Bachelor's Thesis (UAS)

Information Technology

Specialization

2013

Chinedu Eze

DESIGN AND IMPLEMENTATION OF THE LOST AND FOUND WEB APPLICATION

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Networking

Date | Number of pages

Instructor: Balsam Almurani

Chinedu Eze

DESIGN AND IMPLEMENTATION OF THE LOST AND FOUND WEB APPLICATION

Recovering lost items is so common that in places such as cities in Finland, there are offices that are responsible for recovering and keeping such items for the appropriate owners to collect them. However, recovering such items is not as straight forward as it seems. A visit to one of those offices shows that a lot of items have been left there without collection. I believe that if the owners of such items had an easier way of recovering those items, they would.

This thesis presents the design and implementation of a better and easier solution for such case. At the same time, this project does not eliminate the former method but presents an easier solution.

The lost and found web application presents a system that enables users to notify authorities of their missing items and be assured that as soon as their items has been found, they would be notified, as against calling and / or visiting the offices to check.

This thesis presents a detailed description of the problem and how the application solves it. The implementation details and the tools used are also described. Finally, the results of the project are presented.

KEYWORDS:

(Vaadin, Web Application, lost and found)

FOREWORD

With so much time on my hands, I began working on this project last summer. Originally the project was meant to manage items that were lost in city buses only. But over time, it further developed to take care of trains and other places as well. The first version was later ready in the autumn. I enjoyed working in this project and the learning experience was great, with tons of surprises all the way. Finally, I learned the joy of accomplishment.

I would like to thank Mr. Balsam Almurani for being a great thesis supervisor. I would also like to thank Mr. Ezio Melotti for great programming lessons and, without whose help, the matching engine for this project may not have been completed. I would also like to thank my mother, Mrs. Philomina Eze, for being my very own personal hero. Most importantly, I like to thank the Almighty God, without whom, all of this would not be possible.

Spring 2013 – Turku, Finland

Chinedu Eze

CONTENTS

FIGURES

ACRONYMS, ABBREVIATIONS AND SYMBOLS

RDMS	Relational Database Management System
MSDN	Microsoft Development Network
DoS	Denial of Service
DDoS	Distributed Denial of Service
AES	Advanced Encryption Standard
JAWS	Java API for WordNet Searching
SVN	Subversion
API	Application Programming Interface
JDBC	Java DataBase Connectivity
UI	User Interface
URL	Universal Resource Locator
JVM	Java Virtual Machine
OOP	Object Oriented Programming

1 INTRODUCTION

Since inception, the growth of the internet and its uses has been tremendous. This growth has continued to extend to new aspects of people's everyday life. This growth has also meant that people generally turn to the internet for almost anything. This thesis presents yet another use for the internet.

Normally when people lose items, they call appropriate offices. This thesis presents yet another problem and proposes a solution that would depend on a web application to solve it.

1.1 Thesis overview

This chapter gives a general overview of the thesis. The thesis report is split into different chapters, each explain specific aspect of the project. Following is a summary of the contents of each of the chapters:

Chapter 1 – Introduction

This chapter gives an overview of the thesis.

Chapter 2 - MOTIVATION

This is the chapter that introduces the problem that the web application solves in details. The three research projects executed towards getting a better understanding of the problem and their results are also described in details and the limitations of the research works are also given.

Chapter 3 – The Research Process

This chapter presents the readers with an overview of the surveys undertaken in order to complete the project. The outcomes and limitations of the surveys are also given in considerable details.

Chapter 4 - The Web Application

This chapter introduces the web application. Each part of the application is described in details. The user and admin sides of the application and the messaging system are all described in details. The chapter tells what the application is built for and how it works.

Chapter 5 - The match engine

This chapter presents machine engine. Topics that are discussed here include reason why the engine was created, the problem that it solves, other possible approach to solving the same or similar problems and why there were not chosen for this project, the theory behind the implementation of the engine and the limitation of the engine.

Chapter 6 – Design and Implementation Details

This chapter describes the details of the inner workings of the application and how they have been designed. It also presents details of the database and key Vaadin framework items that were used in the application.

Chapter 7 – Tools and technology

This chapter introduces the tools and technologies employed in the development of the project. These tools and technologies include the Vaadin framework, MySQL DBMS, the programming languages, the WordNet dictionary and library, etc. The reasons why the tools are chosen for the purposes are also listed.

Chapter 8 – Web Application Security

This chapter briefly explains the security issues that web applications would have to deal with. Two of the most popular security vulnerabilities are explained. Also, how the Vaadin framework takes care of these vulnerabilities is explained. Furthermore, the various precautionary security measures that have been put in place are discoursed.

Chapter 9 – The Development Method

This chapter presents the software development model that was used to build this application. This chapter describes this model, explaining the various aspects of it and how they have been applied during the development of the project.

Chapter 10 - Conclusion

This chapter presents the conclusion of the thesis. Although the project has been implemented fully to specification, there exist some improvements that may need to be done at least in future releases of the project. These possible improvements are discussed in this chapter.

2 MOTIVATION

Losing items or belongings in public transport systems such as trains and buses, or other places, is so common that in places such as cities in Finland, there are offices set up to collect and store these items for collections by the appropriate owners. Sometimes, these offices are part of the police station or part of the public transport service office. At other times, these are separate offices on their own, which may, or may not, be affiliated with the transport service company.

When people lose items, they normally call or pay a visit to those offices, in order to reclaim their lost items. One problem is immediately evident here. Sometimes, people are not so sure the exact places they lost their items. There are various offices to call and / or visit. Sometimes, it may not be clear which of the offices to call. For example, someone may not be sure if the item was lost in the train, in which case one has to call the train station. On the other hand, the item may be lost in a bus on the way home from the train station, in which case one has to call the bus station. Also, one may have lost it on the way from the bus stop home, in which case s/he may have to call the police station. In rare cases, any of the above mentioned offices may not have the item, hence, one has to call those three offices over and over.

The main target of this project in this regard is to propose a single place to resort to for reporting such cases. With a single web site for reporting lost items, no matter the location, users would be able to just report an item as missing and trust that as soon as the item is found and brought to one of the offices, they would be [automatically] notified.

Also, the process of recovering these lost items could sometimes be very tedious. For example, when users are not sure any more as to when they lost an item, most likely, the workers at the lost and found office may not be able to locate the items, even though they have been found and brought to the office. This may result in frequent calls and / or visits to the office in an effort to recover those items. Obviously, it would be a better idea, if when users lose an item, they simply just report them and wait for the company to contact them as soon that item is found, instead of calling and visiting.

The above described, is the main problem that this web application seeks to solve. In addition to that, this project also seeks to reduce the amount of tasks that the workers at those offices need to do in other to locate an item from the list of items in the database. This is achieved with the message system of the application which automatically sends an email to users when an item is identified as the one that a particular user described.

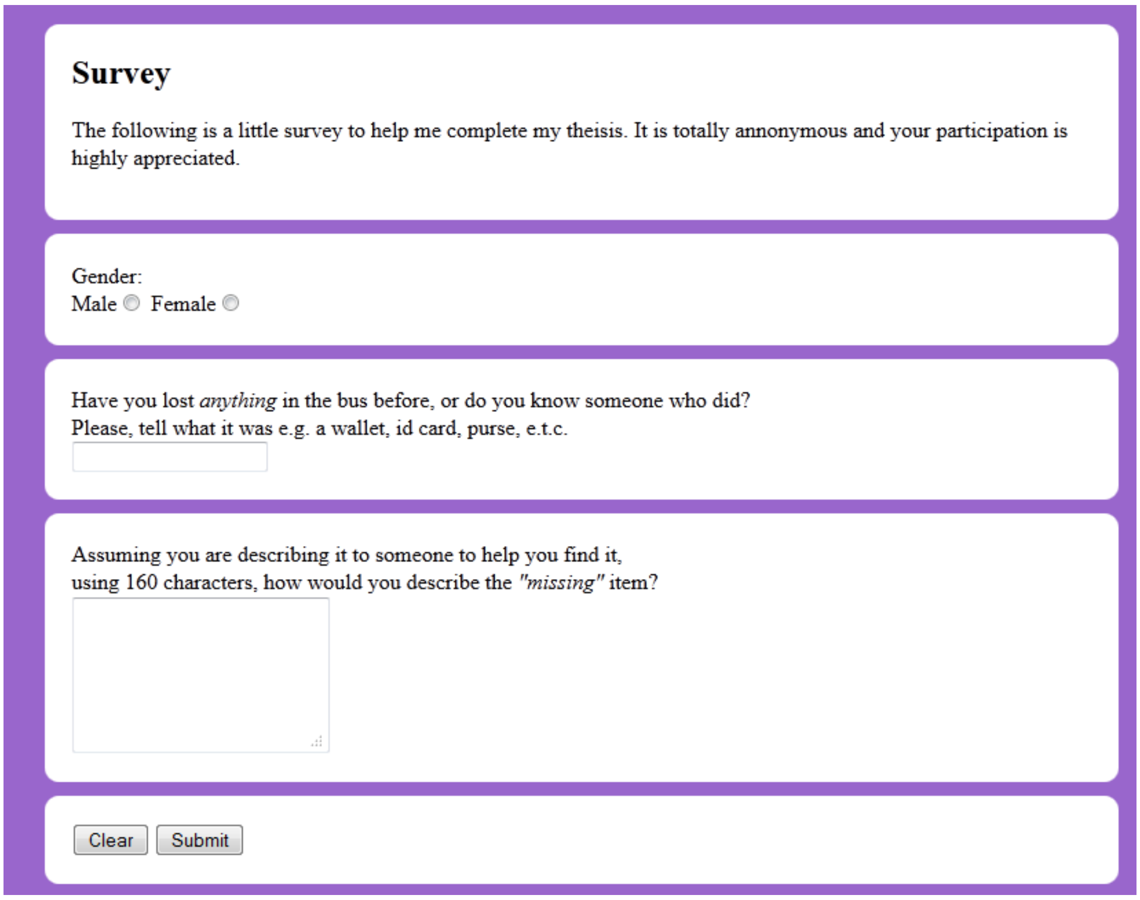
3 THE RESEARCH PROCESS

After studying the problem, the author decided to carry out surveys in other to better understand it and find out a proper solution. The aim of these surveys is to get to a better understanding of the problem and to determine the correct direction to pursue a solution. Three surveys were undertaken in the course of the project. The purpose of each was to study specific aspects of the problem and possible solution(s).

The surveys were all web based. The web pages were written in HTML and JavaScript using the jQuery library (jQuery 2013). The jQuery plugins employed were jQueryUI and jQuery.validate. The server side was programmed in PHP and the data were collected in a MySQL database. No statistics analysis tools were employed in the analysis of the results of the surveys. Rather the results were analyzed by hand. Using the XAMPP package (Apache friends 2013), the pages were run on an apache server during development and testing, and on a free hosting site, http://www.1freehosting.com/, during production.

3.1.1 First survey

The purpose of the first survey was to find out what kinds of items that people lose in buses and trains. The author, not having a good understanding of the problem yet, wishes to design the web app that would be streamlined in terms of items. For example, the items could be categorized. Also, participants were asked to give a brief description of the lost items. This was to aid in putting the items in their proper categories.



Survey

The following is a little survey to help me complete my theisis. It is totally annonymous and your participation is highly appreciated.

Gender:
Male ○ Female ○

Have you lost *anything* in the bus before, or do you know someone who did?
Please, tell what it was e.g. a wallet, id card, purse, e.t.c.

Assuming you are describing it to someone to help you find it,
using 160 characters, how would you describe the *"missing"* item?

Figure 1: Snap Shot of the First Survey

3.1.2 Results of the First Survey

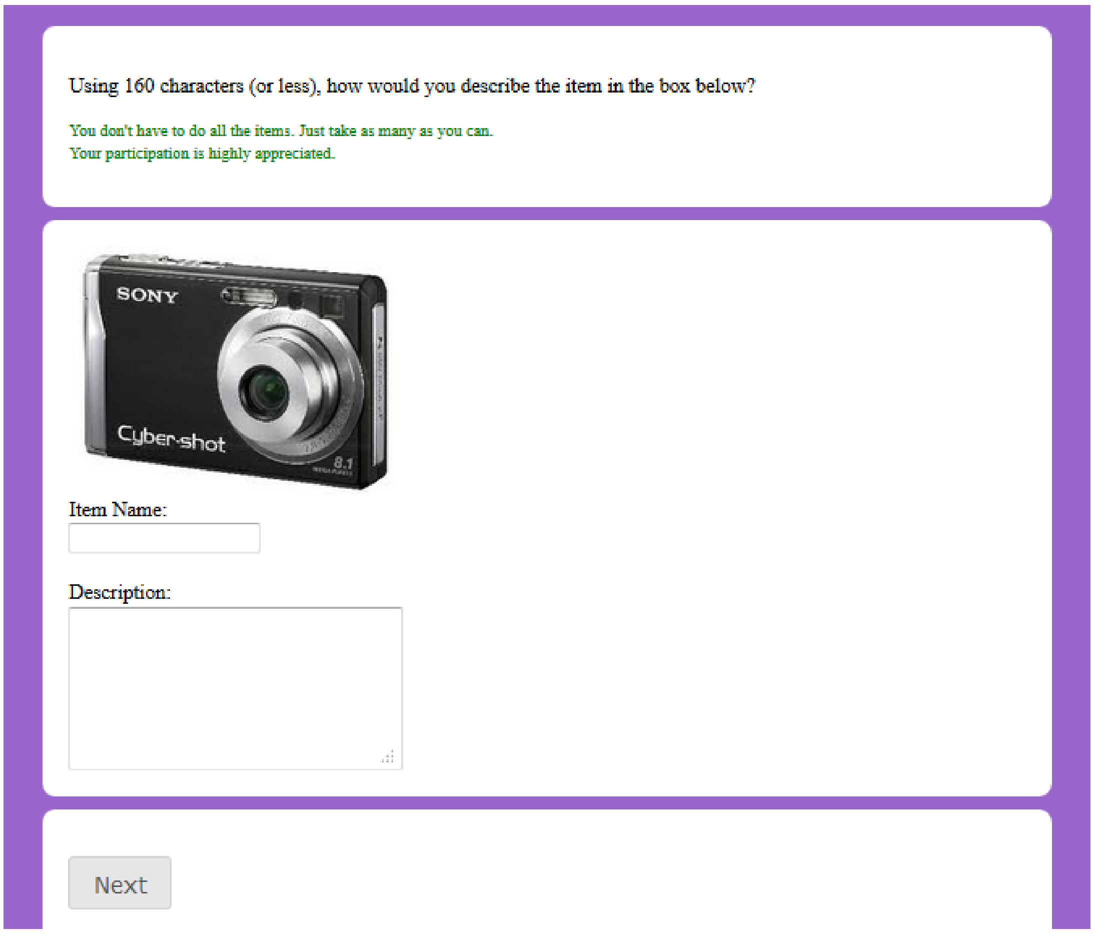
A total of 86 people took the first survey. With a ratio of 56:30, it was gathered that males lose more items than females. This lead the author to speculate that it is mainly because the females participants mostly keep their items in the purse, and hence, there is a reduced chance of losing them. However, this conclusion is not to be trusted, as it could be that male participants are more likely to take part in the survey than female participants.

Nonetheless, the main purpose of the survey, which is find out what kinds of items are lost in public transport systems, was achieved. Chief amongst those were keys, mobile phones, handbags, backpacks and wallets. Most importantly, the author concluded from the survey that putting items into categories would make the web page too complicated for users, defeating the main aim of the app, which is to make the process of recovering lost items easier. For example, users may have a difficult time choosing what category a particular item belongs. Ultimately, users may get frustrated and choose to call the lost and found office instead.

3.2.1 Second survey

Having cancelled out the idea of categorization, the author then sought to find a better, but easier, solution. Going through the descriptions given by users in the first survey, it was gathered that there could be enough information from those to get to the item, without the use of categories. Hence, another survey was carried out.

The aim of the second survey is to find out how people would describe an item. Information, such as, what kinds of words are used to describe items, what would different users say about the same items, if asked to describe it, and most importantly, whether those description would be enough to actually pick out that particular item from a list of others.


Using 160 characters (or less), how would you describe the item in the box below?

You don't have to do all the items. Just take as many as you can.
Your participation is highly appreciated.

Item Name:

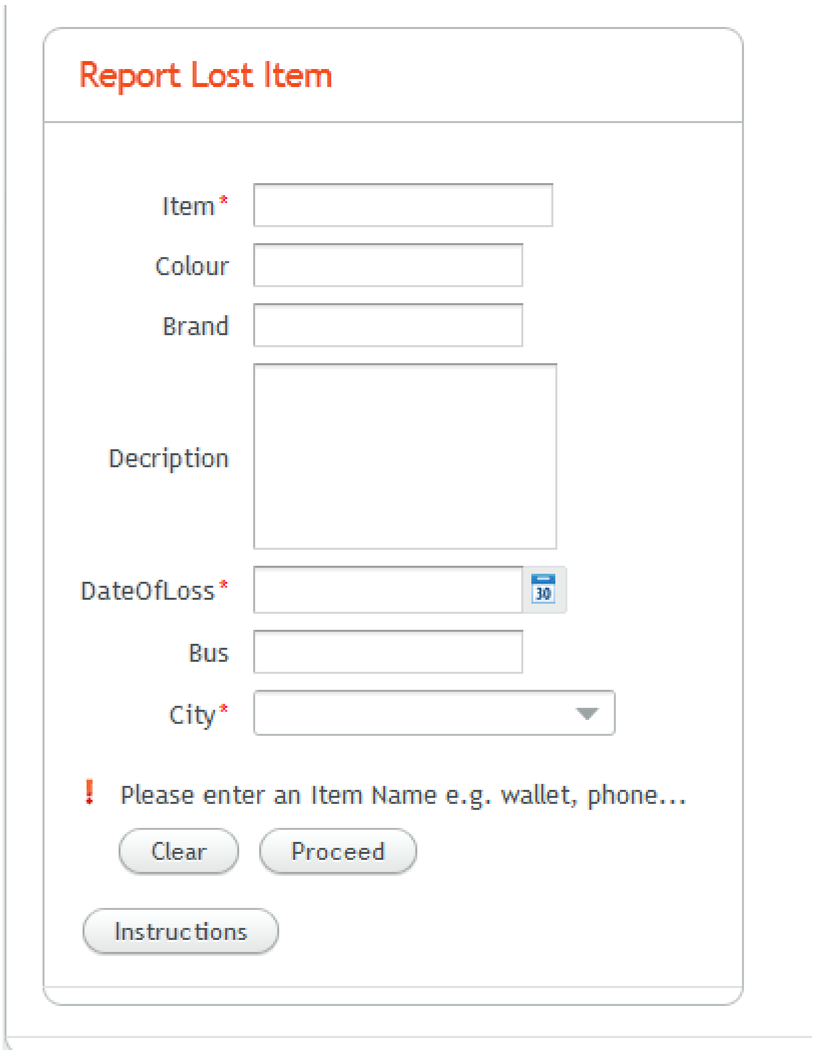
Description:

Figure 2: Snapshot of the second survey

In this survey, users were presented with pictures of items that were listed from the first survey, one at a time and, asked to describe it using a limited number of character (160). This limitation is mainly for computer resource management reasons. Also, with a limitation on the number of characters allowed, users would most likely stick to useful, informative words in the description, as against beating about the bush.

3.2.1 Results of the Second Survey

The second survey had a total of 58 participants. One of the main outcome of this survey was, an albeit incomplete, knowledge of how users describe items. Also, the second survey led to the completion of the first version of the web application. Among other fields in the form for this version, were fields for colour, brand and description of items. These fields were included as a result of the fact that the users most likely included those information in their description of items.



Report Lost Item

Item*	
Colour	
Brand	
Decription	
DateOfLoss*	
Bus	
City*	

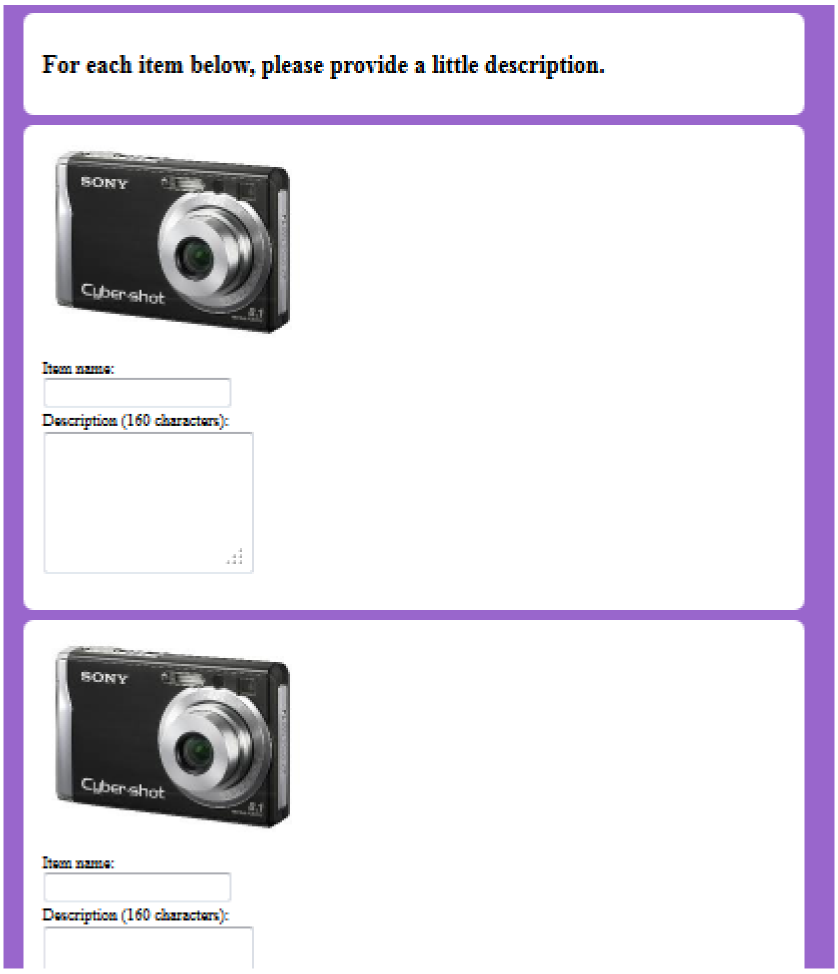
! Please enter an Item Name e.g. wallet, phone...

Figure 3: Snapshot of the first version

However, after the first version was released to a few test users, it was discovered that some information were redundant. For example, after including the brand name for an item in the brand field, most users also included them in the description area as well. Also, some users complain that after including the brand name and colour of an item in the appropriate fields, they are left with almost nothing else to say about those items in the description field.

Finally, it was concluded that all the information regarding the item itself should be limited to the "item name" and "description" fields. At this point, it is clearly seen that the web application would need to do more work on the supplied information in order to locate an item from a list of other items. Thus, the idea of the match engine was born. The main idea behind the engine was, as was discovered from this survey, that different users use certain types or categories of words to describe items. If these words are matched against each other, there is a high probability that similar words would be found amongst descriptions given by different users for the same item. Details of the match engine would be further discussed.

3.3.1 Third survey


For each item below, please provide a little description.

Item name:

Description (160 characters):

Item name:

Description (160 characters):

Figure 4 Snapshot of the third survey

This survey was carried out after the development of the final version of the web application. The main intent was to provide data for testing the efficiency of the matching engine. In this survey, users were presented with pictures of random items, with at least two similar items at a time and, asked to describe them. One of these descriptions were entered from the admin side of the web app, while one other is entered on the users side of the app. This test was done outside of the web app, as the match engine was still under development, and hence, was not yet incorporated into the application.

3.3.2 Results of the third survey

Thirty people participated in this survey. Five of the resulting outputs were discarded as those information were considered incomplete or not in line with the instructions of the survey. With a match threshold of 70 per cent, the match engine did achieve an impressive efficiency.

3.4 Limitations of the surveys

One of the main limitations of the survey was that instead of physical items, users were shown pictures. This means, that it is possible that were those items presented to participants physically, they could probably give a different description than those given on the survey. Also, given that there were only a few participants in each survey, the results may not be an inaccurate description of the problems investigated.

4 THE WEB APPLICATION

The web application can be broadly divided into three parts, viz: the front end, the admin/backend and the internal sub systems. This division is based on the roles that each system plays towards the end goal of the entire system. The front end, for example is responsible for taking user requests and storing them in the database. The backend is responsible for taking details about found items and storing such in the database. On the other hand, the internal subsystem is responsible for helping the other parts of the system to accomplish their tasks. For example, the messaging system is responsible for sending confirmation messages to users after they completed the form on the front end. Both ends are built to function independently but rely on the database for data.

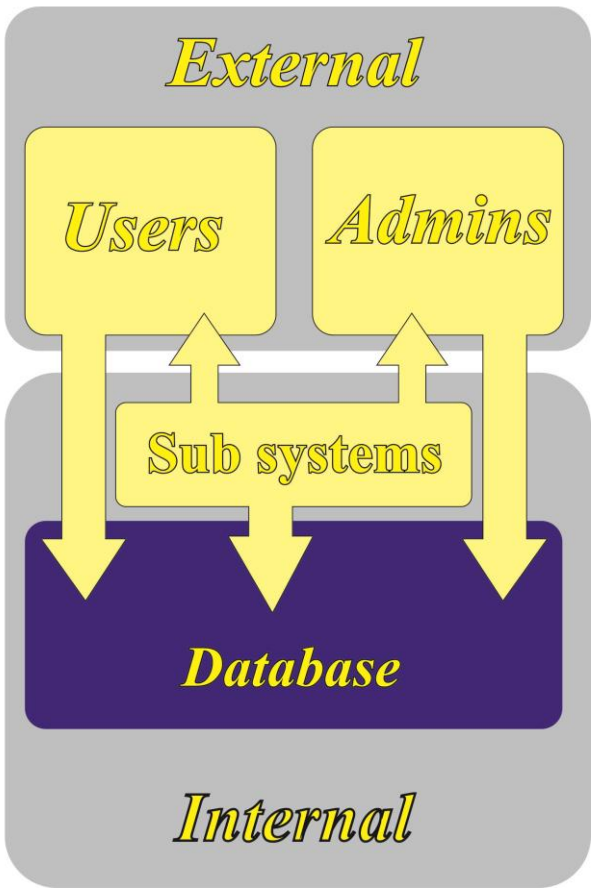
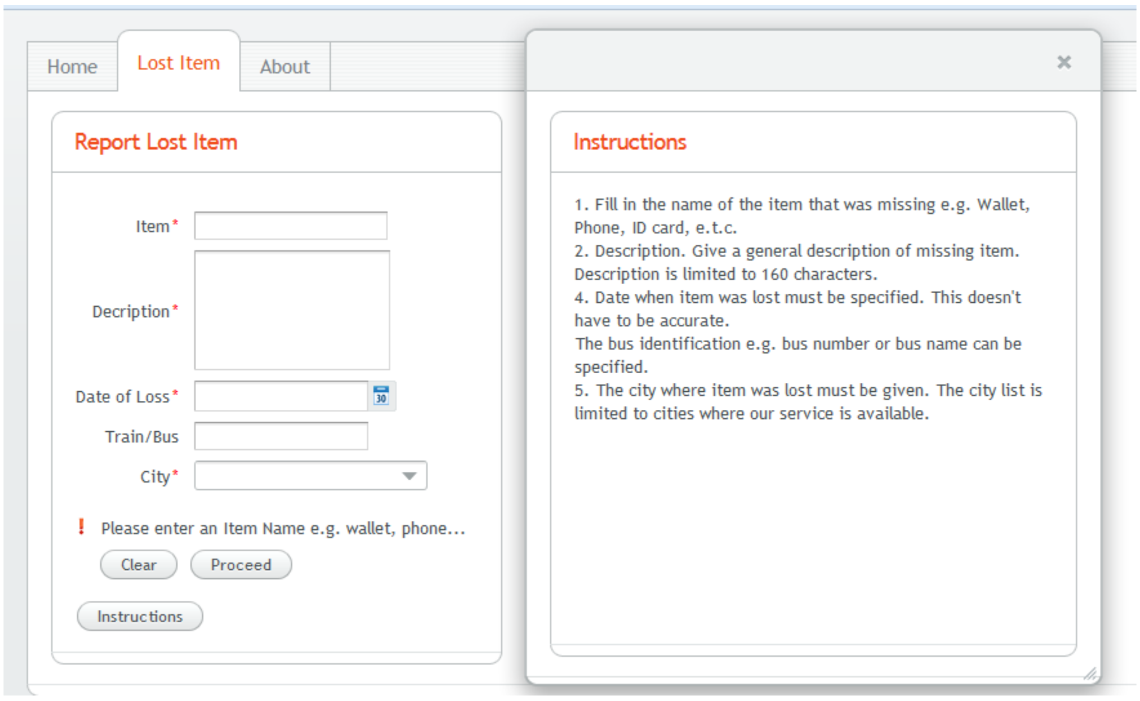


Figure 5. Schematic diagram of the web application

The Front End - The Users' Side


Home
Lost Item
About

Report Lost Item

Item*
Decription*
Date of Loss*
Train/Bus
City*

! Please enter an Item Name e.g. wallet, phone...

Clear | Proceed

Instructions

Instructions

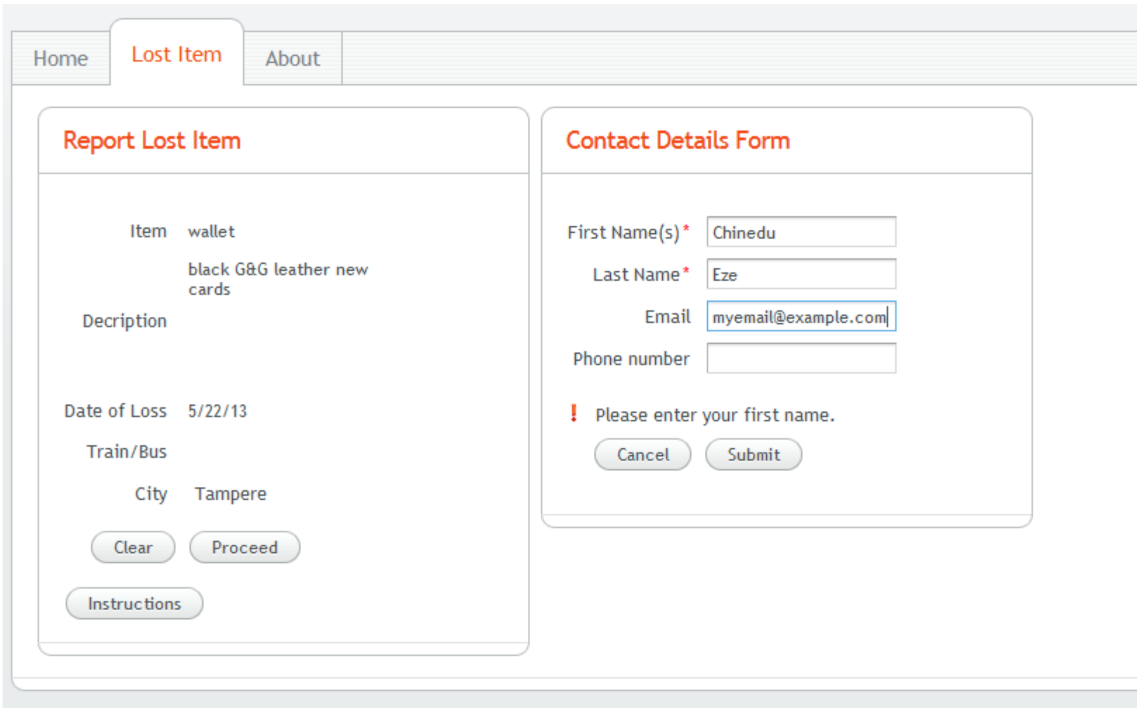
1. Fill in the name of the item that was missing e.g. Wallet, Phone, ID card, e.t.c.
2. Description. Give a general description of missing item. Description is limited to 160 characters.
4. Date when item was lost must be specified. This doesn't have to be accurate.
The bus identification e.g. bus number or bus name can be specified.
5. The city where item was lost must be given. The city list is limited to cities where our service is available.

Figure 6. The users' front page with instructions displayed

This is the part that enables users to report an item missing and enter information about the missing item. When the user is done filling in the item information, then s/he can click on the proceed button. Only then is the form for contact information displayed. This is in line with usability guidelines such that only needed information is displayed at a time.

The contact form includes text fields for names, phone number and email address. It is compulsory that users supply at least a phone number or an email address. Also, there is an “Instructions” button that displays instructions on how to fill in the needed data. The instructions are displayed in a pop up window, which can be dragged to any part of the screen. Users can actually expand the window and place it side by side with the form, so that they can view the instructions while filling the form. This is a usability feature that enables users to display and view the instructions without having to leave the current page. The procedure is kept simple in accordance with the main aim of the application, which is to make the process of recovering a lost item easy.

In addition to those, there is a home screen that is initially displayed once the site is loaded. Also, there is an about screen that displays information about the web site and its purpose. Each of these screens (pages) are well organised into three appropriately named tab sheets. This is a usability feature that enables users to be able to reach necessary information with just a few clicks. Also, uses are able to visit any part of the application at any time with ease.



Item	wallet
Decription	black G&G leather new cards
Date of Loss	5/22/13
Train/Bus	
City	Tampere

First Name(s)*	Chinedu
Last Name*	Eze
Email	myemail@example.com
Phone number	

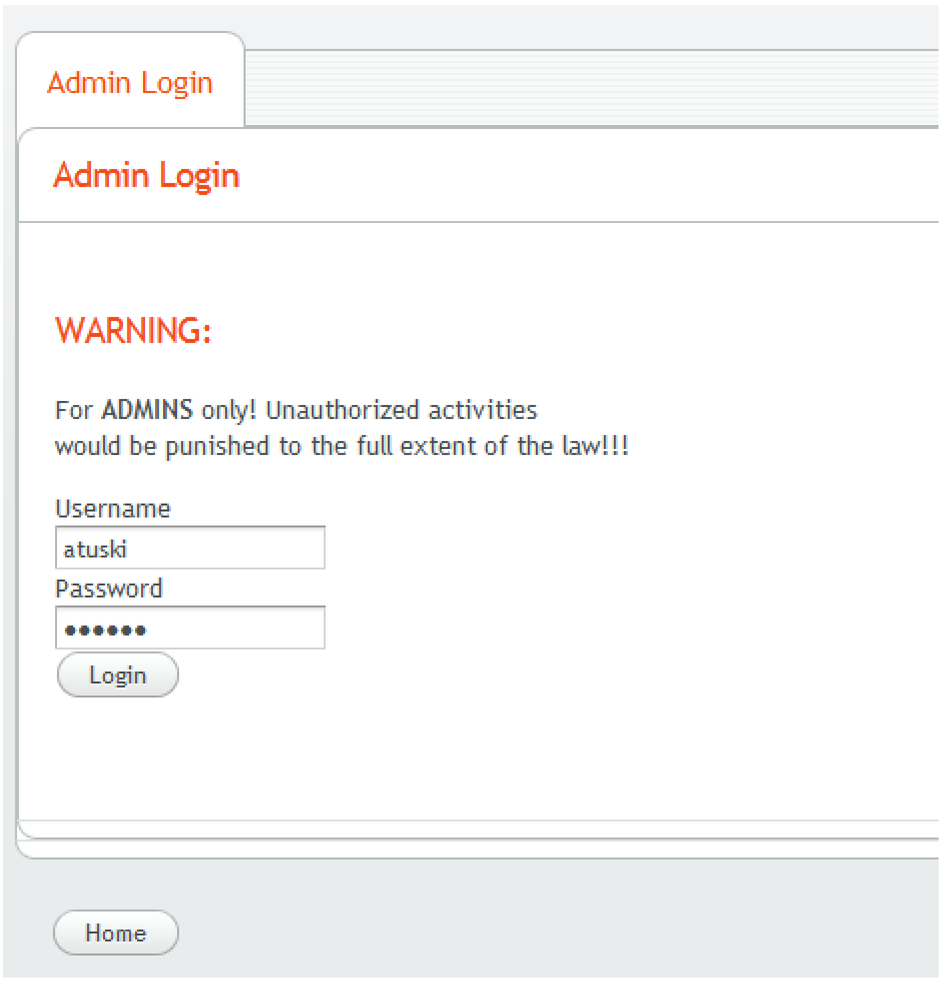
! Please enter your first name.

Figure 7. User contact details form

Once the user has filled in the two above mentioned forms, a confirmation message is displayed, assuring the user that the report has been received. Also, a similar message is sent to the user supplied email address.

The Back End – Admins' Side

The back end can be reached by clicking an admin button on the front page. This button displays an admin login page. The page consists of a form with text fields to enter admin username and password. At the top of the form is warning notice, alerting that that part of the site of solely for admin use. Non admin users are presented with a button at the bottom in other to navigate back to the home page.



Admin Login

Admin Login

WARNING:

For ADMINS only! Unauthorized activities
would be punished to the full extent of the law!!!

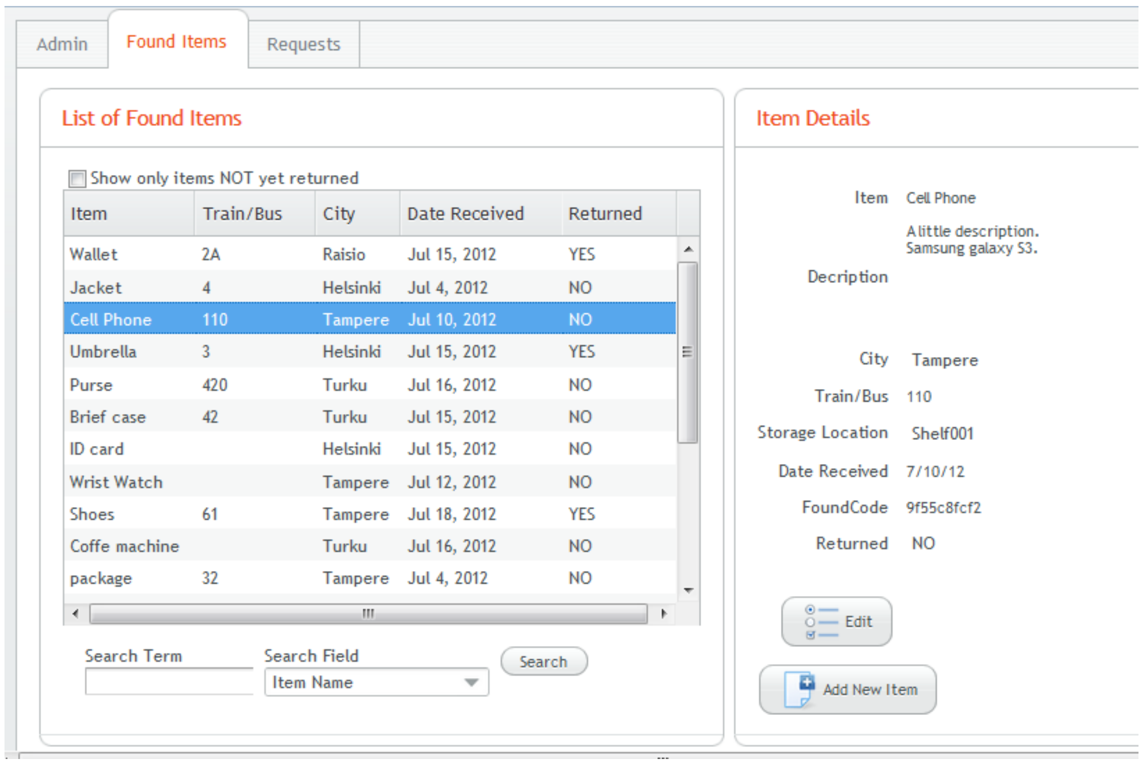
Username: atuski

Password: ●●●●●●

Figure 8. Admin login form

On a successful login, an admin is presented with three tab sheets, each of which displays different data. The first tab sheet displays information about the currently logged in admin. The current version only includes the username, the admin first name and the admin level. Also, a logout button is displayed at the bottom of the sheet, which logs out the currently logged in admin and redirects to the front page.

The next tab displays a list of found items that are currently available in the database. This page is also equipped with a search functionality to enable an admin display results based on search queries on specific fields of the database in real time. Also, there is a check box at the top that allows for the display of only items that have not yet been returned.



List of Found Items

Show only items NOT yet returned

Item	Train/Bus	City	Date Received	Returned
Wallet	2A	Raisio	Jul 15, 2012	YES
Jacket	4	Helsinki	Jul 4, 2012	NO
Cell Phone	110	Tampere	Jul 10, 2012	NO
Umbrella	3	Helsinki	Jul 15, 2012	YES
Purse	420	Turku	Jul 16, 2012	NO
Brief case	42	Turku	Jul 15, 2012	NO
ID card		Helsinki	Jul 15, 2012	NO
Wrist Watch		Tampere	Jul 12, 2012	NO
Shoes	61	Tampere	Jul 18, 2012	YES
Coffe machine		Turku	Jul 16, 2012	NO
package	32	Tampere	Jul 4, 2012	NO

Search Term:
Search Field: Item Name

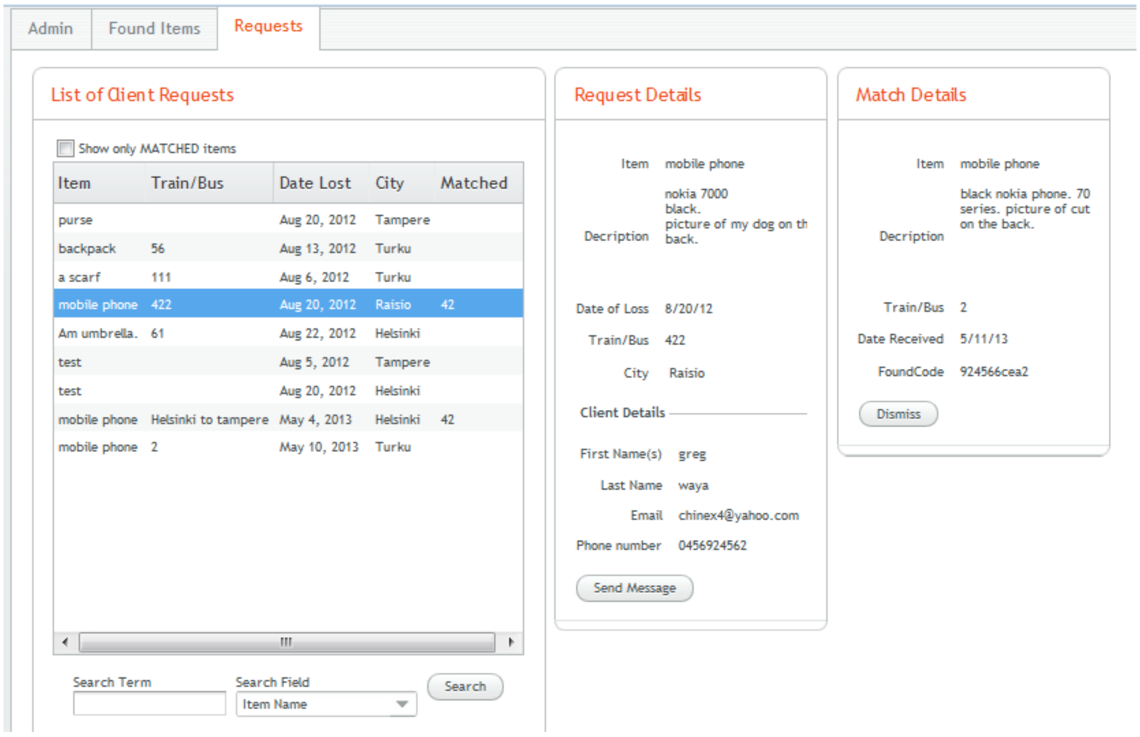
Item Details

Item	Cell Phone
Decription	A little description. Samsung galaxy S3.
City	Tampere
Train/Bus	110
Storage Location	Shelf001
Date Received	7/10/12
FoundCode	9f55c8fcf2
Returned	NO

Figure 9. Admin page showing list of found items

By selecting any of the displayed items, the admin is able to display full details of the selected items to the right of the list enabling the admin to do a thorough examination of the item details. This is needed in other to quickly search for a particular item and get full information about it easily. Also from this tab, the admin able to add new items to the database or edit existing ones. The database is updated automatically after each operation.

The third tab sheet displays a list of items that have been reported missing my users. Selecting any of the items on the list displays that item's details to the right of the page. The displayed details include descriptive information about the current item and the details of the user that sent the request. Admins are also presented with the ability to send a message to the user should the item be found. If the match engine has found a suitable match for the item, that item's details are also displayed to the far right as well.



List of Client Requests

Show only MATCHED items

Item	Train/Bus	Date Lost	City	Matched
purse		Aug 20, 2012	Tampere	
backpack	56	Aug 13, 2012	Turku	
a scarf	111	Aug 6, 2012	Turku	
mobile phone	422	Aug 20, 2012	Raisio	42
Am umbrella.	61	Aug 22, 2012	Helsinki	
test		Aug 5, 2012	Tampere	
test		Aug 20, 2012	Helsinki	
mobile phone	Helsinki to tampere	May 4, 2013	Helsinki	42
mobile phone	2	May 10, 2013	Turku	

Search Term
Search Field: Item Name
Search

Request Details

Item	mobile phone
Decription	nokia 7000 black. picture of my dog on th back.
Date of Loss	8/20/12
Train/Bus	422
City	Raisio

Client Details

First Name(s)	greg
Last Name	waya
Email	chinex4@yahoo.com
Phone number	0456924562

Send Message

Match Details

Item	mobile phone
Decription	black nokia phone. 70 series. picture of cut on the back.
Train/Bus	2
Date Received	5/11/13
FoundCode	924566cea2

Dismiss

Figure 10. Admin page showing a list of items reported missing

A check box is also located at the top of the Requests list, which when checked, filters the list to display only items that has not been successfully matched by the match engine. Also, there is a search tab at the bottom of both tab sheets. With these search tabs, the admin is able to filter the list of times based on certain fields on the times. For example, the search query "umbr" would display all items with the name "umbrella" in the list. This is also very useful to search for an item with the found code.

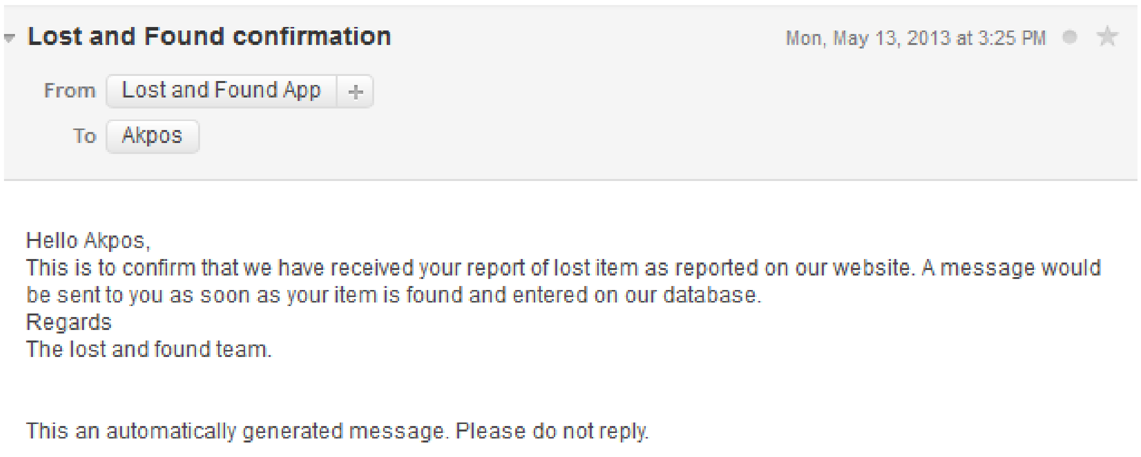
The Internal Subsystems

In order to successfully perform its tasks, the application also comprises internal sub systems each contributing to the overall end goal. These are called internal as they are not directly accessible from outside of the app. Rather; they are used internally by the application to accomplish certain tasks. Following are a brief description of these sub systems:

The Messaging System

This system is used to build and send messages, such as confirmation messages to users of the application. It is built using the Simple Java Mail framework (Simple Java

Mail 2013). With this system, the admins or the application is able to automatically send messages to users with ease. An email address was created for the project and is used at the sender of the each email that is sent from the application. Although the current version only supports sending emails, future release would also include sending text messages.


Lost and Found confirmation
Mon, May 13, 2013 at 3:25 PM
From Lost and Found App
To Akpos

Hello Akpos,
This is to confirm that we have received your report of lost item as reported on our website. A message would be sent to you as soon as your item is found and entered on our database.
Regards
The lost and found team.

This an automatically generated message. Please do not reply.

Figure 11. Confirmation email

Another important subsystem in the application is the matching engine. This system is very important to overall functionality of the application. The next topic describes this engine and its implementation in details.

USABILITY FEATURES

The site was designed with usability in mind. The current system of recovering lost items is already easy, except in few exceptional cases where it could be tedious. Also, users are already familiar with it. Hence, if users would leave the former system for a newer one, the new one has to be easier in addition to being better. This is therefore necessitates better usability. Following are some usability features that have been implemented and a brief explanation.

- Navigation and Accessibility

Navigating the site is very easy and smooth. Both on the admin and users side of the site, every other area can be reached with easy. On the users side, there are three tab sheets. Tab sheet was chosen for this because each of the other pages can be seen from every other page as well. For example, from the welcome page, the user can actually see that the next tab is for notifying of lost item because of the time, “Lost Item”. From there the “About” page can also be seen.

When the lost item page is displayed, only the form for entering information about the lost item is shown at first. The instructions buttons is right below the form, so the user do not have to look for it. For example, if a user is not sure how to fill the form, naturally they read the form all the way to the bottom, and then they find the “Instructions” button. By clicking it, they find all the information that they need to fill the form.

After completing this form and clicking proceed, then the contact information form is displayed without refreshing the page or loading a new one. This is very useful as the user can still see the information that was filled in the previous lost item form. Hence, the user is still able to make any needed corrections in the previous form before proceeding.

On the admin side, the two important item lists, Found Items List and the Requests Lists, are separated into two tab sheets. The admin can view any of those tabs in any order. There is a robust search functionality incorporated into the system on both tabs that allow the admin to search very quickly for items that match certain criteria. The use of tab sheets also ensures that the previous page is not refreshed by visiting another page.

Also, for accessibility, the load time of the site is fast. This is because the lists were created with Vaadin Table UI item. This table loads only a few items at a time. Others are loaded only when the user scrolls past the currently loaded items.

- Content

The content of the site is kept as simple as possible. From the users’ point of view, the main aim of the site is to provide the users with an avenue to notify the lost and found office of a missing item. This is made available to the user as soon as s/he loads the first page. Each content is only displayed when needed. For example, the contact information page is only loaded after completing the form for reporting a lost item. Also, the site is not cluttered with unnecessary information.

- Mistake Tolerance

It is expected that users don’t lose items very often; hence, this is not a site that the same user would visit over and over. As a result, the site is designed such that it is easy to use for the first time. This means that users’ mistakes are to be tolerated and not punished, for example. To achieve this, browser validation is implemented to check that users do not enter wrong or incomplete information, whiles at the same time, guiding users towards the right information. The input validation is implemented such that when a user enters incomplete information, the user is not able to submit, but at the same time, s/he is given a chance to correct it. For example, a user is required to enter at least one of Phone Number or Email address. If these are omitted, the form cannot be submitted but an error message is displayed below the form. However, the previously filled information is preserved, hence, the user do not have to fill it all from the scratch. Also, the area of the form when the error occurred is highlighted in red. There is also a compulsory restriction on the item name, description, etc.

- Feedback

After filling the forms and submitting, it is important that the user is informed whether or not the process was successful. This is a form of feedback. A negative feedback informs the user that they did something wrong, and a positive feedback in this case, informs the user of a successful process.

In this case, the negative feedback is the error message that the user gets when they fill in wrong or incomplete information e.g. when they omit the phone number and email fields. On the other hand, the positive feedback is the confirmation page that is displayed once the user has successfully submitted a completed form. The confirmation page assures the user that everything went well. Also, a confirmation email is sent to the user’s email address.

5 THE MATCH ENGINE

The matching engine is at the heart of the functionality of the lost and found application. The main task of the engine is to search through the database for a potential match for a reported missing item. Thus, in terms of functionality, the match engine could be broadly seen as a search engine. When a user reports an item missing, the engine needs to search through all available items in the database. If an item is found with information matching the user reported item, then that item is selected and the user notified. In an ideal situation, if an item does exist in the database that matches that reported by a user, that item should be found and the user duly notified. If a match is not found, then it means that the item has not been found.

Problem Description

Comparing data from the admins and those from the user side by side, the engine needs to answer the question, "Are these two pieces of information about the same thing?" If the answer to that question is “yes”, then a match has been found. Since the information about items on the database is all in text format, this problem is very similar to that of a text matching algorithm.

Possible Solutions

As trivial as the problem may seem, in reality it is a daunting task. In fact, enabling a computer to understand [a human] language is one of the most difficult problems in artificial intelligence (Google Official Blog, 2010). For humans, due to a well advanced natural communication and analytic skills, it is far less tricky. On the other hand, having almost no clue as to what and how humans communicate and pass informative messages with useful meanings, developing an algorithm to help the computer to solve the aptly named "trivial problem", is far from easy. For example, according to (Google Official Blog, 2010), a computer program [cannot] be written which can understand a sentence anywhere near the precision of a child.

However, over the years, so many excellent approaches have been invented to solve this kind of problem. Most of these have proven to be extensive and very effective in certain specific areas of application. The ability to accurately judge the similarity between natural language sentences is at the backbone of several applications. Areas of application include text mining, question answering, and text summarizing (Ramiz, 2009). One other very similar area is in the area of search engines, for page ranking and query analysis.

Following are a brief description of some of the viable approaches that were considered while developing this project. These can be broadly divided into three categories based on how they view the compared stings. The first category views the stings as a “bag of words” (Yiming Yang and Thorsten Joachims, 2008), hence, each word that make up the strings, is considered separately as a stand-alone word, whilst being compared to each word in the other string. The second category uses statistics measures while the last category considers the sentence or phrase as a whole unit. The latter seeks to extract the meanings of sentences and make comparisons.

- Approximate string matching

Also known as fuzzy string searching, this is the art of comparing two sequences of characters for similarities (Keng & Hui, 2006). Various methods are employed to measure how similar or how different two strings of characters are. Basically, these methods define the ‘distance’ between two strings as the minimum amount changes or altering that has to be done on one sting in order to convert it into the other. The changes are insertions, deletions and substitutions of single characters. These are normally referred to as minimum edit distances. There are various approaches that have been created in applying this methods such Levenshtein distance, Hamming distance, JaroWrinkler edit distance, etc. These are all variations of the basic principle, with Levenshtein’s the most basic. Other variations may, for example, define different weight for insertion, deletion or substitution (Keng & Hui, 2006).

As can be clearly seen, this approach does not take into account the meaning of the stings involved. The strings are merely considered as a set of characters. Hence, there is a high probability that this method would yield a false result on many scenarios. For example, Levenshtein’s distance on the words, “black” and “blck” would be the same distance between the words “black” and “back” (planet calc, 2010), whereas the first two should be closer than the latter. However, there are many areas of applications that these fit in perfectly. Examples include computational biology, text processing and pattern recognition (Dimitris Papamichail and Georgios Papamichail, 2009).

- Statistics/Probability measures (TF-IDF Measures)

These set of measures consider how often a term appear in a document, then some complex mathematical procedure are carried out in order to determine the probability that one sentence is similar to the other or how close the sentences are to each other. In the literature, it is mainly referred to as term frequency-inverse document frequency (TF-IDF). Common measures in this category are TF-IDF Vector Similarity and Novelty Detection and Identity Measure. (Stephen Robertson 2004)

One reason why this is not appropriate for this problem is that these measures are inherently for comparing entire documents rather than short sentences and phrases. Also, the complex mathematical and statistics procedure involved makes it an over kill for the problem.

- Sentence similarity measures

This approach tries to get the similarity of sentences in terms of what they mean. The fact is, when it comes to obtaining the meaning of sentences and comparing them, there are multiple facets to it (Murdock 2006). One of the prominent approaches in this category is the word overlap measure.

Word overlap measures are a family of similarity measures that measures the degree of similarity of sentences as a function of the number of words present in both sentences. Examples in this class are Jacquard similarity coefficient, simple word overlap, IDF overlap, and phrasal overlap.

Among the above listed, the simple word overlap seem to fit better to the problem at hand than the others. This approach uses a “simple word overlap fraction”. This is the proportion of words in one sentence that appear in the other sentence, normalized by the length of the first sentence (Metzler et al. 2007). In this case, the length of the sentence would be the number of words in it.

The Proposed Solution

The matching engine entails the implementation of a simple but decent and effective solution to the problem. Rather than delve into the semantics and syntactic details of phrases and sentences provided by users, it compares them in terms of individual words and their synonyms. That is, it does not try to obtain the meanings of sentences but compares the “keywords” and their synonyms with those obtained from the other. Although it would be necessary in other areas of application, like search engine results (Google Official Blog, 2010), such details are not needed for the problem at hand.

The Theory behind the Match Engine Implementation

The main idea behind how the engine is implemented is the theory proposed by the author, that given descriptions by two different people of a particular physical item, there are words that both would inherently use in describing it. If these two descriptions are matched against each other, there is a high possibility that a good percentage of words used in one description would be found in the other. If those words are not found as they are given, then mostly likely their synonyms would be. And if their synonyms aren't, then there is a high chance that the two people are describing two different items altogether.

This theory was inspired by the work of (Metzler et al 2005, 2007) on Simple word overlap fraction. This method measures the similarity between two sentences as a fraction of how many words in one sentence appear in the other. This value is then normalized by the number of words in one of the sentences. In addition to that, the match engine in this project also considers the possibility of synonyms as well. However, word match with synonyms are given a less weight (score) while performing comparison. The idea of applying synonyms comparison is largely influenced by a similar but much more complicated approach used by the Google search engine for processing user queries (ArnoldIT, 2009).

Synonyms in Google Search Engine Algorithms

According to (Google Official Blog, 2010), synonyms affect 70 per cent of user search queries. Hence, over the years Google has been able to improve their search engine page ranking algorithms and search results through clever application of synonyms. In fact, the engineers at Google have so mastered the art that they have a patent on a very effective algorithm based on the use of synonyms (ArnoldIT, 2009). In this method, phrases in the user query are replaced with synonyms, forming a series of other queries which are then used to generate appropriate search results.

The Theory Applied

Among all the information collected from users in the process of notifying the hypothetical lost and found office of a lost item, the matching engine only makes use of three pieces, namely, "item name", "item description" and "City" fields.

The main use for the name field is to provide a sensible limit for the search query that the match engine uses to retrieve possible match candidates from the database. Say, a user just notified that s/he lost his/her cell phone on a train from Helsinki to Tampere on the 4th of March. When the engine tries to search the database for possible descriptions that match the one provided by this user, it would make sense not to include rows with name field value, "a purse", in the search result. Instead, the engine queries the database for rows that has "cell phone" as the name field from the database.

However, there is one obvious problem; there are different ways to refer to the same item - synonyms. For example, if the admin enters "mobile phone" instead of "cell phone" in the name field for the item. This would mean that that row would be skipped and the purpose for the search defeated. One possible solution would be to provide a limited range of item names and categories of items for users to choose from. Not only is this solution highly limiting, it is also not scalable and ultimately not feasible. One reason is, the engine would have to "know" and take into consideration every item that could possibly be lost in a train, a bus or anywhere else. Then, they would also have to be categorised and every possible synonyms for each of those item names would have to be entered and presented to the user to choose from. Users may have a difficult time choosing which name fits best to an item and thus, be discouraged from using the app.

As was earlier noted, obtaining synonyms for words is not a trivial task. At Google, using complex techniques and algorithms, language models are built that enables the search engine to find alternatives for words (synonyms). However, this technique involves billions of web documents and historical search data obtained from users' previous search queries in order to build these language models that are used to obtain the synonyms (Google Public Policy Blog, 2008). This amount of information and the processing capacity is not available for this project; hence a much simpler approach has been adopted. This approach involves using the WordNet dictionary to provide the synonyms.

The first thing that the engine does with a user request is collect the so-called "keywords" from the name field. From the second survey conducted prior to developing the engine, it was found that users generally use both informative and non-informative words when naming an item. Chief among the "not useful" words are English articles and personal possessive pronouns. For example, a user may enter phrases like "an iPad", "a backpack", "my wallet", etc. as names for lost items. Clearly, the articles and possessives, although they make the phrases more readable to human readers, do not give further information about the item in question. Also, they would not be needed by the match engine. Hence, they are stripped from the name field value prior to working with those items. Also, punctuations and, extra spaces and new line characters are removed from the string in this stage.

Next step is to collect synonyms for those items. For example, synonyms for "mobile phone" would be "cell phone", "cellular", etc. These are known as keywords in the engine. The keywords are then added to the query sent to the database. Hence, rows with any of those keywords would be included in the search results as well.

Even though a little child can easily identify synonyms, getting a computer to do the same is not a trivial task. At Google, various complex techniques are employed in order to extract synonyms for words and phrases. These techniques involve analysing petabytes of web documents and search data in order to get an understanding of the meaning of words in various contexts. (Google Official Blog, 2010)

In this application the Java API for WordNet Searching (JAWS) is used to retrieve synonyms from the WordNet dictionary. The WordNet dictionary is a simpler and yet very efficient alternative for this purpose.

Using the above described procedure, the engine is able to extract keywords from the description field as well. For example, in one the surveys carried out during the development of the project, a user describes her lost mobile phone thusly, *"nokia 7000 black. picture of my dog on the back."* When the engine extracts the keywords from that description string, the following output is obtained, *"Nokia 700 black picture dog back"*. As can be seen from the aforementioned output, apart from punctuations and articles, other non-informative words such as my, there, his, it, etc., are also removed in the process. Assuming the admin gave a description thusly, *"black nokia 7000 with picture of cute dog on the back"*, when this description is passed through the engine, the following output is obtained, *"black nokia 700 picture cute dog back"*. Hence, it is clear that those descriptions now closely match.

Next, the engine compares the keywords obtained from the user request item with each item obtained from the admin query search result. The numbers of keywords found in the user request is used as normalizing factor over a hundred per cent. An arbitrary threshold of seventy per cent is used to determine if a match is found. This means, if a description from the database matches up to seventy per cent of that given by a user, then a match is found.

User Notification

When a match is found, the user is notified using the supplied contact information. The current version of the web application only supports email notification. This is mainly because the author was limited both in terms of hardware and software during the project development.

The email notification sent to the user contains useful information necessary to reclaiming back the lost item. One of the supplied information is a “foundCode”. The foundCode is a unique code that is used to identify a particular item on the database. The code is important, e.g. for the following situation. Say, two similar items, e.g. identical phones, were identified and the appropriate owners were notified. When each user visits the office for reclaim, the admin is able to easily locate the particular item for a particular user, without having to show both phones to the user for him/her to identify. In other words, the foundCode uniquely identifies a particular item from other items. The code is a unique random string of characters automatically generated in the database when a new found item is created.

Also supplied is the office location where the item is located. The project assumes that there are lost and found offices in all the cities where the lost and found service is available. With the location information, the user knows just which of the offices to visit for reclaim. For example, a user loses an item while travelling from Helsinki to Tampere. If the lost item was found in Helsinki, then that item would be in the Helsinki office. Also, there could be more than one office in the Helsinki area. Therefore, the office location information tells the user just the right office to visit or call.

Other information included is the phone number of the office and the email address. This is useful for the user to get in touch with the office and ask for more information. Also, the user could then supply information as to where they want their item sent to, in case they are not able to visit.

Optimization

The task of the match engine is rather system resource intensive. It would slow down the whole system to run such a task in the main thread. Also, it would be a waste of system resources to run the program continuously. Hence, it is necessary to only run the program when an item is entered into the database, either as found item or as a missing item entry. One possible solution to this would be to create a trigger on the database that can call the program on such occasions. Unfortunately, this is only possible with the Oracle DBMS. Using the MySQL DBMS, this application uses an alternative solution.

In this solution, the match engine runs on a separate thread in the background. As soon as the task is done, the engine program terminates and its thread is killed. The thread is recreated again, only when a new item is entered on the database or when an existing item is edited by the admin. Also, since the program does not run on the main thread, it does not interfere with the normal activities of the entire application. This is necessary, as it protects against crashing the whole system, should a fatal error occur.

Limitations of the Match Engine

Although the match engine is very effective at what it does, there are a few limitations. However, given enough time and resources, these limitations could be eliminated.

- Dealing with Spelling Errors

In performing its tasks, the engine does not take into account the inevitable occurrence of spelling mistakes both from the users and from the admins. This could to some extent alter the decision of the engine. For example, if the engine tries to look for the word, "black" misspelled as "blck", in which case, a match may not be found.

- Use of Informal Words

Another limitation that is similar to the above described is the use of slangs or colloquial words to describe items. Sometimes, there are names given to items popular among certain people, which are not official names for those items. For example, in Nigeria, among pidgin English speakers, certain types of shoes are referred to as “rides”. If a user enters that instead of shoes, then that item would not be located by the match engine. However, it is clear that this type of situation is rare. The system applied by Google for obtaining synonyms would avoid this kind of problem. However, that approach is almost impossible to implement in this project.

This would be a useful feature. For example, according to (Search Engine Land, 2012), Google uses a dictionary of words and their common misspelled variations to both correct user query and suggest other queries with the correct spellings. Also, misspelled words act as synonyms as well. However, incorporating such functionality would increase the complexity of the engine exponentially. Given the efficiency of the current implementation, it is safe to leave the engine as is. Nevertheless, this feature could be implemented in future versions.

- Omission of other informative fields

The match engine does not make use of other informative fields such as the "date" and the "Train/Bus" fields. This is intentional. The main reason being, in certain situations, there is a high probability for inconsistent dates between when an item is reported missing and when it was brought to the lost and found office. For example, a user may have forgotten when s/he lost an item and reports a wrong date. On the other hand, the item has long been found and brought to the office; hence, the admin has entered an earlier date. In such situation, if the engine considers date of loss, then that item would not be searched. The Train/Bus field is also left for the admins as well. This is because the engine is not limited to finding only items that were lost in trains and buses but also in other places.

However, those fields are left there for two reasons. The first reason is, the admins could use that information to make decisions, should a person visit the office for claim for an item, without prior notification by the application. This situation is quite normal as users generally visit the lost and found office when they lose an item, whether or not they have reported the item missing on the web site. The second reason is allow for the possibility for future improvements.

- Describing certain items is difficult

There is another limitation that is inherent to the theory on which the engine is based off of. Certain items have limited ways of describing them due to numerous reasons. A good example would be keys. Mostly when users describe keys, they, being the owners of the items, may use phrases like, *“A bunch of keys containing my house keys, my car keys, my work key, etc.”* On the other hand, the admin would most likely not know which is a car key or a house key. If those two descriptions are matched, there is a high probability that the match would fail. However, this limitation could be mitigated if users and admins alike describe things based on their physical appearance. However, it is obvious that this is a rather difficult goal to achieve as, although the users would be instructed accordingly, still it is not guaranteed that they follow the appropriate instructions.

6 DESIGN AND IMPLEMENTATION DETAILS

This project was designed top down systematically, adding one feature at a time. Each part of the system was developed independently and separately. The Vaadin framework has a rich collection of UI items that made the development easy.

Main UI items used

Following are a brief description of some of the important Vaadin UI items that played very important roles in the design and implementation of the application.

- Vaadin SQL Container

This is a container class implementation of the Vaadin container interface, which provides access to data stored in a database. SQL containers can be created either by using a pre-made TableQuery or FreeformQuery. The latter involves writing own query to fetch the data from the database.

With this container, data can easily be read, manipulated and written to the database. Each row on the database is accessed through the RowItem object and is identified with a RowId object, which is an object representation of the primary key of the underlying table. Also, each column is easily accessible through a ColumnProperty and its value is obtained with the value access method of the ColumnProperty object. (Vaadin, 2012)

This container class plays a very important role in this application. All access to the database in this application is done through this class. The classes in the customUI package of the project use SQL containers as the source of data. Also, each table class in the database package extends this class.

- The Vaadin Table component

This is a user interface component that is mainly used for displaying data organised into rows and columns in a tabular form. The source of the data to display is any implementation of the Vaadin Container class e.g. SQL Container. A JavaBean can also be used as a data source. The Table is bound to the data source and, is automatically updated when the table is modified and vice versa.

Apart from being very powerful, the Vaadin Table is also very flexible. For example, the developer can restrict the displayed column as well as restrict the number of rows displayed at a time. Also, the Vaadin Table is highly optimised. For example, for obvious performance reasons, the entire data from the data source is not loaded into the browser but are loaded as need e.g. when the user scrolls past the displayed set. Also, the column headings can be modified according to the application's need.

All tabular display in this application is built with this class. The FoundItemTable and RequestItemTable are both subclasses of this class. The FoundItemTable displays a list of items that are found and the RequestItemTable displays items that are reported missing by users. Although each field of the table rows can be edited directly from the table, they are actually edited on the separate UI object.

- The Vaadin Form

This UI item is used to display item details. It displays one item at a time. The form can be connected directly to a Vaadin container or can be connected to a row in a Vaadin Table item. In this project, this UI item is mainly used to display details of selected items in a table. It can also be connected directly to a database container. When an item is edited, its value is automatically updated in the data source, which could be a Vaadin Table item or a Vaadin Container object.

The Vaadin Form can also be used to collect data from user input and stored in a connected database. For example, the form used to collect user contact information, missing items details, etc. are built using subclasses of this item.

- The Vaadin Layout Items

These are UI items that are used to organise other UI items for displaying. There are two main subclasses that were used in this project, the HorizontalLayout and the VerticalLayout classes. As their names imply, the HorizontalLayout organises items horizontally while the VerticalLayout organizes them vertically.

Most of the pages that make up the application are subclasses of one of these classes. For example, the About page is a subclass of the VerticalLayout class. Hence, each item on the page is displayed one on top of the other.

Packages

The classes that make up the application are organised into packages for readability and maintainability purposes. Following are a brief description of each package and some of the important source files (and classes) they contain.

1 The lostfound package

This package contains only one source file, the lostFoundApplication. This class, being a subclass of the Vaadin Application class, serves as the entrance to the application. This file also holds static references for use by other files in the application. For example all database tables are reached through this class. The class also holds the sub threads that run the email system and the match engine.

2. CustomUI package

In the Vaadin framework, it is possible to create custom UI items for repeated uses. This package contains such UI items. There are three classes in this package representing selectable items in the database, namely, CitySelect, ReturnedSelect and StorageSelect. Creating a custom UI is necessary as the data for these select UI items are located in the database. Each of these UI items is a subclass of the Vaadin select item backed up by a Vaadin SQL container, which is a container for rows in the database.

One great advantage is that new items can be added on the database without any modification to the source codes and they would automatically be presented to the user. For example, a new select option can be added to the CitySelect item by adding a new row to the underlying database table without any modification to the source codes. The display would be automatically updated.

3. The data package

The main idea for this package is to take care of data and their manipulations before they are stored in the database and / or after they are retrieved from the database. There are two source files on the package, namely User and SearchFilter.

The User class contains information about the currently logged in admin. An instance of this class is stored on the server alongside the session information in order to identify the currently logged in admin. This class is made light for performance reasons.

A struct data structure would have been a better alternative due to its light weight but Java does not support it. However, the JVM performs optimizations on such classes, making them as light weight as a struct.

The search filter class takes care of organising a search query on the list of displayed items into manageable units. This class contains a name of the property to apply the search, the display name and the search term. This is very useful for creating search and applying filters on the database. With this, a search filter can be created and applied in several places, without having to recreate it. Also, a search filter can also be passed from one class to another.

4. The database package

The main function of this package is to organise the classes that take care of the database related tasks such as database connection, database tables, etc. Also, if the database is changed in the future, the change would only have to be made in this package and every other part of the application would stay the same. The LFDatabase class in this package holds the database location and login credentials for the currently used database.

Each table in the database is represented as a class in this package. This is so that the database tables can be accessed as objects of classes. Also, there would be no need to recreate a link to a database table whenever it is needed. Instead, an object representing that table is used. This is in line with OOP practices.

Each table class is a subclass of the Vaadin SQL container. Hence, apart from providing connectivity to a database table, this object has the advantages of a Vaadin container class. For example, the class could be used directly as a data source for a table. Also, each row can be accessed easily through one the Vaadin container methods.

Another important class that make up this package is the AppTableContainers class. This class holds a reference to each database table object. This is an optimization technique. The SQL container classes are heavy and creating a new class each time one is needed is resource consuming. Hence, instance objects of each of these classes are created at start up and, a reference is stored in this class. Each time any of these tables is needed, a reference is returned from this class. Also, certain tables which are never modified, such as city table container, are made static. This means that only one object of this class exists at all time for the entire application; hence, eliminating unnecessary duplication.

5. The UI package

This package contains classes that represent user interface items. Every item that is displayed on the browser is a class in this package. Also each page is a class in this package.

6. The matcher package

This package contains the match engine class. Although it only contains one class, it is highly extensible. If future modification is done on the current version of the engine, it would be added to this package.

7. The message package

This package contains classes that are used to send messages to users. The current implementation only supports sending email but can be extended to send SMS as well. It contains the Message class which represents the message that is to be sent to the user e.g. a confirmation message. It also contains the EmailMessenger class. This class uses classes from the Simple Java Mail package to send emails. Also, through this class, certain predefined messages such as confirmation messages can be easily retrieved using function calls.

The database

The database was implemented in MySQL 5.5.25a. Figure 11 show a database diagram of the database tables. Each piece of information that plays a role in the application is organised into appropriately named table. Each table has an 'id' column that serves as the primary key. This is a necessary requirement imposed by the Vaadin SQL container class.

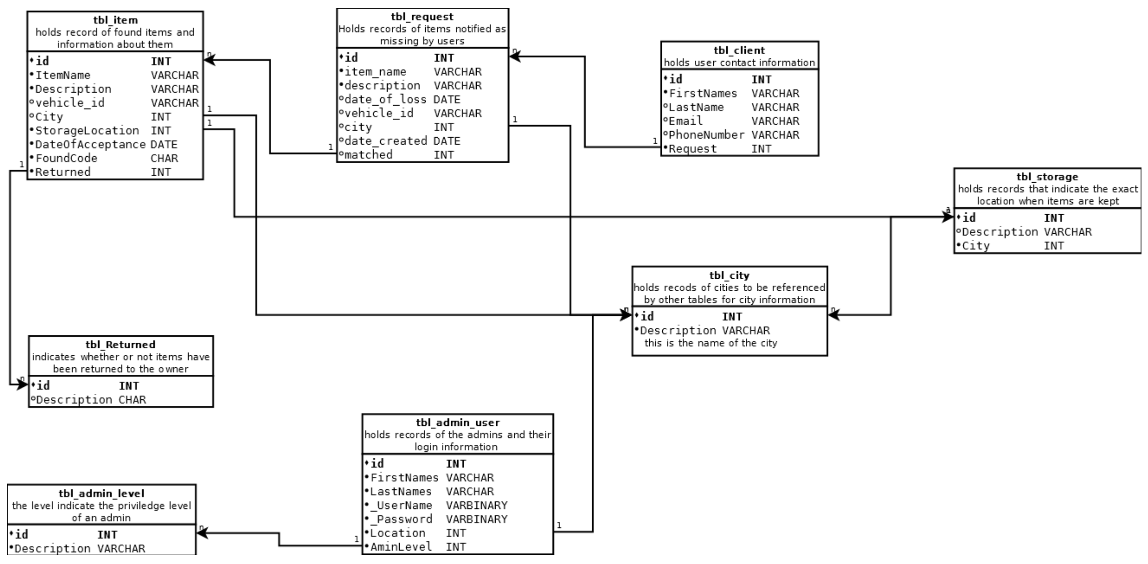


Figure 12. Database diagram of the underlying database tables

The tbl_item table contains information about items that have been found and kept at the lost and found office. The tbl_request table contains information about items that have been reported as missing by users. The tbl_client table contains names and contact information from users. This information is used by both the admin and the system to contact users when their items have been found.

Database Normalization

This is the process of organising the tables and fields of a database such that redundancy and dependency is minimised. It involves breaking down the database into tables and creating relationships between them in a meaningful way (Microsoft, 2013). There are various steps and forms of database normalization such as eliminating repeating information. Database normalization is a wide and complex topic and for the purpose of this project, only the basics are needed.

The main normalization undertaken in this project is at the client and request item tables. For example, the client table is stored separately from the item that corresponds to what that client reported. Also, to avoid repetition, the city is stored into a separate table. This is because there is a city attribute in the item, storage, request and admin user tables. Storing this information in those tables as fields would lead to unnecessary duplication of data. Hence, the city is stored in a separate table and the appropriate tables reference this table as foreign keys.

7 THE TOOLS AND TECHNOLOGIES

In the course of developing this project, including the surveys and the web application itself, several tools and technologies were employed. Following is a brief description of the technologies and tools used.

7.1 The Programming Languages

The web application itself was developed in Java. PHP was used to programm the surveys' server side and the client side was built using HTML and JavaScript with the jQuery library. PHP was chosen for the server side programming due its simplicity and ease of use. Each survey needed to be up and running in the shortest time possible. Experience gained from using the language in previous projects gave the author a big advantage during the survey development.

7.2 The DBMS

A database is a collection of related data or pieces of information organized in such way that makes sense. A database also entails relationships and connection between those data. A database management system (DBMS) is a piece software that allows for managing those data. Managing a database includes actions such as creating, editing and updating such data. It also entails storing and retrieving (reading) such data.

There are various types of DBMS's such as hierarchical databases, Network databases, relational databases, object-oriented databases, etc. The relational database is one of the most popular and the easiest DBMS.

(PennState 2008).

7.2.1 MySQL

The MySQL is a relational database management system. It was originally developed and owned by a Swedish company called MySQL AB and was later bought by Oracle Corporation (MySQL, 2013). The speed and efficiency of MySQL closely rivals that of very expensive proprietary alternatives.

The databases both for the surveys and the application were built in MySQL version 5.5.25a.

7.2.2 Choice of MySQL

The MySQL DBMS was used both for the surveys and the web application itself. The reasons include:

- It’s popular

MySQL is the world’s most popular open source database (Forrester Research, 2008). This is not a reason in itself, but it means that MySQL has a large user community. Hence, if one runs into problem, s/he can easily find solution on the internet. Also, there are numerous great books that have been written, treating every aspect of the DBMS.

- Open source

The MySQL DBMS is constantly improving due to open source community. Also, for such purposes as that of this project, one does not need to worry about licensing issues.

- Easy to find host

It is easy to find a web hosting company that supports MySQL. This is partly due to its popularity and availability and is mainly the reason why it was used for the survey pages. It was necessary to have the pages live without much hassle.

7.3 The Vaadin Web Framework

The Vaadin framework is essentially a library for developing rich internet applications. While release 6 was used for this project, the current release as at the time of writing is release 7. A rich internet application is a web application that has many of the characteristics of a normal desktop application but run on a web browser (msdn, 2013). This is achieved with a virtual machine, JavaScript or a browser plug-in like Flash (keynote, 2013).

The framework consists of a server-side library and a client-side engine (Vaadin, 2013). The server side framework builds the web application while the client side engine is the platform on the web browser in which the application is run. The client engine is built with JavaScript. It communicates with the server using the AJAX technology.

Formerly known as IT Mill Toolkit, work on the framework already started as early as 2000. It became an open source project in 2007 and was renamed to Vaadin in 2009 (Vaadin, 2013).

7.3.1 Choice of Vaadin

Vaadin was chosen for this project for the following reasons:

- Vaadin uses a server-driven programming model

This means that the developer does not need to worry about the client side programming. All the program logic that the application needs would be centered in one place, allowing the programmer to focus on the application and its logic without worrying about how the client side would work or how it would be rendered. At the same time, the framework still allows flexibility, in case the needs arise; the programmer is able to alter certain areas on the client side, albeit also from the server side. (Vaadin, 2012)

- Moderate learning curve

Although the framework is extensive, covering every area of web development, the core of the framework can be learned in a few hours, provided they know the Java programming language or any other OOP language. The programmer does not need to know so much in order to get started. Everything else can be picked up as are needed.

Also, the developer needs little to no knowledge of client side programming and technology such as AJAX in order to develop an application with the framework. This is because it is taken care of transparently.

- Client side security

Client side security can be a very daunting task. Even after extensive works, it is still not sure that the client side is safe. There are varieties of ways that malicious users could attack a web site, especially through JavaScript (studymode, 2013). The Vaadin client side engine takes care of security for the application transparently.

- No need for separate business logic coding

The business logic of the application can be incorporated right into the application code. No need for a separate files or codes to handle certain business logic.

- Vaadin is open source

The framework itself is open source. However, depending on the purposes, licensing may apply. For the purpose of this project, this is a desirable feature. Also, the fact that is open source means it has a good user community for answering questions that may arise while using the framework.

- Browser support

A web application developed using the Vaadin framework can run on any modern browser. This is very important as the developer does not need to do extra work in order for it to run on different browsers. This is taken care of by the Vaadin client engine.

More importantly, the Vaadin web app does not need a browser plugin such as Flash. Users are sometimes skeptical about installing a plugin in order to use an application as it may introduce a potential threat to the user computer. Instead, the Vaadin client engine runs on JavaScript only.

There are other Java-based frameworks as well, such as:

- Play!
- Apache Click
- JBoss Seam
- many more

7.4 Version control – SVN

In software development, there is need to manage the source codes and other files, and the changes that they undergo. This is known as source control. Various methods and technologies are available for this purpose. One of the main reasons why this is needed is for backup purposes. Changes to files in the project are saved to a server and can be retrieved and edited at any time. Also, the developer can revert back to a previous version at any time (Ben C. 2013).

7.4.1 Apache SVN

This project is hosted on Google code and managed using the Apache subversion (SVN) system. Subversion is one of the various methods for managing source codes and the changes they undergo. As at the time of writing, the most popular is GIT. While GIT is mostly fitted for large project, a small project such as this could also benefit from its numerous advantages. However the author, having more experience with SVN, decided to use it instead.

7.5 The WordNet Dictionary

This is a large lexical database for English. The database and its accompanying program, basically performs both the functionalities of a dictionary and a thesaurus. Words are grouped into synonyms called sysnets, based on their meanings and word forms. Using WordNet, a developer can perform a variety of operations such as obtaining the synonyms, meanings, word forms and their application, etc. on a word or set of words (WordNet 2013).

7.6 External Libraries

Some external libraries were added to the project during development. A software library is a collection of implementation of certain desired behaviors. When those behaviors are needed, the appropriate libraries are included and the programmer calls the objects and functions therein, making the process of software development easier and faster.

In the course of developing this project a few external libraries were used. Below is a list of such libraries and brief descriptions:

- Simple Java mail

Formerly known as Very Simple Java Mail (VeSiJaMa), this is lightweight mailing framework built as a wrapper around the JavaMail SMTP mailing API. It is very easy to use due to the high level abstraction of the JavaMail API (Simple Java Mail 2013). Emails can be sent very easily without having to deal with unnecessary details.

Using the Java Mail API directly is very tedious, time consuming and error prone. With this library, the developer just sets a few parameters, such as the sender and receiver email address, the sender password, etc., after which, emails can be sent with ease, using a few function calls. This library is one of the simplest libraries for this purpose and it is open source (Simple Java Mail 2013).

- MySQL connector/J

This is a JDBC Type 4 driver, which means that it is a pure Java implementation that does not rely on any MySQL client libraries (MySQL, 2013). This library enables connectivity between a Java client program and MySQL database. The version used for this project and the accompanying surveys is version 5.1.25.

With this library the developer can log in to, retrieve and manipulate data from a MySQL database. It was used both by the Vaadin framework and also used directly to access the application database.

- Java API for WordNet Searching (JAWS)

This is an API created by one Brett Spell that allows Java programs to retrieve data from the WordNet database (JAWS 2009). This library, together with the WordNet database proved invaluable to the functionality of the application, especially the match engine. This was used in the match engine, mainly to search for words and synonyms from the WordNet database.

8 WEB APPLICATION SECURITY

Any web application is faced with a number of threats. These threats have to be prepared for and prevented. This is not a trivial task and even after extensive security measures have been put in place, there is still a possibility for a successful attack. This is mainly because over the years, attackers have developed highly sophisticated tools and methods for carrying out malicious activities.

Although laws all over the world enforce stringent penalties for such activities, this does not seem to deter highly motivated attackers. To deal with this the rule of thumb is to reduce the surface of attack by reducing amount of damage that can be done to the system should an attack be successful. This is analogous to how the Linux OS works. For example, a virus may successfully enter the system but would be useless as it needs to be granted execution permission by the admin for example, in order to be able to execute. Even then, it may be able to inflict any damage, as it may not have the appropriate rights and privilege to access certain parts of the system.

POSSIBLE ATTACKS

The two most popular security attacks on web applications are cross-site scripting and SQL Injection attacks. Both of these vulnerabilities are automatically taken care of by the Vaadin framework and hence the developer does not need to worry about them. It is however advisable for the developer to work within the boundaries of the framework in order not to reduce the security level of the system. Below are some of the popular attacks that can be launched against web applications.

- **Cross Site Scripting Attacks**

Also known as XSS attacks, cross-site attack makes use of malicious JavaScript codes to gain access to users' cookies and session data. This attack accounted for about 84% of all security vulnerabilities documented by Symantic as of 2007(Symantec 2007). It involves sending malicious JavaScript codes along with "trusted" web content to users. When these codes are executed, the malicious user may then gain access to sensitive browser data such as cookies. These data could be sent to a server somewhere for later user by the attacker.

This kind of attach is inherently taken care of by the Vaadin framework due to the fact that the pages run on customized JavaScript engine that only understands and executes custom data from the Vaadin framework. Also, the server and client side are always in synchrony. Hence, any alteration done on the client side would automatically be detected by the server and the session would be dropped.

Also, this attack is less likely to occur in this application as the attack is targeted at sites where users view other users' input such as comments in forum sites. That way, a malicious attacker could inject some JavaScript codes into a comment, for example. When other users attempt to view these comments, those codes are loaded and executed, possibly compromising the user's system and stealing sensitive cookie information. In this web application, users do not view other users' input; hence, this kind of attack is not likely to occur.

- **SQL Injection Attacks**

The next popular security attack is the SQL injection. This attack is mostly targeted at data driven web application (msdn 2013). This attack is carried out by including SQL statements in an entry field and sent to the web site. If such statements are not filtered out and sent to the database that powers the application, such SQL statement can be executed, causing effects such as damage, dropping of a table or the entire database, dumping of sensitive data to the attacker, or something worse. One popular method used by developers to mitigate this kind of attack is to perform appropriate filter on string fields that are sent to the server before they are passed to the database. Most times, this attack is carried out alongside with other forms of web site attacks such as DDoS attacks (InfoSec aXioms 2011).

The Vaadin framework actually transparently takes care of this kind of attack by means of SQL containers. In the implementation of this project the database queries were handled solely by the SQL containers. Queries are not sent directly from client side. Also, the fields available at client side for making changes to the database are created in the framework using customized forms created on the server side, which are not accessible from the user side.

- **Other Security Vulnerabilities**

Other security vulnerabilities exist such as Denial of Service (DoS) attacks, Distributed Denial of Service (DDoS), DNS hijacking, database management system-specific attack, etc. Most of these are automatically taken care of by the server. This is mainly because these attacks are mainly targeted at the server itself, rather than the application it runs. However, due to the currently sophisticated tactics of modern attackers, they are known to experience a fair amount of success in attacking even industrial size web websites such as Google (pcworld, 2009).

Security Measures Taken

Although the framework takes care of some security vulnerabilities, a few more precautions and measures are put in place. This is necessary as there can no limit to the possibility of an attack. Below are the security measures taken and a brief description.

- Limited Database User Privileges

The application accesses the database as a user specially created for that purpose. This user has limited privileges. For example, this user cannot delete any entry from any table in the database, it cannot create an admin user, etc. Hence, if the application is compromised, as long as the attacker does not have access to the root user of the database, there is great limit to what s/he can do. Nevertheless, if the system is compromised, the attacker may for example be able to go through the rows available on the database. This could allow the user to create a request for a missing item using one the descriptions on the found items table and make claims for items that do not belong to him/her.

- Data Encryption

Another security measure taken on the database is to encrypt the admin users' passwords and usernames. The applied encryption is the Advanced Encryption Standard (AES) encryption. This is currently the most secure encryption provided by the MySQL DBMS (MySQL, 2013). The main advantage of encrypting sensitive data stored on the database is, in the event that the database is compromised, the attacker would still not be able to get the stored information without the key. The key is stored away from the database.

9 THE DEVELOPMENT METHOD

A software development method or model is an imposed structure on the process of developing a piece of software. It is the pattern that the developer decides to follow while developing the software. There are many software development models, each fitting better to different situations. It gives the advantage of having a structure to how things are done in the process of implementing a software project. It also gives the developer a sense knowing what stage of the development process s/he is.

From the start of the project, the full scope of the project was not clear. Requirement kept changing and new features are discovered. For such a project, a development process that would fit well into such a scenario and also be able to cope with the changes as the project evolved is needed.

Incremental build model

This is a software development model where the software is designed, implemented, integrated and tested, and the process is repeated whilst adding more until the product is fully completed (Pressman and Roger 2010). It is a derivation of the waterfall model, with the iterative attribute of prototyping and, is a very popular model used by most commercial software development companies (softdevteam, 2010). Fig.3 depicts a pictorial representation of how it works. It should be noted that this is not a de facto depiction of the model. When compared to diagrams given by some other authors, there may be slight differences. For example, some believe that the test and analysis should be together.

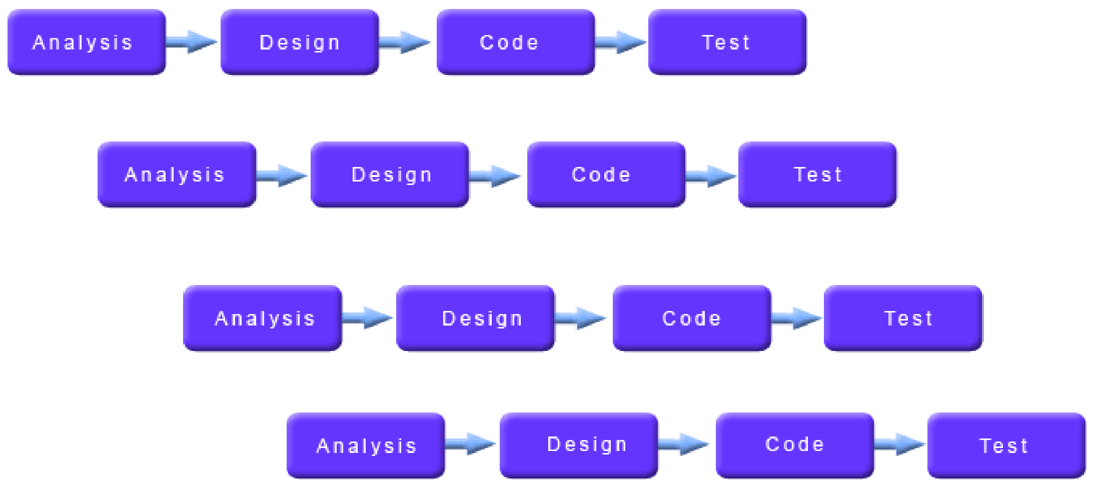


Figure 13. Incremental build model

This is the model used in developing this project. Although there are many other software development model, this model was chosen for the following reasons:

- Flexibility

This model allows for changes to be easily added to the current software at any time without causing too much imbalance in the existing system.

- There are intermittent results

There is a working project each time a build it reached. This means that each build is actually useable and if tests are carried out on the current working build, the entire project would be better understood. This also means that the developer does not have to wait until the final project is released in other to see results. This is a very desirable feature as, without intermittent testing not all parts of the application can be effectively tested. For example, if tests are only carried out on the system when it is fully completed, some parts may be omitted in the testing phase and thus the product is shipped with a bug.

- Easier risks management

This is because the risks are broken into smaller pieces alongside the software. That is, during each iteration the risks are identified and dealt with in smaller pieces.

- Fits perfectly for a one-man team

In a one man team, every aspect of the software development has to be taken care of by the same person. Hence, there is a great need to use an approach that makes for easy understanding of the full scope of the software at every stage of the development cycle.

Also, the developer does not need to understand the full scope of the software project in order to start working on it. The part currently understood is implemented and tested. The other parts surface during and / or after testing, and hence they are taken care of in the next iteration.

- Fits better with a Subversion'ed software project

During development, this project was saved incrementally in Google codes using Subversion. This means for example that after each iteration, there may be a need to revert to a former project state, which in this case is most likely a working build. Hence, as the software is tested and problems or limitations are detected, the developer may need to revert to a previous version and continue in a different implementation direction, thus following the incremental build model naturally.

How it was applied in the project

Starting from the requirement stage of this project, this model was applied all the way until it was entirely completed. The requirement stage, which is an integral part of any software development model, was carried out with only a few available information or knowledge about the problem. This was further analysed, leading to the analysis stage for the first build. Analysis was further carried out using a survey in order to further understand how to go about the implementation.

Based on the results of the analysis realised from the survey, a design was done that led to the implementation of the first version of the software. Tests was carried out on the first build that led to the realisation that the intended approach does not sufficiently solve the software problem. Also, it was understood that the current approach would eventually become too complicated both for the developer and the users. For example, this version added redundant fields to the user report fields that were not needed.

This further led to a series of research which fell into the analysis stage. Having a better understanding of the project, development was directed towards the second version. This version was then tested and hence, the need for a match engine was discovered.

The match engine was further researched, developed and ultimately added to the project. The entire project was then tested using date obtained from yet another survey carried out for the same purpose.

10 CONCLUSIONS

In the end, a working system was developed according to problem specification. The full extent of the possibilities associated with this project could not be reached due to time constraint and limited amount of data available for testing. However, the performance of the system in the tests that were carried out after development was impressive.

The development process was indeed a complex task. The author had to do most of the research by himself with very little help. More so, the problem being solved took a number of dimensions and hence the author had to work with various languages and libraries both in the application itself and the accompanying surveys.

One aspect that proved very difficult was the surveys. People were generally reluctant to take the surveys. This resulted in fewer results than expected. The author had to work with the little that was available. However, the project turned out to be a success in the end.

This is the most complex project that the author has had to undertake and the learning outcome is tremendous. The project greatly increased the author’s experience in the software development process. The project also gave the author an opportunity to demonstrate his knowledge and experience in coding in different languages. The project also increased the author’s problem solving ability. With knowledge and experience gained from this project the author is able to engage in an even bigger and more complex project in the future. More so, given that the idea of this project is unique, it goes to show the innovative ability of the author in his field of studies.

Results

In the current version of the project, a user’s item that has been found and entered into the database, with the appropriate description is sure to be found and the user is sure to be notified.

Also, the admins are able to perform the necessary tasks on the application. Also, without being notified by the application automatically, a client can just visit the office and the admins are able to determine for sure if their items are found. The search functionality in the admin pages work excellently, enabling the admins to search for specific items or group of items easily.

Users are able to report an item as missing without any glitch. Emails are sent both to confirm a report and to notify users. Every part of the system is working as required.

Although it is just a prototype, this project can serve as a proof of concept for example. Also, with little modification, it can be deployed to a production environment.

POSSIBLE IMPROVEMENTS

Although the web application meets its intended expectation, it could benefit from a few improvements prior to production deployment.

- Spelling errors

As was pointed out earlier, the match engine did not consider the inevitable occurrence of spelling mistakes. Spelling mistakes could greatly alter the search outcome of the engine, leading to inaccurate decisions. A spell checker could be added to the match engine in order to make up for spelling errors both from the users and the admins.

- Support for SMS

Another important improvement is the inclusion of a text messaging system. Although as part of reporting a missing item, users supply their phone number, this information is not used by the application to notify users, should their items be found. The main reason is due to lack of resources. Including a service like that would require certain hardware and software resources, and possibly a contract from a telephone company, all of which were not within reach while the application was under development. The use of text messages to notify users would be a great improvement as people generally carry their phones with them wherever they go, as against checking their emails. However, the users' phone numbers are available to the admins in other that they may be able to contact users, should the need arise.

- Language Limitation

The current version of the application only supports English language. If deployed, most users would prefer to use the application in Finnish language. To incorporate another language would require a reimplementation of the match engine. If another language is to be supported as well, the same routine has to be carried out. Hence, it is obvious that this is not scalable. However, this can be taken into consideration in the next version during design.

- Security

One other area that needs serious improvement before deployment to production is in security. Although basic security measures are in place, the system is still very vulnerable. One such area is the admin login. In the current version of the project, the admin login page can be reached right from the user pages. The main reason for this is that during development, the admin and user pages were developed side by and tested alongside each other. However, this could lead to a security issue as a malicious user knows just where to go in order to attempt an attack on the system. In an idea situation, the admin login url should be different from those accessed by the users.

Also, the system is susceptible to brute force attack on the admin login. This is because there is currently no limit on the number of login attempts from the admin login page. A good solution here would be to limit the number of failed login attempts from the login page. However, although this feature sounds simple, it is not straight forward to implement in Vaadin. The main reason is the Vaadin framework refreshes a browser's session each time the page is refreshed, and returns the browser to the initial page of the application by creating a new instance of the web application on the server. This would therefore mean that when a user is locked out after some failed login attempts, all the user has to do is refresh the page, thereby creating a new instance to make further attempts.

Another way to achieve this would be to lock out the user at the database. However, this could lead to situation where an admin is locked out due a malicious attackers attempt on the system. Nevertheless, this feature could be implemented in a future release of the application.

REFERENCES

[1] Vaadin (2010). Book of Vaadin. Available at: https://vaadin.com/book/vaadin6/-/page/intro.html#intro.overview (Accessed 22 May 2013)

[2]Oracle Corporation (2004). About MySQL. Available at: http://www.mysql.com/about/ (Accessed 22 May 2013)

[3]Oracle Corporation (2013). Download Connector/J. Available at: http://dev.mysql.com/downloads/connector/j/ (Accessed 22 May 2013)

[4]Planet Calc (2010). Online Calculator for Measuring Levenshtein distance between two words. Available at: http://planetcalc.com/1721/ (Accessed 22 May 2013)

[5] Forester (2008). Market Update: Open Source Databases. Available at: http://www.forrester.com/home#/Market+Update+Open+Source+Databases/quickscan/-/E-RES46061 (Accessed 22 May 2013)

[6] MSDN (2013). Designing Rich Internet Applications at: http://msdn.microsoft.com/en-us/library/ee658083.aspx (Accesses 22 May 2013)

[7] Keynote (2013). Rich Internet Applications: Design, Measurement, and Management Challenges at: http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf (Accessed 22 May 2013)

[8] StudyMode (2013). Web Security Issues at http://www.studymode.com/essays/Web-Security-Issues-672046.html (Accessed 22 May 2013).

[9] Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato (2002-2013) Version Control with Subversion. O'Reilly Media, Inc., CA: O'Reilly Media

[10] WordNet (2013). What is WordNet? at http://wordnet.princeton.edu/ (Accessed 22 May 2013)

[11] Simple Java Mail (2013) Simple Java Mail: A very simple Java mailing API for sending simple to complex emails (JavaMail smtp wrapper) at http://code.google.com/p/simple-java-mail/ (Accessed 22 May 2013)

[12] Brett Spell (2009). Java API for WordNet Searching(JAWS). Available at: http://lyle.smu.edu/~tspell/jaws/ (Accessed 22 May 2013)

[13] The jQuery Foundation (2013). jQuery: write less, do more. Available at: http://jquery.com/ (Accessed 22 May 2013)

[14] 1freehosting (2013) Free hosting with php, mysql, cpanel and no ads. Available at: http://www.1freehosting.com/ (Accessed 08 June, 2013)

[15] Apache friends (2013). Available at: http://www.apachefriends.org/en/xampp.html (Accessed 8 June 2013)

[16] Black, Paul E., ed. (14 August 2008). "Levenshtein distance", Dictionary of Algorithms and Data Structures [online], U.S. National Institute of Standards and Technology, retrieved 3 April 2013

[17] Mihalcea, R., Corley, C., and Strapparava, C. (2006) Corpus-based and Knowledge-based Measures of Text Semantic Similarity, in Proceedings of AAAI 2006, Boston, July.

[18] Murdock, V. (2006) Aspects of sentence retrieval. Ph.D. Thesis, University of Massachusetts

[19] Metzler, D., Dumais, S. T., and Meek, C. (2007) Similarity Measures for Short Segments of Text. In Proceedings of ECIR 2007, 16-27.

[20] Software Engineering: A Practitioner's Approach. Boston: McGraw Hill. pp. 41–42. ISBN 9780073375977. Published January 20, 2009.

[21] Softdevteam (2006 – 2010) Incremental lifecycle model. Available at: http://www.softdevteam.com/Incremental-lifecycle.asp (Accessed 23 May 2013)

[22] Symantec Internet Security Threat Report: Trends for July–December 2007 (Executive Summary) XIII. Symantec Corp. April 2008. pp. 1–3. Retrieved May 23, 2013.

[23] Microsoft MSDN (2013) SQL Injection. Available at: http://technet.microsoft.com/en-us/library/ms161953%28v=SQL.105%29.aspx (Accessed 23 May 2013)

[24] "WHID 2009-1: Gaza conflict cyber war". Xiom. Retrieved 2011-06-03.

[25] Arnold IT (2009) Google Nails Patent for Query Synonyms in Query Context, 24 December 2009. Available at: http://arnoldit.com/wordpress/2009/12/24/google-nails-patent-for-query-synonyms-in-query-context/ (Accessed 30 May 2013)

[26] Google Official Blog (2010) Helping Computers Understand Language, 19 January, 2010. Available at: http://googleblog.blogspot.fi/2010/01/helping-computers-understand-language.html (Accessed 30 May 2013)

[27] Google Public Policy Blog (2008) Making Search Better in Catalonia, Estonia, and Everywhere Else, 25 March, 2008. Available at: http://googlepublicpolicy.blogspot.fi/2008/03/making-search-better-in-catalonia.html (Accessed 30 May 2013)

[28] Search Engine Land (2012) Is Google's Synonyms Matching Increasing? How Searchers and Brands Can Be Both Helped & Hurt By Evolving Understanding, 27 August, 2007. Available at: http://searchengineland.com/is-googles-synonym-matching-increasing-how-searchers-and-brands-can-be-both-helped-and-hurt-131504 (Accessed 31 May 2013)

[29] PCWorld (2009) DDoS Attackers Continue Hitting Twitter, Faceboo, Google, 8 August, 2009. Available at: http://www.pcworld.com/article/169893/ddos_attackers_continue_hitting_twitter_facebook_google.html (Accessed 2 June 2013)

[30] Microsoft Support (2013) Description of the database normalization basics , 2013. Available at: http://support.microsoft.com/kb/283878 (Accessed 06 June, 2013)

[31] Ramiz M. Aliguliyev, A new sentence similarity measure and sentence based extractive technique for automatic text summarization, Expert Systems with Applications, Volume 36, Issue 4, May 2009, Pages 7764-7772, ISSN 0957-4174, 10.1016/j.eswa.2008.11.022.

[32] Keng, Leng Hui, "Approximate String Matching with Dynamic Programming and Suffix Trees", 2006. Available at: http://digitalcommons.unf.edu/etd/196 (Accessed 06 June, 2013)

[33] Yiming Yang and Thorsten Joachims (2008), Scholarpedia, 3(5):4242. Available at: http://www.scholarpedia.org/article/Text_categorization#Types_of_classifiers (Accessed 8 June 2013)

[34] Dimitris Papamichail and Georgios Papamichail (2009). Improved algorithms for approximate string matching (extended abstract), 2009. Available at: http://www.biomedcentral.com/1471-2105/10/S1/S10/?fmt_math=no (Accessed 8 June 2013)

[35] Stephen Robertson, (2004) "Understanding inverse document frequency: on theoretical arguments for IDF", Journal of Documentation, Vol. 60 Iss: 5, pp.503 – 520

[36] Metzler, D., Bernstein, Y., Croft, W., Moffat, A., and Zobel, J. (2005) Similarity measures for tracking information flow. Proceedings of CIKM, 517–524.

[37] PennState (2008). Database Fundamentals, 2008. Available at: http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_01.html (Accessed 08 June, 2013)