

Syrian Private University  
Faculty of Informatics & Computer Engineering



# **EXTRACTING INFORMATION FROM UNSTRUCTURED DATA USED IN ONLINE NETWORKS**

A Senior Project Submitted to the faculty of Informatics and Computer Engineering at Syrian Private University in Partial Fulfillment of the Requirements for the Degree of Bachelor in Informatics and Computer Engineering

## **Prepared By**

Noor Mohammad Hasan Orfahly  
Shadia Abd Alkareem Hammoud

## **Under The Supervision Of**

Dr. Mohammad Zouhier Sandouk  
Eng. Mohammad Mousa Hamad

All copyrights reserved for SPU University  
2013

# TABLE OF CONTENTS

	PAGE
<b>List of Tables</b> .....	3
<b>List of Figures</b> .....	4
<b>Abstract</b> .....	6
<b>Introduction</b> .....	7
<b>Reference Studies</b> .....	8
<b>Chapter One: Obtaining Data</b>	
1.1: Overview .....	10
1.2: Methodologies, Problems and Solutions .....	10
1.3: System Analysis .....	11
1.4: Technical Details .....	11
1.5: System Design.....	12
1.6: System Implementation .....	12
<b>Chapter Two: Extracting Information</b>	
2.1: Overview .....	14
2.2: Theoretical Introduction .....	14
2.3: System Analysis .....	14
2.4: System Design.....	14
2.5: System Implementation .....	19
<b>Chapter Three: Optimizing Data</b>	
3.1: Overview .....	21
3.2: Theoretical Introduction .....	21
3.3: Semantic Similarity	
3.3.1: Methodologies, Techniques and Procedures .....	21
3.3.2: System Analysis .....	23
3.3.3: System Design.....	23
3.3.4: System Implementation .....	26
3.4: Spell Checking	
3.4.1: Hunspell Checker .....	28
3.4.2: System Analysis .....	29
3.4.3: System Design.....	29

3.4.4: System Implementation .....	32
------------------------------------	----

## **Chapter Four: Search System**

4.1: Overview .....	35
---------------------	----

4.2: Theoretical Introduction .....	35
-------------------------------------	----

4.3: System Analysis .....	35
----------------------------	----

4.4: System Design.....	53
-------------------------	----

4.5: System Implementation .....	54
----------------------------------	----

<b>Conclusions and Future Work .....</b>	<b>58</b>
--	-----------

<b>References.....</b>	<b>59</b>
------------------------	-----------

## LIST OF TABLES

TABLE	PAGE
Table1 Sign Up Use Case.....	35
Table2 Sign In Use Case .....	36
Table3 Sign Out Use Case.....	37
Table4 Edit Profile Use Case .....	37
Table5 Search Use Case .....	38
Table6 Save Search Results Use Case .....	38
Table7 View History Use Case.....	39
Table8 View Saved Search Operation Use Case.....	40
Table9 Clear History Use Case.....	40
Table10 Delete Saved Search Operation Use Case .....	41

## LIST OF FIGURES

FIGURE	PAGE
Figure 1.1 Content Table.....	12
Figure 2.1 User Table.....	14
Figure 2.2 Country Table.....	14
Figure 2.3 Language Table .....	15
Figure 2.4 Skill Table.....	15
Figure 2.5 Organization Table .....	15
Figure 2.6 Degree Table .....	16
Figure 2.7 Major Table .....	16
Figure 2.8 Education Table .....	16
Figure 2.9 Job Table .....	17
Figure 2.10 Experience Table .....	17
Figure 2.11 Project Table .....	17
Figure 2.12 Course Table.....	18
Figure 2.13 Certificate Table .....	18
Figure 2.14 User Language Table.....	18
Figure 2.15 User Skill Table .....	19
Figure 2.16 User Course Table .....	19
Figure 2.17 User Certificate Table .....	19
Figure 3.1 Admin Table .....	23
Figure 3.2 Country Optimization Table.....	24
Figure 3.3 Industry Optimization Table.....	24
Figure 3.4 Job Optimization Table .....	25
Figure 3.5 Language Optimization Table .....	25
Figure 3.6 Major Optimization Table .....	26
Figure 3.7 Skill Optimization.....	26
Figure 3.8 Log In Form.....	27
Figure 3.9 Get Similarities Form.....	27
Figure 3.10 View Similarities Form.....	28
Figure 3.11 Country Errors Table.....	29
Figure 3.12 Industry Errors Table.....	30
Figure 3.13 Job Errors Table.....	30

Figure 3.14 Language Errors Table.....	31
Figure 3.15 Major Errors Table .....	31
Figure 3.16 Skill Errors Table.....	32
Figure 3.17 Log In Form.....	33
Figure 3.18 Get Errors Form .....	33
Figure 3.19 View Errors Form .....	34
Figure 4.1 Use Case Diagram .....	42
Figure 4.2 Sign Up Sequence Diagram.....	43
Figure 4.3 Sign In Sequence Diagram .....	44
Figure 4.4 Sign Out Sequence Diagram.....	45
Figure 4.5 Edit Profile Sequence Diagram .....	46
Figure 4.6 Search Sequence Diagram .....	47
Figure 4.7 Save Search Results Sequence Diagram .....	48
Figure 4.8 View History Sequence Diagram.....	49
Figure 4.9 View Saved Search Operation Sequence Diagram.....	50
Figure 4.10 Clear History Sequence Diagram.....	51
Figure 4.11 Delete Saved Search Operation Sequence Diagram .....	52
Figure 4.12 Member Table .....	53
Figure 4.13 Role Table.....	53
Figure 4.14 Search Operation Table .....	54
Figure 4.15 Search History Table.....	54
Figure 4.16 Main Form.....	55
Figure 4.17 Search Form .....	55
Figure 4.18 Search Result Form .....	56
Figure 4.19 History Form.....	57

## **ABSTRACT**

The invention of the World Wide Web made it possible to connect people all over the world. A new style of employing web applications in communicating people is the online networks. The main concept in these networks is the “Sociality” i.e. linking people with their community to exchange information, news, personal stuff and whatever. We can't ignore that this aspect of online network is urging people to have a good digital figure and to generously share info, ideas and opinions. Facebook, Twitter, Google+, YouTube and LinkedIn are the most popular examples of online networks. These websites are becoming containers for huge piles of visual information without having an ability to get benefit of them. The issue of our project is creating that tool which can extract unstructured information used in online networks and organize them in an appropriate structure to enable applying them in further applications. To make it a specialized project we choose our case study to be LinkedIn.com.

## INTRODUCTION

LinkedIn is the world's largest professional network with 225 million members in over 200 countries and territories around the globe. <sup>(1)</sup> The person who registers to this website can publish his CV online to be viewed by anyone else. Not only people, also companies can have profiles on LinkedIn. There are two types of profiles: Basic profile which is free and Premium profile which is paid. Information can include personal details, Education, Experiences, Skills, Courses, Interests and Certificates. There's also the ability to add connections (friends) and join groups.

Having this data in a well-structured format will enable to build useful applications. Consider a situation in which a company manager wants to employ a new person with specific characteristics, of course he can surf the network to find the efficient individual but it will be easier if there's a tool to search through CVs according to a specified criteria. Another possible case is when an academy wants to advertise a course, it can make a public advertisement but it will be more convenient to forward the ad to those interested in the course's study field. Data mining, One-to-One marketing, Statistical Researches are also possible examples.

In order to implement that, we need to move through 3 steps: First, obtain the data. Second, extract information from that data. Third, manipulate information. This documentation will explain in details these three phases of the system.



## REFERENCE STUDIES

Extracting information from networks was and still an important topic for researchers and software developers. In 2004, University of Massachusetts – Amherst published a paper under the title “Extracting social networks and contact information from email and the Web” by Aron Culotta, Ron Bekkerman and Andrew McCallum. This sheet tells that they present an end-to-end system that extracts a user’s social network and its members’ contact information given the user’s email inbox. The system identifies unique people in email, finds their Web presence, and automatically fills the fields of a contact address book using conditional random fields—a type of probabilistic model well-suited for such information extraction tasks. By recursively calling itself on new people discovered on the Web, the system builds a social network with multiple degrees of separation from the user. Additionally, a set of expertise-describing keywords are extracted and associated with each person.<sup>(2)</sup> Another study was published in 2012 under the title “Extracting Information Networks from the Blogosphere” by YUVAL MERHAV from Illinois Institute of Technology, FILIPE MESQUITA and DENILSON BARBOSA from University of Alberta, WAI GEN YEE from Orbitz Worldwide and OPHIR FRIEDER from Georgetown University. They say that they study the problem of automatically extracting information networks formed by recognizable entities as well as relations among them from social media sites. Their approach consists of using state-of-the-art natural language processing tools to identify entities and extract sentences that relate such entities, followed by using text-clustering algorithms to identify the relations within the information network. They describe an effective method for identifying benchmarks for open information extraction that relies on a crated online database that is comparable to the hand-crafted evaluation datasets in the literature. From this benchmark, they derive a much larger dataset which mimics realistic conditions for the task of open information extraction. They report on extensive experiments on both datasets, which not only shed light on the accuracy levels achieved by state-of-the-art open information extraction tools, but also on how to tune such tools for better results.<sup>(3)</sup>

Now, to speak about our project, the main idea was to create that tool which extracts useful information from social online networks and specifically LinkedIn.com.

Recently, the use of the internet started to wide spread here, in Syria. And organizations began to get benefit from available information on the World Wide

Web (WWW). Some companies, for example, headed for employing people depending on their online data containing facts, details and whatever. This data can be found in some popular websites such as LinkedIn.com where a person adds his CV online listing his degrees, certificates, skills, and experiences. Starting from this point, we found it important to take advantage of this published information by making it well organized and searchable, not only that, but also in a form that enables to build any useful application.

## Chapter One

# Obtaining Data

### 1.1 Overview:

In this chapter we will cover how to obtain data from Linked In and save them temporarily in a data base table.

### 1.2 Methodologies, Problems and solutions:

In order to pull data from the Linked In website we can use the Profile API provided by the website itself. So, what is API? What is the Profile API?

API (or Application Program Interface) is an interface that a specific website offers for developers to write their own applications on the website data without having to understand the inner working of the website or directly communicate with the website database. It guarantees simplicity for the developer and security for the website owner.

The *Profile API* is provided by LinkedIn.com and returns a member's LinkedIn profile. We can use this call to return one of two versions of profile: Standard or Public. Examples of possible returned fields are: Name, headline, industry and so on. Retrieved data can be in one of two formats either XML or JSON. <sup>(4)</sup>

The first step to start developing is registering the application; after that Linked In uses the OAuth 1.0a protocol to give the application authorized access to the APIs. <sup>(5)</sup> And for this reason we need to use a library that allows using this protocol in a certain programming language (C# in our case). There are many libraries for this purpose but all of them are a “geek work” which means they are not supported by a known organization or company.

Some instances are DotNetOpenAuth, OAuth library for .NET and DevDefined OAuth. Unfortunately, we have tried each one of them and more but no one has worked out! The problem was that they have neither simple clear documentation nor stable releases. And so we were in a big trouble! To overcome this problem we decided to switch to another retrieving method which depends on HTML Parsing.

*HTML Parsing* is a technique with which we can move around a HTML file and extract text from it. First we will pull profiles HTML code from the Linked In website and

store it in a database table to avoid snags like internet connection abruption and loss of data. Explanation of procedure and used techniques is coming.

### 1.3 System Analysis:

To continuously collect as many profiles data as possible we have followed this procedure.

#### Procedure:

- First step:
  - Download the source code of a specific profile and store it in the database.
  - For that profile, download other suggested profiles and store them in the database.
- Second step:

Randomly select a profile saved in the database and for that profile, download other suggested profiles and store them in the database.

### 1.4 Technical Details:

We have used the HTML Agility Pack library to navigate through each profile and extract other suggested profiles links. The language used here was XPATH.

*HTML Agility Pack* is an agile HTML parser that builds a read/write DOM and supports plain XPATH or XSLT. It is a .NET code library that allows you to parse "out of the web" HTML files. The parser is very tolerant with "real world" malformed HTML. The object model is very similar to what proposes System.Xml, but for HTML documents (or streams). <sup>(6)</sup>

*XPATH* is used to iterate and access any node within a XML document. Different functions and expressions are available within XPath specifications to help access different kind of XML nodes. HTML Agility pack uses XPath to access any of node within a HTML document. <sup>(7)</sup>

## 1.5 System Design:

We have used three-layer architecture, one layer for the database tables, another one for the process and third one for the interface.

Applying this architecture, offers these benefits:

- Ensuring the security of the database by preventing access to the tables from the user interfaces.
- Specialization of layers' work so the tasks are easily broken-down.
- Ease of modification. Changes in one layer should not have an impact onto other layers.

Figure 1.1 shows the database design.

The Content Table stores pulled HTML documents; it contains id field as a primary key and URL and page content fields which represent the address and the code for each saved profile.

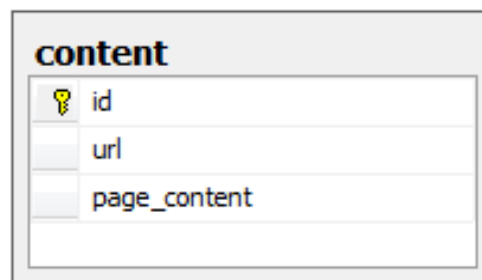


Figure 1.1 Content Table

## 1.6 System Implementation:

### Used Techniques:

- Microsoft SQL Server 2005
- Visual C# with Microsoft Visual Studio 2010
- LINQ to SQL
- HTML Agility Pack
- XPATH

We have used the techniques listed above and built a system which actually has gained more than 10,000 Profile!!!

## Chapter Two

# Extracting Information

### 2.1 Overview:

In this chapter we will cover how to extract data from the saved profiles and store them in a new database.

### 2.2 Theoretical Introduction:

In this phase of our project we need to extract useful information from the stored profiles, why? Simply because the received html code is not in a well structured format that allows applying manipulation techniques to it! So, again we have to use HTML Agility Pack Library and XPATH (both explained in the previous chapter) to navigate through the profiles' HTML code and mine meaningful text from within large number of meaningless html nodes.

### 2.3 System Analysis:

To achieve our task of surfing all profiles' code we need to pull that code from the database and process it. We used I/O File storage method to keep the last reached profile id so we can pause the system and resume it anytime, anywhere.

#### Procedure:

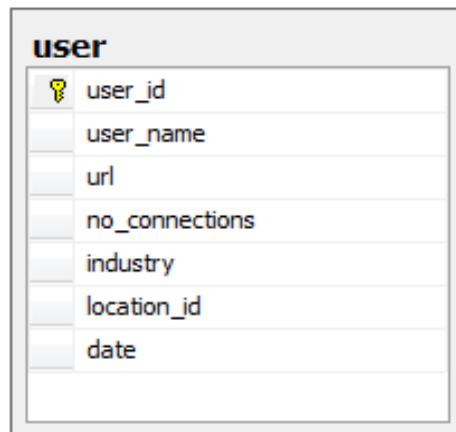
- For the reached user id, load the source code of the corresponding profile.
- Extract the information and store them in the database.
- Increment reached user id and store it in the file.
- Read reached user id from the file, if it's smaller than total number of stored user profiles then repeat procedure, else terminate.

### 2.4 System Design:

We have used three-layer architecture, one layer for the database tables, another one for the process and third one for the interface.

### Database Tables:

- User Table: this table stores the basic information found in a person's profile, it contains user\_id field as a primary key, location\_id field as a foreign key to the Country Table and other fields for some information about the user. (as shown in figure 2.1)



The diagram shows a table named 'user' with the following fields: user\_id (primary key, indicated by a yellow key icon), user\_name, url, no\_connections, industry, location\_id, and date. The fields are listed in a vertical column within a rectangular box.


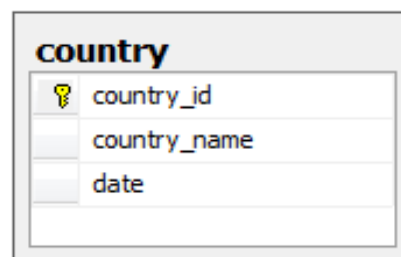
user	
	user_id
	user_name
	url
	no_connections
	industry
	location_id
	date

Figure 2.1 User Table

- Country Table: this table stores names of different locations in the world that were found while visiting users' profiles, it contains country\_id field as a primary key, country\_name field and date field to save the time of insertion. (as shown in figure 2.2)



The diagram shows a table named 'country' with the following fields: country\_id (primary key, indicated by a yellow key icon), country\_name, and date. The fields are listed in a vertical column within a rectangular box.


country	
	country_id
	country_name
	date

Figure 2.2 Country Table

- Language Table: this table stores different languages found while visiting users' profiles, it contains lang\_id field as a primary key, lang\_name field and date field to save the time of insertion. (as shown in figure 2.3)


language	
	lang_id
	lang_name
	date

Figure 2.3 Language Table

- Skill Table: this table stores different skills found while visiting users' profiles, it contains skill\_id field as a primary key, skill\_name field and date field to save the time of insertion. (as shown in figure 2.4)


skill	
	skill_id
	skill_name
	date

Figure 2.4 Skill Table

- Organization Table: this table stores names of different organizations in the world that were found while visiting users' profiles, it contains org\_id field as a primary key, location\_id field as a foreign key to the Country Table and org\_name field. (as shown in figure 2.5)


organization	
	org_id
	org_name
	location_id

Figure 2.5 Organization Table

- Degree Table: this table stores extracted degrees which a person can have, it contains degree\_id field as a primary key and degree\_name field. (as shown in figure 2.6)




degree	
	degree_id
	degree_name

Figure 2.6 Degree Table

- Major Table: this table stores extracted majors which a person can have, it contains major\_id field as a primary key, major\_name field and date field to save the time of insertion. (as shown in figure 2.7)


major	
	major_id
	major_name
	date

Figure 2.7 Major Table

- Education Table: this table stores information about a user's education, it contains education\_id field as a primary key, user\_id field as a foreign key to the User Table, org\_id field as a foreign key to the Organization Table, degree\_id field as a foreign key to the Degree Table, Major\_id field as a foreign key to the Major Table and start\_date and end\_date fields to represent the duration of that education record. (as shown in figure 2.8)


education	
	education_id
	user_id
	org_id
	degree_id
	major_id
	start_date
	end_date

Figure 2.8 Education Table

- Job Table: this table stores extracted job titles which a person can have, it contains job\_id field as a primary key, job\_name field and date field to save the time of insertion. (as shown in figure 2.9)


job	
	job_id
	job_name
	date

Figure 2.9 Job Table

- Experience Table: this table stores information about a user's experience, it contains experience\_id field as a primary key, user\_id field as a foreign key to the User Table, org\_id field as a foreign key to the Organization Table, job\_id field as a foreign key to the Job Table, description field and start\_date and end\_date fields to represent the duration of that experience record. (as shown in figure 2.10)


experience	
	experience_id
	user_id
	org_id
	job_id
	start_date
	end_date
	description

Figure 2.10 Experience Table

- Project Table: this table stores information about a user's projects, it contains project\_id field as a primary key, user\_id field as a foreign key to the User Table and other fields for some information about the project. (as shown in figure 2.11)


project	
	project_id
	project_title
	description
	duration
	user_id

Figure 2.11 Project Table

- Course Table: this table stores names of possible courses a person can list in his profile, it contains course\_id field as a primary key and course\_name field. (as shown in figure 2.12)

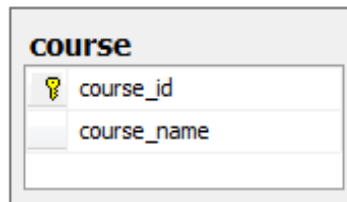


Figure 2.12 Course Table

- Certificate Table: this table stores names of possible certificates a person can list in his profile, it contains cert\_id field as a primary key and cert\_name field. (as shown in figure 2.13)

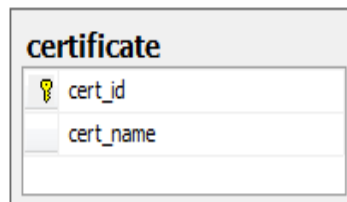


Figure 2.13 Certificate Table

- User\_Language Table: this table stores a certain user's languages; it contains user\_language\_id field as a primary key, user\_id field as a foreign key to the User Table and lang\_id field as a foreign key to the Language Table. (as shown in figure 2.14)

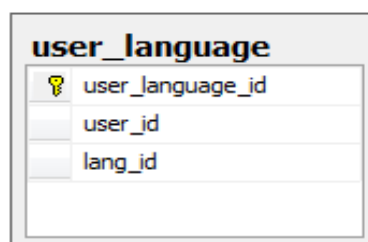


Figure 2.14 User Language Table

- User\_Skill Table: this table stores a certain user's skills; it contains user\_skill\_id field as a primary key, user\_id field as a foreign key to the User Table and skill\_id field as a foreign key to the Skill Table. (as shown in figure 2.15)


user_skill	
	user_skill_id
	user_id
	skill_id

Figure 2.15 User Skill Table

- User\_Course: this table stores a certain user's courses; it contains user\_course\_id field as a primary key, user\_id field as a foreign key to the User Table, course\_id field as a foreign key to the Course Table and org\_id field as a foreign key to the Organization Table. (as shown in figure 2.16)


user_course	
	user_course_id
	user_id
	course_id
	org_id

Figure 2.16 User Course Table

- User\_Certificate: this table stores a certain user's certificates; it contains user\_cert\_id field as a primary key, user\_id field as a foreign key to the User Table, cert\_id field as a foreign key to the Certificate Table and org\_id field as a foreign key to the Organization Table. (as shown in figure 2.17)


user_certificate	
	user_cert_id
	user_id
	cert_id
	org_id

Figure 2.17 User Certificate Table

## 2.5 System Implementation:

### Used Techniques:

- Microsoft SQL Server 2005
- Visual C# with Microsoft Visual Studio 2010

- LINQ to SQL
- HTML Agility Pack
- XPATH

We have used the techniques listed above and built a system which actually has stored the extracted information for all the 10,000 gained Profiles.

# Optimizing Data

### 3.1 Overview:

In this chapter we will cover how to optimize extracted data.

### 3.2 Theoretical Introduction:

Since the data in our project belongs to a huge number of profiles, it's normal to face problems while manipulating it, e.g. dissimilar phrases with the similar meaning like "Computer Software Engineering" and "Software Engineer" (which can be used to describe the person's job) can cause troubles in a search process. Another possible problem is misspelling i.e. having a syntax error in the text like writing "Engilsh" instead of "English" when listing the person's language. In order to solve such problems we had to use some existing techniques and theories that serve our purpose. Details are coming.

### 3.3 Semantic Similarity:

#### 3.3.1 Methodologies, Techniques and Procedures:

Finding a way to determine the degree of semantic similarity between two phrases can help to handle issues concerning different expressions with the same meaning.

There is an open source C# library called "*Open NLP*" provides a procedure that tokenizes an input sentence i.e. converts it to a list of words and then removes stop words from that list (stop words are useless terms like prepositions and determiners) and another procedure to calculate the semantic similarity between two words. So, our task is to build a layer above those available procedures to calculate the semantic similarity between two sentences and this is easily done by tokenizing the two input phrases (using the first procedure) and finding the ratio of similarity between each two words of the tokenized phrases

(using the second procedure). At the end our goal is represented by the mean of all computed values.

But how all of that is achieved?

Open NLP depends on a database known as “WordNet”.

WordNet<sup>®</sup> is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.<sup>(8)</sup>

The relations between different (synsets) vary based on the type of word. For nouns we have:

- hypernyms: Y is a hypernym of X if every X is a (kind of) Y (canine is a hypernym of dog)
- hyponyms: Y is a hyponym of X if every Y is a (kind of) X (dog is a hyponym of canine)
- coordinate terms: Y is a coordinate term of X if X and Y share a hypernym (wolf is a coordinate term of dog, and dog is a coordinate term of wolf)
- holonym: Y is a holonym of X if X is a part of Y (building is a holonym of window)
- meronym: Y is a meronym of X if Y is a part of X (window is a meronym of building)<sup>(9)</sup>

Now, we can use these relations to compute the semantic distance between two words by measuring the distance between the two corresponding nodes in the WordNet Network and the ratio of the similarity can then be calculated by one of these equations:

- $\text{Sim}(s, t) = 1/\text{distance}(s, t)$
- $\text{Sim}(s, t) = \text{SenseWeight}(s) * \text{SenseWeight}(t) / \text{PathLength}$ ; where:

- s and t: denote the source and target words being compared.
- SenseWeight: denotes a weight calculated according to the frequency of use of this sense and the total of frequency of use of all senses.
- PathLength: denotes the length of the connection path from s to t.

In our project we have used the latter one.

### 3.3.2 System Analysis:

We have two concerns, the first one to calculate semantic similarity between each two phrases and if the ratio of similarity is high store the result in a database table for further checking.

The second is to view the saved phrases' similarities and let admins decide to either accept or refuse the result.

The process here doesn't affect the original data; it only improves the data manipulation tasks (in particular, the search operation described in the next chapter).

Security here is achieved by the mandatory logging in step.

### 3.3.3 System Design:

In addition to previous database tables we have added these tables to manage logging in and optimization process.

Database Tables:

- Admin Table: this table stores admins' logging in credentials, it contains admin\_id field as a primary key, admin\_name field and admin\_password field. (as shown in figure 3.1)

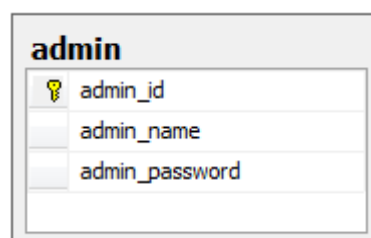
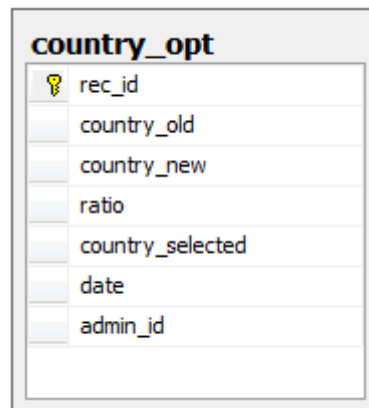


Figure 3.1 Admin Table



- **Country\_Opt Table:** this table stores semantic similarity suggestions found for a certain country name, it contains `rec_id` field as a primary key, `admin_id` field as a foreign key to the Admin Table, `country_old`, `country_new` and `country_selected` fields as foreign keys to the Country Table, `ratio` field and `date` field to save the time of acceptance. (as shown in figure 3.2)




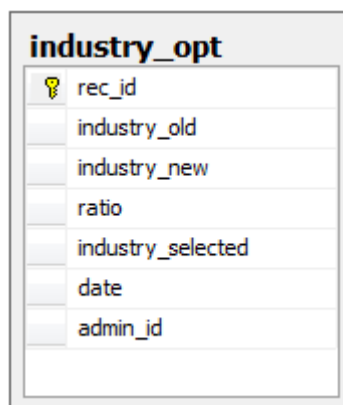
country_opt	
	rec_id
	country_old
	country_new
	ratio
	country_selected
	date
	admin_id

Figure 3.2 Country Optimization Table

- **Industry\_Opt Table:** this table stores semantic similarity suggestions found for a certain user's industry description, it contains `rec_id` field as a primary key, `admin_id` field as a foreign key to the Admin Table, `industry_old`, `industry_new` and `industry_selected` fields as foreign keys to the User Table, `ratio` field and `date` field to save the time of acceptance. (as shown in figure 3.3)




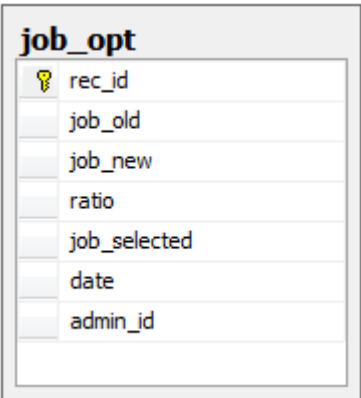
industry_opt	
	rec_id
	industry_old
	industry_new
	ratio
	industry_selected
	date
	admin_id

Figure 3.3 Industry Optimization Table

- **Job\_Opt Table:** this table stores semantic similarity suggestions found for a certain job title, it contains `rec_id` field as a primary key, `admin_id` field as a foreign key to the Admin Table, `job_old`, `job_new` and `job_selected` fields as

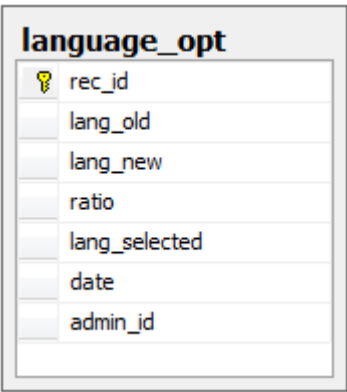
foreign keys to the Job Table, ratio field and date field to save the time of acceptance. (as shown in figure 3.4)



rec_id
job_old
job_new
ratio
job_selected
date
admin_id

Figure 3.4 Job Optimization Table

- Language\_Opt Table: this table stores semantic similarity suggestions found for a certain language name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, lang\_old, lang\_new and lang\_selected fields as foreign keys to the Language Table, ratio field and date field to save the time of acceptance. (as shown in figure 3.5)



rec_id
lang_old
lang_new
ratio
lang_selected
date
admin_id

Figure 3.5 Language Optimization Table

- Major\_Opt Table: this table stores semantic similarity suggestions found for a certain major name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, major\_old, major\_new and major\_selected fields as foreign keys to the Major Table, ratio field and date field to save the time of acceptance. (as shown in figure 3.6)


major_opt	
	rec_id
	major_old
	major_new
	ratio
	major_selected
	date
	admin_id

Figure 3.6 Major Optimization Table

- Skill\_Opt Table: this table stores semantic similarity suggestions found for a certain skill name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, skill\_old, skill\_new and skill\_selected fields as foreign keys to the Skill Table, ratio field and date field to save the time of acceptance. (as shown in figure 3.7)


skill_opt	
	rec_id
	skill_old
	skill_new
	ratio
	skill_selected
	date
	admin_id

Figure 3.7 Skill Optimization

### 3.3.4 System Implementation:

#### Used Techniques:

- Microsoft SQL Server 2005
- Visual C# with Microsoft Visual Studio 2010
- LINQ to SQL
- WordNet
- Open NLP

Now, we have a system with which an admin can log in, start getting similarities and then view these values to make a decision about them.

As we tried the comparing process on some testing examples we found out that the ratio 90% is suitable to our project because it's enough to demonstrate a semantic relation between two phrases. Consequently, the system should retain the result having a ratio of 90% or more to take them in consideration and regard the rest. And this is what we have actually done.

#### Interfaces:

- Log IN Form: The form used by the admin to log in. (shown in figure 3.8)

A screenshot of a web application window titled "Log In". The window has a light blue header bar with standard window controls (minimize, maximize, close). The main content area has a colorful, abstract background. It contains two input fields: "User Name" with the text "admin" and "Password" with three asterisks "\*\*\*". Below these fields is a "Log In" button.

Figure 3.8 Log In Form

- Get Similarities Form: The form used by the admin to get similarities. (shown in figure 3.9)

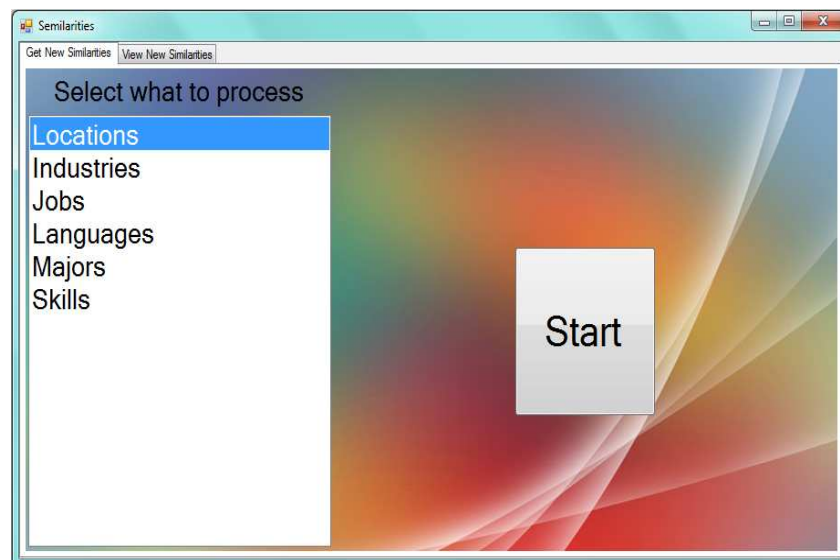
A screenshot of a web application window titled "Similarities". The window has a light blue header bar with standard window controls. Below the header, there are two tabs: "Get New Similarities" (active) and "View New Similarities". The main content area has a colorful, abstract background. It features a section titled "Select what to process" with a list of options: "Locations", "Industries", "Jobs", "Languages", "Majors", and "Skills". The "Locations" option is highlighted with a blue background. To the right of this list is a large "Start" button.

Figure 3.9 Get Similarities Form

- View Similarities Form: The form used by the admin to view similarities and make a decision about them. (shown in figure 3.10)

Old	New	Ratio	Replace ?
Kuwait ...	Miami R...	0.905	<input type="checkbox"/>
MIAMI	Miami R...	0.92	<input type="checkbox"/>
Denmark	Sverige	0.91	<input type="checkbox"/>
Danmark	Sverige	0.91	<input type="checkbox"/>
Sweden	Sverige	1	<input type="checkbox"/>
Norway	Sverige	0.91	<input type="checkbox"/>
Kuwait ...	Seattle	0.905	<input type="checkbox"/>
Seattle, ...	Seattle	0.93	<input type="checkbox"/>

Figure 3.10 View Similarities Form

### 3.4 Spell Checking:

#### 3.4.1 Hunspell Checker:

Hunspell is a spell checker and morphological analyzer designed for languages with rich morphology and complex word compounding and character encoding. It's an open software and can use Unicode UTF-8-encoded dictionaries.<sup>(10)</sup>

Hunspell is the spell checker of LibreOffice, OpenOffice.org, Mozilla Firefox 3 & Thunderbird, Google Chrome, and it is also used by proprietary software packages, like Mac OS X, InDesign, memoQ, Opera and SDL Trados.

##### Main features:

- Extended support for language peculiarities; Unicode character encoding, compounding and complex morphology.
- Improved suggestion using n-gram similarity, rule and dictionary based pronunciation data.
- Morphological analysis, stemming and generation.

And more!<sup>(11)</sup>

### 3.4.2 System Analysis:

We have two concerns; the first one is using the Hunspell Checker to find suggestions to wrong values.

The second is to view the found suggestions and let admins decide to either accept or refuse the result.

The process here affects the original data;

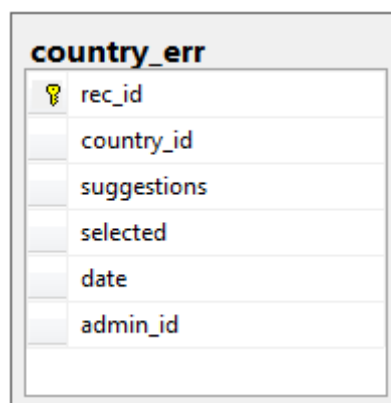
Security here is achieved by the mandatory logging in step.

### 3.4.3 System Design:

In addition to previous database tables we have added these tables to spell-checking process.

#### Database Tables:

- **Country\_Err Table:** this table stores syntax errors' possible corrections found for a certain country name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, country\_id field as a foreign key to the Country Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.11)



rec_id
country_id
suggestions
selected
date
admin_id

Figure 3.11 Country Errors Table

- **Industry\_Err Table:** this table stores syntax errors' possible corrections found for a certain user's industry description, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, industry\_id field as a foreign key to the User Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.12)


industry_err	
	rec_id
	industry_id
	suggestions
	selected
	date
	admin_id

Figure 3.12 Industry Errors Table

- Job\_Err Table: this table stores syntax errors' possible corrections found for a certain job title, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, job\_id field as a foreign key to the Job Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.13)


job_err	
	rec_id
	job_id
	suggestions
	selected
	date
	admin_id

Figure 3.13 Job Errors Table

- Language\_Err Table: this table stores syntax errors' possible corrections found for a certain language name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, language\_id field as a foreign key to the Language Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.14)


language_err	
	rec_id
	language_id
	suggestions
	selected
	date
	admin_id

Figure 3.14 Language Errors Table

- Major\_Err Table: this table stores syntax errors' possible corrections found for a certain major name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, major\_id field as a foreign key to the Major Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.15)


major_err	
	rec_id
	major_id
	suggestions
	selected
	date
	admin_id

Figure 3.15 Major Errors Table

- Skill\_Err Table: this table stores syntax errors' possible corrections found for a certain skill name, it contains rec\_id field as a primary key, admin\_id field as a foreign key to the Admin Table, skill\_id field as a foreign key to the Skill Table, suggestions field, selected field and date field to save the time of acceptance. (as shown in figure 3.16)




skill_err	
	rec_id
	skill_id
	suggestions
	selected
	date
	admin_id

Figure 3.16 Skill Errors Table

### 3.4.4 System Implementation:

#### Used Techniques:

- Microsoft SQL Server 2005
- Visual C# with Microsoft Visual Studio 2010
- LINQ to SQL
- Hunspell Checker

Now, we have a system with which an admin can log in, start getting syntax errors and then view these values to make a decision about them.

Here, replacing values is permanent and so admins should be sure before they decide to change values.

#### Interfaces:

- Log IN Form: the form used by the admin to log in. (shown in figure 3.17)



Figure 3.17 Log In Form

- Get Errors Form: The form used by the admin to get syntax errors. (shown in figure 3.18)

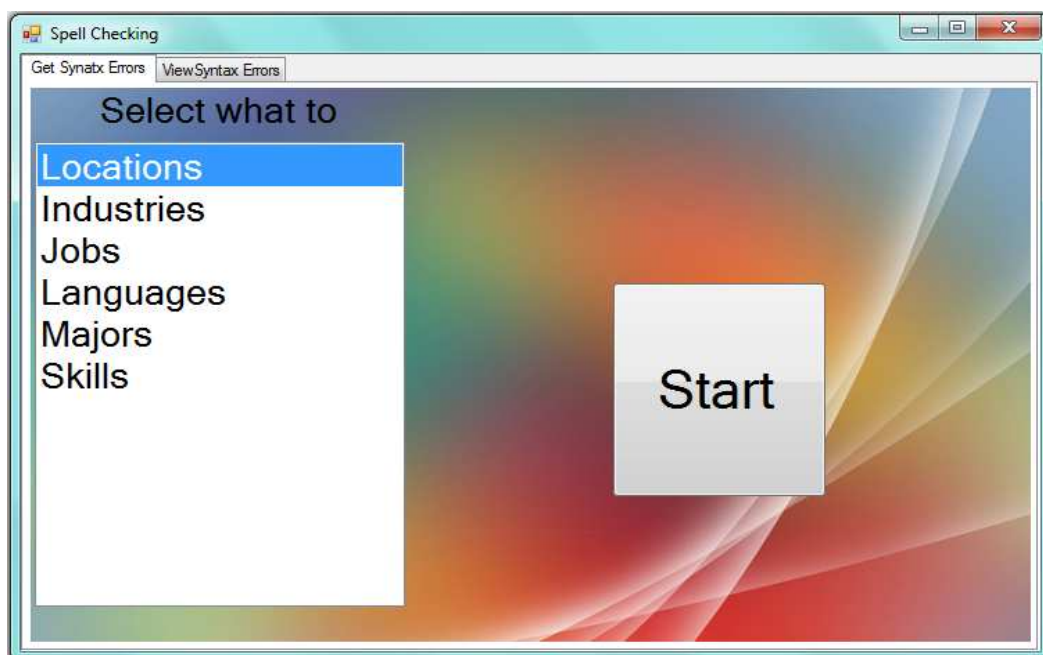


Figure 3.18 Get Errors Form

- View Errors Form: The form used by the admin to view syntax errors and make a decision about them. (shown in figure 3.19)

Spell Checking

Get Syntax Errors View Syntax Errors

Select what to view

Locations  
Industries  
Jobs  
Languages  
Majors  
Skills

View

Original	Suggestion	Replace ?
Dar Es Sal...	Gar Es Sal...	<input type="checkbox"/>
Dar Es Sal...	Mar Es Sal...	<input type="checkbox"/>
Dar Es Sal...	Par Es Sala...	<input type="checkbox"/>
Dar Es Sal...	Bar Es Sala...	<input type="checkbox"/>
Dar Es Sal...	Day Es Sal...	<input type="checkbox"/>
Dar Es Sal...	Far Es Sala...	<input type="checkbox"/>
Dar Es Sal...		<input type="checkbox"/>

Notice that replacement is permanent. (last one is selected)

Replace

Figure 3.19 View Errors Form

## Chapter Four

# Search System

### 4.1 Overview:

In this chapter we will cover how to make processed information available for search.

### 4.2 Theoretical Introduction:

After we manipulated the data and optimized it as much as we could; now the data is ready to be fed into any meaningful transformation process to make it more useful to us.

One of important applications that we can create is a search system that allows looking for a person (i.e. profile) that satisfies a list of characteristics the searcher specifies.

Once a member registers to the search website we have developed, he has the ability to search through profiles, save the search results and view his search history.

### 4.3 System Analysis:

As a result of The Search System analysis we made, we obtained this list of use cases.

Use Cases:

**Table1 Sign Up Use Case**

Use Case Number	1
Use Case Name	Sign Up.
Actors	Primary: Anonymous member. Secondary: None.
Included Use Case	Sign In.
Description	Register a new member.
Preconditions	None.
Main Flow	1 – The member enters values for the member name and passwords fields. 2 – The system registers a new member

	in the website. 3 – Include Sign In.
Alternative Flow	1.1 – If the member didn't enter each field then the system notifies him. 1.2 – If the two passwords mismatch the system shows an error message. 1.3 – If any value exceeds 50 characters the system shows an error message.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table2 Sign In Use Case**

Use Case Number	2
Use Case Name	Sign In.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Sign in a member to the website.
Preconditions	The member is registered.
Main Flow	1 – The member enters values for the member name and password fields. 2 – The system signs in the member.
Alternative Flow	1.1 – If the member didn't enter each field then the system notifies him. 1.2 – If the values are incorrect the system shows an error message.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table3 Sign Out Use Case**

Use Case Number	3
Use Case Name	Sign Out.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Sign out a member from the website.
Preconditions	The member is signed in.
Main Flow	1 – The member clicks on (sign out) link. 2 – The system signs out the member from the website.
Alternative Flow	None.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table4 Edit Profile Use Case**

Use Case Number	4
Use Case Name	Edit Profile.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Edit a member's profile.
Preconditions	The member is signed in.
Main Flow	1 – The member enters values for the member name and passwords fields. 2 – The system updates the member's profile.
Alternative Flow	1.1 – If the two passwords mismatch the system shows an error message. 1.2 – If any value exceeds 50 characters the system shows an error message.

Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table5 Search Use Case**

Use Case Number	5
Use Case Name	Search.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Search for people who satisfy the member's customized criteria.
Preconditions	The member is signed in.
Main Flow	1 – The member enters values for the fields he wants to set the search criteria. 2 – The system search for people who satisfy the search criteria. 3 – If any search result, the system shows a table containing the name of the person associated with the link to his profile and other information.
Alternative Flow	1.1 – If the member didn't enter any field then the system notifies him. 3.1 – If there's no search result, the system shows a message to explain.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table6 Save Search Results Use Case**

Use Case Number	6
Use Case Name	Save Search Results.

Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Save Search Results for future.
Preconditions	The member is signed in.
Main Flow	1 – The member chooses to save the search results. 2 – The system saves the search results in the member's history.
Alternative Flow	None.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table7 View History Use Case**

Use Case Number	7
Use Case Name	View History.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	View the member's saved history.
Preconditions	The member is signed in.
Main Flow	1 – The system views the saved history with a link to each search operation to view its search results and a choice to delete it.
Alternative Flow	1.1 – If the user didn't save any search results before nothing is viewed.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.



**Table8 View Saved Search Operation Use Case**

Use Case Number	8
Use Case Name	View Saved Search Operation.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	View saved search operation results.
Preconditions	The member is signed in.
Main Flow	1 – The member chooses from the history a search operation to view its search results. 2 – The system shows saved results.
Alternative Flow	None.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table9 Clear History Use Case**

Use Case Number	9
Use Case Name	Clear History.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Clear member's saved history.
Preconditions	The member is signed in.
Main Flow	1 – The member chooses to clear all history. 2 – The system deletes the saved history.
Alternative Flow	None.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

**Table10 Delete Saved Search Operation Use Case**

Use Case Number	10
Use Case Name	Delete Saved Search Operation.
Actors	Primary: Website Admin, Registered member. Secondary: None.
Description	Delete one search operation from a member's saved history.
Preconditions	The member is signed in.
Main Flow	1 – The member chooses the saved search operation to delete. 2 – The system deletes that operation.
Alternative Flow	None.
Post Conditions	None.
Exceptions	Internet connection is lost by either the user or the system.

Use Case Diagram: we have three actors: Anonymous Member, Registered Member and Website Admin. (as shown in figure 4.1)

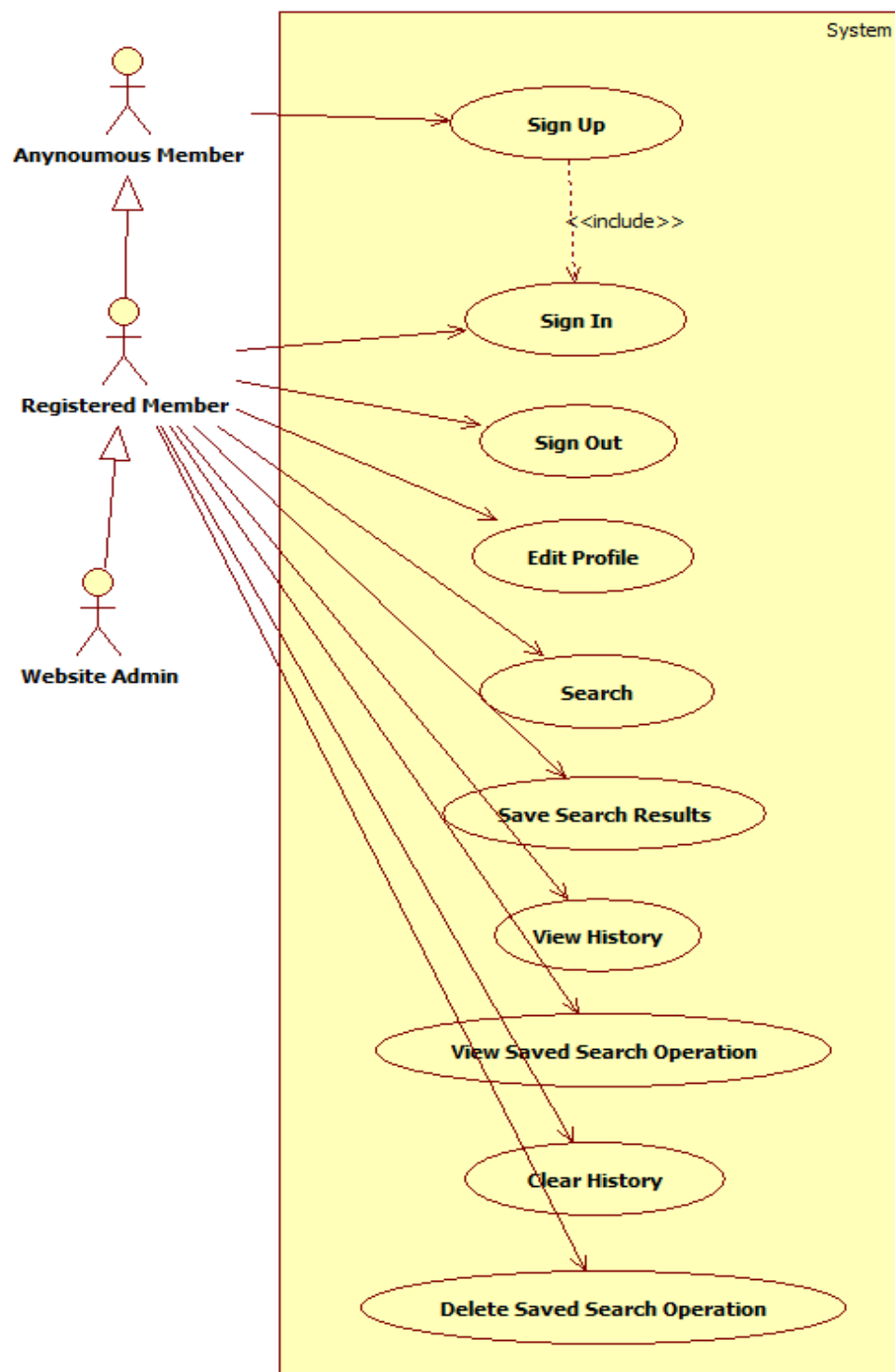


Figure 4.1 Use Case Diagram.

## Sequence Diagrams:

- Figure 4.2 shows the Sign Up Sequence Diagram.

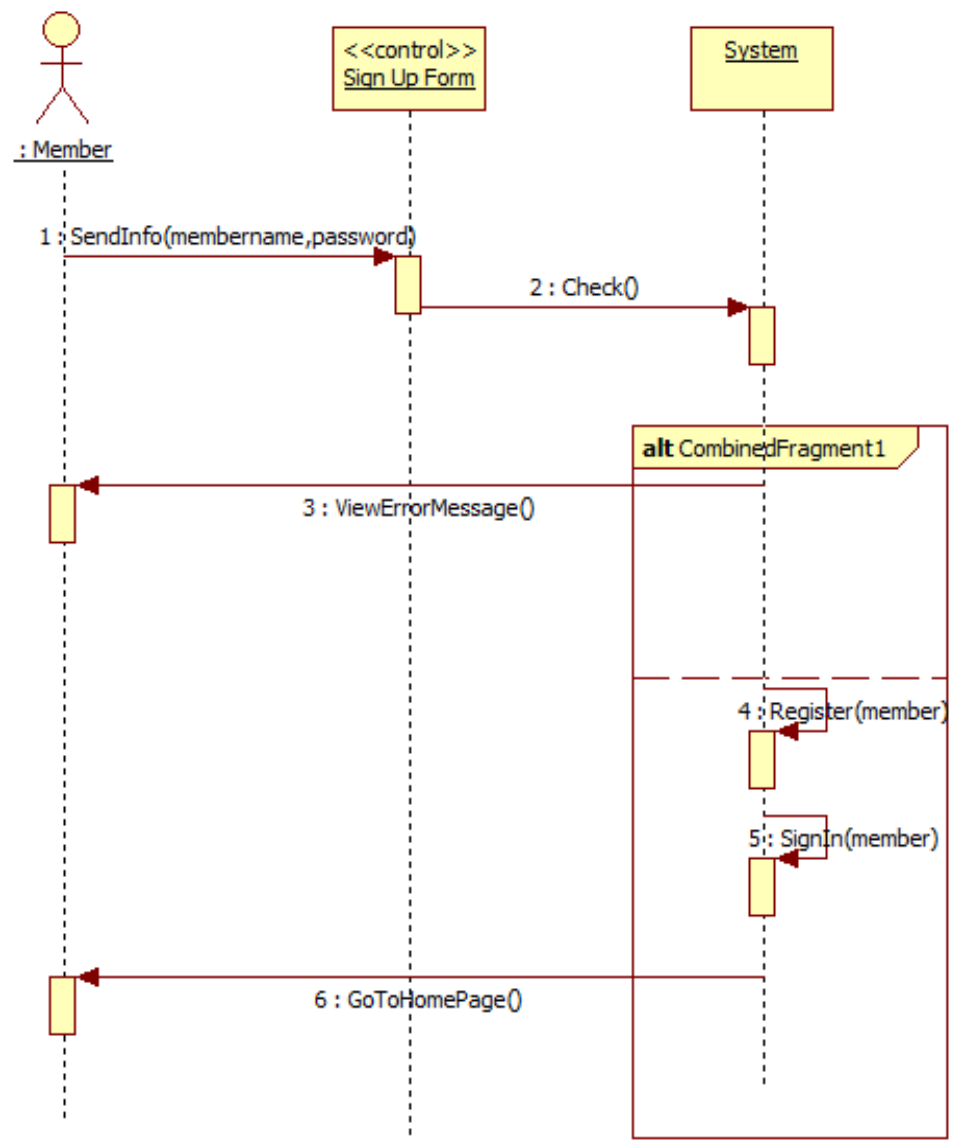


Figure 4.2 Sign Up Sequence Diagram

- Figure 4.3 shows the Sign In Sequence Diagram.

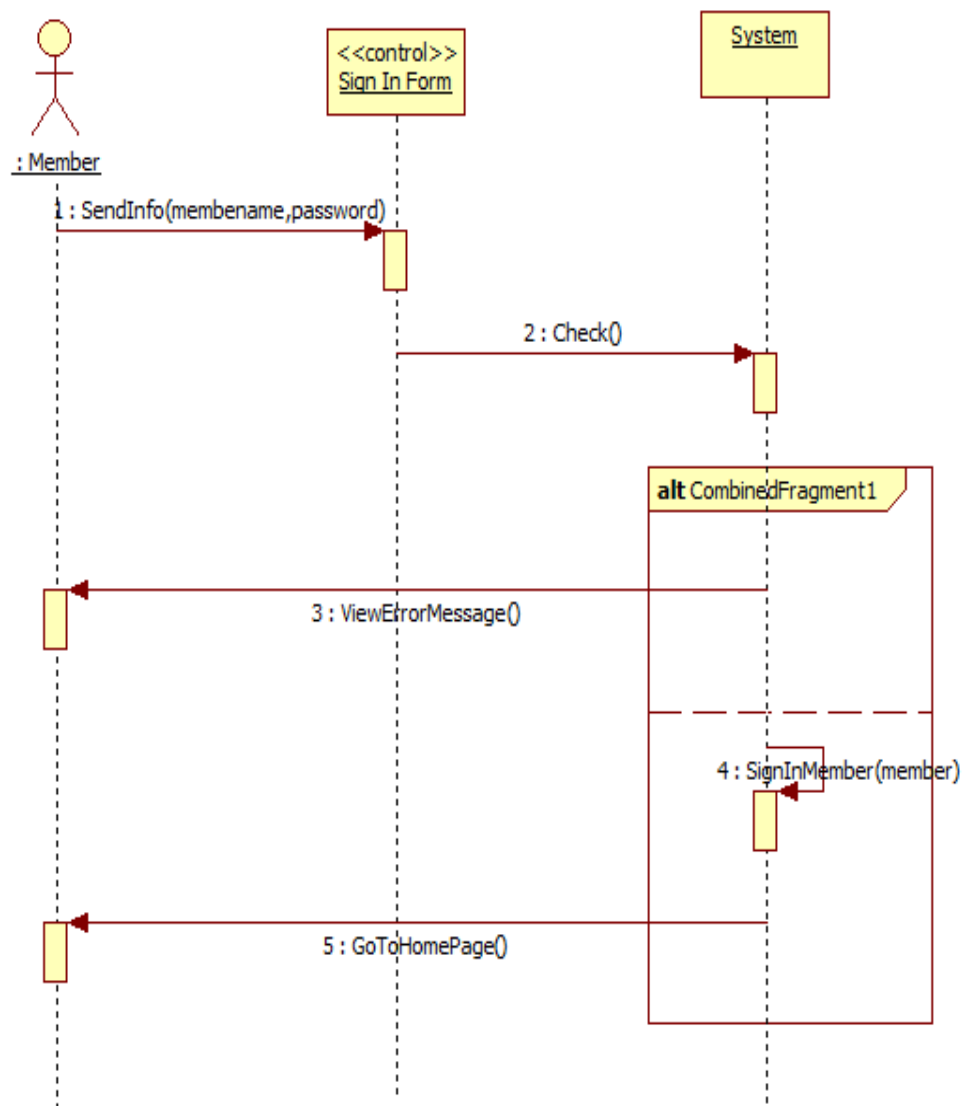


Figure 4.3 Sign In Sequence Diagram

- Figure 4.4 shows the Sign Out Sequence Diagram.

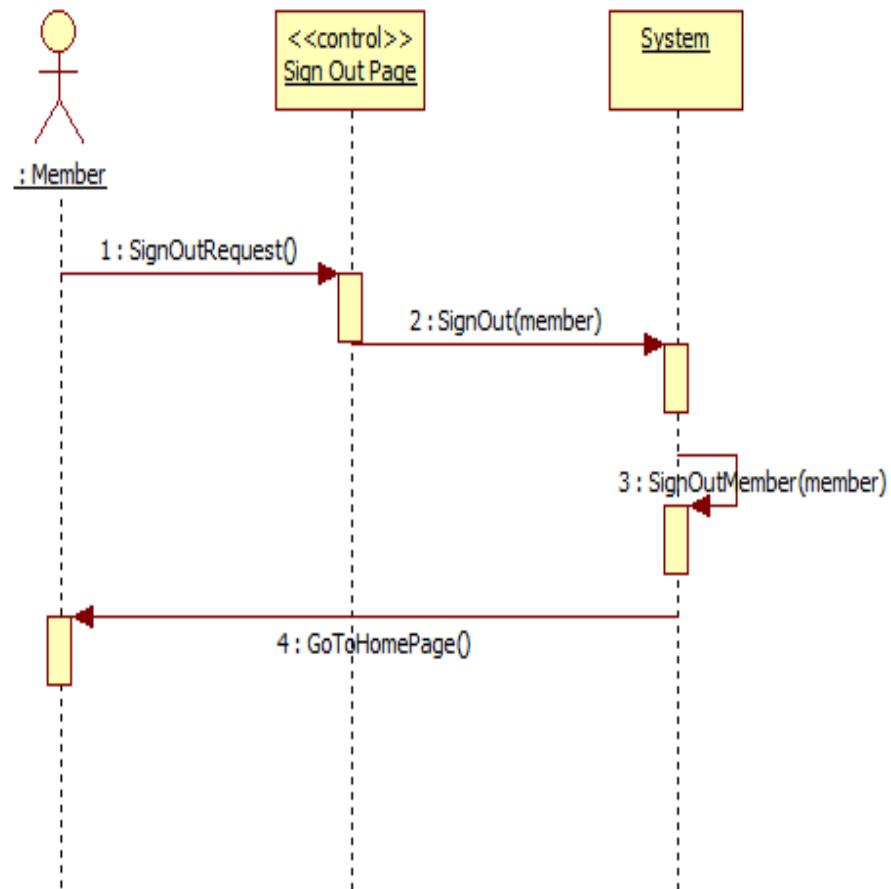


Figure 4.4 Sign Out Sequence Diagram

- Figure 4.5 shows the Edit Profile Sequence Diagram.

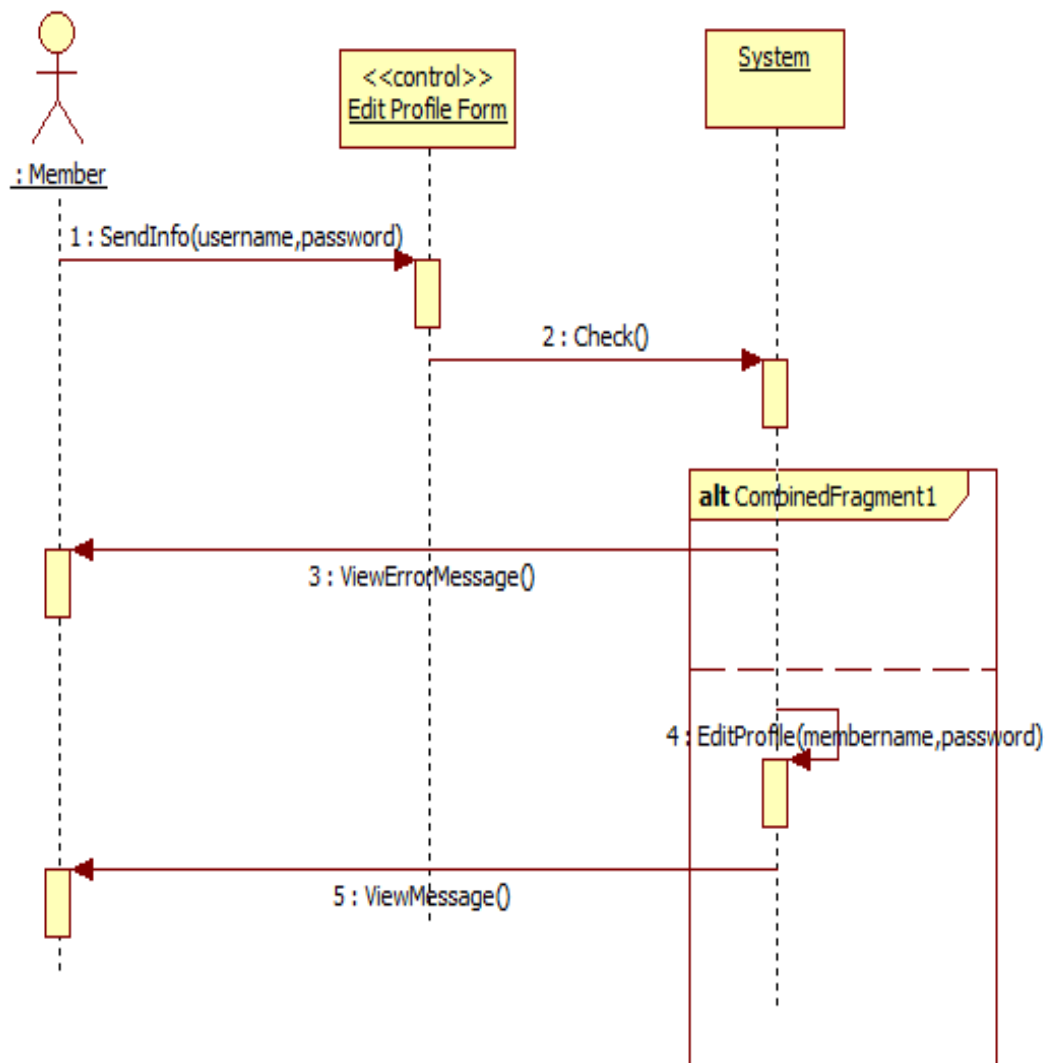


Figure 4.5 Edit Profile Sequence Diagram

- Figure 4.6 shows the Search Sequence Diagram.

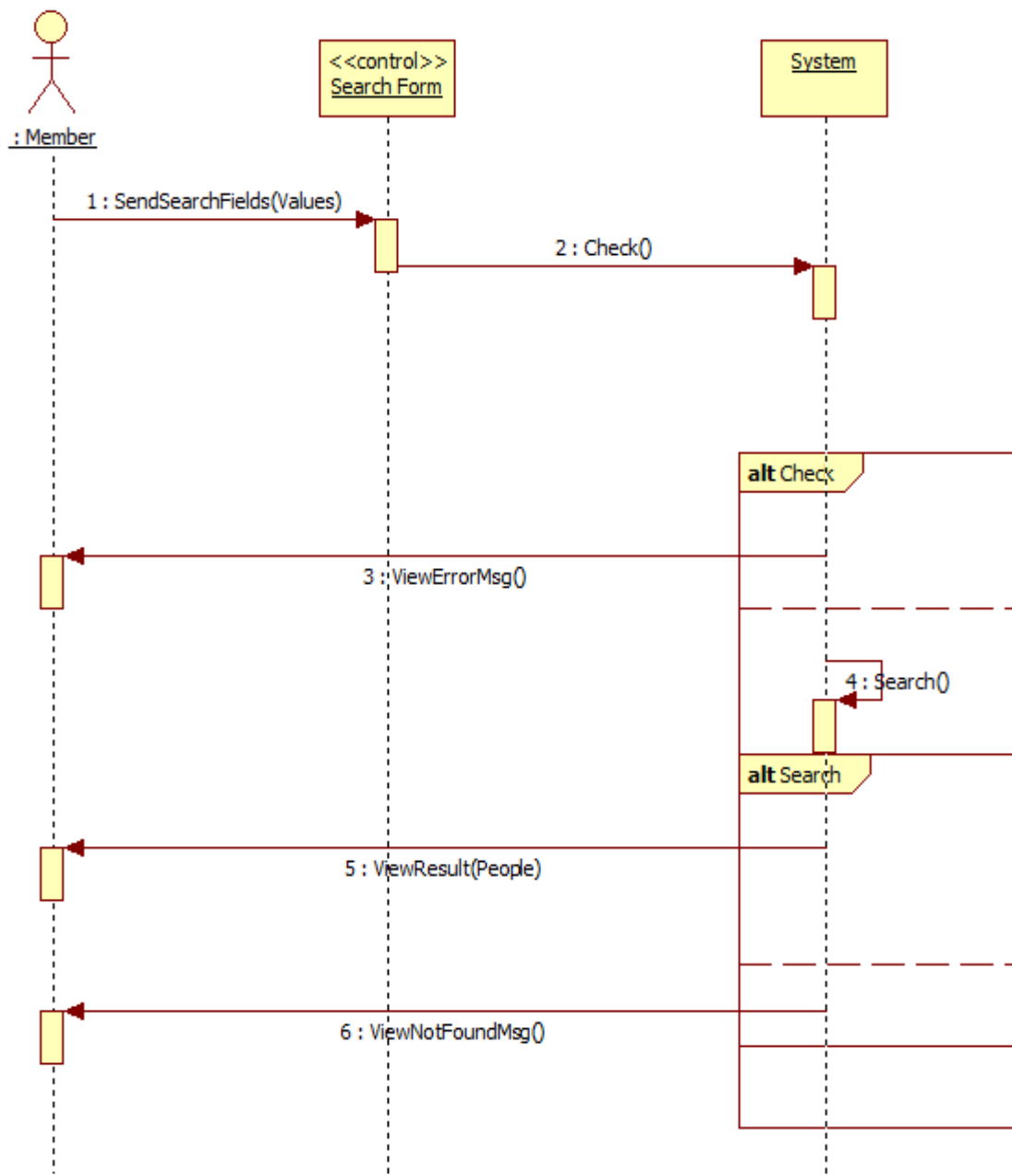


Figure 4.6 Search Sequence Diagram



- Figure 4.7 shows the Save Search Results Sequence Diagram.

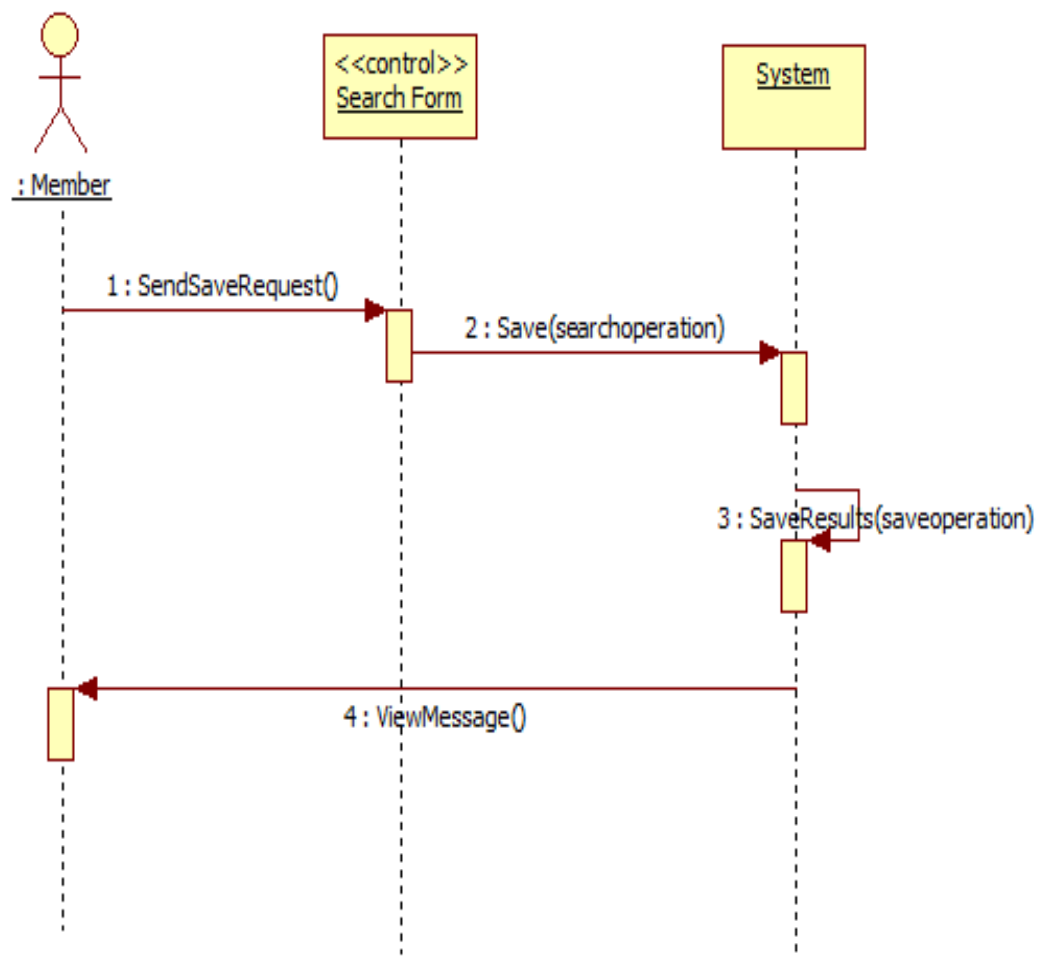


Figure 4.7 Save Search Results Sequence Diagram

- Figure 4.8 shows the View History Sequence Diagram.

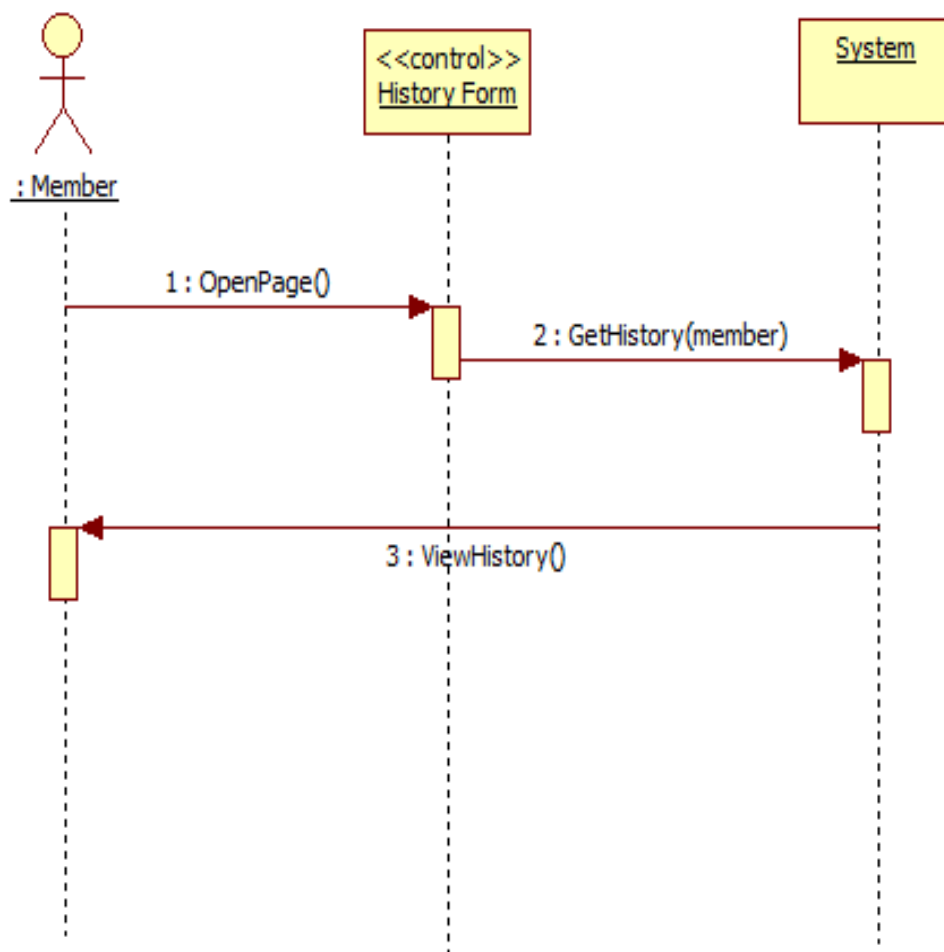


Figure 4.8 View History Sequence Diagram

- Figure 4.9 shows the View Saved Search Operation Sequence Diagram.

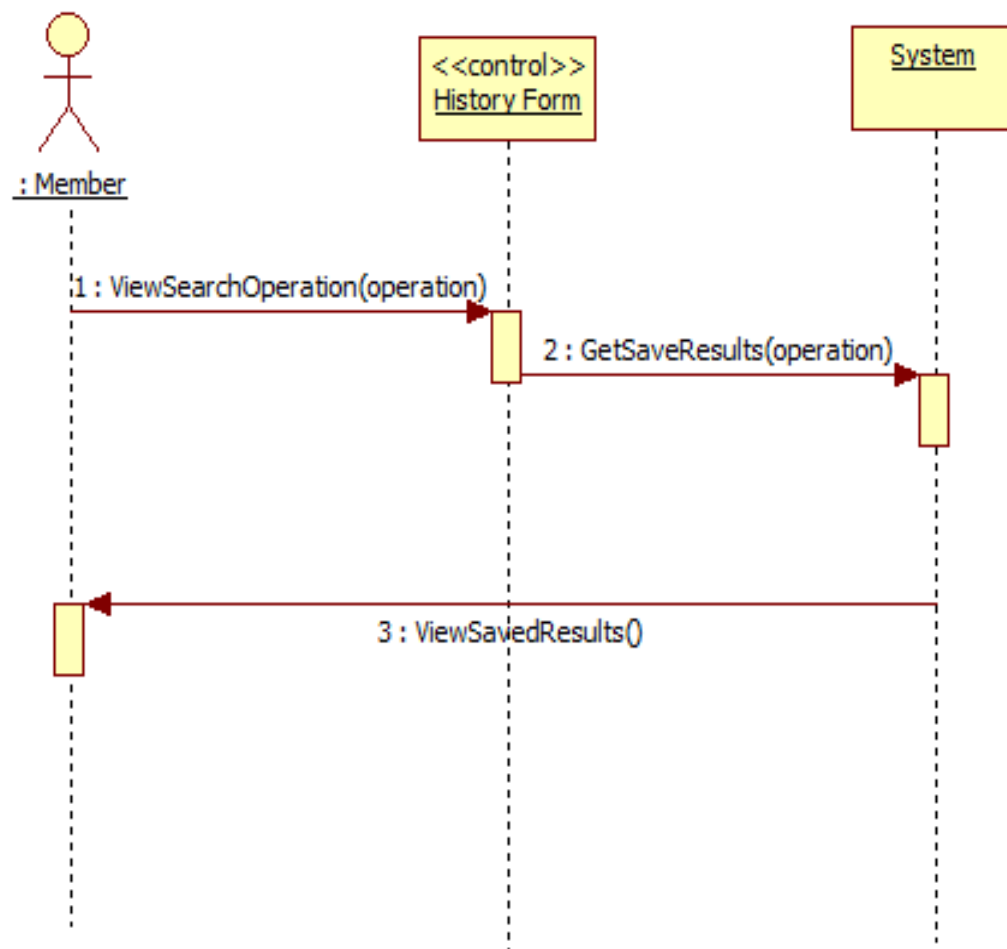


Figure 4.9 View Saved Search Operation Sequence Diagram

- Figure 4.10 shows the Clear History Sequence Diagram.

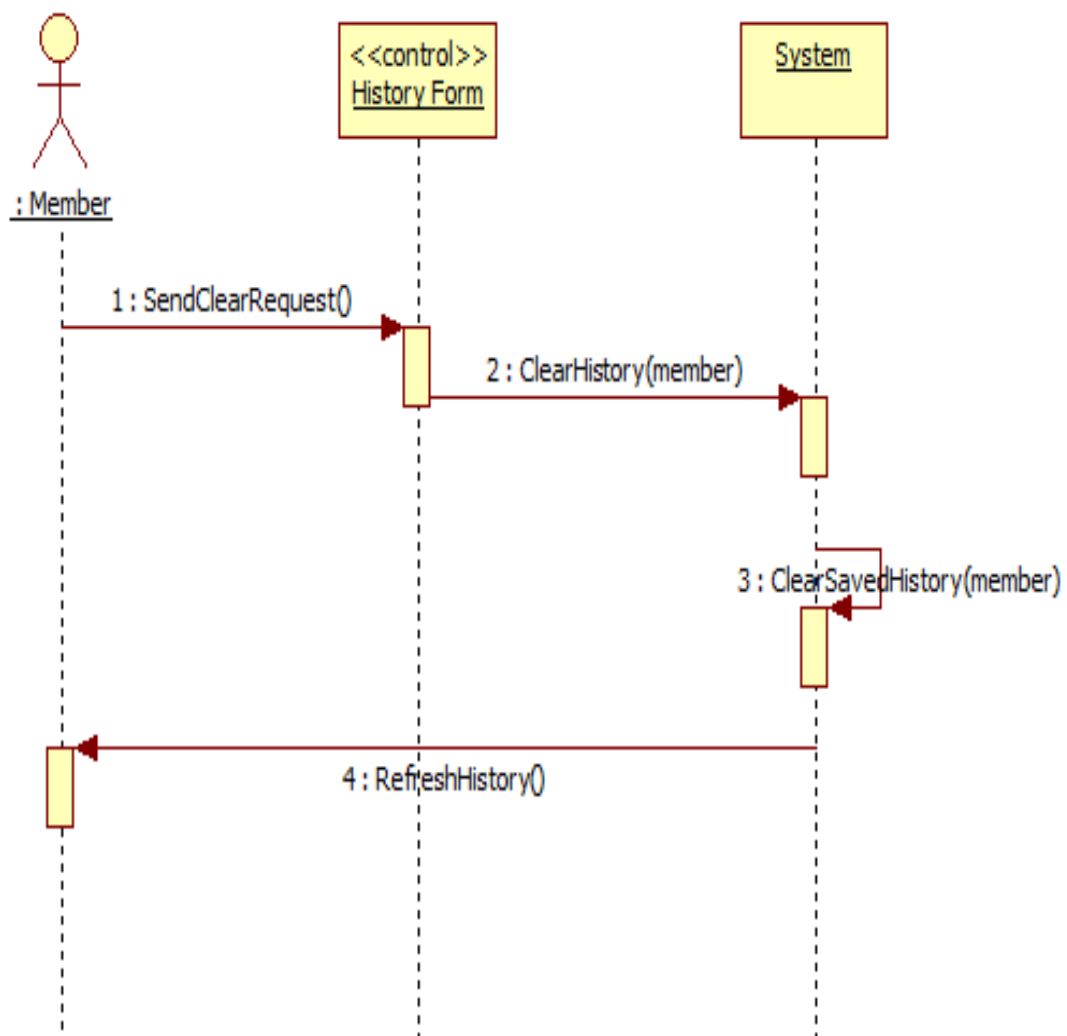


Figure 4.10 Clear History Sequence Diagram

- Figure 4.11 shows the Delete Saved Search Operation Sequence Diagram.

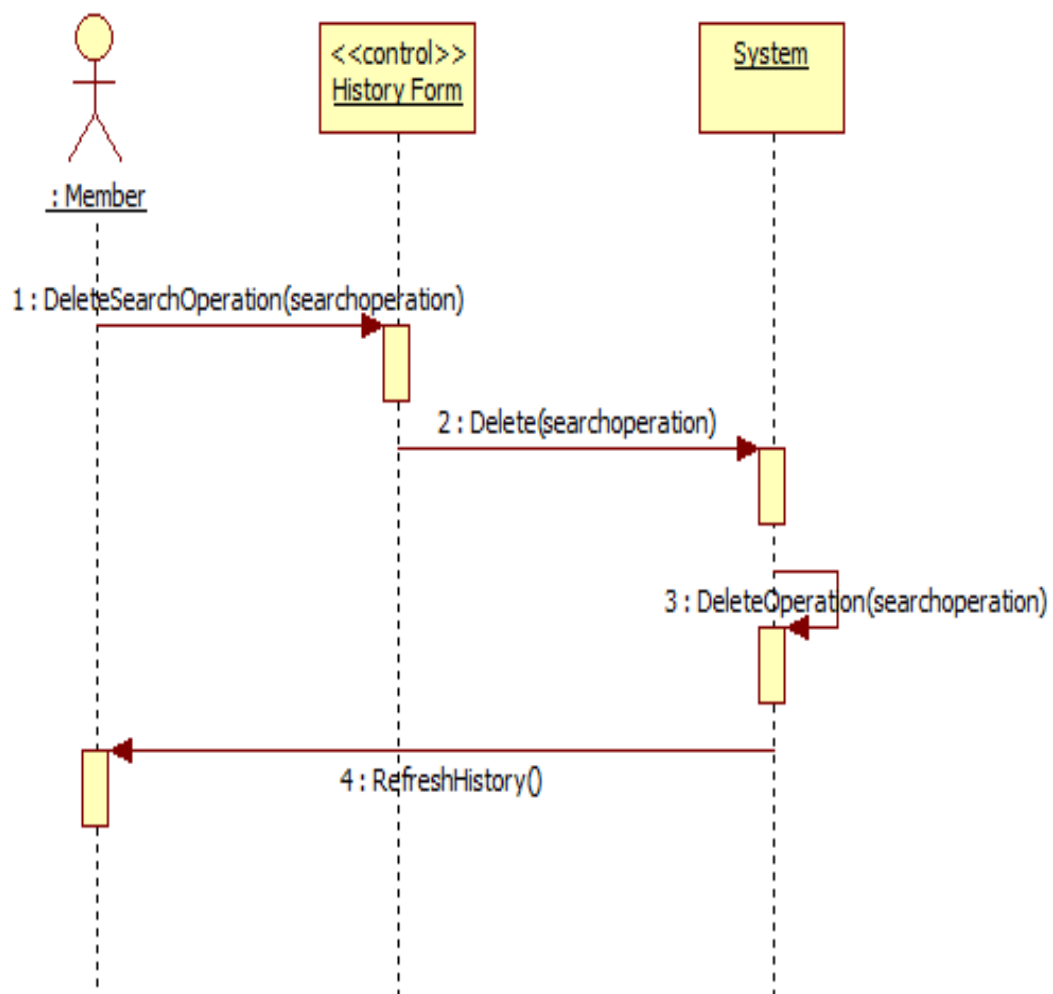


Figure 4.11 Delete Saved Search Operation Sequence Diagram

## 4.4 System Design:

In addition to previous database tables we have added these tables to manage search operation.

### Database Tables:

- Member Table: this table stores a registered member's logging in credentials, it contains member\_id field as a primary key, role\_id field as a foreign key to the Role Table, member\_name field and member\_password field. (as shown in figure 4.12)

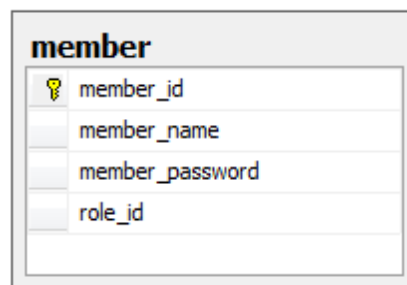


Figure 4.12 Member Table

- Role Table: this table stores possible roles that different registered member can get e.g. admin or normal user; it contains role\_id field as a primary key and role\_name field. (as shown in figure 4.13)

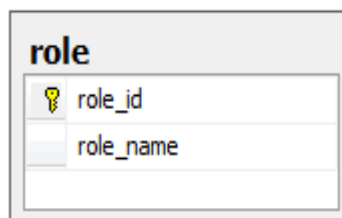


Figure 4.13 Role Table

- Searchop Table: this table stores search operations done by a specific member, it contains searchop\_id field as a primary key, member\_id field as a foreign key to the Member Table, date field and query field. (as shown in figure 4.14)


searchop	
	searchop_id
	date
	query
	member_id

Figure 4.14 Search Operation Table

- Searchhistory Table: this table stores the results of a specific saved search operation, it contains searchop\_id field as a foreign key to the Searchop Table and user\_id field as a foreign key to the User Table. (as shown in figure 4.15)

searchhistory	
	searchop_id
	user_id

Figure 4.15 Search History Table

## 4.5 System Implementation:

### Used Techniques:

- Microsoft SQL Server 2005
- Visual C# with Microsoft Visual Studio 2010
- LINQ to SQL

Now we have a website to search through stored CVs.

### Interfaces:

- Main Form: The first form displayed to the member. (shown in figure 4.16)



## WHAT'S LINKED OUT ?

A tool that enables you to search [LinkedIn](#) Profiles for people who satisfy your customized criteria.  
Available search areas include General Information, Education, and Experience.  
And more is coming !!!

Not a member yet ? [Sign Up](#) Now !!!

Member Name	<input type="text" value="admin"/>
Password	<input type="password" value="...."/>
<input type="button" value="Sign In"/>	

Figure 4.16 Main Form

- Search Form: The form used by the member to search. (shown in figure 4.17)

## WELCOME TO LINKED OUT WEBSITE !

just enter your search criteria and we will show you the result

[Hint: separate mutple values with commas]

<b>BASIC INFORMATION</b>	<b>EDUCATION</b>
Industry <small>comma means [or]</small> <input type="text"/>	Degree <small>comma means [or]</small> <input type="text"/>
Location <small>comma means [or]</small> <input type="text"/>	Major <small>comma means [or]</small> <input type="text"/>
Languages <small>comma means [and]</small> <input type="text"/>	Organization <small>comma means [or]</small> <input type="text"/>
Skills <small>comma means [and]</small> <input type="text"/>	Gradution Year <small>comma means [or]</small> <input type="text"/>
<b>ADDITIONAL INFORMATION</b>	<b>EXPERIENCE</b>
Courses <small>comma means [and]</small> <input type="text"/>	Job Title <small>comma means [or]</small> <input type="text"/>
Certificates <small>comma means [and]</small> <input type="text"/>	Organization <small>comma means [or]</small> <input type="text"/>
	Duration in years <small>comma means [or]</small> <input type="text"/>
<input type="button" value="Submit"/>	<input type="button" value="Reset"/>

Figure 4.17 Search Form



- Search Result Form: The form that displays the search result. (shown in figure 4.18)

## WELCOME TO LINKED OUT WEBSITE !

just enter your search criteria and we will show you the result [Back to search](#)

Save Search Results

Name	Profile	Industry	Location	Languages	Skills	Courses	Certificates	Education	Experiment
Anas Batikhi	<a href="#">Visit</a>	Computer Software	United Arab Emirates		Software Development   Business Analysis   Microsoft SQL Server   SQL   Project Management			MSP/PRINCE2 / Managing Successful Programmes/ PRINCE2 Practioner / SPOCE Project Management Training   MSc / Business IT / Kingston University   Computing & Information Systems / Business IT / University of Brighton	Head of Technology / AccuMed Practice Management   Founder / iWebManage.com   Executive Director, IT & Systems Development / Millennium Global Investments   Technical Project Manager / Millennium Global Investments
Abd Allah Diab	<a href="#">Visit</a>	Computer Software	United Arab Emirates	Arabic   English   German	C#   C++   Qt   Python   Visual Basic   Java   ASP.NET   PHP   JavaScript   HTML5   jQuery   Drupal   Django   Linux   Windows   SQL Server   Oracle SQL   PostgreSQL   MySQL   SQLite			Bachelor of Engineering (B.Eng.) / Software Engineering / Damascus University	Technical Manager / YouGotaGift.com   Manager & Co-founder / iCommunity   Software Engineer & Developer / Syrex FZ-LLC   Tutor / Syrian Computer Society   Software Engineer & Developer / Al Hadara Soft   Senior Web Developer / Brightsoft
Amr El-Nasser	<a href="#">Visit</a>	Computer Software	United Arab Emirates	English   Arabic				* / Computer Science / Yarmouk University   * / Computer Science / Optimal - PMP Course	Technical Team Leader / Global Information Technology - Dubai   Senior J2EE Developer / Integrant INC   Senior Java Developer / ETQ

Figure 4.18 Search Result Form

- History Form: The form that displays the History. (shown in figure 4.19)

LINKED OUT

Edit Profile
Sign Out

Home

Search

History

Search History

[ your saved search results ]

Clear History

Date	Query	View Results	Delete
5/14/2013 9:29:36 PM	industry : Computer Software   location :   languages :   skills :   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle : Developer,Developer and Systems Administrator,Developer - Consultant   expOrg :   expDate : 1,2,3,4,5	<a>view</a>	<a>Delete</a>
5/14/2013 9:19:56 PM	industry : Computer Software   location : Syria   languages : Arabic,English,French   skills : PHP   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle :   expOrg :   expDate :	<a>view</a>	<a>Delete</a>
5/14/2013 2:48:23 PM	industry : Computer Software   location : United Arab Emirates   languages : Arabic,English   skills :   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle :   expOrg :   expDate :	<a>view</a>	<a>Delete</a>
5/14/2013 2:29:45 PM	industry : Computer Software   location : United Arab Emirates   languages : Arabic,English   skills : PHP   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle :   expOrg :   expDate :	<a>view</a>	<a>Delete</a>
5/13/2013 11:41:29 PM	industry : Computer Software   location : United Arab Emirates   languages :   skills :   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle :   expOrg :   expDate :	<a>view</a>	<a>Delete</a>
5/13/2013 12:32:41 PM	industry : Computer Software   location :   languages :   skills :   courses :   certificates :   degree :   major :   eduOrg :   eduDate :   jobtitle :   expOrg :   expDate :	<a>view</a>	<a>Delete</a>

Figure 4.19 History Form

## Conclusions and Future Work

### Conclusions:

- ✚ Online networks are a repository of massive quantity of useful information. Unluckily we cannot get benefit of it because it's in a text format. Mining in these texts, Finding the exciting information and organizing it in a machine-readable format are considered a genius step towards building smart applications.
- ✚ While manipulating large amounts of data it's possible to have several types of problems. The critical point here is to figure out efficient and effective manners to face those problems.
- ✚ As Edison said: "I start where the last man left off." it's a common trend in the software development to build systems implicating existing software units. In our project we used various available libraries to construct the product by assembling ready-made units together.

### Future Work:

Here are some ideas to improve this system:

- ✓ Since the system deals with a huge quantity of data an important future work is to make it distributed application so the work can be divided between multiple servers which results in a faster processing.
- ✓ During the software development we have always tried to minimize the time it takes to perform required arithmetic calculations and logical comparisons. However, the consumed time was still very long due to huge quantity of the input data, so, it's a good thing to stay searching for less-complexity algorithms.
- ✓ It's possible to use GOOGLE Map API in the Semantic Similarity calculating step to find similarities among data representing locations (countries and cities).
- ✓ This software was developed to organize extracted data in an appropriate structure to enable applying it in further applications. One of these applications was the search engine we have created but there still a lot of projects that can be built upon our well structured database. Examples are Data mining systems, One-to-One marketing websites and Statistical researches.

## REFERENCES

1. About. *LinkedIn*. [Online] [Cited: 2 26, 2013.] <http://www.linkedin.com/about-us>.
2. **Aron Culotta, Ron Bekkerman and Andrew McCallum**. *Extracting social networks and contact information*. s.l. : University of Massachusetts - Amherst, 2004.
3. **YUVAL MERHAV, FILIPE MESQUITA, DENILSON BARBOSA, WAI GEN YEE and OPHIR FRIEDER**. *Extracting Information Networks from the Blogosphere*. s.l. : Georgetown University, 2012.
4. Profile API. *LinkedIn Developers*. [Online] [Cited: 2 26, 2013.] <https://developer.linkedin.com/documents/profile-api>.
5. Authentication. *LinkedIn Developers*. [Online] [Cited: 2 26, 2013.] <https://developer.linkedin.com/documents/authentication>.
6. HTML Agility Pack. *CodePlex*. [Online] [Cited: March 07, 2013.] <http://htmlagilitypack.codeplex.com>.
7. HTML Parsing in C# using HTML Agility Pack. *Code Problem*. [Online] [Cited: March 07, 2013.] <http://www.codeproblem.com/articles/languages/81-net-framework/74-html-parsing-in-c-using-html-agility-pack>.
8. About WordNet. *Princeton University*. [Online] [Cited: June 4, 2013.] <http://wordnet.princeton.edu>.
9. WordNet. *Wikipedia, the free encyclopedia*. [Online] [Cited: June 04, 2013.] <http://en.wikipedia.org/wiki/WordNet>.
10. Hunspell. *Wikipedia, the free encyclopedia*. [Online] [Cited: July 29, 2013.] <http://en.wikipedia.org/wiki/Hunspell>.
11. Hunspell. *SourceForge*. [Online] [Cited: July 28, 2013.] <http://hunspell.sourceforge.net/>.