Självständigt arbete på grundnivå

Independent degree project - first cycle

Datateknik
Computer Engineering

Trip planner
A web application for planning carpooling between Mid Sweden University's campuses

Ali Omran

Mittuniversitetet
MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

MID SWEDEN UNIVERSITY
Department of Information Systems and Technology (IST)

Examiner: Ulf Jennehag, ulf.jennehag@miun.se
Supervisor: Patrik Österberg, patrik.osterberg@miun.se
Author: Ali Omran, alom1200@student.miun.se
Degree programme: Master of Science in Engineering – Computer Engineering, 300 credits
Main field of study: Computer Engineering
Semester, year: Spring, 2018

Abstract

Mid Sweden University staff has several different meetings around the year. Most meetings take place in Mid Sweden University's various campuses (Sundsvall and Östersund). In order for the staff to be able to participate in these meetings, rental-cars are rented from different rental companies. The rental cars are used to transport the staff between the university's campuses. The trips are mostly made by one or two people per car which can lead to unnecessary extra costs. The goal of this project has been to provide a solution to this problem. The solution was in the form of a web application which the staff can use to register and join carpool trips. Since there are many carpool applications out there, the usability of the web application will be compared to the usability of a similar carpool application called GoMore to show if the developed application has any advantages in usability over similar applications. This was done by doing user-tests with 10 users followed by a questionnaire for each application. The usability metrics that were evaluated and compared for both applications were the following: effectiveness, time based efficiency and the satisfaction of the user. The results showed a slightly better result in effectiveness and time based efficiency for the developed application. The results also showed a greater satisfaction for users who tested GoMore. Although the developed web application did not show a significant difference in usability over GoMore, the developed application is more specific for the university's problem and can be developed even further to suit the university's needs.

Keywords: MIUN, carpooling, usability, web-application.

Table of Contents

Terminology

Acronyms

CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
XAMPP	X Apache + MariaDB + PHP + Perl
SUS	System Usability Scale

1 Introduction

Mid Sweden University staff has several different meetings around the year. Most meetings take place in Mid Sweden University's various campuses (Sundsvall and Östersund). In order for the staff to be able to participate in these meetings, rental-cars are rented from different rental companies. The rental cars are used to transport the staff between the university's campuses. The trips are mostly made by one or two people per car which can lead to unnecessary extra costs.

This project is based on the problem described above, where the goal is to develop a technical solution that will help the university's staff to plan the trips in a better way than how it is done in the current situation.

1.1 Background and problem motivation

As it stands today, the university uses rental cars to transport staff between the university's various campuses. The trips are made by one or two people per car which is both costly and has a negative effect on the environment since several rental cars can drive towards the same destination during the same time period.

The university currently has no system for planning these trips to get as many travelers per car as possible, instead the trips are made by one or two staff per car traveling to the same destination. The current system does not account for trips where there are multiple stops between the starting point and the destination.

1.2 Overall aim

Since Mid Sweden University has no system for planning trips between the various campus locations for attending meetings, the purpose of the project is to implement a technical solution that will help the university's staff to plan these meetings. The solution should be in the form of a web application where staff can register their trips and be able to search for trips at a specified time. Since the application is going to be used by the staff regularly, it is important to evaluate the applications usability by conducting user-tests with the end-users. Since there are many different applications for carpooling out there, a user-test will be done on a carpooling application called GoMore, the results of the test will be compared with the user-test on the developed application to see if there are any improvements in the usability of the application.

1.3 Scope

The developed application will be a prototype that wont be integrated with Mid Sweden university's current employees platform. Instead, the application will be

developed on a local server. The reason for this is that the security for the application has to be investigated further for the application to be a final product that can be used by the staff.

1.4 Concrete and verifiable goals

The goal with this project is:

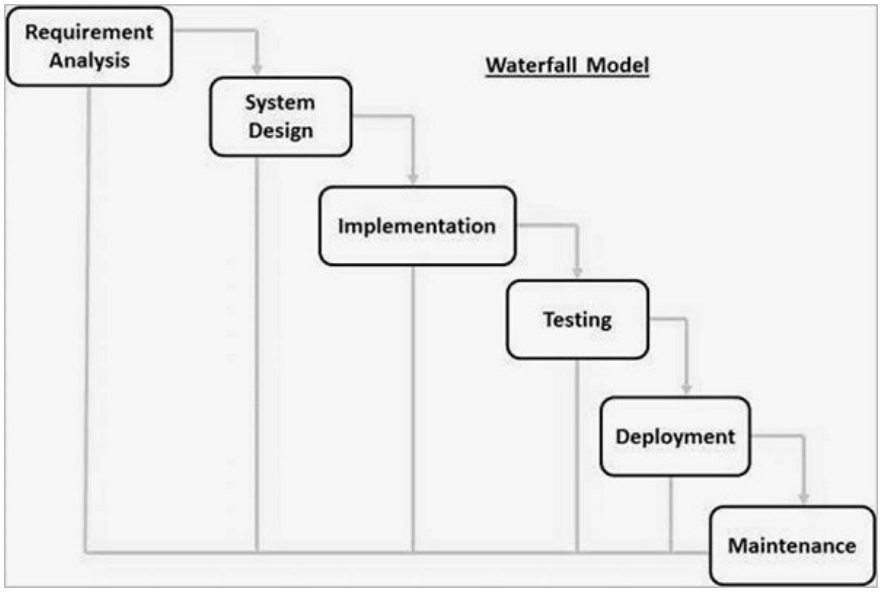
- Gather all the requirements for the application.
- Develop a prototype for a web application that meets the requirements of the requirements specification.
- Evaluate the usability of the prototype by conducting user-test with the end-users.
- Evaluate the usability of GoMore and compare the results with the user-test on the prototype.

2 Theory

This chapter contains necessary information about all the tools and methods that were used during the project.

2.1 Waterfall model

The water fall method is one of the most used developing method that are used in large projects. The method is used to organize the development process in a systematic matter. It consists of five different stages that are for planing, designing, implementing and deploying the application. (see fig. 1) The different stages are described in further details below [1][2].



Figur 1: Waterfall Model.

2.1.1 Requirement Analysis/Requirement gathering

This is the first stage of the development process. During this stage, the gathering of all necessary information and requirements for the application takes place. The requirements that are gathered during this process are often documented to be used as a guideline during the whole development of the application [1][2].

2.1.2 System design/Design specification

During this stage, a design for the system's architecture is presented based on the information and requirements that were gathered during the requirement analysis stage. This includes choosing the necessary tools and requirements that are required to develop the application [1][2].

2.1.3 Implementation

This stage is where the coding of the application takes place. The application is developed according to the requirements gathered and the design specification that was formed during the previous stages [1][2].

2.1.4 Test

This stage is meant for testing the application to attempt to find any faults or failures. If no faults or failures occurred during the testing, the project goes to the next stage which is the deployment and maintenance stage [1][2].

2.1.5 Deployment and maintenance

This is the last stage of the development process. The application is deployed to the customer. Further maintenance might be necessary if any issues occurs in the future. The maintenance could also be some minor improvements/enhancements to the current system [1][2].

2.2 PHP

PHP is an open source program language that is mainly used for the development of web applications where it is used as a server-side programming language. This means that the code executes on the web server Unlike JavaScript where the code is executed on the user's client[3].

PHP supports a wide range of database managers such as MYSQL, MongoDB, ODBC (which is an API independent of which database manager is used), etc[3].

2.3 JavaScript

JavaScript is a scripting language that is mainly used for client-side development of web applications. The program language is mainly used for the development of dynamic web pages due to the added possibility to manipulate HTML-documents. JavaScript could also be used as a server-side language with Node.js environment [4].

2.4 MYSQL

MYSQL is a open source SQL database manager that is used to store data. The data in a MYSQL database are stored in a table. These tables can be accessed and altered using SQL-statements [5].

2.5 Apache

Apache is an open source web server platform. Apache is the most used used platform out there for hosting websites [6].

2.6 XAMPP

XAMPP is a open source web server software which is used to host an Apache server together with a SQL database with ease [7].

2.7 Google's Oauth2-API

Google APIs uses the Oauth 2.0 protocol to authenticate users. The Outh 2.0 protocol is an authorization framework that allows developers to authenticate users to grant them access to the developers application.

This API allows developers to use all Google APIs that requires the user to login with their Google account, such as the Gmail-API. The Oauth2-API is available for the following programming languages: Java, .NET, Node.js, PHP, javaScript, Python and Ruby [8].

2.8 Gmail-API

Gmail-API is google's API to handle the users Email. This includes functionality such as read and send messages, search in messages, etc. To use this API, the user has to authenticate through Google's Oauth2-API [8][9].

2.9 Usability

Usability is an attribute used in developing interactive applications which describes how easy an application is to learn, how effective it is and how satisfied the users feel after using the application. Usability has the following goals that a developer should strive for when developing an application [10]:

- Learnability: How easy an application is to learn for first-time users.
- Efficiency: How efficient an application is to use, i.e. how quickly a user can perform tasks while using the application.
- Effectiveness: How well a new user can perform tasks without failing.

- Memorability: How well the users remember to use the application after some time after they used it for the first time.

- Satisfaction: How satisfied the users feel overall after using the application.

2.9.1 System Usability Scale

SUS (System Usability Scale) is a way to measure the overall satisfaction of the user after using an application. This method uses a questionnaire together with a scoring formula to give a score to the systems usability. The questionnaire consists of 10 statements (The statements were gathered from the template for the questionnaire):

1. ”I think that I would like to use this system frequently.”
2. ”I found the system unnecessarily complex.”
3. ”I thought the system was easy to use.”
4. ”I think that I would need the support of a technical person to be able to use this system.”
5. ”I found the various functions in this system were well integrated.”
6. ”I thought there was too much inconsistency in this system.”
7. ”I would imagine that most people would learn to use this system very quickly.”
8. ”I found the system very cumbersome to use.”
9. ”I felt very confident using the system.”
10. ”I needed to learn a lot of things before I could get going with this system.”

The user answers each statement with a number that ranges from 1 to 5, where 1 indicates that they stronly disagree and 5 indicates that they strongly agree. Each question with an odd number is a positive statement e.g, question 1, question 3, question 5, etc. Each question with an even number is a negative statement. The following steps needs to be followed to calculate the SUS-score [13]:

1. For every possitive statement, substract 1 from the value the user has entered

2. For every negative statement, substract 5 by the value he user has entered.

3. Add all the values of step 1 and 2 to get the total score. This score is then multiplied by 2.5. This score is a range between 0-100.

2.10 GoMore

GoMore is an application for leasing and renting cars and also for planing car-pool trips in Sweden. The application is available on both IOS and Android as a mobile application and also in the form of a web application that can be accessed through any web browser.

3 Methodology

This chapter describes how the different tools and methods were used during the project.

3.1 Work process

This sub-chapter describes how the waterfall model was used during the project.

3.1.1 Requirement gathering

In a meeting with Stefan Eriksson, who represented the client (Mid Sweden University), the problem was discussed as well as the various functions the web application should contain. To get a clearer picture of the various requirements and functionality the application should include, a requirement specification document (see Appendix A) was produced which was done together with Stefan Eriksson.

3.1.2 Design specification

After the requirement gathering stage, a design specification was created. This included the layout of the web application, choice of the tools and also the programming language that was going to be used during the development of the application. (For further details about the choice of tools and languages, see chapter 3.2)

3.1.3 Implementation

The coding of the application occurred during this stage. See chapter 4 for further details about the construction of the application.

3.1.4 Testing

This is the stage where the application is tested. The test checks whether if all the requirements for the application have been fulfilled. The application will also have a user-test. The same user-test will be done on GoMore to see if the developed applications has any improvements in the usability in comparison to GoMore.

3.2 Tools

This chapter presents all the tools that were used and how they were used during the project.

3.2.1 Programming languages

The markup language HTML together with CSS and JavaScript together with the framework Jquery was used to construct the web application. Javascript was used primarily for implementing the Google Oauth2 API. All server-side programming was done using the program language PHP.

The web application was developed locally on an Apache server using the XAMPP tool.

3.3 Evaluation

This chapter describes how the application will be evaluated after the development is done.

3.3.1 Usability tests

In order to evaluate the usability of the application, it is necessary to perform a usability test. The usability tests will evaluate the metrics mentioned in the subchapters bellow.

3.3.1.1 Effectiveness

This metric measures the accuracy of how well a task can be done by a user that uses the application for the first time. This metric can be calculated by a simple percentage formula [11]:

$$Effectiveness = \frac{\text{Number of tasks completed successfully}}{\text{Total number of tasks undertaken}} \times 100\%$$

figure 2: percentage formula for calculating effectiveness [11].

This formula will be used to measure the effectiveness of both applications. The results will be compared.

3.3.1.2 Time based efficiency

This metric is the measured time of how fast a user can complete a set of tasks. This metric can be calculated using the following formula [11]:

$$Time\ Based\ Efficiency = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

Figure 3: Formula for time based efficiency [11]. N = number of tasks, R = number of users, n_{ij} = 1 if task is completed, 0 if it was not completed, t_{ij} = time it took for user to complete a task [11].

This formula will be used to calculate the Time based efficiency of both applications to compare the results.

3.3.1.3 Satisfaction

This metric measures how satisfied the user feels after using the application. This metric can be calculated using a questionnaire revolving how difficult the tasks were and also how satisfied they were after using the application. The questionnaire that is going to be used in this project is the SUS questionnaire. (see chapter 2.9.1)

3.3.2 Tasks

The usability test was carried out together with 10 test users. Each test user was assigned two different roles, either passenger or driver. Each user was assigned a different set of tasks depending on their role.

Tasks for user with the driver role:

- Register a trip in the ”Register trip”- page.
- Change the date and time on a trip that the user is registered as a driver.
- Send a message to all fellow passengers registered to the same trip.
- Delete a trip.

Tasks for user with the passenger role:

- Search for a trip with a specific time, date and destination.
- Join a trip as a passenger.
- Send a message to all fellow passengers registered to the same trip.
- Leave a trip from the ”My trips”-page.

3.3.3 GoMore

The same evaluation mentioned in chapter 3.3.1 will be done on GoMore with the same tasks mentioned in 3.3.2.

4 Implementation

4.1 The architecture of the application

The application consists of four different components that communicate with each other in different ways. These are: client, Apache server, MYSQL database and Google's Oauth2 API.

The client communicates with the server when a trip is booked or changed. The client also communicates with google's Oauth2 API to authenticate the users and also to retrieve necessary information about the user which is then stored to the database of the application. The server is linked to a MYSQL database that stores all information about trips and users.

To be able to use Googles Oauth2 API and other google APIs a project needs to be created on googles cloud platform. The project will be assigned with a unique client ID which will be used with every request that are made from the client. The client ID is used to make sure that all the requests are made from the application and the application only. This is done by specifying the source URL that is allowed to make any request to Googles API. Since the application was running on a local Apache server, ”http://localhost/” and ”http://127.0.0.1/” was used. (See fig. 2)

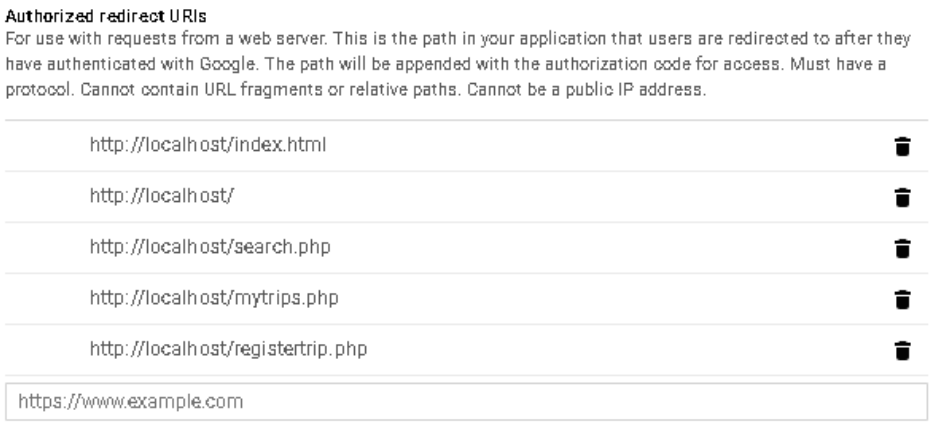


figure 2: Redirect URLs for the application.

4.2 Database

A database was constructed on the local Apache server. This database consists of necessary data about the users, trips and a ”Whats new”-board. The database

was therefore constructed with 4 tables as shown below (Further information about the tables can be seen in fig. 3):

- Users: Contains information about the user that consists of the e-mail address of the user and the users name. The data about the user is retrieved from Googles server after the authentication with Oauth2 has occurred.

- Trips: This table contains information about the trip such as trip ID, seats available, arrival time, departure time, destination and optional information.

- Passengers: This table contains further information about the trips. The passengers table is linked to the ”Trips”-table using the Trip-ID as a foreign key.

- News: This table contains data such as the date and time which the update happened, the ID and date of the user who has been affected by the update , the ID of the affected trip and a message of the update.

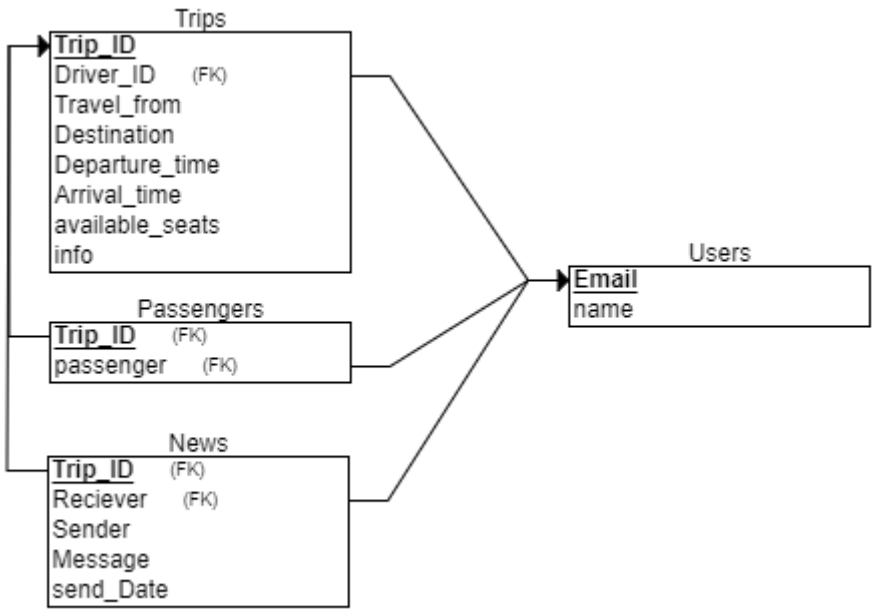


Figure 3: Relational schema for the database

4.3 User interface

The application is divided into 4 pages; main page, Register trip, search trip and

4.3.1 Login

The first thing the user sees when they open the application is the login window. (See fig. 4)

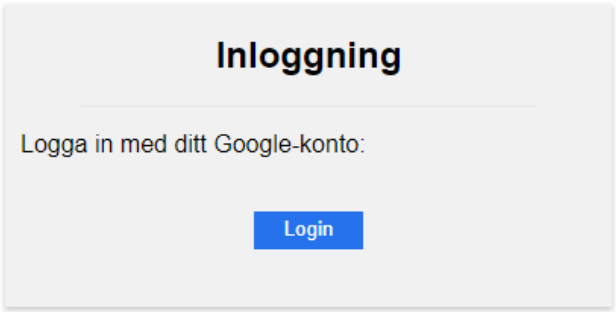


figure 4: Login window.

Login to the application is done with the help of Google Oauth2 API. When the user clicks on "login", the user is sent the Google login portal. (see Fig. 4)

A HTTP request is sent to googles authentication servers together with the client ID. The server checks whether the current URL on the clients browser is included in the authorized URLs list for the specified client ID. If it is, the user is redirected to googles sign in window. (see figure 4) Otherwise, a error message shows up telling the user that the current URL cannot be used to authenticate with Oauth2.

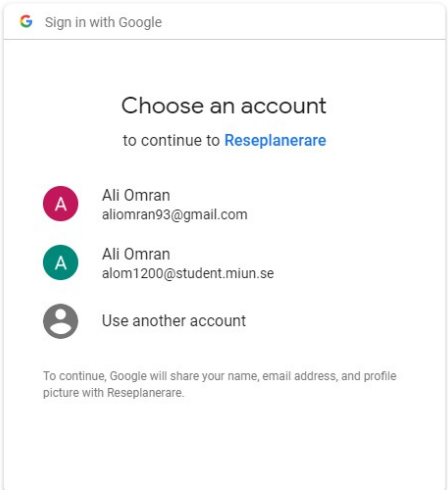


Figure 5: Login with Gmail.

When the user has logged in with their Google account, they are forwarded to a window where the user confirms that the application may have access to the user's email messages and to be able to send messages via the application. (see Fig. 5)

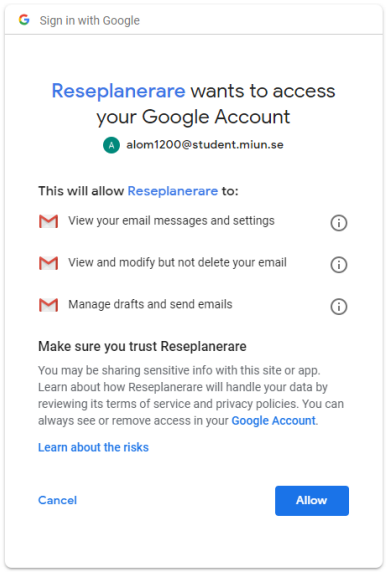


figure 6: Request to access the users Email.

After the login has occurred, it is checked against the database if the user has logged in earlier with the same email address. If the user has not logged in before, a new user is created and stored in the database.

To further verify the authenticity of the user, Googles Oauth2 API creates a token that contains the users unique token-ID, the period for which the token is valid, the client-ID of the application that has been used to authenticate with Oauth2 API and the users e-mail address. The token-ID is sent to the server of the application. The applications server proceeds to make a HTTP-request to Googles Oauth2 API using the token-ID. The Oauth2 API returns a JSON-object containing the information about the user with the corresponding token-ID that was used. The client-ID that is included in the JSON-object is compared with the token-ID that is stored on the server. The same comparison happens for the e-mail address that the user used to login to the application. If the strings does not match, the application displays an error window. If the strings matches, a PHP-session is started. The session stores information such as the time of login, the users e-mail and the name associated with the e-mail. The user is then redirected to the main page. (See Appendix B which contains the whole code for this process)

The verification of the token is done in both the client and the server of the application.

This whole process is done to make sure that an intruder wouldn't be able to use an expired token-ID or a token-ID that is meant for someone else and they would have access to data they shouldn't have access to.

4.3.2 Main page

When the user has successful been logged in, they are redirected to the main page. The navigation bar is shown at the top of the window. (The navigation bar is shown in every page in the application) This page is also where the new's about a trip is displayed. The ”Whats new” section shows all the news about the trip that the user is registered to as either driver or passenger along with the corresponding date, time and destination of the trip. The news are displayed up until the date of the trip, or up until the trip has been removed by the user. (see fig 7)

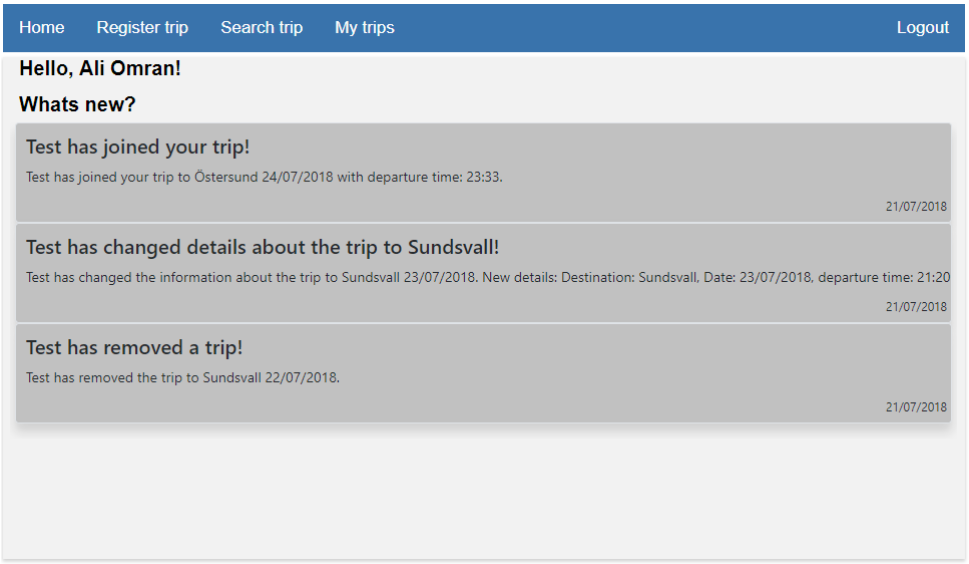


Figure 7: Main page with “Whats new?”-panel.

4.3.3 Register Trip

This is where the user can register a trip. The user chooses the destination, the date of the trip, departure time, expected time to arrive to the destination, the amount of available seats and an optional field where the user can add more information about the trip. If the user leaves the destination field or the date field empty, a error message shows up telling the user that the fields cannot be empty. (see figure)

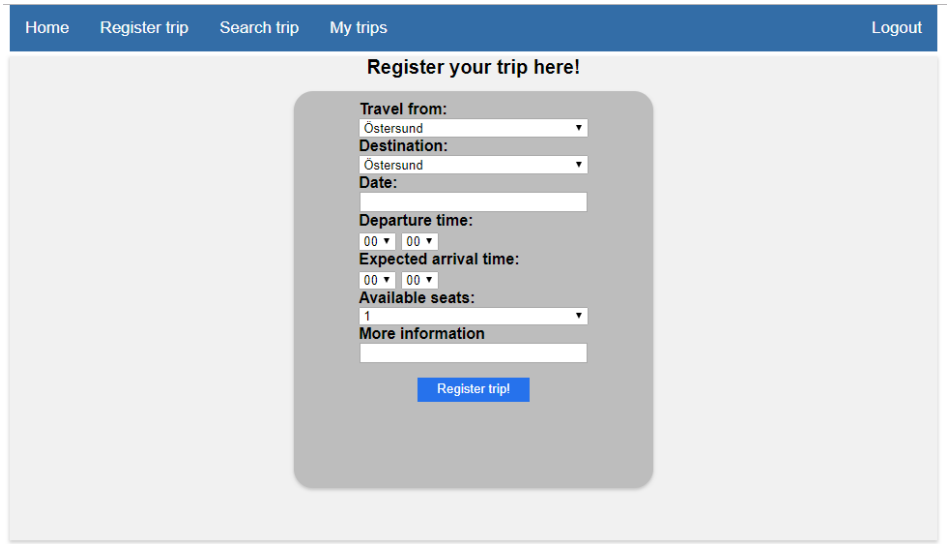


Figure 8: Page for registering a trip.

4.3.4 Search Trip

In this page the user can search for a trip and also register to a trip as a passenger. The user enters the destination, date and departure time for the trip they are looking for. (see figure) When the user submits the form, a HTTP-post request is sent to the server with a SQL expression to return data with the given time, date and destination. The server returns an array that contains JSON-objects with information on available trips (trips with available seats). The JSON-array is then displayed on the web application. If there are no trips with the specified time, the server will return a JSON-array with all the trips that are booked for the same day as the user has specified. If there were no trips found on the date specified by the user, a message shows up telling the user that there were no trips found.

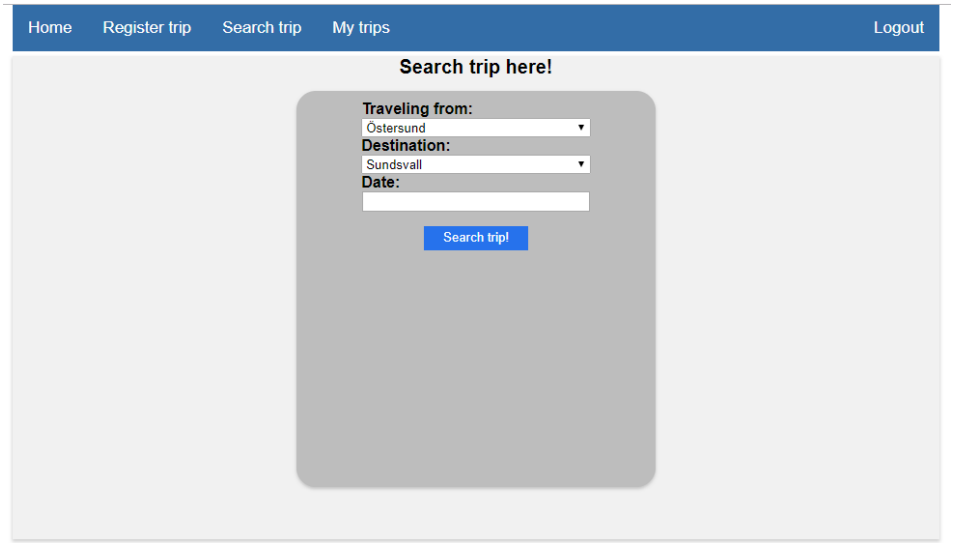


Figure 9: Page for searching a trip.

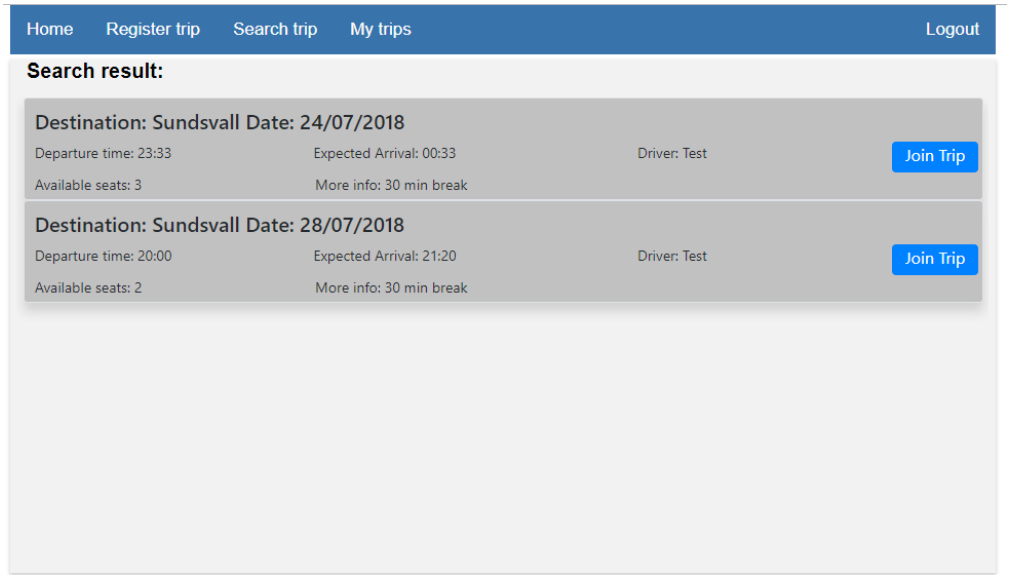


Figure 10: Result of search query.

4.3.5 My trips

This is where all the trips the user has registered to as either passenger or driver. From this page, the user can change the details revolving the trip such as the date and expected arrival time. The user can also remove the trip by pressing the ”Delete” button. (see figure)

Any actions taken in this page whether its changing the time of the trip, changing the date, or removing the trip, a message box will appear informing the user that an E-mail will be sent from the Gmail-account they are logged in to. The content of the message that will be shown depends on the action taken. (see figure)

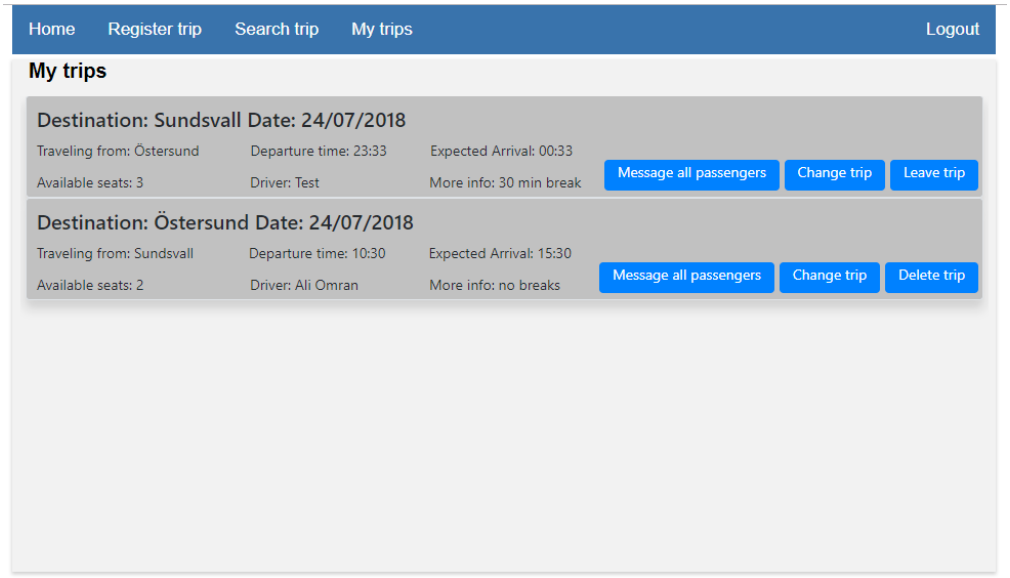


Figure 11: “My trips”-page.

5 Results

The user-tests were performed with 10 test-users. The test-users who were picked were from different ages, genders and educational backgrounds (there were no specific target group for the tests)

The users were told to do each specific task depending on the roles described in chapter 3.3.2. As soon as the test-users starts, the observer takes notes of the time it took to finish a task. The data from the observations were gathered and used in the formulas described in chapter 3.3. When the users were done testing the application, they were given a SUS-questionnaire (see chapter 2.9.1) that they answered.

5.1 Result of usability test on the prototype

5.1.1 Effectiveness

The result of the user-tests for the prototype shows that all the 4 tasks were completed successfully by the users for both the drivers and the passengers.

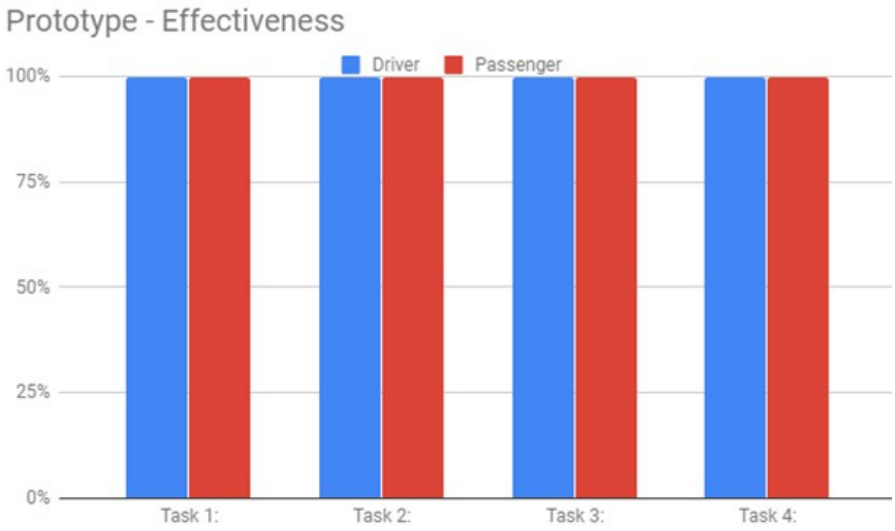


Figure 12: Precentage of test-users that completed each task. (Prototype)

5.1.2 Time based efficiency

The time to complete each task for both the driver and passenger groups for the prototype was gathered and documented by the observer. The result can be seen in Figure 13 and 14. The time based efficiency for the whole application is approximetly 0.027 tasks/sec.

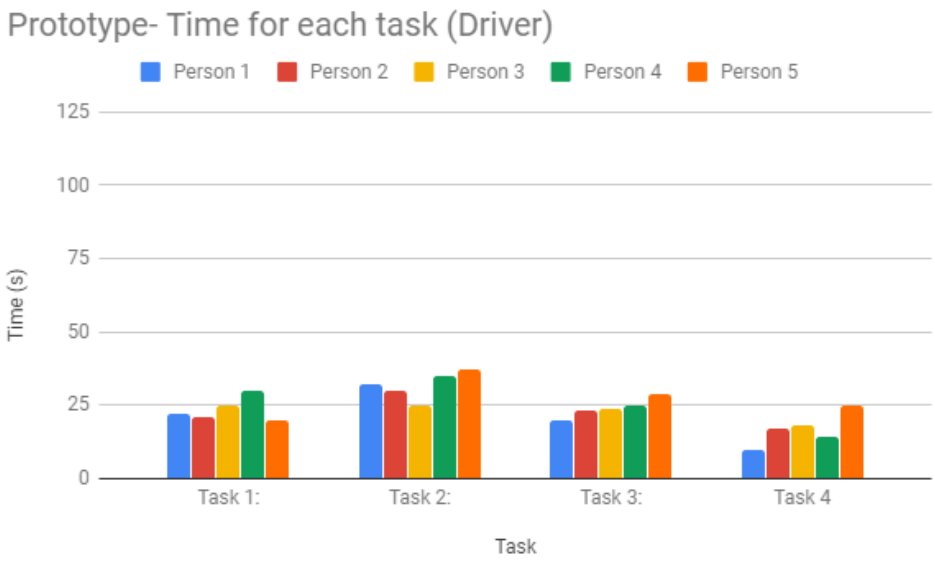


Figure 13: Time to complete each task as a driver. (Prototype)

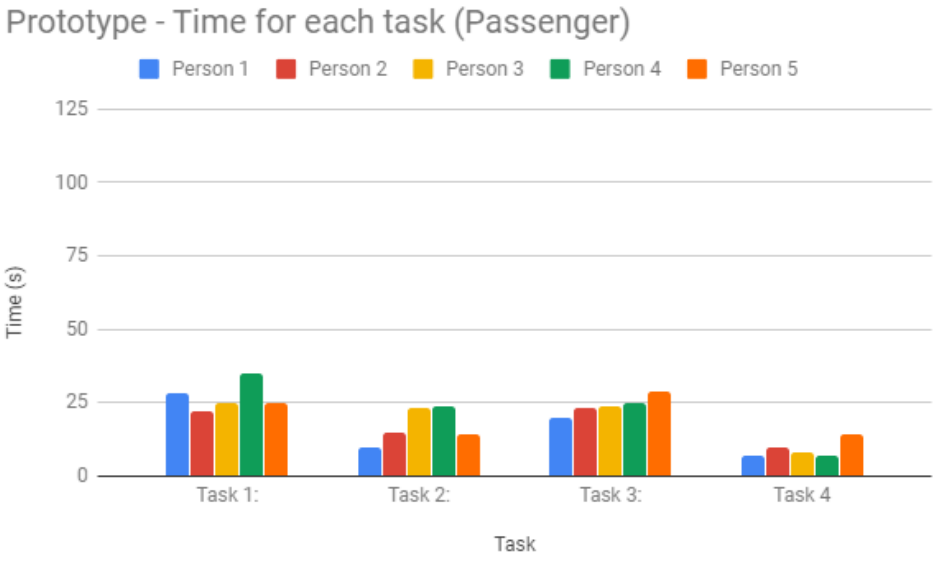


figure 14: Time to complete each task as a passenger (Prototype)

5.1.3 SUS-Score

The SUS score for the prototype is 77.75. (See Appendix D to see what each user answered on each question on the questionnaire)

5.2 Result of usability test on GoMore

5.2.1 Effectiveness

The result of the user-tests on the GoMore-application shows that task 1 (”Register a trip in the ”Register trip”- page”) was not completed succesfully by one user. It also shows that 5 out of 10 users were able to complete task 3.

Figure:

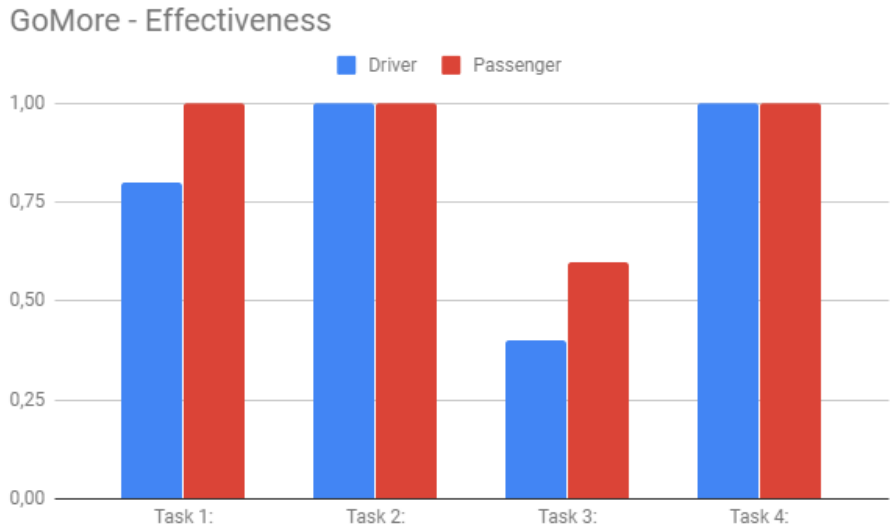


figure 13: Precentage of test-users that completed each task. (GoMore)

5.2.2 Time based efficiency

The time to complete each task for both the driver and passenger groups for the GoMore-application was gathered and documented by the observer. The result can be seen in Figure 15 and 16. The time based efficieny for the whole application is 0.0156 tasks/sec.

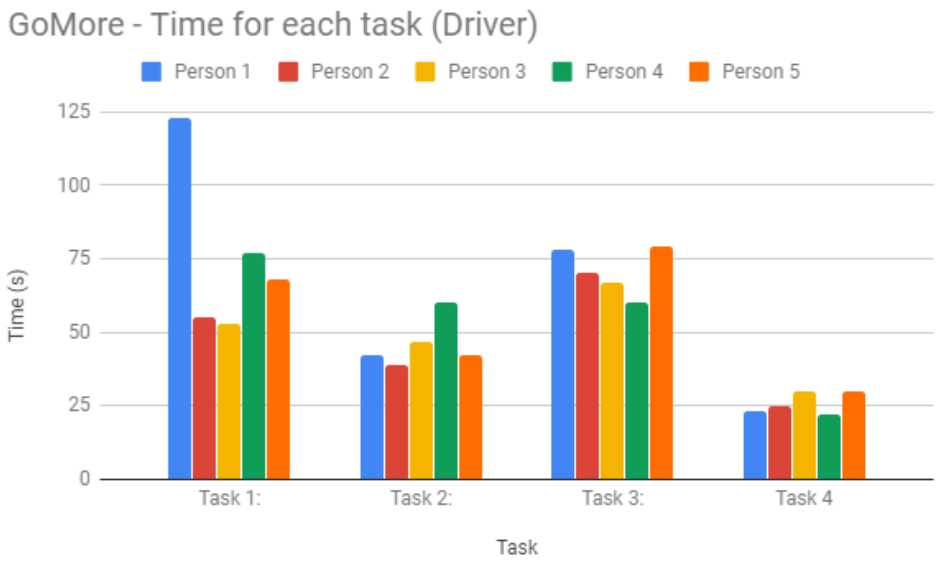


Figure 14: Time to complete each task as a driver. (GoMore)

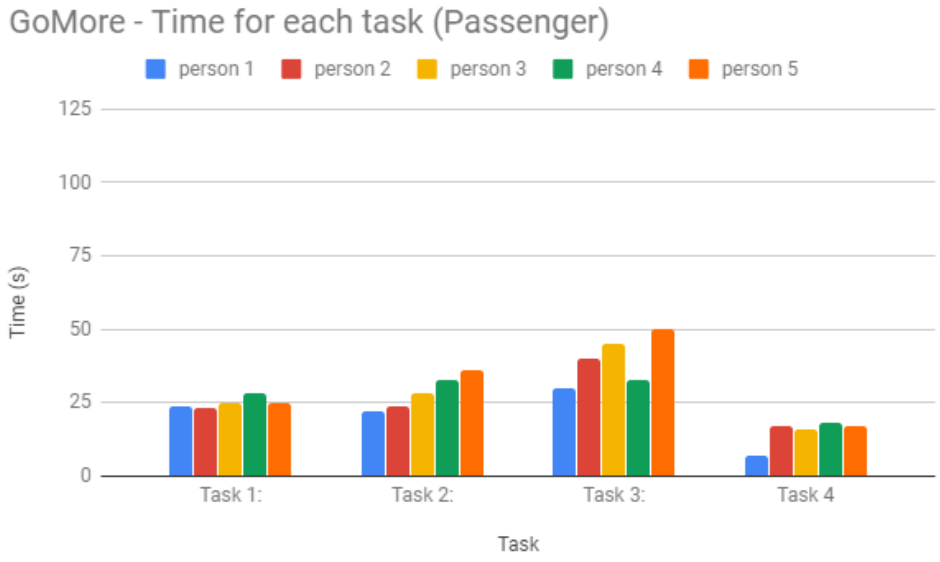


Figure 15: Time to complete each task as a passenger (GoMore)

5.2.3 SUS-Score

The SUS score for the prototype is 84. (See Appendix C to see what each user answered on each question on the questionnaire)

6 Conclusion

6.1 Evaluation of the methodology

6.1.1 Work process

During this project, the development has been done following the waterfall model which is a sequential development method. Since the evaluation of the application has been revolved around the usability of the application, an itterative design approach to involve the user with every step of the development would have been a more appropriate for this kind of project. This could have lead to a better result in the usability-test of the application and an improved satisfaction overall when using the application than what it currently is.

6.1.2 Implementation

One of the requirements of the application was that it should be able to notify the user when a change to a trip has occured. The client was unsure on how they wanted it to be implemented. A sulotion I suggested was that they could use google's Gmail-API and send the notifications through an Email since all the staff uses the university's Gmail on a daily basis. This formed the whole basis of the application since the authentication and authurization for the application is done through Google's Oauth2 API whichi is needed to use the Gmail API. All the data of the trips are stored in the database together with data of the user which is gathered from Google.

6.1.3 User-tests

An improvement to the user-test would be to do the test with more test-users. Since the time based efficiency was calculated using the mean of the time, it is harder to identify outliers when the set of data (time for each user) is too small. This can be solved by having a larger set of users for the tests.

6.2 Evaluation of the result

The results showed that the developed application had a slightly better results in the effectiveness of the application where all the users were able to complete the tasks for the prototype in comparison to GoMore where all the users were able to complete all tasks except task 3. Only 5 of the 10 users were able to complete task 3. This doesn't necessarly mean that the prototype have a better effectiveness as GoMore has more features to offer such as leasing and hireing cars. This means that the navigation bar of GoMore has more elements in it which might lead to the user having difficulty to find a specific page they are looking for when using the navigation bar. The prototype is a very simple design with only one feature (carpooling).

As for the time based efficency, the result showed that the prototype had a higher value of goals per second than GoMore. The reason for this is that some users struguled to send messages to all the fellow passengers (Task 3) as GoMore did not have a function to send to all passengers with one click, instead the user have to send a message to a single passenger at a time. This lead to the user being confused as they expected a ”Send to all”-button.

GoMore showed a better result in satisfaction which was expected since the developed application is just a prototype and should be worked on further before it is ready for deployment, while GoMore is a finished product that is ready to be used.

Overall, the prototype didn't have significant advantages over GoMore. However, the prototype is more specific for the university's needs and can eventually be developed further to suit the university's needs even more.

6.3 Ethical aspects

This project's goal is to provide Mid Sweden university with a solution to a problem they have with planning trips between campuses in an efficient way. According to the client, there have been many instances where there would be only one or two staff-members traveling with one car even if there are multiple of people who have planned to travel to the same destination on the same day. This leads to a negative effect on the environment. The university can avoid this problem by using the solution provided in this project and hopefully contribute to a society with less environmental pollution.

References

[1] Tutorialspoint, ”SDLC - Waterfall Model”, https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm Retrieved: 2018-07-30.

[2] Airbrake.io, “Waterfall Model: What Is It and When Should You Use It?”, https://airbrake.io/blog/sdlc/waterfall-model Retrieved: 2018-07-30

[3] PHP, “What is PHP?”, http://php.net/manual/en/intro-whatis.php Retrieved: 2018-07-30

[4] JavaScript, “JavaScript”, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics Retrieved: 2018-07-30

[5] MySQLTutorial, “What Is MySQL and Why It Is the World’s Most Popular Open Source Database”, http://www.mysqltutorial.org/what-is-mysql/ Retrieved: 2018-07-30

[6] Apache, “The Number One HTTP Server On The Internet”, https://httpd.apache.org/ Retrieved: 2018-07-31

[7] XAMPP, “What is XAMPP?”, https://www.apachefriends.org/index.html Retrieved: 2018-07-31

[8] Google, “Using OAuth 2.0 to Access Google APIs”, https://developers.google.com/identity/protocols/OAuth2 Retrieved: 2018-08-02

[9] Google, “Flexible, RESTful access to Gmail features”, https://developers.google.com/gmail/api/ Retrieved:2018-08-02

[10] Nilsen Norman Group, ”Usability 101: Introduction to Usability”, https://www.nngroup.com/articles/usability-101-introduction-to-usability/
Retrieved: 2018-08-02

[11] UsabilityGeek, ”Usability Metrics – A Guide To Quantify The Usability Of Any System”, https://usabilitygeek.com/usability-metrics-a-guide-to-quantify-system-usability/
Retrieved: 2018-08-03

[12] GoMore, ”Om GoMore”, https://gomore.se/about
Retrieved: 2018-08-10

[13] UsabilityGeek, ”How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website”, https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/
Retrieved: 2018-08-10

Appendix A: Requirement specification

The application should have:

- A portal where the user can register trips
- A page where the user can search for a trip at a specific date and time.
- A page where the user can register as a passenger.
- A notification system for when a trip has been changed, removed or if there is a new passenger who have just joined.
- Possibility to communicate with all passengers connected to a trip.

Other requirements

Development of the application should be in PHP and MYSQL.

Appendix B: login.php

```
<?php


$client_id =
'130780657097-
7unbkhk389qd8m1j71eaddj5jb5bv2f9.apps.googleusercontent.com';

$url      =      'https://www.googleapis.com/oauth2/v3/tokeninfo?
id_token=' . urlencode($_POST["token_id"]);
$curl_handle = curl_init();
curl_setopt($curl_handle, CURLOPT_URL, $url);
curl_setopt($curl_handle, CURLOPT_CONNECTTIMEOUT, 2);
curl_setopt($curl_handle, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl_handle, CURLOPT_USERAGENT, 'Reseplanerare');
$query = curl_exec($curl_handle);
curl_close($curl_handle);

$obj = json_decode($query);

//if the json contains error code or if the client ID doesn't
match the one returned in the json, display failed to login
page.
if (property_exists($obj, "error_description") || $obj->aud !=
$client_id || $_POST["email"] != $obj->email){

?>

<!DOCTYPE html>
<html>
  <head>
   <link rel="stylesheet" type="text/css" href="layout.css">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
       <script src=
"//ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js">
</script>

    <title>Reseplanerare</title>
    <meta charset='utf-8' />
  </head>
  <body>
        <div id= "login_container">
         <div id=login_header_div>
                <h2 id= "login_header">Inloggning</h2>
         </div>
          <div id= "login_message">
           <p>Fel vid Inloggning.</p>
          </div>
        <div id= "submit_container">
          <button id = "login_button">Försök igen</button>
        </div>
       </div>
```

```
    <script src = "authentication.js"></script>
        <script async defer src=
"https://apis.google.com/js/api.js"
        onload="onload=function(){};handleClientLoad()"
         onreadystatechange="if (this.readyState === 'complete')
onload()">
    </script>

  </body>
</html>

<?php

}else{

        session_start();
        $_SESSION["USER_EMAIL"] = $obj->email;
        $_SESSION["NAME"] = $obj->name;
        $_SESSION["LOGIN_TIME"] = time();
       $_SESSION["EXPIRE"]  =  $_SESSION["LOGIN_TIME"]  +  (30  *
60);


        include_once("dbTrip.php");
        $dbcon = new dbTrip();
        $dbcon->uppLoadUser($_SESSION["USER_EMAIL"],
            $_SESSION["NAME"]);

        header("location: /mainpage.php");

}
?>
```

Appendix C: Result of SUS test (GoMore)

Q1. ”I think that I would like to use this system frequently.”

Q2.”I found the system unnecessarily complex.”

Q3. ”I thought the system was easy to use.”

Q4. ”I think that I would need the support of a technical person to be able to use this system.”

Q5. ”I found the various functions in this system were well integrated.”

Q6. ”I thought there was too much inconsistency in this system.”

Q7. ”I would imagine that most people would learn to use this system very quickly.”

Q8. ”I found the system very cumbersome to use.”

Q9. ”I felt very confident using the system.”

Q10. ”I needed to learn a lot of things before I could get going with this system.”

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	4	1	2	1	4	5	5	4	4	1
2	3	2	4	1	4	2	5	1	5	1
3	4	1	5	1	3	1	5	1	4	1
4	4	1	5	1	4	1	5	1	5	1
5	3	1	5	1	3	1	4	1	4	1
6	5	1	5	1	5	1	4	1	4	1
7	2	1	5	1	3	1	5	1	4	1
8	4	2	5	1	2	1	4	1	4	1
9	5	2	5	1	5	2	4	1	4	1
10	8	1	5	1	4	3	5	1	4	1

Avarage score: 84

Appendix D: Result of SUS test (Prototype)

Q1. ”I think that I would like to use this system frequently.”

Q2. ”I found the system unnecessarily complex.”

Q3. ”I thought the system was easy to use.”

Q4. ”I think that I would need the support of a technical person to be able to use this system.”

Q5. ”I found the various functions in this system were well integrated.”

Q6. ”I thought there was too much inconsistency in this system.”

Q7. ”I would imagine that most people would learn to use this system very quickly.”

Q8. ”I found the system very cumbersome to use.”

Q9. ”I felt very confident using the system.”

Q10. ”I needed to learn a lot of things before I could get going with this system.”

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	3	1	4	1	4	4	5	2	5	1
2	3	1	4	1	4	4	5	3	5	1
3	2	1	4	1	5	5	5	1	5	1
4	3	1	5	1	4	4	5	2	5	1
5	3	1	4	1	5	3	5	3	5	1
6	4	1	5	1	4	4	5	2	5	1
7	3	1	4	1	4	4	5	3	5	1
8	4	1	5	1	5	4	5	2	5	1
9	3	1	4	1	4	4	5	2	5	1
10	3	1	5	1	4	4	5	1	5	1

Avarage score: 77.75