Implementing Language Independent Dictionary and spell checkers for Bangla and English language using Web Service.

Computer science Undergraduate Thesis

By

Mohammad Faruk
ID: 03201014

Thesis Supervisor
Dr. Mumit Khan

Acknowledgements

I have been fortunate to have very dedicated supervisors working with me this Year. I would like to sincerely thank Dr. Mumit Khan for his enthusiasm, advice and inspiration throughout this thesis. His efforts in implementing a language independent dictionary saved me a lot of time.

I am also grateful to Abdur Rahman and Fahim, research programmer of CRBLP for their continued help while working with Bangla language.

Finally, sincere thanks to my family for their support, including my younger brother Hafizur Rahman showing me Bangla typing and my friends collaborating me all the times.

TABLE OF CONTENTS

Chapter 1: Introduction:

Every country has its own language. Dictionaries are used for a wide variety of purposes, by people with various backgrounds in the language. If we want to interact with other languages we need to use dictionaries. The potential advantage of a computerized dictionary is that there is an interface between the user and the complete dictionary. This means that the way information is delivered to the user can be altered significantly, for much less cost than the creation of another paper dictionary edition. Electronic dictionaries "e-dictionaries" also have the potential to allow users to customize what and how information is presented to suit their preference. This immediately means that the dictionary can deal with a broader range of intentions and a greater range of Language competency than is possible with printed copy.

How well a dictionary application satisfies a diverse range of users, depends on how Well the developers use the potential of working in an electronic medium. Despite some significant research in the area of computerizing dictionaries, there has been little effort in addressing the challenge of taking this to real speakers and language learners.

This has been a focus of this project, to not be satisfied with simply a better structured dictionary, but also make the dictionary language independent.

1.1 Working for Language Independent Dictionary:

In Bangladesh two types of Languages are frequently used. These are English and Bangla. There are huge English web dictionaries. But not enough dictionaries implemented for Bangla Language. My challenge was to implement a dictionary service which will work for both English and Bangla Language. There may have four probable dictionaries in Bangladesh. Theses are **English to English**, **English to Bangla**, **Bangla to English** and **Bangla to Bangla**. I have implemented a service which will work for all types.

Since the dictionary is language independent it will work for other languages. For example if I want to implement a dictionary for Arabic language like Bangla to Arabic or English to Arabic dictionary it is also possible. Only I need to add a new XML database. The program will remain same.

1.2 Thesis Overview:

The challenge for the work of this thesis has been to creatively (intelligently) address the following areas: making the dictionary language independent, data processing, storing this extracted information in a suitable format , displaying this dictionary information in

a usable way and generate suggestion list for any wrong input . This thesis follows the work in these areas of research and how they relate to each other. The organization is as follows:

Chapter 2: Reviews the past work in the fields of electronic dictionary research. In discussing what is lacking in much of the other dictionaries, I will discuss some of the challenges in attempting to make the dictionary language independent, and summaries the approaches taken this thesis.

Chapter 3: In this chapter I will discuss about the requirement specification of the software. I will also discuss about the architectural Strategies i.e. the software or hardware needs to implement the dictionary.

Chapter 4: In this chapter I will show the schema design of the Xml database. Then I will show some architectural diagram for client and server application.

Chapter 5: This chapter contains the detail implementation of the software. Here firstly I have shown the algorithms that have been used. Then I have shown the procedure to implement the software using available algorithms.

Chapter6: This chapter contains the result of the dictionary. I have just put some snap shots of the taken outputs and the discussion of outputs

Chapter7: This chapter concludes the paper with discussing the problems of the thesis and future improvements.

Chapter 2: Related works

There are huge well formatted electronic dictionaries. I studied three or four papers on dictionaries.

2.1: Past works

Some of the Computer science students of The University of Sydney (Australia) worked on "Intelligent processing, storage and Visualization of Dictionary Information". They name the dictionary Kirrkirr. They implemented the dictionary for Warlpiri language. [6]

They begins with a discussion of the issues associated with using commercial Database systems for the storage of dictionary information and explained that size of the database file increase with dictionary data. That's why this approach was considered inappropriate for their thesis. They used XML as dictionary database.

The most visual part of their application is the representation of the network of words and their relationships via a graph. They discussed the 'spring algorithm' approach to animated graph layout, its relative merits and other issues that arose in applying it to dictionaries. They also discussed the special considerations in constructing a dictionary interface for Warlpiri such as reducing the reliance on the written word, by including sounds, pictures and point-and-click (rather than typed) interactivity with words.

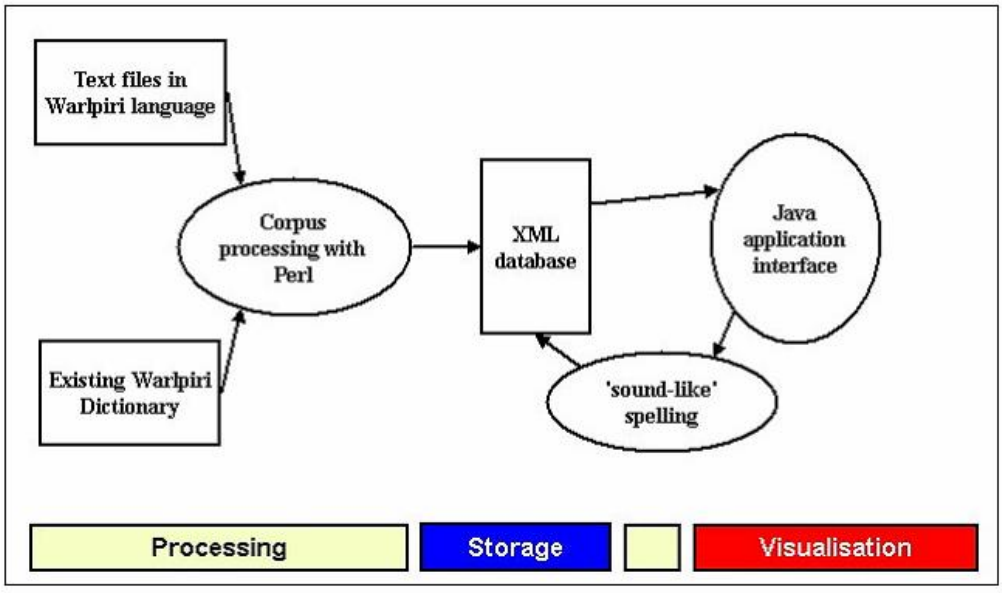


Figure 1: System Overview

Some students of NANYANG Technological University worked on the Kirrkirr dictionary to solve the performance problems namely: slow start-up time and large

memory usage. Revamping the underlying data and usage of the Extensible Query Language and Document Object Model lead to a smaller start-up index file and makes Possible the loading of the dictionary word in small chunks. This resulted in a very visible improvement in the start-up time, which was reduced from 7 minutes to 1 minute. **[7]**

Donny Mack in his paper "Utilizing Goggle's Spell Check APIs on the Client and Server," showed how he implemented Spell checker using web service. He implemented the spell checker for English language using C# .net. They take a dirty string from the client part and send it to the service. Then groups the string into groups of individual words. If any wrong word is found then in the client part it shows the underline in the wrong word. **[1]**

Naushad UzZaman, the student of BRAC University in his undergraduate thesis work implemented Bangla spell checker. He showed some spell checking techniques and their comparison. Finally he used Double Metaphone phonetic encoding. Then he showed how to generate better suggestion list for the miss spelled word. **[8]**

2.2: Where I am Unique?

Reading the above papers and looking some other electric dictionaries I decided to implement a dictionary which will work as Language independent (section 1.1). To store data I have used XML database and I made a general schema for all language. I used web service to make it platform independent for the client. So no matter what programming language is used for the client application. The client application is platform independent from operating system view. It will support for any OS. I worked for spell checking for Bangla and English Language for wrong input. It also generates better suggestion list. So my work is the combination of some of the above works. My dictionary is Unique because of its Platform independent nature.

Chapter 3: Requirement specification

3.1 Programming Language:

1. Microsoft .NET Framework 2.0
2. Microsoft Visual Studio 2005

Writing system-level software that works consistently across various versions of Microsoft Windows is a challenging task. Plus, creating graphical user interfaces using the default (and free) Microsoft Foundation Classes is a daunting endeavor. Securing the software is also quite difficult.

The Microsoft .NET Framework v 2.0 provides a wonderful solution to these problems. Learning from the successes and failures of the Sun Java Platform, the Microsoft .NET Framework provides a similar system of byte code running in a virtual machine, with clearly defined security models, and an extremely rich programming library. The framework is available on all versions of Windows all the way back to Microsoft Windows 98.

Out of the box, .NET supports numerous programming languages, the default being C# ASP.net Web Service which is the language chosen to implement The Online Dictionary. Numerous tools are available for Rapid Application Development, as well as libraries to ease database connectivity – all of which should be availed when developing the software. The Microsoft .NET Framework, by default, has all the libraries necessary to implement my Dictionary, so no external software libraries are necessary.

3.1.1 Web Service [13, 14, 15]

The following benefits of web service convinced me to implement my project using web service.

A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) use the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

Web Services are self-describing software modules which encapsulates discrete functionality. Web Services are accessible via standard Internet communication protocols like XML and SOAP. These Web Services can be developed in any technologies (like c++, Java, .NET, PHP, Perl etc.) and any application or Web Services can access these services. So, the Web Services are loosely coupled application and can be used by applications developed in any technologies. For example, I have heard of people developing Web Services using .NET applications technologies and using the Web Services in JAVA applications.

Web Services are self describing applications, which reduce the software development time. This helps the other business partners to quickly develop application and start doing business. This helps business to save time and money by cutting development time.

Web Services automatic discovery mechanism helps the business to easy find the Service Providers. This also helps your customer to find your services easily. With the help of Web Services your business can also increase revenue by exposing their own Web Services available to others.

3.2 The extensible Markup Language (XML) [16, 17, 18]

XML is a simplification of the Standard Generalized Markup Language (SGML), which is a standard system for defining the content and format of an electronic document.

•**Extensibility**: allowing users to define their own tags (or attributes) to suit the data being represented.

•**Structure**: allowing nested structures of any depth, hence being well suited to lexical database schemas.

•**Validation:** allows the structure of the data to be validated before use by applications.

The need for this sort of markup for the web has influenced rapid development with XML. The advantage of using a globally recognized markup language for the dictionary is that the research benefits from the various general purpose tools available, rather than have to develop specific tools. The dictionary file also becomes more portable than a plain text file, because now for anyone to use the data all they need is access to any XML tools (if not the same products used in this project).

3.3 IIS (Internet Information Server) 6.0 [21]

The web server that specified as the development server must have the .NET Framework and IIS (Internet Information Server) 6.0 or later installation on it. Since the software will be developed first in the local computer IIS is required. After the entire development the dictionary service will be posted on to the Web Server.

3.4 Stylus Studio: [19]

Stylus Studio's support for XML, XQuery, XML Schema, XPath, SQL/XML, HTML/XHTML, SOAP, WSDL, UDDI, EDI and Legacy Data Integration, Web Services, Java/J2EE & Microsoft .NET. To make the schema of my dictionary database and to edit XML file I used stylus studio.

3.5 Platform Requirements:

Since the dictionary service will be developed by Web Service in .NET Framework any windows operating system like ***Windows Vista; Windows XP*** will support it.

From the client side the system is totally platform independent. Clients can use the dictionary service from any platform like Linux, Windows 98, Windows XP, Win Vista, and Mac OS.

3.6 Data source:

To implement the dictionary I needed data to make my database. I collected data for English to English dictionary from Wordsmith dictionary **[12]**. Because, it is data representation of the web dictionary is almost my dictionary. For English to Bangla, Bangla to English and Bangla to Bangla dictionary I used Bangla Academy dictionaries **[9, 10, and 11**].

Chapter 4: Design

4.1 Schema Design

To implement the dictionary I designed a well formatted XML schema. I took the design concept from **[20].**

Building a Dictionary Info XML File for the Dictionary Program:

Dictionary root Element:

```
<dictionary>
.
.
.
</dictionary>
```

Type : Element
Name : Dictionary
Min Occur: 1
Max Occur : 1
Null able : False

This is simply the root element of the specification file. All other specification elements are contained within it.

Dictionary Entry:

```
<dictionary>
 <entry>
   ……..
   ……..
   ……..
 </entry>
</dictionary>
```

For each word <entry> element is used. This element is the first child of the <dictionary> element. There the total number of <entry> element is equal to the number of words present in the XML.

Type : Element
Name : entry
Min Occur: 1
Max Occur : unbound
Null able : False

Headword and Iniquities: This is necessary for the specification file. There should one headword per entry. If multiple headwords are spelled the same way, then iniquities is needed to distinguish them.

```
<dictionary>
  <entry>
   <hw unq="1">bank</hw>
    ……..
  </entry>

  <entry>
   <hw unq="2">bank</hw>
    ……..
  </entry>
 ……..
 ……..
</dictionary>
```

```
Type: Element
Name: hw
Data type: XSD: string
Max occur: 1
Min Occur: 1
```

```
Type: attribute
Name: unq
Data type: string
Restriction: required
```

Pronunciations: After defining the headword element inside entry element pronunciation element <pro> is defined.

```
Type: Element
Name: pro
Data type: XSD:string
Max occur: 1
Min Occur: 1
```

Example:

```
<dictionary>

 <entry>
   <hw unq="1">bank</hw>
   <pro>baengk</pro>
   -----------------
  </entry>
  --------
```

</dictionary>

Compound word:

Type: Element
Name: compound
Data type: XSD:string
Max occur: 1
Min Occur: 0

This tag is used for the Bangla words.

```
<entry>
   <hw unq="1"></hw>
   <compound></compound>
   -----------------
  </entry>
```

Gender:
This tag is used to store the gender of the headword. It is placed after compound tag.

Type: Element
Name: gender
Data type: XSD:string
Max occur: 1
Min Occur: 0

<gender>পুং লিঙ্গ</gender>

source_language:
This tag is placed after gender. It is especially for the Bangla words.

Type: Element
Name: **source_language**
Data type: XSD:string
Max occur: 1
Min Occur: 0

Example:

<source_language>সংস্কৃত</source_language>

Derivation:
This tag is also used for Bangla words.

Type: Element
Name: derivation
Data type: XSD:string
Max occur:1
Min Occur:0

<derivation>কথ্+অক</derivation>

Parts of Speech:

Parts of speech element is required for each entry element. A word may have several parts of speech example: noun, adjective. All are written through the <parts_of_speech> element. If there is one more parts of speech element then one more <parts_of_speech> will be defined inside entry.

For each entry element <parts_of_speech> design:

Type: Element
Name: parts_of_speech
Min Occur: 1
Max Occur: unbound
Null able: False

Example:

```
<dictionary>

 <entry>
   <hw unq="1">bank</hw>
   <pro>baengk</pro>
   <parts_of_speech>
    ----------------
    ----------------
   </parts_of_speech>

   -----------------
  </entry>
  --------

</dictionary>
```

Parts Of Speech Name:

Each parts of speech element contains <pos_name> element.
Design of <pos_name> for each <parts_of_speech>:

Type: Element
Name: pos_name
Data type: XSD:string
Max occur: 1
Min Occur: 1

Example:

```
<parts_of_speech>
    <pos_name>noun</pos_name>
    ----------------
 </parts_of_speech>
```

POS element:
  As one part of speech a word may have one or more meaning. So they are written inside the <pos> element. If one part of speech has more meaning then more <pos> elements are written.

Design of <pos>:

Type: Element
Name: pos
Min Occur: 1
Max Occur: unbound
Null able: False

Example:

```
<parts_of_speech>
    <pos_name>noun</pos_name>
    <pos>

    </pos>

    <pos>

    </pos>
    ----------------
 </parts_of_speech>
```

Meaning element:

Inside the <pos> element <meaning> element is written which contains the definition of the word.

Design:

```
Type: Element
Name: meaning
Data type: XSD:string
Max occur: 1
Min Occur: 1
Example:

<parts_of_speech>
    <pos_name>noun</pos_name>
    <pos>
       <meaning>a heap or mass of something<meaning>

    </pos>

    <pos>

    </pos>
    ----------------
```

```
 </parts_of_speech>
```

Use in sentence element: **After** the <meaning> element the <use_in_sen> element is written which contains a sentence using the word.

Design of <use_in_sen>:

Type: Element
Name: use_in_sen
Data type: XSD:string
Max occur: 1
Min Occur: 1

Example:

```
<parts_of_speech>
    <pos_name>noun</pos_name>
    <pos>
       <meaning>a heap or mass of something<meaning>
       <use_in_sen>there were banks of stones by the road<use_in_sen>



    </pos>
    ---------
    ---------
<parts_of_speech>
```

Synonyms element: Each <pos> element contains one <synonyms> element. One word for a particular meaning may have several synonyms. They are written in the <syn> element. <syn> elements are define inside the <synonyms> element. The word may not have synonyms. In that case no <synonyms> element will be written.

Design of <synonyms>:

Type: Element
Name: synonyms
Min Occur: 0
Max Occur: 1
Null able: False
Design of <syn>

Type: Element
Name: syn
Data type: XSD:string
Max occur: unbound
Min Occur: 1

Example:

```
<pos>
       <meaning>a heap or mass of something<meaning>
       <use_in_sen>there were banks of stones by the road<use_in_sen>
```

```
      <synonyms>
         <syn>pile</syn>
         <syn>heap</syn>
         <syn>mass</syn>
      </synonyms>

   </pos>
```

Antonyms element: <antonyms> element is designed same as <synonyms> element. Different antonyms are defined by <ant> element.

Design of <antonyms>:

Type: Element
Name: antonyms
Min Occur: 0
Max Occur: 1
Null able: False

Design of <ant>

Type: Element
Name: ant
Data type: XSD:string
Max occur: unbound
Min Occur: 1

Figure: hierarchy of XML schema

4.2 Client and Service Design:

4.2.1 Three-Tier Architecture

My dictionary project is a three-tier software system. A basic diagram of the system architecture may be found below.

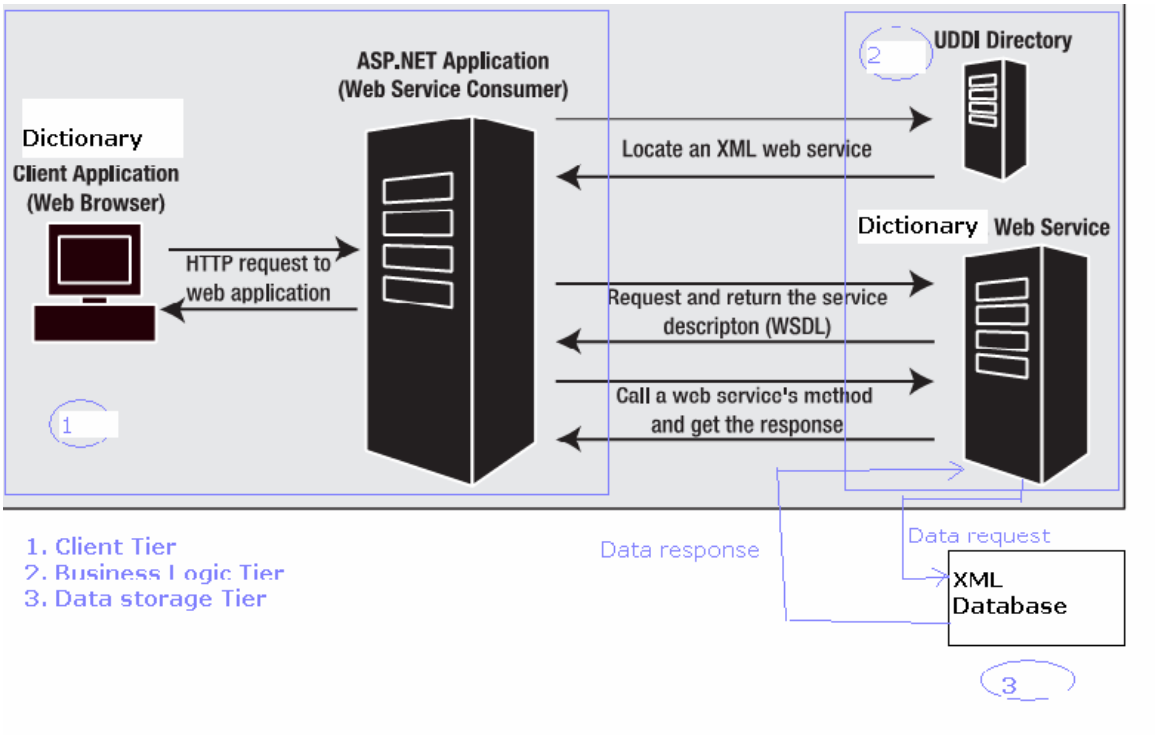


figure: Three-Tier Architecture

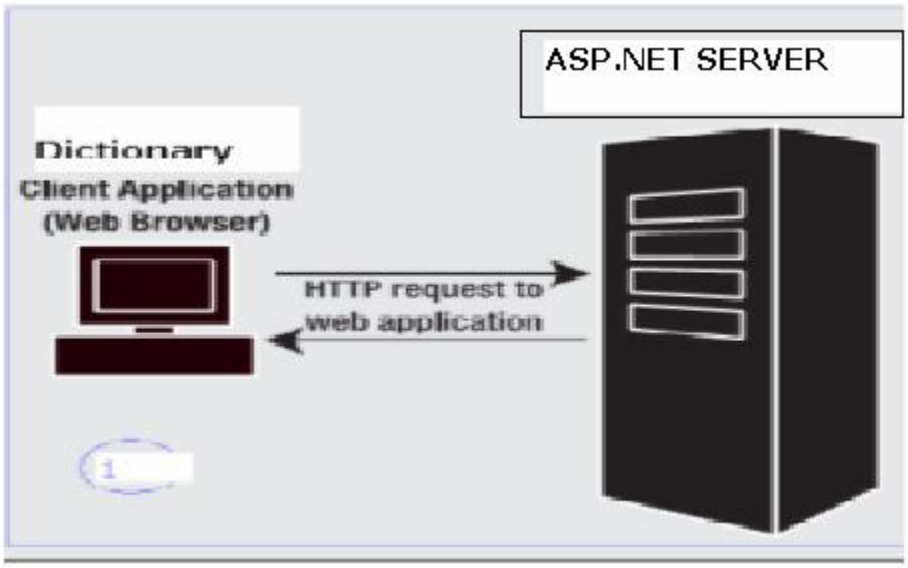
4.2.2: Client Design:

My client side is complete application. It can also be three –tier. The users of the dictionary give their inputs from web browsers. The Client Tier contains the main presentation of the dictionary. The users will get the following presentations from here:

Menu page: Here the users will get the name of the four types of dictionaries. Then the will select any types according to their choice.

English to English dictionary page: From this page the user will give input in a text field and press the submit button. Then he will see the output in the same page. Other dictionary pages work exactly like this page.

In the client application server there will have some business logic. The responsibility of this part is to send input as XML format to the web service. When the service responses it accepts. In my dictionary the web service will return a XML nodeList which contains words information's. Now the logical job of this part is to parse the xml document and display to the browser.



4.2.3: Service Design:

All of the business logics are implemented in the service part.

For a simple word search service request to the database for a word and take the response. After that it sends back to the client application.

If the users request for the words which starts with/ends with then the logical part will just invoke database and return the response to the client.

Generating suggestion list is the other important part of the service. If users give a wrong input the service will generate suggestion list and send back to the client. To do that it uses different algorithms

Chapter 5: Implementation

5.1 Algorithms:

The Metaphone or Double Metaphone algorithm is used for spell checking. If a user looks for a wrong word in not in the dictionary then to generate suggestion list this algorithm is used. For ranking the words of the suggestion list Edit distance algorithm is used.

5.1.1 Metaphone [4]

The Metaphone algorithm, proposed by Lawrence Philips, 1990, is also a system for transforming words into codes based on phonetic properties. Metaphone analyzes both single consonants and groups of letters called diphthongs, according to a set of rules for grouping consonants, and then mapping groups to Metaphone codes.

The Metaphone Rules

Metaphone reduces the alphabet to 16 consonant sounds:
B X S K J T F H L M N P R 0 W Y
That isn't an O but a zero - representing the 'th' sound.

Transformations
Metaphone uses the following transformation rules:

Doubled letters except "c" -> drop 2nd letter.
Vowels are only kept when they are the first letter.

B -> B unless at the end of a word after "m" as in “dumb"
C -> X (sh) if -cia- or -ch-
S if -ci-, -ce- or -cy-
K otherwise, including -sch-
D -> J if in -dge-, -dgy- or -dgi-
T otherwise
F -> F

G -> silent if in -gh- and not at end or before a vowel
In -gn- or -gned- (also see dge etc. above)
J if before i or e or y if not double gg
K otherwise
H -> silent if after vowel and no vowel follows
H otherwise

J -> J
K -> silent if after "c"
K otherwise
L -> L
M -> M
N -> N
P -> F if before "h"
P otherwise
Q -> K
R -> R
S -> X (sh) if before "h" or in -sio- or -sia-
S otherwise
T -> X (sh) if -tia- or -tio-
0 (th) if before "h"
Silent if in -tch-
T otherwise
V -> F
W -> silent if not followed by a vowel
W if followed by a vowel
X -> KS
Y -> silent if not followed by a vowel
Y if followed by a vowel
Z -> S

Initial Letter Exceptions

Initial kn-, gn- pn, ac- or wr- -> drop first letter
Initial x--> change to "s"
Initial wh--> change to "w"

5.1.2 Double Metaphone [5]

Lawrence Philips, 2000, also proposed double metaphone. Metaphone worked fine in most of the cases but there were a few cases that metaphone cannot handle Such as, Bryan (BRYN) was not matched to Brian (BRN). MacCafferey is encoded to

MKKF, an out-and-out bug. Retaining Soundex's choice of preserving the first letter (in Metaphone, only for words that started with vowels), "Otto" for example, cannot be matched to "Auto." More difficult to deal with, and contributing considerably to inelegance, are the consonants that are pronounced differently in different words. For example “gh” in light and rough.

English has a tendency to accumulate a large number of words from non-English sources, notably French, Latin, and Greek. When transliterating from the Greek alphabet, the letter that is pronounced “kh” is Greek (a sound that does not exist in English – think “chutzpah”), is spelled “ch” and pronounced “k”: “orchestra”, “chorus”, etc. Most importantly, some familiar names can just as plausibly be pronounced more than one way. Henry Kissinger and Kim Basinger are example of that type. Basinger is pronounced in both way as “Basin-gger” or “Basin-jer”.

These problems led Philips to propose another phonetic encoding, Double metaphone, which perform better but not perfect. Main improvement of this encoding is it will give two keys for words and names that can be plausibly pronounced more than one way.

For example, in case of Kuczewski, there are two ambiguous sounds, so "Kuczewski" now comes back as KSSK for the American version, "Kuhzooski," as well as KXFS for "Kutchefski." (‘X' is used to represent the "sh" sound, and ‘0', zero, to represent "th," as in original Metaphone.)

Example:
Spelling →SPLN
Spoiling →SPLN

5.1.3 Double Metaphone Phonetic encoding for Bangla [8 page27-47]:

To encode Bangla words we need to consider context and also need to generate multiple codes for the same string. These constraints can be handled in Double metaphone algorithm, which we did for Bangla. It is termed as Double metaphone phonetic encoding. The table is given below for encoding.

Letter	Name	Unicode	Code	Context	Example
্	VIRAMA (Hasant)	\u09CD	*Not Coded*		
ো	SIGN O	\u09CB	*Not Coded*		
ঁ	CANDRABINDU	\u0981	*Not Coded*		
অ	A	\u0985	“o”		
ও	O	\u0993	“o”		
আ	AA	\u0986	“a”		
া	SIGN AA	\u09BE	“a”		
ই	I	\u0987	“i”		
ঈ	II	\u0988	“i”		
ি	SIGN I	\u09BF	“i”		
ী	SIGN II	\u09C0	“i”		
উ	U	\u0989	“u”		
ঊ	UU	\u098A	“u”		
ু	SIGN U	\u09C1	“u”		
ূ	SIGN UU	\u09C2	“u”		
এ	E	\u098F	“e”		
ে	SIGN E	\u09C7	“e”		
ঐ	AI	\u0990	“oi”		
ৈ	SIGN AI	\u09C8	“oi”		
ঔ	AU	\u0994	“ou”		
ৌ	SIGN AU	\u09CC	“ou”		
ক	KA	\u0995	“k”		
খ	KHA	\u0996	“k”		
ক্ষ		\u0995 \u09CD \u09B7	“k”	@ the beginning	ক্ষত /kh̪ɔto/

ক্ষ		\u 0995 \u09CD \u09B7	“kk”	@ middle/end	দক্ষ /d̪okkho/
গ	GA	\u0997	“g”		
ঘ	GHA	\u0998	“g”		
ঙ	NGA	\u0999	“ng”		বাঙলা /banla/
ং	ANUSVARA	\u0982	“ng”		বাংলা /banla/
চ	CA	\u099A	“c”		
ছ	CHA	\u099B	“c”		
য	YA as *phalaa*	x\u09CD\u09AF | x\u09CD\u09AF \u0986	“e”	@ the beginning as YA *phalaa*	ব্যক্ত /bækt̪o/
		...xy \u09CD z \u09CD \u 09AF	*Not Coded*	@ middle/end with conjuncts	সন্ধ্যা /ʃond̪ʱa/
		...xy \u09CD \u09AF	Doubles: yy	@ middle/end	অদ্য /od̪d̪o/
য	YA	\u09AF	“j”		
জ	JA	\u099C	“j”		
ঝ	JHA	\u099D	“j”		
ঞ	NYA	\u099E \u099A	“n”	Before CA	অঞ্চল /ɔncɔl/
		\u099E \u099B	“n”	Before CHA	বাঞ্ছা /bancha/
		\u099E \u099C	“n”	Before JA	অঞ্জলি /ɔnɟoli/
		\u099E \u099D	“n”	Before JHA	যন্ত্রণা /ɟʱɔnɟa/
		\u099A \u099E	“n”	After CA	যাচঞা /ɟacna/
		\u099E \u0985 | \u099E\u0987	*Not Coded*	Before A | I	মিঞা /miã/
		\u099C \u09CD \u099E	“ge”	@ the beginning after JA	জ্ঞাত /g̃æt̪ɔ/
		... \u099C \u09CD \u099E	“gg”	@ middle/end after JA	বিজ্ঞান /bigg̃æn/
		\u099E \u09CD	“n”	With hasant	নঞ /nɔn/
ট	TTA	\u099F	“T”		
ঠ	TTHA	\u09A0	“T”		
ড	DDA	\u09A1	“D”		
ঢ	DDHA	\u09A2	“D”		

র	RA as *phalaa*	x\u09CD \u09B0	“r”	@ the beginning	প্রকাশ /prokaʃ/
		…x\u09CD \u09B0	“r”	@ middle/end	রাত্রি /raṯṯri/ | রাত্রি /raṯri/
র	RA	\u09B0	“r”		
ড়	RRA	\u09DC	“r”		
ঢ়	DDHA	\u09A2	“r”		
ন	NA	\u09A8	“n”		
ণ	NNA	\u09A3	“n”		
ত	TA	\u09A4	“t”		
থ	THA	\u09A5	“t”		
দ	DA	\u09A6	“d”		
ধ	DHA	\u09A7	“d”		
প	PA	\u09AA	“p”		
ফ	PHA	\u09AB	“p”		
ব	BA as *phalaa*	x\u09CD \u09AC y…	*Not Coded*	@ the beginning	স্বদেশ /ʃɔdeʃ/
		…x\u09CD y \u09CD \u09AC	*Not Coded*	BA *phalaa* with conjuncts	তত্ত্ব /ṯɔṯṯo/
		… \u09AC \u09CD \u09AC	“bb”	After BA as conjuncts	তিব্বত /ṯibbɔṯ/
		… \u09AE \u09CD \u09AC	“mb”	After MA as conjuncts	লম্ব /lɔmbo/
		… \u0997 \u09CD \u09AC	“gb”	After GA as conjuncts	দিগ্বিদিক /ḏigbiḏik/
		\u0989 \u09A6 \u09CD \u09AC	“udb”	After Ud- (U DA BA…)	উদ্বেগ /uḏbeg/
		…y \u09CD \u09AC	Doubles: yy	@ middle/end	বিশ্ব /biʃʃo/
ব	BA	\u09AC	“b”		
ভ	BHA	\u09AD	“b”		
ম	MA as *phalaa*	x\u09CD \u09AE…	*Not Coded*	@ the beginning	স্মরণ /ʃɔroṇ/
		…x\u09CD y \u09CD	*Not Coded*	MA *phalaa* with	সূক্ষ্ম /ʃukkhõ/

		... \u0995 \u09CD \u09AE	“km”	After KA as conjuncts	রুক্মিনী /rukmini/
		... \u0997 \u09CD \u09AE	“gm”	After GA as conjuncts	যুগ্ম /ɟugmɔ/
		... \u0999 \u09CD \u09AE	“ngm”	After NGA as conjuncts	বাঙ্ময় /baŋmoi/
		... \u099F \u09CD \u09AE	“tm”	After TTA as conjuncts	কুট্মল /kutmol/
		... \u09A3 \u09CD \u09AE	“nm”	After NNA as conjuncts	মৃণ্ময় /mrinmɔẽ/
		... \u09A8 \u09CD \u09AE	“nm”	After NA as conjuncts	জন্ম /ɟɔnmo/
		... \u09AE \u09CD \u09AE	“mm”	After MA as conjuncts	সম্মান /ʃɔmman/
		... \u09B2 \u09CD \u09AE	“lm”	After LA as conjuncts	গুল্ম /gulmo/
		... \u09B6 \u09CD \u09AE	“sm”	@ middle/end with SHA	কাশ্মীর /kaʃmir/
		... \u09B7 \u09CD \u09AE	“sm”	@ middle/end	কুষ্মান্ড /kuʃmando/
ম	MA	\u09AE	“m”		
য়	YYA	\u09DF	“y”		
ল	LA	\u09B2	“l”		
শ	SHA	\u09B6	“s”		
স	SA	\u09B8	“s”		
ষ	SSA	\u09B7	“s”		
হ	HA	\u09B9 \u09CD \u098B	“ri”	HA with Vocalic R	হৃদয় /rɦidoẽ/
		\u09B9 \u09CD \u09B0	“r”	HA with R as *phalaa*	হ্রদ /rɔd/
		\u09B9 \u09CD \u09A8	“nn”	HA with NA	পূরাহ্ন /purbaɲno/

		\u09B9 \u09CD \u09A3	"nn"	HA with NNA	প্রহ্ন /prannɦo/
		\u09B9 \u09CD \u09AE	"mm"	HA with MA	ব্রহ্মা /brommɦa/
		\u09B9 \u09CD \u09AF	"jj"	HA with YA as *phalaa* @ *middle/end*	উহ্য /uɟɟɦo/
		\u09B9 \u09CD \u09B2...	"l"	HA with LA @ beginning	হ্লাদ /lɦad/
		... \u09B9 \u09CD \u09B2	"ll"	HA with LA @ middle/end	আহ্লাদ /allɦad/
		\u09B9 \u09CD \u09AC	"h" | "o"	HA with BA	আহ্বান /aovan/ | আহ্বান /aɦobɦan/
হ	HA	\u09B9	"h"	Otherwise	
ঃ	VISARGA	x\u0983 y...	Doubles: yy	@ the middle	দুঃসময় /duʃʃomoë/
		x\u0983	"h"	@ the end, strlen = 1 | 2	উঃ /uɦ/, বাঃ /baɦ/

5.1.4 Approximate string matching algorithm [3,25]

This method uses an approximate string-matching algorithm to check the closeness of dictionary words with the misspelled word. In suggestion it gives the words that are close to it.

The words `computer' and `commuter' are very similar, and a *change* of just one letter, p->m will change the first word into the second. The word `sport' can be changed into `sort' by the *deletion* of the `p', or equivalently, `sort' can be changed into `sport' by the *insertion* of `p'.

The *edit distance* of two strings, s1 and s2, is defined as the *minimum* number of *point mutations* required to change s1 into s2, where a point mutation is one of:

1. change a letter,
2. insert a letter or
3. delete a letter

```
EDITDISTANCE(s1, s2)
 1  int m[i, j] = 0
 2  for i ← 1 to |s1|
 3  do m[i, 0] = i
 4  for j ← 1 to |s2|
 5  do m[0, j] = j
 6  for i ← 1 to |s1|
 7  do for j ← 1 to |s2|
 8     do m[i, j] = min{m[i − 1, j − 1] + if (s1[i] = s2[j]) then 0 else 1fi,
 9                      m[i − 1, j] + 1,
10                      m[i, j − 1] + 1}
11  return m[|s1|, |s2|]
```

Example

E("Bannp","Bank")=2

Ban**n**p
Banp [delete n]
Ban**k [replace p by k]**

5.2 Implementation of service:

A user can use any one of the implemented four types of dictionary. From the client part user sends a word as input. From the general overview the functions of the web service is to load the corresponding XML file and send the information to the client.

For XML parsing DOM has been used in this project.

5.2.1 DOM (Document Object Model) [22,23,24]

The DOM is a platform-independent, programming- language-neutral application programming interface (API) for HTML and XML documents.

Its core outlines a family of types that represent all the objects that make up an XML document: elements, attributes, entity references, comments, textual data and processing instructions. With that, it defines the logical structure of documents and the way a document is accessed and manipulated. (DOM specifies how XML documents are represented as objects, so that they may be used in object-oriented programs.) Increasingly, XML is being used as a way of representing many

different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data to allow programs to access and modify the content and the structure of XML documents from within applications. Anything found in an XML document can be accessed, changed, deleted, or added using the DOM, except for the XML internal and external subsets for which DOM interfaces have not yet been provided. After an XML document has been parsed into a collection of objects conforming to DOM, the object model can be manipulated in any way that makes sense. This mechanism is also known as the "random access" protocol, as any part of the data can be visited at any time. The DOM usually resides in memory (it is the output of an XML parser), but it can also be stored on disk (to save on the time needed to parse the XML repeatedly) as a Persistent DOM (PDOM). When an XML document is large and not likely to change much, as is the case for dictionaries, using its PDOM representation can significantly speed up XQL querying.

5.2.2 Searching the meaning of a word:

public XmlNode[] word_search(string word, string dType)

The above method works for word searching. Among the two parameters first one takes user input and second one indicates Dictionary types. I.e. English to English, English to Bangla, Bangla to English or Bangla to Bangla. Based on the dictionary type the corresponding xml is loaded into a DOM object. From there the total information for the inputted word is taken to a XmlNode array. The array is then returned to the client.

For Example, if a user search for the Bangla meaning of the word “capital” then English to Bangla Xml will be taken in the DOM object. Then the information will returned to client application.

5.2.3 Searching words which starts with/ends with *:

If user wants to find the word list which start with “**con”** he will get condition, conflict, control so on.

public XmlNode[] wordsStartWith(string wsw, string dType)

The above method is used to implement this feature. “Wsw” is the inputted string and dType represent the dictionary type. Then from the document object the words

are find which starts with the user inputted string. Then the words are sending to the client application.

To find the words which ends with a sub word then the following method are used which works like the previous method.

public XmlNode[] wordsEndWith(string wew, string dType)

5. 2.4 Spelling Check and making suggestion list:

If user looks for a word which is not in the dictionary database or misspell the word then spell checking is required. After getting the inputted word firstly it will be searched in the xml document. If the word is not found then a suggestion list will be generated which are alphabetically around the inputted word.

To generate the suggestion list firstly, the inputted word will be encoded using Double Metaphone Algorithm. It will generate a key. After that we find the keys for all the words in the dictionary. The keys which match with the input string key they will be in suggestion list.

For example, if the user looks for the meaning of the misspelled word "**supress** " then at first we generate sound of the word using **[5]** which gives **SPRS**. Then I find keys for all the words of the dictionary database and take the words with same sound **SPRS** in the suggestion list.

Surprisingly
Surprising
Spares
Suppressed
Suppress
Surprises
Surprised
Surprise

The following method is used to generate suggestion list.

public String[] MakingSuggestionList(string wrongWord, string dType)

In the above function 1st parameter takes the wrong word as input and the 2nd parameter is dictionary type. Then it calls another function

```
public DoubleMetaphone(String word)
```

This method is used to generate sound of the inputted word.

If the user uses Bangla dictionary, the Bangla wrong word will be encoded to English using the Double Metaphone Bangla encoding transformations. Similarly like English, all the other dictionary words are encoded to English. The encoded words which match are taken to the suggestion list.

```
public String BMEncode(String word)
```

This method is used for Bangla metaphone encoding. It takes a Bangla word as parameter and returns the encoded English key.

5.2.5 Ranking the words in the suggestion list:

Now in the suggestion list if the user gets the expected word in the middle or at the end then user might be angry. So to make the better suggestion list it is necessary to raking the words. For that edit distance is used.

```
Public int EditDistance (string source, string target)
```

This method will take the target string as the user inputted misspelled words and the other words in the primary suggestion list as source string. Then distance will be find and stored for all the primary suggestion list words. Then their distances will be sorted using bubble sort as ascending order. Now the words will be re arranged. Now it will give a better suggestion list.

For example:
X = supress

Words	**Edit Distance**
E (Surprisingly, X)	7
E (Surprising, X)	5
E (Spares, X)	3
E (Suppressed, X)	3
E (Suppress, X)	1
E (Surprises, X)	3
E (Surprised, X)	4
E (Surprise, X)	3

Better suggestion list:

Suppress	1
Spares	3
Suppressed	3
Surprises	3
Surprise	3
Surprised	4
Surprising	5
Surprisingly	7

Similarly, for ranking the suggestion list of Bangla words, I applied
The Edit Distance method for the encoded keys of the Bangla words. I took to ideas of making the better suggestion list from **[8].**

5.3 Implementation of client:

The client application contains the implementation of the parsing data.
When the web service returns a nodelist then in the client application the nodelist is parsed using Document Object Modeling (DOM) parser. After that it just displays the contents of nodelist in the web page. If the user used Bangla to Bangla dictionary then it display the contents in the web page used for Bangla dictionary.

```
wordNodes = dic.word_search(searchWord,"E2E");
```

The above method is used for sending request to the web service .The function parameters are inputted dictionary word and 2nd one is dictionary type. The returned result is stored in nodeList wordNodes. The wordNode is then parsed.

Chapter 6: Result and Discussion

The end users don't have headache how the software implemented. They just want a user friendly and well organized software. I tried to make the dictionary as much informative and well organized as possible.

The user will first see the above webpage. From here they will click any one type of dictionary link. This will dispatch that dictionary page.

Figure: The home page of the dictionary

Say the user clicked "English to English". He/She will see the following web page. In that web page the users will type an English word in the text box and press the search button. Then they will get the meaning of the inputted word.
In the display I separated the meanings of the dictionary words as parts of speech. A word can be used as different parts of speech. Under each parts of speech there may have multiple meanings. For each meaning I displayed uses in sentence, antonyms, and synonyms. The antonyms, synonyms are displayed as link. If users click to that link then the meaning of that link will be displayed.

BRAC University Online Dictionary

ENGLISH TO ENGLISH

Home

Look up a word: capital [Search]

- (•) Exact Word Search
- ( ) Words Start With * [Example: " con " -->Condition , Conflict , Convertion]
- ( ) Words End With * [Example : gry ----> Angry, Hungry]
- ( ) Search Words Alphabatically Arroud *

1. capital

Browse the words alphabetically arround [capital]

Pronountiation: kae pih tEl

Part of Speech		noun
Meaning	**1.**	the city where the central government of a nation or state is located.
Use in Sentence		Dhaka is the capital of Bangladesh
Synonyms		Chief city;
Meaning	**2.**	an upper-case letter.
Use in Sentence		Please change these lower-case letters into capitals.
Synonyms		majuscule; capital letter;
Antonyms		small letter;
Meaning	**3.**	money or other wealth owned or used by a business.
Use in Sentence		I have no capitaL
Synonyms		cash; fund;

Part of Speech		adjective
Meaning	**1.**	relating to financial capital.
Use in Sentence		the capital markets.
Synonyms		financial;
Meaning	**2.**	being of high quality; first rate.
Use in Sentence		This is a capital idea.
Synonyms		fine; super;
Antonyms		third-rate; minor;
Meaning	**3.**	referring to the death penalty.
Use in Sentence		He got capital punishment.
Synonyms		deadly ; serious;

Figure: Display the meaning in English for a English word.

Now If the user misspells the inputted word then it will show the error message like bellow and give the suggestion list. Here you see for the mistake of the spelling "**spelling**" it gives five words as suggestion list. Theses words are also displayed as link. If anyone is clicked then the meaning will be displayed.

Figure: output for misspelled word(English to English)

From the above page if the user click in the "home" link it will returned to the main page of the dictionary. Similarly, like English to English dictionary if they click "Bangla to Bangla" they will get page for Bangla to Bangla dictionary [figure:]. The access of this page is like English to English dictionary. The differences are the input will be in Bangla and the display result will also be in Bangla.

Figure: word meaning from Bangla to Bangla dictionary.

figure: suggestion list for wrong Bangla search.

The other dictionaries like English to Bangla and Bangla to English are same.

Chapter 7: Conclusion

In the above chapters it has been shown that why my dictionary is language independent, where I am unique. It has been discussed how the web service and client has designed and implement. It is also shown the techniques of spell checking for Bangla and English languages.

7.1: After implementing the dictionary I found few problems:

(7.1.1)The dictionary access is slow because it just use flat XML file as database.

(7.1.2)While spell checking it generates metaphone key for all words entry which is also a slow process.

(7.1.3)My web service is not language independent because it has implemented using C#.NET.

(7.1.4)Right now the dictionary is not useable because of in sufficient data in the XML.

7.2: Future improvements

My future works on it based on the above problems.
(7.2.1) I will introduce DBMS to handle data so that it becomes faster.
(7.2.2) Then I will try to improve spell checking technique to make it better.
(7.2.3) I will start inputting data to make my dictionary useable.
(7.2.4)I will implement the dictionary service using JAVA to make my dictionary completely programming language independent.

References:

[1]. http://dotnetjunkies.com/Article/C333976F-1EA4-42C1-BFEE-63F3C09CD94B.dcik

[2]. http://nlp.stanford.edu/kirrkirr/ausweb99/paper.html

[3]. http://nlp.stanford.edu/IR-book/html/htmledition/edit- distance1.html

[4] Lawrence Philip's Metaphone Algorithm, available online at http://aspell.sourceforge.net/metaphone/index.html

[5].Lawrence Phillips, "The Double Metaphone Search Algorithm", C/C++ Users Journal, 18(6), June 2000, available online at http://www.cju.com/document/s=8038/cuj0006phillips/

[6].Kevin Jansz,Student of the University of Sydney, Australia - thesis work on "Intelligent processing, storage and visualization of dictionary information".

[7].Sng Wee Jim, student of the NANYAND Technology University- thesis work on "Optimizing KIRRKIRR: An Electronic Dictionary Browser".

[8]. Naushad UzZaman, student of BRAC University, undergraduate thesis on "PHONETIC ENCODING FOR BANGLA AND ITS APPLICATION TO SPELLING CHECKER, TRANSLITERATION, CROSS LANGUAGE INFORMATION RETRIEVAL AND NAME SEARCHING".

[9] Bangla Academy Bengali-English Dictionary
[10]Bangla Academy Bangla to Bangla Dictionary (Baboharik Bangla Obidhan)
[11] Bangla Academy English to Bangla Dictionary
[12] www.**wordsmyth**.net/
[13]http://www.w3schools.com/webservices/default.asp
[14]http://www.w3.org/TR/2002/WD-ws-arch-20021114/
[15] http://www.altova.com/whitepapers/webservices.pdf.
[16] www.**xml**.com/
[17]www.w3schools.com/**xml**/default.asp
[18] www.w3.org/**XML**/
[19]www.**stylusstudio**.com/docs/v2006/d_xslt.html
[20] http://nlp.stanford.edu/kirrkirr/dictionaries/dictionaryinfo.html
[21] http://en.wikipedia.org/wiki/Internet_Information_Services
[22] www.w3schools.com/**dom**/default.asp
[23] www.dot**net**spider.com/kb/Article1100.aspx
[24] www.xmlfiles.com/
[25] www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/**Edit**/