Free University of Bolzano/Bozen

Faculty of Computer Science

Thesis

Combining user and service provider preferences in a mobile car-rental reservation system

Thomas Schievenin

Submitted in partial fulfillment of the requirements for the degree of Bachelor in Applied Computer Science at the Free University of Bolzano/Bozen

Thesis advisor: Prof. Dr. Francesco Ricci

July 23, 2010

Abstract

This thesis project aims at developing a software application of the latest generation and to study the analysis of two completely different car rental systems. The starting point was to develop a system that took into consideration the needs of the customer and of the supplier. I implemented a mobile application with an android operating system also able to give importance to the necessities of the final user.

The defined architectural system developed an application able to satisfy the needs of the supplier of the service and that of the final user in a way to offer a service based on the needs of the consumer.

Once finished the development of the application it was submitted to colleagues and friends. The system was evaluated through a questionnaire. The user sample expressed judgment regards to the usability of the application and the preferences between the two implemented systems. The results were useful to analyze the two systems, to compare them, to find points of strength and weakness and to complete conclusions. The results have confirmed our expectations placed at the beginning of the project, because the merging system received a positive evaluation from the users.

Riassunto

Questo progetto di tesi mira alla realizzazione di un software per dispositivi mobili di ultima generazione e allo studio e sucessiva all'analisi di due sistemi di autonoleggio completamente diversi. Partendo da un sistema dove solo le esigenze del gestore del servizio venivano prese in considerazione, ho implementato un'applicazione per dispositivi mobili con sistema operativo android capace di dare importanza anche alle necessità dell'utente finale.

Definita l'architettura del sistema è stata sviluppata un'applicazione capace di soddisfare sia le esigenze del gestore del servizio che quelle dell'utente finale, in modo tale da offrire un servizio cucito addosso alle esigenze dell'utenza locale altoatesina.

Non appena terminato lo sviluppo dell'applicazione, questa è stata sottoposta a conoscenti e colleghi. Il sistema è stato valutato tramite un questionario dove il campione d'utenza esprimeva giudizi riguardo l'usabilità dell'applicazione e la preferenza tra uno dei due sistemi implementati. I dati raccolti sono stati utili per analizzare i due sistemi, confrontarli, trovare i punti di forza e debolezza e stilare le relative conclusioni. I risultati hanno confermato le aspettative poste all'inizio del progetto in quanto il sistema di "fusione" delle preferenze ha ricevuto una valutazione positiva dagli utenti.

Kurzfassung

Das Projekt besteht aus einer Anwendung für mobile Geräte neuerster Generation und dazu auch aus dem Studium und der Analyse von zwei voneinander unterschiedlichen Autoverleihsystemen.

Ich habe für einer in Bozner Gesellschaft eine interaktive Webseite entwickelt, welche mittels einer neuen Technologie als Knotenpunkt für Kunden und Angestellte dienen sollte. Das Anfangssystem, welches nur auf die Bedürfnisse des Dienstanbieters achtete, wurde zu einer mobilen Anwendung angepasst, um auch die Erfordernisse der Kunden des Dienstanbieters zu vollbringen.

Die Architektur des Systems basiert auf meine vorherigen Arbeitserfahrungen bei einem südtiroler Autoverleih, und erlaubt daher eine maßgeschneiderte Anwendung für Kunden und Anbieter.

Sobald die Realisierung zu Ende war, wurde die Anwendung Kollegen und Bekannten zum Testen übergeben, welche dazu auch ein Fragebogen ausgefüllt haben. Die gesammelten Daten sind sehr nützlich gewesen, um die unterschiedlichen Systeme zu analysieren, zu vergleichen, und um Stark-und Schwachpunkte aufzufinden. Das Ergebnis der Umfrage wiederspiegelt im Ganzen meine am Beginn erläuterten Erwartungen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 General Information

Nowadays, the use of Internet and mobile devices has become indispensable to the point that we cannot imagine a single day without them. Technology has permanently changed the way we live and work[1]. The need to have access to services at anytime and everywhere seems to be the most important thing in our lives and it has become part of our routine. At work, at university and even on the road, people access Internet to download data and consumer services.

In Trento, for example everyone can surf for free from many points of the city and this is encouraging new users to adopt even more online services.

In the specific business of car rental there are mainly two possible actors. The first is the service provider that gives the possibility to rent its goods for a specific period of time, and the second is the customer that is interested in renting cars for personal purpose. Local car rental services in South Tyrol do not provide a mobile application for requesting cars, and in most cases they do not even dispose of a mobile website that offer their services. This is the starting point of my Thesis Case Study: offering both the customer and the provider a system implemented on the newest platform for mobile systems, Android, that helps them to find a compromise between actors' needs.

1.2 Reservation System

At the beginning of the project an extensive state of the art was conducted to a local car rental reservation company in order to collect all the data possible in order to develop a modern and reliable application based on a real car rental business. After this phase we concluded with a list of requirements set up by the service provider. In order to get the customer's requirements we defined them autonomously.

We have taken into consideration the service offered by two car rental websites (Avis[2] and Eropcar[3]), which are complex systems and give the user the possibility to have a wide range of solution. Another website we analyzed was made by me for a local company in Bolzano (Avis[2]), but in this case the company do not have a large quantity of goods to rent, therefore the website is simpler than that of Avis and Europcar.

We decided to implement this system on mobile phones because they give the user the possibility to take advantage of having an available service at anytime.

Since this application belongs to a niche market it has greater possibility to have success. Both actors in this business have advantages by using this new application: on one side customers always have a car rental system available, on the other side the service provider gets the possibility to offer an on the move systems which can be accessed everywhere.

The decision of making the system runnable on mobile phones is determinated by the fact that we think a mobile application more readable and easier-to-use than a website based system.

1.3 Outcome

The android application implements two different car rental techniques that offer either rental services considering jointly the customers and suppliers needs or just the costumer's. In the first system, only the supplier preferences were taken into consideration in order to determine which goods had to be rented, and in the second, the system makes a merge between customer's and provider's preferences. The main aim of my Thesis Project was to demonstrate that the second system, the one that merges the actors' needs, is preferred to the one that takes only the provider's needs.

The following parts compose the developed application: the Android application that implements the car rental reservation system which gives the user the possibility to interact with the service, the application Server – Tomcat which contains all kinds of services that costumers can call from their cell phones, actually it is the bridge between the android application and the PostgreSQL, and finally the PostgreSQL database where all information about cars, parking and bookings are stored.

Many available applications such as meta-search engines implements ranking functions, in order to retrieve documents and data based on different criteria. These techniques are useful in order to find the best solutions for both costumer and provider. We want to identify a possible relation between actors' preferences. In this way we are able to build merging functions, but this topic will be discussed in the fourth chapter called "system description".

After the implementation of the above mentioned car rental reservation systems, we created a valuation process where users could express their opinion about the two systems. Our hypothesis is that the system that is capable of taking the user preferences into consideration should be preferred to the one that ignore them. Therefore, in order to verify our hypothesis, we created a questionnaire that was filled-in by the users after having used the car rental application. The results confirmed our hypothesis: they show that the merging system is the best rated, therefore a winning approach has been unanimously declared.

2 Background

In a car rental reservation system, as in many other business situations, customers and suppliers have distinct roles and their own interests. Generally, suppliers are businesses actors that sell, or like in this specific case, rent goods, whereas consumers are the final users of a certain product. Very often the needs or preferences of customers and suppliers are not the same, and sometimes these are even conflicting. The most important features of a business transactions for suppliers are integrity, quality, reliability and lastly but not least make profit. On the contrary, consumers want a reliable product/service based on their own needs, constraints and preferences. The fundamental problem of these transactions lies on finding an agreement between the customers and suppliers needs.

2.1 Online Car Reservation System

Nowadays, renting a car has become a routine, in fact who travels for working purpose or for pleasure requires a suitable car. In this section we are going to analyze some existing car rental systems provided by the most popular companies.

Avis[2], is a multinational company that gives their costumers two principle channels for renting a car. The first approach is made personally, this means that the customer has to negotiate with an avis clerk in an office. The second, is possible through Internet, the so called "e-commerce" channel. Here the negotiation is different: first the user specifies the time constraints, then the systems shows all available cars regardless users preferences such as car size, car model, ecc... and finally, the user can choose from the displayed cars.

A second example is provided by EuropCar[3]. They give the customer the possibility to rent cars only through the Internet channel. As before, the user gets a long list of available cars from which it must choose the preferred car. In addition to what is provided by Avis, Europcar also has a mobile website for customers that want access to their services everywhere.

The last car rental service we decided to take in consideration is a website I developed in 2006 named GestioneFlotta[4] for the above mentioned Avis company. This is a service only available for public-administration employees of the Province of Bolzano. In this scenario the user requests a specific car, if the car is available it is assigned to him, otherwise the user must make another request. All requests are managed manually, therefore the query is approved as soon as the avis employee decides to do so. In the first two mentioned examples the user gets a long list of available cars. None of the previous websites has a mobile application that interact with the system, perhaps because they are local companies and suppliers do not see a possible future on the mobile market. On the other hand, we must say that the systems taken into consideration have both a well-made graphical user interface, and are extremely easy to use even at the first approach. In the online reservation systems I have given examples of the companies that rent cars to consumers, therefore it is a "business to consumer" strategy[4], where costumers request a specific good and obtain it. On the con-

trary, in the last example GestioneFlotta they have an internal organization, no money is made in these transactions, they are "business to business"and here the process automation is different from the previous scenario. Since the costumer does not make any money, its aim is to distribute goods equally.

2.2 Negotiation Phase

2.2.1 Webs-Based Systems

Car rental reservation websites act more or less in the same way. Negotiation is made one step at the time. For example, the user must first set the time constraint, then select the preferred car type, and so on.

There are no differences between the first two analyzed systems, Avis and Europcar. Both systems have a wide range of available cars, therefore these websites are more complex and complete. The costumer gets the opportunity to choose the preferred car from a big set, unlike in the third system that the small cars quantity changes negotiation in a stipulated procedure in a such way that only one form has to be filled-in, in order to make a car reservation.

2.2.2 On The Move

Why the need of a mobile application? The fact that in June 2007 the total number of mobile phone connections in the world reached 3.25 billion proves that mobiles are not only used for telephone calls and text messaging[5]. In addition, none of the previous systems described in paragraph 2.1 "online car reservation systems" offers such a mobile application that implements these features, probably because a website running on mobile devices is not based on a specific mobile operating system. On the other hand, a mobile application optimizes the screen features, in fact most of the time, surfing in Internet through a mobile phone can be problematic because the screen dimension does not allow to read small formatted texts.

2.3 Rank Integration

In this Thesis Project we consider the problem of combining ranking[6] results from different sources. Generally, many applications include building meta-search engines, combining ranking functions, selecting documents based on different criteria. These techniques are useful in order to solve the rank aggregation problem. In fact, we tried to calculate differences between the two lists: the customer's and the supplier's. How can we calculate such a distance?

Methods such as, "Borda's method", "Footrule and scaled footrule" and "Markov chain methods" already exist. We decided to adopt a linear sum method with sum weight equal to one. As soon as we obtained these two lists we saw their similarity.

The *Kandall tau distance* counts the number of pair wise disagreements between two lists; therefore thanks to it we can obtain a single resulting list that shows how much the two previuos lists have in common. The obtained list

is a normalized version of the *Kandall distance*, which can be computed in *nlogn* time using simple data structures. Once the Kandall tau distance is computed, we have to declare which pairs of the two lists are better. In the generic context of rank integration, the notion of "better" depends on what distance measure we try to optimize. In this specific case we do not optimize the distance measure, but we only take into consideration the pairs that got the highest marks.

3 Case Study

3.1 Project Description

The main goal of this project is to develop a car-rental mobile application available 24 hours a day, 7 days at week able to offer rental services considering jointly the customers and suppliers needs. One of the most important aspect is that the supplier will try to merge the customer preferences with his own in order to suggest solutions that will enable him to use the resources in an effective way and still satisfy as much as possible the customer needs. In order to prove whether this system is useful or not, this application has two mainly features: the first feature tries to compute the above mentioned merge between the different preferences, the second takes into consideration only the provider needs. The final user does not know when the system does the merge or not, and in order to verify if he is able to recognize these two different scenarios a questionnaire has to be filled after the application utilization.

3.2 Historical Data

With the help of historical data and with help of employees that work for a car rental company, we ended up with the following "winter"/"summer" paragraphs in order to have a clearer idea of the request of cars during the different periods of time throughout the year. This kind of analysis is crucial in order to develop a new application service based on an already existing car rental reservation system. It is helpful in order to have a clearer idea of what the problems in this business are. Once understood all possible scenarios that might occur the application development can start.

Winter Cars get more damaged because of the weather, snow, icy roads and tourism around the city make transport more difficult than usual.

Summer Cars do not get damaged as in wintry months, but there is a higher number of requests because of conventions, exhibitions and conferences. This period is characterized by a huge car request, but the resources available are nevertheless of a small amount.

3.3 Required Functions

3.3.1 Customer Side

Unlike the ordinary car rental reservation system where the user specify first the time constraint like in the Avis and Europcar website, here the customer would like also to set a list of requirements that his/her hypothetical must car have but he does not know if its preferences are taken into consideration. The request must fill in the provider's best-car-usage criteria. It is not trivial to find out what are the most important functions to fulfill from the customer's point

Small	Medium	Large
Manual		
City car (4 pass)	Compact (5 pass)	Premium (5 pass)
Economy (5 pass)	Compact SW (5 pass)	Station Wagon (5 pass)
Economy (5 pass)	Cabio (4 pass)	4x4 (4 pass)
Automatic		
Economy (4 pass)	Compact Carrier (5 pass)	Luxury (5 pass)

Table 1: Cars' availability

of view: the customer wants to be part of the deal and see his/her preferences taken into consideration. The most important functionality for the customer refers to the possibility of making a car request. We have listed all the possible requirements that the user should be able to set in order to book a car in table 1.

The user should also state from which parking he prefers to take the required car from. As soon as all preferences are set by the user, the request can be send. The system responds with maximum 2 possible choices from which the user must pick the preferred one. Once the transaction is made, the car is booked and the system saves the booking information. When the booking phase is completed, the user has the possibility to have a nice car-booking list where the information of every journey is shown. Bookings are put in order with respect to the start date and hour of the trip and are graphically marked by the car type. In case of a car accident, it should be possible to send a message to the service provider. This feature is available only within the booking reservation period plus 15 minutes after the end, because most of the car accidents were done at the end of the trip. While the user makes the request it should consider this aspect and be less accurate specifying the trip end date in order to prevent problems to the service provider. The system keeps track of the user's position every kilometer, and if the user requires this service, the system sends the last known position saved during the trip.

3.3.2 Service Provider Side

From the service provider point of view it is extremely important to accurately follow his business strategy, since it is a B2B and no profit is made. Since the service provider does not make money, it must administrate resources in an optimal way, otherwise it will end up with worn goods. First of all, a good renting strategy should not allow to waste resources. Every time a request is made the car availability situation is not always the same. Resources must be equally used. In other words, one should avoid the situation where there are overused cars and maintenance costs should be drastically reduced. Secondly, the system must not treat users all in the same way, for those who often have car accidents the newest cars on the market should not made available. A lot of the time users are not careful using the property of others because of the low

insurance costs. Keeping the newest resources only for reliable users increases the customer's attention while driving, therefore vehicle and repair costs are lowered. Finally, every journey has an appropriate car for its distance. As we mentioned before there are different types of cars from which the customer can express preferences. The service provider makes a distinction between the cars, based on the travel distance: those who want to travel within the province (A maximum of 10 kilometers) are given a small to medium size car. On the other hand, customers who want to go out of the province are given a medium to large size car.

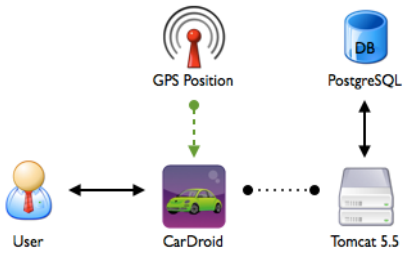

Figure 1: System architecture

4 System Architecture

The system architecture is designed to give to the final user complete access to the car rental reservation system. Communication between the different parts act is as shown in figure 1.

- Cell phone: must be based on the open source platform for mobile systems Android[7] in order to run carDroid applications; CarDroid is the developed application that implements the car rental reservation system;
- Gps sensor: every kilometer, the system keeps track of the current user's position and in case of an emergency the last known position is sent to the web service;
- Application Server - Tomcat[8]: contains all kinds of services that users can call from their cell phones. It communicates with the android application through Soap protocol and also with the PostgreSQL Database[9];
- Database: object-relational database management system (PostgreSQL[9]) where all information about cars, parking and bookings are stored.

4.1 System Functions

The system menu is easy to use. Four tabs are available (figure 2):

- Book a Car: this page is used for making the car request, here the user specifies all of the required characteristics in the car to be hired;
- Booking List: on this page the user can find the list of all accepted bookings and can view all the details of each booking;
- Preferences: in order to use this application the user must specify username and password in order to be used, otherwise the program will not allow any further usage;

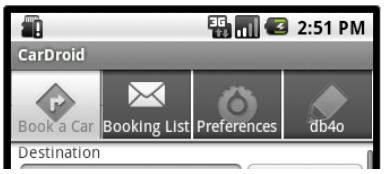

Figure 2: Tab navigation

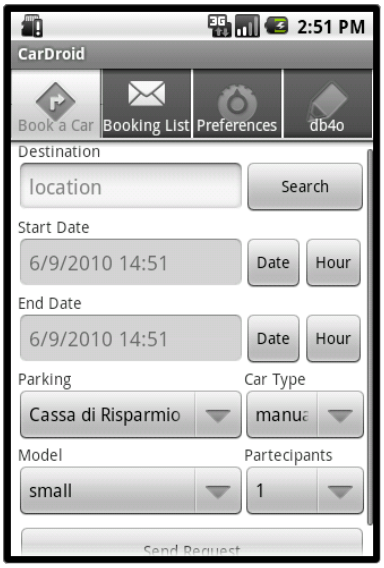

Figure 3: Car request form

- Db40[10]: this tab is useful for developing time in order to modify the open source database for object content. From this tab it is also possible to launch the testing unit.

4.1.1 Form for booking a car

The first tab contains the functions needed for making a car request (figure 3). Every field must be filled- in, in order to make a request, otherwise the system informs the user that some fields are empty.

The following steps show how to make a car request and how the system reacts to the user input. Just few clicks are needed to request the car wanted.

Destination this field acts differently to the others (figure 4). Using this is possible to select the travel destination wanted:

- The user will write the destination, for instance “Laives”;

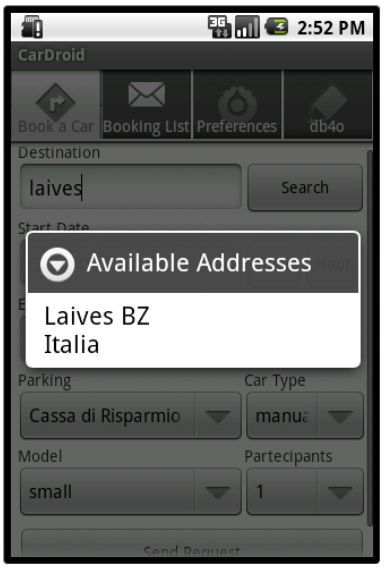

Figure 4: Selecting the destination

Parking List
Cassa di Risparmio Street
Del Ronco Street
Amba Alagi Street
Andreas Hofer Street

Table 2: Parking list

- The application will search for all existing locations named "Laives" and prompt maximum three possible locations. The more specific the location, the more specific the result. Finally, the user must select one of the retrieved results;

Start/End Date In order to specify the amount of time the car is required the user must click on the date and hour buttons. If the start date and end date are not coherent, the application alerts the user when the request is sent off, and gives the possibility to change the time and hour fields

Parking By clicking on the parking field the system shows a list of all the possible car parks from where the car can be rented. The user has to choose one from the table 2:

Car Type the most important car characteristic is the car type. A car can be manual or automatic and this feature is the most important aspect when it

Car Model
Small
Medium
Large
Luxury

Table 3: Cars model

comes to driving a car. By clicking on the button, the user specifies whether the car should be manual or automatic.

Model in this field the user can choose the size of the car, depending on the size preferred. The program allows the user to choose the car size from a custom popup dialog. Possible car models are displayed in table 3.

Participants this field must be filled-in accurately because the number of participants includes the driver.

4.1.2 Web Service Response

Once all the previous fields are filled-in, the last thing that remains to do is to press the "Send Request" button. After that, the system sends the form to the web system, that takes the request, considers the service provider preferences, combines them and sends the result back. Figure 5 shows a possible result to a query.

At this point the user can either go back and restart the process or choose one of the proposed solutions. Once the car has been chosen, the system sends the acceptance message to the web service in order to save the booking. The application also saves the booking in a customized booking list, but more information will be given on this argument in the following section.

4.1.3 Car Booking List

The system keeps track of all accepted requests made by the user in a local database. This is useful in order to look-up information on car reservations whenever it is necessary. To access this functions the user must click on the "booking list" tab and the system lists the most the past bookings (figure 6) with their important details: plate number, car name ,travel destination and the start date of the booking. This is helpful to have a quick general overview. In order to have a simple user interface, every booking has a different icon type with respect to the car type.

If the user wants a more detailed view he can click on a single row, and an information popup shows all the details for that trip (figure 7).

The systems records the user's position in every kilometer and in case of an emergency the user can send the last known position recorded by the application

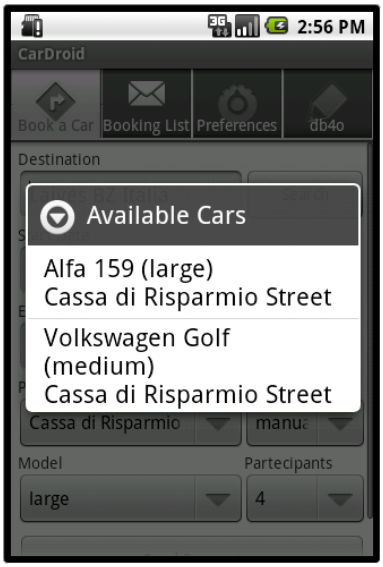

Figure 5: Web service response

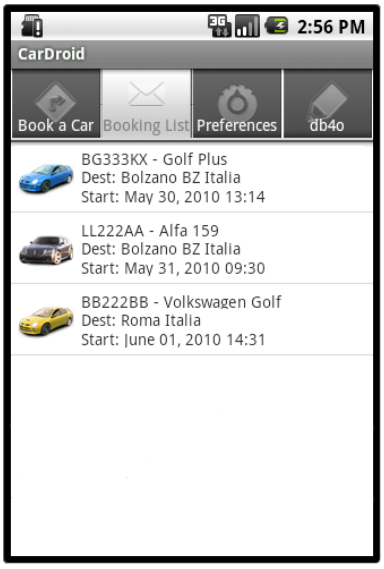

Figure 6: Booking list view

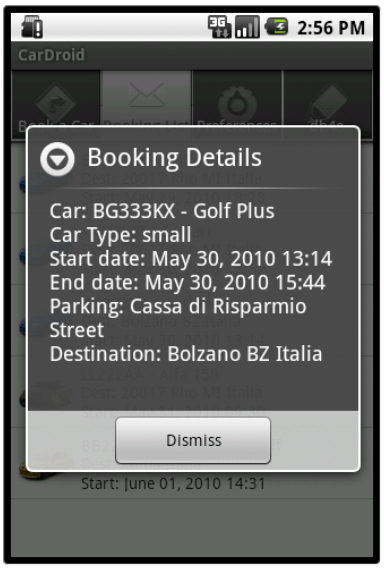

Figure 7: Booking details view

back to the web server to be rescued. This feature is available by clicking on the related booking from the "booking list" view, and press the "car crash help" button. Note that, this button is only available during the travel period plus 15 minutes after the end in order to avoid car parking problems (figure 8).

4.1.4 Preferences

The preferences tab is very easy to use and intuitive. Here the user has to insert its username and password (figure 9). On the "save button" pressure the system checks if the data is present on the service provider database. Make sure to have an internet connection available, otherwise a popup dialog will inform the user to switch it on.

4.2 Computation of the Suggestion

The computation of the suggestion is the most important topic of this thesis project. How is it possible to achieve an integration between costumer and service provider needs? This question will be answered in this chapter. Through the customer request automatically a preferences list is made. Every car is graded and the most preferred cars are the highest rated ones. On the other hand, the supplier rates the cars and automatically makes a list of the best rated cars. The difficulty is making a list to merge both the customer's and supplier's needs, although not impossible. The solution is in finding a suitable mathematical method capable of merging both the actors' needs. In order to

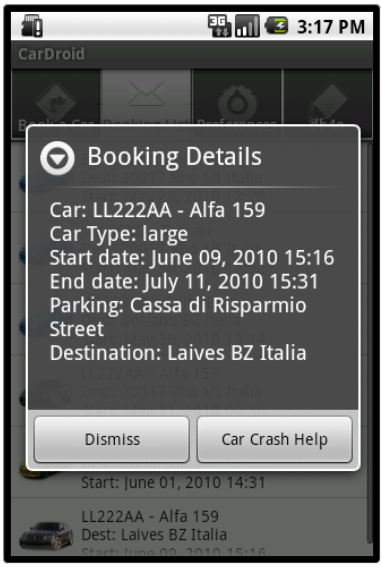

Figure 8: Car crash help

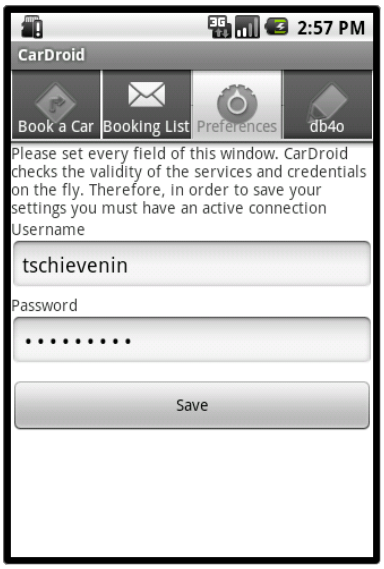

Figure 9: User preferences

unify the two lists to make an optimal rank integration, we must give to both actors' needs the same weight. Only then, we will achieve the perfect merge.

4.2.1 Customer Preferences

The customer sets its own preferences through the car request. Using these preferences the application server generates an ordered list of cars. The highest scored cars are at the top of the list. It acts like the information search engine retrieval. Given a certain input, the output must be ordered with respect to the relevance of the items to the searched information. The following aspects are rated within the interval 0,1 and they will be multiply by a specific ration in order to get a rated list (table 4).

- Car size: available cars have different sizes: small 1, medium 3, large 5, luxury 6. If the customer requires a medium size car, and there is not one available, the system responses with a suitable replacement; this function takes as input two cars and returns as a result their similarity, therefore if both cars are of the same type the function returns value 1, otherwise a number within the interval 0-1. In order to retrieve such value we simply calculate the absolute difference between the requested size and the size of the considered item;

- Car type: cars can be manual or automatic, this feature drastically influences the way one drives. If the customer asks for an automatic car, all cars with that characteristic get a higher score; this method takes as input two cars and if they are both either manual or automatic the score is 1, otherwise it is 0;

- Passenger requirements: in order to optimize the car usage, the system takes into consideration the number of participants for each trip. The closer the number of participants to the seats available in the car, the higher the score for this aspect; The distance between the number of seats available in the car and the number of passengers is transferred in a value of 1 if both numbers are compatible. The formula is shown in table 4.

$$\frac{1-(availableseats-passengers)}{avaiableseats}$$

- Parking requirements: when the customer specifies from which parking it wants to take the car, the system looks at which cars are available in that specific parking. If the cars are not available in the requested car park , the nearest available cars are rated 0 to 1. The nearer the car, the higher the value. Table 4 shows how the computation is made.

We multiply each similarity score by its weight and sum up the obtained results (0.66*0.2+0.3+0.8*0.2+0.3), therefore the similarity value between car A and car B is equal to 0.892.

	Car A	Car B	Formula	Similarity	Weight
Model	Medium(3)	Small(1)	$1 - \frac{idModel_a - idModel_b}{maxIdModel()}$	0.66	0.2
Type	Manual(1)	Manual(1)	$type_a \wedge type_b$	1	0.3
Seats	5	4	$1 - \frac{pass_a - pass_b}{max(pass_a, pass_b)}$	0.80	0.2
Parking	Del Ronco	Del Ronco	$1 - \frac{distance(park_a, park_b)}{maxDistance()}$	1	0.3

Table 4: Similarity example

4.2.2 Service Provider Preferences

Like for the customer, also the service provider sets its own preferences that the system takes into account, in order to build-up a recommendation list. At the top are the most desirable cars for the service provider to rent in that specific case. Clearly, the service provider needs are different from the customer's needs as explained before.

- Trip rating: for trips outside the province the system assigns a higher score to medium/large cars. On the contrary, for short trips the system assigns small cars. All these computations are done by the application server side. We use the "Spherical law of Cosines"[12] that is faster in execution time. The spherical law of cosines formula gives well-conditioned results down to distances as small as around 1 metre, R is the world surface approximation. If the trip is short and the size of the car is small the function returns a value of 1, instead if the trip is long the car should be large in order to have the same value of 1. In all the other cases the returned value is equal to 0. Table 5 shows the formula.

$$d = \arccos(\sin(lat_1)\sin(lat_2) + \cos(lat_1)\cos(lat_2)\cos(long_2 - long_1))R$$

- Good driver rate: if the user ability ratio is high, the system gives a prize to the user by assigning the newest cars available; this function takes as parameters the user's ability and the value of the car usage. If the driver does not damage the car and the car is in good order the returned value is closer to 1. If the driver is careless and the car is not new the same value is returned. In all the other cases the returned value is closer to 0; this method takes as input the result of the following methods;

- Car usage: the same set of cars are not always available. This system gives a high rate to an unused car in order to have an even usage for each car. Using the following formula, we can obtain the value of car usage within the interval 0-1. The closer is to 1, the newer the car.

$$\frac{(maxhours - usagehours)}{maxhours}$$

- User ability: the system keeps track of users behavior, in fact he gets a low rate as soon as they ruin cars or have car accidents. The lower the number of car accidents, the higher the user rate; The more skilled the driver, the higher the value. The formula is the following:

Before the merge:

Customer		Supplier	
Car 1	0.75	Car 1	0.10
Car 2	0.51	Car 2	0.39
Car 3	0.31	Car 3	0.40
Car 4	0.50	Car 4	0.20

After the merge:

Customer and Supplier	
Car 2	0.45
Car 1	0.43
Car 3	0.36
Car 4	0.35

Table 5: Rank aggregation example

$$\frac{(totbookings - chrashes)}{totbooking}$$

4.2.3 Rank Aggregation

Using the above mentioned procedure we can obtain two ranked lists, ordering the available cars according to the customer and service preferences. Then to aggregate[6] these two ranked list we reorder the items according to the the linear combination of the two scores using weight that sums one. At the beginning we have two different ranked lists, then the score of each item in one list is multiplied by a specific weight that model the importance given to that order in the final integrated order. If for instance the two weights are equal (1/2) then this means that the customer and the service provider have equal importance. The example in table 5 shows how to calculate the above mentioned: costumer and supplier have rated two different lists. In order to obtain the final rate for car 1 the following formula is needed: 0.75*0.5+0.1*0.5. The result is shown it the table named "after the merge".

4.3 Architectures

4.3.1 Android Application

The android application has been split up in several packages and each of them has its own specific purpose. A description of each package follows in order to understand the role of each better:

Models In the most general sense models are used to represent in a more abstract way all data. The model is the data representation used by the application in order to compute calculations. In this specific case the models are:

- Booking: represent all information about the trip
- User: represent all information about the user

Persistency The persistency package is in charge to save and retrieve data from a local database located within the android applications. Android developers are typically used to adopt SQLite, an embedded relational database management system, but I preferred to use an open source object oriented database named db4o.

Whoever would like to modify my project thesis, and does not fit in with the db4o library, is free to build its own persistency package adopting its own preferred database manager.

Service This package contains the connection to the application server Tomcat 5.5 and all the different methods called from the application, from which the computation is made, from the outside android phone.

Exception In runtime environments is it helpful to have a class that handles any kind of exception that can occur. Exceptions are special conditions that change the normal flow of the program. The class I have implemented extends the normal java exception class and prompts possible errors in a customized popup dialog.

Test This package checks the efficiency and security of all called methods within the android application. All kinds of database operations such as insert, update, select, and delete are checked here, and also the methods that are connected with the application server.

4.3.2 Application Server - Tomcat

As mentioned before, the application server has been split up in several packages in order to represent every kind of information related to the car rental reservation system world. The packages are the following:

Models The database on the application server side must be more detailed and complex than the small version located into the android device. In fact, below I have shown several models used by the service package in order to compute the merging between customer and service provider preferences. The models are:

- Booking: contains all information about one single booking;
- Car Crash: contains the latitude and longitude of one single car crash;
- Car: describes all car characteristics;
- Parking: contains the location and the name of the parking;
- User: contains all information about the user;

Persistency The package named persistency, contains an extended version of all models defined in the previous package and the class in charge of making the connection between PostgreSQL and the application server itself. Having a different package in charge of communicating with the database management system, is really helpful for those who will work with this kind of system and makes trivial a possible change of database manager. Instead of modifying the entire server application, this kind of structure allows to change the persistency package and still have a working platform.

Service This package contains all methods that join the package "service", in the android application, the user calls for it as it is requested. Its purpose is that when a specific method is called to retrieve information from the database management system (PostgreSQL), it applies specific calculations, and sends the result back to the android application.

4.3.3 DBMS – PostgreSQL

The database structure is the final argument of the architecture chapter. The adopted technology for implementing the database structure is PostgreSQL. This object-relational database management system is easy to use but at the same time powerful and reliable. The system structure is easy to understand. The relational scheme and the entity-relationship[11] model (picture 10) follow in order to get a clearer idea of the database structure:

CarCrash(id_booking, latitude, longitude, timestamp)

User(username, password, merge_technique)

Booking(id, plate_nr, start_date, end_date, latitude, longitude, username)

Car(plate_nr, model, manual, type, max_passenger, latitude, longitude)

Parking(latitude, longitude, name)

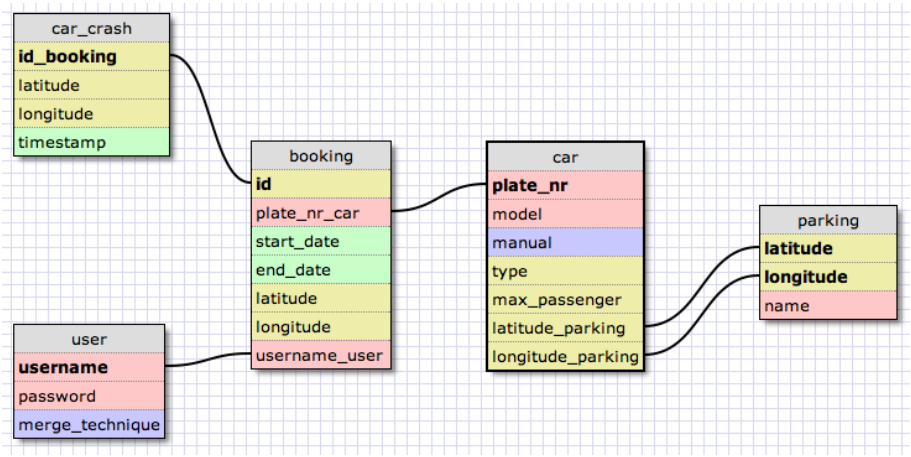

Figure 10: Entity relationship model

5 Evaluation

5.1 Implemented Systems

Before entering into the evaluation process structure we have to describe the characteristics of the two implemented systems. The first system is in charge of retrieving the user's preferences and combine them with the supplier's priority. The outcome will be a set where both user and supplier have equal weight. In the second implemented system only the customer's preferences are taken into consideration, therefore the output will be completely different from the user expectation.

5.2 Experiment Design

In order to test CarDroid, the android application that implements a car rental reservation system, we built a structured survey. The survey we have made includes different areas with its own purpose. These sections are:

Different Scenario The questionnarie starts by describing two different tasks. These were made because the users are to evaluate the two systems, if the same task is assign for both evaluation it is obvious that the second system will get a higher score. This is called "learning". In order to avoid this learning effect we must use the two tasks. These two tasks must be similar otherwise we will not achieve our aim. But since they cannot be completely similar then we must randomize the order and the task so that a times a task is used with one system or the other. The two tasks are shown in table 6.

Task A (working on System 1)	Task 2 (working ok System 2)
Start Date: 23-06-2010 8:00	Start Date: 23-06-2010 8:00
End Date: 24-06-2010 14:00	End Date: 24-06-2010 14:00
Parking: Cassa di Risparmio Street	Parking: Del Ronco Street
Car Type: Manual	Car Type: manual
Model: medium	Model: small
Partecipant: 4	Partecipant: 2

Table 6: Task A and task B

Personal Questions In order to retrieve some information about the user and the its personal taste in the technology, he/she is asked to specify age and if he/she is using a classic mobile phone or a smart phone.

Usability Evaluation There are several questionnaires for evaluating system usability. We choose to adopt the CSUQ (Computer System Usability Questionnaire) in order to evaluate the application CarDroid, because they are standard questions for evaluating the system usability. These questions are general and they can be used for all different type of systems. CSUQ is composed of 19 questions, equal for both the two experimental condition, because the usability does not change, only the system output. We preferred to adopt only the following questions because they are more relevant for the outcome of this study:

- Q1: Overall, I am satisfied with how easy it is to use this system
- Q2: It was simple to use this system
- Q3: I feel comfortable using this system
- Q4: It was easy to learn to use this system
- Q5: The system gives error messages that clearly tell me how to fix problems
- Q6: Whenever I make a mistake using the system, I recover easily and quickly
- Q7: It is easy to find the information I needed
- Q8: The information is effective in helping me complete the tasks and scenarios
- Q9: The organization of information on the system screens is clear
- Q10: The interface of this system is pleasant
- Q11: I like using the interface of this system

- Q12: This system has all the functions and capabilities I expect it to have
- Q13: Overall, I am satisfied with this system

Additional Comments If the user wants to express its feelings about the CarDroid application, we have added some space where positive and negative aspects can be written.

- Most positive aspects (if any)
- Most negative aspects (if any)

Concluding Information The information expressed in this section of the questionnaire are the most important. On the base of this section we will be able to understand what, among the two developed systems, is more useful. The user does not know in supporting which task the system is taking into account her preferences. The provided questions are the following:

- Q14: Which system do you prefer?
- Q15: Which system is the best one?
- Q16: Which system is more useful?
- Q17: Which system is easier to use?
- Q18: Have you noticed differences between the two systems?

5.3 Experiment Strategy

Our hypothesis is that the system that is capable of taking the user preferences into consideration should be preferred to the one that ignore them. In order to prove this hypothesis, the user is asked to perform two similar tasks using the two systems. In one case the system takes into consideration only the supplier preferences and in the other case the system does the merge between user's and supplier's needs. The user does not know in which task/system his preferences are going to be taken into consideration and in order to submit different scenarios we end up with the following solution. Task A, and task B are the previous mentioned tasks, system 1 is in charge of merging the preferences, instead of system 2 that only counts the provider's preferences.

At this point we have four different scenarios (table 7) that help us to avoid the above mentioned learning effects. We have also followed the "within group design"[14], which is a type of experimental design where we observe changes in behavior across different treatments.

Experiment Strategy		
Scenario 1	Task A (system 1)	Task B (system 2)
Scenario 2	Task A (system 2)	Task B (system 1)
Scenario 3	Task B (system 1)	Task A (system 2)
Scenario 4	Task B (system 2)	Task A (system 1)

Table 7: Experiment strategy

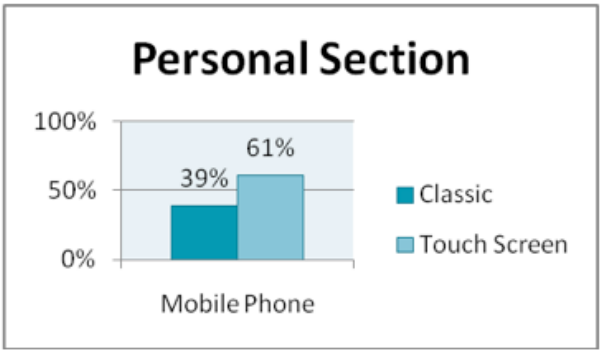

Figure 11: Personal section graph

5.4 Results

This section shows the results we have obtained. The 26 test participants were divided in four groups, around 5 people each. They tested the CarDroid application on the newest Google smart phone: the Nexus One. The obtained results are divided in tree major category: personal, usability and concluding section.

The personal sections shows us the habitual technology used by the sample users. This aspect is really important to analyze, because we wanted to provide an innovative but at the same easy-to-use technology for those who do not adopt usually a smart phone, therefore if the questionnaire outcome shows that the system has a good usability it means that we implemented a simply user interface. Figure 11, shows that the 39% of users that tested the android application, usually make use of classic mobile phones. So from this one can see that there is a good proportion of the users that never used a phone like that used for our experiments.

Usability[13] is all about making things easy and enjoyable to use. According to figure 12, the application had been well evaluated. The user could chose a value between 1 and 5. The higher the rate, the more the usability. All averaged values are grater or equal to 4.1, therefore the application has a good system usability. We can say that good structure, graphical interface and navigability are the bases on which our application CarDroid relies.

Finally figure 13 shows which of the two systems was the preferred one. Question 14 asked the preferred system and as you can see the results are

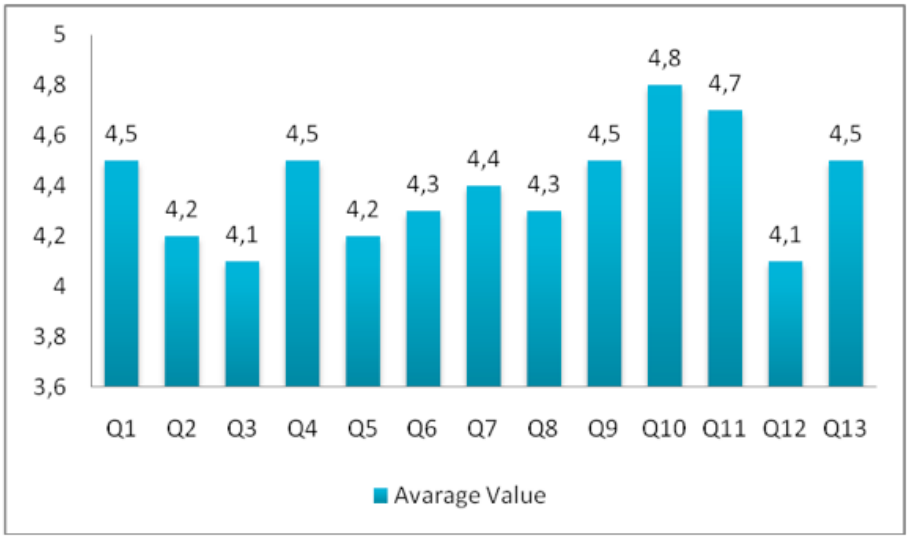

Figure 12: Results of the usability questionnaire

stunning: 98% of users preferred the systems that used the merging technique, the other 2% had no preferences and they specified in the "most positive aspects section" that the systems provided more powerful cars than those requested. Questions number 15 and 16 show that the merging technique had been well rated. Finally, question 17 shows that users did not realize, as expected, any difference in the usability of the system.

Concluding, we can assure that there is a winning technique: 100% of users specified (Q18) that they noticed a non-working properly system. This means that while they were running under the system which was taking into consideration their preferences, no complaints were made. On the on the other hand the users were doubtful about the system output under the second system.

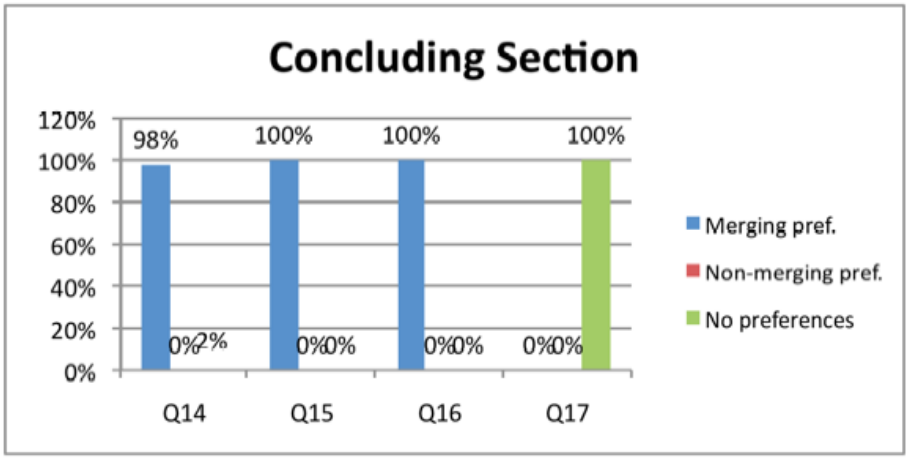

Figure 13: Concluding section graph

6 Conclusions

6.1 Discussion and Results

In this project we provided a new mobile application named CarDroid, that implements a car rental reservation system. It implements two systems where in the first the customer has the possibility to retrieve cars based on its needs merged with the suppliers's and, in the second only the supplier can manage its goods advantagesly. This distinction is useful in order to prove whether the user prefers the system that takes into consideration its preferences. In order to analyze the user's reaction to those systems we created a questionnaire that was submitted to the sample users after the CarDroid utilization. This questionnaire is divided in 5 areas of interest: the first part describes the two tasks that the user must perform; the second part collects information about the personal user preferences and the type of mobile phones used; the third refers to the computer system usability; the fourth refers to the user judgment of the two above mentioned car rental systems and finally, the fifth, where the user is free to express its opinion. In order to have different scenarios, we created a multi factor controlled experiment[14] as described in chapter 5. In this experiment the independent variables are the two different systems. At random we assigned a combination of the variable tasks to the sample users, because we did not want that one particular task or order could bias the outcome of this study. The survey was submitted to a heterogeneous sample of people regardless if they felt comfortable with new mobile technologies or not. The more heterogeneous the users, the better the outcome of the survey. The outcome of this study confermed our hypothesis. Users preferred the system that took their preferences into consideration rather than the one that only took into consideration the one of the supplier's. The outcome of this study completely satisfies

the expectations that we had at the beginning of this Thesis Project. Finally, we can assure that this study has been well evaluated from the final user's point of view.

6.2 Future Work

The CarDroid application implements the basic features on the final user's side. It could be implemented for other mobile platforms such as Iphone or Nokia. Since the application has only been tested by possible costumers, it would be useful to repeat the same experiment strategy where the user samples are the car rental companies of Bolzano.

A missing application is one that gives the supplier the possibility to verify the state of its goods. We think that would be useful to have a website that implements the following features:

Vehicle Situation The supplier must be able to verify the situation of the cars at any time, vehicles available and those in use. At the end of each month the supplier can print-out reports on car usage.

Car Management The supplier can add, edit or erase cars. He can also put a car out of order for a specific period of time for maintenance problems and move a car from a parking to another.

Car-Crash Issue In case of a car accident, the user has the possibility to send an acknowledgement to ask for help and inform the supplier of what is happening. If the car is greatly damaged and the user is no longer able to drive it, a large problem arises. It could be that the car has been booked by other users. In order to solve this problem the website should:

- Extract the all the bookings that overlap with the period in which the car is out of order from the PostgreSQL database;
- Find the most similar and available cars with the functions already implemented on the application server – Tomcat;
- Modify all old values with the new ones in the PosgreSQL database;
- Inform the user with a message displayed on its mobile phone.

At the beginning of the Thesis Project we thought that this feature should also be implemented on the Android application when the user asks for help when a car accident occurs, but this would mean that every user can drastically change the content of the PostgreSQL database, therefore would change all the other bookings made by other users. This feature could also occur due to an error of distraction. On the base of this statement we decided that it would be more adequate to only have this feature available for the supplier.

A Appendix: Questionnaire

The original for the CarDroid System developer

CAR-RENTAL RESERVATION SYSTEM SURVEY

You are going to test an android application that implements a car rental reservation system. Here follows two different tasks where you are asked to book a car with specific preferences. The two tasks will run on different systems. Pay attention to the obtained output. Here follows the two car preferences:

Task A (working on System 1)	Task B (working on System 2)
Start Date: 23-06-2010 8:00 End Date: 24-06-2010 14:00 Parking: Cassa di Risparmio Street Car Type: manual Model: medium Partecipant: 4	Start Date: 23-06-2010 8:00 End Date: 24-06-2010 14:00 Parking: Del Ronco Street Car Type: manual Model: small Partecipant: 2

Your Age:

☐ 18 ☐ 19 ☐ 20 ☐ 21 ☐ 22 ☐ 23 ☐ 24 ☐ 25 ☐ >26

Your mobile phone is:

☐ Classic ☐ Touch Screen

Computer System Usability

	strongly disagree	disagree	neutral	agree	strongly agree
Overall, I am satisfied with how easy it is to use this system	☐	☐	☐	☐	☐
It was simple to use this system	☐	☐	☐	☐	☐
I feel comfortable using this system	☐	☐	☐	☐	☐
It was easy to learn to use this system	☐	☐	☐	☐	☐
The system gives error messages that clearly tell me how to fix problems	☐	☐	☐	☐	☐
Whenever I make a mistake using the system, I recover easily and quickly	☐	☐	☐	☐	☐
It is easy to find the information I needed	☐	☐	☐	☐	☐
The information is effective in helping me complete the tasks and scenarios	☐	☐	☐	☐	☐
The organization of information on the system screens is clear	☐	☐	☐	☐	☐
The interface of this system is pleasant	☐	☐	☐	☐	☐

I like using the interface of this system	☐	☐	☐	☐	☐
This system has all the functions and capabilities I expect it to have	☐	☐	☐	☐	☐
Overall, I am satisfied with this system	☐	☐	☐	☐	☐

Most positive aspects (if any): _____

Most negative aspects (if any): _____

Please consider the two systems that you have used and provide us some comparison evaluation.

Conclusions:

Which system do you prefer?	☐ Syst. 1	☐ Syst. 2	☐ No pref.
Which system is the best one?	☐ Syst. 1	☐ Syst. 2	☐ No pref.
Which system is more useful?	☐ Syst. 1	☐ Syst. 2	☐ No pref.
Which system is easier to use?	☐ Syst. 1	☐ Syst. 2	☐ No pref.
Have you noticed differences between the two systems?	☐ Yes	☐ No	

My name is Thomas Schievenin and I am a computer science studend. This survey allows me to evaluate the system I have developed for my thesis project. The obtained results will show whether my system is useful or not.

References

[1] Itbusiness, "itBusiness.ca", "http://www.itbusiness.ca/it/client/en/Tech-Government/News.asp?id=45753", (Accessed July 6, 2010)

[2] Avis, "Avis Autonoleggio", "http://www.avisautonoleggio.it", (Accessed June 20, 2010)

[3] EuropCar, "EuropCar", "http://www.europcar.it", (Accessed June 20, 2010)

[4] Avis, "GesioneFlotta", "http://www.gestioneflotta.com", (Accessed June 20, 2010)

[5] Francesco Ricci, "Introduction to Internet and WWW", "http://www.inf.unibz.it/~ricci/IT/slides/1-www.pdf", (Accessed June 20, 2010)

[6] Cynthia Dwork, Ravi Kumar, Moni Naor, D. Sivakumar, "Rank Aggregation Methods for the Web", Proceedings of the 10th International Conference on the World Wide Web (WWW 2001), Hong Kong, pp. 613-622, 2001.

[7] Sayed Y. Hashimi and Satya Komatineni, Pro Andoid, Apress 2009

[8] Jason Brittain, Ian F. Darwin, Tomcat: The Definitive Guide, O'Reilly Media June 2003

[9] Peter Eisentraut, Bernd Helmle, PostgreSQL-Administration, O'Reilly October 30, 2008)

[10] Stefan Edlich, Henrik Hörning, Reidar Hörning, Jim Paterson, The Definitive Guide to db4o, Apress, 1 edition (June 15, 2006)

[11] Google app Engine, http://gaesql.appspot.com/, (Accessed June 20, 2010)

[12] Sperical Law of Cosines, http://www.movable-type.co.uk/scripts/latlong.html , (accessed June 20, 2010)

[13] [Brehob, 2001] Brehob, K., et al. "Usability Glossary", "http://www.usabilityfirst.com", 2001 (Accessed June 25, 2010)

[14] Liz Atwater, Letan Babaria, "Controlled Experiments", "http://otal.umd.edu/hci-rm/cntlexp.html", (Accessed June 25, 2010)

Dedicated to my dad.