Development of Web Crawler and Database System for Visually Impaired Students - An Application to Career Support Web System -

Yuji ITOH*!, Toshiya URUSHIHATA**, Toru SAKUMA**, Sachiko IKEMUNE***, Masanori TOJO***, Teruhisa MIYAKE*, Hiroshi TAKAHASHI**, Norio OHKOSHI***, Kazushige ISHIZUKA***, Tsukasa ONO*
Faculty of Health Sciences, Tsukuba University of Technology

Abstract: This report describes a Web application intended for visually impaired users. Today hundreds of millions of people benefit from the Internet (or the World Wide Web), which is the greatest source of information in the world. The World Wide Web Consortium (W3C) has set the guidelines for Web content accessibility, which allows visually impaired people to access and use Web contents. However, many of Web sites do not yet follow these guidelines. Thus, we propose a Web application system that collects desired data related to a specific topic as an agent (i.e., in place of the visually impaired person), stores the collected information in a database, and presents it to the user as per the user's choice of color and font size. We have implemented this system using a Web crawler, and a simple Web database technology that runs on an open source Wiki platform. This system is, then, applied to a career support, namely job opportunity database, for visually impaired students as a pilot project. A test tool confirmed that the system is compliant with the rules and guidelines for Web content accessibility.

Key Words: CSS, database, user-dependent presentation, visual impairment, Web crawling, Wiki

1. Introduction

Today, the World Wide Web has become the greatest source of information for most of the people. This is proven by the fact that people consult with the World Wide Web (through a search engine service) more often rather than with librarian when in need of information. Visually impaired people often run into difficulties when browsing the Web contents as most of it is not designed for people with visual impairments. In order to solve this problem, the World Wide Web Consortium (W3C) devised the recommendations and guidelines for the Web content accessibility [1], with the intention of helping Webmasters who wish to make their Web contents universally accessible even to users with people with visual impairments. However, a large fraction of the Web sites available on the Internet today are not yet compliant with the W3C guidelines.

Even if the Web contents comply with the accessibility guidelines, eye fatigue accumulates during long-duration Web browsing, which often occurs in an extensive Web search for specific terms. It is supposed that visually impaired people easily get tired and thus give up the search on the way (i.e., before achieving the objective). We propose a system that uses a Web crawler based data collection and a simple Web database as a solution for this problem.

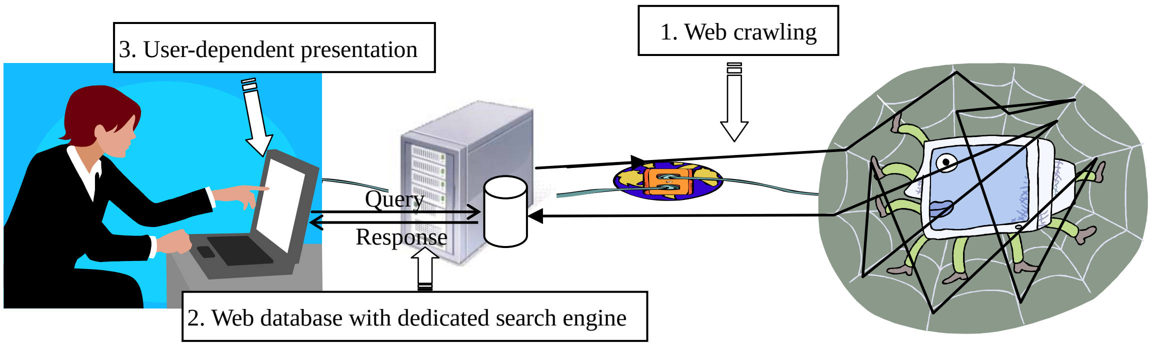

3. User-dependent presentation
1. Web crawling
Query
Response
2. Web database with dedicated search engine

Fig. 1. Illustration of proposed system and component technologies

As illustrated in Fig. 1, the proposed system comprises three component technologies; 1) a **Web crawler** designed to automatically collect data related to specific terms from the Internet, 2) a simple **Web database with a dedicated search**

*: Department of Computer Science
**: Course of Physical Therapy, Department of Health
***: Course of Acupuncture and Moxibustion, Department of Health
!: e-mail itoh@k.tsukuba-tech.ac.jp tel: +81-29-858-9568

engine, and 3) a **user-dependent presentation** with various color and font size choices (later to support alternative audio presentation option). We applied this system to a career support for the visually impaired students (especially those who major in medical and health sciences) as a pilot project, in which the system is tuned to cope with job opportunity related data. Since this system has a generic design, it can be used to develop applications for use in areas such as technical terminology, publication data, etc.

2. Preparation and development environments

We carefully selected the environments for system development. The requirements included:

- Req.1 - The programming language must be capable to readily handle hyper text (mandatory)
- Req.2 - The Web platform must be open source and maintained by many active members (mandatory)
- Req.3 - The Web platform must be capable of handling Japanese character sets (mandatory)
- Req.4 - The Web platform shall be written in the chosen programming language mentioned in Req. 1 (optional)

2.1 Programming language

The Perl programming language was chosen as it is matured and stable. It is also noteworthy that CPAN [2], a famous technical community, provides various modules for Perl programmers. Thus, we can save resources to invest for the development if the desired module is available in the CPAN repository. In fact, we utilized several useful CPAN modules such as HTML::TagParser, which analyzes the structure of an HTML document.

2.2 Web platform

We chose FreeStyle Wiki (FS Wiki) [3] as a Web platform because it met Req. 2 through Req. 4. Wiki technologies, represented by Wikipedia [4], are becoming more popular and have the following features;

- Users can browse, edit, and delete Web contents according to individual access privileges
- Operations can be performed using a Web browser (no special software needs to be installed)
- All of the Web contents can be indexed by a native search function

It shall be noted that the FS Wiki has an extra feature, customization by plug-in module. This useful feature allowed us to support a database function by tailoring the module by ourselves. Moreover, the FS Wiki was designed to comprehend CSS (cascading style sheet), which enabled us to handle the Web presentation style separately from the Web contents. Therefore, we are able to customize the parameters (e.g., foreground color, background color and font size) for each user by tailoring user-dependent CSS mechanism, which is not supported by the original FS Wiki. We installed the FS Wiki as a cgi (common gate interface) over the Windows Server 2003.

3. Collection of job opportunity data from the Internet

We use the *Web crawling* technology to collect the job opportunity data. A *Web crawler* is a computer program that browses the World Wide Web in a methodical, automated manner. In general, it starts with a list of URLs to visit, called the *seeds*. As the crawler visits these URLs, it identifies all of the hyperlinks in the page and adds them to its list of URLs to visit. Therefore, a huge number of URLs will be listed when the Web crawling is complete. In order to save time when crawling over the Internet, we first attempted to find Web sites that provide the service that we were going to perform (i.e., provide lists of URLs for job opportunity candidates). Then, several *seeds* (i.e., Web sites) that introduce hospitals and clinics were sought by using popular Web search engines. Here, we classify the Web crawling into two successive stages: 1) an intensive search that focuses on a single *seed*, and 2) an extensive search that travels according to the URL list obtained by the intensive search. Prior to the intensive search, we need to do *site profiling* through which the structure under the specific domain (i.e. *seed*) will be clarified manually (i.e., not automatically). Then, the intensive search is designed to enumerate the desired list (mostly URLs). Such *seeds* are usually deep[1] and very complicated in structure as to prevent others from stealing the contents easily. An example of an URL path name convention would be: http://(domain)/(search module)?(key)=(identifier), where the domain and search module (e.g., search.cgi) are fixed and there are wide variety of key (e.g., prefecture, section, etc) and identifier combinations.

```
use HTML::TagParser;
use LWP;
our @jobs = ('理学療法士','PT');
our @words = ('採用', '募集', '求人');

main($ARGV[0]);

sub main {
    my $url = shift;
    my $parent = 'root';
    my $hash = {};
    $hash->{$parent}->{LIST} = ();

    # do intensive search
    $hash = &intensive_search($hash,$url,$parent);

    my @list = @{$hash->{$parent}->{LIST}};
    @list = &id_list($hash,$id,@list);
    foreach my $id (@list) {
        my $url = $hash->{$id}->{'url'};
        my $data = {};
        $data->{'::found'} = 0;
        @{$data->{'::list'}} = ($url);
        do {  # do extensive search
            $data = &extensive_search($data);
        } while ($#{$data->{'::list'}}>=0);

        # write id-related data to a file here
    }
}

sub intensive_search {
    my ($hash, $url, $parent) = @_;
    my $src = &get_response($url);
    my $end_flag = 0;

    if ($src =~ m||) { # set termination condition
        $end_flag = 1;
        $hash->{$parent}->{'id'} = $parent;
    }
    my $html = HTML::TagParser->new($src);
    my @list = $html->getElementsByTagName( "a" );
    foreach my $elem ( @list ) {
        my $url = $elem->getAttribute("href");
        if ($end_flag) {
            $hash->{$parent}->{'url'} = $url;
        }
        elsif ($url =~ m|\?key=(.+)|) {
            unless (defined($hash->{$1})) {
                push(@{$hash->{$parent}->{LIST}},$1);
                $hash->{$1} = {id=>$1,url=>$url};
                $hash = &intensive_search($hash,$url,$1);
            }
        }
    }
    return $hash;
}

sub get_response {

    my $url   = shift;
    eval("use LWP::UserAgent;");
    my $ua   = LWP::UserAgent->new();
    my $req = HTTP::Request->new('GET',$url);
    my $res = $ua->request($req);
    return $res->content();
}
```

```
sub extensive_search {
    my $data = shift;

    my $url = shift(@{$data->{'::list'}});
    my $src = &get_response($url);
    return $data if ($src !~ m#<[html|HTML].*>#);

    if ($data->{$url}->{'::found'}) {
        foreach my $kw (@jobs) {
            if ($src =~ m|$kw| && !defined($data->{$kw})) {
                $data->{$kw}->{'url'} = $url;
            }
        }
    }
    else {
        my $html = HTML::TagParser->new($src);
        return $data if ($html eq "");
        my @list = $html->getElementsByTagName( "a" );

        foreach my $elem ( @list ) {
            my $url = $elem->getAttribute("href");
            my $text = $elem->innerText;
            my $flag = 0;
            foreach my $word (@words) {
                if ($text =~ m#$word#) {
                    $flag = 1;
                    last;
                }
            }

            if ($flag) {
                unless (defined($data->{$url})) {
                    push(@{$data->{'::list'}},$url);
                    $data->{$url}->{'::found'} = 1;
                    foreach my $kw (@jobs) {
                        if ($text =~ m|$kw| && !defined($data->{$kw})) {
                            $data->{$kw}->{'url'} = $url;
                        }
                    }
                }
            }
        }
    }
    return $data;
}

sub id_list {
    my $hash = shift;
    my $id = shift;
    my @list = @_;

    if (defined($hash->{$id}->{'id'})) {
        push(@list,$id);
    }
    elsif (defined($hash->{$id}->{'::order'})) {
        foreach (@{$hash->{$id}->{'::order'}}) {
            @list = &id_list($hash,$_,@list);
        }
    }
    return @list;
}

1;
```

Fig. 2. Pseudo Perl script for Web crawling

[1] The deep Web refers to Web content that is not part of the surface Web, which is indexed by standard search engines.

Fig. 2 shows a pseudo Perl script for Web crawling that covers both the intensive and extensive searches for a certain *seed*. Note that this program is intended to find job opportunities for physical therapists. The intensive search seeks out a list of Web sites likely to provide job opportunities for physical therapists (e.g. hospitals/clinics with rehabilitation department), whereas the extensive search visits and analyzes all of the likely Web sites found through the intensive search, and then extracts those that include the key word sets beforehand (e.g., physical therapist).

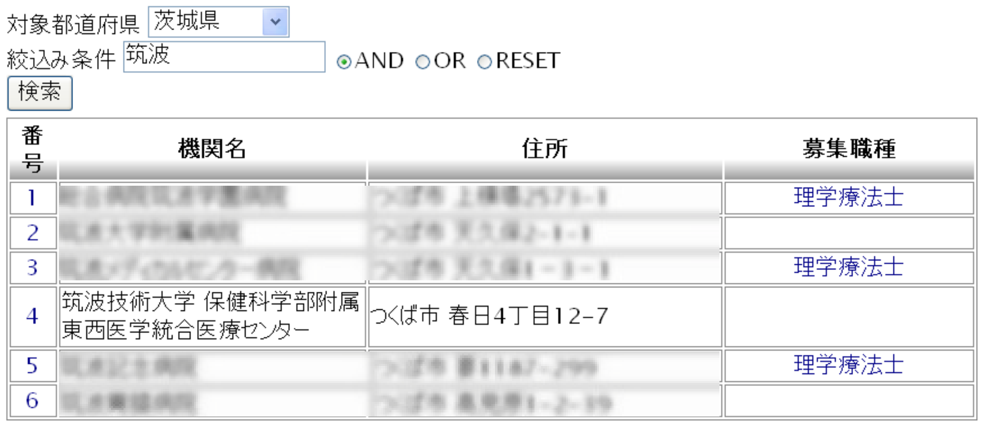
4. Presentation of collected data

First, we need to develop a database module by ourselves since the FS Wiki does not support it. Incidentally, we found that primitive modules useful for the database development were available in the FS Wiki community site. We then devise a database by exploiting those primitive modules. The data collected by Web crawling are converted into structured text format that FS Wiki comprehends. The structured text is a very simple and straightforward set of rules that determines how to format a plain text document. The formatting allows us to logically organize structure of sections, paragraphs, bullet lists etc. FS Wiki is designed to dynamically translate such structured text into an HTML document (i.e., HTML is generated for presentation purposes only, and is not be stored) when invoked by a client. For instance, the translation of the structured text "*Hello" results in "<ul><li>Hello</li></ul>." In addition, the presentation properties of particular logical elements can be systematically changed using style sheets (i.e., CSS). Thus, we can achieve user-dependent presentation by tailoring a mechanism for specifying a CSS for each user.

4.1 Search engine

The Web crawler collected approximately 11,000 hospitals/clinics that are likely to provide job opportunities for physical therapists. The collected data are classified as per prefecture and stored in "one page per prefecture" format (47 pages in all). This is done for two reasons: 1) handling up to 11,000 pages (i.e., one page per site basis) will hinder the server response (users get impatient), and 2) the likely users usually assign higher priority to the work address in a job search.

The data handling is designed to be flexible by separating the database columns (i.e., appearance) from the actual data format. We use a template to read data fields from the actual data. Fig. 3 depicts the search results using the job opportunity database with the query "筑波" (Tsukuba) and the prefecture of "茨城県" (Ibaraki prefecture). The actual data (just two sites excerpted) and the template used for this application are shown in Fig. 4. In this way, we can use any data field as a keyword, where the data fields are represented as %field% in the template. It should be noted that the rightmost column represents the job opportunity data (with its hyperlink) sought through the extensive search process. This indicates that there are three job opportunities for physical therapists available in Tsukuba city.



対象都道府県 茨城県
絞込み条件 筑波 [✓ AND] OR RESET
検索

番号	機関名	住所	募集職種
1	[redacted]	[redacted]	理学療法士
2	[redacted]	[redacted]	
3	[redacted]	[redacted]	理学療法士
4	筑波技術大学 保健科学部附属東西医学統合医療センター	つくば市 春日4丁目12-7	
5	[redacted]	[redacted]	理学療法士
6	[redacted]	[redacted]	

Fig. 3. Search results from job opportunity database with query "筑波" (Tsukuba) in data for "茨城県" (Ibaraki prefecture)

```
!!!筑波技術大学 保健科学部附属 東西医学統合医療センター
*住所: つくば市 春日 4 丁目 12-7
*URL: http://www.k.tsukuba-tech.ac.jp/cl/
*TEL: 029-858-9590
!!採用情報（鍼灸、理学療法関係のみ）

!!機関情報
*設立母体:
*施設種別:
*ベッド数:
*施設基準:
*対象疾患:
*在籍卒業生:
*関連スタッフ数
**PT:
**OT:
**ST:

!!![redacted]
*住所: [redacted]
*URL: [redacted]
*TEL: [redacted]
!!採用情報（鍼灸、理学療法関係のみ）
*募集職種: [理学療法士
|[redacted]]

!!機関情報
*設立母体:
*施設種別:
*ベッド数:
*施設基準:
*対象疾患:
*在籍卒業生:
*関連スタッフ数
**PT:
**OT:
**ST:
```

(a) Actual data

```
!!!%subject%
*住所: %address%
*URL: %url%
*TEL: %tel%
!!採用情報（鍼灸、理学療法関係のみ）
*募集職種: %job%

!!機関情報
*設立母体: %found%
*施設種別: %type%
*ベッド数: %beds%
*施設基準: %base%
*対象疾患: %target%
*在籍卒業生: %obog%
*関連スタッフ数
**PT: %numPT%
**OT: %numOT%
**ST: %numST%
```

(b) Template

Fig. 4. Actual data and template used for job opportunity database

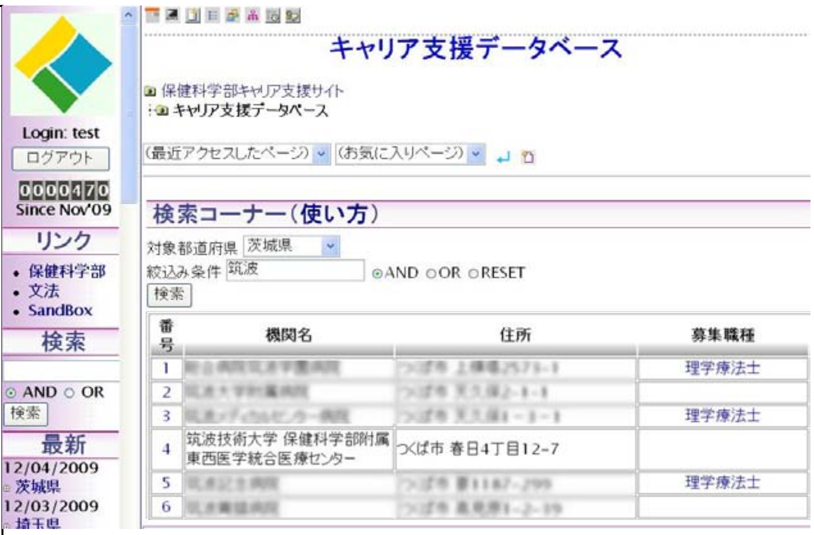
4.2 User-dependent presentation

There are mainly two methods to make it easier for visually impaired people to read Web contents: 1) screen magnification to enlarge what is displayed on the monitor, and 2) Web-browser-specific parameters related to presentation (represented by a style sheet). We assume that the latter is preferred for Web browsing because it allows only the Web contents to be enlarged while irrelevant application windows, icons, etc. remain unchanged. Thus, we adopt the latter, and instantiate it with a user-dependent style sheet. The original FS Wiki uses a CSS (cascading style sheet) based presentation. However, with this method, the CSS is fixed for all users (i.e., not user-dependent). Therefore, we first add a CSS file to the user parameters that are stored in the system configuration file and activated at user login. For instance, setting "default::100" in the user parameters indicates the use of the default.css with a 100% font size. Extra attention has been paid to Web content accessibility to ensure that our Web system is compliant with the recommendations on the CSS techniques for Web content accessibility defined by W3C [5]. The guidelines we mainly focus on are shown below.

- Use style sheets to control layout and presentation.
- Create a style of presentation that is consistent across pages.
- Use relative rather than absolute units in markup language attribute values and style sheet property values.
- Ensure that foreground and background color combinations provide sufficient contrast.
- Ensure that all information conveyed with color is also available without color.

Fig. 5 shows a part of a style sheet that we tailored and its appearance in the actual application window. In the style sheet, the color parameters are defined by absolute values whereas the font sizes are specified by values relative to the font size of the "body" element (i.e., main contents). Each user can set the color parameters by choosing a favorite style sheet from several choices. Note that we can create and add a new style sheet to the choices if a user doesn't find a favorite one among them. Then, the font size is activated by adding (or cascading) the text "body {font-size: ".$emsize."em;}" to the style sheet chosen above, where $emsize equals the font size parameter (e.g., 100) divided by 100. An example of a user-specific presentation with a black background and 150% font size is illustrated in Fig. 6.

```
body {
    color                   : #000000;
    background-color        : #FFFFFF;
    font-family             : "MS ゴシック
",sans-serif;
}
h1 {
    font-size               : 170%;
    color                   : #0000CC;
    background-color        : #FFFFFF;
}
h2 {
    font-size               : 150%;
    color                   : #444488;
}
h3 {
    font-size               : 135%;
    color                   : #444488;
}
h4 {
    font-size               : 120%;
    color                   : #444488;
}
```



キャリア支援データベース

保健科学部キャリア支援サイト
キャリア支援データベース

Login: test
ログアウト
0000470
Since Nov'09
リンク
- 保健科学部
- 文法
- SandBox
検索
AND OR
最新
12/04/2009
茨城県
12/03/2009
埼玉県

検索コーナー（使い方）

対象都道府県 茨城県
絞込み条件 筑波 AND OR RESET

番号	機関名	住所	募集職種
1	[illegible]	[illegible]	理学療法士
2	[illegible]	[illegible]	
3	[illegible]	[illegible]	理学療法士
4	筑波技術大学 保健科学部附属東西医学統合医療センター	つくば市 春日4丁目12-7	
5	[illegible]	[illegible]	理学療法士
6	[illegible]	[illegible]	

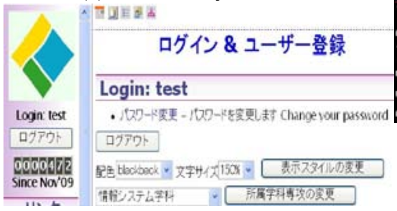
(b) Appearance in an actual application window

(a) Part of a style sheet

Fig. 5. Part of a style sheet we tailored and its appearance in an actual application window

```
body {
    color                   : # FFFFFF;
    background-color        : #000000;
    font-family             : "MS ゴシック
",sans-serif;
}
h1 {
    font-size               : 170%;
}
h2 {
    font-size               : 150%;
}
h3 {
    font-size               : 135%;
}
h4 {
    font-size               : 120%;
}
```

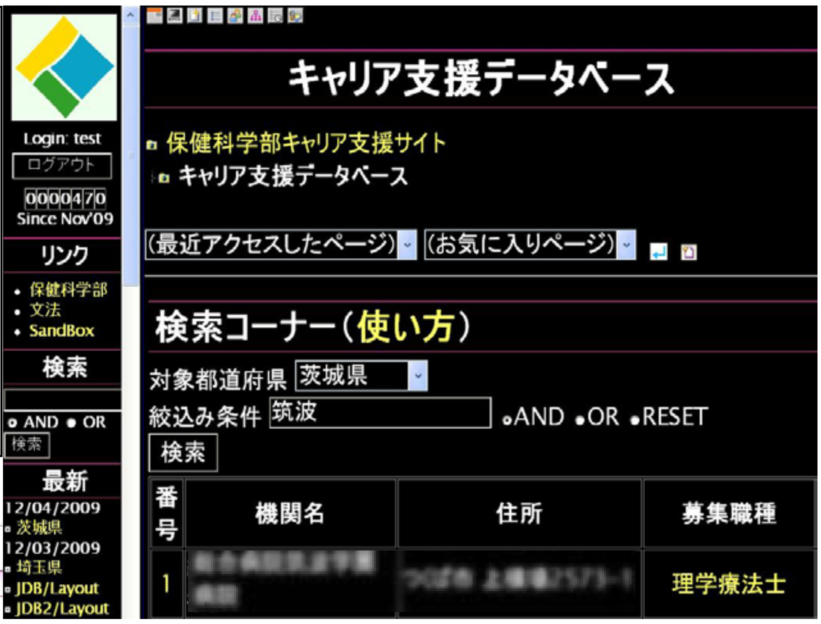
(a) Part of a style sheet



ログイン & ユーザー登録

Login: test
- パスワード変更 - パスワードを変更します Change your password
ログアウト
配色 blackback 文字サイズ 150% 表示スタイルの変更
情報システム学科 所属学科専攻の変更

(b) Setting of presentation related user parameters



キャリア支援データベース

保健科学部キャリア支援サイト
キャリア支援データベース

Login: test
ログアウト
0000470
Since Nov'09
リンク
- 保健科学部
- 文法
- SandBox
検索
AND OR
最新
12/04/2009
茨城県
12/03/2009
埼玉県
JDB/Layout
JDB2/Layout

検索コーナー（使い方）

対象都道府県 茨城県
絞込み条件 筑波 AND OR RESET

番号	機関名	住所	募集職種
1	[illegible]	[illegible]	理学療法士

(c) Appearance in an actual application window

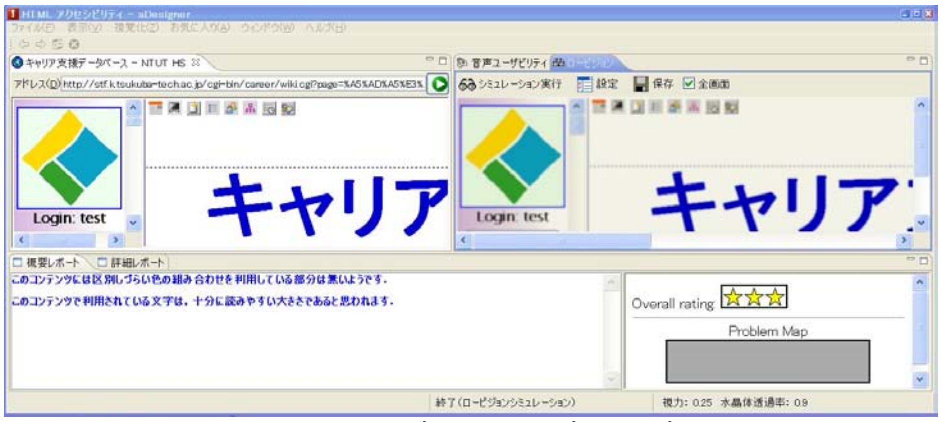
Fig. 6. An example of a user-specific presentation with a black background and 150% font size

5. Tests

We have conducted Web content accessibility tests using an evaluation tool called "aDesigner" [6], which was developed by IBM and is publicly available. The aDesigner tool was designed to test the accessibility and usability of Web pages for people who are blind or have limited vision. This tool is useful because it provides suggestions for further possible improvements, as well as the overall scores. The examination is classified into two tests, one each for the blind and those with limited vision: 1) a presentation of visual information (for limited vision), and 2) a screen reader[2] aware design (for the blind). After each test, a score is calculated based on the compliance with the related rules and guidelines: the Web Content Accessibility Guidelines [1], Section 508 of the Rehabilitation Act §1194.22 [7], Japanese Industrial Standard (JIS) X8341-3 [8], and so on. Table 1 lists several examples of the error and warning messages that the aDesigner generated and its suggested countermeasures when we made a preliminary evaluation of our Web system. After incorporating the countermeasures, we conducted the evaluation test again. As illustrated in Fig. 7, our system got through all of the criteria.

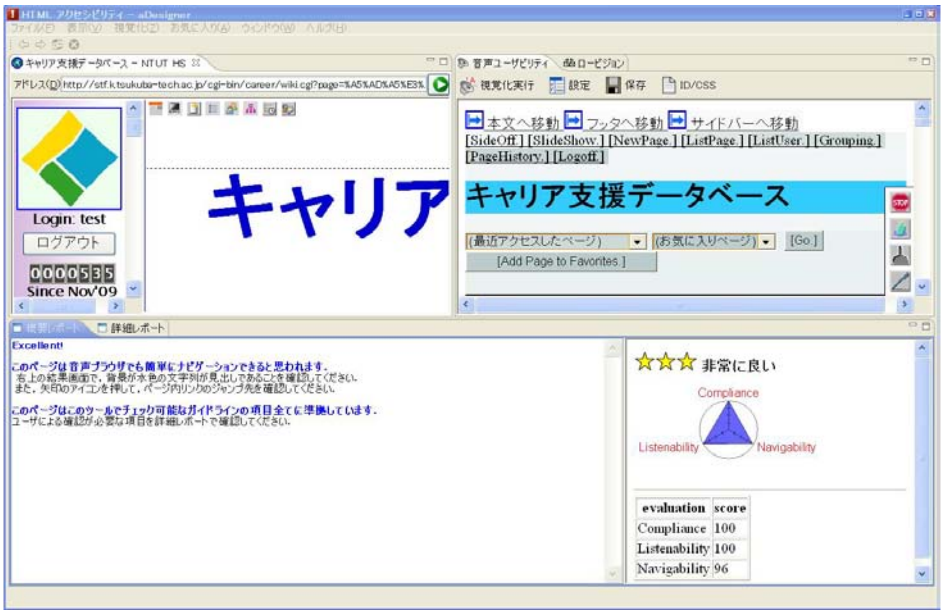
Table 1. Examples of error and warning messages and its suggested countermeasures

Test type	Error and warning messages	Countermeasures
Presentation of visual information	Insufficient color contrast	Make foreground color complementary to background color as much as possible
	Small font sizes	Tailor user dependent font size to be chosen between 0.5em and 3em with 0.1em step
Screen reader-aware design	Too long to reach the main contents	Add a hyperlink at page top to the main contents
	Missing or inadequate alternate text for images	Add alternate text for every image
	Shortage of intra-page links for navigation	Add intra-page links to major sections



このコンテンツには区別しづらい色の組み合わせを利用している部分は無いようです.
このコンテンツで利用されている文字は，十分に読みやすい大きさであると思われます.

(a) Test results in terms of presentation of visual information



Excellent!

このページは音声ブラウザでも簡単にナビゲーションできると思われます.
右上の結果画面で，背景が水色の文字列が見出しであることを確認してください.
また，矢印のアイコンを押して，ページ内リンクのジャンプ先を確認してください.

このページはこのツールでチェック可能なガイドラインの項目全てに準拠しています.
ユーザによる確認が必要な項目を詳細レポートで確認してください.

evaluation	score
Compliance	100
Listenability	100
Navigability	96

(b) Test results in terms of screen reader-aware design

Fig. 7. Results of evaluation test using aDesigner

[2] A **screen reader** is a software application that attempts to identify and interpret what is being displayed on the screen.

6. Conclusion

We have proposed and implemented a Web system that comprises a Web crawler, and a simple Web database technology that runs on an open source Wiki platform. The system was applied to career support Web system as a pilot project. It was confirmed with a test tool that the system is compliant with the rules and guidelines for Web content accessibility. Currently, only five fields out of the total of 15 per site have been filled up. Therefore, we need to implement extensive search modules for the remaining fields such as the number of beds.

Acknowledgments

This work was supported by Grants-in-Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT), and was performed under the program for improving education for visually impaired students majoring in health sciences.

References

[1] The World Wide Web Consortium (W3C): Web Content Accessibility Guidelines 1.0, 1999 (http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/).
[2] CPAN (Comprehensive Perl Archive Network): http://www.cpan.org/
[3] FreeStyle Wiki: http://fswiki.org/
[4] Wikipedia: http://www.wikipedia.org/
[5] The World Wide Web Consortium (W3C): CSS Techniques for Web Content Accessibility Guidelines 1.0, 2000 (http://www.w3.org/TR/2000/NOTE-WCAG10-CSS-TECHS-20001106/).
[6] Kentarou Fukuda, Hironobu Takagi, Junji Maeda, Shin Saito, Chieko Asakawa, "aDesigner: tool for improving Web accessibility," Journal of Japan Society for Software Science and Technology, 2006, pp. 26-35.
[7] Architectural and Transportation Barriers Compliance Board: "Web-based intranet and internet information and applications," *Federal Register*, 2000, Vol. 65, No. 246.
[8] JIS X 8341-3:2004: Guidelines for older persons and persons with disabilities -- Information and communications equipment, software and services -- Part 3: Web content, 2004.