

Web Mail Information Extraction

by

Bahariah Binti Latan

**Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information and Communication Technology)**

JANUARY 2011

**Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

CERTIFICATION OF APPROVAL

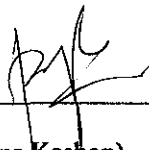
Web Mail Information Extraction

by

Bahariah Binti Latan

A project dissertation submitted to the
Computer & Information Science Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION & COMMUNICATION TECHNOLOGY)

Approved by,



(Rozana Kasbon)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

January 2011

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.



BAHARIAH BINTI LATAN

ABSTRACT

This project is conducted as to deliver the background of study, problem statements, objective, scope, literature review, methodology of choice for the development process, results and discussion, conclusion, recommendations and references used throughout its completion. The objective of this project is to extract relevant and useful information from Google Mail (GMail) by performing Information Extraction (IE) using Java programming language. After several testing have take place, the system developed is able to successfully extract relevant and useful information from GMail account and the emails come from different folders such as All Mail, Inbox, Drafts, Starred, Sent Mail, Spam and Trash. The focus is to extract email information such as the sender, recipient, subject and content. Those extracted information are presented in two mediums; as a text file or being stored inside database in order to better suit different users who come from different backgrounds and needs.

ACKNOWLEDGEMENT

The author wishes to take the opportunity to express her utmost gratitude to the individuals that have given their time and effort to assist the author in completing the project. Without the cooperation of these individuals, no doubt the author would have faced some minor complications through out the course.

First and foremost the author's utmost gratitude goes to the author's supervisor, Miss Rozana Kasbon. Without her guidance and patience, the author would not be succeeded to complete the project. Million thanks also should be given to the Final Year Project Coordinator, Miss Siti Rohkmah Bt M Shukri with all the initial information required to begin the project.

To all individuals that have helped the author in any kind of ways, but whose name is not mentioned here, the author thank you all.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL	i
CERTIFICATION OF ORIGINALITY	ii
ABSTRACT	iii
ACKNOWLEDGEMENT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLE	viii
LIST OF ABBREVIATIONS	viii
CHAPTER 1: INTRODUCTION	1
1.1 Background of Study	1
1.2 Problem Statements	1
1.3 Objective	3
1.4 Scope of Study	3
CHAPTER 2: LITERATURE REVIEW	4
2.1 Concepts of IE	4
2.2 Concepts of WG	5
2.3 From IE to WG	6
2.4 Recent IE Projects	8
2.5 Email Forensic Analysis	10
CHAPTER 3: METHODOLOGY	12
3.1 Waterfall Methodology	12
3.2 Development Tools	14

CHAPTER 4:	RESULTS AND DISCUSSION	16
4.1	Proposed Framework	16
4.2	Process Flow of Web Mail Information Extraction	17
4.3	System Interface	23
4.4	Testing Outputs	24
4.5	Test Cases	27
CHAPTER 5:	CONCLUSION AND RECOMMENDATIONS	28
5.1	Conclusion	28
5.2	Recommendations and Future Works	29
REFERENCES		30
APPENDICES		32

LIST OF FIGURES

Figure 2.1	Chronological history of IE and WG	7
Figure 2.2	The architecture of LiXto, from [11]	8
Figure 2.3	The architecture of Fetch Agent Platform, from [13]	9
Figure 2.4	Email mining in email forensic analysis	11
Figure 2.5	Email statistic in email forensic analysis	11
Figure 3.1	Waterfall methodology	12
Figure 4.1	The proposed framework	17
Figure 4.2	Process flow of web mail information extraction	18
Figure 4.3	Details of <i>Extract emails</i> function	18
Figure 4.4	Details needed to connect to mail server	19
Figure 4.5	To get folder for extraction	19
Figure 4.6	Message number	20
Figure 4.7	Get the sender's email address	20
Figure 4.8	Get the recipient's email address	20
Figure 4.9	Get the subject element	21
Figure 4.10	Get the content of the email	21
Figure 4.11	Connection to Derby database	22
Figure 4.12	Connection to database and email is been flagged as seen	22
Figure 4.13	Main page of the system	23
Figure 4.14	Dropdown menu for <i>Folder</i>	24
Figure 4.15	Dropdown menu for <i>Mark as</i>	24
Figure 4.16	Original emails in <u>bahariah.latan@gmail.com</u>	24
Figure 4.17	Final output to be saved in text file	25
Figure 4.18	Final output to be saved in text file (cont.)	26
Figure 4.19	Final output to be stored in database	26

LIST OF TABLE

Table 4.1 Test cases

27

LIST OF ABBREVIATIONS

IE	Information Extraction
WG	Wrapper Generation
SQL	Structured Query Languages
JAF	JavaBeans Activation Framework
NLP	Natural Language Processing
WI	Wrapper Induction
JRE	Java Runtime Environment
SDK	Software Development Kit
JSP	Java Server Pages
XML	Extensible Markup Language
HTML	Hyper Text Markup Language
API	Application Programming Interface
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
IMAP	Internet Message Access Protocol
POP	Post Office Protocol
GMail	Google Mail

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 Background of Study

The field of Information Extraction (IE) is a fairly new field that has evolved over the last decade. It then leads to the appearance of Wrapper Generation (WG) which aims to automate the traditional ways of performing IE. As the Web acts as the central source of an enormous amount of online documents nowadays, it turns out to provide a special challenge and has been the driving force behind research on IE [1] [2] [7]. IE is conducted not only to perform on Websites, files and documents but also many other sources like Web mail.

Thus, this project is initiated with the aim to extract useful and relevant information from Web mail which are the sender, recipient, subject and content in particular email whereby those extracted information can be used for many purposes and help different people who come from different backgrounds and have different needs.

1.2 Problem Statements

1.2.1 Problem Identification

As the Web is becoming huge and provides a vast source of information, the issue of extracting, integrating information from various Web sources and then presenting those relevant, desired information becomes important to the users [22]. Therefore there is a stringent necessity to learn on how IE works, and then implement it at certain level in order to help people and contribute to the society.

For example, for a regular user of Web mail, it is troublesome to every time log in to the mail account just for sake of checking new emails. If there are huge in number of emails come, user needs to one by one click at the email so that they can view the emails' contents.

As a staff in a company, there is a situation whereby they need to send promotion's email or newsletters to potential customers and clients via email and the number is not small. Furthermore, they also intend to keep track the status of delivery to all expected recipients in order to ensure the emails sent reach their mailbox.

Moreover, especially in Malaysia, from research done, forensic experts apparently did not have an adequate, efficient tool to deal with email forensic analysis and to perform a multi-staged analysis of email ensembles in a timely fashion [26].

1.2.2 Significant of the Project

This project is significant as there is a need for Web mail information extraction in such a way to help people and society.

In terms of personal use, regular users can reduce their frequency of logging in to their mail accounts just in case new incoming messages arrive and need not to click every email in order to see the content as with the use of this system, users can view many emails in a glance and just scrolling down the text file if the emails are huge in number. This technique may help active users who value their time the most and prefer simple, clean method of reading emails.

As the extracted information of Web mail is to be stored in a database, any keyword searching activities like SQL query can be done to deal with large number of emails collection. For a company which tends to send promotion or newsletter to their target customers or clients in a huge amount at a one time, the same approach of this system can be implemented to extract their email addresses from any Web sources.

At the other side, this system also can help forensic experts to deal with cyber crimes by performing email information extraction and then continue with the next stage of multi-staged email analysis activities in which aims to identify unforeseen pattern and behavior in a collection of emails.

1.3 Objective

The main objective of this project is to extract useful and relevant information from Google Mail (GMail) by performing IE using Java programming language. Extracted information can then be saved as a text file or to be stored in a database for post-processing activities.

1.4 Scope of Study

This project focuses on performing IE on GMail accounts. Elements to be extracted are the sender, recipient, subject and the content itself.

Extracted information is to be saved as a text file and to be stored in a database depending on users' preferences and purposes of using the system. Text file and database are mediums in presenting those extracted information. As a starting point, IE shall be conducted in such a way irrespective the emails are read or unread. Other constraint is only one email address can occupy into one column in a particular table inside database.

The system shall be written using Java as the main programming language for the IE process at the back end side while other programming languages act as complements to each other. Some third party software are used to speed up the process of development such as JavaMail and JavaBeans Activation Framework (JAF).

CHAPTER 2

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 Concepts of IE

According to Muslea (1999)

Information Extraction (IE) is concerned with extracting the relevant information from a collection of documents. (p.1)

The key component of any IE system is its set of extraction patterns or also known as extraction rules (Eikvil, 1999, p.5) (Muslea, 1999, p.1) (Mecca, 2002, p. 9).

Muslea (1999) added, as “IE systems rely on the extraction rules” to extract relevant information, several research efforts have focused on “learning the extraction rules from training examples provided by the users as writing traditional extraction rules is a difficult, time-consuming task” (p.1) [21] [22] [23]. Several types of “extraction rules can also be generated by machine learning algorithms” [3] [21] [22]. For example to extract information from “grammatical, free text document, AutoSlog, LIEP and CRYSTAL extraction rules can be used while for online documents, WHISK, RAPIER and SRV extraction rules” [2].

On the other hand, Sarawagi (2002) defines IE task by given E: a set of structured elements which is the target schema, S: unstructured source, and leading to a statement of extract all instances of E from S. As level of difficulty in extracting information varies depending on input and kind of rules, she classified extraction rules into three which are text segmentation, HTML wrapper and classical IE.

Chang and others (2006) proposed a three-dimensional representation of IE features: the first dimension evaluates the difficulty of an IE task, the second compares the various techniques and the third compares both the training effort of a user and the necessity to port an IE system across different domains.

2.2 Concepts of WG

Wrapper Generation (WG) field has actually appeared from the necessity of extracting and integrating data from multiple Web-based sources in the field of Information Extraction (IE) [1] [2] [3] [22] [23]. A typical wrapper application is often written by hand [1] [23, p.6]. As some efforts have been conducted, wrappers then can be generated by machine learning algorithm which is less time-consuming [24]. “Wrapper induction systems”, the next generations of wrapper, on the other hand, have the “functionality of generating delimiter-based rules that do not use linguistic constraints” (Eikvil, 1999, p.13) (Muslea, 1999, p.5). Today’s IE systems use extraction patterns are based on one or both approaches, the syntactic/semantic constraints and delimiter-based [1] [2].

Hsu and Dung [5] classified wrappers into four categories which are hand-made wrappers using general-purpose programming languages, designed programming languages, heuristic-based wrappers and Wrapper Induction (WI) approaches.

Beside that, a complete categorization was made by Laender [6]. They proposed the taxonomy of languages for wrapper development, HTML-aware tools, NLP-based tools, wrapper induction tools, modeling-based tools and Ontology-based tools. They also compared among the tools using several features like degree of automation, support for complex objects, page contents, availability of a GUI, XML output, support for non-HTML sources, resilience and adaptiveness.

According to [1] [7] [21]

“Information Extraction (IE) from Web pages is normally performed using wrappers”. “A wrapper is a procedure” that is designed to access HTML documents and export relevant information to a structured, fixed and unambiguous format as the output. “Wrappers consist of a series of rules and some codes should be written in order to apply those rules and generally are specific to a source”.

Cosulschi M. (2000) noted that a wrapper is also known as a program that aims to identify data of interest and put them into some suitable formats and eventually store back into a relational database. According to Eikvil [1, 8] a classification of Web wrappers can be made on the base of the kind of HTML pages that each wrapper is able to deal with. There are three different types of Web pages that can be distinguished by free, unstructured pages, semi-structured pages and structured pages [7]. Besides the HTML page structure, effective wrappers consider also the structure of hyperlink as it may reveal relevant information [1].

Sarawagi (2002) classified Web sites wrappers according to the amplitude of the tasks they are able to face as distinguished below.

- Record-level wrappers - Extract elements of a single list of homogeneous records from a Web page and discover record boundary by detecting regularity.
- Page-level wrappers – Extract elements of multiple kinds of records.
- Site-level wrappers - Extract elements and convert it into structured format for an entire Web site.

2.3 From IE to WG

IE has started with a typical wrapper application, where people needs to write extraction rules to create wrapper by himself and figures out the structure of Web pages before sets the pattern of returned result. This technique is rather tedious and time-consuming, and then it led to another technique of generating wrapper, which is using machine learning algorithm. In this technique, the machine learning algorithm learns the extraction rules by providing it a lot of training examples and it still need human intervention [10]. Then researchers found out that it would be better if they discover a way of automating the wrapper generation and the “construction of wrapper classes” [10]. Here WG came, where it focuses on generating wrapper automatically. After that, more and more wrapper induction systems have been *produced for information extraction purposes.*

Wrapper can also be designed by hand using programming language. The golden rule of IE is, to know the structure of the Web page. The more structured the Web page is, the easier to inspect its elements and then exploit the content itself. With today's technologies, many open source software and libraries can be used to fasten the speed of creating wrapper. See Figure 2.1 for chronological history of IE and WG.

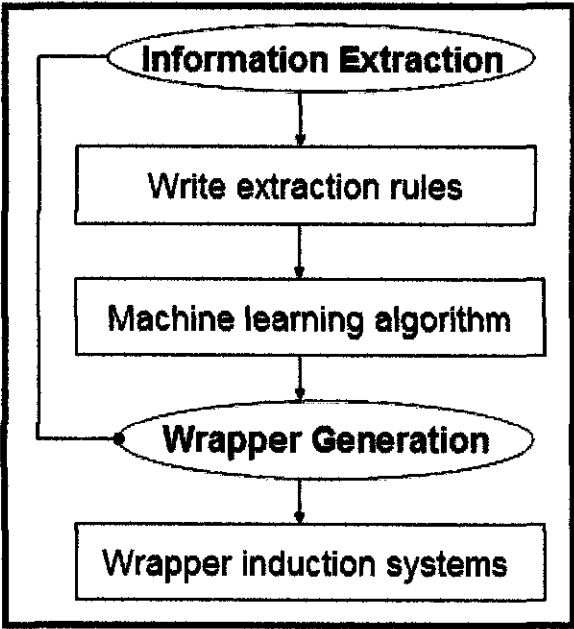


Figure 2.1: Chronological history of IE and WG

2.4 Recent IE Projects

According to Cosulschi M. (2004), various tools of information extraction have been proposed in the literature and developed for the purpose of performing Information Extraction (IE) [1] [2] [9]. In this section, a brief overview will be given on some recent projects that have been successfully developed over the last decade.

2.4.1 LiXto

LiXto [10] [11] was a project started by Gottlob, and it is a method for visually extracting HTML/XML wrappers under the supervision of a human designer. It allows a wrapper to view the information extraction patterns on the Web pages. It uses -like programming language called Elog to represent extraction knowledge internally [12]. Among the most interesting features of LiXto is the ability to access Web data even if protected by means of a username/password authentication mechanism, if the user provides them. Finally, the extraction process can be scheduled in order to be repeated at fixed times [7]. Figure 2.2 as shown on the next page is the architecture of LiXto.

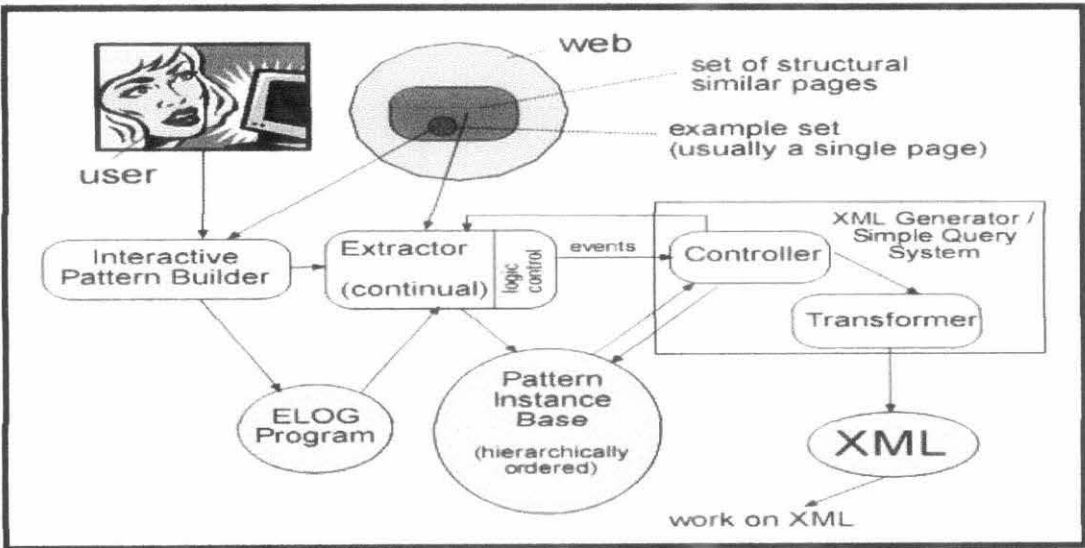


Figure 2.2: The architecture of LiXto, from [11]

2.4.2 Fetch Agent Platform

Fetch Agent Platform [13] is an example of commercial information extraction tool. It has two major components, which are AgentBuilder and AgentRunner, whereby it provides a visual environment that allows a user to construct web agents, while another one, automatically performs the tasks specified by the agent as well as produces structured data. The extraction rules are based on landmarks (groups of consecutive tokens) that enable a software agent to locate the start and end of fields within a page [7]. Figure 2.3 as shown on the next page demonstrates the architecture of Fetch Agent Platform.

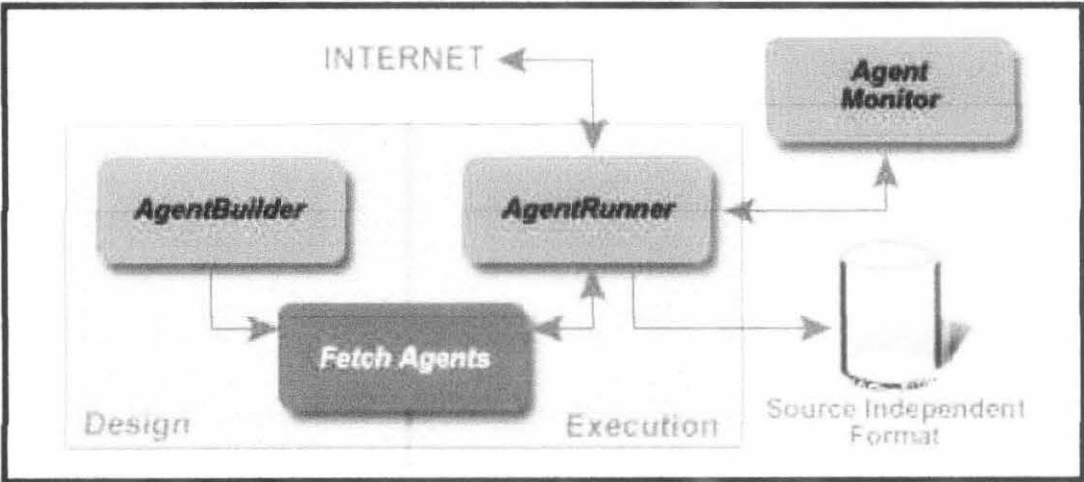


Figure 2.3: The architecture of Fetch Agent Platform, from [13]

2.4.3 RoadRunner

RoadRunner [7, 10, 11, 12, 13, 14, 23, 24, 25]

It is a research project on information extraction. The goal is to develop fully automatic techniques to perform information extraction from large Web sites. It needs multiple input Web pages of the same template to work and then generates a schema during an iterative process. This schema is used as a starting point for the inference of a grammar which is capable to recognize the instances of attributes. The sample pages are taken as the wrapper. The extraction procedure is based on an algorithm that compares the tag structure of the set of sample pages and produces regular expressions able to handle structural differences found in the set of sample pages. This procedure will result to generalized wrapper. A peculiar feature of RoadRunner is that this procedure is completely automatic and no user intervention is required. It also has contributed to bridge the gap between wrapper induction and traditional grammar inference techniques.

2.5 Email Forensic Analysis

Today, cyber crime has rapidly increased from time to time. Email which is the medium of communication has been a victim of uncontrolled illegitimate activities. Email spamming, phishing, cyber bullying, child pornography, sexual harassment are some common mediated cyber crimes. Apparently, no adequate, efficient automated tools and techniques that are available for securing email systems in a timely fashion. Thus, in the context of email forensic analysis, email information extraction becomes vital in order to examine suspected email accounts. It is critical to gather clues and evidences before prosecuting criminals in a court of law.

CHAPTER 3

METHODOLOGY

3. METHODOLOGY

3.1 Waterfall Methodology

In this project, waterfall methodology is chosen for the system development life cycle process. As shown in Figure 3.1 below, there are six phases altogether that will have to take place on developing this system. Even though it looks linear and sequential, but it still allow continuous improvement as it can go back to the previous phase to refine and correct anything whenever and wherever necessary. The big arrow shown in the figure means it can go back from one phase to other phase regardless where those phases reside are and it does not necessarily to go by order.

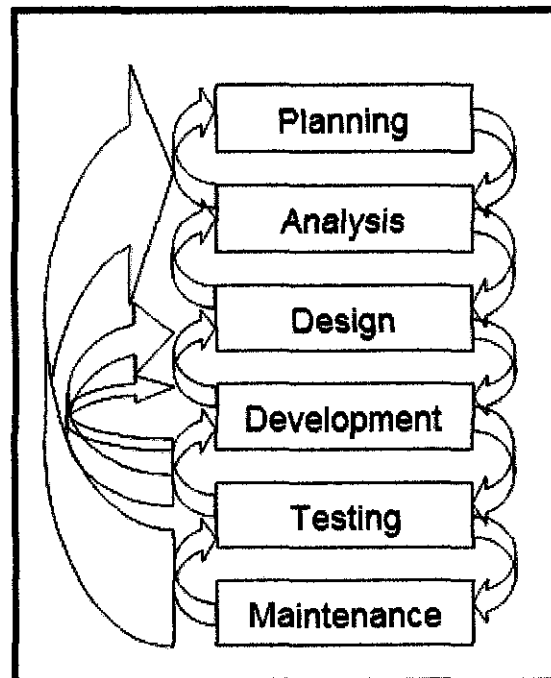


Figure 3.1: Waterfall methodology

3.1.1 Planning Phase

Planning phase is an important stage, whereby fundamental process of understanding of why the project should be carried on and its aims are conducted. During project initiation, several discussions on project feasibility have been done with project supervisor.

In order to work out the proposed project, thorough researches on Information Extraction (IE) shall be conducted as the pre-requisite, in which the research should give better overview about the project and knowledge on how to go about designing and developing it later. Research on email forensic with regards to cyber crime investigation appears to be crucial as it can be an added advantage to the project itself and might be a good contribution to the society.

To ensure the project goes smoothly within its scope in timely manner, it should have Gantt chart and key milestone as guidance. See *Appendix A* and *Appendix B* for Gantt chart and full key milestone, respectively.

3.1.2 Analysis Phase

Analysis phase shall be done by doing research on IE and Wrapper Generation (WG) areas as well as email forensic analysis, so that information and knowledge can be gathered. Other than that, research on development tools and environment should be conducted in the hope to discover available technologies that might fit and advantageous to the system at the end of the project. Selection of right tools and technique may lessen or even avoid the probability of project to fail.

3.1.3 Design Phase

The proposed framework should be created during design stage. It shall help to gain insights on how the system works and the workflow of project development as well as system function requirement and specification. The input and output of the system should be specified to ensure the project focus goes within its limit of scope and time.

3.1.4 Development Phase

The development of the system has been conducted during development stage. Here it should make use of the framework predefined earlier as the guideline to build the system.

Software installation and configuration should be done properly with its accompanying third-party software before start developing the system so that the environment settings fit to each other. All the libraries used should be located in the right directory of the development workspace to make sure it provides help as it supposedly to. Installing a server also is crucial since the system needs a servlet engine to channel out the delivery of output. All the development tools, platform, libraries and server used are elaborated in details in Section 3.2 of this chapter.

3.1.5 Testing and Maintenance Phase

Upon finishing the development process, several testing have to be done to ensure that the system works and performs perfectly as it expected. Any errors or bugs should be fixed after reviewing back the system requirements and the design.

Maintenance phase shall be conducted to maintain the system functionality as the input structure of Web mail used is expected to change in the future. Besides, any enhancement to the system can be conducted to create added values.

3.2 Development Tools

The main tool and platform used to develop the system is Eclipse version 3.3 and Java TM Standard Edition Software Development Kit version 6 (Java TM SE SDK 6) respectively. The Java TM SE SDK 6 platform is chosen as it allows this system to be written in Java technology. It contains the Java Runtime Environment (JRE), set of API classes, Java compiler and additional files.

Other than that, third-party software used in developing the system are JavaMail and JavaBeans Activation Framework (JAF). JavaMail is a standard Java extension which has javax.* package structure. Meanwhile JAF is used for handling Multipurpose Internet Mail Extensions (MIME) types since usually all Internet email is transmitted in MIME format, allowing messages to have a tree structure.

The servers involved are Apache Tomcat Server version 6 which is one of the most popular servlet engines and Apache Derby Network Server for the database. Derby database is chosen as the database because it comes as a package together with Eclipse in which it is easier for database set up and settings.

The programming languages used in the development process are Java, Java Server Pages (JSP), and Structured Query Language (SQL). In order to retrieve emails from mail account, Internet Message Access Protocol (IMAP) is chosen as the mechanism since it is one of the two most prevalent Internet standard protocols beside Post Office Protocol (POP) and proven to be more stable overall. It offers two-way communication between Web GMail and email client and also provides a better method to access mail account from multiple devices.

CHAPTER 4

RESULTS AND DISCUSSION

4. RESULTS AND DISCUSSION

4.1 Proposed Framework

Throughout this section, it discusses the proposed framework that is used for developing the system. Figure 4.1 as shown on the next page portrayed the proposed framework. Firstly, the system should be able to access the Web mail by passing over HTTP and provides authentication in order to gain access on particular mail account by providing several information such as mail server, username and password.

After take control on that email account, here the information extraction process goes and users may select to extract the whole emails on the account or certain folder such as Inbox, Drafts, Sent Mail or others according to their preferences and purposes. This system proposes two mediums of output presentation, either in text file or being stored in a database. Hence, it again depends on the users according to their preferences and the purpose of using the system.

In this project, three different targets of users are identified where they come from different backgrounds so do their needs. First, users who just want to easily get the new incoming emails without regularly logging in to the mail account and check on it one by one. Second, staffs in a company with the tasks of sending promotion or newsletter via email to their potential customers or clients and to keep track the status of delivery of each email. Third, forensic investigator may perform post-processing activities in a multi-staged email forensic analysis like email mining, email statistic analysis, email authorship attribution and more on the extracted information of emails that have been stored in the database to discover any unusual pattern or behavior.

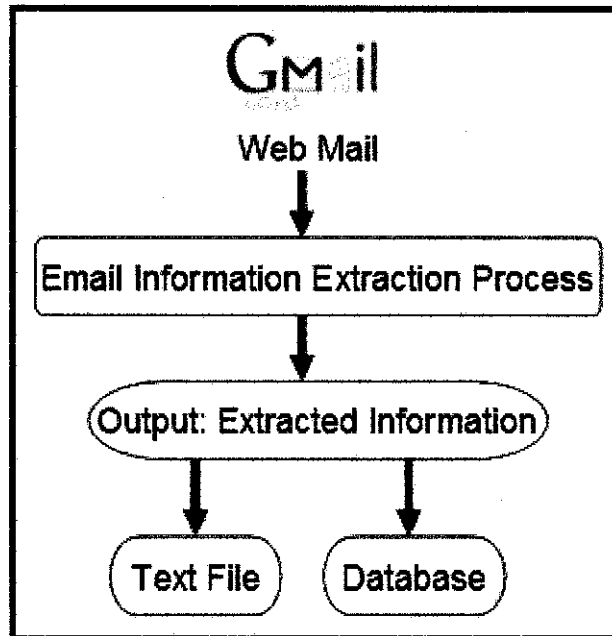


Figure 4.1: The proposed framework

4.2 Process Flow of Web Mail Information Extraction

Web mail information extraction process begins by establishing connection to the mail server provided specific details which are the mail server name, username and password in order to gain access of particular mail account. Figure 4.2 shows the process flow of web mail information extraction. It starts with establishing connection to the mail server by providing details of mail server name, username and password. After that, based on folder selection by users, the system will extract emails from that particular folder. If the user chooses to extract the whole mail account, extraction will be conducted to all folders inside the mail account. If email has been detected in a particular folder, the process of email extraction will take place and will end with displaying the output of extracted information. Once no email is to be extracted anymore, the execution process will end automatically.

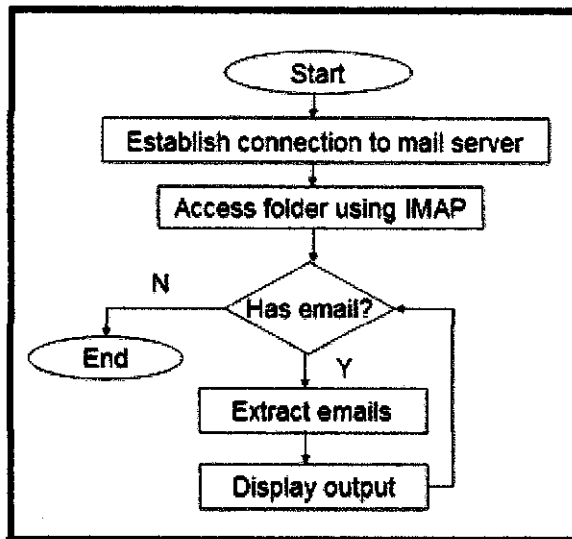


Figure 4.2: Process flow of web mail information extraction

In *Extract emails* function, there are several operations have to take place in order to retrieve particular information in the mail folder. Figure 4.3 displays the process of operations inside *Extract emails* function. Each and every email is to be set with unique number to differentiate one email from another. After that, relevant information from email will be retrieved accordingly which start from the sender, recipient, subject and the last is the content itself. Upon email's information retrieval has completed, the particular email will be marked as "read", "unread" or "delete" based on users' preferences. If the user did not select any, therefore as the "Mark as" field is set to non-mandatory class, the email will then be marked as read by default.

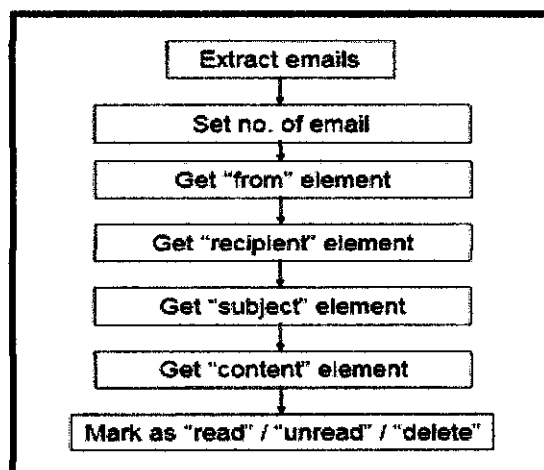


Figure 4.3: Details of *Extract emails* function

Figure 4.4 as shown below displays the Java code to connect to the mail server by providing required information which are the mail server name, server port number, username and password. Server port number is left by default number which is -1.

```
Properties props = System.getProperties();  
Session session = Session.getDefaultInstance(props, null);  
  
store = session.getStore("imaps");  
store.connect(server, -1, user, password);
```

Figure 4.4: Details needed to connect to mail server

After being granted access to the mail server and before performing information extraction process on the emails, specifying the folder is a necessity. The folders available for information extraction at this stage of system development are All Mail, Inbox, Drafts, Starred, Sent Mail, Spam and Trash. Figure 4.5 displays the simple source code of this process. For example from the figure below, the user prompts the system to extract information from Inbox folder. The system will search through the mail account for Inbox folder, if not found, it displays "No Inbox".

```
folder = folder.getFolder("Inbox");  
if (folder == null)  
    throw new Exception("No Inbox");
```

Figure 4.5: To get folder for extraction

Next is the information extraction process. Figure 4.6 is the code to keep track the sequence of emails and put number on it to differentiate one email with another. It is safe to ensure the users did not get confused. The message number is sequentially accorded to how many emails in a particular folder that has been chosen have or how many emails in that mail account has if it runs for the whole mail account. For example, if the user chooses to extract emails from Inbox folder only, then, the message number starts with one and the end number is equal to total emails in that Inbox. Same goes if the user chooses to extract particular folder. If the user wants to

extract the whole mail account, the number of the last message will equal to total count of emails in that mail account.

```
// -- Get the message number and add as an element --
SourceElement sourceElement
=new SourceElement("\n"+"No."+msgNum+" ", "");
elements.addElement(sourceElement);
```

Figure 4.6: Message number

Vital elements to be extracted are the sender of the email, recipient, subject and content. For this project, extracted elements are limited to only those four. Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10 shows the source code of to retrieve each element. Using JavaMail and JavaBeans Activation Framework (JAF) library, it should be easy to retrieve the elements of email because the library provides a lot of functions like `getFrom()`, `getAllRecipients()`, `getSubject()` and more. Each of them serve its unique function and understandable as it acts as self-explanatory.

```
// -- Get the sender and add as elements --
Address[] sender=message.getFrom();
for (int r=0; r<sender.length; r++)
{
    String thisSender=sender[r].toString();
    sourceElement=new SourceElement("Sender:"
    ,thisSender);
    elements.addElement(sourceElement);
}
```

Figure 4.7: Get the sender's email address

```
// -- Get the recipients and add as elements --
Address[] recipient=message.getAllRecipients();
for (int r=0; r<recipient.length; r++)
{
    String thisRecipient=recipient[r].toString();
    sourceElement=new SourceElement("Recipient(s):"
    ,thisRecipient);
    elements.addElement(sourceElement);
}
```

Figure 4.8: Get the recipient's email address

```
// -- Get the subject and add as an element --
String subject=message.getSubject();
SourceElement
    =new SourceElement("Subject:",subject);
elements.addElement(sourceElement);
```

Figure 4.9: Get the subject element

In Figure 4.10, `BufferedReader` is used to read through the content of email message and parse it as text. Every line in email content is read to retrieve the value.

After retrieving all values from each element, `addElement()` method is used to add those element to the object that will be presented as an output later on. In this example, the variable name for that object is *elements*. *elements* carries all values assigned to it till the end of the email information extraction process.

```
// -- Parse content as text --
BufferedReader reader
    =new BufferedReader(new InputStreamReader(is));
int lineNum=0;
String thisLine=reader.readLine();
while (thisLine!=null)
{
    if(lineNum == 3)
    {
        sourceElement=new SourceElement("Message:",thisLine);
        elements.addElement(sourceElement);
    }
    if (lineNum == 4 || lineNum == 5 || lineNum == 6)
    {
        sourceElement=new SourceElement("",thisLine);
        elements.addElement(sourceElement);
    }

    lineNum++;
    thisLine=reader.readLine();
}
```

Figure 4.10: Get the content of the email

Right after getting all the elements of email, connection to database is needed. In this system, Derby database is used as it comes together in a package with Eclipse platform. Thus, Figure 4.11 as shown below displays how to set database properties in Eclipse to connect the Java application with Derby database. All details as follows should be specified to ensure the connection is established between respective parties. The database name created is named as *sample* as shown in Figure 4.11.



Figure 4.11: Connection to Derby database

To store the extracted information of emails into the database, Figure 4.12 shows how it is done. Thus, the connection to the database needs to be established first before executing any SQL queries in the Web mail information extraction process. The four elements that have been extracted earlier which are the sender, recipient, subject and content will be stored in the database. Therefore there are four columns in the database that need to be created to store each extracted element. The columns are Sender, Recipient, Subject and Content.

In the same figure, the last code portrays that the emails that have been executed by Web mail information extraction process will then be flagged as seen, which means it has been read. Being marked as "Read" is default if the user did not specify the *Mark as* field. This is just to mark the message and distinguish one email with another new email.

```
for(String temp:mails)
    System.out.println(temp);
connecttoDerby();
message.setFlag(Flags.Flag.SEEN, true);
```

Figure 4.12: Connection to database and email is been flagged as seen

4.3 System Interface

In this section, system interface is discussed as follows. Figure 4.13 shows the main page of the system. Users have to key in all required information in order to use the system. They need to enter his GMail's account information which are username and password.



Figure 4.13: Main page of the system

The *Extract To Text File* button at the bottom of the system has the function of performing Web mail information extraction and save the output in a text file. The middle button, *Extract Into Database*, stores the output of extracted information of emails into a database instead. Reset button is used to clear all input fields at once. Users have to be clear with their own preferences and understand their purposes of using the system otherwise they could not use this system's functionalities to the fullest as it supposed to provide to users.

Figure 4.14 and Figure 4.15 display how the dropdown menu appears for *Folder* and *Mark as* field, respectively. The dropdown gives users the chance to select their preferences accordingly. Users may select their preferred folder to be extracted or just extract the whole folders which represents as *All Mail*. Available folders for extraction at this point of time are All Mail, Inbox, Starred, Drafts, Sent Mail, Spam and Trash.

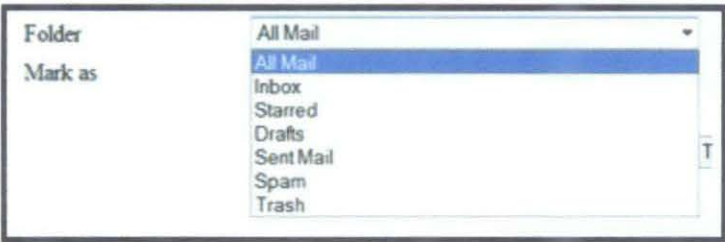


Figure 4.14: Dropdown menu for *Folder*

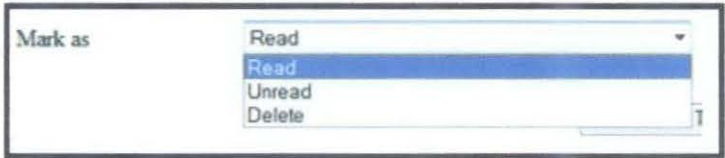


Figure 4.15: Dropdown menu for *Mark as*

4.4 Testing Outputs

As mentioned several times before, the final output of this system in which to be specific, the extracted email information will be saved in two kinds of way, either in a text file or in a database. Figure 4.16 is the original emails in a particular GMail account namely bahariah.latan@gmail.com. For the time being, there are currently five unread email messages in the Inbox folder.

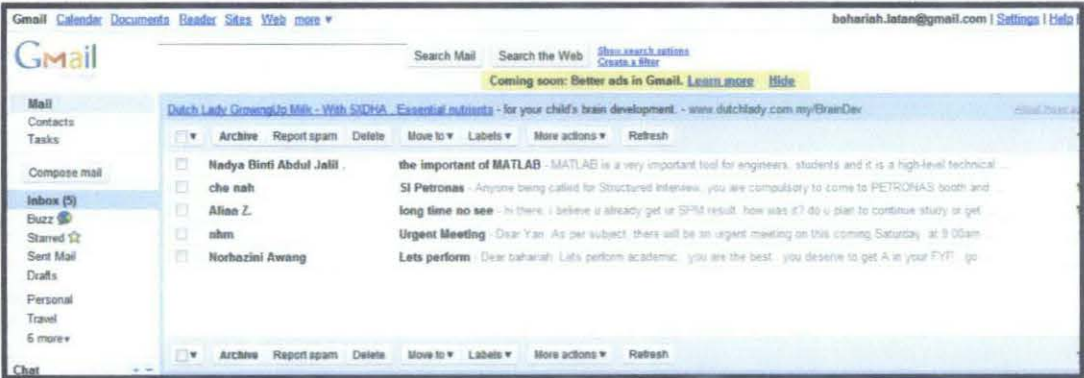


Figure 4.16: Original emails in bahariah.latan@gmail.com

In Figure 4.17 and Figure 4.18, it shows how it looks like the output of extracted elements of those emails in bahariah.latan@gmail.com that will be saved in a text file while Figure 4.19 shows it on the other way round, which is in a database. Figure 4.17, Figure 4.18 and Figure 4.19 contain exactly the same extracted information from each email but they are being presented in different formats.

```
No.0
Subject: Lets perform
Sender: Norhazini Awang <zini.me88@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: Dear bahariah,

Lets perform academic.. you are the best.. you deserve to get A in your
FYP.. go. go.go bahariah can do it..

--20cf3071ca2ccf654904a061392d

No.1
Subject: Urgent Meeting
Sender: nhm <nham146@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: Dear Yan,

As per subject, there will be an urgent meeting on this coming Saturday, at
9.00am. Please reply your availability to come to the meeting.

Thank you.

No.2
Subject: long time no see
Sender: "Aliaa Z." <zii.memories@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: hi there, i believe u already get ur SPM result. how was it? do u plan to
continue study or get into career world?

--
DAYANG ALIAA ZAWANI BT AWANG BAKAR
Bachelor (Hons) Information and Communication Technology
```

Figure 4.17: Final output to be saved in text file

```

No.2
Subject: long time no see
Sender: "Aliaa Z." <xii.memories@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: hi there, i believe u already get ur SPM result. how was it? do u plan to
continue study or get into career world?

--
DAYANG ALIAA ZAWANI BT AWANG BAKAR
Bachelor (Hons) Information and Communication Technology

No.3
Subject: SI Petronas
Sender: che nah <student.life.32@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: Anyone being called for Structured Interview, you are compulsory to come to
PETRONAS booth and see Tengku Nasruddin or any PETRONAS Officers to fill up
a form.

--bcaacc53f908df0d8b004a0890b33
Content-Type: text/html; charset=ISO-8859-1

No.4
Subject: the important of MATLAB
Sender: Nadya Binti Abdul Jalil Nadya <ict.nadya@gmail.com>
Recipient(s): bahariah.latan@gmail.com
Message: MATLAB is a very important tool for engineers, students and it is a
high-level technical computing language and interactive environment for
algorithm development, data visualization, data analysis etc. And we have
collected some of the good topics and project reports for students doing
project on Matlab. You can also add your own Matlab project topic and report
here through your comments.

```

Figure 4.18: Final output to be saved in text file (cont.)

SQL Results				
Execution Plan				
Console				
Servers				
Ty Status Result1				
A				
SENDER				
RECIPIENT				
SUBJECT				
CONTENT				
1	Norhazini Awang <zini.meb8@gmail.com>	bahariah.latan@gmail.com	Lets perform	Dear bahariah, Lets perform academic.. you are the best.. ..
2	nhm <nhm146@gmail.com>	bahariah.latan@gmail.com	Urgent Meeting	Dear Yan, As per subject, there will be an urgent meeting ..
3	"Aliaa Z." <xii.memories@gmail.com>	bahariah.latan@gmail.com	long time no see	hi there, i believe u already get ur SPM result. how was it? ..
4	che nah <student.life.32@gmail.com>	bahariah.latan@gmail.com	SI Petronas	Anyone being called for Structured Interview, you are co...
5	Nadya Binti Abdul Jalil Nadya <ict.nadya@gmail.com>	bahariah.latan@gmail.com	the important of MATLAB	MATLAB is a very important tool for engineers, students a..

Figure 4.19: Final output to be stored in database

4.5 Test Cases

Table 4.1 as displayed below shows the result of test cases for the system testing. Several testing have been run to ensure the accuracy and integrity of the extracted information during Web mail information extraction process. The test cases are developed with the aim to provide testing results based on the functionalities of current system. In this system, email account of bahariah.latan@gmail.com and yan.bahariah@gmail.com are used to serve for testing purposes.

No. of emails [Ran as Java application]							
GMail Account	All Mail	Inbox	Starred	Drafts	Sent Mail	Spam	Trash
bahariah.latan	22	5	5	0	2	1	1
yan.bahariah	21853	2253	10	10	1526	7	34
Total emails correctly extracted and marked as read							
GMail Account	All Mail	Inbox	Starred	Drafts	Sent Mail	Spam	Trash
bahariah.latan	24	5	5	0	3	1	1
yan.bahariah	22013	2807	10	10	1621	7	34
Total emails correctly extracted and marked as deleted, applied to Spam & Trash only							
GMail Account	All Mail	Inbox	Starred	Drafts	Sent Mail	Spam	Trash
bahariah.latan	N/A	N/A	N/A	N/A	N/A	1	1
yan.bahariah	N/A	N/A	N/A	N/A	N/A	7	34

Table 4.1: Test cases

Total all emails in All Mail folder have huge different from the sum of emails in other folders like Inbox, Drafts, Sent Mail, Starred, Spam and Trash, because the user of bahariah.latan@gmail.com is implementing customized folders. Other than that, total actual emails in a particular folder like All Mail, Inbox and Sent Mail is differ from total extracted emails because GMail is implementing one subject of an email with many replies. Therefore this system did not recognize the emails based on per subject, but based on how many replies (email messages) per subject altogether. Hence, the actual total emails in particular folder is lesser than total extracted emails.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5. CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

With the growth of the amount of online information, the availability of Information Extraction (IE) systems has become an important necessity. Thus, this project aims to perform Web mail information extraction, transform emails into a clean format saved in text file and store extracted information from emails into database. Extracted information is stored in two different ways to satisfy users from different backgrounds.

Therefore, this project has achieved its main goal as mentioned above by successfully extracting useful and relevant emails' information such as the sender, recipient, subject and content from Google Mail (GMail) using Java. It is proven based on the testing outputs and test cases that have been presented earlier.

Extracted information saved in a text file helps users personally in which they need not to log in all the time to their mail accounts just to check if there are new incoming emails. They also do not need to click every email in order to see the content as with the use of this system, users can view many emails in a glance and just scrolling down the text file if the emails are many. This method will better facilitate users who value their time and energy the most.

As the extracted information of Web mail is to be stored in a database, any keyword searching activities like SQL query can be conducted. For a company which tends to send promotion or newsletter to their target customers or clients in a huge amount at a time, they can implement the same approach of this system to extract the email addresses. This will simplify their tasks as they can keep track of to whom the promotion emails already been sent to because the flag in the database will be changed automatically once any execution of IE was successfully completed.

At the other side, this system also can help forensic investigators to deal with cyber crimes by performing email information extraction and then proceed with the next

stage of multi-staged email forensic analysis such as email mining, email statistic analysis, email authorship attribution and more in order to identify unforeseen pattern and behavior.

5.2 RECOMMENDATIONS ON FUTURE WORKS

This project may be further enhanced by extracting other useful and relevant information in emails such as the sent date, cc, bcc and capturing more than one recipient. To encounter with the constraint of extracting irrespective the emails are read or unread, the system should be able to extract only unread emails to avoid redundancy and to better facilitate the users. In order to make this system useful and flexible to be used by many different target groups of users, it is advisable to provide the functionality of extracting emails from different mail accounts such as Yahoo! Mail and Hotmail.

Moreover, for the sake of enhancement values, this project is recommended to further expand by upgrading the system's capabilities by not only doing email extraction but may proceed to the next level of multi-staged email forensic analysis in order to help forensic experts deal with cyber crimes by moving forward in doing email mining, statistical analysis and email authorship attribution.

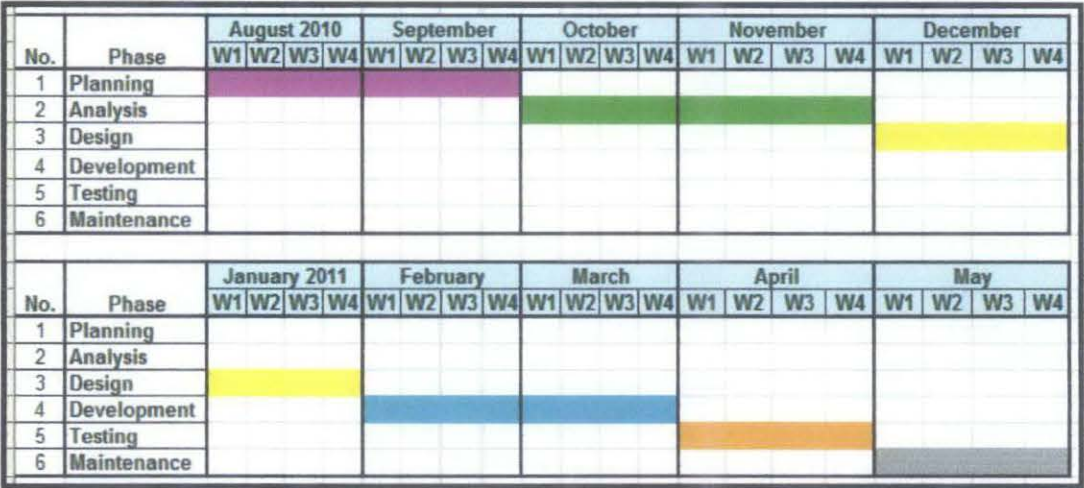
REFERENCES

- [1] Eikvil L. 1999. Information Extraction from World Wide Web - A Survey - .Technical Report 945, Norwegian Computing Center, Norway.
- [2] Muslea I. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. University of Southern California, USA
- [3] Sarawagi S. 2002. Automation in Information Extraction and Integration, Tutorial of VLDB. In Proceedings of the 28th International Conference on Very Large Data Bases, 2002.
- [4] Chia-Hui Chang, Kayed M., Girgis M.R., Shaalan K. (2006) A Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering, TKDE-0475-1104.R3.
- [5] Hsu, C., and Dung, M. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *J. of Information Systems* 23(8):521–538, 1998.
- [6] Laender A.H.F. , Ribeiro-Neto B.A., da Silva A.S., Teixeira J.S. (2002) A Brief Survey of Web Data Extraction Tools. SIGMOD Records 31(2) 2002.
- [7] Fiumara G. Automated Information Extraction from Web Sources: a Survey. Universit'a degli Studi, Messina, Italy.
- [8] Flesca S., Manco G., Masciari E., Rende E. and Tagarelli A. (2004) Web wrapper induction: a brief survey. *AI Communications* 17 (2004) 57 – 61.
- [9] Cosulschi M., Constantinescu N. and Gabroveau M. 2004. Classification and comparison of information structures from a web page. *Annals of University of Craiova*.
- [10] Baumgartner R., Flesca S., Gottlob G. (2001) Visual Web Information Extraction with Lixto. In Proc. of VLDB, 2001.
- [11] Baumgartner R., Flesca S., Gottlob G. (2002) Declarative Information Extraction, Web Crawling and Recursive Wrapping with Lixto. In Proc. of LPNMR, 2002.
- [12] Baumgartner R., Flesca S., Gottlob G. (2002) The Elog Web Extraction Language.
- [13] Fetch Technologies. 10 March 2011, <<http://www.fetch.com/>>.
- [14] Mecca G., Grumbach S. (1999) In search of the lost schema. ICDT(1999).

- [15] Crescenzi V., Mecca G., Merialdo P. (2001) RoadRunner: Towards Automatic Data Extraction from Large Web Sites. VLDB(2001).
- [16] Crescenzi V., Mecca G., Merialdo P. (2001) The RoadRunner Project: Towards Automatic Extraction of Web Data. ATEM (2001).
- [17] Crescenzi V., Mecca G., Merialdo P. (2001) Automatic Web Information Extraction in the RoadRunner System. DASWIS (2001).
- [18] Crescenzi V., Mecca G., Merialdo P. (2002) Wrapper Oriented Classification of Web Pages. ACM SAC (2002).
- [19] Loton T. 2002, *Web Content Mining with Java*, UK, John Wiley & Sons.
- [20] V. Crescenzi, G. Mecca, P. Merialdo, RoadRunner: Automatic Data Extraction from Data-Intensive Web Sites, International Conference on Management of Data and Symposium on Principles of Database Systems (SIGMOD02), 2002.
- [21] Zheng S., Wu D., Song R. 2007. Joint Optimization of Wrapper Generation and Template Detection. California, USA.
- [22] Chia-Hui Chang, Shao-Chen Lui, Yen-Chin Wu. Semi-Structured Information Extraction Applying Automatic Pattern Discovery. National Central University, Taiwan.
- [23] Giansalvatore Mecca. 2002. Extracting Data From Web Sources. Università della Basilicata.
- [24] Bing Liu. 2005. Web Content Mining (Tutorial). Department of Computer Science University of Illinois at Chicago (UIC).
- [25] Soderland S. Learning Information Extraction Rules for Semistructured and Free Text in Machine Learning.
- [26] Rachid Hadjidj, Mourad Debbabi*, Hakim Lounis, Farkhund Iqbal, Adam Szporer, Djamel Benredjem. 2009. Towards an integrated e-mail forensic analysis framework. Concordia University, Canada.
- [27] Rohas Nagpal. 2008. Evolution of Cyber Crimes. Asian School of Cyber Laws. India.

APPENDICES

Appendix A: Gantt chart



Appendix B: Key milestone for semester 1 and semester 2

