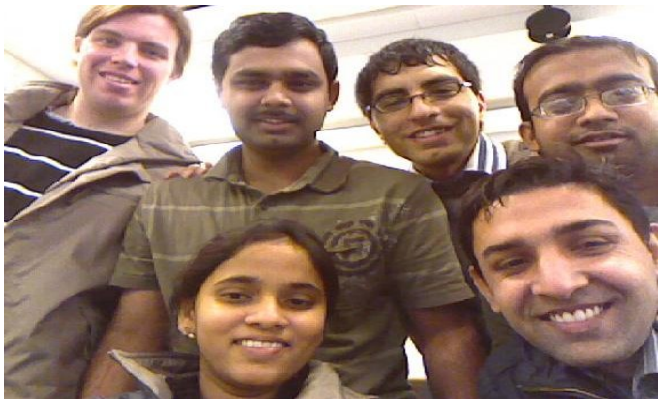
PATH

Public **A**dvice **T**ravelling **H**elp

Mostafa Pordel	mpl08001@student.mdh.se
Anand T A	atr08001@student.mdh.se
Abhishek Palod	apd08001@student.mdh.se
Nicholas Cockran	ncn08001@student.mdh.se
Beaulah Vineela P	bpi08001@student.mdh.se
Riasat Abbas	ras05001@student.mdh.se



University of Zagreb, Croatia

Faculty of Electrical Engineering and Computing

&

Malardalen University Vasteras, Sweden

School of Innovation, Design and Engineering

Executive Summary

Public Advice Travelling Help (PATH) is a planning system that helps travellers to get detailed information about any route or place. Users search for routes and places where they want to travel. PATH filters advices according to user requirements and shows them on the output map. PATH philosophy is to encourage support from users to users. Advices are provided by registered users according to the experiences they want to share with others.

The project team consisted of six members and the project was performed as a part of the Distributed Software Development course which is done in collaboration between Mälardalen University, Sweden and University of Zagreb, Croatia and which was performed during 10 weeks, with an assumption of intensive work (20 hours of work per student). The objective of the course was to work on software development, process orientation, team work, distributed development and to work with new open source technologies.

For the requirements we used SCORE project description, and a web survey. In the web survey we got the feedback from potential users of PATH. The targeted users were mostly students. According to the feedback and SCORE project description, we collected PATH requirements.

PATH design uses a Three Tier Architecture pattern. It includes a User Interface Layer (UIL), Business Object Layer (BOL) and a Data Access Layer (DAL). The implementation was based on the open source tools. We used MySQL, Eclipse, Apache server, PHP and Google map API. The main challenges in PATH were to successfully complete the work within the course duration, selecting the right Geographical Information System (GIS) technology and then the selected GIS API library's dependencies.

Initially we used the Rational Unified Process (RUP) model for software development but due to the short period of the project course and many uncertainties we decided the development process into agile development. After that, two long meetings were considered in each week and team members worked closely. As a result members cooperated more with each other, project requirements and project design became more clearly understood and the progress of the project improved significantly. Furthermore, all team members shared their knowledge and experiences comfortably, using the new methodology.

The main characteristic of PATH is enabling users to add some advices as well as advice types. Advice types could be a bus station, bar or any other user preference. The next important feature of PATH is managing conflicts of advice entries. Users can change existing advices several times. If changes of one advice occur more than the number that PATH Administration defines, advice conflict would happen. In a conflict scenario, managers have the permissions to decide the correct advice. The system users are promoted as managers of the system based on their proven track record. As a result PATH is self managed and it implemented a wiki management.

1. Introduction

The objective of the **PATH** project is to create a web application which helps travellers to get detailed information about any route or place. The following are the major system functionalities:

- Register user in PATH web site.
- Suggest the suitable advices for different routes or places according to user preferences.
- Store advices given by users.
- Store advice types entered by users.
- Manage advice entry conflicts.
- Allow Administrator to manage conflict parameters

1.1. Scope of the Project and Main Challenges

This project is performed as a part of "Distributed Software development**"** (DSD) course at Mälaradalen University, Västerås, Sweden in collaboration with University of Zagreb, Croatia. DSD course duration was 10 weeks. The project team consisted of six members with each member to contribute about 20 hours weakly according to course load.

The main challenges in PATH were time duration. According to DSD course we should deliver at milestones during the course duration. For PATH we should use a Geographical Information System (GIS) service. There are several GIS tools and MAP APIs and each of them has its own benefits and disadvantages. We had to select one of them and it was a bit challenging. After all we decided to use Google map API because of the good documentation we found on the web. The integration was difficult, as it took a long time for us to merge Google API calling method in JavaScript. Working on existing samples and solutions took longer time than providing newer solutions and methods.

This document is organised as follows. The next section describes the project management and communication. Section 3 includes requirement specification that is followed by PATH design and Implementation. In section 6 includes how PATH was tested and verified. Outcomes and what we learnt and finally the summary are the last parts of this document.

2. Requirements Specification

Our SCORE supervisor recommended to the team to find real potential customers of PATH. We designed a set of questions which were used to capture the requirements from the potential users of system.

Requirement survey was done with following set of questions:

- What details of an area do you think are important when planning a journey?
- What aspects of other route planning systems (e.g. Google maps) do you find most useful?
- What would you like to see in a route planning system?

- What is the single most important piece of advice you like to know while planning a journey?
- What features would be important for you to provide advice to another person about a route or location?
- Which methods of giving advice do you find easiest/best (Audio/Text/Video/Image)?
- How important is it to be able to personalize the system? Example Change settings which control which information is displayed. If possible give some examples of things you would like to be able to customize.

30 people who were interested in the topic and familiar with similar tools like Google map responded. Responses were collected and analysed. Figure 1 shows the result of the survey. We showed different group of requirement on different colours. As it is clear, mostly picture and text are preferred by users to store their advices. Talking about journey, most users wanted to know about restaurants and mountains.

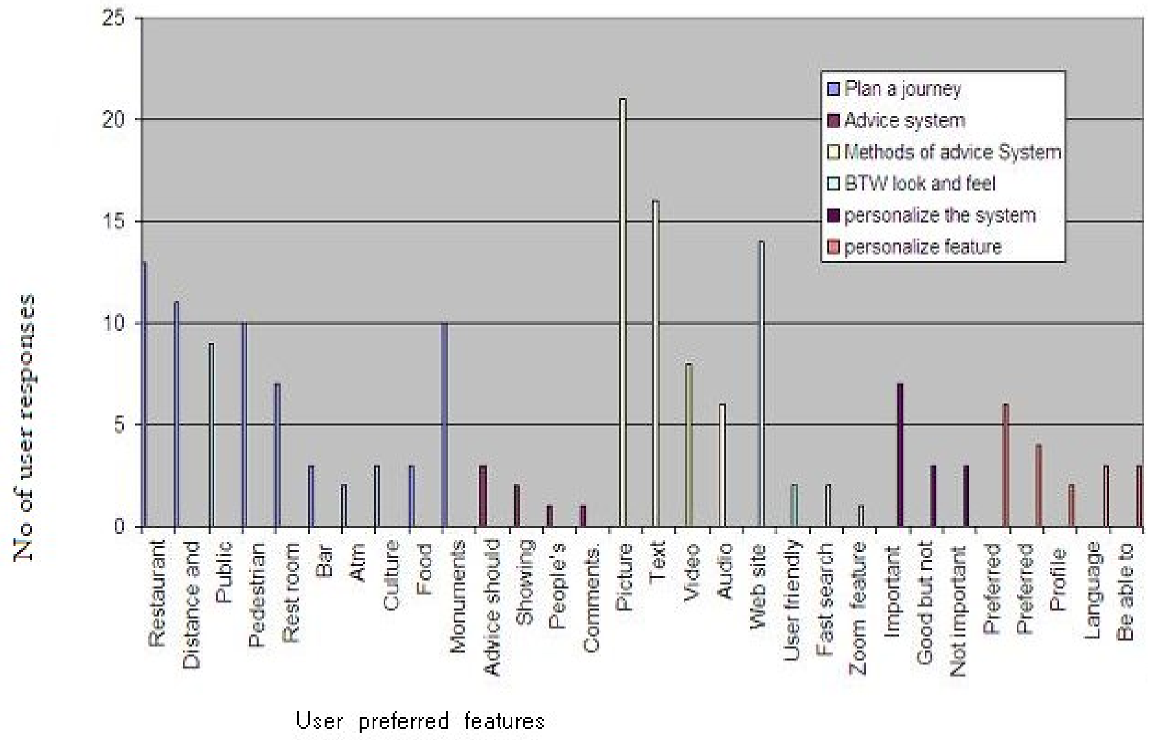


Figure 1: Result of the questionnaire from 30 people regarding to their interests

According to the SCORE project description, requirement survey and some suggestions by PATH team, we provided a list of requirements shown on

Table 1.

Table 1: Requirement Prioritizing

Priority	Description	Source
1	Find route from point A to point B	Customer
1	Seek advice on the map to travel	Customer
1	Register and login	Customer
1	Update personal profile	Customer
1	Save their filter settings to save time	Customer
1	Filter advice	Customer
1	Add advice profile based on their experience	Customer
2	Add new advice template	Customer
2	Add comments in web log of a location in map	Customer
2	Refute or approve previous comments	Customer
1	Manage conflicts	Customer
1	Manage comments	Customer
2	Set effects of dynamic features	Customer
1	Manage conflicts	PATH Team
1	Define Conflicts	PATH team
2	Define Advice Feature affects on measurement	PATH team
1	Manage managers	PATH team

We recognized three users for PATH. The first one is so called User, which is a simple user of the system. Most of the functionalities are available for User. User use cases are the outcome of studying requirement tables. Users can change existing advices several times. Two other PATH users are Manager and Administrator. Manager is a promoted user and able to decide about the conflicts. If changes of one advice occur more than the number that PATH Administration defines (Conflict Threshold or CT), advice conflict would happen. User use cases are shown in Figure 2. Use cases' descriptions are described in PATH requirement specification document.

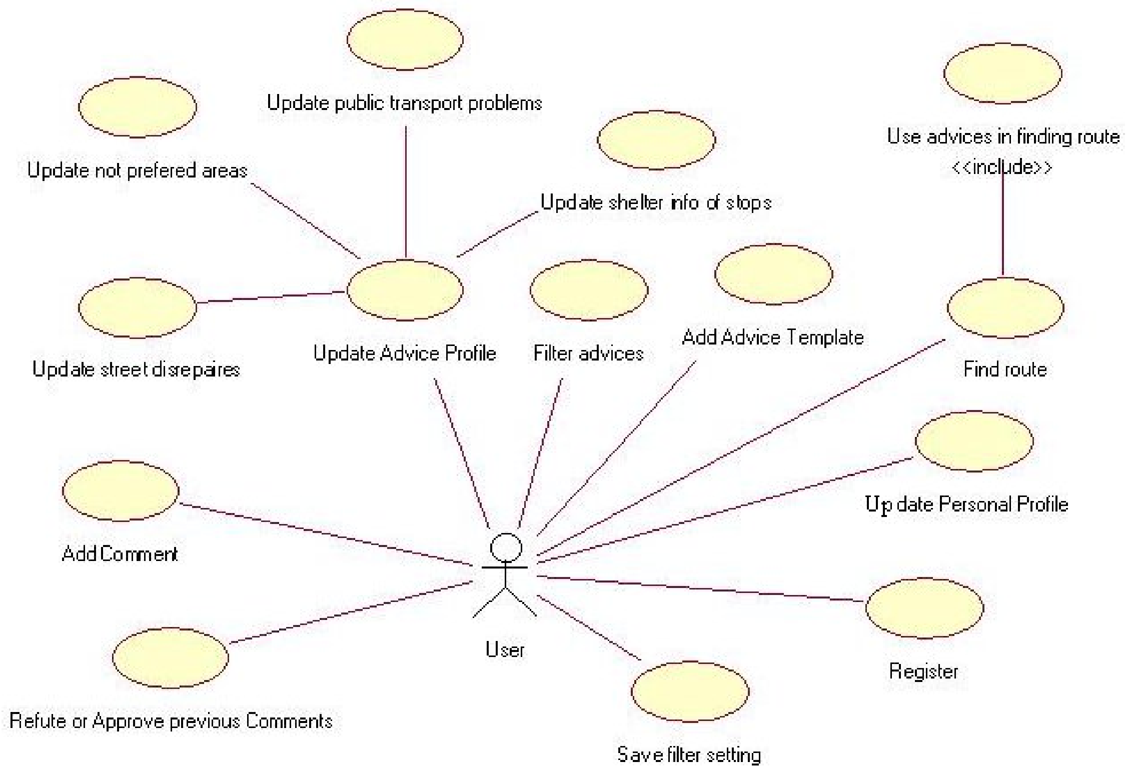

Update public transport problems
Update not prefered areas
Update shelter info of stops
Use advices in finding route
<<include>>
Update street disrepaires
Update Advice Profile
Filter advices
Add Advice Template
Find route
Up date Personal Profile
Add Comment
User
Register
Refute or Approve previous Comments
Save filter setting

Figure 2: User Use Case diagram of PATH

1.2. Requirement verification and validation

As soon as implementation was started, corresponding output of the project was hosted on http://btwmdh.rasip.fer.hr/. Initially requirements of PATH are tested by the use case models. Use case models include use case diagrams, sequence diagrams and activity diagrams which are created from the requirements documentation that is provided by the customer. Also we continually reviewed user perspectives through a web survey. The web survey consisted of responses of 30 people using the application and polling on basis of judging criteria, for example simplicity, interactivity, features and usefulness. This was a useful method which allowed us to interact with customers. Based on user feedback and PATH use case models we verified our requirements.

3. Project Management and Development Process

1.3. Development Process

Initially we used the Rational Unified Process (RUP) model for development. This is a continuous iterative process which let the teams get back to each phase when needed. Team members' Responsibility[1] emphasised on the RUP Role they had and their tasks were about a unit

[1]Members Responsibility is described in Project Plan

of work. We fixed two meetings per a week and meeting duration was about two hours. We developed the project with this model for about four weeks. Each member spent about 16 hours individual work and 4 hours during the meetings, but the result was not good enough and development process was slow. It was because of short meetings and long time working separately. There was a delay in understanding the problem or requirement while working alone. These problems mostly were software installation, configuration and primary project environmental need. So we decided to change the development process into agile development. After that, meetings duration extended to 5 hours each and team members worked less at home (about 10 hours) and more in agile meetings (10 hours meeting in two sessions per week). As a result we could cooperate more with each other, project requirement and project design became more clearly understandable and the progress of the project improved significantly. Furthermore all team members shared their knowledge and experiences more comfortable in the new approach. Agile meeting supported changes better and therefore we could easily change classes and database design. The table below shows the improvement in methodology changes. After week 5, although the number of assigned tasks increased but the number of missed tasks are reduced. Also less number of posts was needed in the second approach.

Table 2: RUP used for first five weeks to develop PATH and in last three weeks Agile was used. The project progress improved within Agile more.

Weeks Number	No of Tasks Completed	No of missed Tasks	Number of meetings held	Number of documents posted in group	Number of post in group	Weekly hours of team	Methodology
1	6	2	2	10	31	139.5	RUP
2	8	3	2	13	30	146	RUP
3	7	2	1	15	28	150	RUP
4	5	3	2	6	31	116	RUP
5	9	2	4	7	30	130	RUP
6	12	2	1	3	20	131	Agile
7	11	4	1	4	10	140	Agile
8	7	1	1	1	5	120	Agile

1.4. Project Plan

In order to define more related tasks and also to encourage all members to be more active during the project, we considered a “Main Responsibility” for each person. Each member took several tasks mostly in his area and helped the project manager for task definition in his scope. Members were able to help each other when needed. Table below shows project team members and main responsibilities during the project.

Table 3: Project Team and Main responsibilities

Name	Initials	Main Responsibility
Mostafa Pordel	MO	Project Manager
Anand T A	AN	Document Manager
Beaulah Vineela P	BE	Project Tester
Abhishek Palod	AB	Development Manager
Riasat Abbas	RI	PATH Designer
Nicholas Cockran	NI	Customer Manager

We used iterative model for developing the project. Table 4 shows the project phases done in each week. During W5 we redesigned our project as a result of some problems we faced with the previous design. Initially we considered some fixed advices that users could select them and fill the proper entries. In the second design we changed advices in order to let the users add their custom advice types. Therefore we could strengthen our project flexibility while we reduced implementation. More details could be found in design section of the report.

Table 4: List of Milestones and project phases (W: Week).

Activity	W1	W2	W3	W4	W5	W6	W7	W8
Project preparations	✓							
Requirements analysis & definition	✓	✓		✓				
Project design	✓	✓			✓			
Implementation		✓	✓	✓		✓	✓	
Testing			✓	✓			✓	
Integration				✓			✓	
Testing		✓		✓			✓	✓
Product Finalisation								✓
Documentation	✓	✓	✓	✓	✓	✓	✓	✓
Final project report								✓

1.5. Problem Management and Communication

We started our first meeting to get introduced with each other. In that meeting we discussed our skills and interests in the project and also we fixed regular meetings for next weeks. Each member expressed the effort he really wanted to put for the project. We found 5 people more interested while one member was not able to put 20 hours a week due to his busy schedule. We selected a project manager and assigned each member a main responsibility as mentioned in previous section. Although we could not find exactly what each member was looking for out of this project, we could distribute the tasks according to the declared interested part. Each week we defined some tasks regarding milestones and during each meeting we discussed problems. We used Analysis Search and Conclude (ASC) approach to manage problems. It contains simple steps that are based on reviewing problems in peers. The chart below shows ASC technique phases.

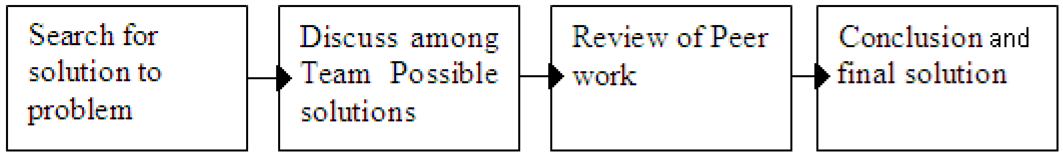

Search for solution to problem
Discuss among Team Possible solutions
Review of Peer work
Conclusion and final solution

Figure 3: ASC Model for problem management

We used Google tools (GTalk, Google email, Google group, Google calendar and Google document) to communicate and share documents among team members during development. A Google group "btwmdh@Googlegroups.com" was created to post any concerns, arguments, suggestions, problems, criticisms and various communication. We can track project problems and events in that group and study difficulties.

A Google spreadsheet was used to set tasks. Once the project manager change task list all members could view it as a Google document. Apart from the Google group, source codes and documents were uploaded to our SVN server regularly. SVN is an open source version control tool. Also we used DSD official web site to upload weekly report, project documents and news for the project steering group. The steering group were sending us feedback regarding to our presentation during the course and documents that we uploaded on DSD web site. The feedback were thoroughly analysed to incorporate them in project.

4. Architectural Design

Architectural design consisted of high level architecture and detail design including sequence diagrams and class diagrams.

1.6. High Level Architecture

Project design consisted of three tier architecture. It included User Interface Layer (UIL), Business Object Layer (BOL) and Data Access Layer (DAL). The user can view the system through UIL. This layer was built using HTML, PHP and JavaScript. There are just a few client side processes such as "Input Control" which are implemented entirely in UIL.

All other processes are passed on to the BOL for processing. BOL was built using PHP. It contains business modules and classes that take the appropriate decision and fetch the required data from Google Maps through the Google Maps API and from PATH database through DAL. The interaction with Google API is through JavaScript. The database is built with MySQL.

Figure 4 shows a high level architecture of PATH. As the users decide types of Advices, a Generic Type was used to store different advice kinds.

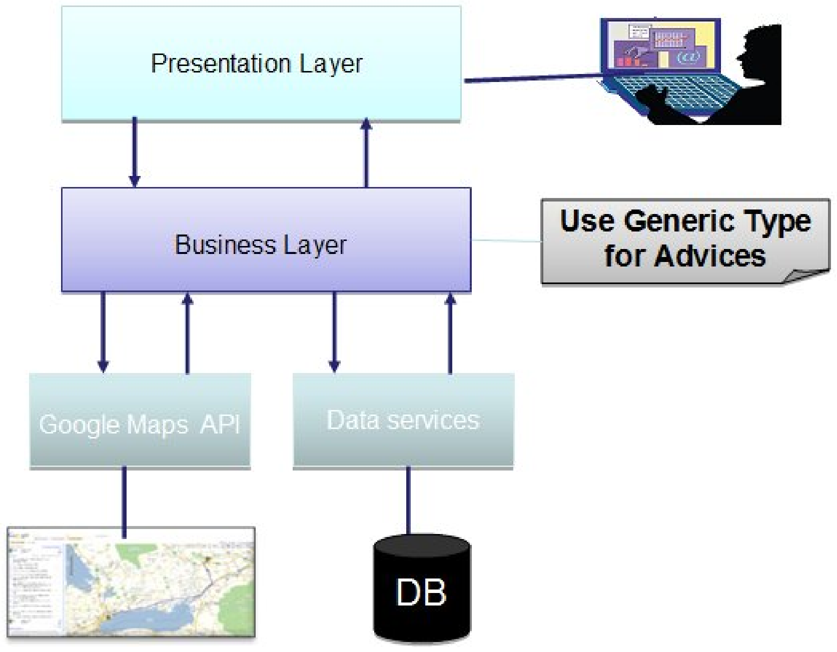


Figure 4: High level architecture of PATH

1.7. PATH Class Diagram

The class diagram is briefly explained in this section. The three users of the system are Administrator, Manager and User. All users are inherited from base user. Advices are stored in Advice and Advice Fields. This made the project very flexible because there is no limitation for any changes that user want for any advice. Advice entries are stored in advice profile. Here we save all changes needed happens for each advice in a special location. The related class diagram is shown in .

The last entries for each location are the valid data. Administrator defines Conflict Threshold (CT) for advices. If PATH notices so many changes (More than CT) for an advice entry, Conflict would occur and nothing more could be saved from users. Managers then are able to review conflicts and decide the final values.

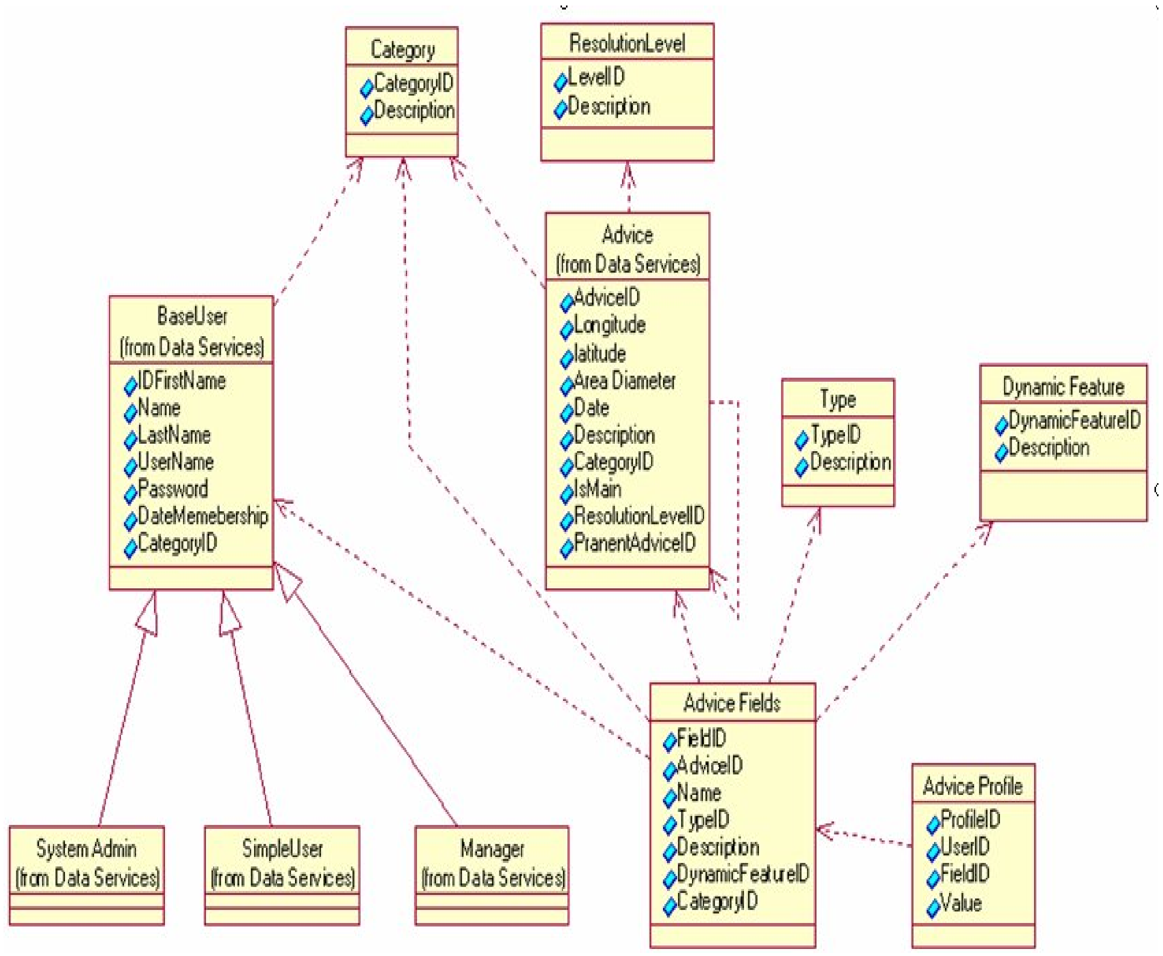


Figure 5: Simplified PATH Class Diagram

1.8. User search Sequence Diagram

This sequence diagram is depicting all the interactions between the layers that serve the user request. The user can search for a route with search criteria entered. The system will interact with the Google API to get the map as requested. The map then needs to be embedded with the advice that corresponds to the requested route. PATH search engine will then fetch advices from the database and add them as flags on the map fetched with the Google API. The final map is returned to the user.

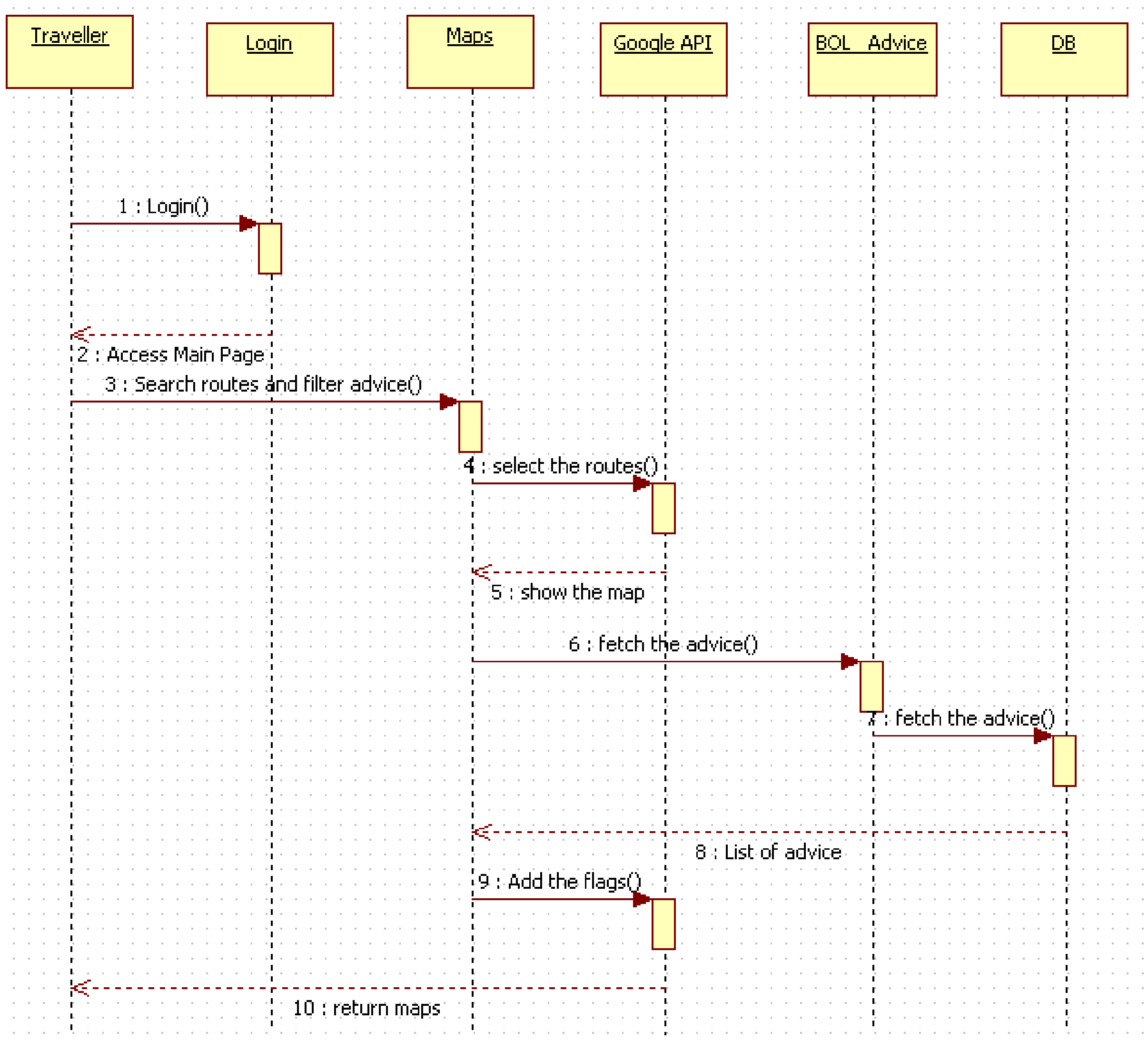
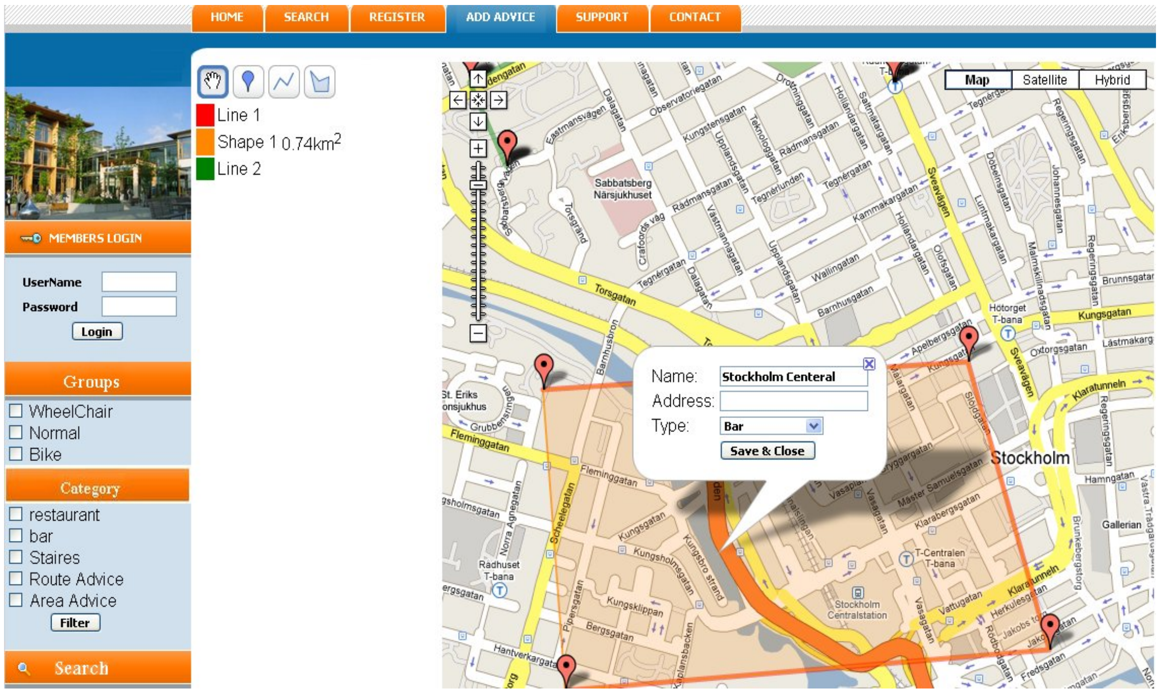


Figure 6: Search route sequence diagram

5. Implementation

The implementation was based on the open source tools. We used My SQL 5.51, Eclipse 3.4.1, Apache server 2.0 and PHP 3.0 and Google map API. PATH presentation layer contains 13 files. BOL contains 10 files and finally DAL includes two files. The average number of lines of code for each file is about 100. We used a big header file that include left side menu bar also upper menu bar. Also a common footer was used for all PATH web pages.

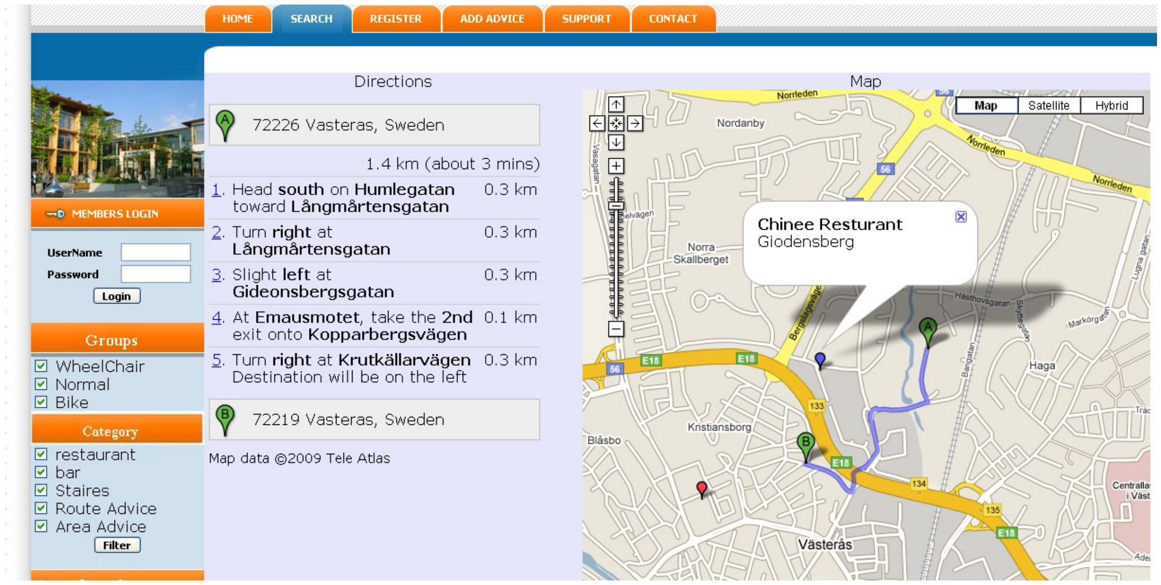
Figure 7 shows an image of the process of adding advice to the system based on the user experiences. The system supports adding point based, route based and area based advices. Each of these three mentioned sets includes its own list of advices that are set when a new Advice Type is adding to the system. Accordingly users select a marker or multi line or polygon to draw on the map. These advices are saved to database along with the co-ordinates of the advice. Each advice could belong to a category. We have added restaurant, bar, stairs and some other categories in the current application as samples of what could be created by users. Each Advice Field could show something specifically useful for a group. For example if we think of Underway as an advice and stairs as a field of it, then this field could have a group like Wheelchair. So within this design users are able to filter the fields they want rather than all fields. It is very helpful when the advices are complicated and big.



Name: Stockholm Centeral
Address:
Type: Bar
Save & Close

Figure 7: Add advice. Advices can be line based, point based or region based

Figure 8 depicts route finding aspect of the system. The user enters the source and destination location and the system generates the route on map as well as route description. Once the route is generated, the user would like to see the advices for the route and nearby places. Also it depicts the functionality of advice filtering. The left column provide the checkboxes for various kind of filters. The user selects the required and the same is displayed on the map. We can see a marker, line, and area based advice on the map.



Directions

72226 Vasteras, Sweden

1.4 km (about 3 mins)

1.	Head **south** on **Humlegatan** toward **Långmårtensgatan**	0.3 km
2.	Turn **right** at **Långmårtensgatan**	0.3 km
3.	Slight **left** at **Gideonsbergsgatan**	0.3 km
4.	At **Emausmotet**, take the **2nd** exit onto **Kopparbergsvägen**	0.1 km
5.	Turn **right** at **Krutkällarvägen** Destination will be on the left	0.3 km

72219 Vasteras, Sweden

Map data ©2009 Tele Atlas

Figure 8: Filter advices while finding a route

6. Verification and Validation

Early testing was done in the project as mentioned in project plan. This is done to check that all the required functionalities are implemented in the project. PATH Testing is done in two levels:

1.9. Requirements Testing

In PATH we have given importance to requirements testing (/i.e. to validation) Table 5 lists requirements test cases. All the priority 1 requirements are successfully implemented. In addition to this, some of priority 2 requirements are also implemented successfully. Here we list specifications some of the tests. The whole tests specifications are documented in the Test specification document.

Table 5: Requirement test cases

Requirements	Input	Expected result	Actual result
Register User	Enter Values	Register user with specified	Successful user registration
Login	Enter Username and Password	Login to system	Successful login
Route Finding	Find route from Point A to Point B given point **A**	Display route on map	Route is shown from point A to point B

Advice giving	User gets the advice from system for specified place	Display advice on map for the specified point	Advice is displayed on map.
Filter Advice	User gets advice depending on the filter criteria that are selected	Display advice based on filter conditions	Advice is displayed on map
Register a user with preferred	User to select one of the categories while registration	Able to register a user with category he/she	Successful selection of user category
Search for a place	Enter place name	Display the given place information on the map	Successful display of the information about the place on map
Display of advice flags	System will display the flags corresponding to the advices	Advice flags are displayed on the map.	For given route or a place flags should be displayed corresponding to advices
Advice given to user based on profile information	Advice should be given to user depending on his/her profile information	Advice is given independent of user profile information	Advice needs to be given depending on profile information of user

7. Outcomes and Lessons learned

This project was a good experience for us. Team members are from different countries and each has a different background and culture. We are from Iran, Australia, Pakistan and India. We spent some times to improve mutual understanding. Also the PATH supervisor is in Zagreb, Croatia. Finally a well-organised and self managed system was developed. We put 5 advice types but users could add some more when needed. The application is now hosted on the web site that was mentioned and there is a chance to get more feedbacks from users and improve it a bit. All project documents are uploaded on DSD course website that is http://www.fer.hr/rasip/dsd/. This link is available for public and there is no limitation for accessing project documents. Project documents include, project requirements, design and project analysis. Some detail information of the process of the project development such as weekly reports, minutes of meetings and presentations' slides are also available for download. Project files are also available in SVN server that was used for version controlling and source sharing.

According to the development process, learning time was higher than we expected in the beginning. Because of configuration and installation problems on some team member's machine, we used remote development as well as local development. We redesigned the architecture of PATH once to allow users themselves decide the advice type rather than providing some fixed type of advice. At the beginning each team member was working individually on his tasks and we had short time meetings (5 hours a week). We faced slow development pace of project. So we decided to extend meeting duration and pair programming (10 hours a week). As result team able to learn and share each other mistake and peer communication among team solve many technical and functional issue in less time. So we found out that a design is a process to be done in several iterations. Team was able to implement full functionality of SCORE minimum requirements.

8. Summary

In this project we developed a web site which provided travellers help to get detailed information about any route and search required routes and places. This web site filters the advices according to user requirements and displays these on the output map. The advices are provided only to registered users and any registered user can enter their travel experiences in this web site.

The development team consisted of six members and each person had a specific role in the development of the project. We learned new technologies and gained new knowledge in the development of this project. We used the Rational Unified Process (RUP) and agile model for software development.

The project design consisted of three tier architecture and it included the User Interface Layer, Business Object Layer and Data Access Layer. These layers are independent and interacting with each other. The layer architecture provided many advantages such as easily divided work. We used open source technologies and software.

The main challenges in this project were time duration, selecting the right Geographical Information System (GIS) and selected GIS API library dependency. For the requirements we used SCORE project description, and a web survey. In the web survey we got the feedback from internet users. According to the feedback and SCORE project description, we collected requirements of this project. This web site provided many characteristics to registered users in the sense of adding the advices as well as advice types. Advice types refer to different types of advices such as bus station, bar and restaurants. The next important feature is managing conflicts of advice entries. In a conflict scenario, managers have the permissions to decide the correct advice. The system users are promoted as managers of the system based on their proven track record. As a result the PATH web site itself managed and implemented similar to wiki management.

9. References

[1] Colin Potts, Kenji Takahashi, Annie I.Anton, "*Inquiry-Based Requirements Analysis*", Georgia Institute of Technology, Nippon Telegraph and Technology Corp.

[2] Kousik Sankar R, Raman Venkat, "*Total Requirements Control at Every Stage of Product Development*", Philips Electronics India Limited.

[3] Anthony Finkelstein, Steve Easterbrook, Jeff Kramer & Bashar Nuseibeh, "*Requirements Engineering Through Viewpoints*", Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ.

[4] Harry M. Sneed, Ancon GmbH, "*Testing a Web Application*", Vienna, Austria & Budapest, Hungary

[5] Tauhida Parveen, Scott Tilley, George Gonzalez "*On the Need for Teaching Web Application Testing*", Department of Computer Sciences, Software Quality Management, Florida Institute of Technology, Sabre Holding Inc.

[6] Google Maps API, "*What is the Google Maps API*", Jan 2009;

http://code.Google.com/intl/sv/apis/maps/

[7] PHP, "*Documentation*", Jan 2009; http://www.php.net/docs.php

[8] Sun, "*MySQL 5.1 Reference Manual*", Jan 2009; http://dev.mysql.com/doc/refman/5.1/en/

[9] Apache, "*HTTP Server Project*", Jan 2009; http://httpd.apache.org