, the administrator edits the information, the member edits the information or the user deletes his/her membership. Up until a user activates his/her account the only way for a user to delete or edit information if by emailing the administrator. After activating a membership the member has the option for self-deletion of their information from the system. Self-deletion is when the member initiates the deletion of their personal information be selecting membership deletion option in the member interface.

Member ship deletion has been successfully implemented into the Carleton University Carpool System. Once a member registers and activates his/her account they have complete control over deleting their information from the system.

A user control over their personal information registered in the database will build a users confidence in the system, which will improve a user's use of the system

2.3.12 Connection Pool

An important characteristic about a web server is that it is online all day all the time, in a state where it is ready to accept request and fulfill them. The Carleton University Carpool System requires the web server to be online and the database server (MySQL) online continuously. The system requires connections to the MySQL to fulfill request at anytime. These connections need to be maintained at all time for successful operation of the system.

MySQL database accepts connections from applications that have the correct authentication. When the Carleton University Carpool System goes online it starts a connection pool manager supplied by Tomcat web server to manage the database connections for the application. At all time a certain amount of connections are made available to the web application so continuous request can be fulfilled.

Connection Manager has been successfully implemented in all database request events. Random events that occur like connection time-outs are handled easily by the connection manager. Heavy loads will cause the connection manager to create more connects to handle the requests.

If the connection becomes stale, timeout, or is not available, the connection manager is capable of reacting to resolve the situation to maintain continuous connections for the web application.

2.3.13 Trip

Initially the carpool system provides carpooling service limited to Ottawa area to Carleton students and stuffs. However, there is a need to expand the service to other major cities. Therefore, this new features provide carpooling services which allows students to get together and travel to other cities.

Now the user has this feature. If the user is a driver and would like to share the ride with other students, all he/she has to do is to select the “add trip” option from the menu. Then submit the trip information to the system. Each user is allowed to add up to 3 trips to the system; and if he/she wants to add more trips to the system, then he/she must delete the already submitted trips. The reason to have limit on amount of trips to the user is that we do not want to user to flood the system with non-genuine trips. User has to be a driver in order to add a trip other wise, user will get an error message. A new table in the database is created in the MySQL in order to accommodate the new feature. This new table is used to store all the trips added by all drivers, each row in the table contain the trip information and the driver’s ID. For the “trip” table please refer to **figure 7** on page 11.

This feature has been successfully implemented and it is a available for registered members. This feature enables the system to serve the students who frequently travel to other cities. Since this system is capable of providing service within Ottawa area, it is also capable to provide service to other cities. This system is more useful, versatile with this new feature.

State Chart Diagram:

Figure 23 is the state chart diagram for member to add trip. It shows the states a member can go through when the member need to add a trip to the system.

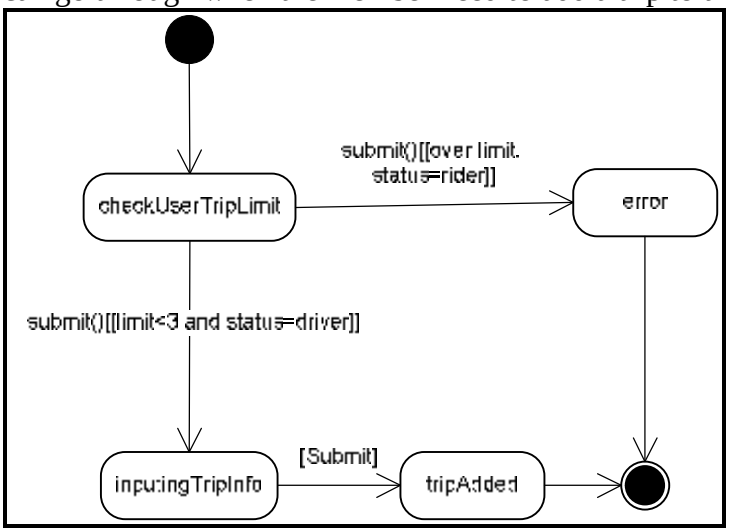


Figure 23

Scenarios:

1. user click on add trip bottom
2. user inputs trip information
3. user submit trip to the system
4. user receive a message that system t has added a new trip to the system

2.3.14 Registration

This feature allows the students or staff to register in the system, so then they can enjoy the service for free. This feature was already implemented by the previous team, however, it is doesn't ask the user enough information to allow the system to protect registered member's privacy. Therefore, we added more question fields in registration form. So then the user has more control on what kind of personal information they would reveal to other members. In addition to that, the revised registration form also ask more personal information of people who try to register such as what gender they prefer to share the rides with and what type of cars they drive.

Any legitimate staff and students are welcome to registration. To do that, user will input their personal information, such as address, contact information, preference. In this revised version of registration form, user has the opportunity choose what personal contact information they would like to reveal to other members. On top of that, users also have more freedom to choose the people they want to share rides with for example user can now choose to share ride with male or just female or both. When they have filled in the registration form user will need to submit the information to the system. If there is no error in the information provided by the system a new message will display and notice the user that they have successfully register an account. An email will be sent to the new member requesting the new member to active their account. New member must activate their accounts after they registered in order to use the carpool system. To active the user account, user need to check his/her email and follow the instruction emailed to the new him/her. The activate account feature allows system to determine legitimacy of new member's e-mail.

If the registration was unsuccessful, an error page will be displayed on the user's computer and the system would indicated what information was entered incorrectly. If user is free to resubmit their information. The registration is implementing if an error shows up.

This addition question fields have been addition to the registration form. Everything was completed as planned for this feature. It is available for the user.

This feature enables the system to protect member's personal information against spammer, give freedom to the members on what personal information they like to reveal to other member.

State Chart Diagram:

Figure 24 show the process for user to register an account.

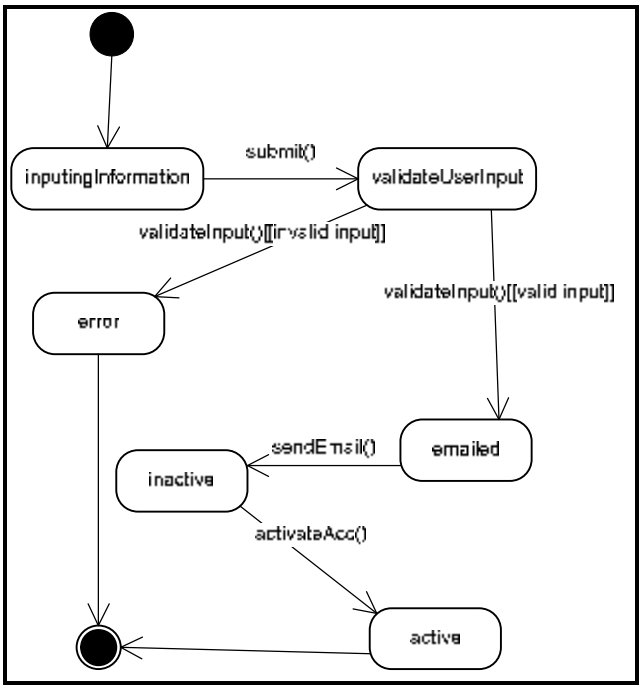


Figure 24

Scenarios:

1. user click on registration bottom
2. user input personal information, preference.
3. user submit registration form

4. system accepted registration request, and sent email to new member
5. new member activate new account
6. new member's account is active

2.3.15 Additions to the database.

We had to revise the database to accommodate the revision of our new design for the system since the previous design of the database could not provide enough storage to the new design. The new implementation is focused on the design in the database structure no the software we use, therefore we are still using the same software as the previous team, the software we use is MySQL.

The change in the database includes additional tables, and new columns to the already exist tables. There new columns added to the “member” table. And the new table created are “cities”, “trips”. The “cities” table are created to store cities so then the member can search rides available to those cities. When a driver want to “add trip”, she/he can only pick the cities they want from the list store in the “cities” table. The cities can be input to the MySQL manually, and if the cities are not available to choose in the “search” option or not available in the “add trip” option, members can email administrator and request to make the city available to the member.

All the require addition change in the database has been implemented; they are the available to the system to use.

The addition tables in the database help the system to provide new features to the member that was not available before. Moreover, the members can enjoy more service, and travel places outside of Ottawa. Also it also helps the system to build features that protect user’s privacy.

Database diagram:

Below is **Figure 25**, it shows the new relational database design for the new system.

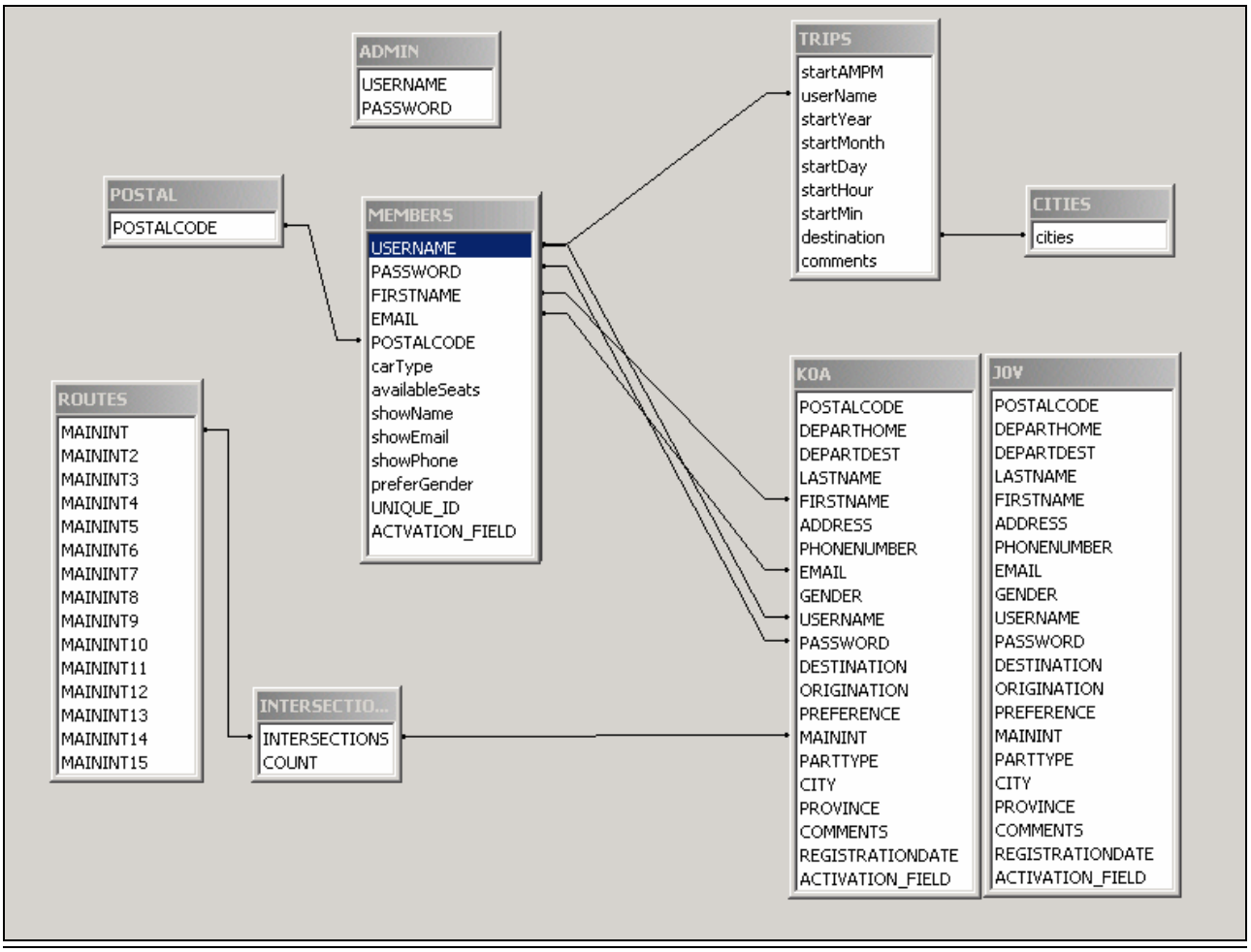


Figure 25

- Table members stores member's account information, contact information, preference(gender, privacy level)
- Table admin contain administrator's account information only. Administrator does not have the search option.
- That's administrator table has no connection to other tables.
- Table postal contains the postal codes available in the Ottawa area.
- Table cities contains the cities available to member to search or add trip to.
- Table routes contains all the major streets in the Ottawa area
- Table intersections contains all the major intersections available in Ottawa area
- Table trips contains trips information added by member.

2.3.16 Request for Other Intersections.

Intersections are the street intersections in the Ottawa area. When the user register, it is required for he/she to choose an intersections that is closest to his/her home. Although the intersections in our database cover most of the major intersects however, it is not 100% covered. Therefore when a user happen to live near an intersection that is not list in the database, the user has to choose another closest intersection near to his/her home. To tackle this problem, we make an request form for the user to request an intersection to be added to the system by email.

When the user register an Carleton University Carpool System account and can't find any intersection close to where they live, she/he can choose the "request intersection" form inside the "registration" form. When the "request intersection" form opened up, user will be prompted to enter his/her desired intersect and submit it to the system. The system validate the user's input, and then sends an email to inform the administrator that a user need a new intersection to be added to the system, so the she/he can choose it as the intersection closest to where she/he lives.

This feature has been implemented and it is available to users.

With this in the system, user is now able to request for intersection that was not available in the system. It helps the system to expand its collection of intersection.

State Chart Diagram:

Figure26 is the state chart diagram for a user to register and request for new intersection to be added to the system. It shows the process for a member to request new intersection during registration.

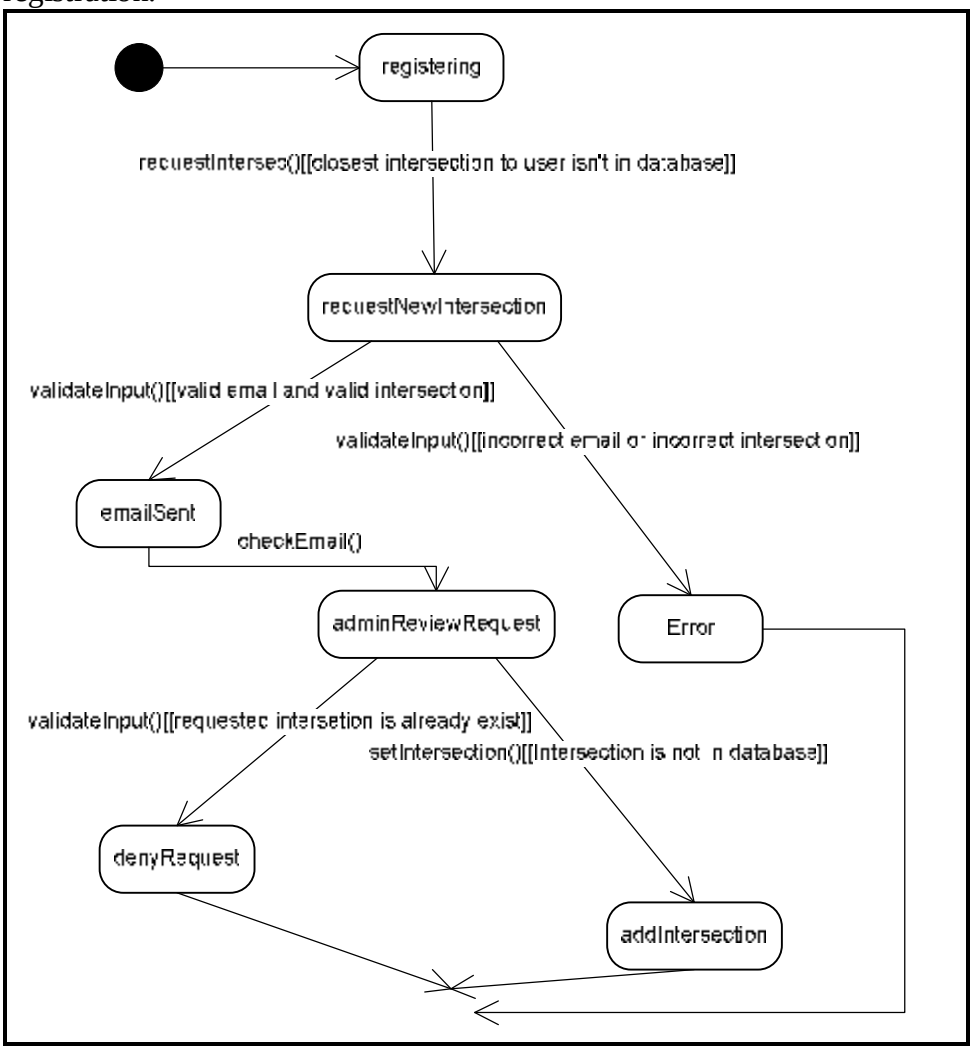


Figure 26

3.0 Future improvement

3.1.1 Private Message Member

When a member searches for rides, it is often that search results disclose other **searched members**' personal information. To protect **searched members**' personal information, the new suggested improvement will hides the information until **searched members'** agree to show their personal information to the searcher.

This improvement will allow the searchers to see available rides in the search result; however, the **searched members**' information is hidden. The searchers are provided a function called "**Private Message Member**", and then searchers can use this function to emails the **searched members**; after the **searched members** receive the emails from searchers, the **searched member** has the choice to allow searcher to see contact information, or they can communicate directly using their preferred method: email, phone, and face-face. The suggested improvement provided by Carpool System allows searchers to email **searched member** without disclosing **searched member's** email address.

- Searcher searches for rides.
- Searcher receives result with hidden contact information.
- Searcher contact Searched Member by using "**Private Messaging Member**" function.
- Searched Member received email about new Private Message available.
- Searched Member sends contact information to searcher by PM (private message).
- Both searcher and searched member will exchange arrange for the ride.

Figure 27 shows the collaboration during searcher search for ride, and how searcher contact driver using private message system. Using this private message system, there is no personal information disclosed.

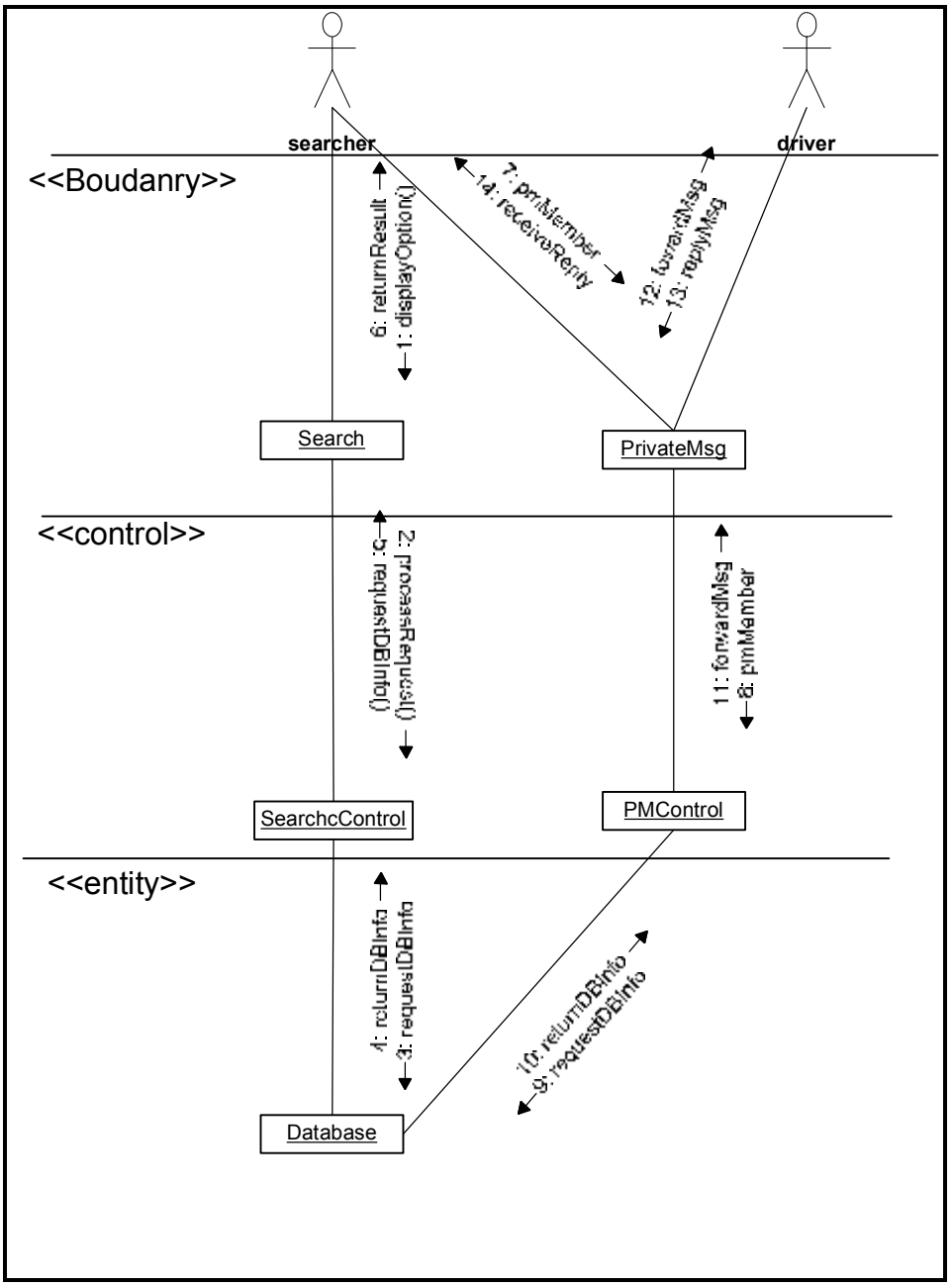


Figure 27

Adding this feature can further strengthen the security in the system and provides better privacy protection to the members' personal information. Also, this system provides an easy solution for anonymous message protocol so then members can exchange messages without disclosing their contact information. This feature could make the Carpool system's most professional compare to other corporation's project.

4.0 Conclusion / Discussion

The Carleton University Carpool System before starting required changes to make the application usable for daily use and further develop of its features all create a more complete application to web application standards of today. We proceeded to setting up the original application on our own server, debugging it to work properly and then doing an analysis on the systems additional requirements. We defined new features and reorganize the application to an ideal web application structure for the tomcat server. The new features including modulizing the system into a more object-orientated system. This approach allowed upgrades, maintaining consistency and debugging easier. We created excellent documents about the application, so that future designs could be implemented quickly and easily (See Appendix B, C, and D). Other documents include maintenance such as backup/restore implementation (See Appendix E). Finally, an overview of the system security is documented in Appendix F.

The system currently requires one more feature to be added before the web application can be made an official Carleton University Carpool System. We have detailed out the designs and requirements of the new features. One thing that could be done more is testing. Testing using JUnit would allow the application to show in a more documented manor how each class handles input. By having already implemented a beta version with quick online feedback capabilities on system, bugs can be quickly tracked and handled once the system is up and officially running.

5.0 References

(1) Google, “Google Search Results,” March 2005, http://www.google.ca/search?hl=en&rls=GGLD%2CGGLD%3A2004-37%2CGGLD%3Aen&q=%22Email+activation%22&meta=.

(2) Monster.Ca, “The popular job site uses a forgotten password system,” March 2005, http://www.monster.ca.

(3) Java, “Java Technology and Java References,” March 2005, http://java.sun.com/j2se/1.4.2/download.html (Java SDK).

(4) The Apache Tomcat Jakarta Project, “Tomcat Documentation,” March 2005, http://jakarta.apache.org (Jakarta tomcat).

(5) MySQL, “Documentation and Resources,” March 2005, http://dev.mysql.com/ (MySQL).

(6) Carleton University, “System and Computer Engineering Department,” March 2005, http://www.sce.carleton.ca/courses/sysc-4907/index.html.

(7) Rober Holwell, Linton DonBosco, “Carleton University Carpool System”. Apr. 4, 2003.

6.0 Appendices

Appendix A – User Manual

User Manual
Carleton University Carpool System (CUCS)

Table of Contents

1.0 Introduction

Carleton University Carpool System is a web application, designed to act as the medium that will allow Carleton University Students and faculty to Carpool to and from campus and other cities. The goal is to reduce traffic, pollution, save money and improve student social networking. The carpool system works by allow registered users to search for users that meet his/her carpool needs.

This document is the User Manual for the web application of the Carleton University Carpool System. The following information describes how users and administrator can interact with the system.

2.0 Members' Manual

These are a list of actions that a user can perform in interacting with the web application online. Details on each action are given and screenshots are used to visually describe the actions performed.

2.1 Registering

Before a user can access the system he/she must setup an account by registering to the system and activating their account. The registration process includes the following actions:

- Navigate to the public registration form (See Figure 1)
- Enter and Submit the user's personal information in the registration form (See Figure 2)

- Finally, activate the user's account through the activation email sent to the email used during registration. (See Figure 3)

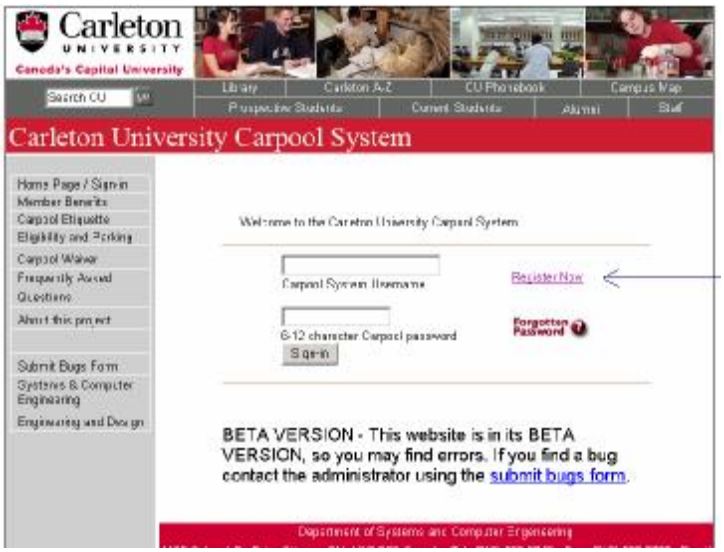


Figure 1 - Navigate to the Public Registration Form

Figure 2 - Enter and Submit the User's Personal Information in the Registration Form

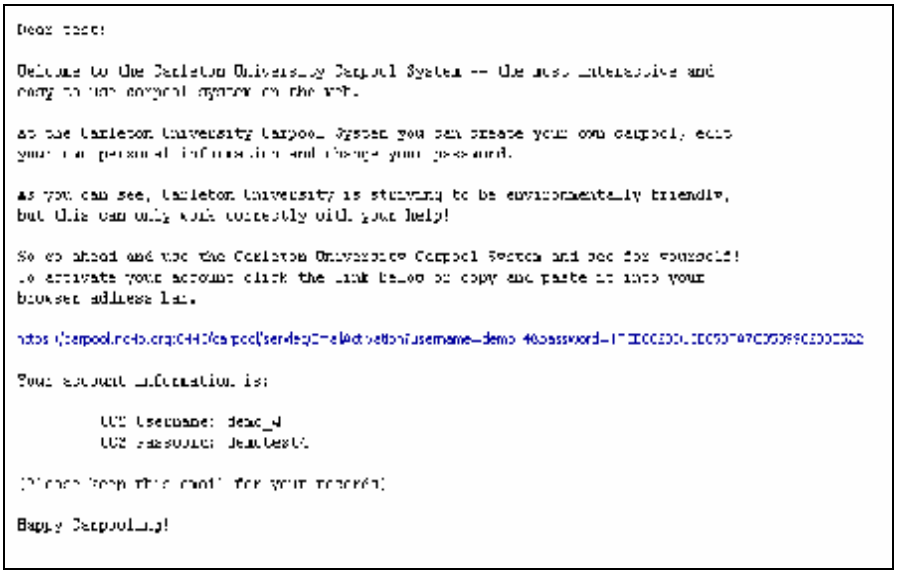


Figure 3 - Activate user's account through the activation email sent to the user.

2.2 Forgot Password

Passwords that have been forgotten can be recovered through the "Forgot Password" option of the web application. In this option the user submits his/her email that was used in registering to the system. The system will then send the user an email asking for the user to click a link in the email to change the password to the user's postal code. The "Forgot Password" process includes the following actions:

- Navigate to the Forgot Password Interface (See Figure 4)
- Enter and Submit the User's Email used in Registering to the System (See Figure 5)
- Click the Reset Password Link in an Email Send by the System (See Figure 6)

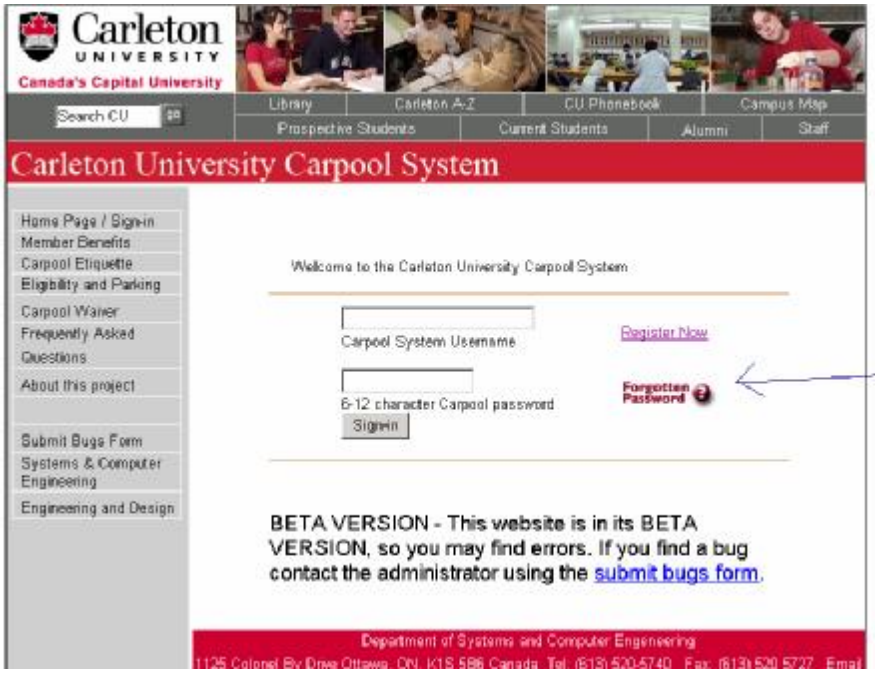


Figure 4 - Navigate to the Forgot Password Interface

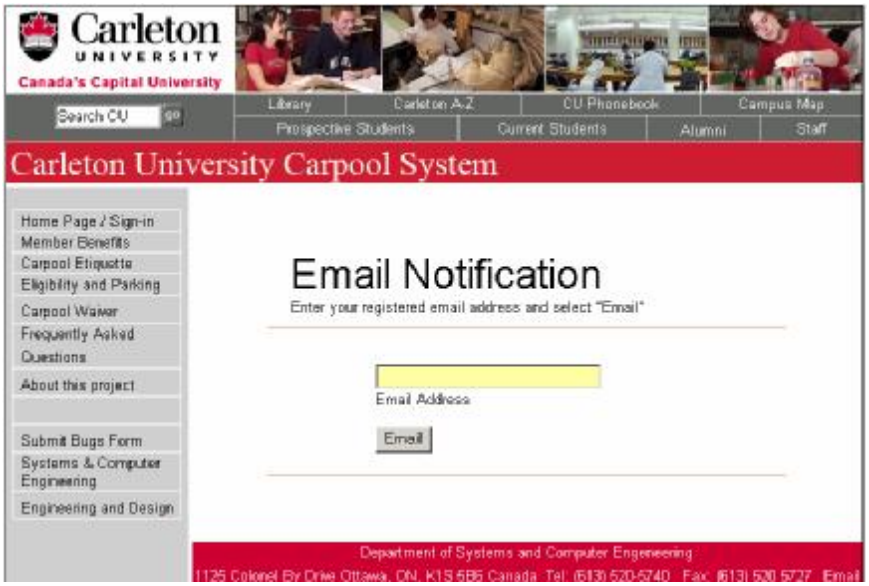
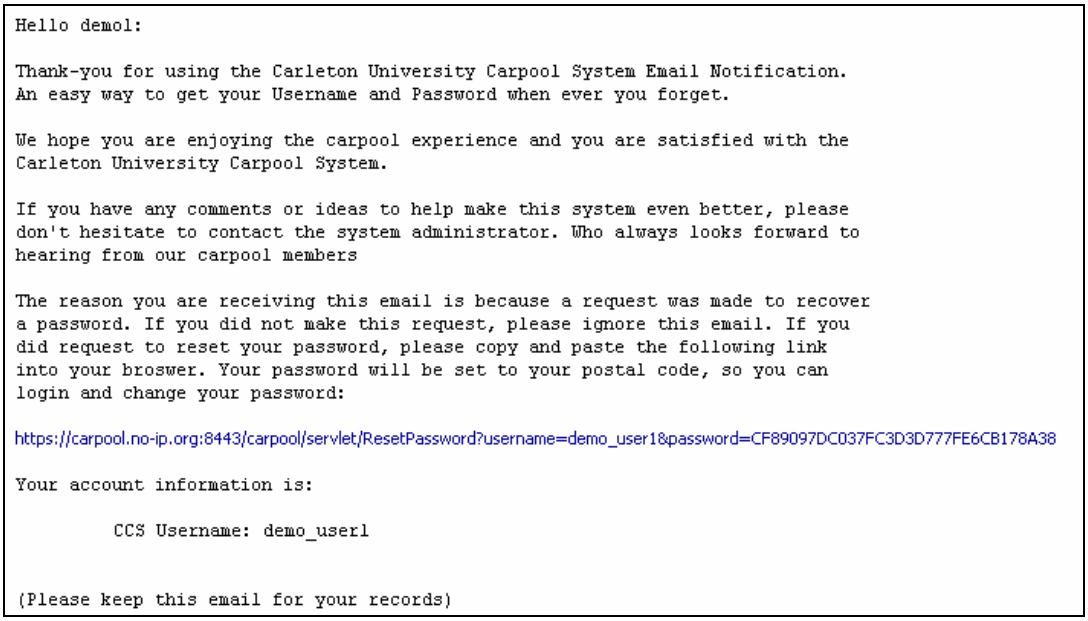


Figure 5 - Enter and Submit the User's Email used in Registering to the System



Hello demo1:

Thank-you for using the Carleton University Carpool System Email Notification.
An easy way to get your Username and Password when ever you forget.

We hope you are enjoying the carpool experience and you are satisfied with the
Carleton University Carpool System.

If you have any comments or ideas to help make this system even better, please
don't hesitate to contact the system administrator. Who always looks forward to
hearing from our carpool members

The reason you are receiving this email is because a request was made to recover
a password. If you did not make this request, please ignore this email. If you
did request to reset your password, please copy and paste the following link
into your broswer. Your password will be set to your postal code, so you can
login and change your password:

https://carpool.no-ip.org:8443/carpool/servlet/ResetPassword?username=demo_user1&password=CF89097DC037FC3D3D777FE6CB178A38

Your account information is:

CCS Username: demo_user1

(Please keep this email for your records)

Figure 6 - Click the Reset Password Link in an Email Send by the System

2.3 Sign-in

A member accesses the system by signing into the system, through a login form. The user is required to enter in his/her username and password selected during registration. A user who has not yet activated his/her account will to be able to successfully login to the system. The “Sign-in” process includes the following actions:

- Navigate to the “Sign-in” Interface (See Figure 7)
- Enter and Submit the Member’s Username and Password (See Figure 8)

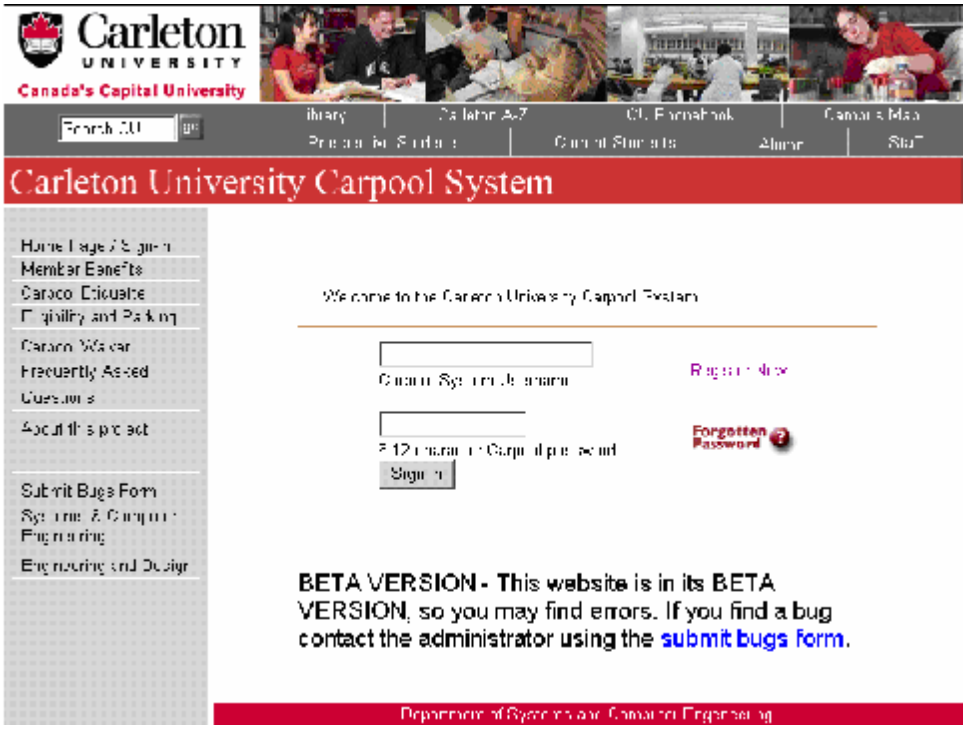


Figure 7 - Navigate to the Sign-in Interface

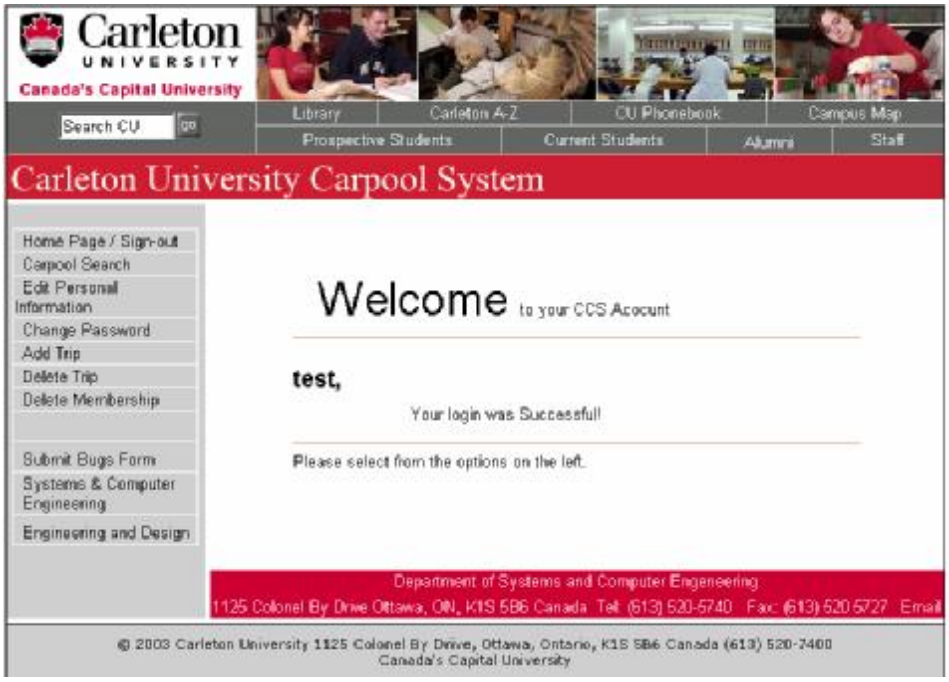


Figure 8 - Enter and Submit the Member's Username and Password.

2.4 Search

A member can start Carpooling with other user by first finding other user's that match his/her carpooling needs. In the member's interface he/she has the option to search and selecting searching options that they desire. The search process currently returns personal contact information of other user's. In the future this will be changed to have the search only return the basic carpooling information (times, destination, general location…) and then the system would allow the user who searches to have the system email a request to releasing a user's contact information (Email address and/or phone number). The current search process includes the following actions:

- User has already signed in (see Sign-in)
- Navigate to the "Search" menu option (See Figure 9)
- Select search criteria and submit the information (See Figure 10)
- View the search results (See Figure 11)

Figure 9 - Navigate to the "Search" menu option

Figure 10 - Select Search Criteria and Submit the Information

Figure 11 - View the Search Results

2.5 Add/Delete Trips

When a user registers he/she enters information pertaining to their desired carpooling needs to Carleton University, but if a user wants create carpool to other destinations (Toronto, Montreal, Hamilton…) he/she will add a trip to the system. Trips are carpools to certified destinations in the system. A user can add up to three trips in the system and then must delete old trips before adding any new trips. The current add/delete trip process includes the following actions:

Add Trips

- User has already signed in (see Sign-in)
- Navigate to the “Add Trip” Interface (See Figure 12)

- Enter and Submit Trip information (See Figure 13)

Delete Trips

- User has already signed in (see Sign-in)
- Navigate to the “Delete Trip” Interface (See Figure 14)
- Select and Submit a Trip to Delete (See Figure 15)

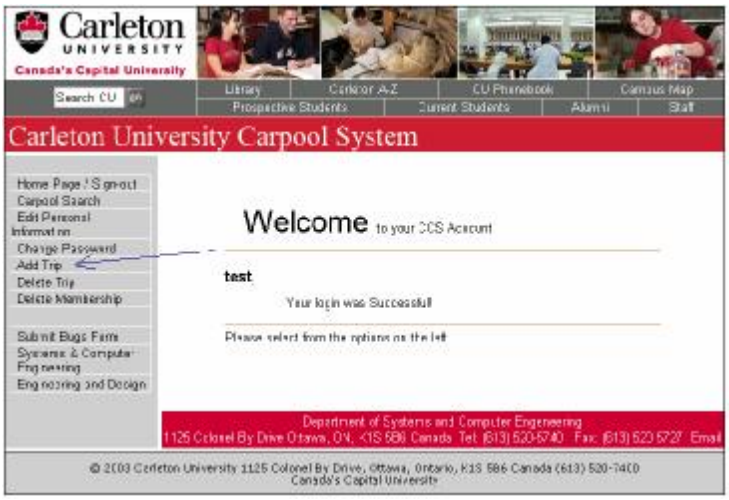


Figure 12 - Navigate to the “Add Trip” Interface

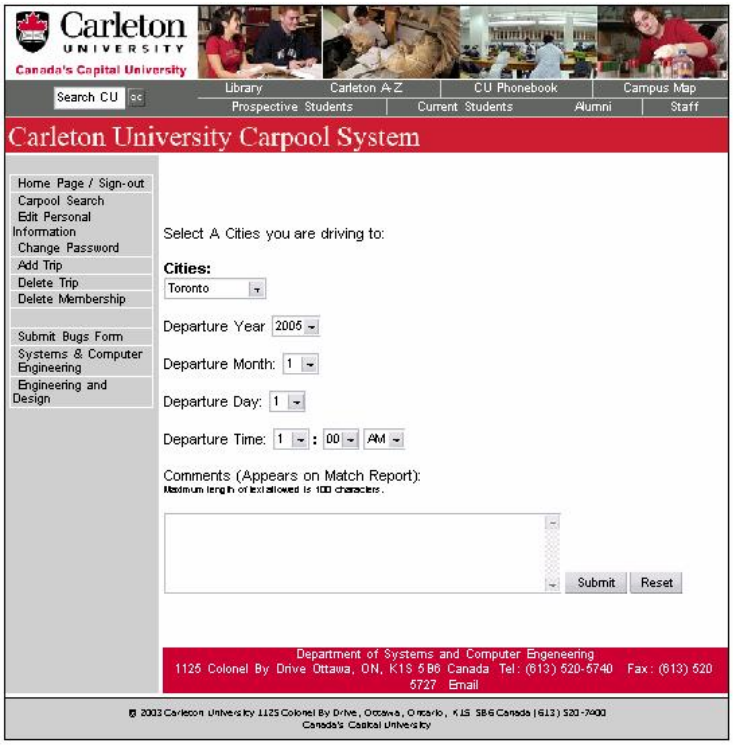


Figure 13 - Enter and Submit Trip information

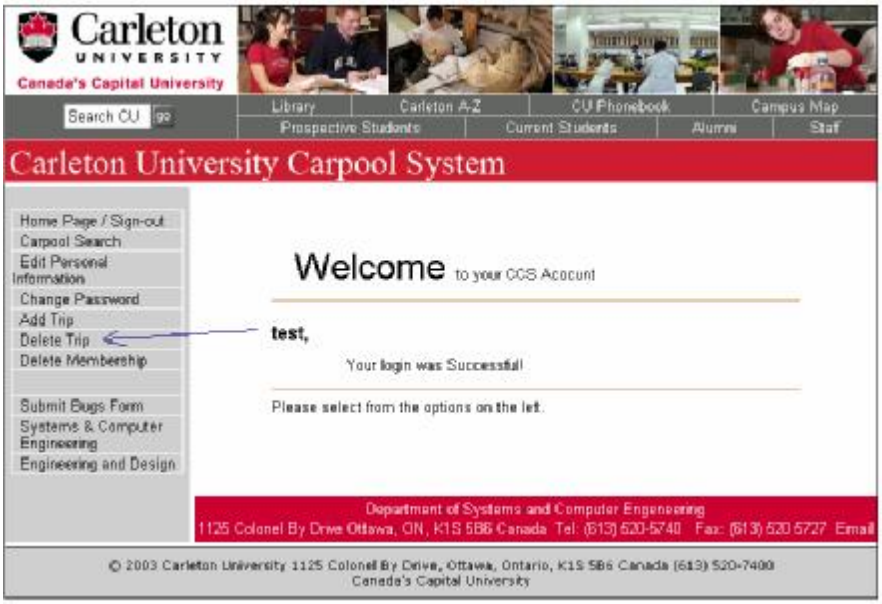


Figure 14 - Navigate to the “Delete Trip” Interface

Figure 15 - Select and Submit a Trip to Delete

2.6 Edit Personal Information

A member can change his/her personal information stored within the Carpool System Database, through the member's menu. The member cannot change his/her password or username through this interface (password changed through a separate interface). The current process to edit personal information includes the following actions:

- User has already signed in (see Sign-in)
- Navigate to the "Edit Personal Information" Interface (See Figure 16)
- Edit and Submit Personal Information (See Figure 17)

Figure 16 - Navigate to the “Edit Personal Information” Interface

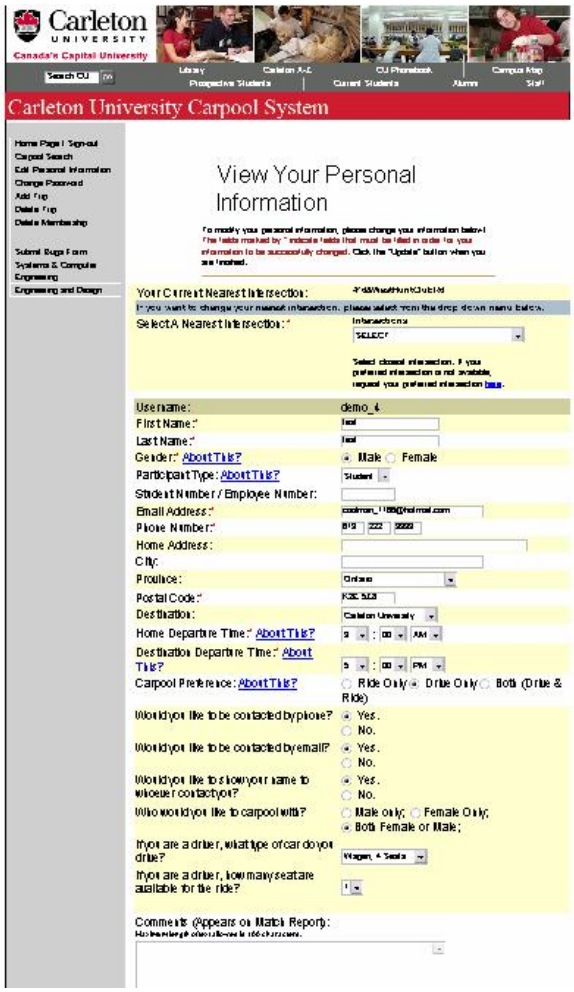


Figure 17 - Edit and Submit Personal Information

2.7 Change Password

A member can change his/her password through the change password interface. The member is required to know his/her original password and must be currently logged into the system. The current change password process includes the following actions:

- User has already signed in (see Sign-in)

- Navigate to the “Change Password” Interface (See Figure 18)
- Enter Current Password and New Desired Password (See Figure 19)

Figure 18 - Navigate to the “Change Password” Interface

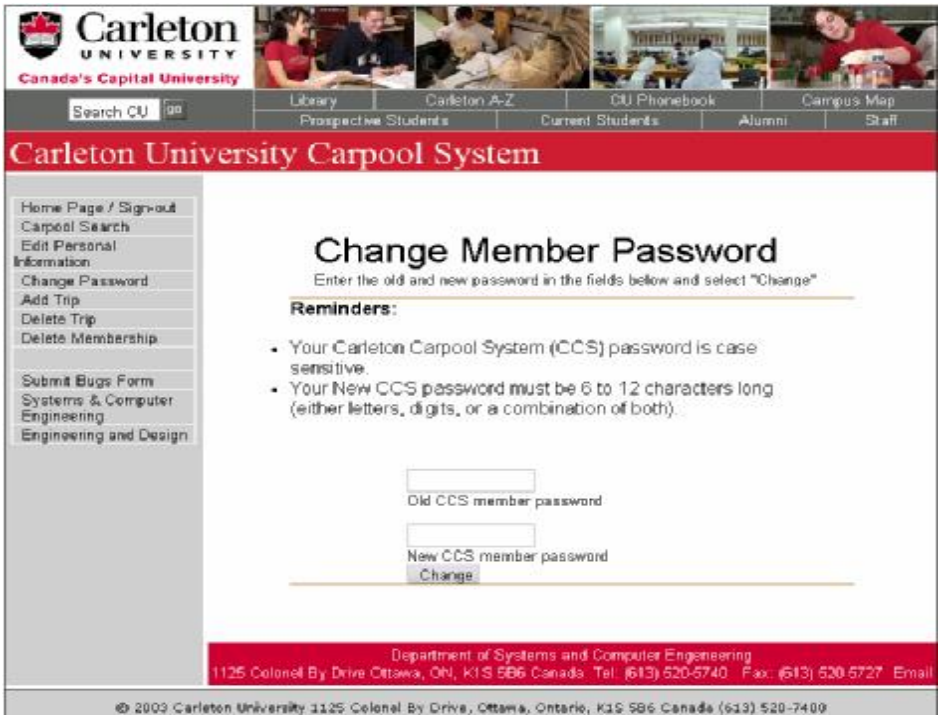


Figure 19 - Enter Current Password and New Desired Password

2.8 Delete Membership

A member can delete his/her account to the Carpool System by selecting the “Delete Membership” option. All of the user’s information will be removed from the system. The current Delete Membership process includes the following actions:

- User has already signed in (see Sign-in)
- Navigate to the “Delete Membership” Interface (See Figure 20)
- Submit delete membership form (See Figure 21)

Figure 20 - Navigate to the “Delete Membership” Interface

Figure 21 - Submit Delete Membership Form

3.0 Administrator's Manual

3.1 Add/Delete Intersection

Administrator has the permission to modify the intersection such as add intersection and delete intersection. To do so, the administrator need to login to the Carpool System and then:

1) Select the "Add Intersection" on the menu, see Figure 22
2) On the next screen, type the desired intersections to the provided space, and then submit. see figure 23

Administrator can also delete any intersection. To do so follow steps below:

1) Select the “delete Intersection” on the menu, see Figure 22

2) On the next screen, select the desired intersection to delete and then submit. See figure 24

Figure 22

Figure 23

Figure 24

3.2 Add/Delete Memberships

Administrator has the ability to add account. To do so, follow the steps below:

1) Select “Add Member” on the menu on the left. See **figure 25.**
2) Fill out the registration form and then submit. See **Figure 26.**

Figure 25

Figure 26

Administrator can also delete member account; here are the step to do that:

1) Select the “Delete Member” on the menu on the left. See **Figure 27.**

2) Select the account that you like to delete and then submit. See **Figure 28.**

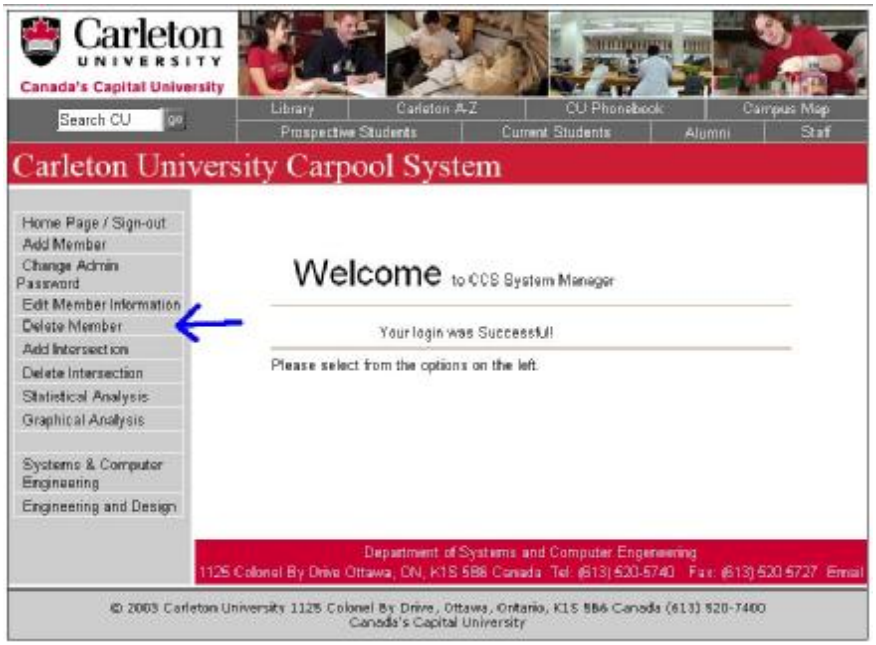


Figure 27.

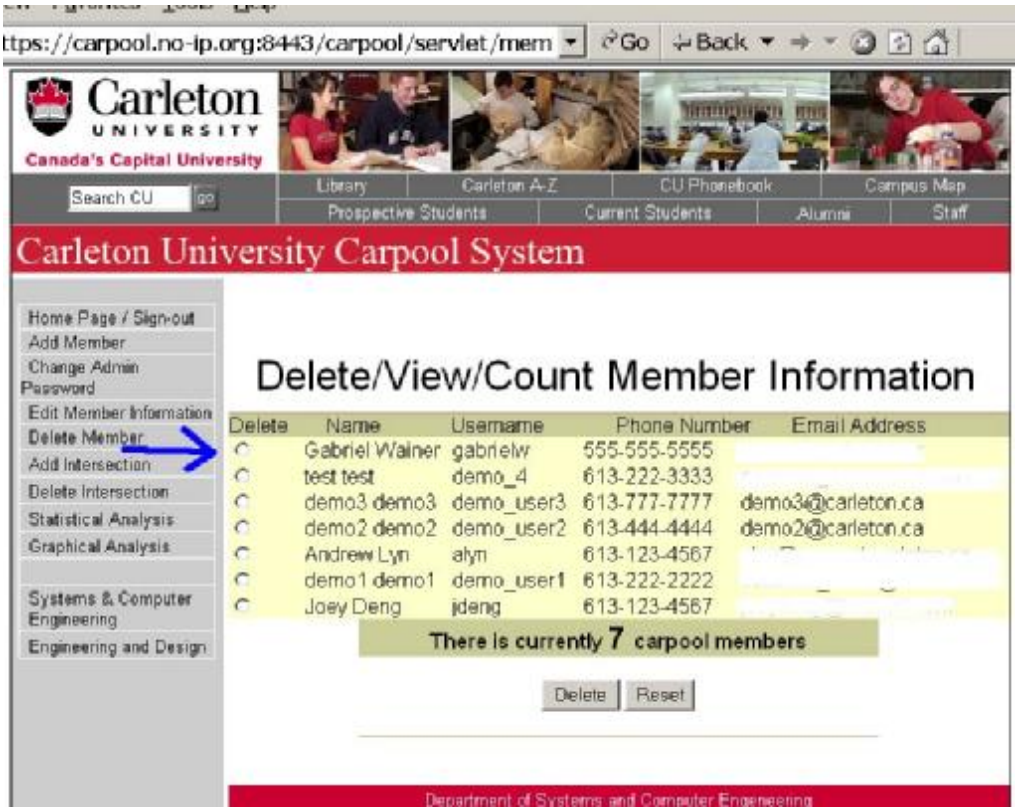


Figure 28.

3.3 Deactivate Accounts

If administrator needs to suspend a member's account, the administrator can certainly do so by following steps:

1) Select "Edit Member Information" on the menu, see **Figure 29.**
2) Choose the member account and submit, see **Figure 30.**
3) On the next screen, when the user's information is displayed, choose the "no" for user's "Activate Account" (as show by the blue arrow). See **Figure 31.**

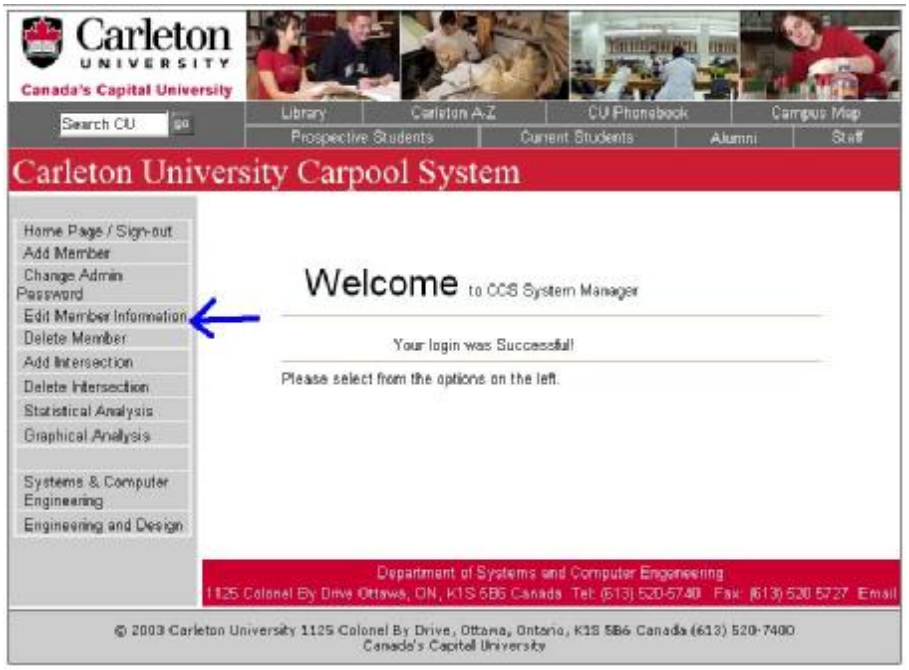


Figure 29

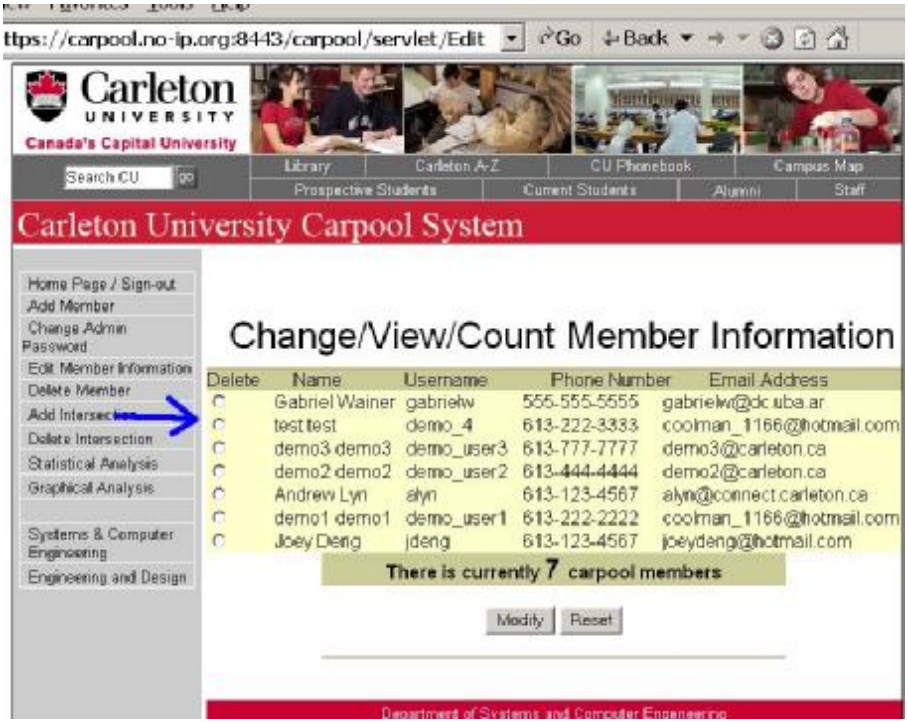


Figure 30

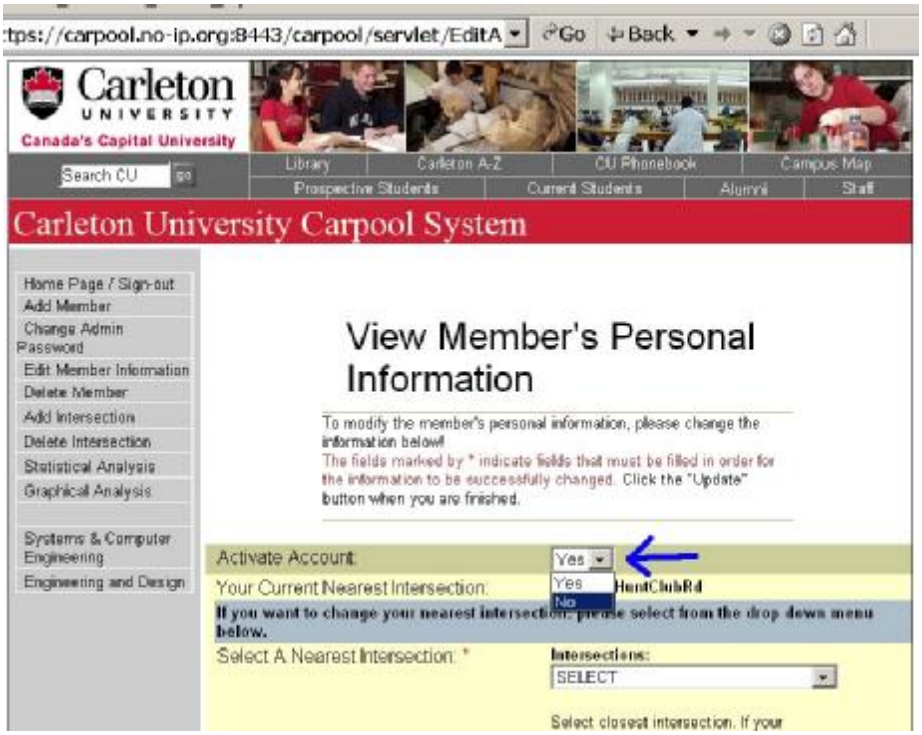


Figure 31

3.4 Change Password

Administrator can change its account password. Here are the steps to do that:

1) Choose “change Admin Password” on the menu, see **Figure 32.**
2) On the next screen, type in the old pass word, and the new password into the space provided respectively. See **Figure 33.**

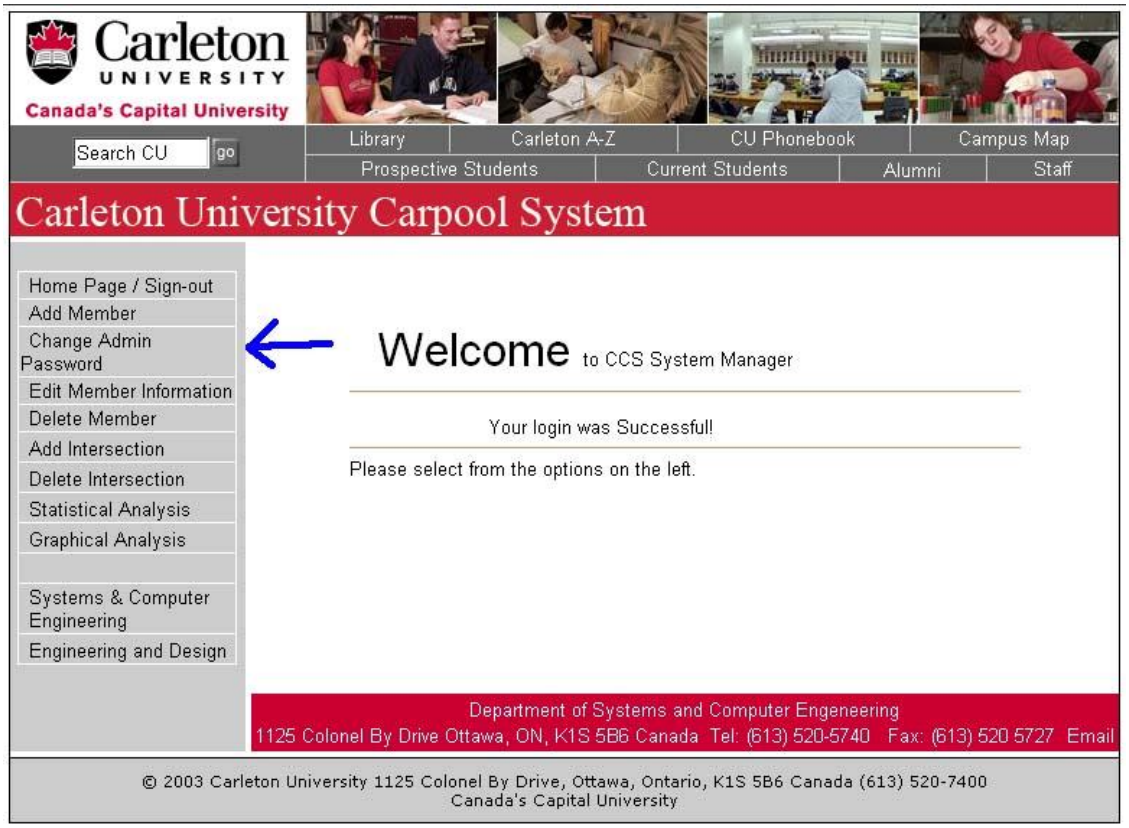


Figure 32

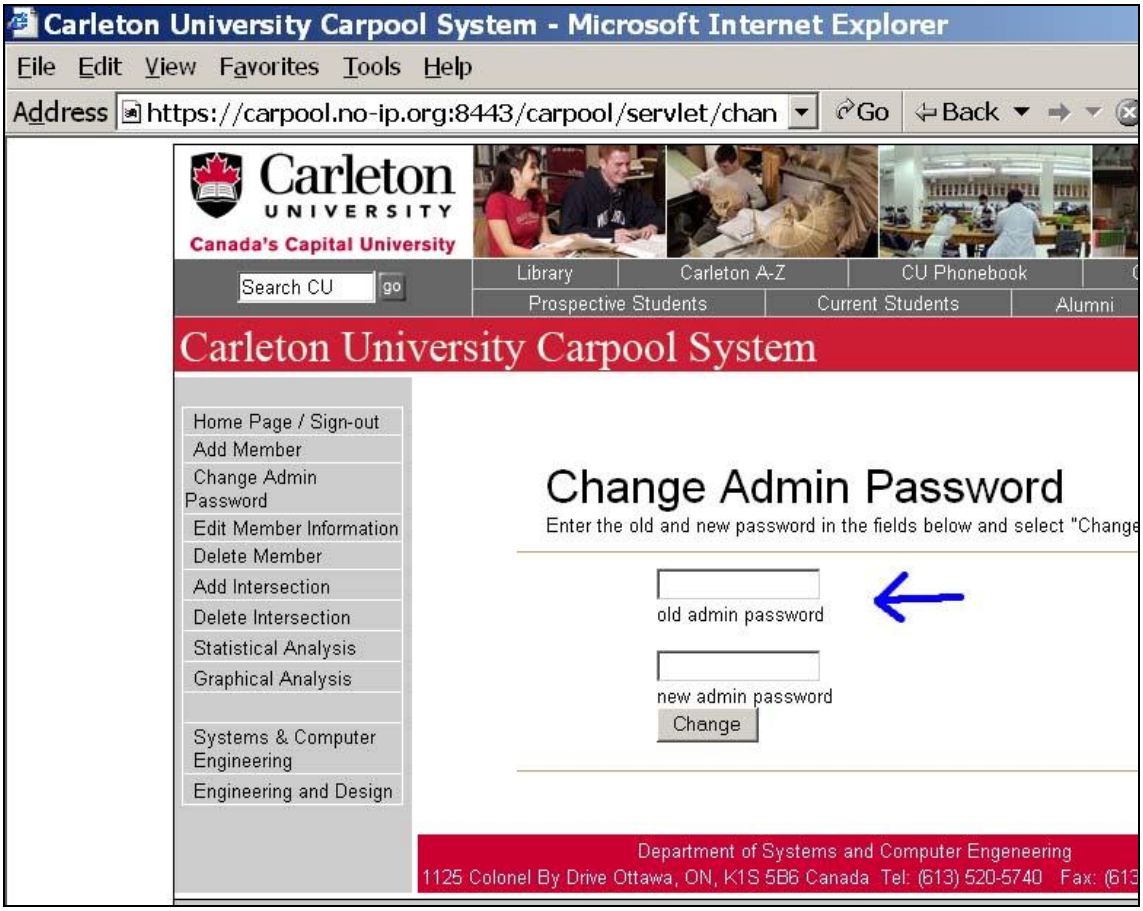


Figure 33.

3.5 Edit Member Information

Administrator has the ability to modify any members given personal information. Here are the steps:

1) Select “Edit Member Information” on the menu. See **Figure 34.**
2) On the next screen, choose the member account and then click Modify. See **Fgure 35**
3) On the next screen, user’s information is displayed on the screen, administrator can modify the user information with the new valid information. See **Figure 36.**

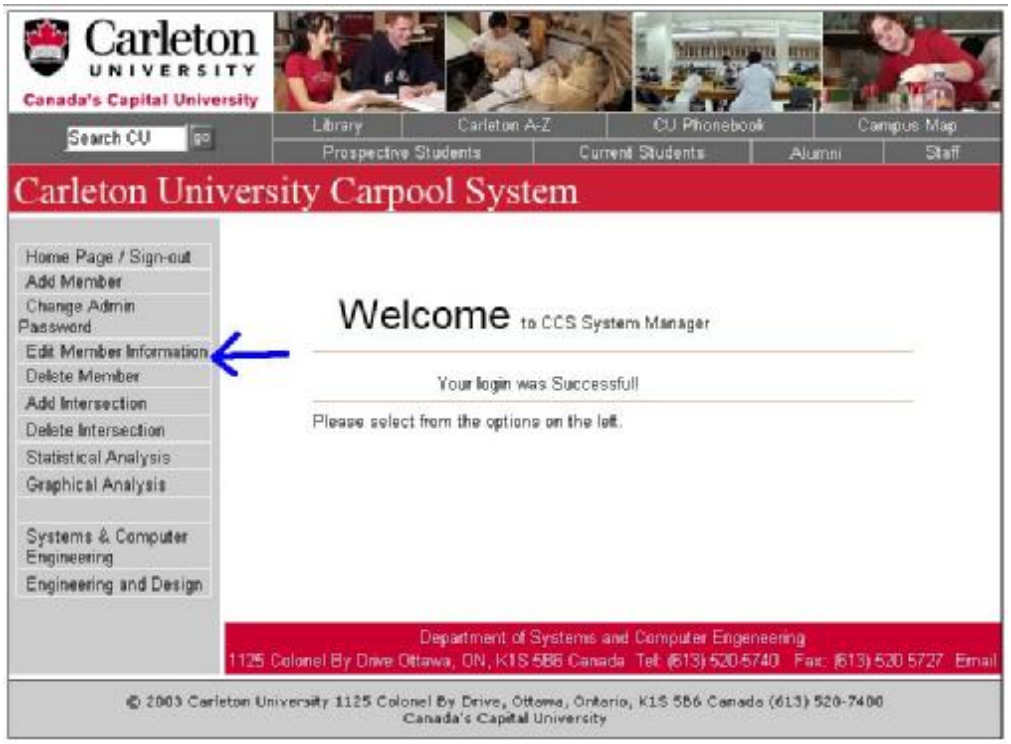


Figure 34

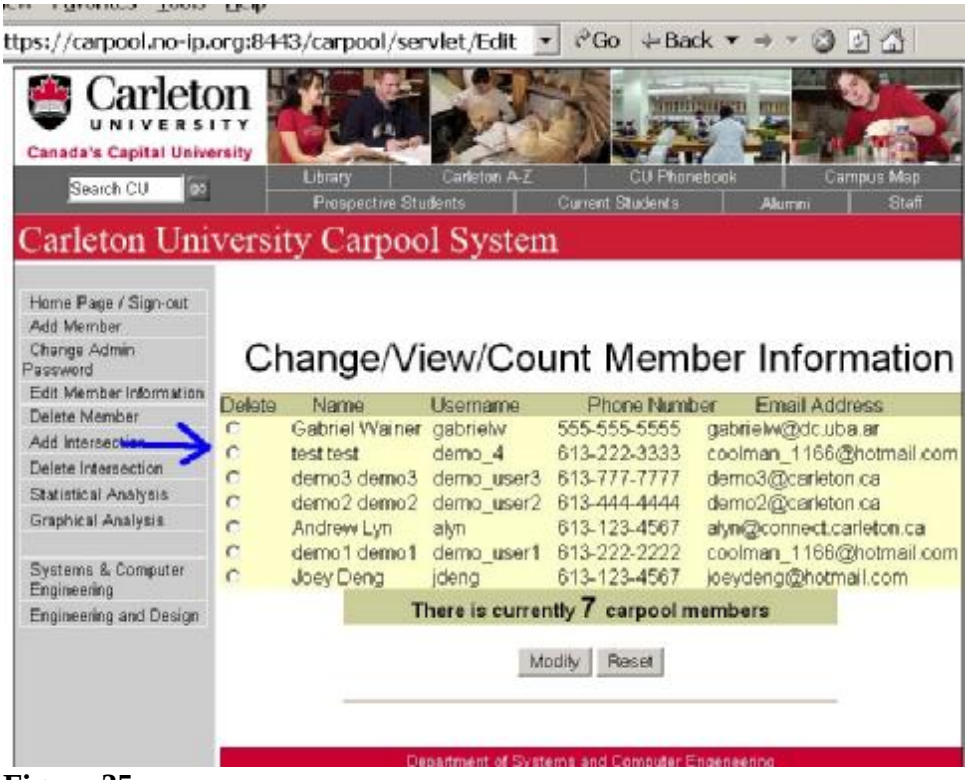


Figure 35

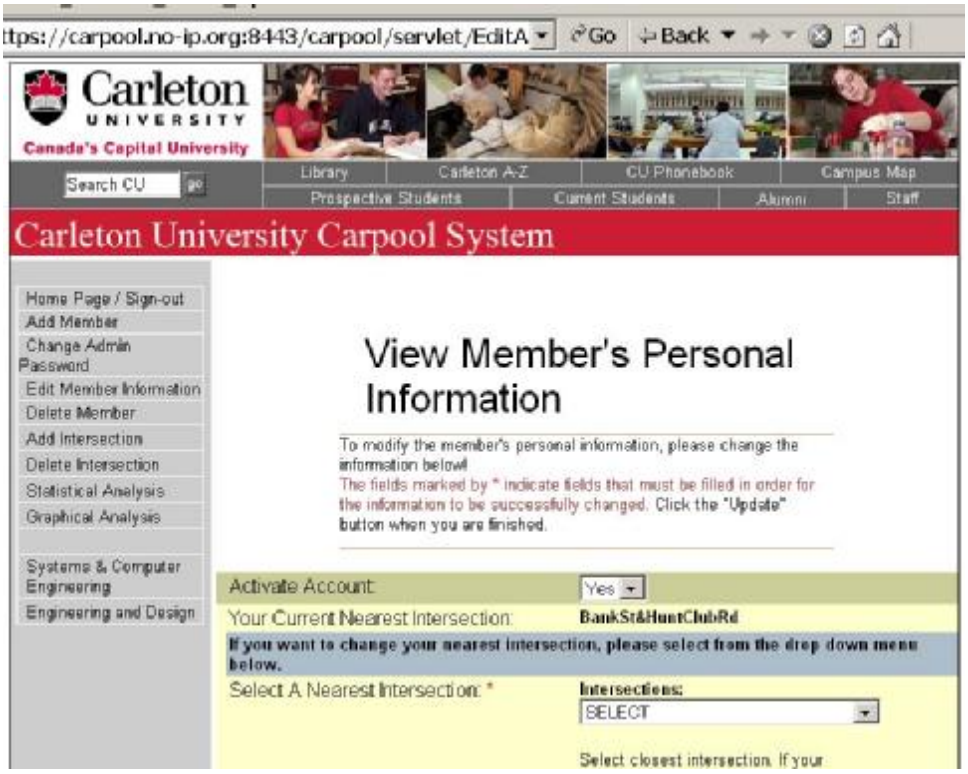


Figure 36

3.6 Statistical/Graphical Analysis

It is useful to have the analysis result that shows the administrator the condition of the system, and that's why there is "Statistical Analysis" and "Graphical Analysis". A statistical analysis shows the administrator the number of people traveling at a particular time range and where what intersection they travel across. Here are the steps to generate Statistical Analysis:

1) Select "Statistical Analysis" on the menu. See Figure **37.**
2) Select the time range, and hit submit. See **Figure 38.**
3) A new page will display the result on the screen. See **Figure 39.**

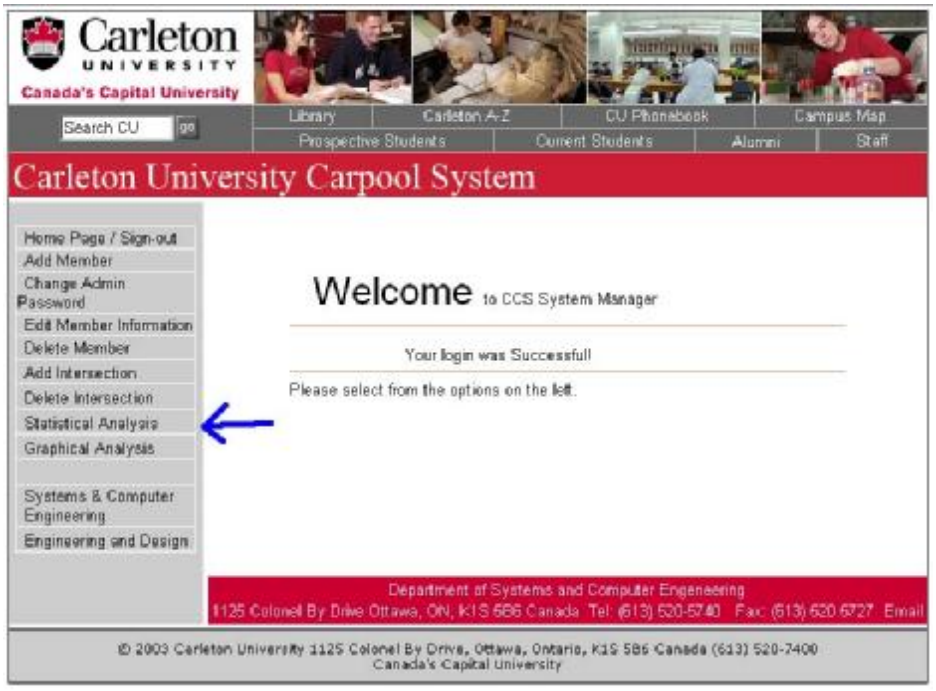


Figure 37,

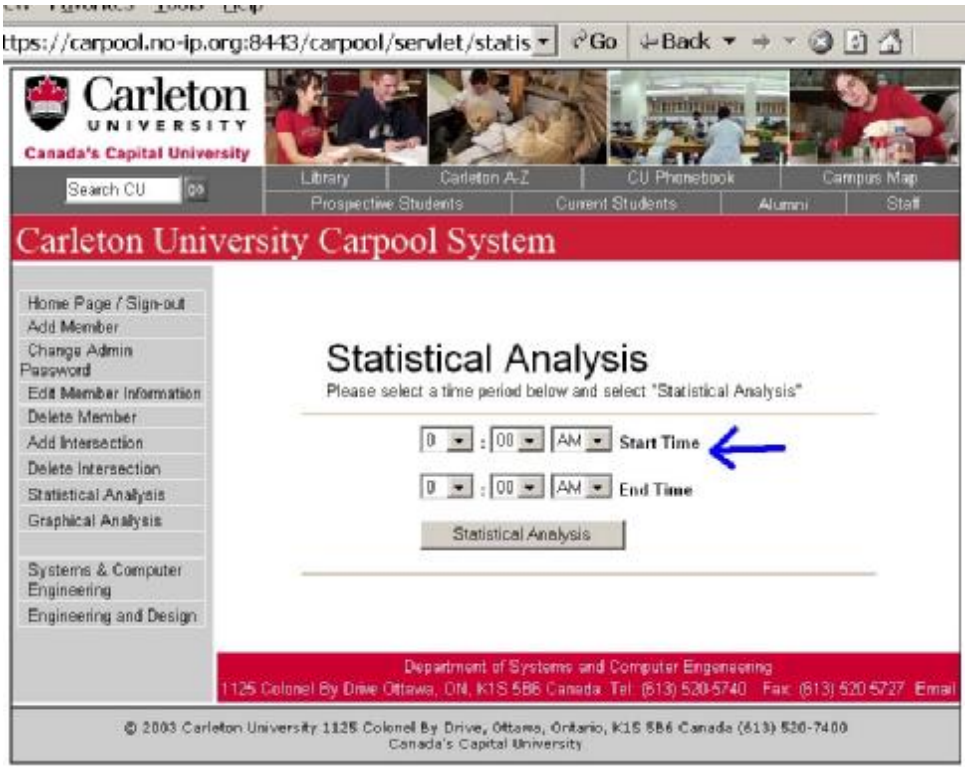


Figure 38.

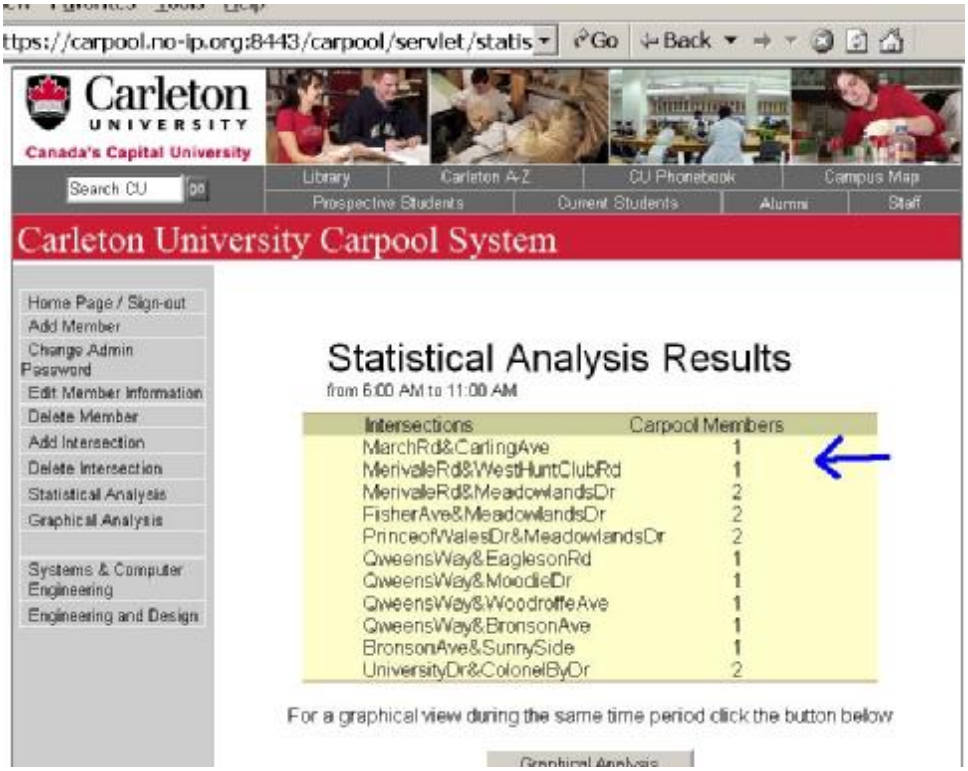


Figure 39.

A graphical analysis shows the administrator the number of people traveling at a particular time range and where what intersection they travel across, further the result is displayed on a map of Ottawa. Here are the steps to generate Graphical Analysis:

4) Select “Graphical Analysis” on the menu. See Figure **40.**
5) Select the time range, and hit submit. See **Figure 41.**
6) A new page will display the result on the screen. See **Figure 42.**

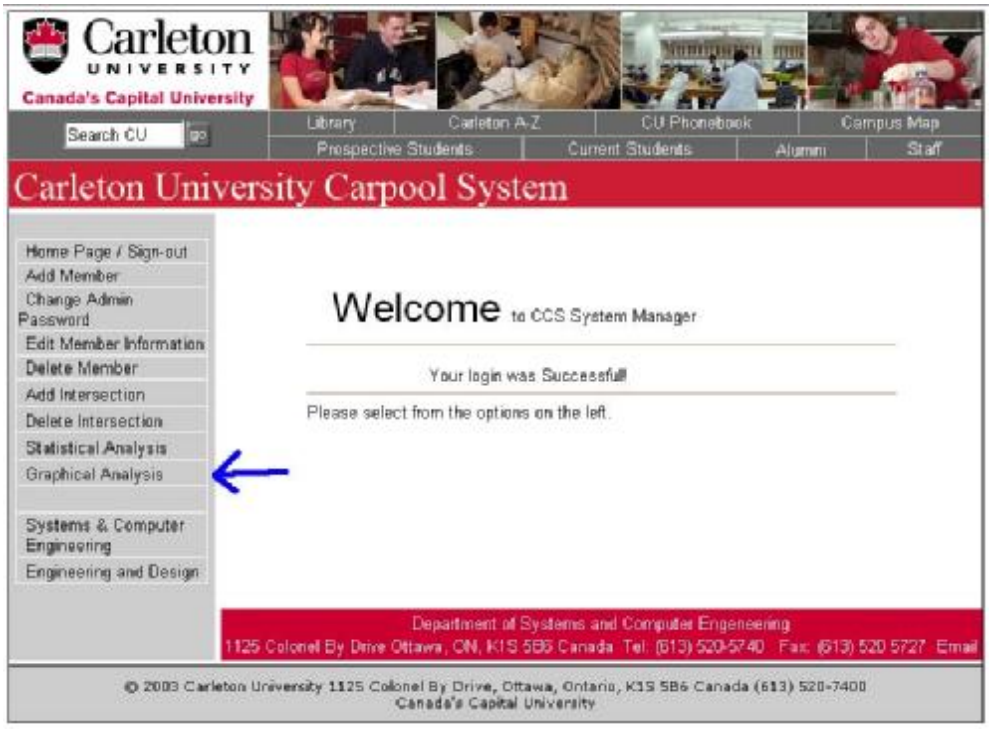


Figure 40

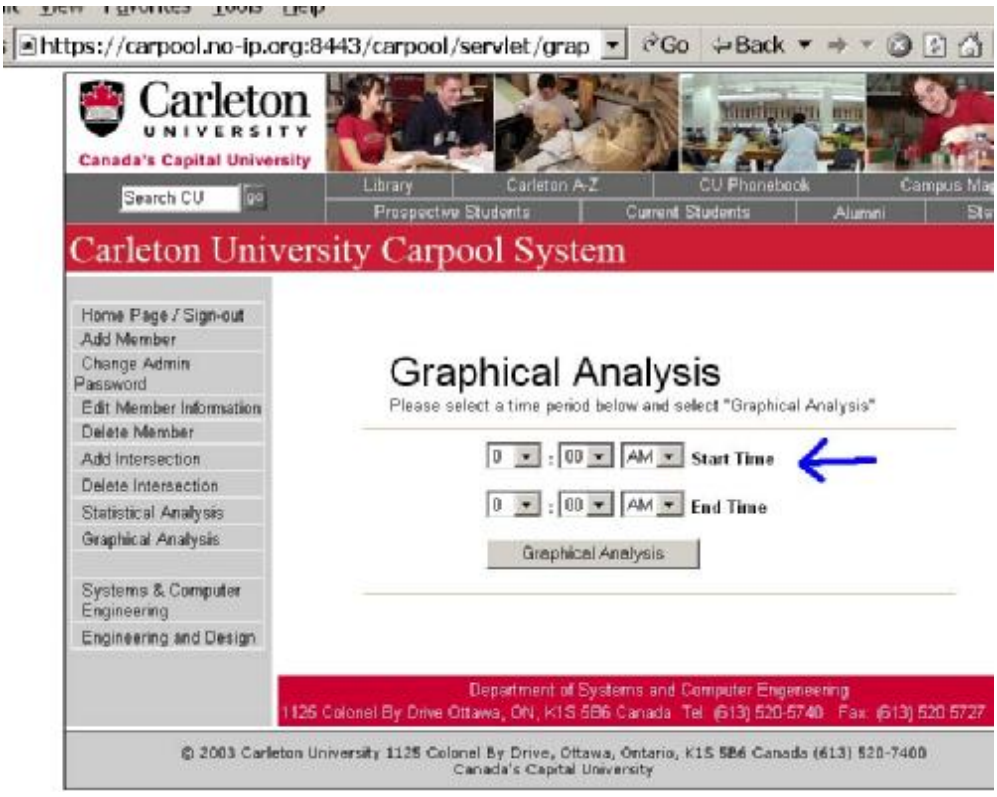


Figure 41

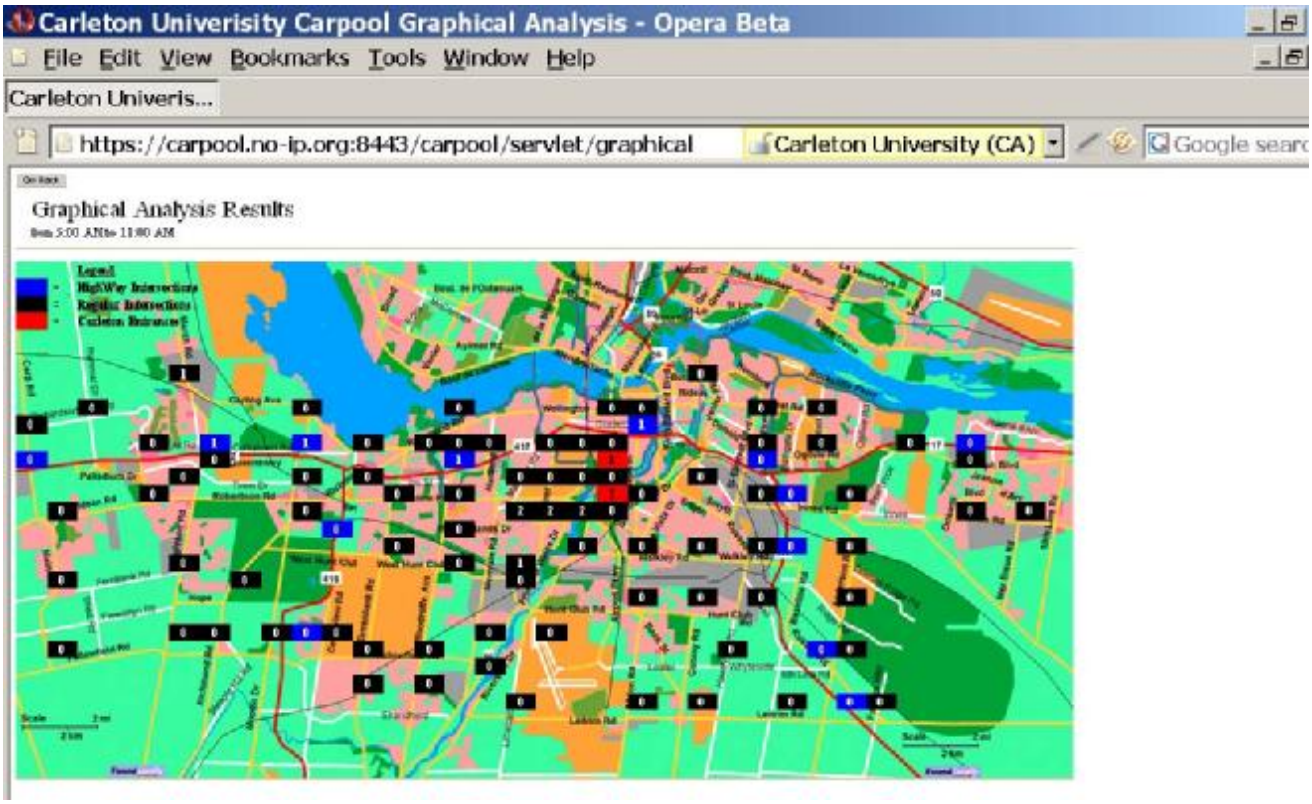


Figure 42.

4.0 Troubleshooting/Cookies

A problem with accessing the website could be that browser cookies are not enabled and the user cannot login as a member or administrator. To enable cookies in “Internet Explorer” click tools then internet options. From internet options click the privacy tab and then lower the security to “Medium High” or lower and then try again to access the website. The administrator can check for errors in the log file located at: Tomcat 5.0\logs\stdout.log

Carleton UNIVERSITY

Appendix B - Programs Required Installation Guide

Carleton University Carpool System

Programs Required Installation Guide v3.0

This document will help you install all the required software to allow you to get your Carleton University Carpool System Running. After reading and completing this Guide, please read the "Carleton University Carpool System Configuration Guide v3.0." There are five parts to this guide: **download**, **before installation**, **installation**, **after installation** and **troubleshooting**.

Download

Download the following software and additional drivers from the URL provided. Note that all the required software and drivers are freeware. Go to the URL and download the setup/installation/zip file.

Software Required:

1. Java SDK (J2SE 1.4.2)

   http://java.sun.com/j2se/1.4.2/download.html (Java SDK)

2. Jakarta tomcat (aka Apache Tomcat)

   http://jakarta.apache.org/site/binindex.cgi (Jakarta tomcat)

3. MySQL

   http://dev.mysql.com/downloads/ (MySQL)

Additional Drivers Required:

1. mysql-connector-java

   http://dev.mysql.com/downloads/connector/ (mysql-connector-java)

2. javabeans activation framework (JAF) (Used with javamail driver)

   http://java.sun.com/products/javabeans/glasgow/jaf.html (JAF)

3. javamail

   http://java.sun.com/products/javamail/downloads/index.html(javamail)

Before installation

Generally the first thing to note is that Apache Tomcat is your server and mysql is your database. JAVA SDK and the additional required drivers listed above are there to allow you to use java-servlets to interact with Apache Tomcat and mysql and hence create your program.

Please install Java SDK first then the others in any order. For the *additional drivers* I recommend that you install (they are just zip files no real installation except putting it somewhere and then pointing to it, read on) the additional drivers on your c drive and use the default folder that is extracted to hold the contents. I used "C:\mysql-connector", "C:\j2sdk1.4.2_01\jaf-1.0.2", and "C:\j2sdk1.4.2_01\javamail", but you can use any folder naming as long as your point to it right. Now how do I point to the drivers and driver folders (see the after Installation comments at the end of this guide), you are going to basically create environment variables so that programs like the server Tomcat Apache can use the drivers.

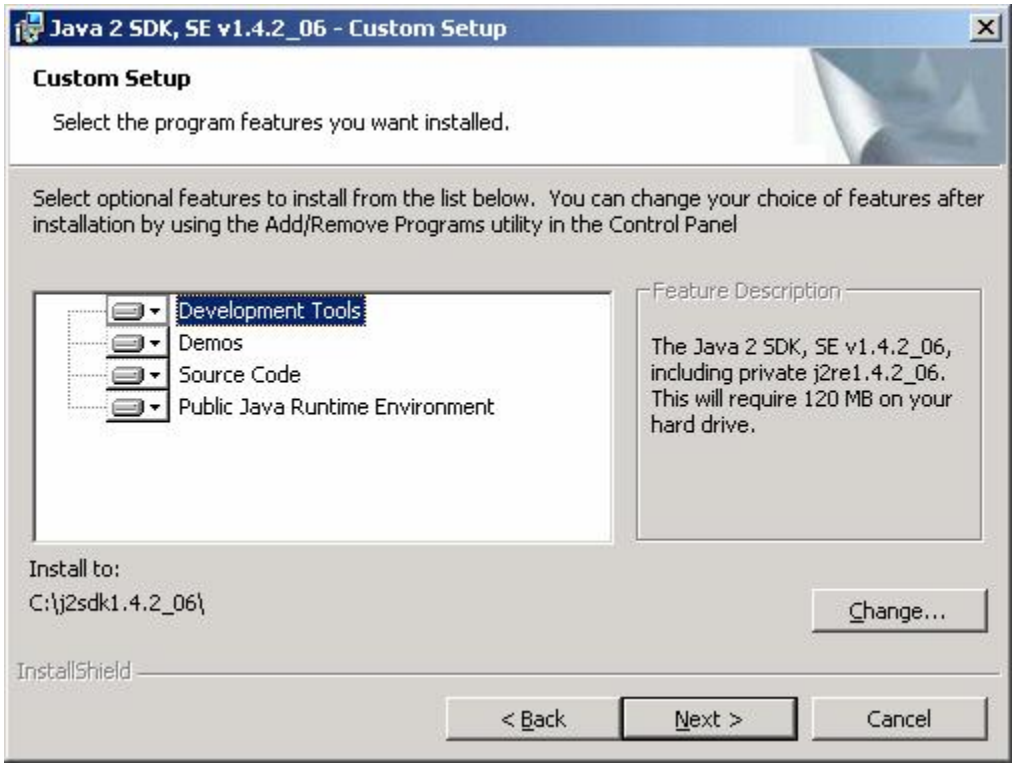
Apache Tomcat is a server that is made for hosting java-servlets and jsp pages on the internet (Carleton University Carpool System uses java-servlets). Apache Tomcat is capable of serving more than just java-servlets (i.e. static html pages), but is slow at serving static html pages; therefore, many users will install Apache Tomcat and also Apache (aka Apache HTTP server). We however did not install both Apache Tomcat and Apache because currently there is no need to increase speed and complicate more the installation guide.

Installation

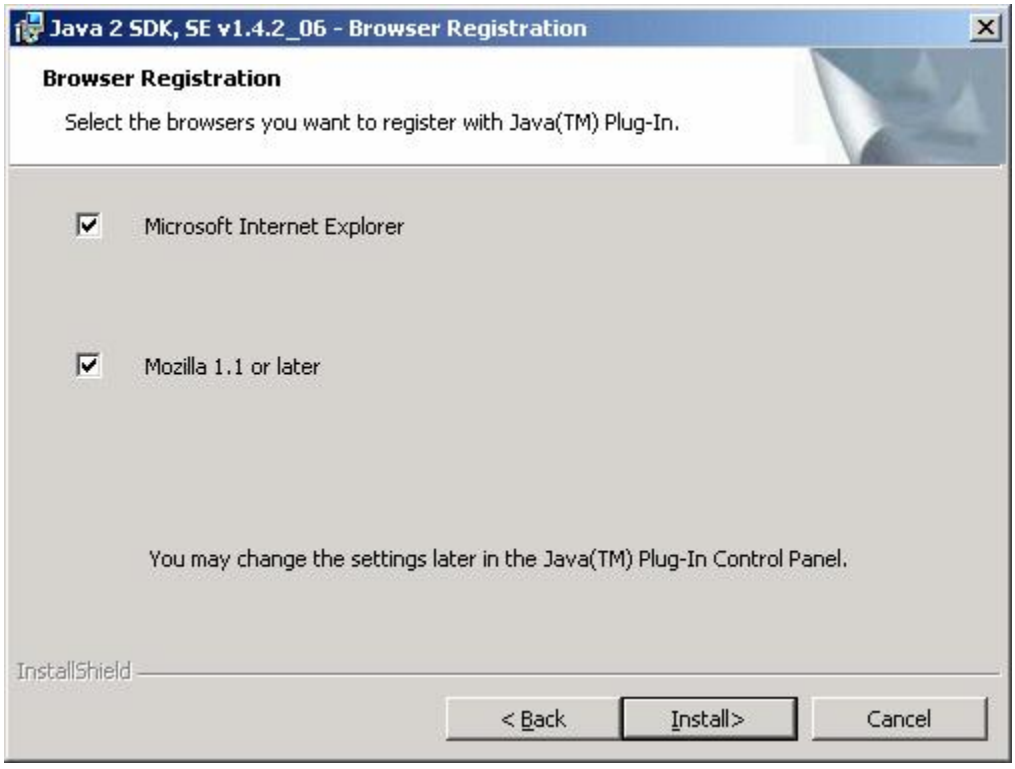
This part may vary with different versions. Each new version of the application may have similar features, but the installation process changes overtime usually to simplify the process. Here are the installation instructions step by step please first install JAVA SDK. You can generally use the installation guide that comes with the files, because the configuration aspect of the program is done later on in Appendix B: "Carleton University Carpool System Configuration Guide v3.0."

JAVA SDK installation

a. Double click the setup file.
b. Select yes for any default request.
c. Install at the default location C:\j2sdk1.4.2_06 (depends on version number, see next page for screenshot)



d. When asked about java plug-ins select browser IE (does not matter for our purposes), so select yes.



e. Done

f. Do not need to test it

MySQL Installation (mysql-essential-4.1.9-win32)

a. Double-click the installation file.

b. Typical installation selected.

c. Select skip sign-up

d. Configuration(options selected)

   i. Select detailed configuration

   ii. Select server machine

   iii. Select multifunctional database

   iv. Store on c:\MySQL Datafiles\

v. Use auto configuration for Online Transaction Processing (OLTP)

-up to 500 connections

vi. Enable TCPport 3306

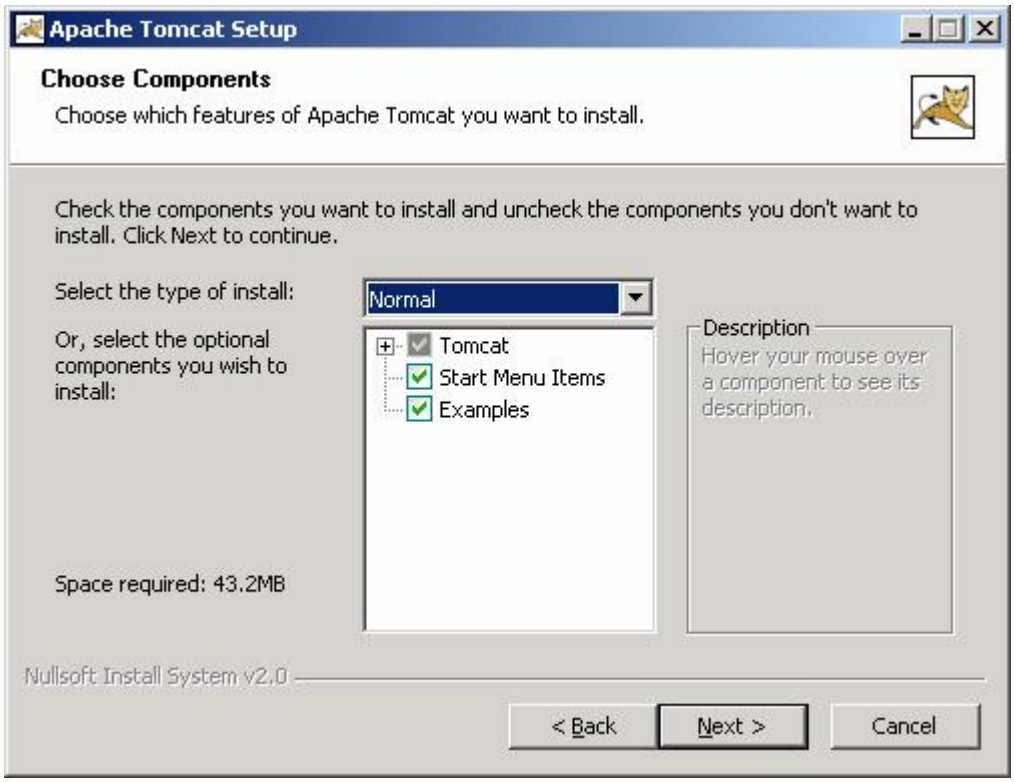
vii. Use standard Character Set

viii. Install as Windows Service

ix. Set admin/root -ROOT/PASS = 'ccsadminG1'

TOMCAT Installation (jakarta-tomcat-5.0.28)

a. Select defaults



b. When prompted, set admin password

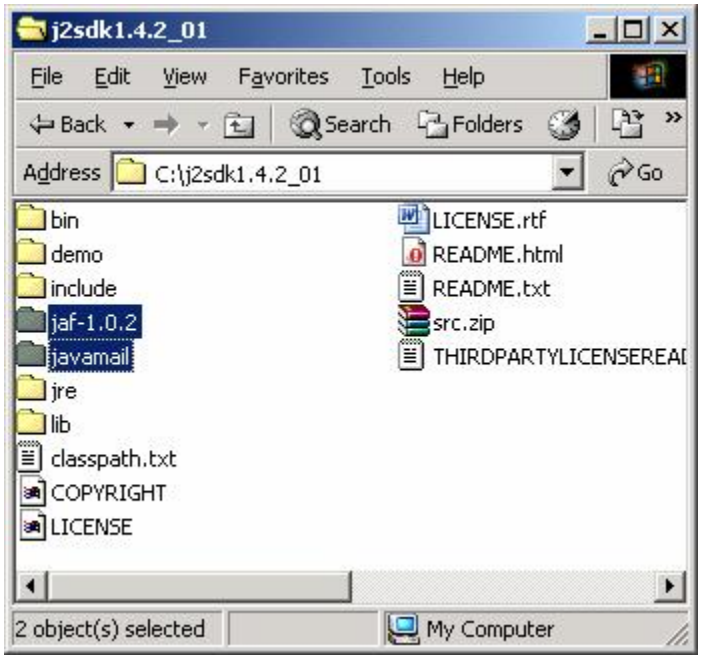
   i. admin:ccsadmin

   ii. pass: ccsadminG1

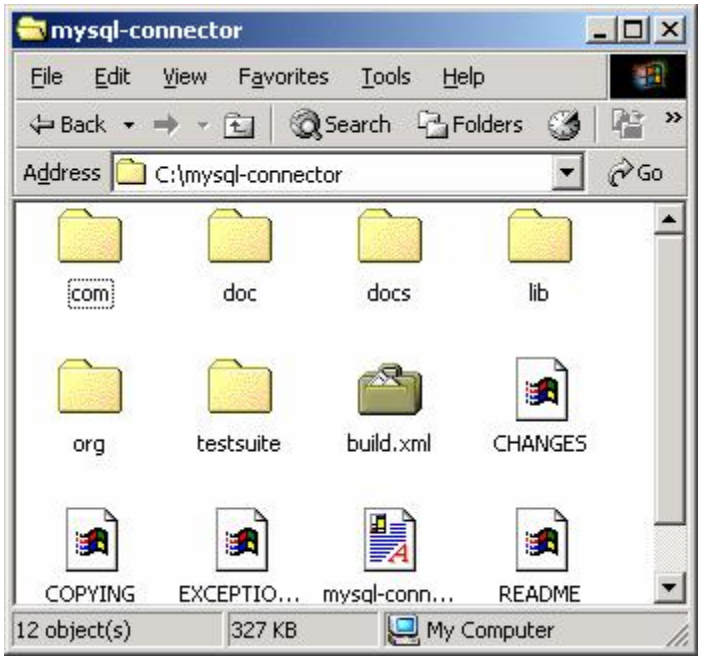
c. It asks for the JVM(java virtual machine just installed) the path is something like this C:\Program Files\Java\j2re1.4.2_06.

Installation/Extract all Additional Drivers to C drive

a. Javamail unzip and JAF unzip into C:\j2sdk1.4.2_06 (see next page for screenshot).



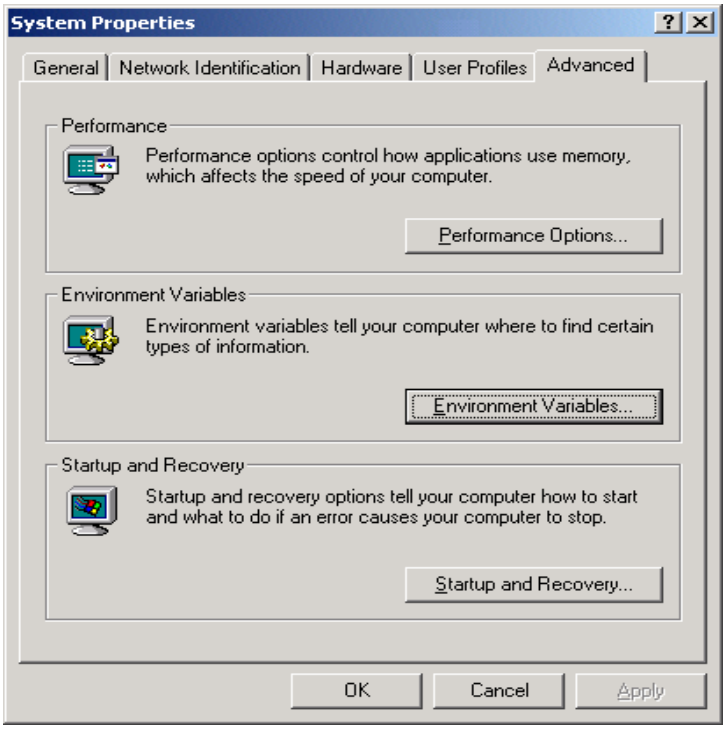
b. MySQL-connector unzip to C:\mysql-connector



c. Done

<u>After</u> Installation (Very Important)

- Time to point to all those Additional Drivers and Java SDK so that programs like Tomcat can use them.
    - Go to “System Properties” (right-click My Computer), select the Advance Tab, Click on the **“Environment Variables”** button. You are going to create the following variables under the “user’s variables” option here: CLASSPATH and JAVA_HOME.



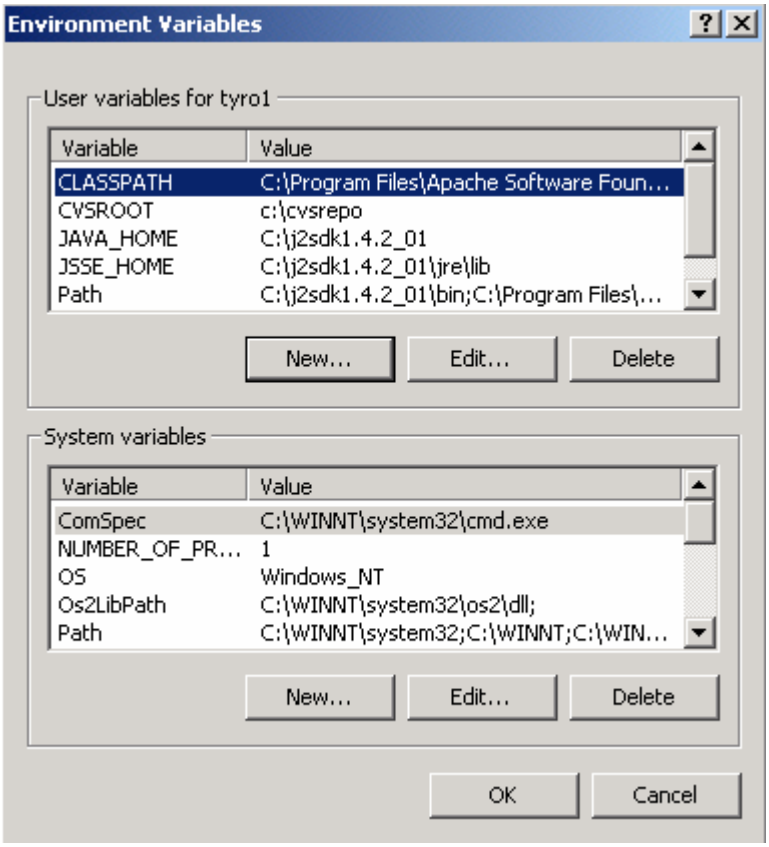
    - Create(Edit if already exist) the following Variables with associating values:

- Variable Name: CLASSPATH

  Variable Value: C:\Program Files\Apache Software Foundation\Tomcat 5.0\common\lib\servlet-api.jar; C:\mysql-connector; C:\mysql-connector\com; C:\mysql-connector\org; C:\mysql-connector\mysql-connector-java-3.0.15-ga-bin.jar;C:\j2sdk1.4.2_01\javamail\mail.jar; C:\j2sdk1.4.2_01\javamail\lib\mailapi.jar; C:\j2sdk1.4.2_01\javamail\lib\pop3.jar; C:\j2sdk1.4.2_01\javamail\lib\smtp.jar; C:\j2sdk1.4.2_01\javamail\lib\imap.jar; C:\j2sdk1.4.2_01\jaf-1.0.2\activation.jar (This is what my CLASSPATH looks like yours will be different depending on where you installed(extracted) the additional drivers and where you install your programs).

- Variable Name: JAVA_HOME

  Variable Value: C:\j2sdk1.4.2_01 (This is what my JAVA_HOME looks like but yours will be different depending on where you installed your JAVA SDK. My Java JSK was installed at C:\j2sdk1.4.2_01 path)



- Restart computer for Environment Variables to take in effect.

- Done!
- You are now ready to read “Carleton University Carpool System Configuration Guide v2.0”

Troubleshooting

1. Mysql interface problems. You may find mysql difficult to use when accessing the database through the command line. We recommend you download and use “MySQL Control Center” (http://www.mysql.com/products/mysqlcc/).

2. Tomcat setup issues: The best source is through reading forums for the specific Apache Tomcat version you are installing (usually it is best to install the latest version). On the main tomcat webpage (http://jakarta.apache.org/tomcat/) you will find documentation for each version available to install, there is an extensive FAQ section (http://jakarta.apache.org/tomcat/faq/).

If you have problems installing you should check out the websites:

All required software and drivers are freeware:

http://java.sun.com/j2se/1.4.2/download.html (Java SDK)

http://jakarta.apache.org/site/binindex.cgi (Jakarta tomcat)

http://dev.mysql.com/downloads/ (MySQL)

http://dev.mysql.com/downloads/connector/ (mysql-connector-java)

http://java.sun.com/products/javabeans/glasgow/jaf.html (JAF)

http://java.sun.com/products/javamail/downloads/index.html (javamail)

Carleton UNIVERSITY

Appendix C - Configuration Guide

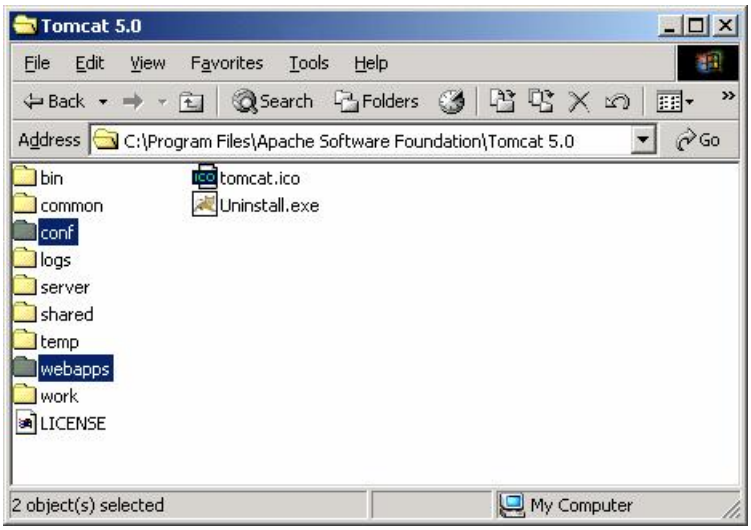
Carleton University Carpool System

Configuration Guide v3.0

Configuration is important it allows you to control and properly run your program. You are going to need to **Configure Apache Tomcat** and **create a SSL key**, **Configure MySQL**, and then **test the Carpool Application**. At the end of this document is a **troubleshooting** section.

Configure Apache Tomcat and create a SSL key

Navigate the tomcat program folder (C:\Program Files\Apache Software Foundation\Tomcat 5.0). Within this folder there are the "conf" folder and the "webapp" folder.



Apache Tomcat is configured with the following program files: **\conf\server.xml**, **\conf\web.xml** and **\webapp\Carpool\WEB-INF\web.xml (the second web.xml file is used to declare servlets for web access for security reasons and is required because the connection pool manager used in the application is partially declared in the file. Connection pool**

manager manages the MySQL connections). Configuration: add the following or remove the commenting-out because some should already be there:

1. Open **\conf\server.xml** add the following:

   For SSL: After this code <Service name="Catalina"> add the following:

```
<Connector port="8443"
                maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
               enableLookups="false" disableUploadTimeout="true"
               acceptCount="100" debug="0" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS"
               keystoreFile="C:\Program Files\Apache Software Foundation\Tomcat 5.0\conf\.keystore"
          keystorePass="changeit"/>
```

   Declare Carpool Program & MySQL connection pool (at bottom) before </HOST> add:

```
          <Context path="/carpool" docBase="carpool" debug="0">

               <Logger className="org.apache.catalina.logger.FileLogger"
                        directory="logs"  prefix="localhost_log." suffix=".txt"
                    timestamp="true"/>

              <Resource name="jdbc/carpoolDB"
                        auth="Container"
                        type="javax.sql.DataSource"/>
                         <ResourceParams name="jdbc/carpoolDB">
```

(Continues on next page)

```
                    <parameter>
                     <name>factory</name>
                     <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
                    </parameter>

                    <parameter>
                    <name>removeAbandoned </name>
                     <value>true</value>
                    </parameter>

            <!--Use the removeAbandonedTimeout parameter / seoonds.-->
                     <parameter>
                    <name>removeAbandonedTimeout </name>
                    <value>60</value>
                    </parameter>

                    <parameter>
                    <name>validationQuery</name>
                    <value>SELECT 1</value>
                     </parameter>
<!-- Maximum number of dB connections in pool. Make sure you
    configure your mysqld max_connections large enough to handle
    all of your db connections. Set to 0 for no limit.
    -->
                    <parameter>
                    <name>maxActive</name>
                    <value>100</value>
                    </parameter>
```

(Continues on next page)

```
<!-- Maximum number of idle dB connections to retain in pool.
    Set to -1 for no limit.  See also the DBCP documentation on this
    and the minEvictableIdleTimeMillis configuration parameter.
    -->
                    <parameter>
                    <name>maxIdle</name>
                     <value>30</value>
                     </parameter>

<!-- Maximum time to wait for a dB connection to become available
    in ms, in this example 10 seconds. An Exception is thrown if
    this timeout is exceeded.  Set to -1 to wait indefinitely.
    -->
                     <parameter>
                     <name>maxWait</name>
                     <value>10000</value>
                    </parameter>

<!-- MySQL dB username and password for dB connections  -->
                    <parameter>
                    <name>username</name>
                    <value>carpool</value>
                    </parameter>
                    <parameter>
                    <name>password</name>
                     <value>system</value>
                    </parameter>
```

(Continues on next page)

```
<!-- Class name for the official MySQL Connector/J driver -->
                    <parameter>
                    <name>driverClassName</name>
                    <value>org.gjt.mm.mysql.Driver</value>
                    </parameter>

<!-- The JDBC connection url for connecting to your MySQL dB.
    The autoReconnect=true argument to the url makes sure that the
    mm.mysql JDBC Driver will automatically reconnect if mysqld closed the
    connection.  mysqld by default closes idle connections after 8 hours.  -->
                    <parameter>
                   <name>url</name>
  <value>jdbc:mysql://localhost:3306/carpool?autoReconnect=true</value>
                    </parameter>
                </ResourceParams>
            </Context>
```

2. Open **\conf\web.xml** add the following:

   Start the "invoker" (Additional Information at bottom of guide) on server startup. Where you see <servlets>…</servlets> add the following lines:

```
<servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>
      org.apache.catalina.servlets.InvokerServlet
    </servlet-class>
    <init-param>
      <param-name>debug</param-name>
```

(Continues on next page)

```
    <param-value>0</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

and then, where you see <servlet-mapping>…</servlet-mapping> add the following lines:

```
<servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Make servlet “index.class” your homepage/welcome page. Add the following after <welcome-file-list>:

```
<welcome-file>servlet/index</welcome-file>
```

3. **Create \webapp\Carpool\WEB-INF\web.xml** file and put the following code in it (This is just another file required for the mysql connection pool used by the Carpool system, see a servlet’s code for more details):

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
```

(Continues on next page)

```
<description>Carpool Application</description>
<resource-ref>
   <description>DB Connection</description>
   <res-ref-name>jdbc/carpoolDB</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
</web-app>
```

4. **Create SSL key**. Open a command prompt window (Dos-prompt) then navigate to where Java JDK was installed. Navigate to the "bin" folder within the folder that contains Java SDK and type the following commands:

(Enter) keytool -genkey -alias tomcat -keyalg RSA

(Enter default password for tomcat) changeit

(Enter first and last name) SCE

(Enter name of your organizational unit) Carpool System

(Enter name of your organization) Carleton University

(Enter of your City) Ottawa

(Enter of you Province) Ontario

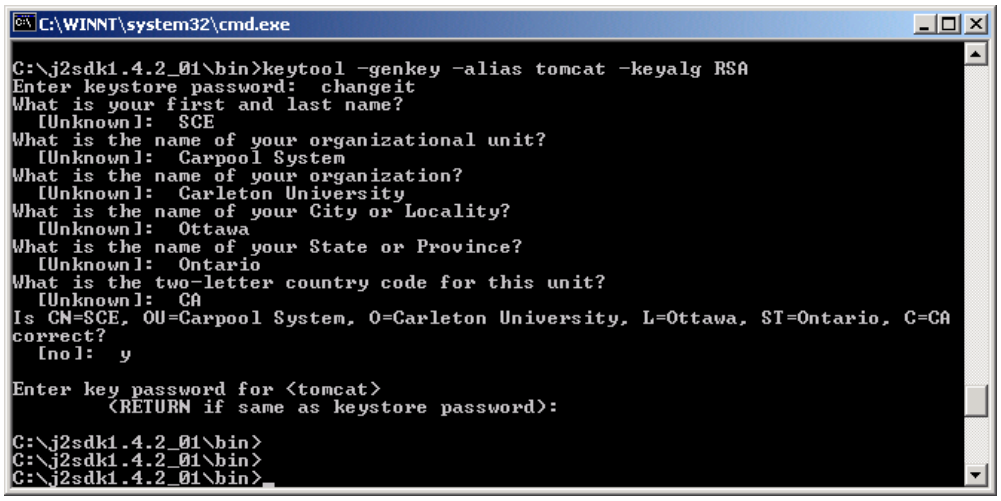
(Enter two-letter Country) CA

(Is information correct) Y

(Enter Key Password for Tomcat) <Press Enter – Don't enter anything>

(Continues on next page)

(Press Enter automatically create password)



The SSL key is called .keystore and is located in your user folder (i.e. C:\Documents and Settings\USERNAME)

5. **Copy Drivers and SSL key** into certain Tomcat folders.

   a. Copy the SSL key called .keystore and is located in your user folder (i.e. C:\Documents and Settings\USERNAME). Paste this file into the conf folder of tomcat (i.e. C:\Program Files\Apache Software Foundation\Tomcat 5.0\conf).
   b. Copy your javabeans activation framework driver file (activation.jar), your
   c. javamail driver file (mail.jar), and your mysql driver file (mysql-connector-java-3.0.15-ga-bin.jar) into two folders C:\Program Files\Apache Software Foundation\Tomcat 5.0\server\lib and C:\Program Files\Apache Software Foundation\Tomcat 5.0\common\lib

Configure MySQL

MySQL will need to be configured so that it has the database required for the system to run and then will need to add a user with all privileges so that the Carpool System can access the database. Start the MySQL database (Start menu Select RUN. Enter ‘cmd’ [PRESS ENTER]. Enter ‘ NET START “Mysql” ’ [PRESS ENTER]).

Create database:

1. Login to mysql as admin. Open a dos command prompt and navigate to the bin folder within the program files installed for mysql. To login type mysql –u root. (If you have a password on the root/admin type the following mysql -- user=root --password= ‘yourpassword’)
2. Open the file “Carpool_db_initialization_v3.0.txt”. Copy the sections and paste sections of commands in the command prompt (copy then right-click mouse over command prompt to paste). You will see that the database is being created.

Create the user used to login to mysql from the servlets:

1. Login to mysql as admin. Open a dos command prompt and navigate to the bin folder within the program files installed for mysql. To login type mysql –u root. (If you have a password on the root/admin type the following mysql -- user=root --password= ‘yourpassword’)
2. Copy and paste the following: GRANT ALL PRIVILEGES ON *.* TO 'carpool'@'localhost' IDENTIFIED BY 'system' WITH GRANT OPTION;

Edit the my.ini file to set wait_timeouts and interactive_timeout so that mysql does not allow idle connections for too long, taking up the connections.

1. Search for my.ini on your windows system. If you have installed mysql on windows 2000 it can be found in the C:\WINNT folder.
2. Add the following lines to the bottom of the file:

   set-variable = interactive_timeout=20

   set-variable = wait_timeout=20

   Note: We set the timeout to 20 seconds so that connections so not remain idle for longer than 20 seconds.

Test System

Steps to take to test the Carleton University Carpool System you just created:

1. Copy the carpool folder that contains the following: images folder, WEB-INF folder(contains servlets, drivers), and style.css
2. Paste the carpool folder into the Apache Tomcat folder called WebApps
3. Navigate to the CarletonBranding.java(carpool **à** WEB-INF **à** classes **à** carpool **à** CarletonBranding.java)
4. Change the constant class variables CARPOOL_URL, CARPOOL_URL_SSL and CARPOOL_DIRECTORY. These three variables are used in the application for all the links (images, servlets…). Use the following if you do not know what

(Continues on next page)

   to use (Note if you do not know your ip address just use “localhost” and it will represent your computer’s address (aka ‘ip’)).

   CARPOOL_URL – “http://localhost:8080”

CARPOOL_URL_SSL – “https://localhost:8443”

CARPOOL_DIRECTORY – “/carpool” (This is the name of the folder used in used in the tomcat\webapps to for the carpool system and it is also declared in the tomcat\conf\server.xml)

5. Start (restart if already running) the Tomcat application (run **à** cmd(ENTER) NET START “Apache Tomcat”)
6. Start the MySQL database(run **à** cmd(ENTER) NET START “Mysql”)
7. Open Internet Explorer enter: http://localhost:8080/carpool

Troubleshooting / Extra information (Highly Recommended Read)

- You are going to configure apache to do the following: work with SSL on port 8443 (or another port depending on what port you want to receive request to use SSL), start the invoker on server startup (only if you are developing or testing else if you are releasing to the internet you should know that you need a web.xml file from your distributor of Carleton University Carpool System), declare your web-application, and copy additional drivers and your SSL key to certain Apache Tomcat program folders.

- The invoker is used in the development of servlets. The invoker allows all servlets in the "classes" folder to run, without the invoker for security reasons no servlet will run. So turn it on, unless this is a release. If this is a **release do not uncomment** any xml statements that contain the term invoker, and make sure that the WEB-INT\web.xml file exist in the carpool folder of tomcat\webapps. Remember this web.xml file is different then the one in the tomcat\conf folder.

- Another thing to note if developing the application further is that changes to the servlets are not immediately seen when accessing the servlet online. You must reload the servlets by shutting down tomcat and restarting it, using the reload feature when declaring the invoker, or best use:
  http://<ServerAddress>:<ServerPort>/manager/reload?path=/carpool
  (Example: http://carpool.no-ip.org:8080/manager/reload?path=/carpool)

- MySQL connection pool is used to connect to the database it is API is supported by Tomcat. Basically you set a couple of parameters and when you ask for a connection to perform some queries the connection pool will give you a connection. The connection pool also takes care of dead connections caused by the software and mysql timeouts of connections.

- The Carleton University Carpool System uses SSL for some of its pages to encrypt sensitive information (Personal information) being sent to the server. We used port 8080 for regular html serving and port 8443 for serving the pages with SSL.

  Example:

  http://localhost:8080/carpool/index.html (non-secure)

  https//localhost:8443/carpool/index.html (Secure, note both the port is now 8443 and that http is now **https ß important to remember**)

- *Localhost* refers to your computer that is going to serve your html pages to the internet you can also use your computer's ip (Internet Protocol) address. The reason why you would want to connect to your server (localhost) is to test if your servlets, jsp's and html pages are correctly working. To find your ip type "ipconfig" in a command prompt and press enter. This ip address will not work if your server (localhost) is on a LAN (Local Area Network) and trying to connect to your server computer from outside your network. In this situation you must forward request that come from the internet to your router's ports (8080 and

8443). This is easy connect to your router and find options to forward ports or better use the virtual server options and forward the request to your computer's IP.

- **Important! The application used to compile the project will need to have reference to certain files to compile the java source into a class.** These reference are the drivers for mysql, java-mail, servlets, tomcat-utilities(basically classes used to create the functionally of the application). If using JCreator, to add the references you need edit the JDK profile for the JAVA SDK you installed: Configurations **à** options **à** select the jave you installed and click edit. Fina and then add the following achieves to the classes tab:
    - Mail.jar (javamail)
    - Activation.jar (javamail)
    - Tomcat\common\lib\servlet-api.jar
    - Mysql-connection-java-VERSION-ga-bin.jar
    - Tomcat\server\lib\catalina.jar
    - Tomcat\server\lib\tomcat-util.jar

  Now you should be able to compile the java sources into class files.

Appendix D - Moving the server/application setup

Moving the server/application setup

This document outlines the following tasks:

- How to change the modulize URL?
- How to change the administrator's email Address?
- How to change the systems emailing address and smtp address and password?

How to Change the Modulized URL?

1. Navigate to Tomcat Home Directory\webapps\carpool\WEB-INF\classes
2. Three Major System Variables must be set through changing static variables of Carpool\CarletonBranding.java:

   CARPOOL_URL: Value= "http://<Address>:<Port>"

   CARPOOL_URL_SSL: Value= "https://<Address>:<Port>"

   CARPOOL_DIRECTORY: Value= "/<webapp's folder name>"

   Once changed, compile Carpool\CarletonBranding.java, and then compile a class that uses Carpool\CarletonBranding.java (ex. index.java). Note that in compiling these files you must set certain variables within the compiler application (see "Extra Information" last point marked "Important!" in the configuration document "Carleton University Carpool System Configuration Guide v5.doc". The variables that are set allow the compiler to find classes that are required to compile the Carpool application (example: import javax.servlet.*; import javax.servlet.http.*;…).

How to change the administrator's email address?

1. Navigate to Tomcat Home Directory\webapps\carpool\WEB-INF\classes
2. One Major System Variable must be set through changing static variables of classes\EmailAdmin.java:

emailAdmin1 = email address of admin one

emailAdmin2 = email address of admin two

Once changed, compile classes\ EmailAdmin.java, and then compile a class that uses classes\ EmailAdmin.java (ex. SubmitBugs.java). Note that in compiling these files you must set certain variables within the compiler application (see "Extra Information" last point marked "Important!" in the configuration document "Carleton University Carpool System Configuration Guide v5.doc". The variables that are set allow the compiler to find classes that are required to compile the Carpool application (example: import javax.servlet.*; import javax.servlet.http.*;…).

How to change the systems emailing address and smtp address and password?

1. Navigate to Tomcat Home Directory\webapps\carpool\WEB-INF\classes
2. Major System Variables must be set through changing three lines of code of classes\EmailAdmin.java, classes\Carpool\CarletonRegister.java, classes\ChangePassword.java, classes\SendEmail.java:
    - **SMTPServer** =<SMTP Server Address>;
    - **emailFrom** = <System Email Address> ;
    - **return new javax.mail.PasswordAuthentication**(<System Email Address>, < System Email Password>);

    Once changed, compile all classes. Note that in compiling these files you must set certain variables within the compiler application (see "Extra Information" last point marked "Important!" in the configuration document "Carleton University Carpool System Configuration Guide v5.doc". The variables that are set allow the compiler to find classes that are required to compile the Carpool application (example: import javax.servlet.*; import javax.servlet.http.*;…).

Carleton UNIVERSITY

Appendix E - Backup and Restore

Backup and Restore

Setup a Backup Administrator's Application

Backup the Carleton University Carpool System daily by executing the following MS-Dos command from within the mysql "bin" directory (this directory contains the file mysqldump.exe):

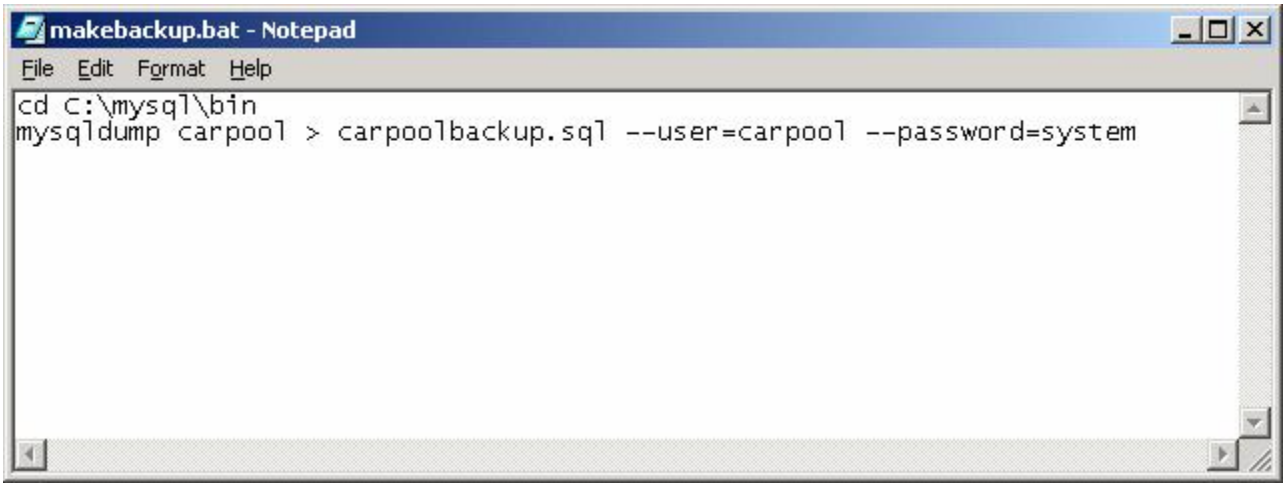
```
mysqldump carpool > [filename for backup] --user=[username] --password=[password]
```

A quick application can be written in a batch file and have the windows "task scheduler" run it once a day or every week. The period should be based on how often the admin checks the system, so that if an error is detected the administrator can retrieve a good backup based on the last time he/she check for a stable system. The batch file might look something like this:

```
cd C:\mysql\bin

mysqldump carpool > carpoolbackup.sql --user=carpool --password=system
```

To make this a batch file, just put the two lines above in notepad and save the file as "makebackup.bat". To execute just double click. See the figure below for a screenshot of the carpoolbackup.sql file.



Restore the database from the backup created

If for some reason an error occurs in the database corruption or virus. The system can be fully restored by complete recreating the database with all its records. **Note all changes that occurred after the backup will be lost.** First delete the carpool database and recreate it by logging-in as admin:

<Logging-in> mysql --user=root --password=[password]

<Delete Database> drop database carpool;

<Recreate Database> create database carpool;

<Recreate web application user> GRANT ALL PRIVILEGES ON *.* TO 'carpool'@'localhost' IDENTIFIED BY 'system' WITH GRANT OPTION;

<Exit> exit

<Restore the Database> mysql --user=root carpool< carpoolbackup.sql

This should restore the system to the original state to when the backup took place.

Note: if you do not want to lose information that was entered in after the last backup was taken you must research the use of the mysql tool in the bin directory called *mysqlimport.exe*.

Carleton UNIVERSITY

Appendix F – Security: Quick Overview

Security – Quick Overview

Database

- Database is not encrypted, because it is seen as not necessary as the security of the system at this level is not required.
- Accessing the database from the Carpool System involves a system password hard-coded into the top of all the classes.
- Passwords are the only thing encrypted
- Forgot password uses an email link that contains the user's password MD5 encrypted
- Activation email link uses the user's password plus a secret constant MD5 encrypted. This encrypted variable is described as a password in email sent to the user, even through it is not (extra security).

Tomcat

- Publicly accessible servlets are declared in the webapp/carpool/WEB_INF/web.xml document to prevent access to unauthorized internal servlets.
- Cookies are enabled

Carpool System

- Session management is used to track online members for real-time tracking.
- Secure Socket Layer (SSL) is used when sending data from any of the member and administrator interfaces.