Auto-Translation Instant Messenger

---

using XMPP/Jabber Protocol

By

Mohd Syafiq Fauzan bin Ariffin

**Dissertation submitted in partial fulfillment
of the requirement for the**

**Bachelor of Technology (Hons)
Information & Communication Technology**

September 2011

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR
31750 TRONOH
PERAK DARUL RIDZUAN

CERTIFICATION OF APPROVAL

Auto-Translation Instant Messenger using XMPP/Jabber Protocol

by

Mohd Syafiq Fauzan bin Ariffin

A project dissertation submitted to the
Information & Communication Technology Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Technology (Hons)
Information & Communication Technology

Approved by:

Mr. Mohd Hilmi Bin Hasan
Project Supervisor

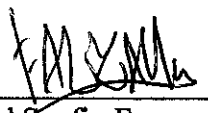
September 2011

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Mohd Syafiq Fauzan bin Ariffin

TABLE OF CONTENTS

ABSTRACT **1**
CHAPTER 1: INTRODUCTION **2**
1.1 Background 2
1.2 Problem Statement 2
1.3 Project Significance 3
1.4 Objectives 3
1.5 Scope of Study 4
CHAPTER 2: LITERATURE REVIEW **5**
2.1 Introduction 5
2.2 Existing Technologies 5
2.2.1 Instant Messaging 5
2.2.2 Jabber/XMPP 7
2.2.3 Machine Translation 10
2.3 Related Works 15
2.3.1 IM using XMPP Protocol 15
2.3.2 Auto Translation IM 17
CHAPTER 3: METHODOLOGY **22**
3.1 Modified Waterfall Development Model 22
3.2 Project Activities and Key Milestones 23
3.3 Gantt Chart 26
3.4 Tools Required 27
3.4.1 Hardware 27
3.4.2 Software 27
CHAPTER 4: ANALYSIS & DISCUSSIONS **28**
4.1 Questionnaire 28
4.2 Result Analysis 37
4.3 Discussion 38
CHAPTER 5: CONCLUSION & RECOMMENDATIONS **49**
REFERENCES **50**

LIST OF FIGURES

No.	Title	Page
2.1	Most preferred engine and margin of preference compared to second-best engine.	12
2.2	The interface of Meglobe as in beta release.	18
2.3	The official MeGlobe website	19
2.4	Statistic graph of unique visitors for MeGlobe	19
2.5	User chat with English-to-Chinese bot (en2zh) using GTalk	21
3.1	Modified Waterfall Development Model	22
4.1	Respondents by Nationality Column Chart	28
4.2	Respondents by Race Column Chart	29
4.3	Question 1 Responses Pie Chart	30
4.4	Question 2 Responses Pie Chart	31
4.5	Question 3 Response Column Chart	32
4.6	Question 4 Response Pie Chart	33
4.7	Question 5 Response Column Chart	34
4.8	Question 6 Response Pie Chart	35
4.9	Question 7 Response Column Chart	36
4.10	Question 8 Response Pie Chart	37
4.11	Client Side System Architecture Flowchart	40
4.12	Server Side System Architecture Flowchart	41
4.13	Prototype system architecture for concept proving purpose	42
4.14	Sample run of the automatic translation	45
4.16	Sender language selection	46
4.17	Translation Service Selection	47
4.18	Receiver Language Selection	47

ACKNOWLEDGEMENT

There were many people involved throughout completing this Final Year Project, which has contributed to give help, guidance, assistance, motivate, advice, supports, and also supervise. It is a good experience going through all the documentation, presentation and development of a project entitled Auto-Translation Instant Messenger using XMPP/Jabber Protocol. I would like to express my appreciation and highest gratitude to the individual that have taken the time and effort to assist me in completing the project. Special thanks go to my supervisor, Mr. Mohd Hilmi bin Hasan. Without his guidance, assistance and motivation, I would face a lot of obstacle. I would like to express my appreciation to all my friends who supported me directly and indirectly in completing this project. Without the cooperation and motivation from these individuals, no doubt I would have to face some problem throughout the course.

ABSTRACT

Effective communication is a vital component in decision making process. However, the language barrier established from the differences in culture and origin can interrupt the process of coming to an understanding. Various translation methods have been used to break this barrier. The traditional ways of using human translator or the usage of lingua franca imposed some problem and limitations.

Auto-Translation Instant Messenger is an IM program that aimed to provide instant translation to users when they communicate with people of different speaking language. It is developed under the XMPP protocol that provides standards and flexibilities at the same time.

In the process of completing this project, Modified Waterfall methodology was chosen as guidance in the development of the working program. Important project activities and milestones are explained. As part of analysis process, a set of questionnaire have been distributed and its result will act as guidance in designing and developing the program. This project is aimed to learn in deep about XMPP, machine translation and instant messaging.

CHAPTER 1

INTRODUCTION

1.1 Background

Auto-Translation Instant Messenger is an instant messaging program developed using XMPP protocol that provides real time translation retrieved from Google Translate engine to ease the communication between two parties that doesn't speak similar language. This project combines the usage of XMPP protocol and Google Translate API version 2.

1.2 Problem Statement

It is crucial that the parties that communicate to understand each other to reach to a solid agreement. However, it is difficult to reach to that understanding if the parties speak different language resulting in language barrier between the communicating parties. Language barrier is defined as a figurative phrase used primarily to indicate the difficulties faced when people, who have no language in common, attempt to communicate with each other.

To break this barrier, lots of methods have been implemented. One of the methods is using human as translator or middle man. Unfortunately, human only talks several or limited languages and the cost to hire human translator is high. The issue of confidentiality is also one of the concerns while using human translator especially in business world.

Besides that, lingua-franca is also said to be a method of breaking the language barrier. People understand English, but often cannot think in English: serious barriers to intercultural collaboration exist, because the collaboration often requires elaborating new ideas in English.

Thus, a cheap yet effective alternative is needed to overcome this disadvantages and more importantly, breaking the language barrier.

1.3 Project Significance

Taking into consideration the need of efficient communication, this project addresses communication ineffectiveness that occurs due to language barrier between the communicating parties. This project also aimed to provide an alternative to human translator and adding the variety of machine translation application available.

Machine translator is being used widely as substitute for human translator as early as 1940s. Since then, many parties have join forces to develop a better, if not perfect machine translator. The rapid development of machine translator has motivated software developer to come out with variety program aimed to break the language barrier. This project focus is the implementation of real-time translation in instant messaging in breaking the language barrier as instant messaging is a widely used as a communication medium in this multimedia technology era. This project is taking advantage of the development of machine translation that has become better and more open as the medium to translate the chat text.

Using this program, user can communicate with anyone, in the comfort of speaking (typing) in their own language, comfortably knowing that the other parties are reading it in their understood language. This way, a better communication can be achieved.

1.4 Objectives

The objective of this project is:

i. To study the XMPP protocol in Instant Messaging development.
ii. To study how machine translation (Google Translate) service works and how to link the service in instant messenger program using XMPP protocol.
iii. To design and develop a XMPP based IM that support instant translation retrieved from Google Translate.

1.5 Scope of Study

This project is targeting instant messaging users; corporates as well as individuals that intend to communicate with peoples of different languages that supported by machine translator engine.

The program will be designed to support users that are using Windows operating system and expected to expand to Linux-based and MacOS in future.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

As documented by Calefato *et al* (2010), most of the projects or works takes over long distance, and communication will often involve distant cultures with different languages and communication styles resulting in communication problem.

This project aimed to overcome such problem. The program, Auto-Translation IM is to be used as a means to communicate for parties that doesn't speak same language as it provides real time translation using Google Translate translation technology.

In order to develop this program, it is important to understand the current technology involved, namely understanding on Instant Messaging, XMPP/Jabber, and Machine Translation technology. Apart from that, it is also best to look at some of the related works like existing XMPP-based IM, and existing auto translation IM.

2.2 Existing Technologies

2.2.1 Instant Messaging (IM)

Instant messaging (IM) and Internet chat communication have seen enormous growth over the last several years. IM is the private network communication between two users, whereas a chat session is the network communication between two or more users (Jennings *et al* 2006). Sessions can either be private, where each user is invited to join the session, or public, where anyone can join the session. There are on the order of 100 million Internet IM users, where a user is defined as a unique name on one of the major public IM networks — Google Talk (GTalk) AOL Instant Messenger (AIM), Microsoft Messenger (MSN), or Yahoo! Messenger (YMSG).

As mentioned by Shigeoka (2002), the idea of IM has been around for a long time. All of the visible IM features like one-on-one chat and group chats existed in other Internet applications long before IM entered the scene. For example, the classic

UNIX talk application allowed users to chat over the network years before IM ever appeared, and group chats have been carried out on Internet Relay Chat (IRC) systems almost as long as talk has been around.

IM systems using the Java programming language are poised to become a major part of both consumer and enterprise networking, and will play a core communication role similar to email. Messaging of course has always been a core feature of the Internet. For example, one of the first and still most pervasive Internet technologies is email. It remains an Internet killer app. However, it is said that Internet communication can be even more interesting and powerful than "plain old email." People should be able to better exploit it as an inexpensive medium for transferring data almost instantaneously.

Translation technology has become customizable and easier to use via GUIs and pull-down menus. The technology is also moving in other new directions in response to computer-industry trends. Companies are developing more sophisticated translation products that are increasingly useful. As suggested by Paulson (2001), some technologies in the future should be able to translate user input whether e-mail, chat, or even speech in near real time. The usage of IM as the medium to breaking the language barrier in this project is backed by the facts and figures of IM market research published in December 2009 by Radicati Group (2009) that stated:

- there are 47 billion worldwide instant messages per day in 2009
- there are 2.1 billion IM accounts worldwide in 2009
- there are 1 billion IM users worldwide in 2009
- there are 53 IM messages per user per day in 2009
- it is predicted that 2 billion of IM user growth per year

2.2.2 Jabber/Extensible Messaging and Presence Protocol (XMPP)

The Jabber or Extensible Messaging and Presence Protocol (XMPP) (used interchangeably) as explained by Smith *et al* (2009) are an open technology for real-time communication, using the Extensible Markup Language (XML) as the base format for exchanging information. In essence, XMPP provides a way to send small pieces of XML from one entity to another in close to real time. Ozturk (2010) wrote, XMPP is an open, XML-based protocol aimed at near-real-time, extensible (IM) and presence information. It has been expanded into the broader realm of message-oriented middleware. Built to be extensible, the protocol has been extended with features such as Voice over IP (VoIP) and file transfer signaling. XMPP protocol has been used by many social networking platforms including Gtalk, and Facebook; collaborative services like Google wave, and gradient; geo-presence systems like Nokia Ovi Contacts; multiplayer games like Chesspark, and by many online live customer support and technical support services.

Saint-Andre (2005) stated the fact that XMPP is an open wire protocol standardized by the Engineering Task Force (IETF), rather than a single open-source codebase, encourages multiple implementations and licensing schemes because protocols are not viral in the sense that open-source licenses such as the GNU general public license (GPL) are. So far, there are open-source server implementations in C, Java, Python, and Erlang, as well as closed-source implementations produced by software vendors such as Jabber Inc., Antepo, Coversant, and Sun Microsystems. There are open-source, freeware, shareware, and commercial clients for common (Windows, MacOS, Linux) and not-so-common (Amiga) personal computing operating systems, handheld devices running PalmOS and Windows CE, and cellphone platforms such as Symbian and Java 2 Micro Edition (J2ME).

Jabber client can be code in various machine languages; there are Jabber code libraries for C, C++, C#, COM, Delphi, Erlang, Flash, Java, JavaScript, Mono, Objective-C, Perl, PHP, Python, Ruby, Tcl, and more. Most server implementations are quite modular, so it's relatively easy to write server-side components for custom functionality.

According to Saint-Andre (2005), the first, and still dominant, application of XMPP is IM and presence. These implementations are often called "the Linux of instant

messaging" because they provide a fully open alternative to closed, proprietary IM services. Since the first Jabber code's release in January 1999, millions of end users have downloaded one of the many Jabber clients; hundreds of thousands of system administrators have installed Jabber servers, and thousands of developer's have contributed code to the various open-source projects or written custom Jabber extensions for commercial or internal use.

Jabber is a compelling IM solution that is well-suited to meet today's and tomorrow's IM needs. Jabber is not a particular piece of software. Instead, it is an open, freely available set of protocols for building IM systems (Shigeoka 2002). Existing messaging systems can implement the Jabber protocols to add IM to their list of features. Alternatively, new systems are being built from the ground up to support the Jabber protocols and prepare for the rapidly expanding responsibilities being assigned to IM systems.

XMPP has been competing with SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) to be the dominant presence protocol. Based on the study done by Ozturk (2010), SIMPLE is a set of extensions to the established SIP protocol (Session Initiation Protocol) that initiate, set up, and manage a range of media sessions, including voice and video. SIMPLE extensions define SIP signaling methods to handle the transport of data and presence. Pioneers of XMPP argue that an XML-based data-transport technology is better suited than a signaling technology to handle IM and presence. According to its designers, one major benefit of XMPP is that it can be extended across disparate applications and systems because of its XML base. XMPP has been gaining ground especially in the social networking and IM applications domain.

IM in XMPP relies heavily on the use of the <message> and <presence> stanzas. A <message> stanza is sent whenever a user communicates with another. Shigeoka (2002) suggest that, Jabber messaging model is composed of four main elements: XML packets containing marked-up data, XML streams used to transport XML packets, and Jabber clients and servers that exchange XML packets over an XML stream. XMPP technologies use decentralized client-server architecture with a direct federation model. Every XMPP entity needs an address; the addresses can be

associated not only with people but also devices, terminals, routers, and any other network-aware entity (Saint-Andre, 2005).

Implementation of Jabber in this project as a platform to develop the program is backed by several key advantages over other messaging protocol as according to Kozakai (2004):

- Open – the Jabber protocols are free, open, public, and easily understandable; in addition, multiple implementations exist for clients, servers, components, and code libraries.
- Standard – the Internet IETF has formalized the core XML streaming protocols as an approved instant messaging and presence technology under the name of XMPP, and the XMPP specifications are moving forward rapidly within the IETF's standards process.
- Proven – the first Jabber technologies were developed by Jeremie Miller in 1998 and are now quite stable; hundreds of developers are working on Jabber technologies; there are tens of thousands of Jabber servers running on the Internet today, and millions of people use Jabber for IM. In fact, it has been estimated that the number of Jabber IM users has recently surpassed the number of ICQ users.
- Decentralized – the architecture of the Jabber network is similar to email; as a result, anyone can run their own Jabber server, enabling individuals and organizations to take control of their IM experience.
- Secure – any Jabber server may be isolated from the public Jabber network (e.g., on a company intranet), and robust security using SASL (Simple Authentication and Security Layer protocol) and TLS (Transport Layer Security protocol) has been built into the core XMPP specifications.
- Extensible – using the power of XML namespaces, anyone can build custom functionality on top of the core protocols; to maintain interoperability, common extensions are managed by the Jabber Software Foundation (JSF).

- Flexible – Jabber applications beyond IM include network management, content syndication, collaboration tools, file sharing, gaming, and remote systems monitoring.
- Diverse – a wide range of companies and open-source projects use the Jabber protocols to build and deploy real-time applications and services.

2.2.3 Machine Translation (MT)

As mentioned by Ishida (2006), language barriers remain the biggest barrier to intercultural collaboration. This problem is more serious in Asia than Europe. Asian people are not taught neighboring languages. Japanese people do not understand Chinese or Korean and vice versa. Chin-Yew (1999) stated that with the increasing amount of online information and the rapid growth of the number of non-English speaking Internet hosts, it is becoming increasingly important to offer users universal access to valuable information resources in difference languages. One of many ways to break the language barrier is by using Machine Translation (MT) as a medium to translate as substitute for human translator.

MT are considered to date from 1947, when Warren Weaver, whose experience in code-breaking during World War II led him to presume that MT would be a fairly simple affair, convinced the American authorities to invest heavily in MT.

Even though MT have made rapid progress over the last decade, it is still sometimes considered as problematic, but many of the problems MT finds difficult to solve are similar to those experienced by human translators as suggested by McGinity (2003). MT is important for a variety of reasons. Human translation is expensive, takes time and is usually unavailable when it is needed for communicating quickly and cheaply with people with whom we do not share a common language. There are also the obvious political reasons deriving from the ideal of a multi-lingual, multi-cultural society, an ideal which, in its turn, results in its commercial importance.

For this project, the translation of chat text will be done using Google Translate. Google Translate is a free statistical machine translation service provided by Google Inc. to translate a section of text, document or webpage, into another language. The

service was introduced in April 28, 2006 for the Arabic language (Franz, 2006). Google Translate used Systran's engine until 2007 (Schwartz, 2007) when it developed its own proprietary statistical translation engine.

Google Translate gained huge technical advantages over their main competitor with the usage of own proprietary statistical translation engine. According to Tcworld E-Magazine author, Shen (2010), the main benefit of the statistical approach is that rule-based translation systems require the manual development of linguistic rules, which is costly and does not carry over to other languages. Statistical-based systems are not tailored to any specific pair of languages; they simply need big bodies of parallel text to train from. This is the reason why Google has over 50 languages that can be paired by users and Babelfish has only 14 languages with fixed pairing, even though it has been in operation significantly longer.

An experiment conducted by Shen (2010) in finding out the best online translation tools based on hypothesis that the relative performance of various translation engines will change depending on the language to be translated and the character length of the requested translation. For example, Engine X may be consistently more effective than Engine Y for English-Spanish translations under 50 characters, but the opposite will be true for translations over 150 characters. The hypothesis is tested by directly compared the quality of outputs from Google Translate (a statistical translation engine), Yahoo Babelfish (a traditional rule-based translation engine) and Microsoft Bing Translator (a hybrid statistical engine with language specific rules). Volunteers also invited to enter text of their choice into our survey form, which routed user requests to each of the three translation engines via their server-side Application Programming Interfaces (API's). These API connections allowed user to return the results of all three engines and allowed the user to vote on the engine which produced the best and worst results. The experiment runs for 6 months period involving professional translators and interpreters as well as non-professional multilingual users to participate in the experiment. **FIGURE2.1** display the result of the experiment is generated by analyzing the distribution of the "best" and "worst" votes according to the following parameters:

i. The input and output languages
ii. The length of the text given in characters

iii. Single sentence or phrase vs. multiple sentence paragraphs

iv. Presence or absence of a question mark

	Under 2000 Characters (Full Documents)		Under 500 Characters (Single Paragraphs)		Under 150 Characters (SMS, Twitter post)		Under 50 Characters (Basic Phrases)	
English to Chinese	Google	23%	Google	19%	Google	8%	Google	0%
Chinese to English	Google	22%	Google	0%	Babelfish	20%	Bing	17%
English to Spanish	Google	50%	Google	50%	Google	50%	Google	28%
Spanish to English	Google	36%	Google	28%	Bing	0%	Google	25%
English to French	Google	40%	Google	36%	Google	25%	Google	7%
French to English	Google	76%	Google	76%	Google	73%	Google	83%
English to German	Google	3%	Google	0%	Bing	7%	Bing	0%
German to English	Google	56%	Google	52%	Google	45%	Google	60%
English to Italian	Google	10%	Bing	44%	Bing	63%	Bing	33%
Italian to English	Google	33%	Google	25%	Google	17%	Bing	33%
English to Arabic	Google	44%	Google	45%	Google	55%	Google	75%
English to Japanese	Google	35%	Google	33%	Google	45%	Google	33%
English to Korean	Babelfish	20%	Babelfish	0%	Bing	50%	Bing	33%
English to Portuguese	Google	11%	Bing	29%	Google	40%	Google	25%
English to Russian	Google	52%	Google	47%	Google	57%	Google	25%

FIGURE 2.1 Most preferred engine and margin of preference compared to second-best engine.

The result describes the relationship between user preferences and translated text character length for 15 single direction language pairings. The most preferred engine is given at each intersection (Google, Babelfish, or Bing) along with the magnitude of its lead over its closest competitor in that category (colored percentage). The language pairings excluded from this table represent sets for which preferences was overwhelming (over 100%) or insufficient data was available. From the result, Shen (2010) conclude:

1. For long passages of text up to 2000 characters, survey takers generally prefer Google Translate's results across the board.

a. The extent of Google's lead varies dramatically from language to language. In some languages such as French, the strength of Google Translate's engine is overwhelming. However, in several others like German, Italian, and Portuguese, Google holds only a very slim lead when compared to its closest competitor.
b. These observations validate our Hypothesis 1 that no single engine can perform equally well across a spectrum of languages or conditions.

2. The greatest relative strength of a statistical translation focused engine (Google Translate) has not clustered around the European Union working languages as expected. German, Italian, and Portuguese, all EU working languages are the most hotly contested from a performance perspective.
    a. One possible explanation is that large additional bodies of parallel English-French text are available from the government of Canada for which official documents are translated into both. To a lesser extent this could explain the strength of Google Translate in Spanish as many Latin American countries offer English translations of official documents.

3. Traditional rule-based translation engines (Babelfish) performed generally well in East Asian languages such as Chinese and Korean.
    a. One possible reason for this performance could be that the language specific grammar and word usage rules are more effective than association-based transliteration in these situations.

4. Across almost every language Bing Translator and Yahoo Babelfish gain ground or surpass Google Translate as the text length gets shorter.
    a. In Chinese, the gradual erosion of Google's relative performance as total text length shrinks from 2000 characters to 50 characters is stark. Respectively, as phrases get shorter and more straightforward, rule-based or hybrid translation engines perform better.
    b. Though data is not shown, a similar effect is seen for passages that are only one sentence compared to passages with multiple sentences.

5. The most interesting observation is, that translation quality is not a two way street. The engine that is best for translating in one direction is not necessarily the best tool to translate back the other way.
    a. The two most obvious cases of this are French and German. Though Google Translation dominates when translating both of these languages to English, it faces heavy competition when translating from English to the foreign language.

The experiment result suggest that Google Translate stands higher than other competitor due to its translation quality and provide better translation for longer text.

Another reason for choosing Google Translate is simply because it is the easiest to integrate with programs as most of the API codes is made available in Google Code under Google Language API Family site and free for everyone to develop. One of the features offered in Google Code is implement Google Translate to translate websites or application into one or more different languages. This feature is not available for other translation services.

2.3 Related Works

2.3.1 IM Using XMPP Protocol

According to the XMPP Standards Foundation (XSF), on their official website, there are currently 23 XMPP servers to date providing basic messaging, presence, and XML routing features that can be used by anybody to run their own XMPP service, either over the internet or on local area network. Also listed are 80 XMPP clients that enables user to connect to an XMPP for IM. All of listed servers and clients are free and some of them are even listed as open source and the code is listed under libraries in XSF website. There are 55 libraries entries available for many different languages, thus enabling developers to build a wide variety of XMM-enabled applications.

The fact that XMPP is an open protocol for presence, instant messaging, and real-time communication resulting in hundreds of application developed based on XMPP protocol. One of the most used XMPP client program is GTalk and Pidgin as it is listed as the fourth and fifth most popular IM clients in IM figures and facts released in early 2010.

2.3.1.1 GTalk

As cited from Google website, Gtalk is a freeware Microsoft Windows (XP, Server 2003, Vista and Windows 7) instant messaging and voice over internet protocol (VoIP) client application offered by Google Inc. Instant messaging between the GTalk servers and its clients uses an open protocol, XMPP, allowing users of other XMPP/Jabber clients to communicate with Google Talk users.

Google has always been supporting open technology; most of the codes in developing GTalk are available in Google Talk for Developer page. There is a section described about Gtalk XMPP extensions, which describes the non-standard XMPP extensions used by the Google Talk server. It is open for any developer that building XMPP application to use the extension for their program. Some of the extensions may become proposed XEP extensions in the future, but it is considered to be Google-specific for now. They are documented so that developer can design a client that can take advantage of specific Google Talk features.

It also specified in Google Code website that extensibility is one of the greatest strengths of XMPP, the IETF standard protocol on which GTalk is built. While XMPP itself defines a bare set of features, the protocol encourages third parties to develop their own extensions. During the development of GTalk, Google found it useful to define extensions to implement features not already found in XMPP or any of its currently defined extensions. The protocols defined in the website are currently used by the Google Talk clients and servers. However, it is important to note that all of the extensions are not currently part of a proposed standardized extension, and therefore may change as Google work to standardize these features.

2.3.1.2 Pidgin

Taken from the official Pidgin website, Pidgin is a chat program which lets user to log in to accounts on multiple chat networks simultaneously. This means that user can be chatting with friends on MSN, talking to a friend on Google Talk, and sitting in a Yahoo chat room all at the same time. Pidgin runs on Windows, Linux, and other UNIX operating systems. Pidgin supports many features of these chat-networks, such as file transfers, away messages, buddy icons, custom smilies, and typing notifications. Numerous plugins also extend Pidgin's functionality above and beyond the standard features. Pidgin is free and contains no ads. All of the code is open source and licensed under the GNU General Public License. This means developer can get Pidgin's underlying code and modify it to suit their needs, as long as the changes made is published for everyone to benefit from as well.

Given that Pidgin is an open source program, lots of other IM program developed and some research is based on Pidgin codes. Pidgin code is used to develop IM program called HoneyIM to study on 'Fast Detection and Suppression of Instant Messaging Malware in Enterprise-like Networks' (Mengju et al, 2007).

2.3.2 Auto Translation IM

2.3.2.1 MeGlobe

As mentioned in Free Language, MeGlobe is a web-based IM client that provides real-time translation into more than 14 languages. Meglobe is an IM develop using Jabber technology. As per their released of stable version in 2009, MeGlobe support translation from and to Arabic, Dutch, English, French, German, Greek, Italian, Japanese, Korean, Mandarin Chinese (Simplified and Traditional), Portuguese, Russian, Spanish and Swedish (Lowensohn, 2009).

MeGlobe as in their official website in 2008 mentioned, MeGlobe was built to diminish language barriers from online communication. The free web client lets you type in your own language, but send a translated version, in real time, specific to the native tongue of whomever you are chatting with. With MeGlobe there is no such thing as 'lost in translation. **FIGURE 2.2** shows the interface of MeGlobe retrieved in 2008.

The text is translated using MeGlobe own-developed translation engine that rely on contributor feedback for improvement. This explained the small number of languages supported. Every time user send a message on MeGlobe's™ network, they have the opportunity to make MeGlobe translations engine better. User can let MeGlobe know by "editing" the translation when they notice that a translation on MeGlobe™ is a little off. This will update the translation engine knowledge based.

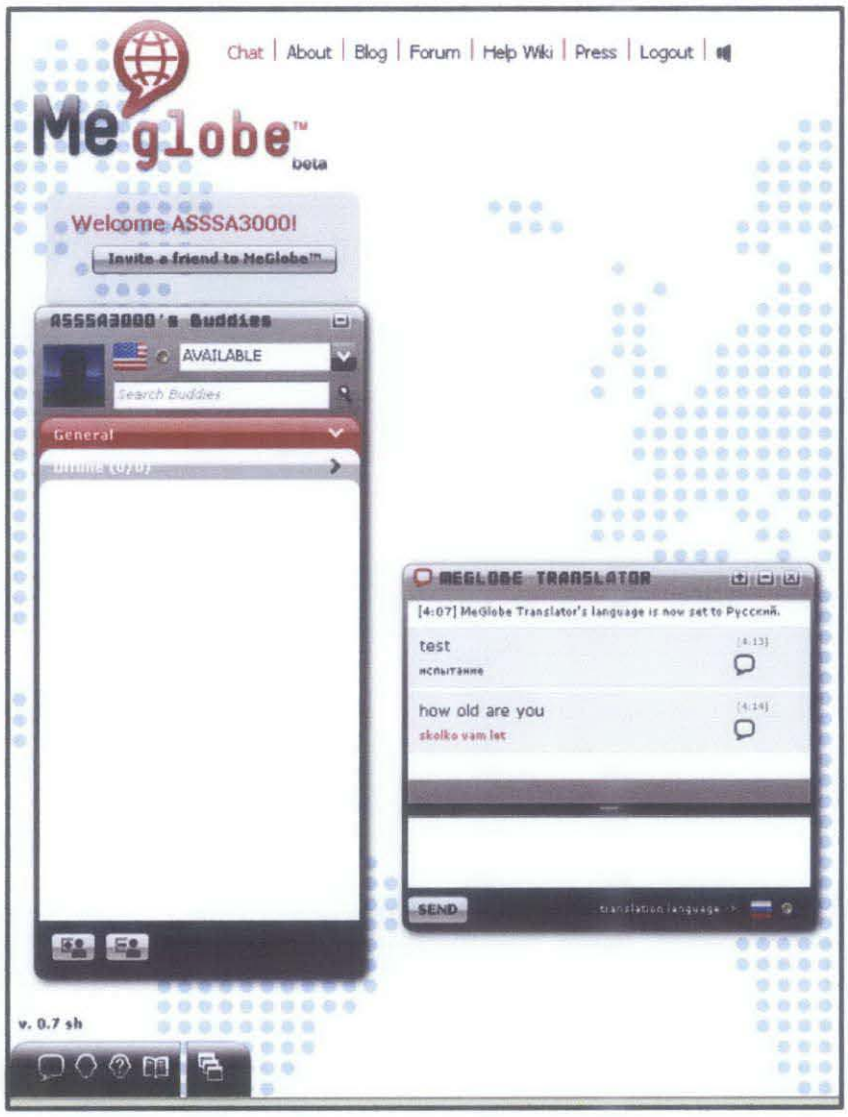


FIGURE 2.2: The interface of Meglobe as in beta release.

However, as of today, the official MeGlobe site seems no longer supported instant messaging as the main page is empty with just showing the MeGlobe logo and contact info. **FIGURE 2.3** shows the current MeGlobe official website retrieved in March 2011.

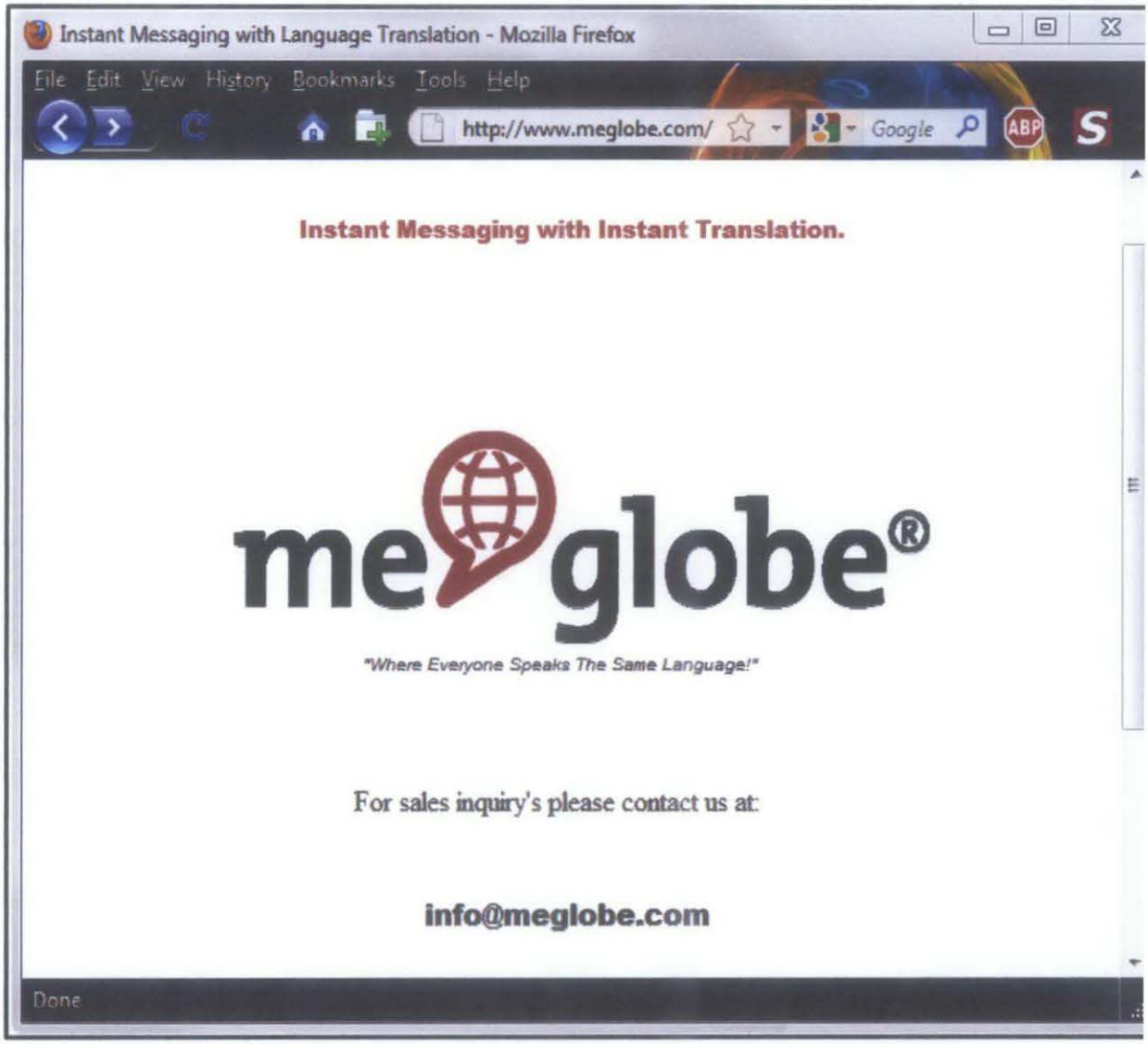


FIGURE 2.3: The official MeGlobe website

FIGURE 2.4 shows the statistic of unique visitors to the official website as provided by Compete. The statistic graph shown that since November 2010, there is no visit from unique visitor to the site.

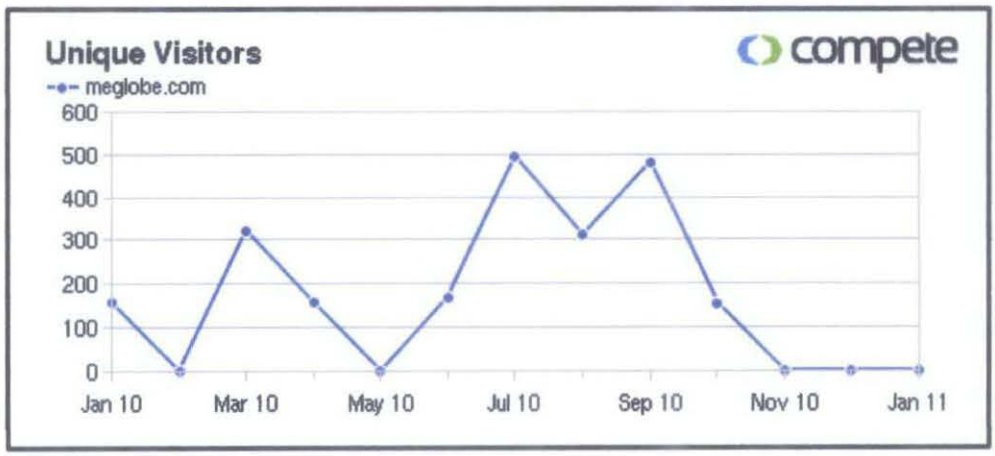


FIGURE 2.4: Statistic graph of unique visitors for MeGlobe

While MeGlobe program might have been discontinued, it does prove that the instant translation IM is possible. However, several key elements distinguish this project from MeGlobe, namely:

- MeGlobe offers chat application only in web browser (web-based) only
- MeGlobe is using own-develop translation engine which depending on users' (contributors) feedback for improvement which can be risky and costly
- Due to limitation of their own translation engine, MeGlobe only offers translation from and to 14 differrent languages.

2.3.2.1 GTalk Translate Bot

GTalk Translator Bot was released December 18, 2007 (Google, 2007). The translation bots provide a way to translate between GTalk contacts in a group chat or as a translation tool. This bot simply help users translate words while you carry on the conversations with friends in real time from one language to another. All users have to do is add one of 29 bots as a contact using their two letter language abbreviation. So in order to translate from an English conversation to a French one, user would add "en2fr@bot.talk.google.com" as a Google Talk contact.

The bot task is to translate whatever that user type in and translate to language that the bot speak. For instance, invite en2es@bot.talk.google.com (English to Spanish), open a chat with it, and then whenever user type something in English, the bot will repeat the same in Spanish (say e.g. "hello" and the bot correctly translates to "hola"). This feature might come in handy in group chat when you talk to someone with another native language. **FIGURE 2.5** shows an example of user chat with bot. The complete list of available bot offered can be found at Google support page, as of March 2011, there are 25 bots offered. The bot translation is done based on Google Translate engine.

FIGURE 2.5: User chat with English-to-Chinese bot (en2zh) using GTalk

Just like GTalk, the translator bot is also built on XMPP protocol and the source code is made available for developer who wants to manipulate the translator into their own program. For the development of this project, the coding process will be highly based on open source code, with the code of this bot made available; it will make it easier to study how the process of translation using Google Translate is done.

CHAPTER 3

METHODOLOGY

3.1 Modified Waterfall Development Model

The Modified Waterfall Development Model was selected as the method on developing this project. The component of this model is adapted from the traditional waterfall development model introduced by Winston W. Royce. While there are many different version of modified waterfall model, the best model selected for this project contains seven main progress flows (phases) as shown in **FIGURE 3.1**. The primary characteristic of the model is that developers are granted the flexibility to move up or down the flow process as necessary, especially to perform changes in previous steps.

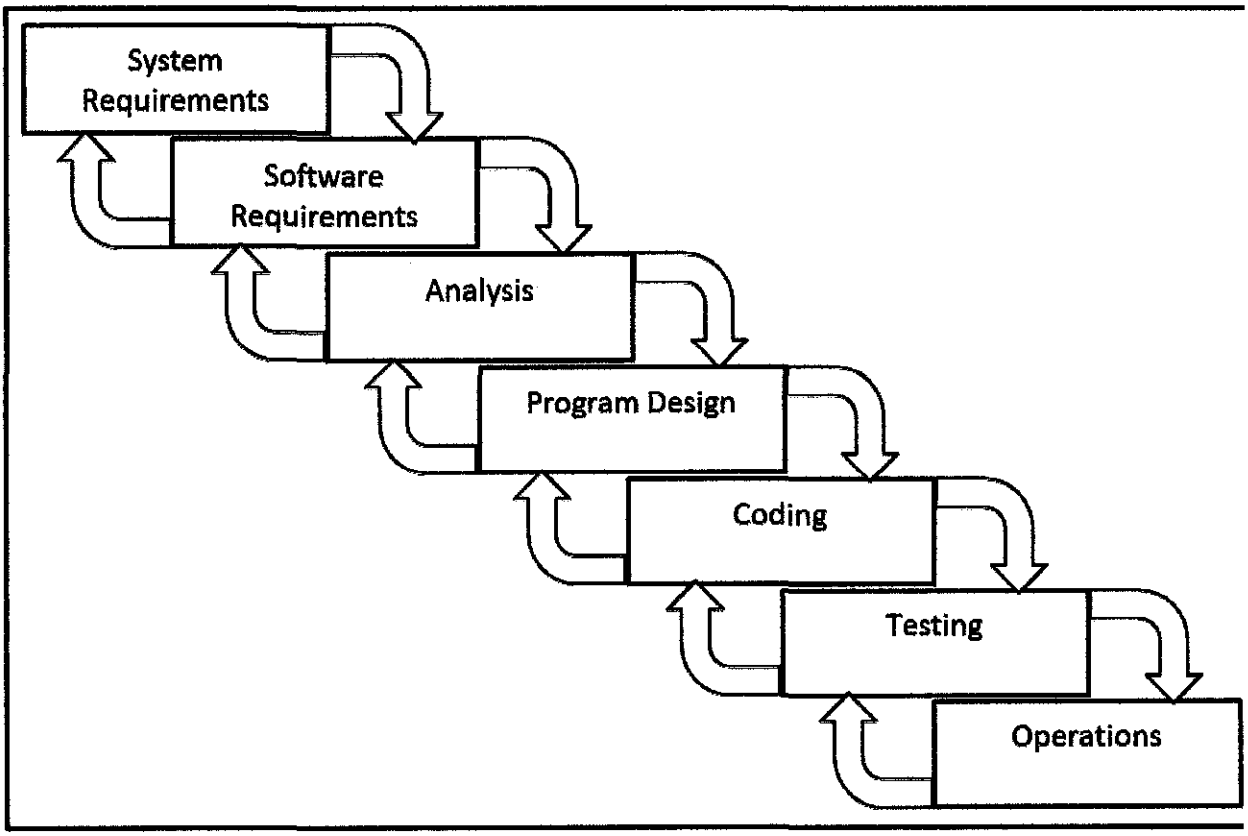


FIGURE 3.1: Modified Waterfall Development Model

3.2 Project Activities and Key Milestones

Project activities and key milestones are described based on phase breakdown of the modified waterfall model. This is to say that any activities and milestone defined can be revisit as the process continues following the nature of the modified waterfall model.

3.2.1 System Requirements

The majority of the system requirements phase will revolve around identifying system (i.e. hardware) requirements. System requirements will be developed based on a comprehensive literature review of technologies involved and similar system as described in chapter 2.

3.2.2 Software Requirements

The software requirements phase is functionally similar to the system requirements phase, but with a focus for software requirements. Both functional and user requirements will be identified through usability study, surveys, and interviews.

3.2.3 Analysis

The analysis phase will primarily consist of interpreting and providing context to the data and information gathered in the first two phases. Through proper organization and analysis, system & user requirements can be prioritized accordingly.

To-do List:

- Collect and study the graphical user interface (GUI) of existing related program that suitable too be implemented in the project.

3.2.4 Program Design

The program design phase will involve the process of designing the system architecture and functionalities according to the requirements developed in the analysis phase. Additionally, any material needed to develop the GUI based on previous requirements and analysis will be collected.

To-do list:

- Draw the draft the GUI as preparation for coding phase.

3.2.5 Coding

Basic functionalities of the system are coded using predetermined coding languages.

To-do list:

- Collect the open source code from existing project (i.e. from Google Code) that suitable to be implemented in the project. The code may need modification or to be write back in the chosen programming language.
- Build the basic functionalities of the program part-by-part.
- Combine and link the functionalities suitable to GUI in program design phase.

3.2.6 Testing

Once the first build of the program is ready, the testing phase will be conducted to identify any bugs and errors in the initial program. Program testing will be carried out to ascertain any technical and design flaw.

To-do list:

- Testing to be performed once coding phase is complete.
- Testing to be done in different type of operating system (environment).

3.2.7 Operations

This phase revolves around the implementation and integration of the most current (updated from testing result) built program. Al changes made during the coding phase will be finalized and employed in the current program build.

To-do list:

- Eliminate bugs and correct errors detected in testing phase.
- Finalization of the program designs and functions.

3.3 Gantt Chart

ID	Task	Week												
		1	2	3	4	5	6	7	8	9	10	11	12	13
1	System Requirement													
2	Software Requirement													
3	Analysis													
4	Analysis Result			X										
5	Program Design													
6	GUI Design					X								
7	Coding													
8	System Prototype								X					
9	Testing													
10	Evaluation and Modification													
11	Operations													

Legend:

	Progress
X	Milestone

TABLE 3.1: Project Timeline

3.4 Tools Required

The development of this project will require a variety of tools with which to develop and build the program including but not limited to:

3.4.1 Hardware

- Personal computer that runs multiple OS for development and testing purpose.

3.4.2 Software

- Windows XP Mode
- DEV C++
- Bidirectional-streams Over Synchronous HTTP (BOSH)
- Simple API for XML (SAX)
- Notepad++

CHAPTER 4

ANALYSIS & DISCUSSION

4.1 Questionnaire

4.1.1 Introduction

A questionnaire entitled "Perceptions on the use of Instant Messaging as instant translation tool" was randomly distributed to participants. In total 106 respondents are recorded to have completed the questionnaire. The questionnaire is designed to be as simple so those participants only have to spend little time completing the questionnaire. The following chapter shows the result of the questionnaire.

4.1.2 Section A: Participants Background Information

1. Nationality

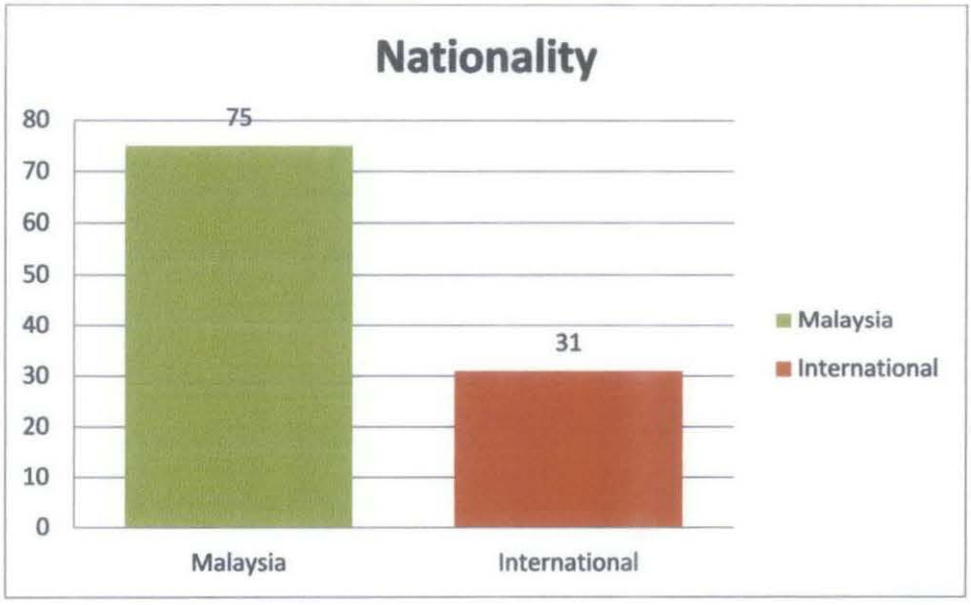


FIGURE4.1: Respondents by Nationality Column Chart

Out of 106 most of the respondents are Malaysian (71%) and the remaining is composed of international respondents of different nationalities.

2. Race

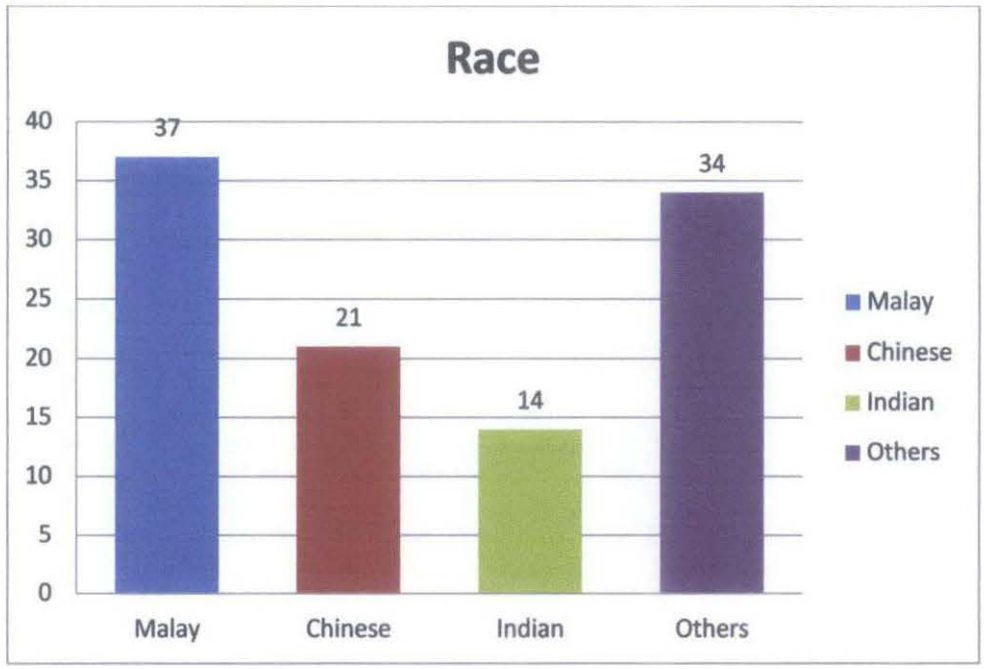


FIGURE4.2: Respondents by Race Column Chart

Malay respondents represent the majority with 37(35%). Those of other races makes up by 34(32%). By default, respondents of nationality other that Malaysian counted as "Others" race. Chinese respondents consist of 21(20%) and Indian consist of 14(13%) of total respondents.

4.1.2 Section B: Perceptions on the use of Instant Messaging as instant translation tool

In total there are 10 questions presented to respondents. The questions consist of fixed answer question and also open ended questions.

1. Have you ever used Instant Messaging (IM) program (eg: GTalk, Yahoo! Messenger, MSN, Office Communicator, etc)

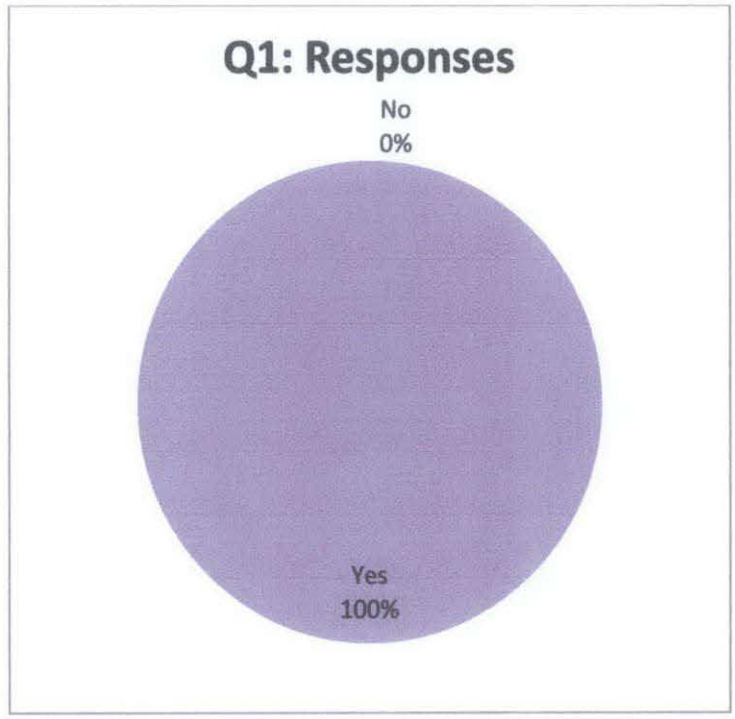


FIGURE4.3: Question 1 Responses Pie Chart

All of the respondents have experience with IM program. Since IM program is popular, this kind of result is expected.

2. If your answer to Q1 is YES, name one IM program that you mainly use.

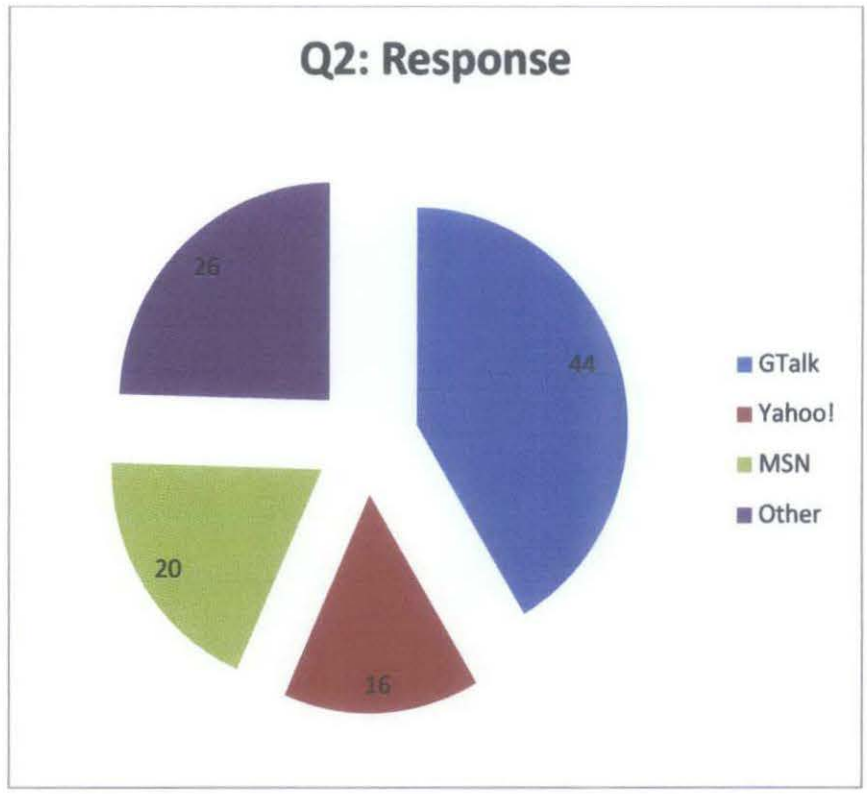


FIGURE4.4: Question 2 Responses Pie Chart

The responses to the second question are quite scattered. The popular IM programs namely GTalk and MSN are getting highest responses with 42% and 19%. The "other" is ranging from office program like Office Communicator to free IM program like Miranda and Digsby. Yahoo! Messenger and other messenger program contribute to 15% and 25% of the responses.

The result of surprisingly high responses on GTalk is due to the nature of the participants which most of them are UTP students. GTalk have been used widely by UTP students as a mean to communicate. The IM popularity is reflected in this questionnaire result.

3. Do you agree that the IM program should be simple in design and provide just enough essential functionality to avoid lagging and process hoarding. Rate in scale 1-5 (1 being strongly disagree and 5 being strongly agree).

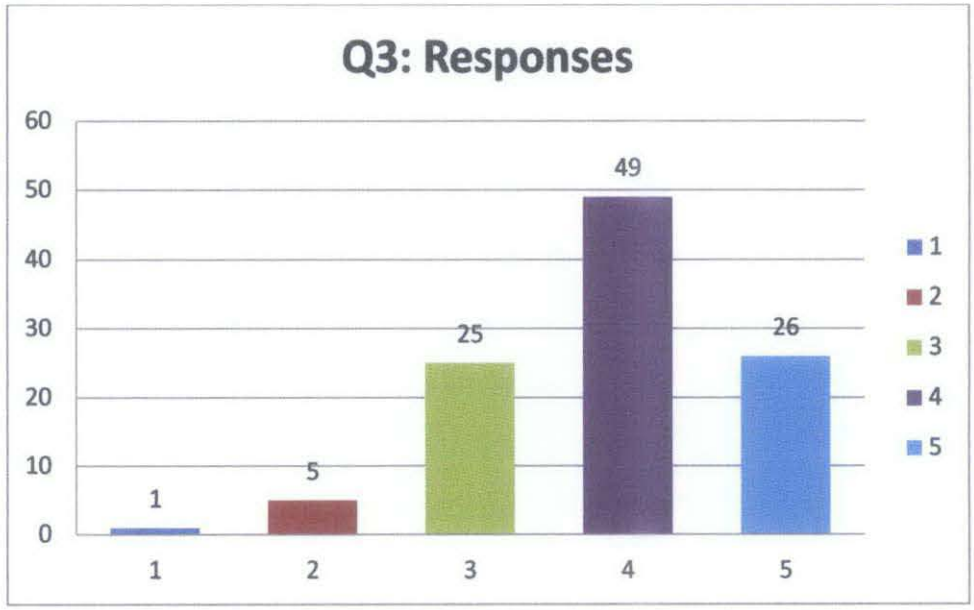


FIGURE4.5: Question 3 Response Column Chart

Majority of the respondents agreed with the statement, more than 70%. While there is 24% of the respondents "not sure" and in total of 6% respondents doesn't agree in the statement.

This shows that majority of the respondents prefer IM that is simple and not process hoarding. This result is tally with the previous question which GTalk is the main IM program used by participants as GTalk interface and functionality is simple yet delivers.

4. Do you often communicating with people with different mother tongue?

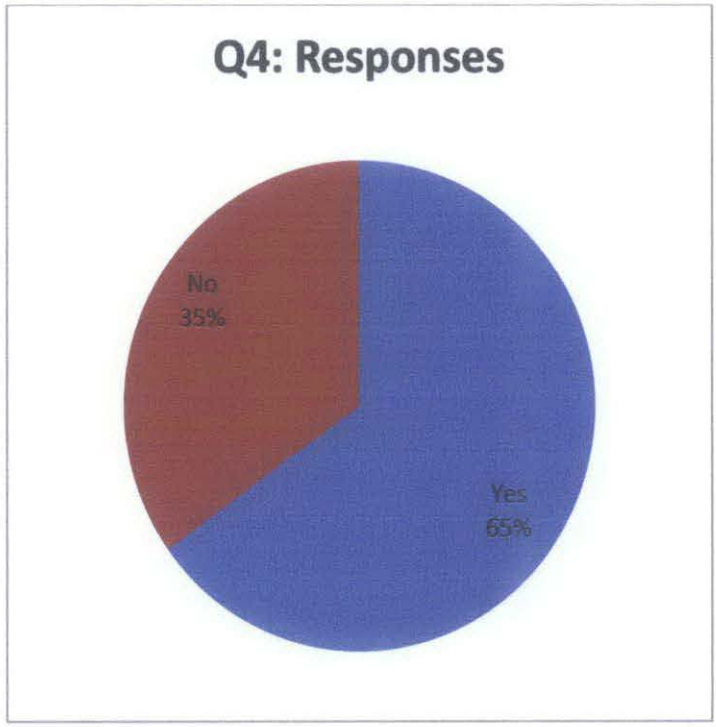


FIGURE4.6: Question 4 Response Pie Chart

The much larger number of respondents (65%) answered "Yes" for this question.

5. Continue from Question 4 (if your answer is YES). How hard it is to come to an understanding with each other? Rate in scale 1-5 (1 being very easy and 5 being very hard).

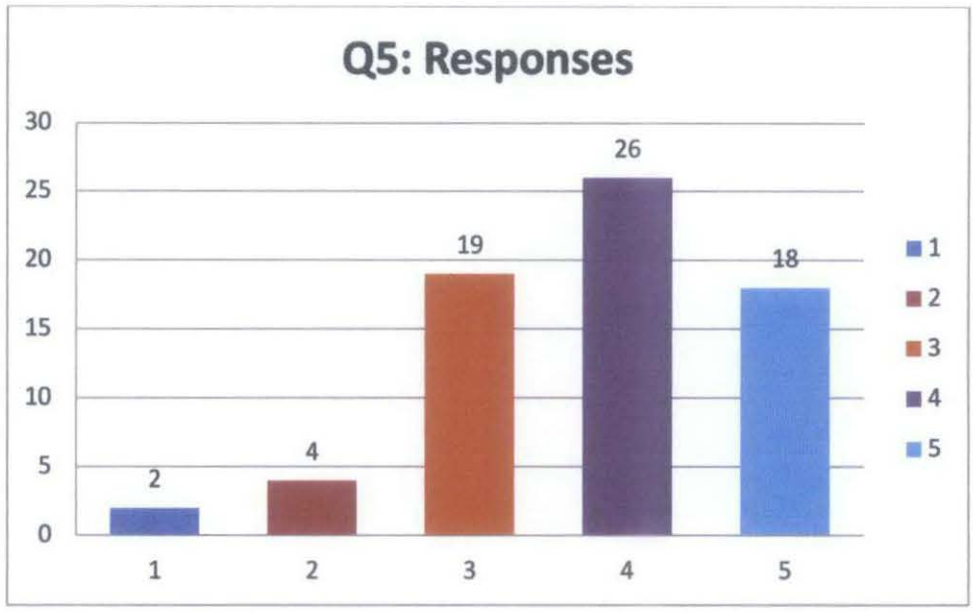


FIGURE4.7: Question 5 Response Column Chart

From the Chart, out of 69 respondents that answered "YES" in Question 4, 6 respondents find it easy and 44 finds it hard to come to an understanding when communicating with people of different mother tongue. The other 19 finds it somewhere in the middle.

This shows that majority of the respondent have difficulties in communicating with people that speaks different mother tongue. The group of respondents that rate the statement at 3 also should be counted as they do sometimes finds it difficult, even though there are times that they finds it easy to communicate.

6. Do you often use translator (either human or machine) to help you in translation process?

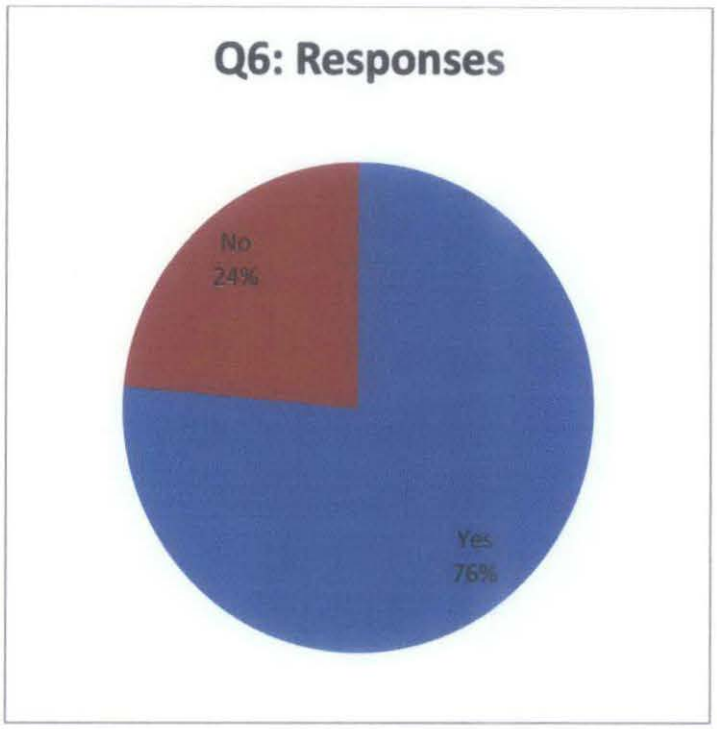


FIGURE4.8 Question 6 Response Pie Chart

76% of respondents provided a positive response to the statement, whereas only 24% answered NO.

This shows that the most of the respondents are familiar with the idea of machine translation. This question is aimed to quantify the number of respondent that are aware of the advantages as well as disadvantages of Machine Translator that they gain by experience.

7. How do you find the idea to have an IM that provides instant translation of the chat text? Rate in scale 1-5 (1 being strongly disagree and 5 being strongly agree)

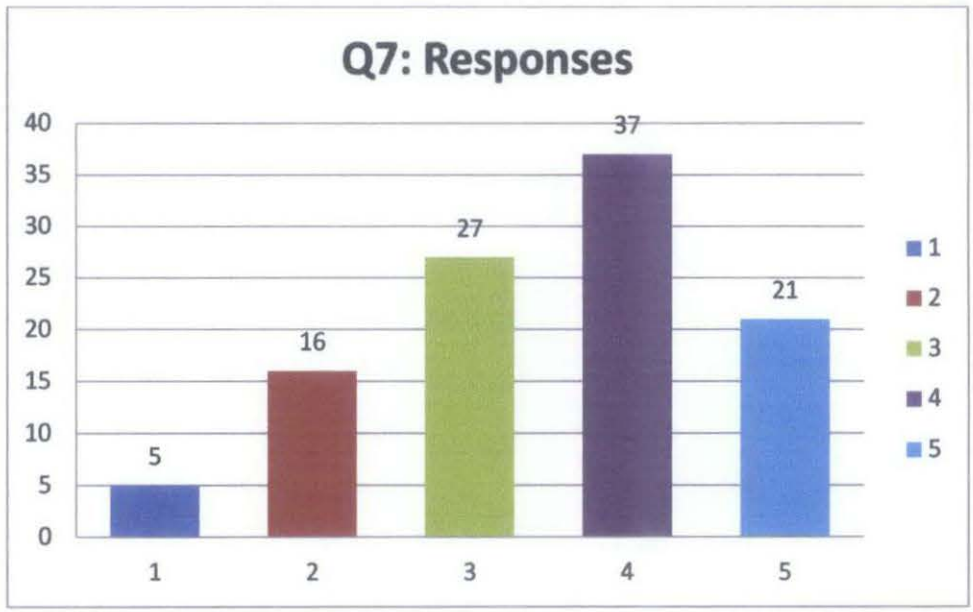


FIGURE4.9: Question 7 Response Column Chart

31 respondents seem to disagree with the idea and 27 of them fall under 'not sure' group. The other 48 respondents agreed with the idea.

A significant large number of respondents agreed means that they are ready for the evolution of IM which will allow the chat text to be translated instantly. However of note that there are group that is unsure with the idea, this could mean that they doesn't really understand the functionality of the idea.

8. Are you willing to sign up for another account just to access this new IM

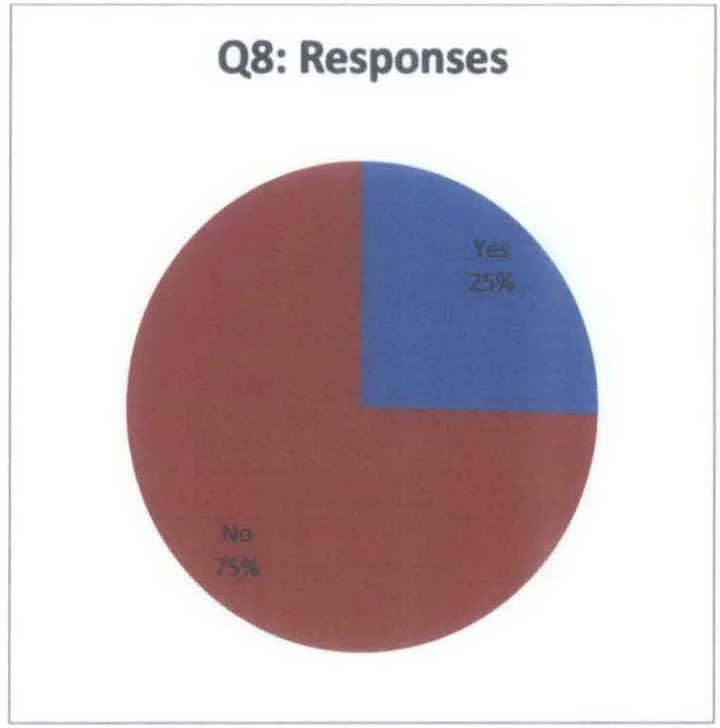


FIGURE4.10 Question 8 Response Pie Chart

The chart revealed that 75% (79) of the respondent not willing to sign up for new account just to have one function addition.

This could be because most of the participants already have multiple accounts and managing multiple accounts could be a hard job. IMs that allow multiple login using user's existing accounts is needed to attract user.

4.2 Questionnaire Analysis

The questionnaire results help the author to understand better on the design requirement that satisfy user the most and user willingness to accept this new functionality in their communication process.

Some of user preferences that makes clear from this questionnaire is the user prefer programs that is simple yet included the basic functionality. The main concern here is user does not prefer program that takes most of the PC process.

It is also clear that users are more attracted to IM program that has simple design and GUI. Besides that, it also clear that the program must allow user to login using their existing account. This means that users are not willing to sign up another account to use a new IM.

4.3 Discussion

4.3.1 System Requirements

After analyzing the questionnaire and the existing project, it is clear that user wanted a 'lightweight' program. Therefore the programs must be able to run in a lowest specification of PC so that it will run in all kind of computers. For this program, it should be able to run on minimum specification of: Windows XP basic or higher / Pentium III 233 or higher / 256MB RAM / 500M Free Hard Disk Space. The first prototype will be tested on specified requirements, if the program runs smoothly; it should impose no problem running on the higher specification machine.

4.3.2 Software Requirements

The software requirement of the program is based mostly on the existing system. For this program, the software requirement is quite low as based on most of the questionnaire respondent suggest that they prefer a simple program. However, the program should not ignore basic features such as connection configuration, help pages, and interesting GUI. The main priority is to figure out the mechanism to allow the program to established connection to a different server. This is as per user's requirement that specifies that they want the program to allow them to login using their existing accounts.

4.3.3 Analysis

The analysis are mostly done through literature review and the questionnaire. The analysis is mostly focusing on the requirement for both hardware and software requirement and also on the program design.

4.3.4 Program Design

4.3.4.1 System Architecture

This part will discuss the flow of the program. Since XMPP is implementing the client-server network architecture, the system architecture should cover both for client and server side. However, for the ease of understanding, the system architecture described here mainly covers the main features of the program which is to provide translation to the chat text. Most of other detail of functionality such as login and sending chat text architecture will be based closely on the existing systems.

FIGURE4.11 and **FIGURE4.12** shows the flowchart of the program on client and server side..

For FIGURE4.12 users need to enter their username, password and their speaking language upon logging in. This information will be send to the server side. Once the connection and chat session is established, users can send chat text to each other. However, the chat text will be send to server first to check whether the text needs translation or not. The process of identifying and translating the text is explained in FIGURE4.13. In case that both user speaks the same language, no translation is needed so the chat text will be send directly to the receiver.

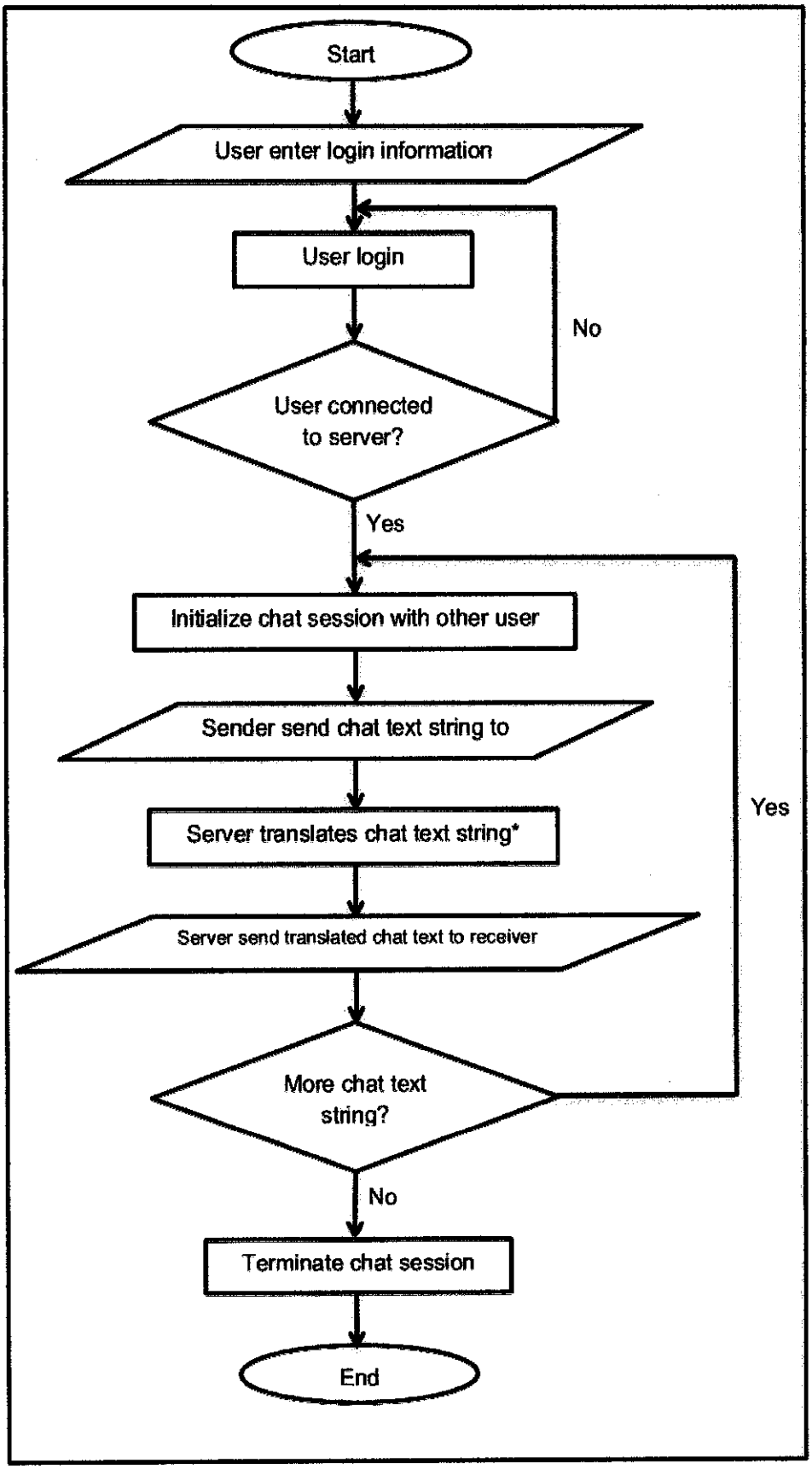


FIGURE4.11 Client Side System Architecture Flowchart

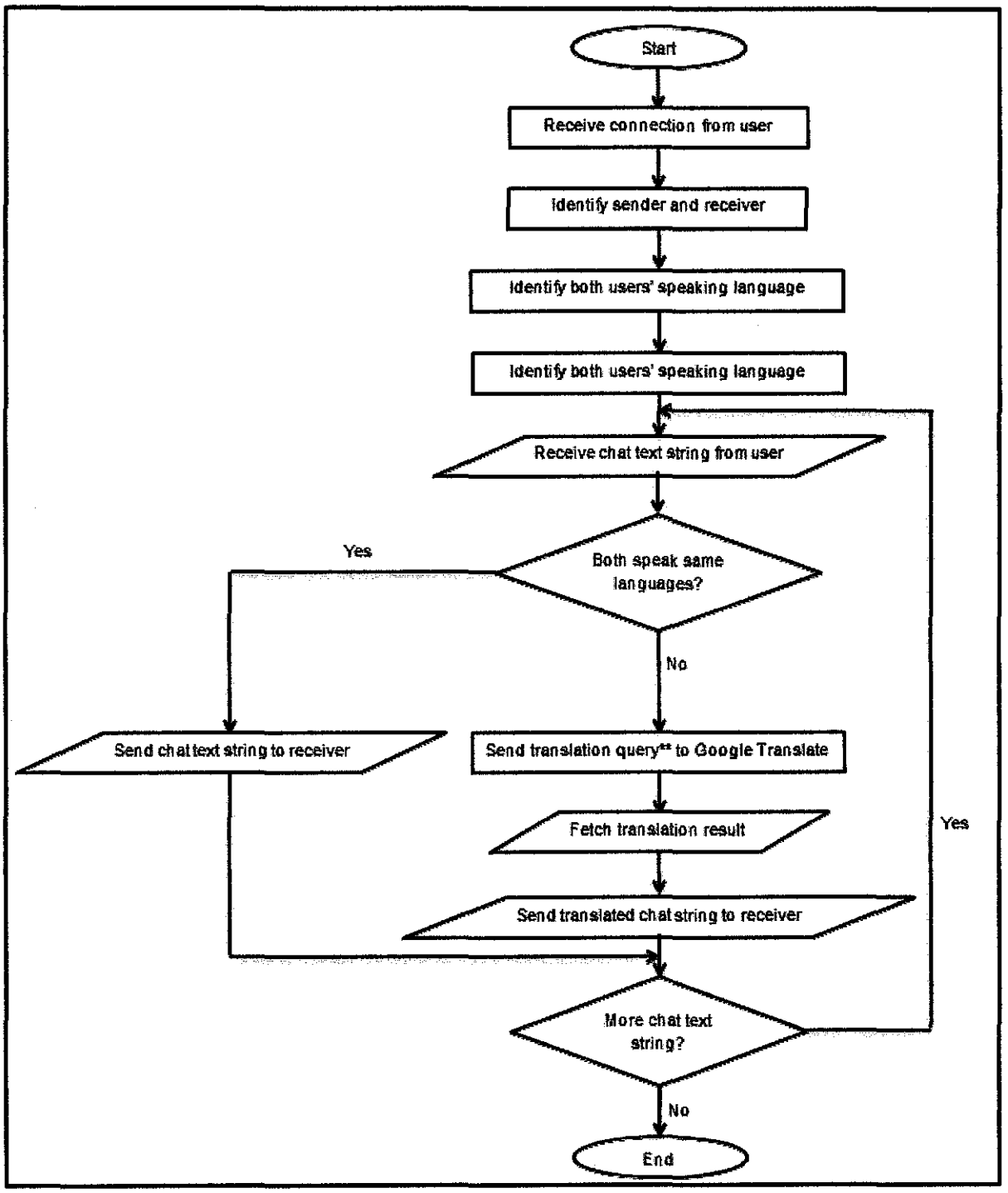


FIGURE4.12 Server Side System Architecture Flowchart

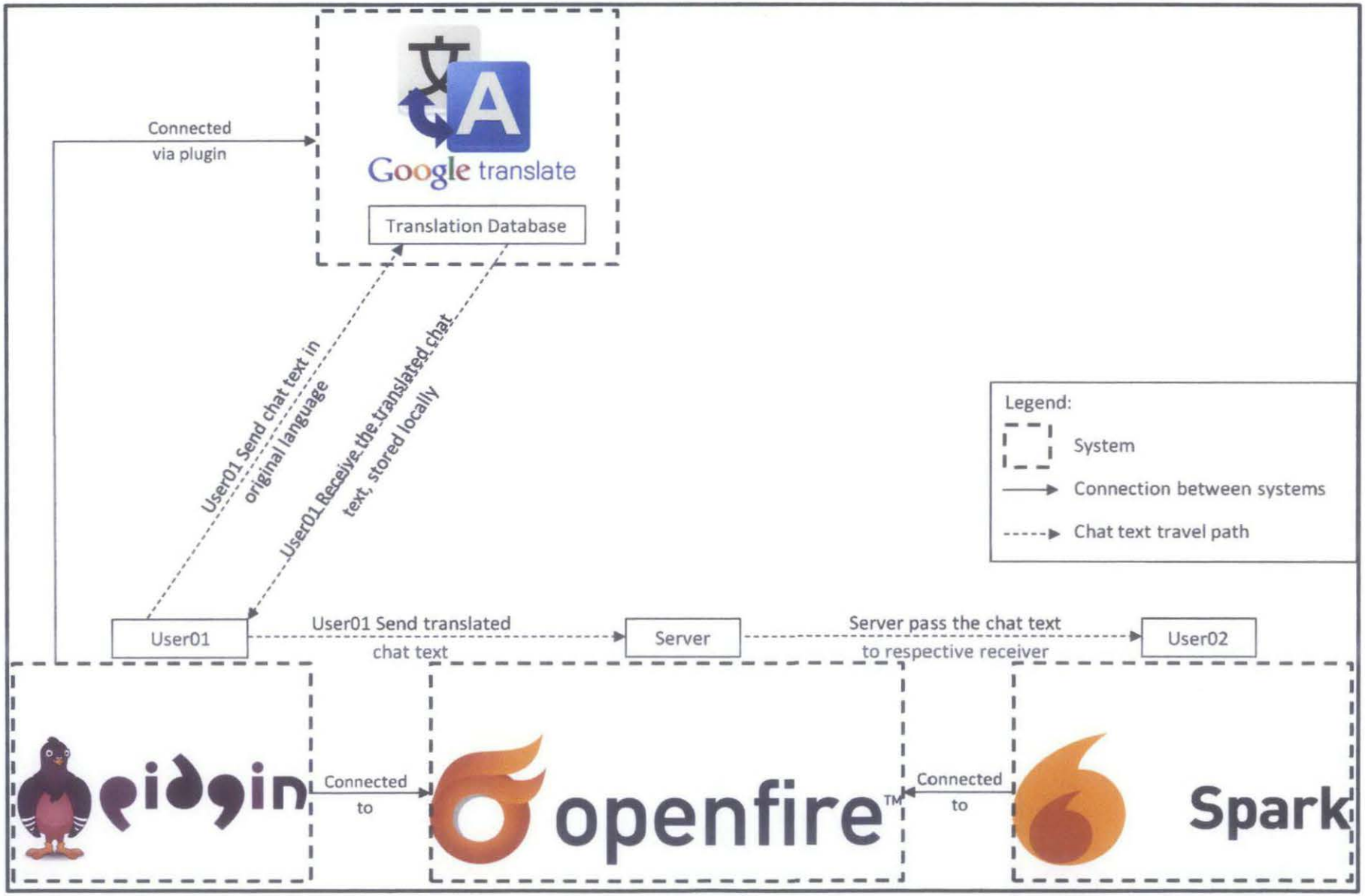


FIGURE4.13: Prototype system architecture for concept proving purpose

4.3.4.2 Prototype

FIGURE4.13 shows the architecture of the prototype built for this project to prove the concept of linking Instant Messenger program with translation service.

The IM is connected through local server hosted by free open-source XMPP server program called Openfire. The server is set up locally to avoid any network issues, ensuring continuous availability, and for easy troubleshooting. This set up is merely for testing purpose only, hosting the IM server on more stable and proper server space is recommended for the real implementation. Openfire is selected for its stability, simplicity, and easy installation. Moreover, Openfire is built on XMPP protocol, aligned with this project. The server preferences can simply be changed using browser. Server preferences for this project doesn't requires major changes, thus most of the options are set as default.

To test the workability of the translation function, two separated IM program are used, both free and open-source called Pidgin and Spark. This is to show that the project works despite the difference in user's IM program.

The auto-translation function is loaded into Pidgin IM. Pidgin IM is selected because it supports plugin which allow anyone to build plugin to suit their need. Pidgin also allows multiple login, support almost all accounts and support local XMPP server account. Pidgin is the suitable IM program to be used as the testing platform for this project.

The plugin is coded in C language as it is the only language supported by Pidgin's plugin. The compilation is done using Cygwin to compile the C code into .dll format in Windows system. The plugin can also be compiled into .so format that can be used in Linux machine. The compiled plugin can be loaded into Pidgin by simply added the .dll or .so file into the plugin directory.

For this pilot run, the selected translation service is Google Translate. The plugin send the outgoing chat text from Pidgin IM based on user's language preferences to the

Google Translate server, and retrieve the translated text which then will be sent to the respective user. The plugin also will translate the incoming chat text sent to the Pidgin IM user. The detailed functions of the plugin is discussed in the next section.

FIGURE4.14 below shows the screenshot of a sample run of the architecture. The figure show the chat between admin (English speaking user) and user1 (Malay speaking user). The sample is ran on the Windows 7 system under the architecture as explained.

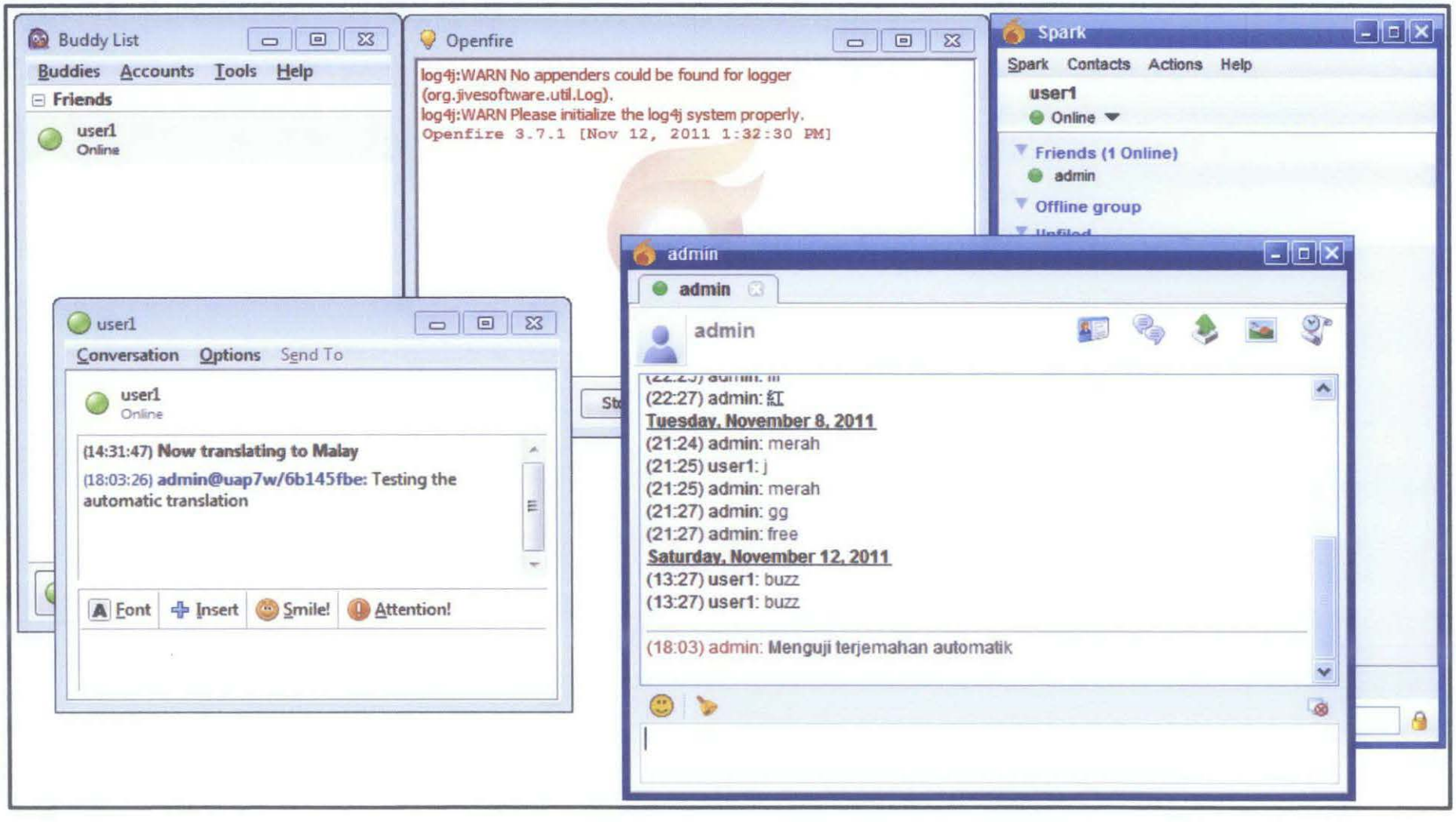


FIGURE4.15: Sample run of the automatic translation.

4.3.4.3 Graphical User Interface (GUI)

The GUI is developed under the plugin as an additional functionality to the Pidgin IM. The following function is added to the IM:

Sender language selection.

FIGURE4.16: Sender language selection

The list of supported language is populated from all the supported language by Google Translate. The code will used the language selected as the origin in the translation pair later. The language is understood by the code by its corresponded two-letter representation as denoted by Google. For example, English is denoted as EN, Malay as MS, Japanese as JA and so on. If the user didn't specify the language, the plugin will set it to default language, English.

Translation Service Selection

This feature is for future development, the idea is to have user to choose different translation service according to their preference. This is to eliminate the weakness suffered by one translation service by having a choice to select another. The default for the current version is Google Translate. The proposed additional translation service would be Yahoo! Translate and Bing Translate. However, the addition will require the code modification thoroughly as different translation service using different query and notation for the translation.

FIGURE4.17: Translation Service Selection

Receiver language selection

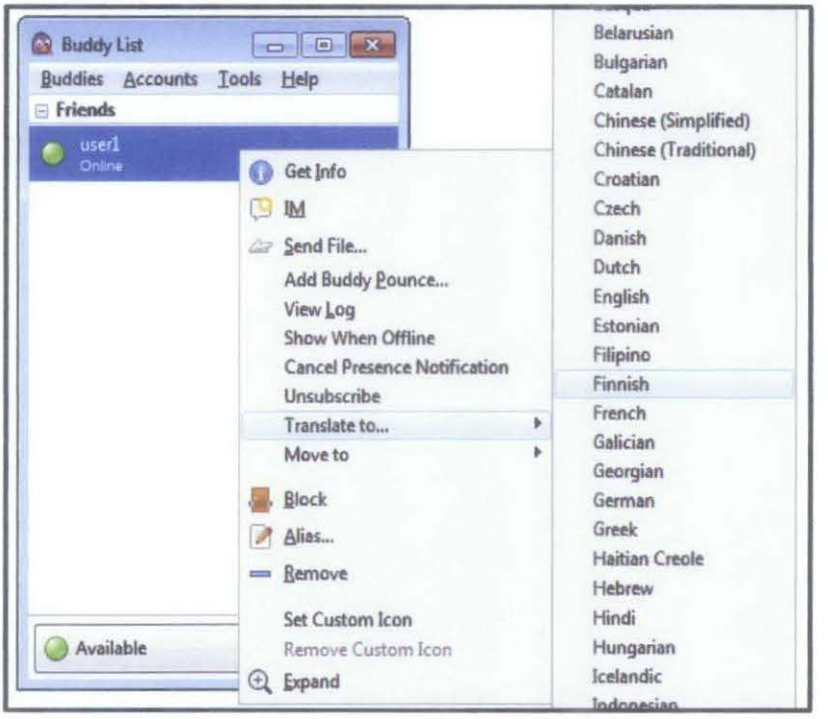


FIGURE4.18: Receiver Language Selection

After selecting user's own spoken language, he/she can specify the language the chat text will be translated to another user. The plugin will add an additional right-click menu option that says "Translate to...". The language list available is the same as the previous one. User can specify one language for every friend and it can be changed anytime. User will be notified as they selected the language in the chat box of that particular user. In this case, lets chose Malay as "Translate to..." language for user1 and whenever user click on the user to start a chat, notification of what language the chat text will be translated will appear.

4.3.5 Coding

The coding process was started soon after the whole program design is finalized. The program is implementing lots of open source code that is widely available and free for use. Most of the code is from Google Translator API projects which modified and re-wrote from Java to C programming language. Sufficient time was allocated for selection and modification of the code.

As part of proving the concept, the initial plan of coding the whole IM was dropped due to time constraint. Instead, this phase is focused on building the translation function that can be used on top of Pidgin IM.

4.3.6 Testing

The testing is conducted on the prototype to ensure that it run smoothly and free from any bug. The testing methods follow the standard procedure and changes was made to the coding part as required. The test run had been conducted in three different operating system; Windows 7, Windows XP and Ubuntu (Linux based). Since three main component used, Pidgin, Spark and Openfire is built for both, Windows and Linux system, there are no problem detected. The testing in all three OS are done successfully.

The testing proved that the plugin works well, however, recent changes in Google Translate API version one (V1) which limiting the character in translation post some significant issue to the system. Since the current prototype is built using Google Translate API V1, the program suffers the same limitation. The limitation causes only the first string of chat text to be translated, and the other to be ignored.

To tackle the problem, the codes need to be modified to suit the Google Translate API version two (V2) which allow 1 million character of translation per day for the cost of USD20. Since the translation rely on the Google Translate API, the changes is unavoidable to ensure the smooth run. Another solution to the problem is by adding more translation service option. However, this method require heavy changes in the codes therefore it is recommended for future development.

CHAPTER 5

CONCLUSION & RECOMMENDATIONS

The smooth communication between communicating parties is important as a path to come to an understanding. The language barrier can be break by having a translation program that allows them to communicate in real time.

Despite successfully connecting the IM and the translation service, the quality of the translation provided is limited to the one provided by the translation service. It is recommended that the future program will allow more selection on available translation service, means that users are allowed to choose from a list of translation services available not only Google Translate. With this feature made available, the limitation of any one particular translation engine can be eliminated and user can choose the translation service that they feel is the best.

Apart from that, in the current development, it is known that the program is lacking of error notification. It is recommended that the program allow connection testing to ensure the connectivity to the translation server prior to connecting to the account server.

It is understood that IM users tend to use short form or informal language in chatting. However, the usage of informal language can cause the chat text to not be translated as desired as the machine translator only understand lists of words in their database. So it is recommended to include the auto-correction or suggestion function into the IM so that any short-form used will be corrected by the IM and will be understood by the machine translator.

In conclusion, the development of Auto-Translation Instant Messenger provides a pathway to users who want to communicate with friends who don't speak the same language in the comfort of speaking their mother tongue. At the same time, it is hoped that throughout the process of developing this program and upon completion, the author has achieved all of the objectives successfully.

REFERENCES

1. Jack Moffitt (2010) *Professional XMPP Programming with Javascript® and jQuery.* Indianapolis, Wiley.
2. Peter Saint-Andre, Kevin Smith, Remko Tronçon (2009) *XMPP: The Definitive Guide.* First Edition. California, O'Reilly.
3. Iain Shigeoka (2002) *Instant Messaging in Java: The Jabber Protocols.* Greenwich, Manning.
4. Saint-Andre, P.; (2007) "Jingle: Jabber Does Multimedia," *Multimedia, IEEE* , vol.14, no.1, pp.90-94, Jan.-March 2007
5. Jennings, R.B.; Nahum, E.M.; Olshefski, D.P.; Saha, D.; Zon-Yin Shae; Waters, C.; (2006) "A study of Internet instant messaging and chat protocols," *Network, IEEE* , vol.20, no.4, pp.16-21, July-Aug. 2006
6. Saint-Andre, P. (2005) "Streaming XML with Jabber/XMPP," *Internet Computing, IEEE* , vol.9, no.5, pp. 82- 89, Sept.-Oct. 2005
7. Chatterjee, S.; Abhichandani, T.; Haiqing Li; Tulu, B.; Jongbok Byun; (2005) "Instant messaging and presence technologies for college campuses," *Network, IEEE* , vol.19, no.3, pp. 4- 13, May-June 2005
8. Paulson, L.D.; (2001) "Translation technology tries to hurdle the language barrier," *Computer* , vol.34, no.9, pp.12-15, Sep 2001
9. Ozturk, O. (2010) "Introduction to XMPP protocol and developing online collaboration applications using open source software and libraries," *Collaborative Technologies and Systems (CTS), 2010 International Symposium on* , vol., no., pp.21-25, 17-21 May 2010
10. Calefato, F.; Lanubile, F.; Minervini, P.; (2010) "Can Real-Time Machine Translation Overcome Language Barriers in Distributed Requirements Engineering?," *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* , vol., no., pp.257-264, 23-26 Aug. 2010
11. Mengjun Xie; Zhenyu Wu; Haining Wang; (2007) "HoneyIM: Fast Detection and Suppression of Instant Messaging Malware in Enterprise-Like Networks," *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* , vol., no., pp.64-73, 10-14 Dec. 2007

12. Ishida, T. (2006), "Language grid: an infrastructure for intercultural collaboration," *Applications and the Internet, 2006. SAINT 2006. International Symposium on* , vol., no., pp.5 pp.-100, 23-27 Jan. 2006
13. Maiko Kozakai (2004) Master Thesis, *Jabber/J2ME based Instant Messaging Client for Presence Service*, University of Applied Science Stuttgart – Hochschule der Medien, June 4, 2004
14. Lin, Chin-Yew. (1999) Machine Translation for Information Access across Language Barrier: the MuST System. In *Machine Translation Summit VII*, Singapore, September 13-17, 1999
15. Royal Pingdom (2010). *Amazing facts and figures about Instant Messaging.* Retrieved from http://royal.pingdom.com/2010/04/23/amazing-facts-and-figures-about-instant-messaging-infographic/
16. Ethan S. (2010) *Comparison of online machine translation tools.* Retrieved from http://www.tcworld.info/index.php?id=175
17. The Radicati Group, Inc. (2009) *Instant Messaging Market, 2009-2013.* Retrieved from http://www.radicati.com/?p=4647
18. Josh L. (2009) *Meglobe opens up to other IM networks (and you).* Retrieved from http://news.cnet.com/8301-17939_109-10162889-2.html
19. Barry S. (2007) *Google Translate Drops Systran For Home Brewed Translation.* Retrieved from http://searchengineland.com/google-translate-drops-systran-for-home-brewed-translation-12502
20. Franz O. (2006) *Statistical machine translation live.* Retrieved from http://googleresearch.blogspot.com/2006/04/statistical-machine-translation-live.html
21. Google Inc. (n.d) *Google Language API Family.* Retrieved from http://code.google.com/apis/language/