Interactive File Transformer

Nurul Mirjant Binti Ismail
MSc Advanced Computer Science
2013/2014

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

Abstract

Transformation or conversion between different digital text file formats involves repetitive tasks. Industry standard applications only offer solution between popular file formats. A user needs a flexible application to create their own customisable transformation rather than juggling between programmes. Even switching between different converter applications does not guarantee will produce exactly intended ouput. It still involes a lot of tedious tasks.

Regular expressions can be applied to create transformation by matching structure within a source file and replacing it with desired substitution. However, we do not build transformer or converter applications using regular expressions because it is difficult. The idea behind this project is to connect the power of configurability with regular expressions and reusabilty of customisable transformation to enhance the efficiency of repetitive text editing problem. Apart from that, it allows easy navigation to see changes between transformations and act as a platform to promote training for understanding regular expressions to beginners.

The following report supports the successful proof of concept of the Interactive File Transformer.

Contents

Section	Page
1. Introduction	1
1.1 Overview	1
1.2 Motivation	2
1.3 Project Aim	2
1.4 Project Objectives	2
1.5 Minimum Requirements	2
1.6 Deliverables	3
1.7 Project Schedule	3
1.8 Methodology	4
1.8.1 Research Methodology	4
1.8.2 Development Methodology	4
1.8.3 Evaluation Methodology	5
1.8.4 Methodolgy Model	5
1.9 Report Structure	5
2. Literature Review	7
2.1 Introduction	7
2.2 Text editing and human computer interaction	7
2.3 Repetitive Text Editing	8
2.3.1 Keyboard Macros	8
2.3.2 Programming by Demonstration	9
2.3.3 LAPIS	10
2.4 Regular Expressions	10
2.4.1 Regex Engines/ Libraries	11
2.4.2 PCRE and Perl	11
2.4.3 What regular expression can match	12
3. Requirement Analysis	13
3.1 Introduction	13
3.2 Functional Requirements	13
3.3 Non functional requirements	13

3.4 User groups 14
3.5 Feasibility measure 14
3.5.1 Goal of the development 14
3.5.2 Technical Feasibility 15
3.6 Advantages of Open Source Software 15
3.7 Advantages of web-based application 16
3.7.1 Easier installation and maintenance 16
3.7.2 Cost effective development 16
3.7.3 Accessible for a range of devices 16
3.8 Additional development tools 16
3.9 Development machine 17
3.10 IFT vs. text editors and word processors 17

4. System Design 19
4.1 Introduction 19
4.2 Inspiration of system design 19
4.3 Web-Based System Architecture 20
4.4 Functionality Design 20
4.5 Database Design 21
4.5.1 Entity Relationship Schema 21
4.5.2 Data dictionary 21

5. Implementation 22
5.1 PCRE 22
5.2 Basic Operations 22
5.2.1 Match 22
5.2.2 Match all 23
5.2.3 Search and replace 23
5.2.4 Split 24
5.3 System flow 24
5.3.1 Create task 24
5.3.2 Creating a predefined sequence 26
5.4 Reusabilty 28
5.5 Customising Transformation 31

5.6 View details and results of every transformation 32
5.7 Rearrange the order of sequences 32
5.8 Limitation 33

6. Evaluation 34
6.1 Introduction 34
6.2 Walk-through 34
6.2.1 Methodology of evaluation 34
6.2.2 Observation 35
6.3 Reusability 35
6.4 Reusability Performance 37
6.4.1 Methodology 40
6.4.2 Assumption 40
6.4.3 Results and observation 41
6.5 Training Tool 41

7. Conclusion and Future Work 42
7.1 Project Reflection 42

Appendix
A: Personal Reflection
B: External Materials
C: Ethical Issues
D: Gantt Chart
E: Use Case
F: Walk-through
G: Screen
H: Reusability Performance

Chapter 1
Introduction

1.1 Overview

Digital text files and digital documents usually contain structured data. Structured data can be described as information that resides in fixed fields[1], displayed in titled columns and rows[2], and organised in a manageable way[3]. An address book, a bibliography, a calendar, a table of statistics, a list of events, a music playlist, an email file and a LaTex document are the examples of such files. Structured data would be more informative and meaningful if it can be manipulated by editing, rearranging the order of the fields, changing the structures to finally form a desired meaningful output.

Given a situation when a person wants to convert his address book into a nicely formatted table. He then would like to put the information on a web-page. This task will involve converting the address book to an HTML table format. In a general scenario, usually a user has at least two approaches for the conversion process between the original source to the final output, either by switching between different converter applications or to apply relatively mundane direct manipulation. Both options would be tedious, repetitive, boring, prone to errors and time consuming. "Regular structure" characteristics of a file gives an advantage for automating repetitive operations. An idea behind this is by using regular expressions to match and substitute texts and structures.

It is undoubtedly numerous existing text processing tools ranging from text editors, word processors, parsers, converters or even terminal text editors offer the functionality for text substitution. Many general tasks such as converting CSV to other formats and vice versa, XML to JSON and vice versa, MS Words to LaTeX can be automated by those tools, yet the one that is very specific to user requirements still needs to be automated by end users themselves because the flexibility to create abstraction usually is not provided by the tools.

1.2 Motivation

File transformation or file conversion in this project refers to the process of converting digital text files from one format to other formats. In conventional method, file conversion involves repetitive text editing between transformation stages. Existing applications offer general file conversions but they do not record the details of each transformation. Some applications record the history of changes but it is just a matter of history that do not contribute to reusabilty. Imagine the same task to be performed many times again in the future, a mechanism should be invented to simplify the process in just a few clicks, thus eliminate the repetitious procedure that is boring and time consuming. As a training tool for non expert users, they can view the effects of each transformation and help to gain the understanding on the text processing using regular expressions.

1.3 Project Aim

The aim of the project is to develop a convenient tool which interactively do transformation including conversions and/ or manipulation of text, data and structured information from one format to other formats.

1.4 Project Objectives

The objectives of the project are to:-

- Build file transformation functionality which will enhance the limitation of existing similar tools.
- Develop a tool that can apply regular expressions to match text within given input data and apply a substitution to the matched data to create a separate output.
- Build interactive construction of file transforming sequences. Interactivity is focused on functionalities that could help novice user to navigate the application and for experts to save more time on data transformations.

1.5 Minimum Requirements

The minimum requirements for this project are:

- Develop a transformer tool that, when given a set of regular expressions and substitutions, able to produce a desired output.

- Develop a web based graphical user interface that allows user to interactively change regular expressions on an input dataset and view the output in separate window panes.
- Allow the users to rearrange the order of expressions and view the transformation at different stages.
- The tool should not be platform-specific and should perform similarly on every desktop architecture.
- To output a set of expressions to file, called a transformer, and to allow input from a
- transformer file.

1.6 Deliverables

- Project report detailing background reading, requirement analysis, design, implementation and testing and evaluation.
- An application called the Interactive File Transformer

1.7 Project Schedule

The project schedule presented below is a revised version after a few weeks the project starts. The original schedule (refer Appendix D) was initiated to plan the stages to be achieved. However, as the project is going on after drawing the design of the system flow and analysing the system requirements, it is found that each major occurring task is better to be breakdown. It would be clearer for the readers to understand the stages. Some stages may not identical to the time outlined in the original schedule but this revised version is more realistic to myself to follow.

	Week												
Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Weekly Meeting													
Background Research													
Aim, Objectives & Minimum Requirements													
Submit Interim Report													
Collect marked Interim Report													
Reflect upon Interim Comments													
Extend Current Literature													
Study Existing Tools													
Requirements Analysis of the system													
Blueprint of the system flow													
Installation of development tools													
Protoype of GUI and basic functions													
Database design													
Write up													
Study on Regular Expressions													
Coding and testing													
Progress meeting													
Reflect upon Progress Meeting Feedback													
Evaluation													
Final Submission													

Year 2014

Week 1:	09/06 – 14/06
Week 2:	15/06 – 21/06
Week 3:	22/06 – 28/06
Week 4:	29/06 – 05/07
Week 5:	06/07 – 12/07
Week 6:	13/07 – 19/07
Week 7:	20/07 – 26/07
Week 8:	27/07 – 02/08
Week 9:	03/08 – 09/08
Week 10:	10/08 – 16/08
Week 11:	17/08 – 23/08
Week 12:	24/08 – 30/08
Week 13:	31/08 – 04/09

Table 1.1: Revised project schedule

1.8 Methodology

The methodology is break down into 3 parts as follows:

1.8.1. Research Methodology

- Study the human interaction and behaviour towards text editing
- Understand current literature and various techniques to solve repetitive text editing problems
- Revise different regular expressions features and its limitation

1.8.2. Development Methodology

- Analyse the system requirements and tools to support the development
- Observe similar existing text processing tools and their limitations
- Installation of Apache Server on localhost, MySQL database, PHP compiler and Macromedia Dreamweaver on the development machine
- Study the syntax of regular expressions and familiarise with the usage
- Polish programming skills by visiting online tutorials

1.8.3. Evaluation Methodology

- The application is evaluated on common use cases and compared with a few similar tools.
- The reusabilty performance is evaluated on huge use cases data records.
- The aim for 'training tool' is evaluated on user groups.

1.8.4. Methodology Model

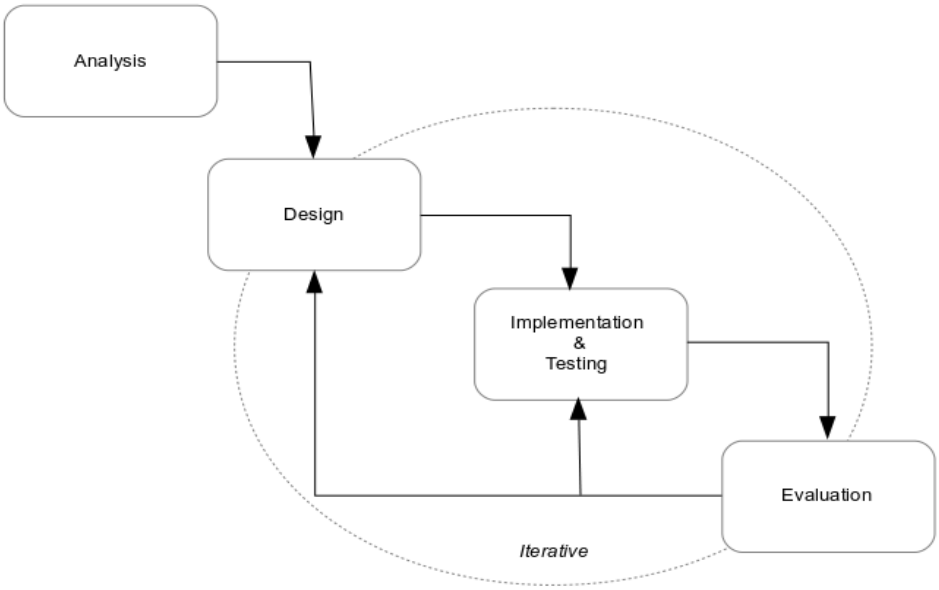


Figure 1.1: Modified Waterfall Model

This project adapts a Modified Waterfall Model. Pure Waterfall Model would only be suitable to projects without any failure between the stages. This model is flexible to allow changes in between the stages, taking account with testing results and feedbacks until producing a satisfied project as outlined by the requirements.

1.9 Report Structure

The remaining chapters of the report are organised as follows:-

- **Chapter 2: Literature Review**

  In this chapter, background research on current literature is presented to construct a clear understanding for the problem statement and visualise the direction towards

producing a successful solution.

- **Chapter 3: Requirement Analysis**

  In this chapter, requirements and feasibility studies are analysed to determine the scope of the development.

- **Chapter 4: System Design**

  This chapter explains the design for system architecture, system flow and functionality, interface and database model.

- **Chapter 5: Implementation and Testing**

  This chapter demonstrates the implementation of the project.

- **Chapter 6: Evaluation**

  This chapter demonstrates evaluation based on a few use cases and reusabilty performance. The result shows that IFT can substantially be more effective than conventional techniques for repetitive text editing that are provided by industry-standard application.

- **Chapter 7: Conclusion and future work**

  This chapter is the project reflection and explains how the minimum requirements and some possibilities for future work is outlined

Chapter 2

Literature Review

2.1 Introduction

This chapter outlines and presents information gathered from background study of the literature. This research was carried out in order to gain a deeper understanding of the problem and will help to draw the direction to design a proper solution.

There are 3 main areas of research took place:

- Human-computer interaction factor that gives an idea of users behaviour towards text editing
- Repetitive text editing problems and implemented techniques/ mechanisms to solve it
- Regular expressions, the engines/ libraries, and what it can match

2.2 Text editing and human computer interaction

Before discussing on related works in text editing, it is also important to know the user group and understand their behaviour towards the subject. Measuring interactivity, usability and performance is highly subjective when it involves interaction between a user and a program. With the limitation of the allocated project time, the developed solution is not expected to overcome the issues on this matter as the scope is not to focus on user interaction, but it will help to visualise common techniques as guidelines on how the evaluation should be carried out.

In terms of user performance, a demonstration [26] conducted to measure the time taken and the number of keystrokes pressed by the user to perform text editing found that to determine the principle components of task time distribution and defining “optimal” is not easy. Novice users spend time removing typing errors so the number of keystrokes will increase, while expert users may use keyboard shortcuts to perform a task. Some novice users are careful and do not cause many errors, while expert users consume more keystrokes switching between wrong shortcuts.

Many models were introduced to examine how people typically execute tasks on text editors. One of an initial model called GOMS (stands for Goals, Operators, Methods, and Selection rules), which was developed by Card, Moren and Newell. GOMS is a cognitive model to observe user's interaction with text editor. Based on the relevant literature [6][7][8][9] on this subject, the demonstration showed a clear difference between expert and novice users. Expert users were typically rapid and efficient in carrying out editing task, while novice users would not use many commands and did not have many switching techniques.

It still remain uncertain to measure the behaviour patterns through an observation. Behaviours observed under artificial situations sometimes are not parallel with natural daily editing environments. User might feel an implicit demand to perform the best editing behaviour in a recorded and timed observation. However, it is believed that eventually text editing skills will develop and and improve with many possible ways and proper training techniques.

2.3 Repetitive Text Editing

Text editing involves a lot of repetitive tasks. Church & Blackwell [12] quoted observation from Caroll & Rosson [10] that “many users prefer just to carry on with a repetitive or inefficient task rather than seek out better ways to do it”. As skills will develop from time to time during continuous usage, user should realise that they have a choice between using repetitively mundane manipulation techniques or putting an effort to devise a programming-like approaches or utilise any existing tools to simplify a task. Depending on the experience, creativity, knowledge level of a user and the complexity of the task itself, some existing tools can already be utilised to fulfil the generic process.

Many tools for automating text editing tasks such as text-processing languages like Perl or awk, keyboard macros, and even 'search and replace'. Each of them have their own strength and limitation.

2.3.1. Keyboard Macros

Many applications are equipped with keyboard macros – a utility to record the sequences of keystrokes and can be re-executed by another single operation to automate repetitive task

including text editing. There are many versions of it, but the main concept behind it remains the same. Keyboard macros support tail recursion where the last step in the macro recalls the macro. Fujishima [13] claimed that keyboard macro cannot solve many text editing problems and none of the tools are satisfactory. For more complicated task, user may proceed with writing a script in a text-processing language like Perl or awk.

2.3.2. Programming by Demonstration

Programming by Demonstration (PBD) or also known as Programming by Example (PBE) is a form of machine learning approach, where the computer will copy the actions by the users. The computer is initially set up with a model. The user will "train" the computer to perform some task until they are satisfied with the model. The system will interpret the demonstration and the task will be automated. Traditional programming generally requires learning and using programming languages or programming-like approaches. The idea of PBD is to make the process easier. There are two classes of PBD, action-based systems and and result-based systems [13].

Editing By Example (EBE) is an example of result-based PBD to automate regular expression replacement. It is easy to use once the user learns and understands the grammar of the language and write a complete programme after carefully analysing the sequences of the task to automate. For non-programmers, the requirements would be difficult if the sequences are longer and complicated.

Church and Blackwell [12] performed a demonstration of this method on repetitive editing of structured text measured by the length of time. It is proven effective and became far more efficient with this model. In certain situations, building a satisfied generalised model consumes more time if the user needs to feed all possible concrete examples although action-based systems can make use of richer information. Users are prone to making many errors, but this problem is considered severe because they will make many corrections. PBD is an advantage when it enables the non-programmer users to automate the tasks [13].

2.3.3. LAPIS

Developed by Miller and Myers [14] based on the idea of lightweight structure, LAPIS is one of the instances of PBD systems for text editing. As explained in [15], "Lightweight structure is the ability to recognize text structure automatically, using an extensible library of patterns and parsers. Structure can be detected in lots of ways: grammars (e.g. Java or HTML), regular expressions, even manual selections by the user. With lightweight structure, it doesn't matter how the structure was detected. You don't have to bend over backwards to represent something as a regular expression, or as a grammar, or as some other formalism. All that matters is the result: a *region set*, which is just a set of substrings in the text."

LAPIS was Miller's PhD thesis. It is a programmable web browser and text editor that demonstrates how lightweight structure can be useful. Its interesting and novel features are text constraints, selection guessing, simultaneous editing, outlier finding, structured text tools and browser shell [14][15]. Combination of these features create a powerful environment for automated text editing. LAPIS would be an evolution for search-and-replace with its ability of previewing the results with one click compared to a series of replacement and it also works to specify global structure transformations [12]. Compared to other PBD systems, LAPIS provides more feedback [14]. Through direct visibility and incremental feedback of every change, this technique stimulates user confidence [12].

2.4 Regular Expressions

Regular expressions or simply abbreviated as regex (plural: regexes) or regexp, is a string that contains combination of symbols and characters, formed as a pattern. Regex is first popularised from grep, a utility from the UNIX after the concept of regular language was introduced by Stephen Kleene in 1950. Regexes are often synonym with search-and-replace functionality in text processors that allows text replacement while editing and execute automation of process. Each search pattern and replace pattern should be defined by the user with a specific formal language on the regions to be modified and the corresponding desired changes accordingly [17].

In terms of usability, regexes are interesting and have powerful computational properties. Regexes are used as the core of powerful language for specifying text processing scripts as in

sed, awk or the Perl language. Regexes are supported by most programming languages but the features, performance and matching behaviour depend on the engine. To properly use a regex library one must understand its engine. Regexes are expected to be difficult for non-programmers to learn [20], but there are extensive tutorials, guidelines and trainings materials which will help the user to understand. Investing a mental effort and a little time to learn regexes and devise an abstraction to simplify a task would be worthy in future as it will be more efficient and save time.

2.4.1. Regex Engines/ Libraries

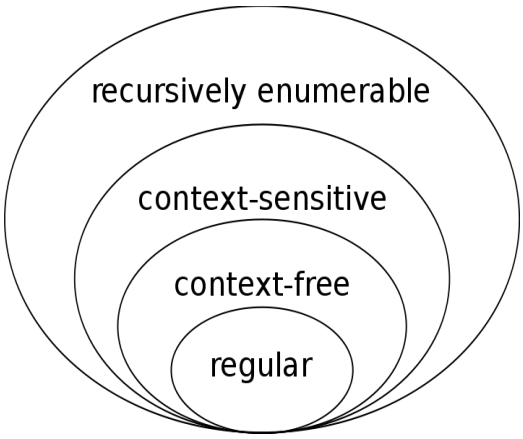
PCRE (Perl Compatible Regular Expressions), POSIX (Portable Operating System Interface for uniX), Perl, GNU and Ruby are a few examples of the regex engines or libraries. The features such as lookahead, lookbehind, backreferences, named capture, conditionals, unicode property support and others are different between each engine. An application using a library for regular expression support does not necessarily offer the full set of features of the library, as an example GNU Grep which uses PCRE does not offer lookahead support, though PCRE does.

2.4.2. PCRE and Perl

PCRE stands for Perl Compatible Regular Expressions, an open source regular expression library in C written by Philip Hazel. PCRE syntax is more powerful and flexible than many classic regular expression libraries. Though the name is "Perl Compatible", it is misleading because PCRE and contemporary versions of Perl have wide differences that can be considered as distinct regex flavours. PCRE library is compatible with many C compilers, operating systems and other programming languages since many people have derived the libraries to expand its compatibility. In an attempt to make Perl more PCRE-compatible, the latest version of Perl have even copied the features from PCRE that PCRE had beforehand from other programming languages. PCRE is widely used nowadays compared to Perl because PCRE is a part of so many libraries applications [22][23].

2.4.3. What regular expression can match

Based on Chomsky hierarchy, there are four types of languages:-



Type 0: Recursively enumerable languages

Type 1: Context-sensitive languages

Type 2: Context-free languages

Type 3: Regular languages

Figure 1: Chomsky Hierarchy

The 'regular expressions' used by programmers have very little in common with the original notion of regularity in the context of formal language theory. A modern regex flavour can match more than just regular languages and most programming language are of the type 2 grammar including well-formed HTML. Surprisingly, regexes can still match at least some of context-sensitive languages [21].

Chapter 3

Requirement Analysis

3.1 Introduction

This chapter describes the analysis on the requirements followed by feasibility studies. This analysis document will help to narrow down the scope of development within the given project timeline and drive the design and development stage.

The application to be developed is called Interactive File Transformer (IFT), is in a form of web-based system. The user will interact with the system through a graphical user interface (GUI) screen accessed from any Internet browsers such as Google Chrome, Mozilla Firefox, Internet Explorer and such.

3.2 Functional requirements

Through a graphical user interface, the application should be able to

- Allow user to upload an input file called *source* for transformation.
- Allow user to input a set of regexes called *search pattern* and its substitutions called *replace pattern* and choose the *operation* [match, match all, replace, split].
- Produce the output called *results* for each transformation on a different window.
- Record every transformation step called *sequences* entered by the user including *source*, *search pattern*, *replace pattern*, *operation* and *results*.
- Recorded *sequences* can be viewed again with all the attributes.
- Allow user to rearrange the order of *sequences,* delete, edit and reuse any of them.
- Allow user to select the set of recorded *sequences* and apply them to different *source*.
- Ouput *results* file, called a transformer

3.3 Non functional requirements

- Ease of access from any machine
- Compatible with any operating systems platform.

- Ease of interaction with the interface design
- Easy construction of tasks and handling transformation sequence.

3.4 User groups

The target user group for the developed solution are

- programmers - to test on regular expressions
- non - programmers and IT Students - to learn and understand about regular expressions and its power in text processing
- users who experienced repetitive text editing - can create an customisation on transformation and apply the sequences to other files with the same format.

3.5 Feasibility measure

Feasibility study is drawn to measure the best way to implement the tools in order to achieve the goal, both on delivering a good report and a good solution. The major factors to be considered is project time and current experience. My own previous experience and knowledge around 10 years – on and off in web based application development drives the choice to build the project through web based approach. The reason behind this is because time spent to focus on delivery of the report is more important than investing it to learn other programming language or techniques in a strict project time. In order not to affect the timeline and lost the focus, it is best to proceed the development in web-based environment.

3.5.1. Goal of the development

- Merging the motivation, aim of the project, objectives and minimum requirements that are listed in Chapter 1, the goal of the development are listed as follows:-
- User can create their own abstraction to convert a file. They can customise the transformation sequences specific to their task to transform a file to another format. User can re-apply the transformation sequence again to other files with the same format only with a few clicks that will save time.
- Reduce the typing errors occurred during conventional text editing when the task can be automated.
- Bridge the gap that existing tools do not offer: Reusability of transformation sequence

that will decrease the number of keystrokes and mouse clicks and improve efficiency of text editing task.

- Regular expressions training tools for the IT and non-IT users – every transformation details and results is displayed to see the effects of each operation. Trainee will understand how regular expressions operate.
- Helpful tool for the programmers to automate tasks.

3.5.2. Technical Feasibility

The minimum requirements in Para 1.5 mentions that the application should not be platform-specific and should perform similarly on every desktop architecture. Since web-based approach is decided for the development, it suits well with the requirement. Web-based is accessible by clients from any desktop architecture and operating systems platform. The language chosen is PHP. PHP have two different regex libraries which are POSIX-extended and PCRE. The PCRE is faster and powerful compared to POSIX and this project will be using PCRE.

3.6 Advantages of Open Source Software

PHP is an open-source language and usually installed together Apache and MySQL. LAMP (stands for Linux Apache, MySQL, PHP) is a free bundle installer for these Open Source Software . This combination is popular because of its low acquisition cost and the ubiquity of its components. LAMP will be installed as the package to construct the development environment.

If the project is going to be extended in future, PHP can still be executed on all major operating systems including Microsoft Windows and Mac OS X. Apart from Apache, PHP can also run on Microsoft's Internet Information Server. PHP supports other databases too such as Oracle, PostgresSQL and MS Access. It is also compatible with other web technologies including Java.

There are many online forums and references discussing on open source language like PHP. Large group of developers contributes to the library development and extensions and many expert users support the usage by sharing online tutorials and references. It is guaranteed that

the support is longer for future.

3.7 Advantages of web-based application

Apart from the flexibility to access to application from a uniform environment – a web browser, and cross platform compatibility, the other advantages are explained below:

3.7.1. Easier installation and maintenance

Installation and maintenance is easier because it is directly done on a server. Unlike other standalone software that requires user to download and install updates themselves, any updates or upgrade of web-based application will not involve the client's PC.

3.7.2. Cost effective development

The development does not need to be tested on all possible operating system versions and configuration, it means the testing and troubleshooting becomes easier. The user interaction needs to be tested on different browsers, but the application itself is only developed from a single machine with the same code for all browsers. The architecture is simplified.

3.7.3. Accessible for a range of devices

This advantage may be left-out behind. Although there is no requirement for the project to run from any mobile devices, surprisingly the project soon can be accessed from any smart phones. Future work may consider extensions to access from mobile devices.

3.8 Additional development tools

Any regular text editors can be used in to write PHP code. However, Dreamweaver is a good companion to work in a WYSIWYG mode to design the GUI and process flow. It is more convenient for the development compared to normal text editors that does not offer WYSIWYG mode.

The version to be used for this project is Macromedia Dreamweaver 8.0. Although the version is quite backdated compared to the latest one by Adobe Systems, Dreamweaver CC2014 (Version 14, Relased on 18 June 2014), through my experience using both from Macromedia and Adobe Systems, I would prefer Macromedia because the application is

lighter to load on the machine. The features older version is still capable to support the development for this project.

One feature that makes Dreamweaver as a good web-based development tool is the ability to directly link the project with the database details. In order to create a database query from the editor, it will suggest the related field name and shows the resulting effects.

3.9 Development machine

Below is the hardware specification of the machine and the required software to develop the project:

Operating System:	ubuntu 12.04 LTS
Memory:	3.7 GB
Processor:	Intel Core i3 CPU M330 @ 2.13GHx X 4
Database:	MySQL Version 5.5.38
Server:	Apache 2.2.22 running on localhost
PHP:	Version 5.5.38
PCRE Library:	Version 8.12 2011-01-15
Editor:	Macromedia Dreamweaver 8

3.10 IFT vs. text editors and word processors

It is a good approach to highlight and understand the differences in order not to mistakenly implement or understand something out of scope.

Before word processors exist, text editors were the precursors. Text editor is used for editing and composing plain text files but it does not add additional formatting informations to the files. They usually come with the operating systems. Although with many limitations, people are still using it. A word processor is a type of text editor with formatting capabilities. Styles can be applied to the text such as changing font size, applying font colours, centring or indenting. If a word processor file is opened in a text editor, we will notice that the file contains formatting codes.

This project (IFT) is not a text editor or word processor and does not provide features that are common with both. This project only focus on the scope as outlined in this chapter.

Chapter 4

System Design

4.1 Introduction

This chapter covers explanation of interface design, system architecture, functionalities, and database design.

4.2 Inspiration of system design

Two similar existing tools have inspired the design of the systems in terms of user interface, user interaction and common basic functionalities and aim for training purpose. The web-based technology from My Regex Tester, and the history style of RegexBuddy has outlined some ideas to start with the design. Their limitation, however is bridged by this project by offering reusabilty of transformation sequences to be applied to automate other similar tasks.

RegexBuddy [www.regexbuddy.com] is an interesting standalone application to work with regexes and can be used as a teaching tool. It offers variety of choice with programming languages and different regex libraries for testing. It has history feature but it cannot be reapply to other tasks for automation of text processing. While My Regex Tester [http://myregextester.com/] is a web-based application for testing and training on regular expressions. It uses the PCRE engine.

4.3 Web-Based System Architecture

The figure below shows the architecture of web-based system.

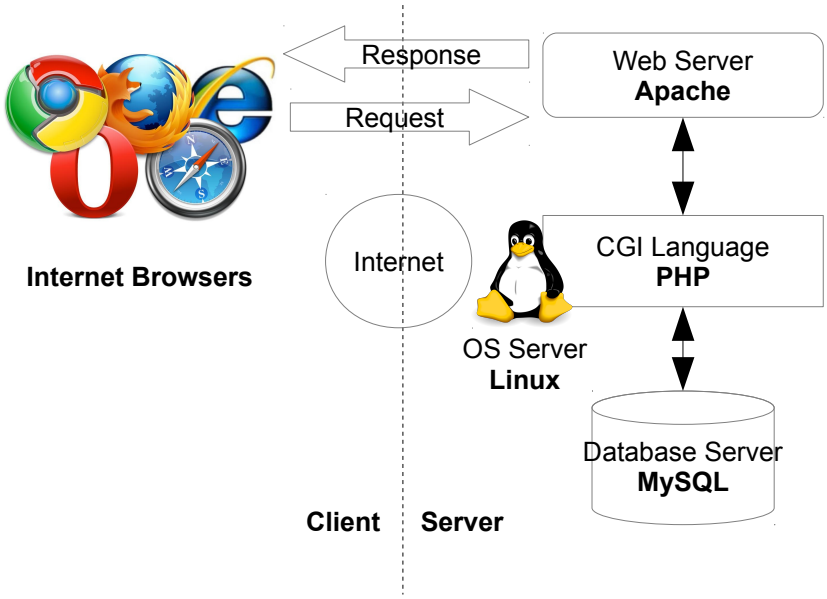


Figure 4.1: System architecture

4.4 Functionality Design

Functional requirements as listed in Para 3.2 is break down into three main processes and functions as in the table below. The implementation will follow the stages from this functionality design

Process	**Functions**
Task	- Create a new task - Create a new task by copying the sequences from existing task - Load existing or newly created task to begin
Sequences	- Apply *search pattern* and *replace pattern* using regular expessions - Every transformation and its details are recorded as a *sequence* - User can view/ edit the previous transformations and apply again to the souce - The *results* of transformation will be displayed in different window pane - *Results* of previous transformation can be redisplayed - User can create a predefined transformation for a specific task - User can select *sequences* and apply to other source

	- *Results* will be displayed on Results panel.
Result	- *Results* are saved in database as text format - User can export/ output the selected *results* to desired formats - User can load the *results* as souce between transformation.

4.5 Database Design

4.5.1. Entity Relationship Schema

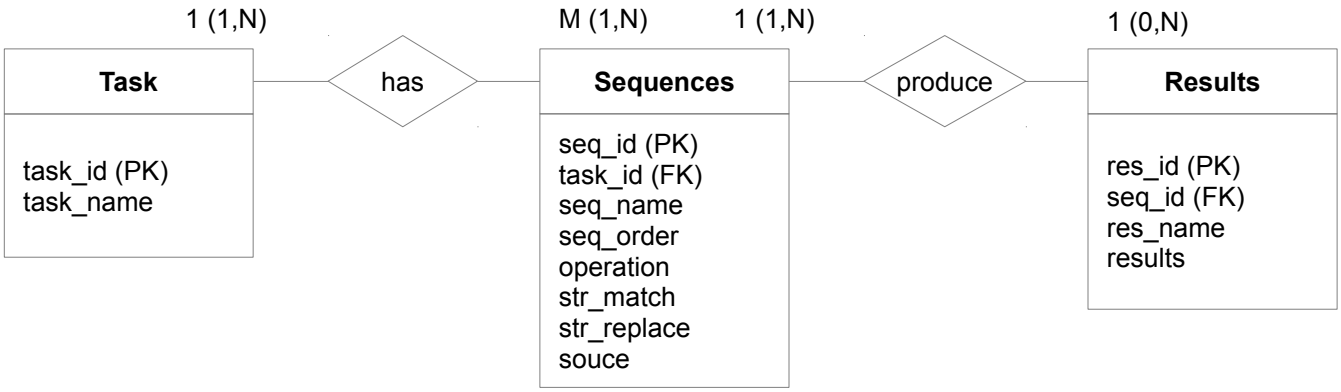


Figure 4.2: Entity Relationship Schema

4.5.2. Data dictionary

Data Dictionary for Database IFT				
Table	**Attribute**	**Type**	**Length**	**Description**
task	task_id	integer	3	Primary Key
	task_name	varchar	10	Task name
sequence	seq_id	integer	3	Primary Key
	res_name	varchar	10	Sequence Name
	seq_order	integer	3	Sequence Order
	task_id	integer	3	FK References task
	str_match	text		Match pattern
	str_replace	text		Replace pattern
	operarion	varchar	10	Operation [match, matchall, replace, split]
	source	mediumtext		Soource for operation
result	res_id	integer	3	Primary Key
	res_name	varchar	10	
	seq_id	integer	3	FK References sequence
	results	mediumtext		

Table 4.1: Data Dictionary of IFT

Chapter 5

Implementation

This chapter explains the implementation of the project.

5.1 PCRE

The regex library used in the implementation is PCRE. There are 9 PCRE functions [27] but only four are used for this project and the most related one is only for seach and replace feature.

	PCRE function	
1	preg_match	Perform a regular expression match
2	preg_match_all	Perform a global regular expression match
3	preg_replace	Perform a regular expression search and replace
4	preg_split	Split string by a regular expression

The difference between preg_match and preg_matchall is that preg_match stops after the first match and preg_matchall gets all matches.

5.2 Basic Operations

Before going further with more complex functionalities, let us look at the basic operations.

5.2.1 Match

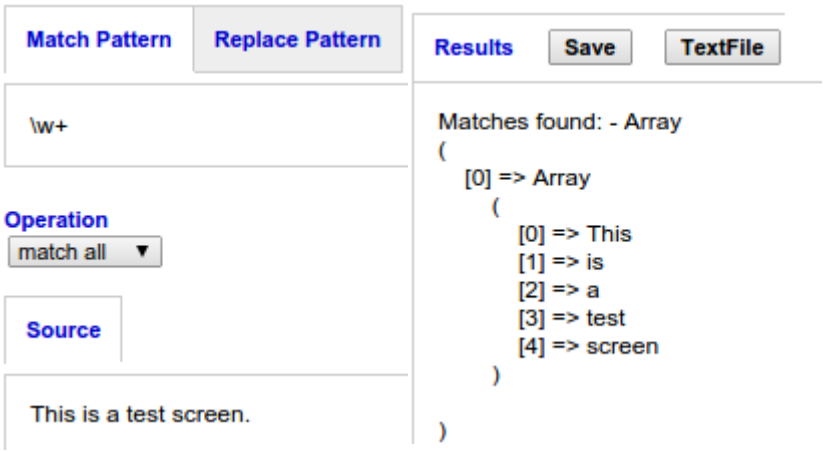
Code: preg_match (string match_pattern, string source, array matches)

Ouput: first match

5.2.2 Match all

Code: preg_match_all (string match_pattern, string source, array matches)

Output: An array of matches



5.2.3 Search and replace

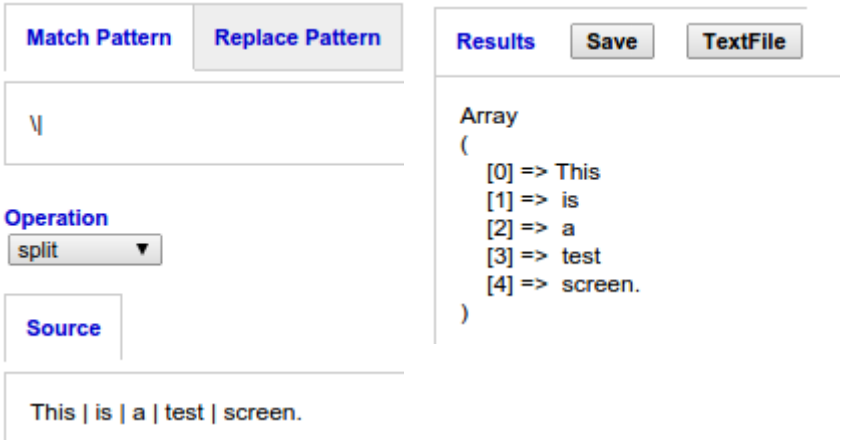
Code: preg_replace(string match_pattern, string replace_pattern, string source);

Output: The replaced input.

5.2.4 Split

Code: **preg_split**(string split, string source);

Output: An array of strings or characters split by the token.

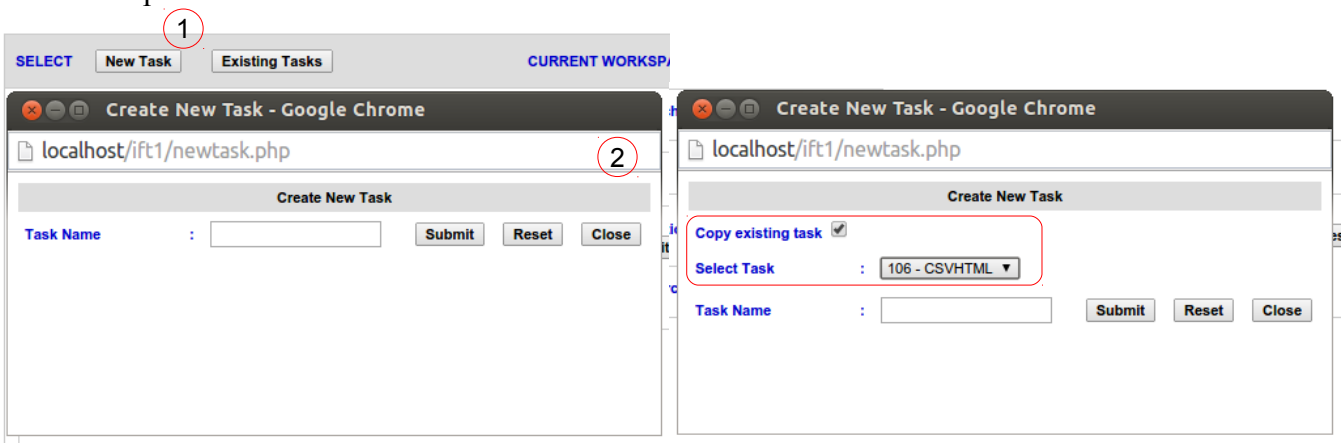


5.3 System flow

The next description will be in narrative format according to screen captures and numberings attached to them.

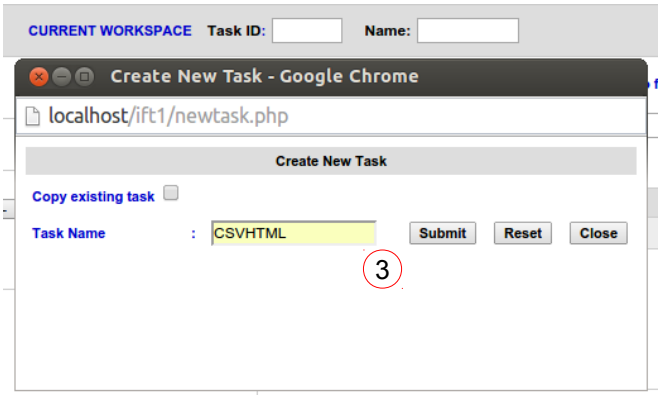
5.3.1 Create task

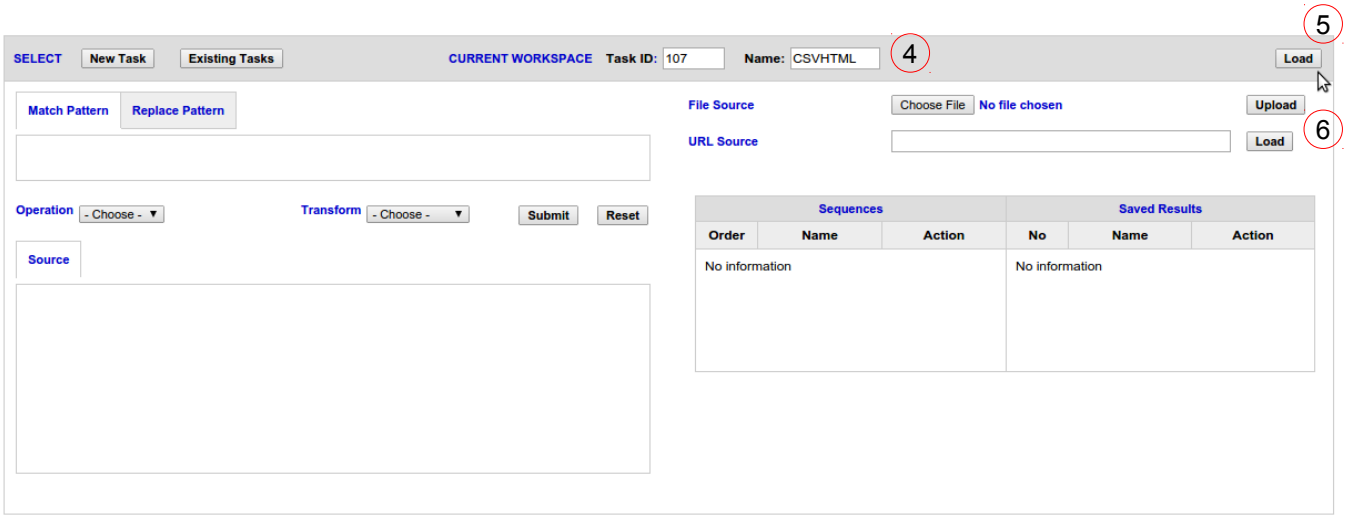
Before a user wish to convert a file, he should first create task. The reason behind this is because the system will record transformation sequences specific to the task itself and allow the reusability of the sequences next time. The user can customise the transformation until the desired output is achieved.



Before start, user needs to click on "New Task Button" (1) to create the task. It is important to create task because all related transformation sequences will be kept according to the task itself. It is easier to manage and promotes reusability in future.

The system will prompt a small pop-up window (2). If there is any previous task, it will allow to copy any selected previous task as a new one with all the sequences and results information. If there is no existing task, the choice to copy would not appear. User should give a name for the task so it would be easier to recognise (3).





The system will assign an ID to the new task created (4). User will need to click "Load" (5) in order to create the environment to begin. The source file can be loaded from the machine or from any URL in any textfile readable format (6). For this project, only a few format are focused such as txt and csv.

5.3.2 Creating a predefined sequence

Use Case: Transformation from CSV format to HTML table

A complete, well-formed HTML table with visible border lines will begin with *<table border = '1'>* tag and end with *</table>* tag, every row opens with *<tr>* and closes with *</tr>* and every column begins with *<td>* and ends with *</td>*. In order to transform a CSV file to a HTML table, we can analyse the transformation steps as follows:-

i) Add *<table border = '1'>* to the start of the source string

ii) Add *</table>* to the end of the source string

iii) In every line of the data, add *<tr>* before the first *<td>*, and each data should be enclosed with *<td>*data*</td>* and will end with *</tr>*

Here we analyse the format of sample data [Appendix B1]

```
street,city,zip,state,beds,baths,sq__ft,type,sale_date,price,latitude,longitude
4712 PISMO BEACH DR,ANTELOPE,95843,CA,5,3,2346,Residential,Mon May 19 00:00:00 EDT 2008,320000,38.707705,-121.354153
4741 PACIFIC PARK DR,ANTELOPE,95843,CA,5,3,2347,Residential,Mon May 19 00:00:00 EDT 2008,325000,38.709299,-121.353056
310 GROTH CIR,SACRAMENTO,95834,CA,4,2,1659,Residential,Mon May 19 00:00:00 EDT 2008,328578,38.638764,-121.531827
```

The first row contains the column names. The regexes for match pattern be explicitly written as follows:-

Capture Group:	$1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12
Match Pattern:	(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+)
Source:	street,city,zip,state,beds,baths,sq__ft,type,sale_date,price,latitude,longitude

The following lines after the column names contain data and the regexes are:-

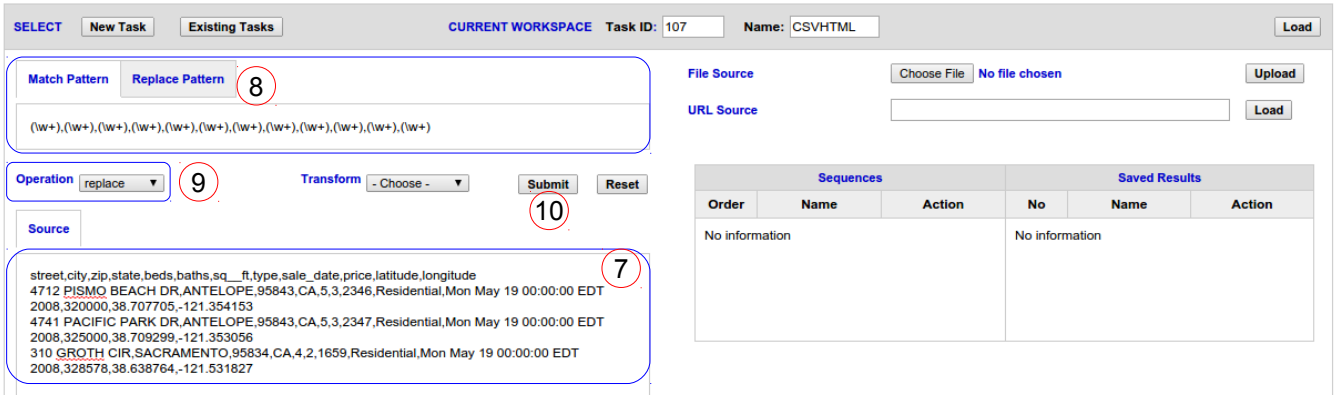
Capture Group:	$1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12
Match Pattern:	(\d+ \w+.*),(\w+.*),(\d{5}),(\w+),(\d+),(\d+),(\d+),(\w+.*),(\w{3} \w{3} \d+ 00:00:00 EDT \d{4}),(\d+),(\d+.\d+),(-\d+.\d+)
Source:	4712 PISMO BEACH DR,ANTELOPE,95843,CA,5,3,2346,Residential,Mon May 19 00:00:00 EDT 2008,320000,38.707705,-121.354153

Replace Pattern for both column names and data is:

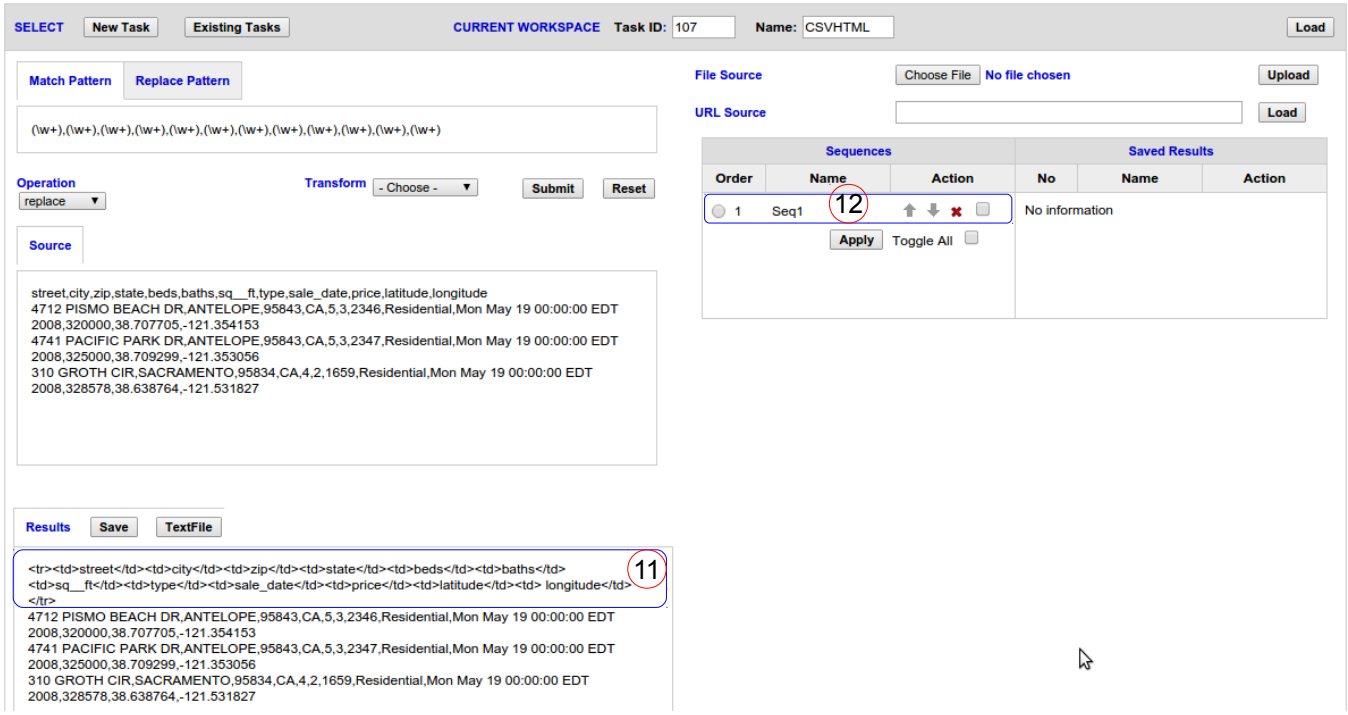
<tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td><td>$6</td><td>$7</td>

<td>$8</td><td>$9</td><td>$10</td><td>$11</td><td> $12</td></tr>

Now, we can create the sequence through the system.

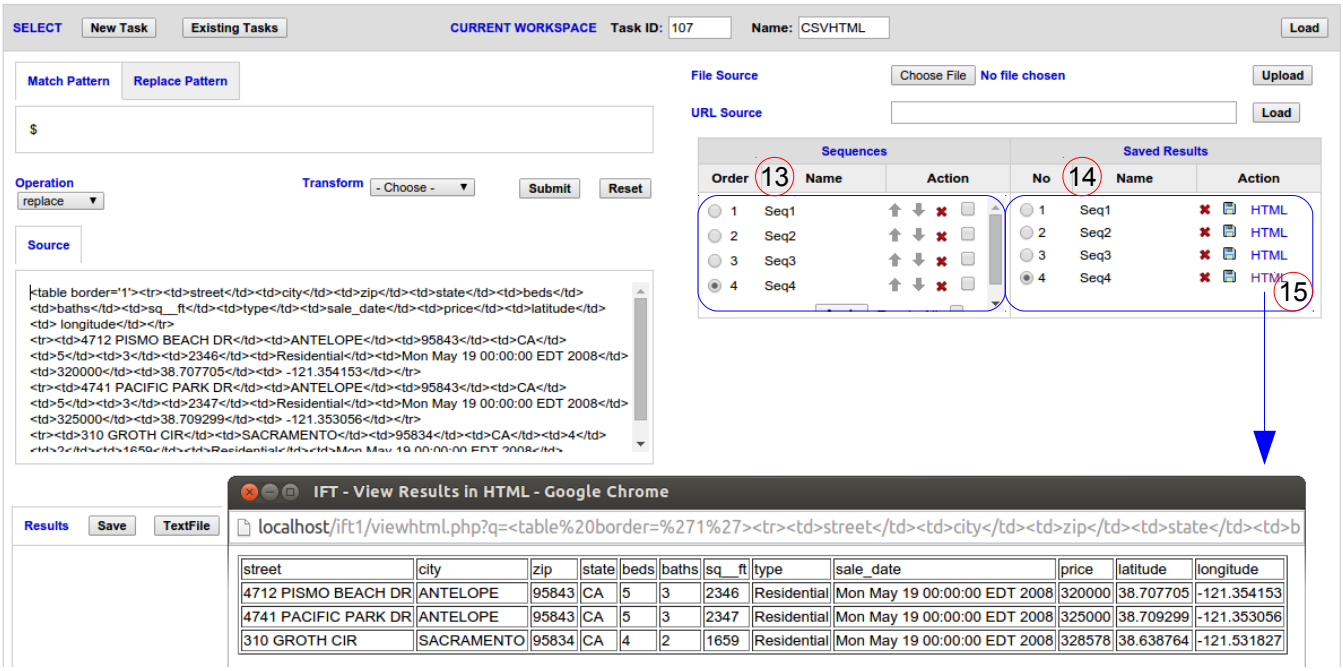


Insert the source in (7), match parttern and replace pattern in (8) accordingly, choose "replace" in operation (9) and press submit button. The results will be as the following figure.



The results will be displayed in (11) and the sequence will be recorded as (12). Repeat the process by applying the next match and replace pattern until the sequences are complete. Assume that the regexes will match all pattern without any errors, the final screen of the

transformation should appear like this.



All 4 sequences of are recorded in (13) and results in (14). Clicking on the final HTML link in (15) will preview the results of transformation. The HTML code of the table can also be downloaded in textfile by clicking the "Save" Icon.

5.4 Reusabilty

A new task can be created by copying any existing tasks. Let's say the user wants to transform other files with the same format as previous one. He can start create a new task by copying sequences of exisitng task (16). All sequences from previous task will be copied as in (17) when the new task is load on the application. User will have to load the new source (18) and check Toggle All (19) to apply the sequences to automate the conversion process. As in (20), the results will appear and the user can click on HTML link to preview the transformation. The result can also be downloaded in text file format. The following figure is the flow and algorithm of the reusability process.

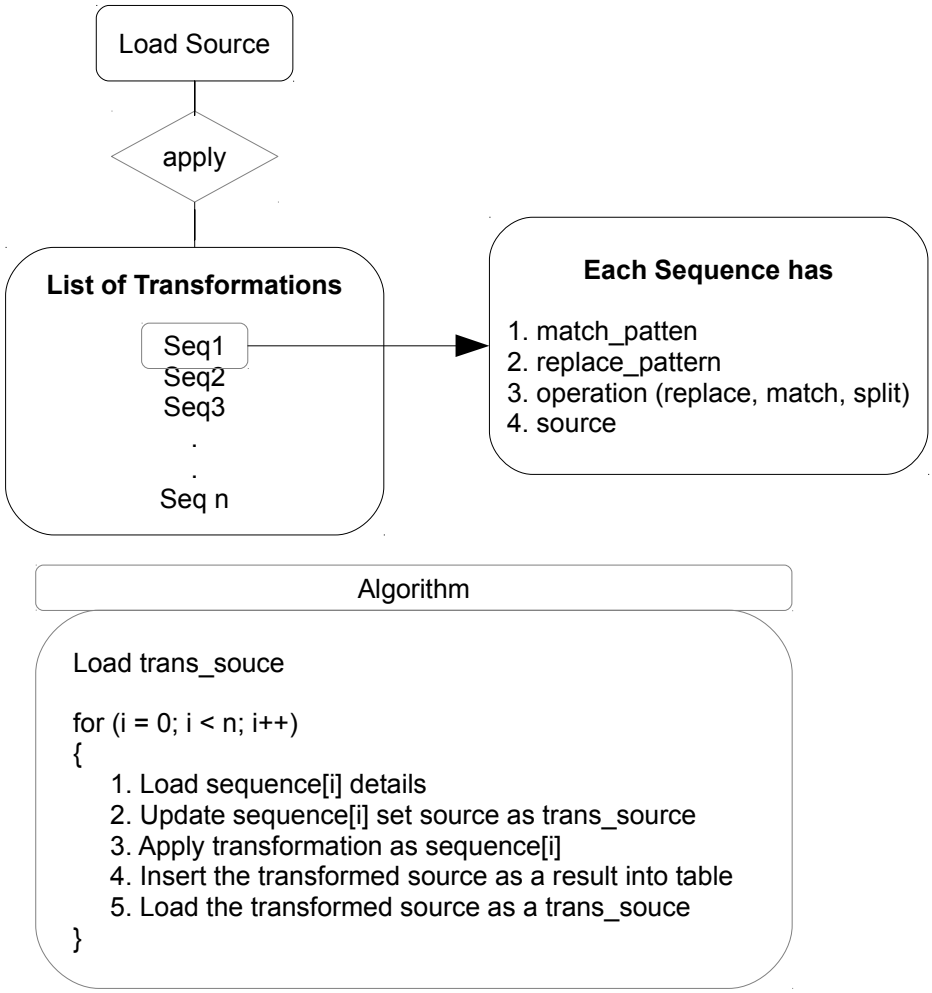
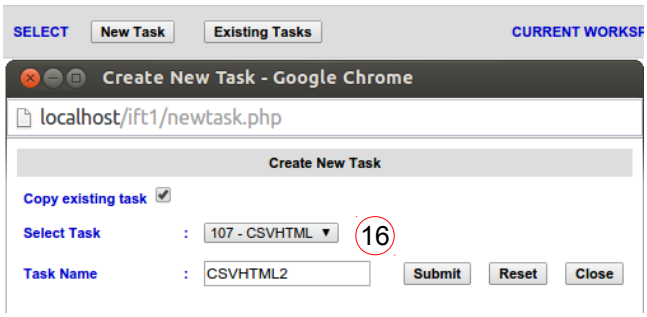
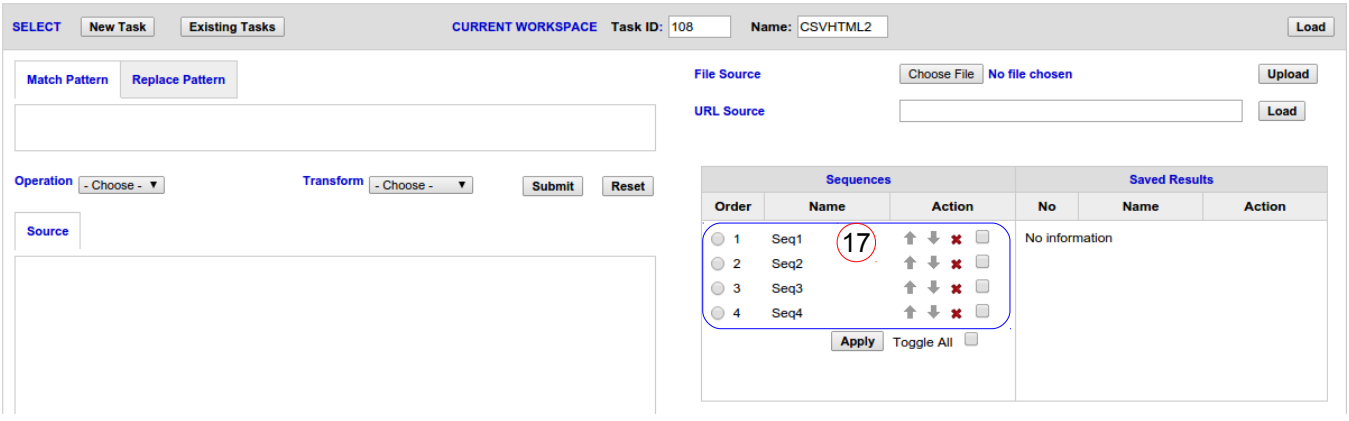
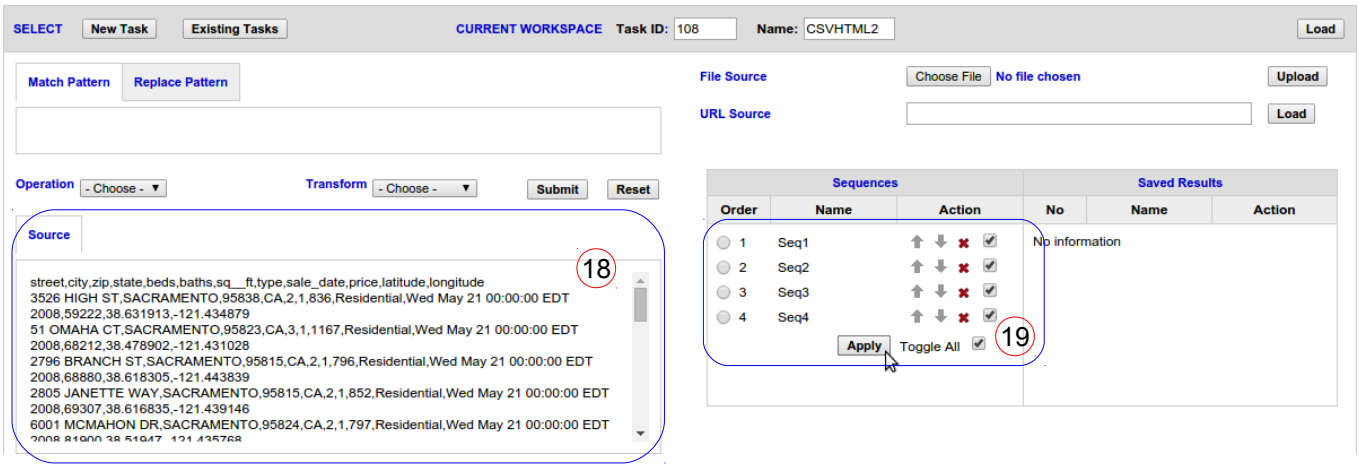
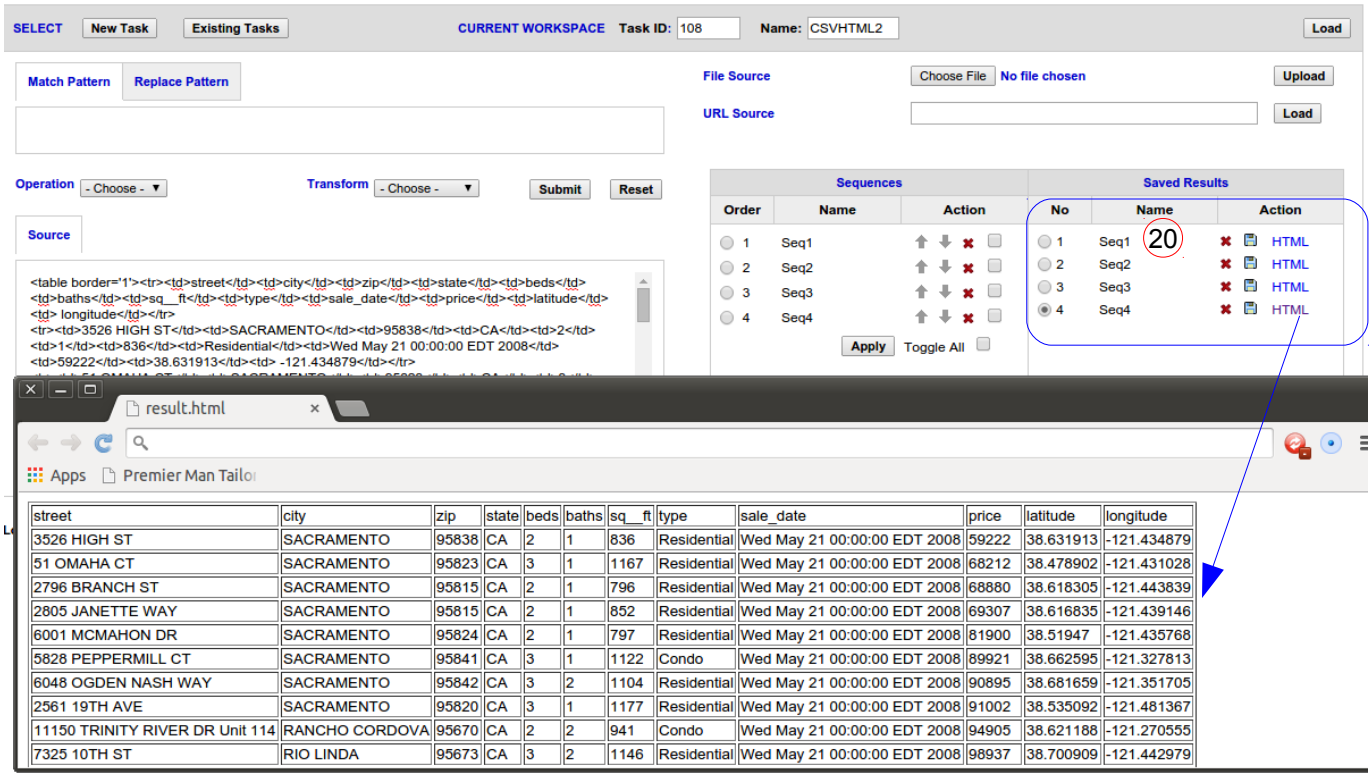


Figure 5.1: Flow and Algorithm of Reusabilty`









street	city	zip	state	beds	baths	sq__ft	type	sale_date	price	latitude	longitude
3526 HIGH ST	SACRAMENTO	95838	CA	2	1	836	Residential	Wed May 21 00:00:00 EDT 2008	59222	38.631913	-121.434879
51 OMAHA CT	SACRAMENTO	95823	CA	3	1	1167	Residential	Wed May 21 00:00:00 EDT 2008	68212	38.478902	-121.431028
2796 BRANCH ST	SACRAMENTO	95815	CA	2	1	796	Residential	Wed May 21 00:00:00 EDT 2008	68880	38.618305	-121.443839
2805 JANETTE WAY	SACRAMENTO	95815	CA	2	1	852	Residential	Wed May 21 00:00:00 EDT 2008	69307	38.616835	-121.439146
6001 MCMAHON DR	SACRAMENTO	95824	CA	2	1	797	Residential	Wed May 21 00:00:00 EDT 2008	81900	38.51947	-121.435768
5828 PEPPERMILL CT	SACRAMENTO	95841	CA	3	1	1122	Condo	Wed May 21 00:00:00 EDT 2008	89921	38.662595	-121.327813
6048 OGDEN NASH WAY	SACRAMENTO	95842	CA	3	2	1104	Residential	Wed May 21 00:00:00 EDT 2008	90895	38.681659	-121.351705
2561 19TH AVE	SACRAMENTO	95820	CA	3	1	1177	Residential	Wed May 21 00:00:00 EDT 2008	91002	38.535092	-121.481367
11150 TRINITY RIVER DR Unit 114	RANCHO CORDOVA	95670	CA	2	2	941	Condo	Wed May 21 00:00:00 EDT 2008	94905	38.621188	-121.270555
7325 10TH ST	RIO LINDA	95673	CA	3	2	1146	Residential	Wed May 21 00:00:00 EDT 2008	98937	38.700909	-121.442979

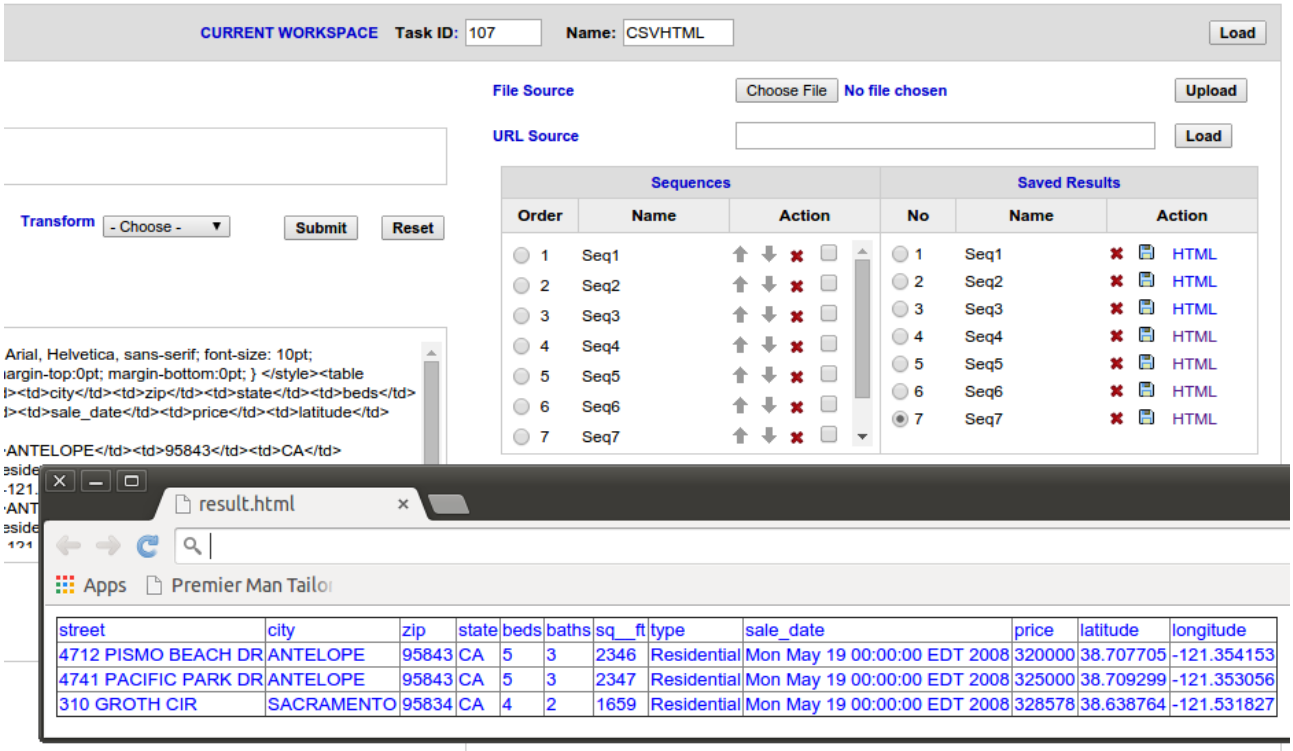
5.5 Customising Transformation

A little advanced user may again reuse the predefined sequences from the previous task ID 105 (CSVHTML) and modify Seq3 to insert a simple CSS (Cascading Style Sheet) to format the table with a nice border and font colour.

```
<style type='text/css'>
.theme {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 10pt;
        color: #0000FF;
        border-collapse:collapse;
        margin-top:0pt;
        margin-bottom:0pt;
}
</style>
```

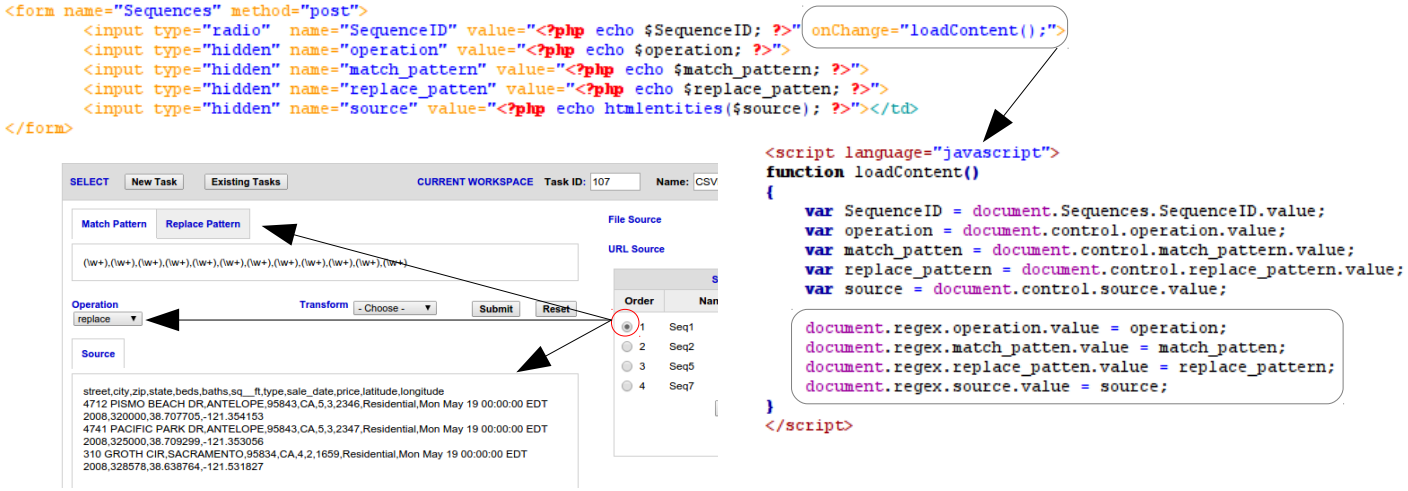
This code should be place before <table border = '1' > tag, and <table boder = '1'> should be edited as <table boder = '1' class = 'theme'>.

It will ouput a nice HTML table as follows:



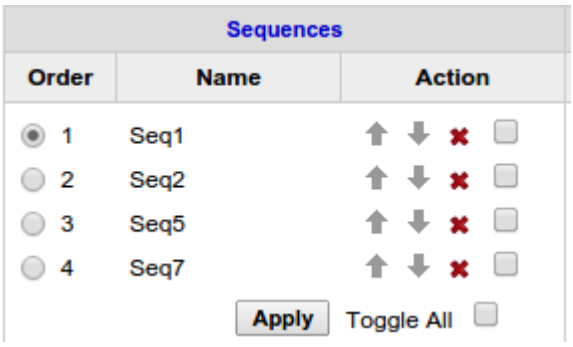
5.6 View details and results of every transformation

The user can view the details and results of transformation by clicking on the radio button next to the sequence order and result list. The application will automatically load all the details. This feature is done by using javascript as in the figure below.

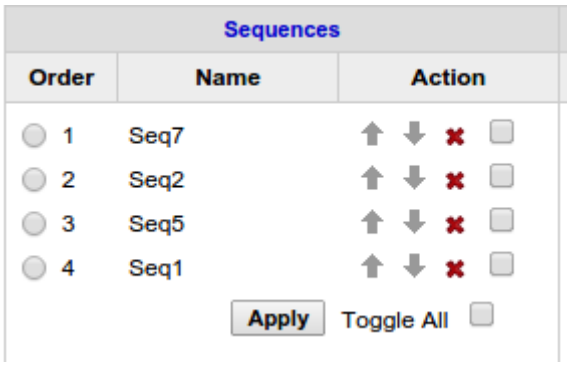


5.7 Rearrange the order of sequences

Before



After



The following figure is the algorithm for controlling the changes of sequence.

```
/* the current order of the sequence to move*/
SequenceToMove

/* Move up one position above */
if (move == "Up")
{
    /* Check the upper position not to be less than 1 */
    if ((SequenceToMove - 1) > 0 )
    {
        Check if there is any sequence on the preceding order.
        If true
        {   1. Assign the order of that sequence to a temporary position.
            2. Update the location of SequenceToMove to SequenceToMove - 1
            3. Reassign the previously affected sequence to the succeeding order of SequenceToMove.
        }
        else
        {
            1. Assign SequenceToMove to preceding order.
        }
    }

    Refresh the webpage with the updated Sequence order.
}

/* Move up one position below */
if (move == "Down")
{
    Check if there is any sequence on the succeeding order.
    If true
    {   1. Assign the order of that sequence to a temporary position.
        2. Update the location of SequenceToMove to SequenceToMove + 1
        3. Reassign the previously affected sequence to the preceding order of SequenceToMove.
    }
    else
    {
        1. Assign SequenceToMove to its succeeding order.
    }

    Refresh the webpage with the updated Sequence order.
}
```

5.8 Limitation

Due to limitation of development time, this project does not develop the following features:

- Delete task
- Rename task
- Rename sequence
- Highlighting of matched replacement

Chapter 6
Evaluation

6.1 Introduction

This chapter explains the evaluation carried out for the implemented application and shows the results. Methodology of the first and third evaluation is by using 4 use cases as in Appendix E. Each use case is readily supplemented with their substitution patterns.

CSV Converter, however is not really significant to be compared on mouse clicks and reusability because it is a ready built transformer. However, it is put under the evaluation to show the comparison that readily built application is very static, cannot convert customisable formats and does not offer reusability despite of the less mouse click.

6.2 Walk-through

This evaluation involves counting the number of mouse clicks to perform 3 use cases. IFT is compared with 3 applications; RegexBuddy, My Regex Tester and CSV Converter. The print screen of other applications can be referred in Appendix G. The results of transformations from all applications meet the desired output pattern.

6.2.1 Methodology of evaluation

Assume that the initial source and each substitution patterns are copied and pasted on the evaluated applications and not being typed by the user in order to avoid counting the number of keystrokes and typing errors. The copying of substitution patterns does not count any click in this because it is outside of the application. However, the 'copy' and 'paste' action that happens inside the applications counts as 3 clicks per each 'copy' and 'paste', considering as putting the cursor on the field, right click the mouse and paste the input as one click each.

The details of walk-through is presented in Appendix F. The overall results of comparison are simplified in the table as follows:-

Use Case		Application			
		RegexBuddy	**My Regex Tester**	**CSV Converter**	**IFT**
1	**CSV to HTML**	49	58	4	53
2	**HTML to CSV**	47	53	4	35
3	**VCF to HTML**	154	177	N/A	135

Table 6.1: The number of mouse clicks to perform use cases

6.2.2 Observation

In two use cases, IFT contributes the most less clicks. IFT can have less more clicks if the user interaction is improved, such as removing the Replace tab which is currently placed next to Match tab. Switching between tabs consumes extra mouse clicks.

IFT saves more clicks when loading the previous results as current source just by one radio button click, compared to other applications which need 3 clicks for re-pasting the source; 1 click to locate the cursor in the source field, one for right click and another one for 'paste'.

IFT can even save more clicks if the substitution result is automatically saved. However, saving the result to the database will consume other memory resources on the server. Apart from resource issues, the reason why the current design does not automate the process because it would not save the results that do not match, come with errors and do not execute properly.

Overall, IFT still wins the evaluation in terms of walk-through.

6.3 Reusability

Given a scenario where a user or an organisation has a high frequency deals with structured text conversion in daily tasks, the same type of conversion over the same file structure may repeat. This is where IFT becomes an advantage because the sequences are saved and managed according to tasks. The other applications do not offer reusability where user has to rebuild the transformation again from the beginning every time they would like to do transformation. RegexBuddy, however keeps the history but it is not well managed according

to tasks and may confuse the user.

From the evaluation drawn as in Appendix D, the following formula for counting the mouse clicks is obtained for each application:

Application 1: RegexBuddy

Formula: (The number of substitution patterns * 13) – 3

Application 2: My Regex Tester

Formula: (The number of substitution patterns * 15) – 2

Application 3: CSV Conversions

4 clicks (for any CSV files to some formats).

IFT

Formula: (The number of substitution patterns * 12) – 5.

Due to the benefit of reusability offered by IFT, the formula does not apply to the next transformation of the same file structure because the number of mouse clicks would totally reduce. Regardless the number of mouse clicks used to create the sequence for initial task, it only takes a fixed number of clicks at most 13 for any transformations as presented in the table below, assuming all transformation sequences execute with no errors and no additional substitutions are modified in between conversions. User does not need to type or copy and paste again the substitution patterns, manually load the source over every transformation. All transformations are automatically run. This will definitely save more time and make file conversions more efficient.

Steps		**Number of clicks**
1	Create new task	1
2	Check copy existing	1
3	Scroll list and select task	2
4	Insert task name	1
5	Click submit button	1

6	Click load task	1
7	Paste the initial source	3
8	Check Toggle All	1
9	Click Apply	1
10	View final result	1
Total number of clicks		**13**

Table 6.2: Reduced number of mouse clicks to perform the next task on IFT

6.4 Reusability Performance

Appendix E explains detail on each transformation, where a list substitution patterns is first drawn for each use case. The number of substitution patterns depend on the format of the original source file and the final intended output. The previous two evaluations discuss on the number of mouse clicks each application contributes to every use case and how IFT manage to reduce the number of clicks tremendously with its reusability. But how about the performance on processing time for a large data on a longer list of sequences?

This evaluation will only focus on the performance of IFT itself since the other 3 applications do not offer reusability, except for RegexBuddy that keeps history but still confuses the user since it is not arranged according to specific tasks.

A regexes expert may implicitly devise more complex substitution patterns, combining some features such as lookahead, lookbehind, zero-width assertions and backreferences that will execute advanced substitutions in a shorter number of characters or lines of regexes. The technique may reduce more steps for transformation but may increase processing time to parse complex regexes in a higher volume of data since regexes can be considered 'a simple text processor'.

In this evaluation, the approach is straight-forward and at the same time promotes understanding the basic operation of regexes for non-familiar users.

Some examples of direct substitution patterns:

Change Column Names in CSV to a HTML row

Source: Stu_ID,Stu_Name,Birth_Year,Country,Marital_Status

Match: (\w+),(\w+),(\w+),(\w+),(\w+)

Replace: <tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td></tr>

Reformat Date

Source: 20080424T195243Z

Match: (\d{4})(\d{2})(\d{2})T(\d{2})(\d{2})(\d{2})Z

Replace: $3/$2/$1 - $4:$5:$6

	Use Case			
	CSV to HTML 1	CSV to HTML 2	HTML to CSV	VCF to HTML
Number of substitution patterns	4	4	5	13

Table 6.3: The number of substitution patterns to perform Use Case

6.4.1 Methodology

Evaluation is run on a machine with following details:

OS: ubuntu 12.04 LTS

Memory: 3.7 GB

Processor: Intel Core i3 CPU M330 @ 2.13GHx X 4

MySQL Database Version 5.5.38

Apache 2.2.22 running on localhost

PCRE Library Version 8.12 2011-01-15

Two browsers, Mozilla Firefox and Google Chrome are used in this evaluation. A javascript code which functions as timer as in Figure 6.1 is placed on the top of the screen to record the execution time. The time is recorded once the user click on “Apply” button and the system will automate the transformation until complete. Each evaluation is done one at a time and not in multiple access and run as 5 times to obtain the average running time and observe the pattern. The affected database tables are cleared after each transformation, so that a new evaluation will start in a clean environment without being affected by other unrelated data. A

detailed evaluation result can be referred in Appendix H.

```
<script Language="JavaScript">
<!-- hide script from non compliant broswers
// author: Susan Lee
var from_time = new Date();
from_time = from_time.getTime();
function  show_loading_time()
{
    var to_time = new Date();
    to_time = to_time.getTime();
    var secs = (to_time - from_time) /1000;
    document.f.t.value = secs + " secs";
}
// end hiding from non compliant browsers-->
</script>
<body onLoad="show_loading_time()">
<table width="100%" border="0" style="border-collapse:collapse" cellpadding="0" cellspacing="0">
<tr><td width="100%" class="stylelb">
<form name="f" onSubmit="0"> Loading time
<input size="10" Name="t" Value="...." class="button"></form>
</td></tr></table>
```

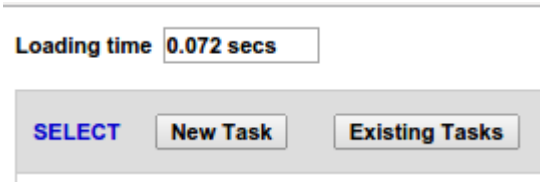
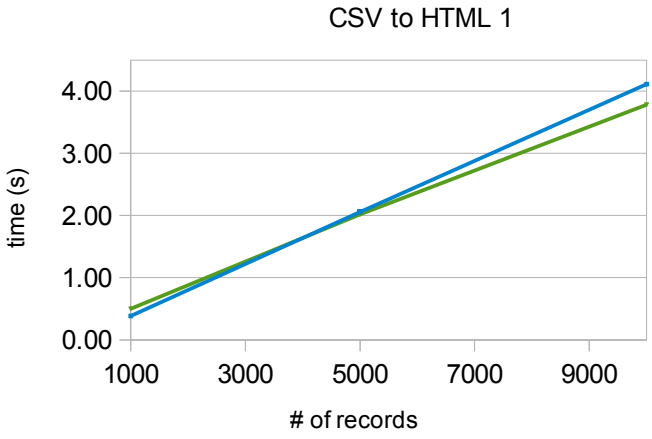


Figure 6.2: Javascript to calculate loading time

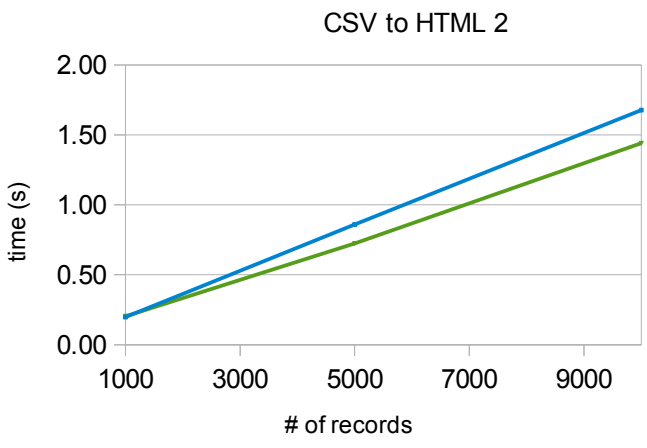
The running time on larger data are shown in the following figures:-



Browser	# records	Avg Time (s)
Chrome	1000	0.383
	5000	2.059
	10000	4.113
Firefox	1000	0.499
	5000	2.019
	10000	3.780

Chrome
Firefox

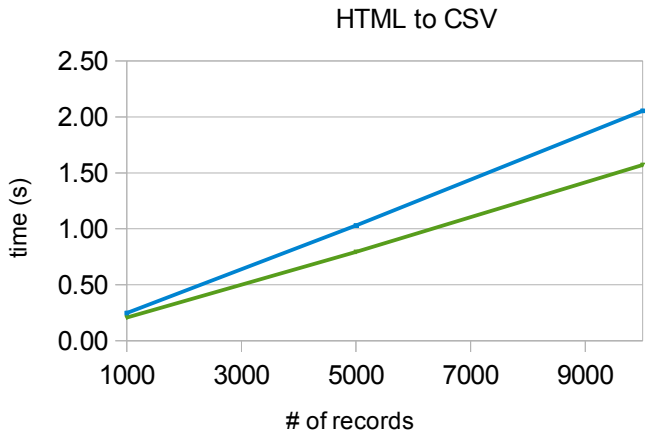
Figure 6.3: Execution Time on Use Case 1



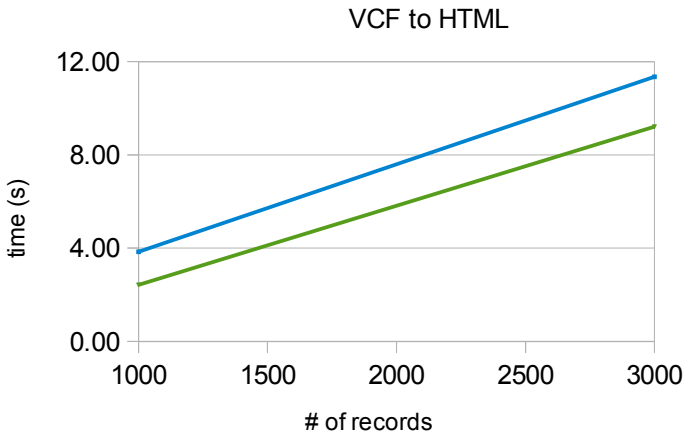
Browser	# records	Avg Time (s)
Chrome	1000	0.197
	5000	0.860
	10000	1.676
Firefox	1000	0.204
	5000	0.724

Chrome
Firefox

Figure 6.4: Execution Time on Use Case 2



Browser	# records	Avg Time (s)
Chrome	1000	0.248
	5000	1.029
	10000	2.054
Firefox	1000	0.208
	5000	0.792
	10000	1.570



Browser	# records	Avg Time (s)
Chrome	1000	3.845
	3000	11.350
Firefox	1000	2.424
	3000	9.213

6.4.2 Assumption

The timer is placed on the web page and starts its calculation when the “Apply” button is clicked. It shows the loading time once the automation is complete and the web page is fully reloads. There would be other contributing factors to the execution time in the background processes. It is not easy to properly measure each factor due to limitation of project time.

In this situation, execution time may depends on or affected by the capacity and capability of a machine both on hardware and software. It also depends on the complexity of the algorithms and system flows, the size of the record or source data to be transformed, the

sequences of regexes to get desired output, the connection time to the server and accessing database and the platform to run the application from the client side.

6.4.3 Results and observation

Despite all assumptions drawn before, it has been surprising when the execution time for huge amount of data still requires very little time. One factor to observe is when the evaluation runs faster in Firefox compared to Chrome in most cases. It means, different browsers used as the platform for the user to access the application also contribute to execution time.

This evaluation proves the power of regexes coupled with reusability will make text editing far more efficient and save a huge amount of time.

6.5 Training Tool

Another evaluation just to measure whether IFT is helpful as a regex training tool was carried out to 6 respondents, which 2 of them are not from IT background. 5 of them are not familiar at all with regular expressions. The method carried out was by approaching them personally and describe how the system works with a basic explanation on regex and showed how it works and let them tried the functionality. All of them agreed that the application met the purpose to act as a training tool.

Chapter 7
Conclusion and Future Work

7.1 Project Reflection

The developed application has met the aim and objectives and minimum requirements as outlined in Chapter one. However, there are still many improvements can be done. I personally did not see any big flaws regarding functionality with IFT during the Implementation stage. I am not concerning much on the additional accessories with the design because the important thing is that it delivers the main purpose to create a concept.

The Evaluation stage has given me a good view to reflect on my project. With just comparison on the number of mouse clicks with other applications has opened my eyes on the impact. It means, if IFT improve the user interaction, less mouse clicks will involve on each task and will affect the rest of other process. The difference can clearly be seen as in Appendix F when the formula to count for mouse clicks on each application is obtained and how it affects the process with more substitution patterns. IFT should improve on this matter so for reusability, the results would be less than 13 clicks (results from Section 6.3) for any transformation.

For calculating the performance of reusability on a huge record of task (Section 6.4), due to strict timeline, I was not able to properly assign timers to background processes in order to just evaluate the automation algorithm. Although the result was quite impressive on 10,000 records on use cases it can perform less than 15 seconds. It is a good work if in future, the algorithm is tested on huge records in more complex transformation with the timer measuring all the contributing parameters, so all aspects would be improved.

Overall, I am satisfied with this project, although I slightly feel I should have improved further. If given a longer schedule, I would run the future work to be whole heartedly satisfied with the research.

References

[1] Webopedia.com. 2014. What is Structured data? *Webopedia: Online Tech Dictionary for IT Professionals*. [Online] 3 Aug 2014. [Cited: 3 Aug 2014]. Available: http://www.webopedia.com/TERM/S/structured_data.html

[2] J. Johnson. 14 Nov 2012. Structured Data vs Unstructured Data. *KPI Partners News & Blog.* [Online] 3 Aug 2014. [Cited: 3 Aug 2014]. Available: http://www.kpipartners.com/blog/bid/137981/Structured-Data-vs-Unstructured-Data

[3] Sherpa Software. 26 Aug 2013. Structured and Unstructured Data: What is it? *Sherpa Software - Information Governance Solution*. [Online] 3 Aug 2014. [Cited: 3 Aug 2014]. Available: http://www.sherpasoftware.com/blog/structured-and-unstructured-data-what-is-it/

[4] MOM. Text Processing vs. Word Processors. *Macros for GNU troff*. [Online] 6 Aug 2014. [Cited: 6 Aug 2014]. Available: http://www.schaffter.ca/mom/mom-02.html

[5] R. C. Miller & A. M. Marshall . "Cluster-Based Find and Replace" in *Proc. Conference on Human Factors Computing Systems*, Vienna, Austria, Apr 2004, vol.6, no.1, pp. 57 – 64.

[6] S. K. Card, T. P. Moran, and A. Computer Newell. Text Editing: An Information-Processing Analysis of A Routine Cognitive Skill. Cognitive Psychology, 1980, 12, pp. 32-74.

[7] S. Bovair, D. E. Kieras, & P. G. Polson. The Acquisition and Performance of Text Editing Skill: A Cognitive Complexity Analysis, Human-Computer Interaction, Vol 5, Issue 1, Mar 1990, pp. 1 - 48

[8] S. W. Tyler, S. Roth, & T. Post. "The Acquisition of Text Editing Skills" in *Proc. Conference on Human Factors in Computing Systems*, 1982, pp. 324 - 325

[9] L. J. Folley and R. C. Williges. "User Models of Text Editing Command Languages" in *Proc. Conference on Human Factors in Computing Systems*, 1982, pp. 326 – 331

[10] J. M. Carroll & M. B. Rosson. 1987. Paradox of the active user. In J. M. Carroll (Ed.) Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, Bradford Books, pp. 80 -111

[11] T. L. Roberts & T. P. Moran. The Evaluation of Text Editors: Methodology and Empirical Results. Communications of the ACM . Vol. 26 Issue 4, Apr 1983, pp. 265-283

[12] L. Church & A. F. Blackwell . Structured Text Modification Using Guided Inference. PPIG, Lancaster 2008 . Available: **www.ppig.org/papers/20th-church.pdf**

[13] Y. Fujishima. “Demonstrational Automation of Text Editing Tasks Involving Multiple Focus Points and Conversions” in *Proc. 3rd International Conference Intelligent User Interfaces*, San Francisco, CA, USA, 1998, pp.101-108.

[14] R. C. Miller & B. A. Myers. "LAPIS: Smart Editing with Text Structure" in *Proc. Extended Abstracts Human Factors Computing Systems*, Minneapolis, Minnesota, USA, 2002, pp. 496 – 497

[15] LAPIS. Editing Text with Lightweight Structure. LAPIS. [Online] 9 Aug 2014. [Cited: 9 Aug 2014]. Available: **http://groups.csail.mit.edu/uid/lapis/index.html**

[16] LAPIS. A Tool for Lightweight Structured Text Processing. LAPIS. [Online] 9 Aug 2014. [Cited: 9 Aug 2014]. Available: **http://www.cs.cmu.edu/afs/cs.cmu.edu/project/amulet-3/rcm/lapis-0.96/**

[17] A. De Lorenzo, E. Medvet, & A. Bartoli. "Automatic String Replace by Examples" in *Proc. 15th Annual Conference Genetic and Evolutionary Computation*, 2013, Amsterdam, The Netherlands, pp. 1253 - 1260.

[18] E. Spishak, W. Dietl, & M. D. Ernst. “A Type System for Regular Expressions” in *Proc. 14th Workshop Formal Techniques for Java-like Programs*, Beijing, China, 2012, pp.20-26.

[19] J. Goyvaerts. Regular Expressions: The Complete Tutorial. Jul 2007. [E-book], Jul 2007. Available: http://www.princeton.edu/~mlovett/reference/Regular-Expressions

[20] Blackwell, A.F. 2001. SWYN: a visual representation for regular expressions. In Your Wish Is My Command: Programming By Example Morgan Kaufmann Publishers, San Francisco, CA, pp. 245-270.

[21] N. Popov. 15 Jun 2012. The True Power of Regular Expressions. Blog by nikic. [Online] 21 Aug 2014. [Cited: 21 Aug 2014]. Available: http://nikic.github.io/2012/06/15/The-true-power-of-regular-expressions.html

[22] J. Goyvaerts. 16 Sep 2013. The PCRE Open Source Regex Library. Regular Expressions.Info. [Online] 21 Aug 2014. [Cited: 21 Aug 2014]. Available: http://www.regular-expressions.info/pcre.html

[23] Wikipedia. 1 Aug 2014. Perl Compatible Regular Expressions. Wikipedia. [Online] 21 Aug 2014. [Cited: 21 Aug 2014]. Available: http://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions

[24] R. C. Miller, "Lightweight Structure in Text", Ph.D. dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 2002

[25] Just Great Software Co. Ltd. 16 Sep 2013. RegexBuddy. [Online] 23 Aug 2014. [Cited: 23 Aug 2014]. Available:(**http://www.regexbuddy.com**)

[26] D. W. Embley, G. Nagy. "Can we expect to improve text editing performance?" in *Proc. Conference on Human Factors in Computing Systems*, 1982, pp. 152 - 156

[27] PHP.net. PCRE Functions. [Online] 27 Aug 2014. [Cited: 27 Aug 2014]. Available: **http://php.net/manual/en/ref.pcre.php**

[28] Nix, R.P. Editing by example. ACM Transactions on Programming Languages and Systems. Vol. 7 Issue 4, Oct. 1985, pp. 600 - 621.

Appendix A: Personal Reflection

I would just be truthful to express my thoughts and feelings along the project period so it really means a pure reflection. Many students from previous dissertations suggested in their reflection part that it is important to choose a project that would bring the interest to ourselves, so did I advise myself the same way from the start to properly list my choice. This project has been attracting me since the beginning, in a matter that I am more interested with a development project compared to others since I want to polish my long unused programming skills, and my interest is more on development areas. I was lucky that I was assigned this project.

The story did not end there. The reality began when I have to deal with regular expressions and understand how it works. Something new to myself that I rarely heard of it previously. The regexes syntax rather seemed disturbing to me. I tried many times to understand it by reading tutorials and references, but maybe my heart was not present to grip. The issue of understanding regexes was mentioned quite frequently in many literature I found during background research and I was experiencing it myself. Despite the inner-self battle, I managed to develop the platform for basic search and replace, sequence transformation and task automation. The moment I ran the evaluation, was the moment I started to properly understand regexes, and it made me fall in love with it – testing it over and over again through some examples. Thus, I can relate here – even it is not highlighted in the content of the report, that the developed tool itself has been a training tool to myself. That's a real reflection.

Another reflection is on the intention to polish my programming skills. I was a web-based programmer since 2004. From 2008 when I changed my career to join the Government, the skills were seldom used. I was familiar in developing page to page interaction, where the form control (POST, GET) is more direct. But it was different when developing IFT, because interactions were processed in the same page and became a bit complex, so I had to properly design the flow and structure to avoid errors. To re-polish the skills was also a problem I have to face because I felt like I knew too little and have to start all over again. However, it was an interesting experience since I learn a lot of new things during the development. One thing I remembered that I managed to create the function to load details of each transformation just

by clicking on the radio button using JavaScript, which I never tried before. Online tutorials have supported me very much to regain my skills.

I was a person who did not really favour into reading journals and papers. The phase of doing background research, however turned me the other way around. I fell in love with the wide exploration of knowledge and findings in the research field, getting impressed and felt appreciative to what people had been doing all this while to contribute in the area. I was spending quite a long time reading many related journals because I became so attracted to the related subjects. From human-computer interaction aspects and deep exploration with techniques to solve repetitive text editing problem. At first, I thought that my work did not give any effect to compare with previous contributions, but their work had inspired me to stay through. Another thing is, reading journals and papers have given me the idea to style the research report. I am anticipating to do more research next time.

On the writing part, I found out that the provided report guideline as put on the MSc Project web page is a good sketch to follow. When I began writing, I could see the flow from one chapter to another. It has helped me a lot to construct the final report.

The most interesting part is on the evaluation, again. Apart from seeing my own application helped me to understand regexes, I was very impressed with the processing time during task automation on huge record. I am satisfied with the results because it takes less than 15 seconds to run on use cases which I did not initially expect at all. Another thing is during the comparison of walk-through the number of mouse clicks, I could see how to improve the interface in future. To my own reflection, the evaluation part has proved that to myself that this project is meaningful and valuable and has its own contribution even just a little.

I spent some times to reflect upon comments from both my supervisor and assessor from the Interim Report, Progress Meeting and and weekly meetings. They have supported me very much and gave many ideas to improve on the project. I took half or maybe more time from the project schedule to encourage myself that I was doing a correct thing was on the right track. I was struggling to really understand what I was doing. I guess, everyone was experiencing the same situation in the beginning, to finally see in the end that everything is worth to struggle for in the road to complete the MSc degree.

Appendix B: External Materials

Resources to test for implementation and evaluation.

[B1] Downloaded: 17 Aug 2014
Tested: 17 Aug 2014
Cited: 28 Aug 2014
Available:
http://samplecsvs.s3.amazonaws.com/Sacramentorealestatetransactions.csv

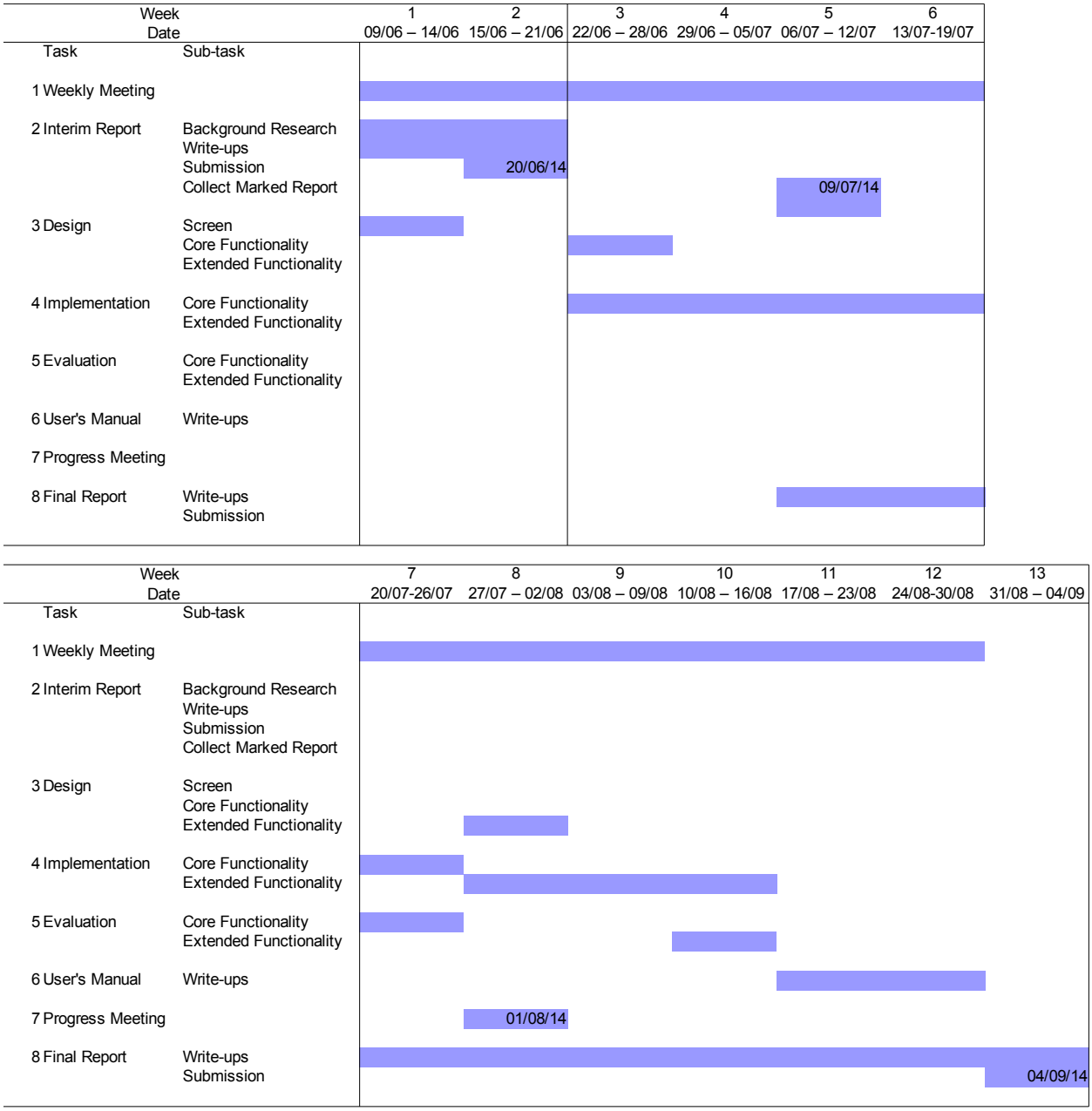
Appendix C: Ethical Issues

This appendix is the questionnare distributed to users to evaluate the objective of the application as a training tool.

Question				
How would you rate your knowledge of programming?	Advanced	Moderate	Beginner	No Experience
Are you familiar with regular expressions?	Very familiar	Heard of it but no idea	No idea at all	
How would you rate your knowledge of regular expressions?	Very Good	Basic	Minimal	None
Do you agree that the application can be a training tool for regular expressions?	Totally Agree	Neutral	Disagree	No idea

Appendix D: Gantt Chart

Original Project Schedule



Appendix E: Use Case

This appendix is the use cases for this project and used for Implementation and Evaluation.

Use Case 1: Convert CSV Data into HTML Table (1)

CSV Data

```
street,city,zip,state,beds,baths,sq__ft,type,sale_date,price,latitude,longitude
4712 PISMO BEACH DR,ANTELOPE,95843,CA,5,3,2346,Residential,Mon May 19 00:00:00 EDT 2008,320000,38.707705,-121.354153
4741 PACIFIC PARK DR,ANTELOPE,95843,CA,5,3,2347,Residential,Mon May 19 00:00:00 EDT 2008,325000,38.709299,-121.353056
310 GROTH CIR,SACRAMENTO,95834,CA,4,2,1659,Residential,Mon May 19 00:00:00 EDT 2008,328578,38.638764,-121.531827
```

The expected output is a set of transformed strings as a HTML table as below:

```
<table border='1'>
<tr><td>street</td><td>city</td><td>zip</td><td>state</td><td>beds</td><td>baths</td><td>sq__ft</td><td
>type</td><td>sale_date</td><td>price</td><td>latitude</td><td> longitude</td></tr>
<tr><td>4712 PISMO BEACH DR </td><td>ANTELOPE</td><td>95843</td><td>CA</td><td>5</td><td>3
</td><td>2346</td><td>Residential</td> <td>Mon May 19 00:00:00 EDT 2008</td><td>320000</td>
<td>38.707705</td><td> -121.354153</td></tr>
</table>
```

Substitution Patterns

1. Substitute the first row (column names)
   Match: (\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+),(\w+)
   Replace: <tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td><td>$6</td><td>$7</td><td>$8</td><td>$9</td><td>$10</td><td>$11</td><td> $12</td></tr>

2. Substitute data rows except the first rows
   Match: (\d+ \w+.*),(\w+.*),(\d{5}),(\w+),(\d+),(\d+),(\d+),(\w+.*),(\w{3} \w{3} \d+00:00:00 EDT \d{4}),(\d+),(\d+.\d+),(-\d+.\d+)
   Replace: <tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td><td>$6</td><td>$7</td><td>$8</td><td>$9</td><td>$10</td><td>$11</td><td> $12</td></tr>

3. Substitute the beginning of the source string with start table tag with border
   Match: ^
   Replace: <table border='1'>

4. Substitute the end of the source string with the end table tag
   Match: $
   Replace: </table>

Use Case 2: Convert CSV Data into HTML Table (2)

CSV Data

Stu_ID,Stu_Name,Birth_Year,Country,Marital_Status
1001,Nurul Ismail,1983,Malaysia,Married
1009,Nik Ibrahim,1986,Malaysia,Single

The expected output is a set of transformed strings as a HTML table as below:

```
<table border='1'>
<tr><td>Stu_ID</td><td>Stu_Name</td><td>Birth_Year</td><td>Country</td><td>Marital_Status</td></tr>
<tr><td>1001</td><td>Nurul Ismail</td><td>1983</td><td>Malaysia</td><td>Married</td></tr>
<tr><td>1002</td><td>Yasmeen Al Barak</td><td>1984</td><td>Saudi Arabia</td><td>Married</td></tr>
</table>
```

Substitution Patterns

1. Substitute the first row (column names)
   Match: (\w+),(\w+),(\w+),(\w+),(\w+)
   Replace: <tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td></tr>

2. Substitute data rows except the first rows
   Match: (\d+),(\w+\s+.*),(\d+),(\w+.*),(\w+)
   Replace: <tr><td>$1</td><td>$2</td><td>$3</td><td>$4</td><td>$5</td></tr>

3. Substitute the beginning of the source string with start table tag with border
   Match: ^
   Replace: <table border='1'>

4. Substitute the end of the source string with the end table tag
   Match: $
   Replace: </table>

Use case 3: Convert HTML table to CSV data

HTML data

```
<table border='1'>
<tr><td>Stu_ID</td><td>Stu_Name</td><td>Birth_Year</td><td>Country</td><td>Marital_Status</td></tr>
<tr><td>1001</td><td>Nurul Ismail</td><td>1983</td><td>Malaysia</td><td>Married</td></tr>
<tr><td>1002</td><td>Yasmeen Al Barak</td><td>1984</td><td>Saudi Arabia</td><td>Married</td></tr>
<tr><td>1009</td><td>Nik Ibrahim</td><td>1986</td><td>Malaysia</td><td>Single</td></tr>
</table>
```

Exprected Output in CSV format:

```
Stu_ID,Stu_Name,Birth_Year,Country,Marital_Status
1001,Nurul Ismail,1983,Malaysia,Married
1002,Yasmeen Al Barak,1984,Saudi Arabia,Married
```

Substitution Patterns

1. Substitute the start table tag with nothing
   Match: `<table border='1'>`
   Replace:

2. Substitute the end table tag with nothing
   Match: `</table>`
   Replace:

3. Substitution </td><td> tags with nothing
   Match: `<\/td><td>`
   Replace:

4. Substitution <tr><td> tags with nothing
   Match: `<tr><td>`
   Replace:

5. Substitute </td></tr> with nothing
   Match: `<\/td><\/tr>`
   Replace:

Use case 4: Convert VCF data to HTML

Sample VCF

```
BEGIN:VCARD
VERSION:4.0
N:Ismail;Nurul;;;
FN:Nurul Ismail
ORG:University of Leeds
TITLE:Computer Science Student
PHOTO;MEDIATYPE=image/gif:http://www.example.com/dir_photos/my_photo.gif
TEL;TYPE=work,voice;VALUE=uri:tel:+07459129491
TEL;TYPE=home,voice;VALUE=uri:tel:+07459129491
ADR;TYPE=work;LABEL='100 School of Computing\nUniversity of Leeds\nUnited Kingdom'
  :;;100 School of Computing;University of Leeds;United Kingdom
ADR;TYPE=home;LABEL='11 Kelso Road\nLeeds, LS29PR\nUnited Kingdom'
 :;;42 Kelso Road;Leeds;LS29PR;United Kingdom
EMAIL:sc13nmbi@leeds.ac.uk
REV:20080424T195243Z
END:VCARD
```

Expected Output in HTML format

```
<table border='1'>
<tr><td>FULL NAME</td><td>Nurul Ismail</td><tr>
<tr><td>ORG</td><td>University of Leeds</td></tr>
<tr><td>TITLE</td><td>Computer Science Student</td></tr>
<tr><td>PHOTO</td><td>http://www.example.com/dir_photos/my_photo.gif</td></tr>
<tr><td>TELEPHONE (work)</td><td>+07459129491</td></tr>
<tr><td>TELEPHONE (home)</td><td>+07459129491</td></tr>
<tr><td>ADDRESS (work)</td><td>100 School of Computing<br>University of Leeds<br>United
Kingdom</td></tr>
<tr><td>ADDRESS (home)</td><td>11 Kelso Road<br>Leeds, LS29PR<br>United Kingdom</td></tr>
<tr><td>EMAIL</td><td>sc13nmbi@leeds.ac.uk</td></tr>
<tr><td>REV</td><td>24/04/2008 - 19:52:43</td></tr>
</table>
```

Substitution Patterns

1. Substitute BEGIN:VCARD with open table tag
   Match: BEGIN:VCARD
   Replace: <table border='1'>

2. Substitute END:VCARD with close table tag
   Match: END:VCARD
   Replace: </table>

3. Remove version, substitute with nothing
   Match: VERSION:+\w+.*+\n
   Replace:

4. Remove N tag in VCF, substitute with nothing
   Match: N:\w+;.*\n
   Replace:

5. Reformat FN tag and its information
Match: (FN):(\w+.*)\r
Replace: <tr><td>FULL NAME</td><td>$2</td><tr>

6. Reformat tags
Match: ([A-Z]+.*[^http]):
Replace: <tr><td>$1</td>:

7. Reformat information
Match: :(\w+.*)\r
Replace: <td>$1</td></tr>

8. Remove the second format of address
Match: \s+:;;\d+ \w+.*
Replace:

9. Reformat address tag and its information
Match: (ADR);TYPE=(\w+);LABEL='(\d+ \w+.*)'
Replace: <tr><td>ADDRESS ($2)</td><td>$3</td></tr>

10. Change \n (new line tag in VCF) to
 (new line tag in HTML)
Match: \\n
Replace:

11. Reformat Telephone tag and its information
Match: (TEL);TYPE=(\w+),\w+.*<\/td>:
Replace: TELEPHONE ($2)</td><td>

12. Remove ;MEDIATYPE=image\/gif
Match: ;MEDIATYPE=image\/gif
Replace:

13. Reformat Date
Match: (\d{4})(\d{2})(\d{2})T(\d{2})(\d{2})(\d{2})Z
Replace: $3/$2/$1 - $4:$5:$6

Appendix F: Walk-through Evaluation

This appendix is the details on walk-through evaluation based on 3 applications and compared with IFT. The use case as in Appendix E is referred together.

Task 1: Convert CSV Data into HTML Table
This task can be either from Use Case 1 or Use Case 2, it does not make any difference because the substitution patterns are the same.

Application 1: RegexBuddy
Regex buddy has the ability to view substitutions automatically without submit button. It is in the style of WYSIWYG. Once substitution patterns and source are put into the application, it will automatically show the transformation. Once replace mode is chosen on the menu, it will stay until the transformation finishes. This is and advantage to save the number of clicks as it maintains the mode compared to IFT, which user needs to choose "replace" almost every time each step is applied.

To perform the task, the steps are follows:-

Steps		**Number of clicks**
1	The system will automatically initiate a sequence named Regex 1.	0
2	Choose "Replace" mode from the menu on the top	1
3	Paste the initial source	3
4	Paste match pattern 1	3
5	Paste replace pattern 1	3
6	Ouput for substitution 1 is automatically displayed	0
7	Copy the result of substitution 1	3
8	Click add sequence (Regex 2)	1
9	Paste the result of substitution 1 as source	3
10	Paste match pattern 2	3
11	Paste replace pattern 2	3
12	Ouput for substitution 2 is automatically displayed	0
13	Copy the result of substitution 2	3
14	Click add sequence (Regex 3)	1
15	Paste the result of substitution 2 as source	3
16	Paste match pattern 3	3
17	Paste replace pattern 3	3
18	Ouput for substitution 3 is automatically displayed	0

19	Copy the result of substitution 3	3
20	Click add sequence (Regex 4)	1
21	Paste the result of substitution 3 as source	3
22	Paste match pattern 4	3
23	Paste replace pattern 4	3
24	Ouput for substitution 4 is automatically displayed	0
Total number of clicks		**49**

The process is iterative except in the beginning, the first sequence named Regex 1 is initialised by the programme, the "Replace" mode is only turned on once and the final result does not need to be copied anymore.

The number of clicks can be analysed as follows:

- Total number of substitution patterns is 4.
- Each substitution requires 13 clicks.
- 4 steps does not need click count (initialisation of Regex 1 and the final result).
- 1 click to turn on "Replace" mode.
- Total number of clicks are (4 X 13) – 4 + 1 = 49.

  Hence, it can be concluded in a formula as follows:
  The number of clicks to perform a task using RegexBuddy =
  (The number of substitution patterns * 13) – 3.

Application 2: My Regex Tester

My Regex Tester is an online tool to evaluate regex substitutions. The first time user wants to apply substitution, he should select "replace" on the operation which will require 2 cliks; one for scroll and another for select. This will explicitly adds 2 clicks to overall transformation. On almost every transformation, user needs to switch tabs between Source, Results, Match Pattern and Replace Pattern. This is a drawback using tabs that will consume more mouse clicks. The steps taken to perform the task are as below:

Steps		**Number of clicks**
1	Paste the initial source on source tab	3
2	Paste match pattern 1 on Match tab	3
3	Choose operation "replace"	2
4	Click on "Replace" tab and paste replace pattern 1	4
5	Click submit button	1
6	Copy the result of substitution 1 from Results tab	3
7	Click Source tab and paste the result of substitution 1 as source	4

8	Paste match pattern 2 on Match tab	3
9	Click on "Replace" tab and paste replace pattern 2	4
10	Click submit button	1
11	Copy the result of substitution 2 from Results Tab	3
12	Click Source tab and paste the result of substitution 2 as source	4
13	Click on "Replace" tab and paste replace pattern 3	4
14	Click submit button	1
15	Copy the result of substitution 3 from Results tab	3
16	Click Source tab and paste the result of substitution 3 as source	4
17	Paste match pattern 4 on Match tab	3
18	Click on "Replace" tab and paste replace pattern 4	4
19	Click submit button	1
21	Final result will be displayed on Result tab	0
Total number of clicks		**58**

My Regex Tester consumes more mouse clicks due to switching between tabs and submitting the form, while the rest of the other clicks are similar to RegexBuddy that needs copying of the previous result as current source.

The number of clicks can be analysed as follows:

- Total number of substitution patterns is 4.
- Each substitution requires 15 clicks.
- 2 clicks to turn on "Replace" operation.
- The first step does not need to click Source tab. Minus 1 click.
- Final result does not need to be copied. Minus 3 clicks.
- Total number of clicks are (4 X 15) + 2 - 4 = 58.

Hence, in a formula it can be concluded as follows:
The number of clicks to perform a task using My Regex Tester =
(The number of substitution patterns * 15) – 2.

Application 3: CSV Converter
The application only needs two steps with a total of 4 clicks. Pasting the source consumes 3 clicks and submitting the form needs 1 click and the transfomation is completely done.

Interactive File Transformer
This is the walk-through of the developed application for this project:

Steps		Number of clicks
1	Create new task	1
2	Insert task name	1
3	Click submit button	1
4	Click load task	1
5	Paste the initial source	3
6	Paste match pattern 1 on Match tab	3
7	Click on Replace tab and paste replace pattern 1	4
8	Choose operation "replace"	2
9	Click submit button	1
10	Save the result of substitution 1	1
11	Load the result of substitution 1 as source using radio button	1
12	Paste match pattern 2 on Match tab	3
13	Click on Replace tab and paste replace pattern 2	4
14	Choose operation "replace"	2
15	Click submit button	1
16	Save the result of substitution 2	1
17	Load the result of substitution 2 as source using radio button	1
18	Paste match pattern 3 on Match tab	3
19	Click on Replace tab and paste replace pattern 3	4
20	Choose operation "replace"	2
21	Click submit button	1
22	Save the result of substitution 3	1
23	Load the result of substitution 3 as source using radio button	1
24	Paste match pattern 3 on Match tab	3
25	Click on Replace tab and paste replace pattern 3	4
26	Choose operation "replace"	2
27	Click submit button	1
28	Final result is displayed on Result tab	0
Total number of clicks		**53**

IFT saves click during reloading the previous results as source. However, 2 clicks are consumed during each substitution for choosing operation "replace". The idea of using Match and Replace tabs also contribute to 1 click on each substitution.

The number of clicks can be analysed as follows:

- Total number of substitution patterns is 4.
- 4 Fixed clicks from creating and loading a task.
- Each substitution makes 12 clicks.
- Final result would not be copied as next source. Minus 1 click.
- First intial source is by 'paste' action thus it makes 2 extra clicks .
- Total number of clicks are (4 X 12) + 4 + 2 – 1 = 53.

  Hence, in a formula it can be concluded as follows:
  The number of clicks to perform a task using IFT =
  (The number of substitution patterns * 12) – 5.

Task 2: Convert HTML table to CSV data

This task is referring to Use Case 3 in Appendix E.

Since there are no replacement strings in this use case, this evaluation will place two kinds of scenarios. The first one is specific to this task where click for 'replace' is not considered (best case) and the other one is for general situation where 'replace' steps are still applicable.

In RegexBuddy, the 'replace' step requires 3 clicks while in My Regex Tester and IFT, each requires 4 clicks including clicking on the Replace tab. For evaluation specific to this task, Regex Buddy saves (5 X 3) clicks totalling 15. My Regex Tester and IFT each saves (5 X 4) 20 clicks.

Scenario 1: Total number of clicks in general task
Scenario 2: Total number of clicks specific to the task

Application 1: RegexBuddy
Formula: (The number of substitution patterns * 13) – 3
Scenario 1: 62 clicks
Scenario 2: 62 – 15 = 47 clicks

Application 2: My Regex Tester
Formula: (The number of substitution patterns * 15) – 2
Scenario 1: 73 clicks
Scenario 2: 73 – 20 = 53 clicks

Application 3: CSV Converter
Both for situation 1 and 2 consumes 4 clicks

Interactive File Transformer
Formula: (The number of substitution patterns * 12) – 5.
Scenario 1: 55 clicks
Scenario 2: 55 – 20 = 35 clicks

Task 3: Convert VCF data to HTML

This task is referring to Use Case 4 in Appendix E.

Four substitution patterns in this task do not need any 'replace' action. Again, two scenarios are considered. In RegexBuddy, (3 * 4) clicks are saved totalling 12, while My Regex Tester and IFT each saves (4 * 4) totalling 16.

Scenario 1: Total number of clicks in general task
Scenario 2: Total number of clicks specific to the task

Application 1: RegexBuddy
Formula: (The number of substitution patterns * 13) – 3
Scenario 1: 166 clicks
Scenario 2: 166 – 12 = 154 clicks

Application 2: My Regex Tester
Formula: (The number of substitution patterns * 15) – 2
Scenario 1: 193 clicks
Scenario 2: 193 – 16 = 177 clicks

Application 3: CSV Converter
This application cannot convert VCF to HTML.

Interactive File Transformer
Formula: (The number of substitution patterns * 12) – 5.
Scenario 1: 151 clicks
Scenario 2: 151 – 16 = 135 clicks

Appendix G: Screen

This appendix shows the print screen of 3 different applications during the evaluation to compare with IFT on their walk-through of the number of mouse clicks used to perform transformation.

Application 1: RegexBuddy

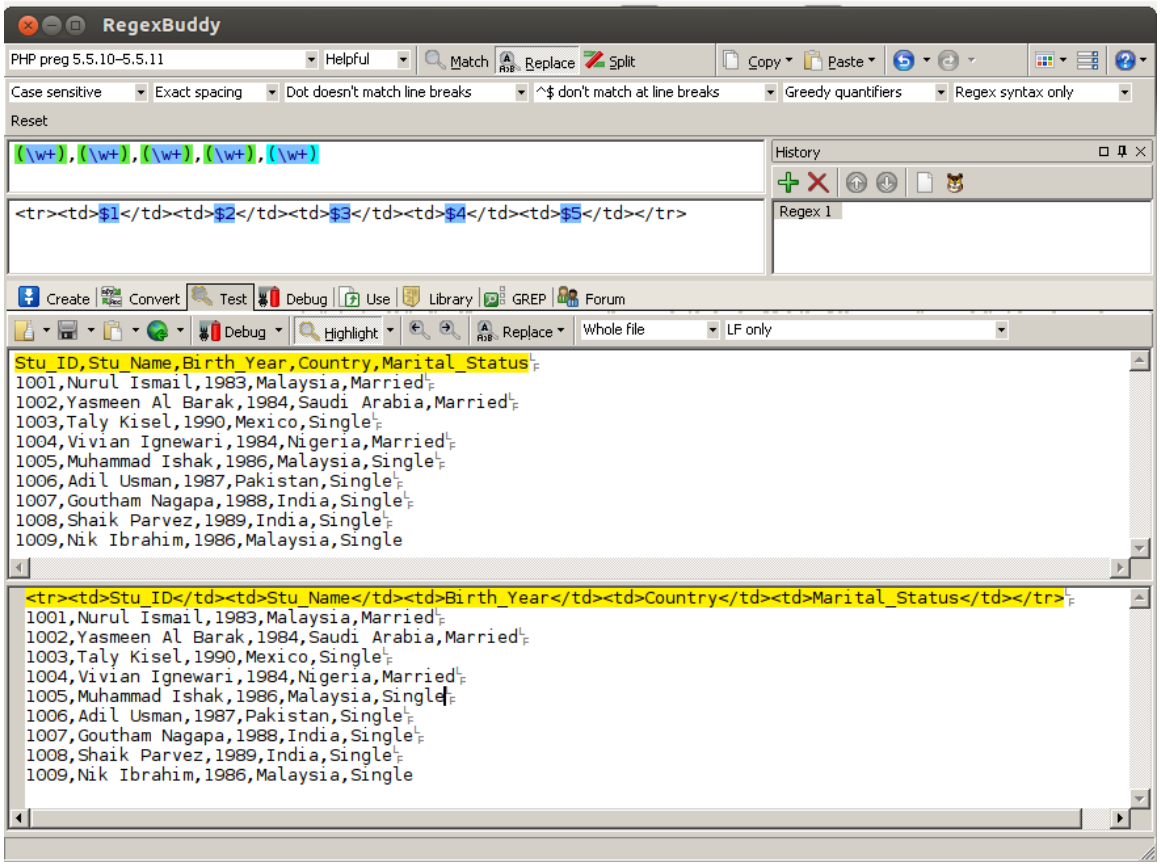


Figure F.7:RegexBuddy performing substition pattern 1 on Use Case 2

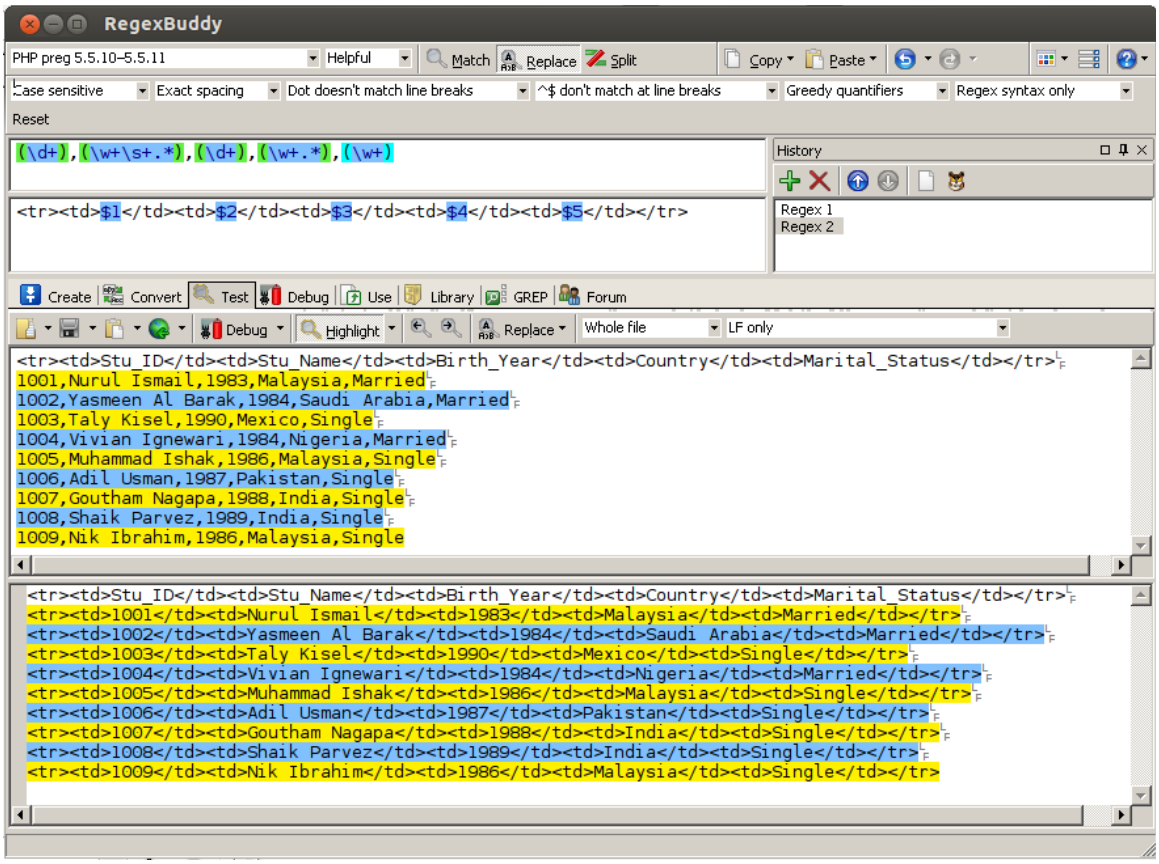


Figure F.8: RegexBuddy performing substition pattern 2 on Use Case 2

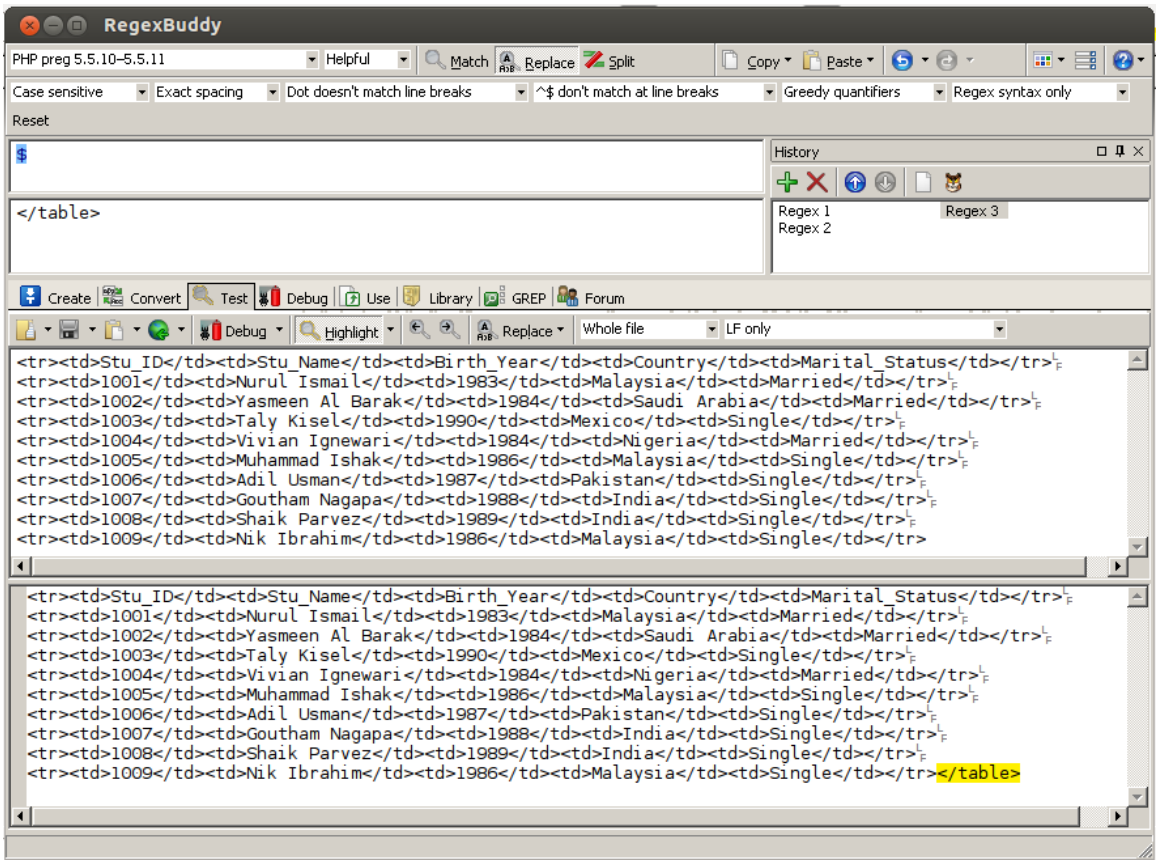


Figure F.9: RegexBuddy performing substition pattern 3 on Use Case 2

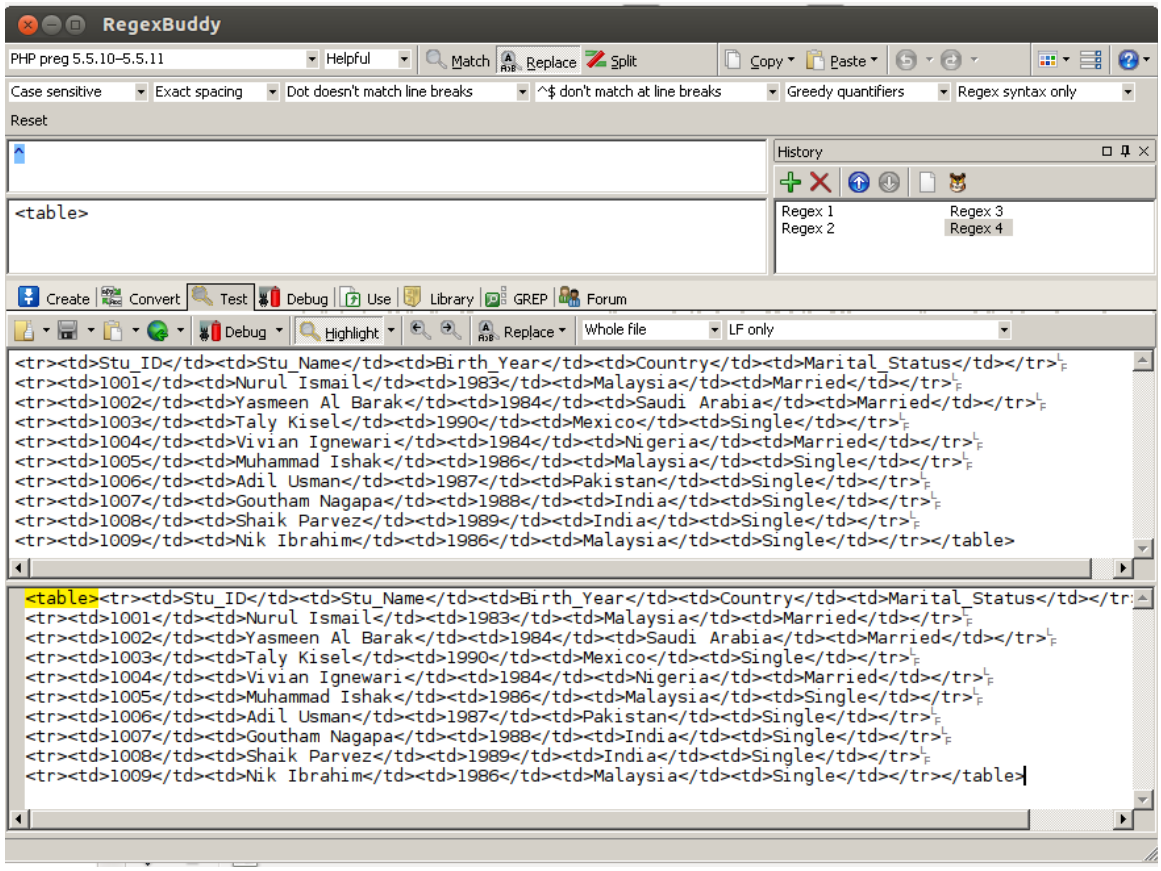


Figure F.10: RegexBuddy performing substition pattern 4 on Use Case 2

Application 2: My Regex Tester

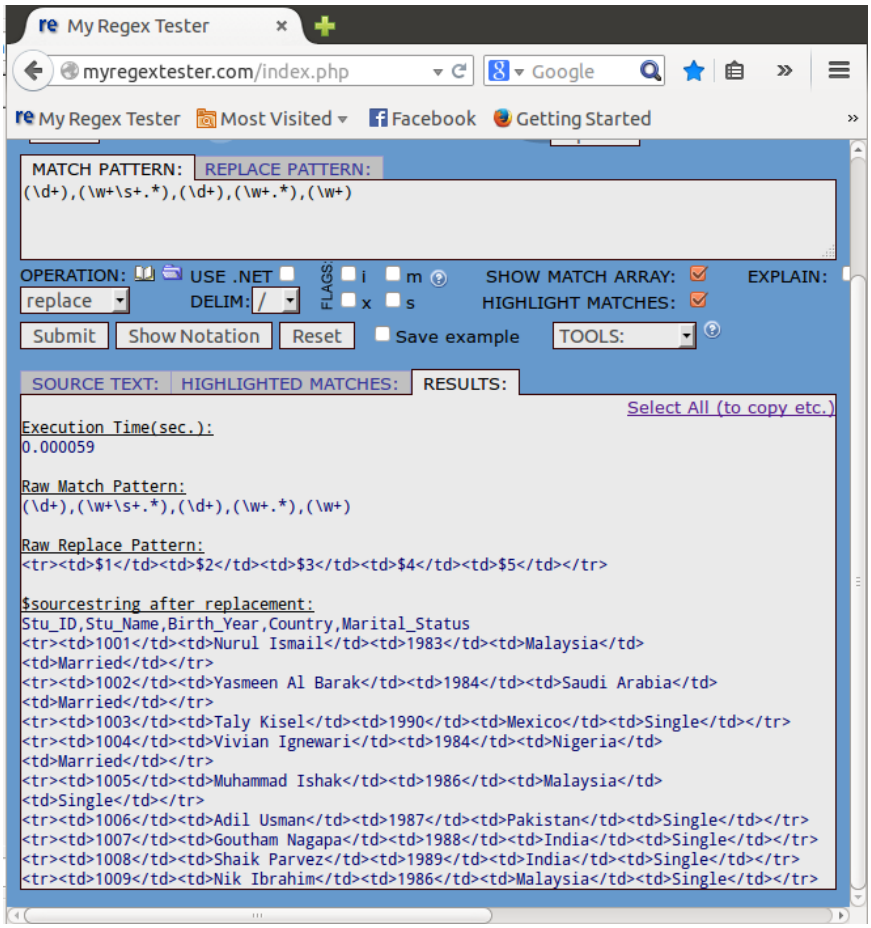


Figure F.11: MyRegex Tester performing substition pattern 2 on Use Case 2

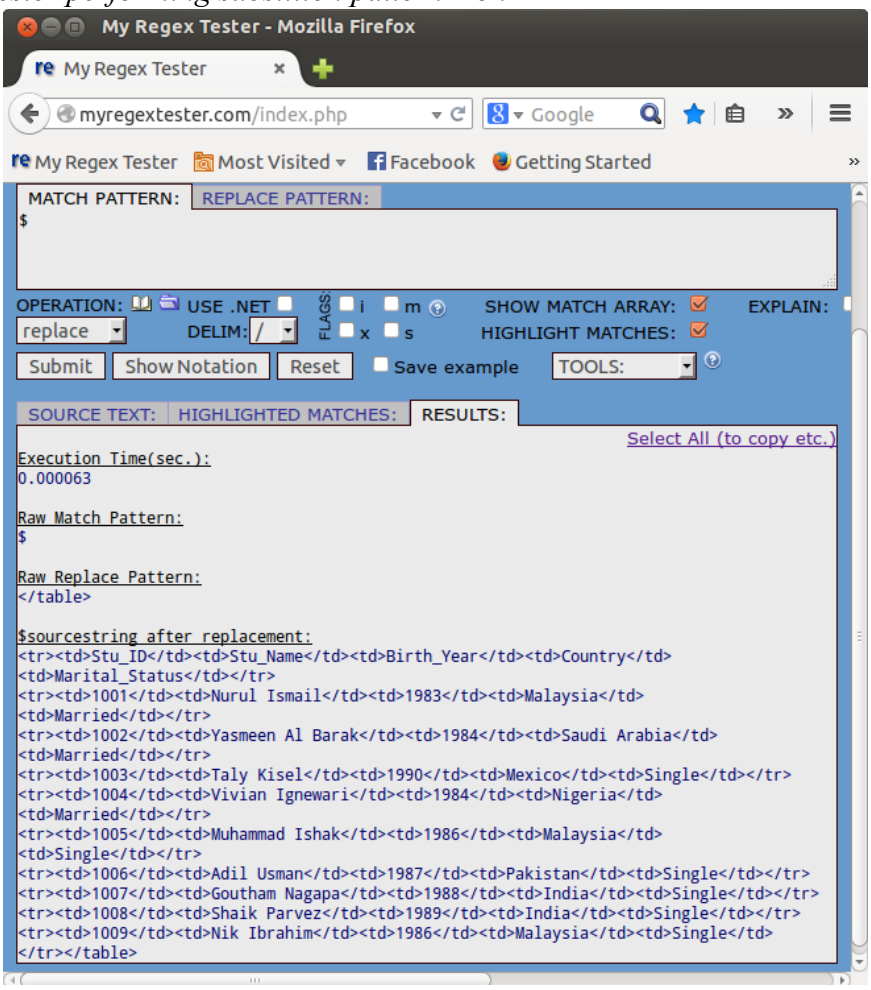


Figure F.12: MyRegex Tester performing substition pattern 4 on Use Case 2

Application 3: CSV Converter

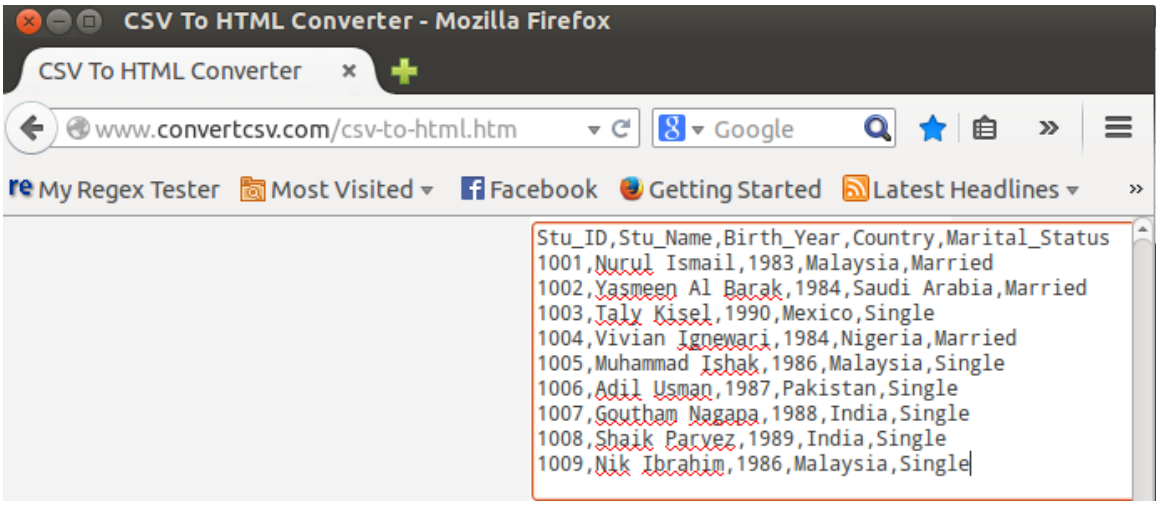
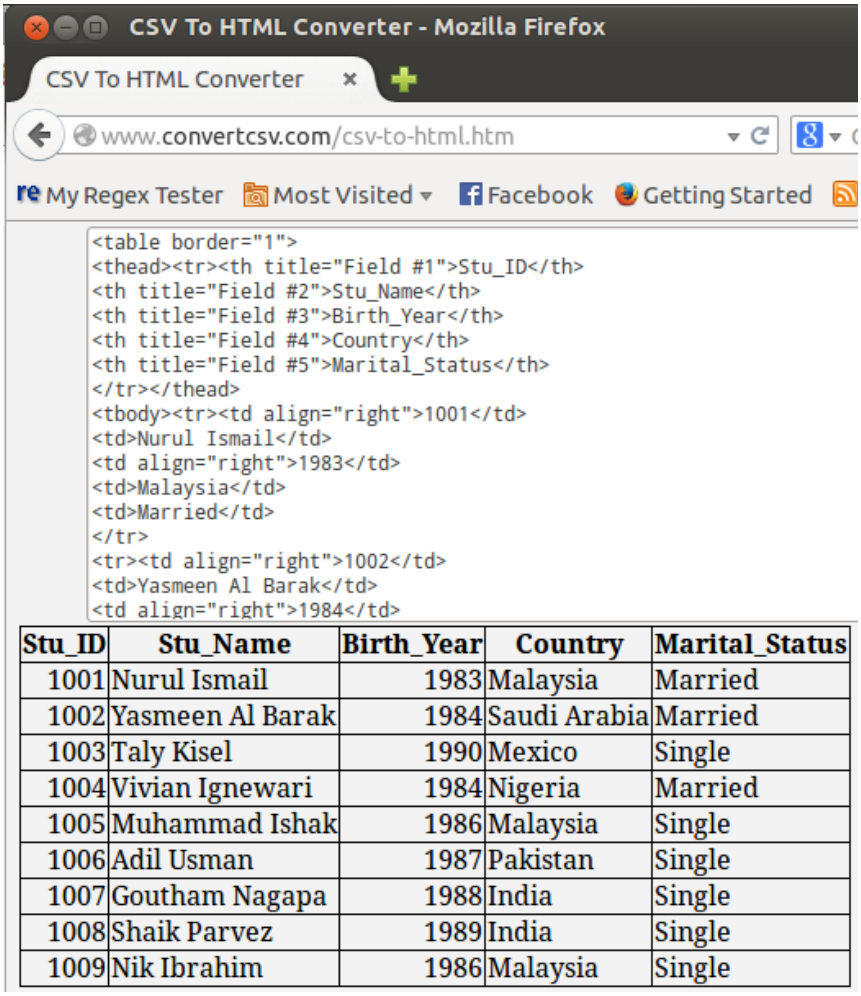


Figure F.13: CSV Converter performing transformation on Use Case 2



Stu_ID	Stu_Name	Birth_Year	Country	Marital_Status
1001	Nurul Ismail	1983	Malaysia	Married
1002	Yasmeen Al Barak	1984	Saudi Arabia	Married
1003	Taly Kisel	1990	Mexico	Single
1004	Vivian Ignewari	1984	Nigeria	Married
1005	Muhammad Ishak	1986	Malaysia	Single
1006	Adil Usman	1987	Pakistan	Single
1007	Goutham Nagapa	1988	India	Single
1008	Shaik Parvez	1989	India	Single
1009	Nik Ibrahim	1986	Malaysia	Single

Figure F.14: Results of Transformation of Use Case 2 by CSV Converter

Appendix H: Reusability Performance

This appendix shows the execution time for four use cases. Appendix E is referred together. In each use case, some data records between 1000 to 5000 are tested to run upon the predefined sequences according to their transformation type. The sample data in each use case is repeated to create sufficient amount of record. Each test is run for five times on Mozilla Firefox and Google Chrome web browser. Based on the results, the average execution time can be estimated thus proving the reusability performance of the project.

Use case 1: CSV to HTML 1

		Execution Time (seconds)					
Browser	**Number of records**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
Chrome	**1000**	0.380	0.481	0.337	0.338	0.378	**0.3828**
Firefox	**1000**	0.606	0.427	0.490	0.481	0.493	**0.4994**
Chrome	**5000**	2.021	2.062	2.113	2.038	2.063	**2.0594**
Firefox	**5000**	1.985	2.074	1.997	1.983	2.055	**2.0188**
Chrome	**10000**	4.125	4.082	4.142	4.111	4.106	**4.1132**
Firefox	**10000**	3.780	3.857	3.652	3.756	3.857	**3.7804**

Use case 2: CSV to HTML 2

		Execution Time (seconds)					
Browser	**Number of records**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
Chrome	**1000**	0.202	0.202	0.200	0.182	0.198	**0.1968**
Firefox	**1000**	0.204	0.261	0.196	0.194	0.165	**0.2040**
Chrome	**5000**	0.836	0.883	0.859	0.843	0.879	**0.86 00**
Firefox	**5000**	0.760	0.693	0.729	0.734	0.702	**0.7236**
Chrome	**10000**	1.675	1.746	1.434	1.771	1.756	**1.6764**
Firefox	**10000**	1.434	1.416	1.484	1.425	1.448	**1.4414**

Use case 3: HTML to CSV

		Execution Time (seconds)					
Browser	**Number of records**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
Chrome	**1000**	0.211	0.278	0.256	0.239	0.257	**0.248**
Firefox	**1000**	0.186	0.197	0.244	0.206	0.208	**0.208**
Chrome	**5000**	1.006	1.037	1.059	1.045	0.999	**1.029**
Firefox	**5000**	0.819	0.841	0.805	0.679	0.816	**0.792**
Chrome	**10000**	2.042	2.048	2.033	2.062	2.084	**2.054**
Firefox	**10000**	1.590	1.586	1.567	1.490	1.617	**1.570**

Use case 4: VCF to HTML

		Execution Time					
Browser	**Number of records**	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
Chrome	**1000**	3.962	3.895	3.724	3.895	3.748	3.8448
Firefox	**1000**	2.930	2.857	2.972	0.329	3.032	2.4240
Chrome	**3000**	11.209	11.228	11.30	11.50	11.512	11.350
Firefox	**3000**	9.3670	9.277	9.147	9.169	9.107	9.2134