A Tool for the Conversion of a Graphic Intensive Website to a Text Browser Friendly Website

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Gaurav Gaur
Spring 2006

Committee Members

Dr. Dulal C. Kar, Ph.D.
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
A Tool for the Conversion of a Graphic Intensive Website to a Text Browser Friendly Website

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Gaurav Gaur
Spring 2006

Committee Members

Dr. Dulal C. Kar, Ph.D.
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
ABSTRACT

The use of computers in today’s life, and World Wide Web being the source of vast array of information, accounts for a need to ensure that the design of Web pages embraces the Universal Design requirements. Universal Design calls for design flexible enough to accommodate the accessibility needs of the broadest range of computer users, and users of other telecommunications equipment, regardless of age or disability. This project is an effort to design a software tool to convert a graphic intensive web site into a text browser friendly format. It allows user to browse to the text browser Webpage which it accepts as an input, parse it and convert it into a text browser friendly Website. While doing so the requirements for the text browser friendly Website are implemented programmatically, and as such the layout of the Website, filtering of the images, list and other issues are handled automatically. Once the parsing is done the user is provided with an option to view the parsed Webpage with the text browser, Web browser, and the text pad.
# TABLE OF CONTENTS

Abstract ........................................................................................................ii

Table of Contents ..........................................................................................iii

List of Figures ..................................................................................................v

1. Introduction and Background .................................................................1

2. A Tool for the Conversion of a Graphic Intensive Website to a
   Text Browser Friendly website ...............................................................10
   2.1 Overview .............................................................................................10
       2.1.1 Web Accessibility and the law .......................................................10
       2.1.2 Screen Readers, Tool for Visually Impaired ......................... 12
   2.2 Features ..............................................................................................13
   2.3 Description of User Interface ............................................................13

3. System Design .........................................................................................17
   3.1 Initial Requirement Analysis ...............................................................17
   3.2 Systems Used to Develop this Project ..............................................18
   3.3 Design Process ..................................................................................19
   3.4 Data Flow Diagram and Flow Chart ................................................20
   3.5 Concept of Parsing ............................................................................22
   3.6 HTML Parser ....................................................................................23
   3.7 List of Major Files .............................................................................23
   3.8 Implementation ..................................................................................24

4. Testing and Evaluation ............................................................................28
   4.1 Usability Testing ...............................................................................28
5. Conclusion and Future Work ................................................................. 38
   5.1 Future Work .................................................................................. 39
6. Bibliography and References ............................................................... 40
7. APPENDIX A: Data Dictionary of High-Level Data Flow
   Diagram and Flow Chart of the system Bibliography and ......................... 41
8. APPENDIX B: Data Dictionary of the basic HTML tags ......................... 42
LIST OF FIGURES

Figure 1.1 Graphic Intensive Website with Internet Explorer 6.0.................................3
Figure 1.2 Graphic Intensive Website with Lynx (DOS based browser).......................5
Figure 1.3 Text Browser Friendly Website with Internet Explorer 6.0.........................6
Figure 1.4 Text Browser Friendly Website with Lynx (DOS based browser).................8
Figure 2.1 Initial User Interface Screen ........................................................................14
Figure 2.2 Final User Interface Screen ........................................................................15
Figure 3.1 High Level Data Flow Diagram....................................................................20
Figure 3.2 Flow Chart of the System............................................................................21
Figure 3.3 List of Major Files......................................................................................24
Figure 4.1 EH&S Graphic Intensive Website with Firefox 1.0.7....................................29
Figure 4.2: EH&S Graphic Intensive Website with Lynx............................................30
Figure 4.3: EH&S Website after parsing......................................................................31
Figure 4.4: The Mercury News Graphic Intensive Website with IE 6.0.....................33
Figure 4.5: Mercury News Website after parsing with Lynx....................................37
1. INTRODUCTION AND BACKGROUND

The former president Clinton once said “In a competitive global economy, our country does not have a single person to waste. Opportunity must be open to everyone... I believe our entire nation will share in the economic and social benefits that will result from full participation of Americans with disabilities in our society” [- President Clinton, 12/1/1992]. This holds true for the over all development of any country.

As per definition, Web accessibility is the ability for a person using any agent (software or hardware that retrieves and renders Web content) to understand and fully interact with a Website's content [Sierkowski, 2002]. This implies that the Website design, layout and the elements used should incorporate those features which can be optimally interpreted by all the available set of browsers.

Web accessibility in many countries has become a concern, governed by legal and practical reasons, and as such laws are being introduced in legislation to avoid Website accessibility discrimination based on handicap.

With the rapid advancement in the World Wide Web from a text-based communication format to a robust and embellished graphic format including audio and video clip tools, there is a significant number of population whose accessibility is hindered by this development. These sections of the population are visually challenged people, aged people with problems with vision, and illiterate people. Today these people can independently access Websites, and surf information on the Internet with the aid of relatively inexpensive screen readers or Braille translators connected to the personal computer. However, the aforementioned technical aids are limited to little use with the
advancement in the Website designing, which now is more graphic intensive, in order to
design an attractive interface. This design or format of the Webpage, for instance may
distort the complete layout of the Webpage making it almost impossible and frustrating
for the visually challenged user to browse.

The design of the Website commonly include HTML (hypertext markup language)
combined with multimedia to facilitate the access to a vast network of educational,
governmental and commercial resources. However, the multimedia nature of the medium
imposes certain constraints on the visually impaired surfers to navigate certain features.

The problem areas may include [Pompano, 2005]:

- Inability to see graphics due to visual impairments
- Intricacies involved in navigation of clumsily organized Websites with imprecise
  links and directions.
- Difficulty in the use or availability of adaptive technology with computer to
  access the Website.

Apart from the compatibility issues in the graphic intensive Website and the text
browser friendly Website, there are visual differences also. The figures which follow give
an idea of this.

Figure 1.1 shows a snapshot of a graphic intensive Webpage of the Webb County
Appraisal District, Laredo, Texas, when viewed with Internet Explorer 6.0. In this
graphic intensive Website, tables have been used to organize the layout of the Webpage
as intended and low resolution images have been used extensively for visual appeal.
While the Webpage looks fine when viewed in the Web browser, however, it is not
interpreted optimally when viewed with a text browser, as discussed later in the section.
Figure 1.1 Graphic Intensive Website with Internet Explorer 6.0

Figure 1.2 shows a snapshot of the same webpage when viewed with a text browser. The text browser used is a DOS based browser, Lynx v2.8.3. The graphic intensive Website in Figure 1.1 is designed using tables, which helps in organizing the layout of the webpage. When viewed with the text browser the tables are not interpreted in the intended way i.e., the text browser interprets the webpage by reading the sentences in the same row from different columns as a
single sentence, thereby distorting the whole layout of the Webpage, and making it extremely frustrating and literally impossible for the visually handicapped user to browse. In addition to this the text browser is not able to interpret the images which do not have an “ALT” (attribute of IMG tag in Hyper-Text Markup Language i.e. HTML). The images are discarded by the text browser and the description of the image as specified by the ALT tag is retained. Thus, if the image does not contain the ALT tag, the name of the image will be captured by the text browser, thereby resulting in insufficient information for the visually handicapped user to analyze it correctly.
Figure 1.2 Graphic Intensive Website with Lynx (DOS based browser)

Figure 1.3 shows a text browser friendly-version of the same Webpage when viewed with Internet Explorer 6.0. The tabular layout of the page is designed in a way which is properly interpreted by the text browser. The Webpage is designed with the minimum number of images (only one in this case), as the text browser is not able to interpret the images. In addition, it is ascertained that the images retained do have an “ALT” tag, with a proper description of the image, so that the
text browser can comprehend it optimally. In addition, none of the images serves as a hyperlink to other Webs...
1.3) is properly interpreted by the text browser (Figure 1.4), thereby leading to easy and efficient browsing for this section of the society. For example: the images at the top in the Webpage in Figure 1.1 (after the logo), which are hyperlinks to other section of the Website are interpreted as a single line without separators by the text browser (provided those images contained ALT tags, with a proper description of the image), as shown in Figure 1.2. Again the links (actually images) at the left side of the Webpage in Figure 1.1 are mixed up with the text (welcome note) as shown in Figure 1.2, due to the way the tables are interpreted by the text browser. These shortcomings are efficiently met in the Webpage designed for the text browser in Figure 1.3, where the layout and the links are designed keeping in mind the aforementioned factors, and as such, can be seen the way that page is interpreted by the text browser in Figure 1.4.
Figure 1.4 Text Browser Friendly Website with Lynx (DOS based browser)

The software tool designed and developed under this project can convert a graphic intensive Website to a text browser friendly Website with the proper format and layout. These resulting Web pages are optimally interpreted by the screen reader, thereby making browsing simple and efficient for the visually challenged people. This text browser friendly version of the Website is a linear (row-wise) implementation of the former version (table/grid, i.e. rows and columns), keeping in-mind the coding complexities,
design and layout of the former. However, the Web-page design does not embrace a
generic design pattern, and varies from developer to developer. Thus a generic
conversion tool is almost impossible to design, thereby in some cases requiring a little bit
of manual editing after the conversion.

The following chapters discuss the project implementation in detail from both a
user’s and a programmer’s perspectives. Chapter 2 gives a brief overview of the legal
aspect associated with Web Accessibility, the assistive technology used by the visually
handicap users, the features of the tool developed, and detailed description of the User
Interface. Screen-shots are used to aid in understanding the interface design. Chapter 3
deals with system design. This section describes the analysis and designing process. It
contains a brief overview of parsing to help reader understand the underlying concept.
Chapter 4 describes the testing and evaluation process of the system. Finally Chapter 5
includes concluding remarks on the project and possible future work.
2. A TOOL FOR THE CONVERSION OF A GRAPHIC INTENSIVE WEBSITE TO A TEXT BROWSER FRIENDLY WEBSITE

2.1 Overview

Be it educational institutions, offices, or any other sector, the Internet serves as a primary source of information and one which is highly referred to and relied upon. Physically challenged people, being a part of all the sectors, need to access this information in order to keep themselves well informed and up to date. The graphic intensive design of the Web-pages poses hurdle in clear and optimal translation of Web pages for visually challenged people, who use different assistive technologies for browsing. The main aim of this project was to develop a software application that could translate the graphic intensive Website into a text browser friendly Website, keeping the format and layout of the Website intact, which can be optimally interpreted by the screen reader (a Web accessibility tool for the visually handicapped community), thereby making Web browsing simple, efficient, and fun for physically challenged people.

2.1.1 Webb Accessibility and the Law

Section 508:

“Almost one in five American adults has a disability which can affect their capacity to use conventional Web browsers” [Sierkowski, 2002]. This fact along with the emergence of Web as a source of Information and communication technology (ICT), and
accessibility need of all the sections of the society, including people with disabilities, served as a motivation for application of laws and policies to this new medium i.e., ICT. “Recent reports in the popular press characterize 95-99% of the Web sites as inaccessible” [Sullivan, 2000]. The Congress in 1998 amended the section 508 of the Rehabilitation act of 1973, thereby mandating that: “All electronic and information technology used, procured, developed, or maintained by agencies and departments of the U.S. government must be accessible to people with disabilities” [Slatin, 2003]. Section 508 applies to federal institutions and employees, and states that if the federal agencies have tasks that need to be done online then they must be accessible to the disabled employees also [Section 508, 2005].

American with Disabilities Act (ADA):

This law was formulated as a result of an intense research which helped in reaching to facts in black and white, thereby leading to the need for a law, for the population with disabilities. This law was signed in 1990, and its purpose was to establish a clear and comprehensive prohibition of discrimination on the basis of disability. This law states that: “The ADA extends the requirements of the Rehabilitation Act to the entire nation, not just programs that receive federal funds. It requires public colleges to ensure communications with persons with disabilities are as effective as communications with others, unless doing so would result in a fundamental alteration to the program or service
or in an undue burden; it also mandates a plan of compliance be established and
maintained” [Sierkowski, 2002],[W3C, 2005].

2.1.2 Screen Readers, Tool For Visually Impaired

Assistive technologies are used by the physically challenged section of the society
to accomplish the tasks which otherwise they can’t accomplish. The technology usually
refers to software adaptations, specially designed hardware devices, and/or standard
devices used in alternative ways which can help user access. These devices include:
screen readers, screen magnifiers, showsounds and soundsentry, scanning software,
alternate keyboards, braille and dynamic Braille [Vanderheiden, 2005]. For the purpose
of this project we focused on screen readers. An open source text browser, such as Lynx
was used to test a handicap friendly Website. The Website which is properly interpreted
by the text browser will be properly interpreted by the screen reader.

Screen readers assist the user in the navigation around the computer screen by
presenting the happenings on the screen in a meaning full way using speech output. Thus
the screen readers produce audio translations of text from the computer screen. There are
various types of screen readers available for different environments such as: JAWS, Hal,
and Windows Eyes etc. The cost of the screen readers varies from $750 to $1600. There
are other less costly screen readers available in the category of “Budget Screen Readers”
which are usually designed for the home market. They have limited features, and provide
restricted screen reading in comparison with the other more expensive screen readers.
The cost of the budget screen readers varies from $140 to $350 [Technology, 2005].
2.2 Features

This tool is a standalone desktop application, accessible to each and everyone be it instructor or students, with the necessary user rights to log into the system over the network. Thus, this application can be installed and used through any computer or laptop, thereby providing different users with the capability of converting their Websites into the text browser friendly Website. The application provides the following features:

- The users can install the tool on any stand alone system.
- The users can browse a HTML file, which they want to convert into the text browser friendly format. The selected file is then parsed and converted into a text browser friendly file.
- Once the file is parsed, a new directory is created in the same folder where the original HTML file was, and the new parsed file is stored there with the same name, thereby allowing creation of an entire new version of the Website with the same file names.
- The user is provided with the option to view the file in the text browser to verify the correctness of parsing.
- The user is provided with the option to view the file in the Web browser.
- The user is allowed to open the parsed file with a text editor, in case if minor editing is needed.

2.3 Description of the User Interface

This is a standalone desktop application. Any and every user having access to the system will be able to run the application. Thus, all users have a uniform user interface,
allowing them to browse the file which needs to be converted into a text browser friendly format. A screen shot of the initial user interface is shown in Figure 2.1 below:

![Initial User Interface Screen](image)

**Figure 2.1 Initial User Interface Screen**

Once the file is selected, it is then parsed and the users are provided with options to view with a text browser, Web browser, or open with the text editor for manual editing, if required. A screen shot of the final user interface is shown in Figure 2.2.
During the research for project analysis, designing and development internet was a primary source of information and in order to get a better insight of the concept, extensive research was done to look for the similar kind of work done by others in this field. As a result, a good number of Websites were found which have been developed in a text browser friendly format, so that users with physical limitations can browse these Websites efficiently, accounting for awareness with this concept. However all of these Websites have been developed manually and guidelines for making webpage accessible in text a browser were incorporated during designing phase, and as such no automated conversion tool was used. Although the research did helped in finding out a simulation of screen reader, which helps to understand what it is like for a person with vision loss or low vision to access the Internet using a software program called a "screen reader"
[WebAIM, 2005]. This simulation tool has been linked to a dummy Website to give a look and feel for the actual software tool. There is enough material on the Internet providing guidelines of what factors to consider while developing a text browser friendly Website, to give an idea of what to consider if the designing process needs to be automated. However no tool was found out to automate this process.
3. SYSTEM DESIGN

In the previous sections the background information, the main objective of the project, legal aspects, and a brief description of the user interface were discussed. This section deals with a brief overview of system analysis, system design, and implementation phase of the project.

3.1 Initial Requirement Analysis

This project was the idea of Dr. Dulal C. Kar, Associate Professor of Computer Science at Texas A&M University-Corpus Christi. This was suggested while researching for a topic for graduate project. The idea was than discussed in detail, including the areas to focus, intricacies involved, importance of the project, and platform that the system has been built.

This project has been a research and implementation of that research hand in hand with in the limited time constraint. During the requirement analysis phase, the Internet was the primary source of information. As such a rigorous research was done to get an insight of the different text browser friendly Website, their design, the issues to consider while designing those Website, and finally to look for a simulation of Screen-reader which can give an idea of how the Websites are interpreted using the assistive technologies. This helped in understanding the need of the new system, and establishing the scope of this project.
3.2 Systems Used to Develop this Project

This application is designed in Java, and can run either on the windows platform or Linux. The front end i.e., the user interface is an Applet (GUI interface for front end users), which allows the user to locate the file on the system. Once the file is selected the parser program performs the required operation and returns another applet with options to open the parsed file with the Web browser, text browser, or text editor. Figure 3.1 shows the high level data flow diagram of the system.

In order to execute this application, only Java Virtual Machine needs to be installed. Java being open source software, this can be downloaded from the Sun Website. Thus no additional cost is incurred in integrating this software into the current setup. Java Virtual Machine is the run-time package that provides the runtime environment for the Java programs, by interpreting the byte code of the Java program.

Some other reasons to use Java as the programming language for the project were:

- A very large range of built-in API, libraries, and functions, which reduces the burden on the part of the programmer.

- Similar language constructs as the other Object Oriented Programming Language like C++, and PHP, which helped reducing the learning curve.

- Designed and implemented a Parser based on the context free grammar in the Programming Language Class under Dr. Li in Java, which helped gaining familiarity with the concept and the programming language.

The other software tool that was used to aid in the process of the development is:

- Macromedia Dreamweaver – A Web design Integrated Development Environment (IDE) used for the design and development of the HTML
pages and forms and DHTML scripting. This software tool was used to understand and analyze the tabular layout of the different kind of graphic intensive and text browser friendly Web-pages.

3.3 Design Process

The steps that were conducted during the designing phase include:

- Studied the different graphic intensive Web-pages available on the Internet, to get an insight of the layout and designing.
- Studied the different text browser friendly Web-pages available on the Internet, to get an insight of the layout and designing, and relate to the changes incorporated in designing and development in comparison to the graphic intensive Webpage.
- Analyzed and designed a general procedure for handling the layout of graphic intensive Web-pages for the purpose of coding.
- Analyzed and designed a general procedure for handling other issues with the graphic intensive Web-pages such as; tables, images, list, hyperlinks etc, for the purpose of coding.
- Studied Java, its features and usage in current context, from different sources such as books, and websites. It was during this research that I came across the org.HTMLParser library which is an integral part of the project.
- Designing of the user interface.
3.4 Data Flow Diagram and Flow Chart

A high-level data flow diagram is shown in the Figure 3.1.

![High Level Data Flow Diagram](image)

**Figure 3.1 High Level Data Flow Diagram**

The flow chart which gives an abstract picture of the system developed can be further expanded as shown in Figure 3.2.
Figure 3.2 Flow Chart of the system
3.5 Concept of Parsing

“Parsing is the process of analyzing an input sequence (read from a file or a keyboard, for example) in order to determine its grammatical structure with respect to a given formal grammar. It is formally named syntax analysis. A parser is a computer program that carries out this task. The name is analogous with the usage in grammar and linguistics. The term parseable is generally applied to text or data which can be parsed” [ Parsing, 2005]. The parsers are commonly used to parse computer programming languages, based on context free grammars (CFG). A CFG consists of a series of definitions of non-terminal symbols. Each definition contains the non-terminal's name and a sequence of terminal and non-terminal symbols. Uses of non-terminals on the right of a definition refer to the non-terminal with the same name on the left of a definition. A terminal symbol represents an element of the token stream processed by the parser.

The process of parsing is handled by the lexical analyzer and parser, and as such a parser is a combination of lexer and parser. Parsing involves determining how the input can be derived from the start symbol with in the rules of the formal grammer. This can be accomplished in two ways:

Top-down parsing: This means that the parser can start with the start symbol and try to transform it to the input. Intuitively, the parser starts from the largest elements and breaks them down into incrementally smaller parts [ Parsing, 2005].

Bottom-up parsing: This means that the parser can start with the input and attempt to rewrite it to the start symbol. Intuitively, the parser attempts to locate the most basic elements, then the elements containing these, and so on [ Parsing, 2005].
Lexical analysis is the process of taking an input string of characters (such as the source code of a computer program) and producing a sequence of symbols called "lexical tokens", or just "tokens", which may be handled more easily by a parser.

A lexer is an object that transforms a stream of characters into a stream of tokens, which are supplied to the parser. The parser on the other hand is a program which accepts the leximes or tokens from the lexer and outputs if the string is a well formed sentence, a structural description of the sentence. Generally, parsers operate in two stages, first identifying the meaningful tokens in the input, and then building a parse tree from those tokens.

3.6 HTML Parser

This conversion tool is basically a top down parser which takes an HTML page, parses it in an intended way and returns a text browser friendly Webpage as output. The tool was developed using the ORG.HTMLParser library in Java. The two basic use-cases handled by the parser are “extraction” and “transformation”. Under the “extraction” process the links, text, images etc. are first of all extracted from the input HTML page and then a new HTML page is created from the scratch under the transformation process. The transformation process creates the new page based on the design, analysis, and coding done by the programmer for the parser. The HTML Parser is an open source library released under GNU Lesser General Public License [HTMLParser, 2005].

3.7 List of Major files

Figure 3.3 shows the listing of major files and their brief description.
<table>
<thead>
<tr>
<th>S. No.</th>
<th>File Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ConversionTool.java</td>
<td>This provides the front end user interface screen, which allows the user to browse the HTML file that has to be parsed.</td>
</tr>
<tr>
<td>2</td>
<td>TableNodeScanner.java</td>
<td>Scans the Table used for the layout in the graphic intensive Website, and retrieves the information about the number of rows, columns, row-span, and column-span. Creates an array with the obtained information, for use in the construction of the table while designing the text browser friendly Website.</td>
</tr>
<tr>
<td>3</td>
<td>Project.java</td>
<td>This uses the information gathered by the file TableNodeScanner.java, to design the layout of the text browser friendly Website. Apart from this it designs the entire Website keeping in mind the other design considerations.</td>
</tr>
</tbody>
</table>

**Figure 3.3 Lists of Major Files**

### 3.8 Implementation

The HTML input file is first parsed in a stream of nodes. There are two major types of nodes or lexemes which have been handled during parsing. They are:

- Text: This will be the regular text on the Webpage, apart from the HTML tags.
- Tag: This includes the HTML tags.

The entire HTML file is scanned for all the nodes, be it Text node or Tag node, and a list of nodes is created. Then this list of nodes is used to create a new HTML page from scratch. While designing the new HTML page the concept of recursion is used. First of all the top level Tag or the parent tag, i.e. “<HTML>” is scanned, followed by the child nodes, i.e. “<HEAD>”, “<BODY>”, “<TABLE>”, and so on. Again these tags are also scanned using the recursion approach and as such are scanned for their child nodes also. For example, if the “<TABLE>” tag is encountered than it goes into recursion where the child nodes, i.e. “<TR>” (table row tag) is scanned, followed by the child node for “<TR>” i.e. “<TD>” (table data tag), followed by the child node of “<TD>” which can be anything including the text or another table itself. If another table is encountered within a “<TD>” than this calls in for the top level recursion and the overall approach is repeated.

If an “<IMAGE>” tag is encountered, it is scanned for “ALT” attribute. If that is found then the image is filtered out and the text within the attribute is placed in its place. If however the “ALT” attribute is missing, than the name of the image is picked up. According to the American Disability Standards (ADA), this text is then followed by “[D]” which is appended to it. This “[D]” is a link to the HTML page which is created in a directory “ImageDescription” (created dynamically during parsing) by the name of the “image.html”. This is the blank HTML file where the developer is supposed to provide the description of the image. This directory is created under the same folder where the original HTML file was. The same approach is followed if the image is a hyperlink to another image. In either case a note about the existence of the image in the document is
placed just before the image. If an audio-video link is encountered, it is scanned for “PARAM” attribute. If that is found than the text within “PARAM value” is picked and placed in its place. According to the American Disability Standards (ADA), this text is than followed by “[CC” which is appended to it. This “[CC]” is a link to the HTML page which is created in a directory “VideoDescription” (created dynamically during parsing) by the name of the “param-value.html”. This is the blank HTML file where the developer is supposed to provide the description of the video. This directory is created under the same folder where the original HTML file was.

Whenever an ordered list or unordered list is encountered, these lists are turned into numbered list with all the elements in the list numbered in hierarchical order. The regular text within the document is picked as such and placed at the same place in the new Webpage. The script tags are also not filtered out and picked as such.

Thus all the tags within a HTML file are scanned and while creating a new page the requirements for the text browser friendly Webpage are considered and as such the tags requiring special handling are handled analytically implementing design considerations through programming. The attributes of these tags are also scanned and picked while designing the new Webpage. Examples of these tags are: “<TABLE>”, “<TR>”, “<TD>”, “<IMG>”, “<UL>”, “<LI>”, “<OL>”, “<A>” etc. Thus the final Webpage which is created is a text based browser Webpage.

While implementing and running the tool couple of things should be kept in mind. They are:
• Make sure that a Java development system (JDK) is installed on the system. This can be found out by typing the command “javac” on the DOS prompt. If this gives the help information than that implies that JDK is installed.

• It should be ensured that the version for java installed is greater than 1.2. This can be found out by typing the command “java –version” on the DOS prompt, if java is installed.

• The CLASSPATH for running the tool is set.
4. TESTING AND EVALUATION

Quality in software is a prominent factor which developers strive for. Thorough testing is what accounts for the credibility and robustness of software. This software tool was tested for usability testing so as to fix the pitfalls and ensure proper functioning.

4.1 Usability Testing

The system was tested for usability testing so as to ensure smooth and efficient functioning. In this phase, the application was checked for ease of use, robustness, and user-friendliness. The extent of correctness of the parsed WebPages was ensured by viewing them in the text browser, e.g. Lynx v2.8.3, where they were checked for proper layout, easy navigation, correct linking to other pages, easy to use forms, and other significant factors. Feedback and suggestions were taken from fellow students regarding the interface design and usability of the system. Based on their feedback, the changes were made to the system to satisfy the users of the system. A simulation model of the screen reader was researched and found, so as to give an insight of how the Webpage is interpreted if a user is using a screen reader.

Figure 4.1 shows the snapshot of the Environmental Health & Safety Website for Texas A&M University when viewed in Firefox 1.0.7. The Menu on the left hand side is shown using the “IMAGE-MAP”.
Figure 4.1: EH&S Graphic Intensive Website with Firefox 1.0.7

Figure 4.2 shows the snapshot of the same Website when viewed in the text browser Lynx. The menu which was a part of the “IMAGE-MAP” is filtered out when viewing in the Lynx and as such there is a loss of information.
Figure 4.2: EH&S Graphic Intensive Website with Lynx

Figure 4.3 shows the snapshot of the same Website after being parsed through the conversion tool, and when viewed in the text browser Lynx. Here the menu which was a part of the “IMAGE-MAP” is not filtered out completely and as such the links contained within the AREA attribute of the IMAGE-MAP are still a part of the display. The “#” in the snapshot shows that these links were directed to the same Webpage itself.
Figure 4.3: EH&S Website after parsing

Figure 4.4 shows the snapshot of a commercial Website “MercuryNews.com” when viewed in Internet Explorer 6.0. This Webpage is the site index for the aforementioned Website, which uses unordered list for indexing the Web-pages and as such the hyperlinks to those in the Website.
Figure 4.4: The Mercury News Graphic Intensive Website with IE 6.0

Figure 4.5 shows the snapshot of the same Website after being parsed through the conversion tool, and when viewed in the text browser Lynx. Here the unordered list is converted into ordered list by the conversion tool. This makes it easy for the visually impaired section of the society to browse the site index, since they are able to decipher between the different levels of indexes.
<table>
<thead>
<tr>
<th>10.1.</th>
<th>Iraq: The Aftermath</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.</td>
<td>Breaking News</td>
</tr>
<tr>
<td>12.</td>
<td>Lottery</td>
</tr>
<tr>
<td>13.</td>
<td>National</td>
</tr>
<tr>
<td>14.</td>
<td>Politics &amp; Government</td>
</tr>
<tr>
<td>14.1.</td>
<td>Elections</td>
</tr>
</tbody>
</table>

**Business**

1. 60-second Business Break
2. Financial Markets
3. Technology
4. Personal Technology
5. Personal Finance
6. People and Events
7. Business Extra
8. Drive

**Opinion**

1. Perspective
2. Columnists
3. Action Line
4. Meriessa Pace
5. Mike Antonucci
6. Morning Buzz
7. Board of Contributors
8. Mike Cassidy
9. L.O. Chung
10. Davey B
11. Karen D'Souza
12. Laurie Daniel
13. Mark de la Vina
14. John Fentener-Wald"n
15. Jack Fischer
16. Patty Fischer
17. Bud Geracie
18. Linda Goldston
19. Miguel Heff
20. Scott Herbold
21. Home plate
22. Sue Hutchinson
23. Immigration
24. Donna Katz
25. Tim Kawakami
26. Brad Kaw
27. Ann Kilian
28. Mike Langberg
29. Marian Lia
30. Glenn Lussier
31. Male Call
32. Matt Marchall
33. Barbara Marshman
34. Charlie McCollum
35. Bruce Neuman
36. Mark Purdy
37. Mr. Roadshow
38. Joe Rodriguez
39. Janice Ronbeck
40. Dean Takahashi
41. Hot Stuff
42. Alexa Watson
43. Steve Yavaska
44. Alan Hess
45. Internal Affairs
46. Master Gardener
47. Richard Scheinick
48. Sal Pizarro
49. Tech Test Drive

**Sports**

1. High school sports
2. College sports
3. Cal
4. San Jose State University
5. Santa Clara University
6. Stanford
7. Soccer / San Jose Earthquakes
8. Golf
9. Motorsports
10. Other sports
Figure 4.5: Mercury News Website after parsing with Lynx
5. CONCLUSION AND FUTURE WORK

With World Wide Web being a source of vast array of information, and attempts being made to make this information available to broadest range of users, Web accessibility is gaining importance in this day and age. It is not just a black and white issue but is being worked into law and actions are being taken to ensure that they are being acted upon. This project is an attempt to develop a tool capable of converting a graphic intensive Website to a text browser friendly Website, so those users who use assistive technologies to view Websites can browse these Websites.

This project is a standalone desktop application developed in Java which can be installed in the TAMUCC computer labs on each and every system without any additional cost to the current setup. Testing and implementation was conducted using the text base browser, i.e. Lynx v2.8.3. This system will help the Web developers to test and convert their Websites into text browser friendly version, thereby making it available to the broadest range of users.

However this tool is bounded by certain limitations. One of the limitations is that being a stand alone desktop application, this tool does not offer conversion of Web-pages on the fly. Since this tool is not based on client server technology, the user will have to install it on the local machine so as to perform the parsing. Another limitation is that, a little bit of manual editing might be required after the Webpage is parsed depending on the complexity of the Webpage. This is because of the different designs and layouts the Web-pages incorporate, making it difficult to generalize a uniform coding technique and as such development of a tool which can parse these Web-pages optimally.
5.1 Future Work

This project was an attempt to capitalize on the idea suggested by Dr. Kar, Associate Professor in the Computer Science Department. Due to time constraints, only the important and influential factors were taken into consideration. As such this project can be carried further to the next level, thereby broadening the functionality domain of the project, and generalizing the tool, so that it can parse the vast variety of Web-pages, without needing or extremely minimizing the manual editing.


APPENDIX A: Data Dictionary of High-Level Data Flow Diagram and Flow Chart of the system

- **Parser**: The parser performs the parsing of the input HTML file into stream of nodes which are used to create new HTML page from scratch.
- **Frame**: A new window which provides the initial graphic user interface (GUI).
- **Graphic Intensive Website**: Fancy Websites with lot of images, and graphics.
- **Text Browser Friendly Website**: Website which is designed to be viewed in a text browser optimally.
- **Web Browser**: A tool that is used to access the HTML pages (Web-pages) available through the World Wide Web over the internet. Best for viewing the graphic intensive Websites. For Example Firefox, Internet Explorer etc.
- **Text Browser**: A tool that is used to access the HTML pages (Web-pages) available through the World Wide Web over the internet. Best for viewing text browser friendly Websites.
- **Text pad**: A basic type of word processor that comes with a Windows operating system. Can be used to write in HTML code.
APPENDIX B: Data Dictionary of the basic HTML tags

- HTML: This marks the start of an HTML document. The entire content of the Webpage is contained within the HTML tag.
- BODY: This mark begin and ending of the body of the Webpage, and contains all the contents of the Webpage.
- HEAD: This contains information about the document or Webpage.
- TABLE: This helps in creating a table structure within a document. Helps in arranging the layout of the Webpage.
- TR: This is an attribute of the TABLE tag, which accounts for the creation of new row within the table.
- TD: This is an attribute of the TABLE tag, which accounts for the creation of new column within the table.
- IMG: This allows the insertion of images within the HTML documents.
- MAP: Netscape and Microsoft (version 2.0 and later) supports the client-side image map element. It helps to map different regions of the image with the different URL’s.
- ALT: This is an attribute for the image tag and it helps in providing the brief description of the image being inserted.
- SRC: This is an attribute of the image tag which specifies the path where the image exists.
- OL: Allows the creation of ordered list within an HTML document. This list is a numbered list.
• UL: Allows the creation of unordered list within an HTML document. This list is a not a numbered list.

• LI: An attribute of the ordered list, and unordered list that helps in adding elements in the list.

• A: This is an anchor tag which helps in linking the HTML document to another HTML document, or an image.

• FORM: Allows the creation of a form within the document. Provides different type of input element which can be inserted within the form. For example text-box, labels, text area, check box, radio button, button, drop down list etc.

• SCRIPT: Allows the insertion of the executable script within the HTML document. Generally comes within the HEAD tag.