Windows Automated Site Map Generator

Vamsi K. Yenigalla

December 2008

Department of Computer Science
California State University, Fresno

Windows Based Automated Site Map Generator

Vamsi K Yenigalla

December 2008

Department of Computer Science
California State University,
Fresno, CA 93740

Project Advisor
Dr. Shih-Hsi Liu

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

Abstract

Windows Based Automated Sitemap Generator is a Windows application which brings ease in generating XML sitemaps that are necessary for building SEO (i.e. Search Engine Optimization) friendly websites. Currently there are PHP based sitemap generation programs that run as corn job (i.e., automated server services) on Linux based systems. Control panels like Plesk and cPanel provide good interface for setting up these corn jobs. Windows based systems could not take the advantage of these unattended automation processes requiring administrators to run the job themselves. Novice Web programmers are in need of a system that has simple interface and time scheduling system that can generate sitemaps and upload them to the Web space. Once uploading the sitemaps it is required to notify all major search engines in order to specify them regarding the change.

The main advantage of a client side sitemap generation is avoiding the load on the server. A typical sitemap PHP program requires immense amount of processor resources and also bandwidth. Client side sitemap generation can be done from any remote system. Once the sitemaps are generated users have the capability to analyze it and take appropriate actions. Generating sitemaps brings in control to the Web programmers on how they want the search engine crawlers to behave. With the help of keyword analysis, different methods of optimization and de-optimization can be applied to a certain website using sitemaps. Specifying all the fields for each page or directory is a difficult task as the modern day websites are immense in quantity of pages and content. To tacking this problem Sitemap Generator program will calculate the fields using the pattern in which they are updated to specify keywords such as hourly, weekly, monthly or yearly. This saves a lot of time for the webmasters who are forced to choose a single entry time for an entire directory.

Preface

Most people look at Google and Yahoo and wish their website was in the first page. To bring it into action the first thing is generating a good sitemap along with the *Search Engine Optimization* Techniques.

Section 1 Introduction and advantages of sitemaps and motivation.

Section 2 Background of sitemap introduction, sample sitemap and its components and sitemaps in detail.

Section 3 Discussion about existing work and their disadvantages and analysis

Section 4 Time schedule followed

Section 5 Discussion about the Software Design Life Cycle

Section 6 System Design Document

Section 7 Detailed Implementation Procedures and Important code snippets.

Section 9 Conclusion and Future Work that can be implemented.

TABEL OF CONTENTS

Acknowledgement

First, I take the pleasure to thank my Project Advisor Dr. Shih-Hsi Liu, for his continuous support, interactive sessions, constant pursuance towards practical knowledge and guidance throughout the master project. It was under his guidance I have gained lot of practical knowledge through different types of projects throughout the semester. He showed me different ways to approach a problem and the need to be persistent to accomplish my goal. He is always there to meet and talk about my ideas, and help me think through my problems. He also provided me with necessary assistance in writing the report. I would like to thank Dr. Seki for her generous support and many useful discussions. Special thanks go to Gloria for her support and encouragement.

I would also like to thank all the faculty members and staff from the Computer Science Department, for helping me at any time throughout my academic career at Fresno State.

Most importantly, I thank my family: my parents, Mr. Gopi and Mr. Seshu for allowing me to be as ambitious as I wanted and reminding me value and importance of honesty. It was under their watchful eye that I gained so much drive and an ability to tackle challenge ahead on.

I would like to thank all my friends Shwetha, Pranay, Gopi, Santosh, and Ravi for providing their immense support for my experimental results.

1. Introduction

1.1 Introduction to Windows Based Automated Sitemap Generator 1.0

Automated sitemap generator is a tool to easily form and upload sitemap files for any particular website. Each website has different skeleton and formation. Search engines easily get confused and may skip some of the key pages which are very vital for survival of a particular website. Automated sitemap generator supports scheduling where a novice user can give minimal details and generate a sitemap file. This file will be automatically uploaded to the Web server with the help of the FTP details specified. The program uses FTP protocol to connect to a Web server and transfer the file using Stream IO method. Once the upload is done there is an option where the user can specify the list of the search engines to be notified.

The program uses search engine API calls (i.e. Application Programming Interface Calls) to actually notify them. The search engine notification option list can be scaled to a more number of options to choose from. Once scheduling the program runs as a background worker and keeps on listening for a particular date on the schedule to begin. Once a date is found the programs runs the sub routine where it generates and uploads a sitemap file to the server. It also notifies all major search engines in this process. Once a sitemap is generated it checks with the previous sitemap. It compares the number of URLs from the previous sitemap files. For example, if there are 100 pages in the current version and 79 pages in the previous version the percentage increase is 21%. So it automatically sends an email to the Webmaster regarding this and recommends him/her to make changes to the schedule. This approach is different where the schedule suits to the actual Website and Webmaster settings.

1.2 Motivation

The main motto of this project is to design a Windows based program where a novice user can easily automate the process of sitemap generation and also limit the effort it takes to generate a valid effective sitemap.

Below are the motivation factors for automated sitemap generator.

- **Ease of Use**

  Some of the existing systems lack clean simple user interface as they are Web based. Users should browse to a particular Web address and give some parameters so that the sitemap is generated. The process takes immense amount of time depending on the size of the website. The existing programs are prone to freeze without giving much of information there by prompting the webmasters to restart the program.

- **Strong Authentication**

  Existing programs are usually accessed through a Web address hence making it easy for hackers to get hands on it. The only way to resolve this is to make the directory level password protected. Hence automating the process in Linux based control panels gets complicated as users should build custom parameters to pass through get method. Using programs like Microsoft Network Monitor 3.2 [1] will allow the intruder to see the secure information like FTP authentication etc. Sitemap Generator program tackles this by storing these details securely as it's a client side application.

- **Automation**

  Existing program requires immense amount of manual work where these tasks can be automated. They rely on a point where the program should be customizable but this makes the task of the webmasters very difficult. Sitemap Generator program automates the tasks that are not required to be customized thereby saving labor cost and work hours.

- **Training Period**

  Existing programs are complicated; this complication brings in another critical fact where training takes longer than needed. To tackle this problem the user interface is designed in such a way that even a novice user can generate and upload a sitemap file.

- **Deployment**

  Existing programs consume considerable amount of Web space and bandwidth which are crucial in the Internet applications. By making the sitemap generation process client side valuable Web space is preserved. Some programs save the previous copies of the sitemaps for comparison.

1.3 Summary of Achievements

Windows based Sitemap Generator is designed with the purpose of making the process of sitemap creation easy for the novice webmasters. One of the main goals of the project is to eliminate a lot of manual work that needs to be done after generation of sitemaps and in upload functionality. There are programs that perform similar work in Linux based environment by using the capabilities of corn jobs (i.e., automated time scheduled jobs which are performed unattended), but a need for Windows based systems with simple scheduling methodology is fulfilled by this project.

2. Background

2.1 Introduction to Sitemaps?

A sitemap is XML representation of all the files in the root folder of a website. The XML follows the schemas mentioned in the w3.org website. As we know many spiders crawls a site as a day-to-day activity. These spiders work by crawling a page and queuing the outgoing links and coming back to crawl them again. In this scenario important files may be missed due to non-existence of the link to the file on any of the physical Web page or due to the on-the-fly creation of file itself. Current competition in many fields requires every site to optimize their websites and get as much as traffic from search engines as possible. To tackle the problem of missing pages search engines came up with structured representation of the files on the Web server which can be easily fetched and used as a guide in crawling. Sitemaps were first introduced by Google and soon other search engines followed the lead.

2.2 Sample Sitemap

```
<urlset xmlns:xsi = http://www.w3.org/2001/XMLSchema-instance>

    <url>
        <loc>http://www.csufresno.edu/</loc>
        <priority>0.75</priority>
        <changefreq>daily</changefreq>
        <lastmod>2008-09-03T22:18:39+00:00 </lastmod>
    </url>
</urlset>
```

A basic valid sitemap consists of at least one <url> node with some optional sub-nodes. <loc> designates the physical Web address from where the crawler can fetch the current page. <priority> designates the page's priority with respect to other pages in the same root. For example, a "Home" page may have priority 1.0 where as a "Contact Us" page may have priority 0.25. <changefreq> denotes the modification frequency of a webpage. For example a news website may change every 15 to 60 minutes where as a site like Wikipedia may change weekly. This field saves a lot of bandwidth as crawlers will crawl only the pages that are supposed to change frequently than other pages. <lastmod> this field denotes the last time the page was changed.

2.3 Do Search Engines Care?

All major search engines like Google, Yahoo, Ask, Live and many more follow sitemaps while crawling. It has two-way advantages, in the sense that the load on the crawler and that on the Web server both are minimized with valid sitemaps. Search engines try to crawl as many websites as possible in any given day. With increase in websites two to three folds every year there is a need for effective solution for this search patterns.

2.4 Sitemaps in Detail

Sitemap is an XML file with each node representing a URL in the respective website. Sitemaps allow search engines to crawl complicated websites that require extra fields to crawl. It is a structured representation of multi level links related to a website in a simple XML file. It makes sure that the required priority is given to the important and frequently updated pages rather than infrequent ones.

The structure of a typical sitemap consists of different child nodes. A sitemap's first node is usually protocol related to the XML version used in building it.

<?xml version="1.0" encoding="UTF-8"?>

The next node specifies the namespace of the XML currently being used

<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/1.0">

<urlset> node is always followed by </urlset> enclosing child nodes if any.

Each link in a website is enclosed with in a URL field. The URL field has many different optional child fields.

- Loc
- Lastmod
- Changefreq
- Priority

Enough care should be taken to avoid directories that we wish the crawler shouldn't crawl. Many websites use a simple txt file to specify this called robots.txt usually in the root folder. We can use commands such as Disallow: * to disallow any of the directories to be crawled. Or Disallow: admin to eliminate admin directory from being crawled. Most of the famous search engines respect this file but this method is not guaranteed as some copying crawlers just ignore this file. There is no method to make information safe online. This feature can be overridden by sitemap so enough care should be taken to eliminate such directories from the sitemap generation path. One way to do this is allow two fields which state the directories that should be included and once that shouldn't be.

<loc> field is the physical location of the current URL along with the protocol. On some server settings http:// differs from https://. In short they use separate directory structure for each content directory. The entire value should be less than 2048 characters. If there is a lengthier URL an exception should be used.

<lastmod> field is the field that specifies the last modified date and time of a particular page. These properties are basic for any server and can be accessed easily in any development environment. This field is very productive as it saves a lot of bandwidth by elimination from crawling the pages that are not modified from the last crawl. The date time format should be that of W3C (http://www.w3.org/TR/NOTE-datetime) a standard followed by many crawlers.

<changefreq> field determines the approximate frequency of change expected in a particular page. E.g. a typical news website like cnn.com may have “always” value to this field as the content changes very frequently sometimes in minutes. The field can bear values like hourly, weekly, monthly, yearly or never.

<priority> field determines the priority of a particular page with respect to other pages in a particular domain. The default value for this is 0.5, It varies from 0 to 1 depending on the priority we wish to have.

Usually a sitemap is at the root level of any domain e.g. *http://www.exp.com/sitemap.xml* this is optional. If webmaster wishes to change it to a different name the path should be specified in robots.txt. robots.txt is the first file that any crawler tries to find for initial information regarding a site.

Many tools exist to validate a sitemap which are free of cost. It returns the errors that may exist in a particular sitemap.

e.g. http://www.xml.com/pub/a/2000/12/13/schematools.html

3. Related Work

Existing Systems

Many sitemap generation programs are available that run sitemap generation as a corn job (i.e. an automated task). What they lack is direct compatibility to Windows based hosts. Windows based Web developers need a handy tool which they can use to schedule generation of sitemap and analyze it for any flaws. The application should be simple and dynamic in nature with a lot of statistics. The application should automatically pings 4 to 5 major search engines using their API calls once the sitemap is generated and uploaded to the Web space.

3.1 Disadvantages of Existing Systems

- All of the existing systems[4] are Linux Web server based making it hard for entry level webmasters.
- Some of them lack communication where the user is not notified about the whole process.
- Work done by the user is huge. In some cases thousands of files are given where the user should manually select the priority and change frequency fields.
- Most of them are server based and utilize a lot of band width and server processing.
- Many of them lack automation in filling the sitemap XML fields.
- Some of them are not W3C standards complaint.
- Many of them lack good simple user interface and end up confusing the webmaster.

3.2 Case Studies

3.2.1 auditmypc.com

auditmypc.com/xmlsitemap.asp[8] is the location of the current tool. In the following section this tool is analyzed to know its disadvantages.

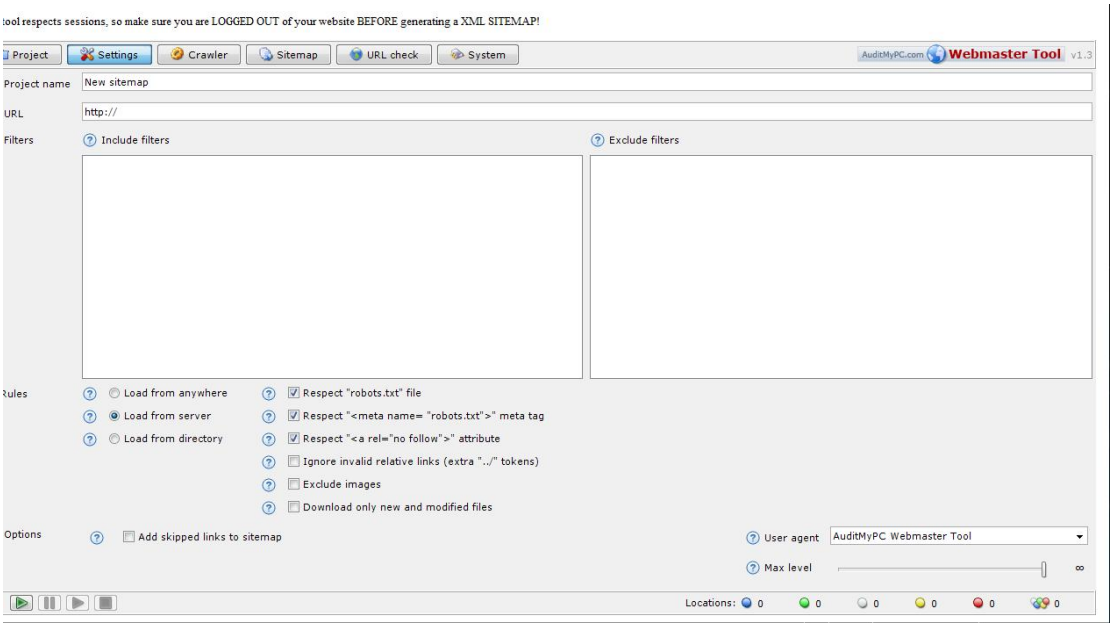


Figure 1 Screenshot of the tool

Analysis

1. No URL elimination scheme. There is no ability to eliminate certain key directories in a website file.
2. File type selection is not present. Certain programs allow webmasters to eliminate some file types like picture files.
3. No scheduling is available. The webmasters should frequently generate sitemaps using this program.

4. Time Schedule

4.1 Time Schedule

Sep 3rd – Sep 15th	Project proposal design
Sep 15th – Sep-30th	Project Agreement Signing
Oct 1st – Oct 10th	Project Planning and Deciding technologies to use.
Oct 10th – Nov 15th	PHP coding for back end server side implementation. .Net coding for windows forma application.
Nov 15th – Nov 20th	Project report writing.
Nov 20th – Dec 5th	Correction where required according to the comments made.

4.2 Flow Chart

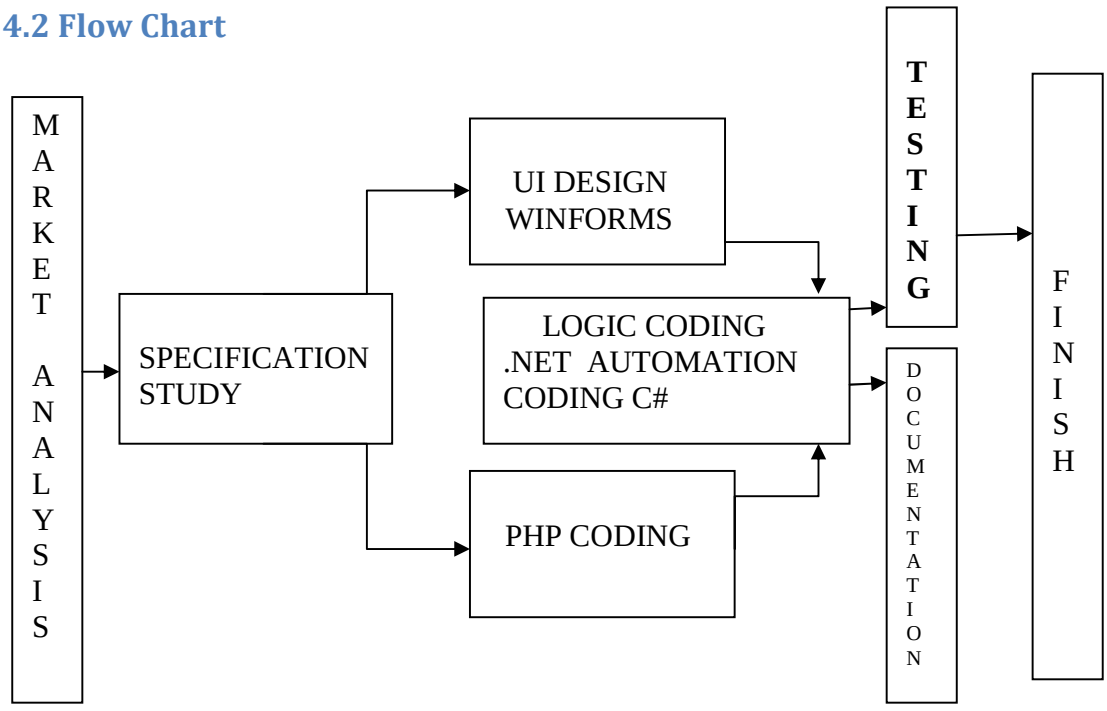


Figure 2 Design Flow Chart

5. SOFTWARE LIFE CYCLE MODEL

Every software development process follows a basic life cycle to develop a software system. Software life cycle model is commonly referred as software development life cycle (SDLC).

SDLC is composed of several phases and it is a systematic approach to problem solving

- Analysis
- Design
- Development
- Testing
- Deployment

Waterfall model is used to develop Windows Based Automated Sitemap Generator. For the reference purpose please find below Water Fall model diagram.

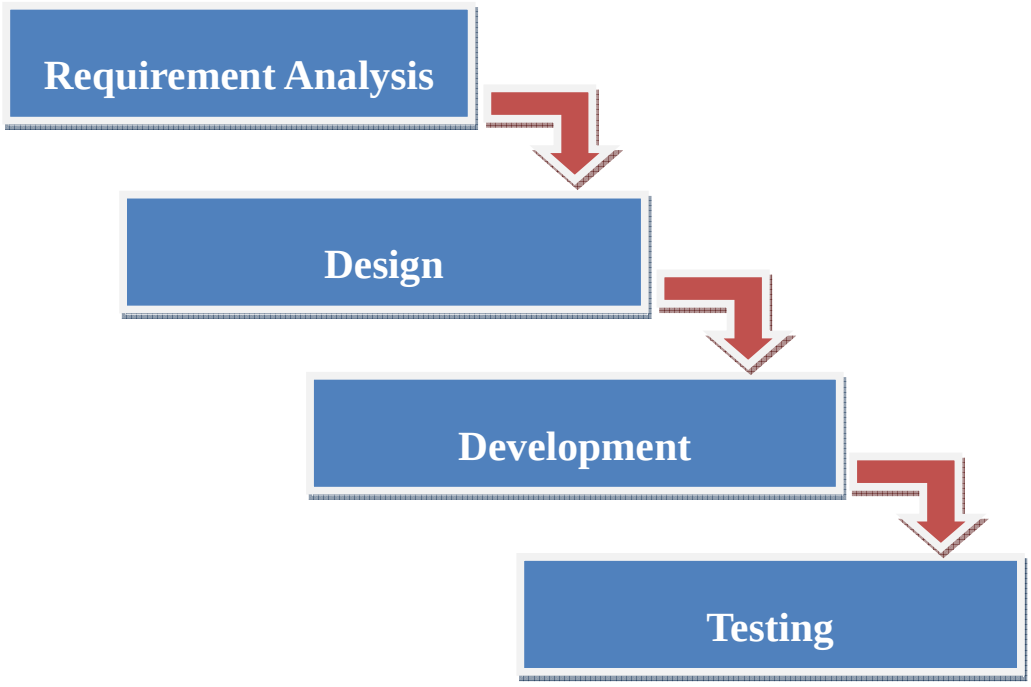


Figure 3 Waterfall Model Block Diagrams [12]

5.1 Requirement Analysis

Developers and IT specialists gather and analyze the requirements in this phase. This is very crucial phase for any successful project development. In this project development analysis is done by identifying end users and need for the system. The end users in the present scenario are web masters and the requirements are divided into functional and non-functional classifications and are discussed in next chapter [6]. It is very important to identify new system users, making them comfortable with the implication of the new system and to satisfy all their needs.

5.2 Design & Development

Front end design is done using .Net bases Windows forms application. As .Net framework is made popular by Microsoft Vista[12] platform it is a easy way to create applications rather than java[8] classes. Back end need is eliminated by using configurations files as they are secure and robust. App.config file contains and stores all the necessary details for the program to execute correctly. A helper file AppSettings.cs is used to write classes to access the variables in App.config file. PHP is used at the core of the backend where it provides web server related input to the .Net Windows Application.

5.3 Testing

The implemented code is deployed on the system to execute without errors. Many test cases are written and executed in this project which can be found in chapter [8].

5.4 Deployment

Deployment phase is not implemented in the current project. Deployment phase is related to client side installation of the project and performing maintenance work.

6. REQUIREMENTS DOCUMENT

6.1 Purpose

The main purpose of the system is to build a Windows based program that is easy to schedule even by novice webmasters and also to minimize the manual work that needs to be done in assigning change frequency and other fields in sitemaps.

Functional Requirements

- To develop a FTP module that connects to the Web server.
- To develop a user settings module that stores the user details in a configuration file.
- To develop a scheduling module that brings in ease for scheduling to the end users.
- To develop a ping module that pings all major search engines after successful generation of sitemap.

Non Functional Requirements

- Eliminate manual work in the sitemap generation process.
- Create good user interface.
- Simplify the scheduling process.
- To design multi-platform compatible system.
- Automate and improve the interval between each sitemap generation for better Search Engine Ranking.

Design Goals

- To provide secure authentication for automated generation of sitemaps.
- To provide user friendly, easy and self explanatory forms.
- To bring higher level of usability in sitemap generation.

6.2 Use Case Model

Use case model is UML[2] (Unified Modeling Language) representation of the functional requirements of a system. Entities are identified according to the requirement. The use case model is model representation of functional representation along with dataflow and relations between the entities and modules.

Entities

- End user - webmaster or novice user who uses the system to generate sitemaps.
- System - The sitemap generation program itself.
- Web Server – System where the files are stored on which the sitemap generation operation should be done.

6.3 Use Case Diagrams

Use Case	Scheduling
Participant	End user

Actions	1. End user schedules the generation process by selecting the interval. 2. Grid is populated automatically with scheduling for next 15 intervals. 3. Thread is started which constantly checks for date match between current date and scheduled date. 4. Once a match is found Generation process starts.

Table 1 Use Case – Scheduling

Use Case	**Generate**
Participant	System
Actions	1. Once User clicks “Generate Now Button”. 2. System generates an XML file containing the sitemap of the pages in Web server. 3. Once sitemap is generated the file is available for analysis phase.

Table 2 Use Case – Generate

Use Case	**Ping**
Participant	End User
Actions	1. End User selects the Search engines to ping from Ping tab. 2. Then he clicks “Save Settings” button. 3. After the sitemap generation the search engines are pinged accordingly.

Table 3 Use Case – Ping

Use Case	**Upload**
Participant	System

Actions	1. After sitemap generation FTP module becomes active. 2. FTP module fetches the generated XML file and uploads it to the Web server.

Table 4 Use Case – Upload

Use Case	Analysis
Participant	End User
Actions	1. After sitemap generation user clicks the Analyze tab 2. Once in analyze view user can analyze the sitemap generated and make necessary changes.

Table 5 Use Case - Analysis

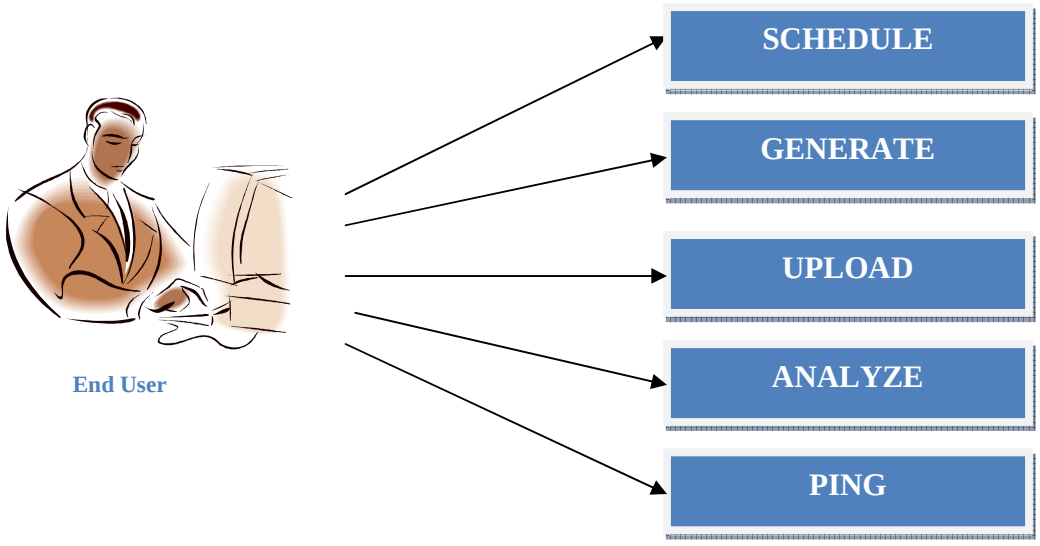
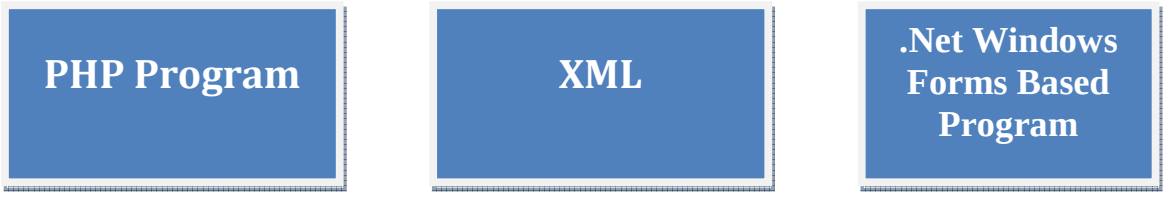


Figure 4 End User and System Events or Actions

As shown in Figure 4, end user and system events or actions diagram represents graphically all the actions or events performed by the end user or system. Schedule node in the above diagram starts a new thread that stays alive as long as the schedule is active. Once cancel schedule button is clicked the schedule thread is terminated.



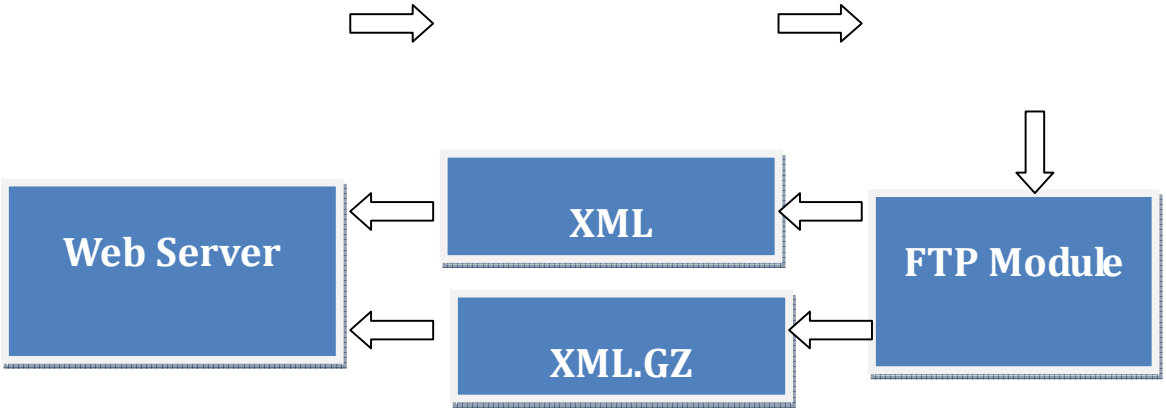


Figure 5 Data Flow Diagram

Data Flow Diagram (Figure 5) states the flow of data between all the major components in a system. The communication between these modules is achieved as XML format. FTP module uploads the XML file in two different formats, one a regular XML file and the other one xml.gz[6] file.

6.4 Standards Followed

The sitemap generated are according to the W3C (i.e., World Wide Web Consortium) standards. The sitemaps generated are validated using XML validators[5] that are available on Internet which validate the files according to the standards of WC3. The program is built without using inbuilt timers as they are prone to be inefficient. Instead threads are created dynamically when the need arises, so as to keep the program efficient and processor sensitive.

6.5 Technologies Used

The program is designed for Windows environment which works on .Net framework[4]. The Web server interaction part is done using PHP. Initially during settings phase a file is uploaded to the server which passes information like the directory structure and last modified time to the program. The program then uses these parameters to automate the job of calculating the fields in sitemap file such as change frequency, and priority.

The program is built based on .Net 3.0 framework[4] in Visual Studio 2005 IDE. The backend of the program is done using configuration files as they are more secure than database and also their ease of use and availability. PHP5[7] compatible programming is done on the background worker file.

6.6 Minimum System Requirements

- Processor Speed 1GHz or more
- RAM 1GB or more
- Operating System Win XP or later
- Framework .Net 3.0
- Internet Access

Minimum Server Requirements

- PHP5 compatible
- High Processor speed.

7. IMPLEMENTATION

Implementation section deals in details the important logic in coding, screen shots and working of the program.

7.1 Front End Implementation

Front end implementation is developed using Windows Forms Application template in .Net Framework.

Basic Structure of Windows Based Automated Sitemap Generator

Different modules and sub modules are shown below

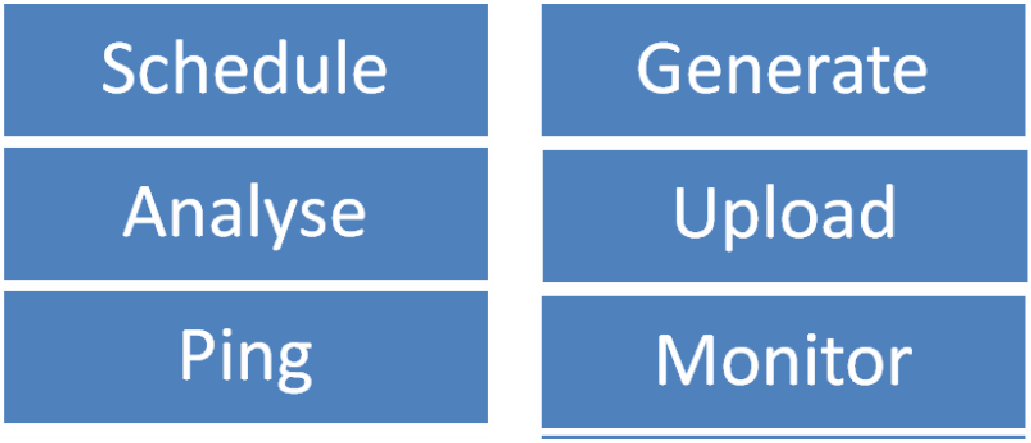


Figure 6 Basic structure of WASMG

Each module of the above mentioned Figure 6 is described in detail below.

Advantages of using .Net

- **Easy development and deployments**

  Creating .Net application in Visual Studio 2005[3] is not as tough as other programming. Creating user interface in visual studio is easy as the IDE has drag-

and-drop facility. All the properties and events can be accessed from the properties window for any component.

- **Built In Validation**

  Validation functions (For example SSN number validation, Phone Number validation etc.) in .Net applications are easy to use and customize. Custom validation functions can be easily written and can be integrated. It supports client and server side validation (using Ajax[14]).

- **Multi Programming Support**

  .Net applications can be written in more than 30 different languages, which ultimately get converted into MSIL[9] Microsoft Intermediate Language. MSIL is a low level language which can be understood by the .Net compiler. .Net framework supports many languages like C#, Visual Basic, VC++, VB Script and JavaScript.

- **In-built Security Features**

  .Net framework supports inbuilt authentication support. So designing a Login form is very easy which require minimal coding. It allows method level security and also strongly supports OOPS (Object Oriented Programming System) functionalities like data abstraction and encapsulation.

7.2 Designing the User Interface

While designing the user interface the primary goal is to bring in ease to the end user. Even a novice user should be able to use the system with minimal effort and training. The basic entity to start with in a Windows based application is the form control. While designing the forms the primary things to consider are simplicity, placement on screen, consistence with design and the colors used.

- **Localization**

  During the market analysis phase the target audience of the application is studied. .Net has the simplest mechanism to cater the needs of different people and their languages. There is a file specifically for language parameters called resx file. Once controls are labeled in a default language the program automatically converts them into many different formats if required. Not all languages are inbuilt but it is easy to import fonts and define new languages.

- **Font and Color**

  Fonts should be kept consistent throughout the application. It is better to use the default fonts of Windows as they are available widely. It's better to stick to fonts people are used to rather than complicated looking fonts. End users need simple decent look and nice functionality. Good font should be used to show important information, and decorative fonts for visual effect.

  Use of colors is another important factor in bringing interest among the end users. People differ in color likening. Some people like vibrant and rich colors where as others like dull and non shiny color. Choosing appropriate colors is a critical step in User interface design.

7.3 Software Architecture

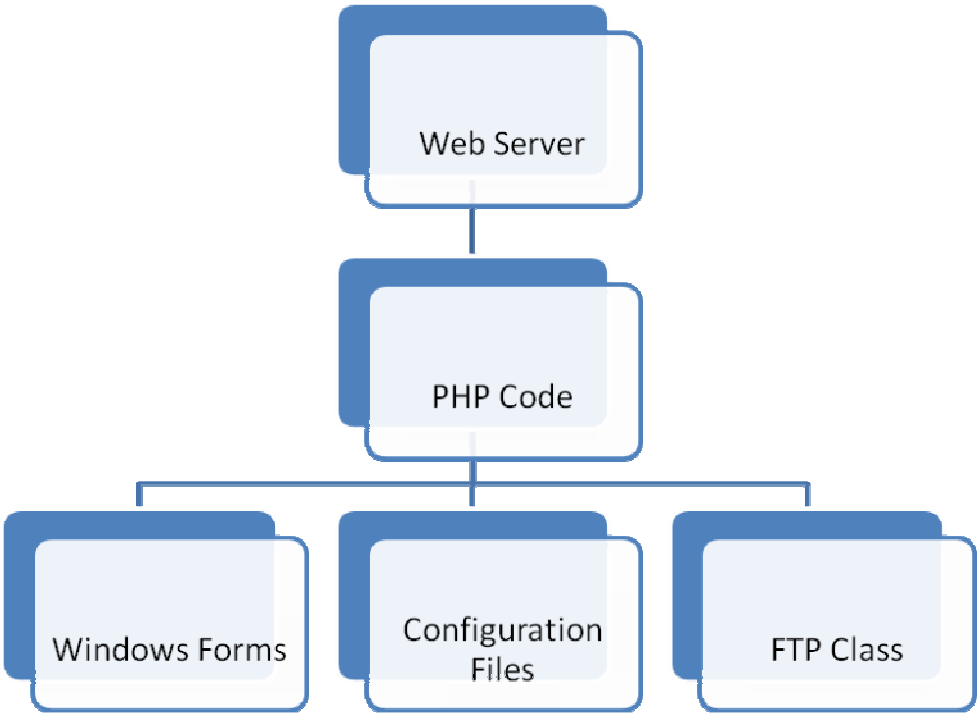


Figure 7 Software Architecture

Each component of Figure 7 is described in detail in the following subsections.

- **Web Server**

  Web Server in this case is the machine where the website resides. The Web server should be compatible with PHP5 version for the intermediate PHP code to function properly. It can have any Operating System like Windows, Linux etc. The details required by the end user are the FTP authentication details. Almost 99% of Web servers support FTP protocol (File Transfer Protocol). The FTP server address or the file server address and the email account.

- **PHP code**

  Implementing XML reading and writing is a complicated task so a file called XML regex[10] file is used and included in the background worker file, as shown in Figure 8. The REGEX file contains classes which are implemented in the background worker file.

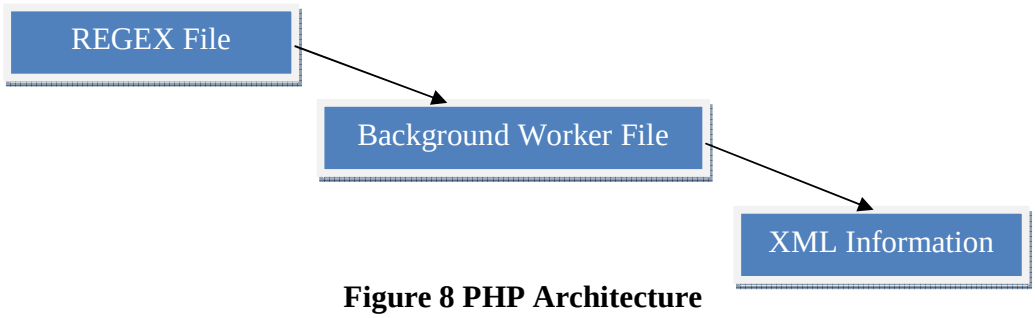


Figure 8 PHP Architecture

```
<?php
header("Content-type: text/xml");
$xml_Open  = "<?xml version=\"1.0\"?>\n";
$xml_Open .= "<urlset>\n";
echo $xml_Open;
$xml_close = "";
$basedir = ".";
```

Figure 9 PHP Code Blocks -1

Explanation

- The PHP program starts with <?php and ends with ?> tags
- The output format expected is an XML file so the header("Content-type:text/xml") defines it in line 2.
- The variable $xml_Open is used to hold the XML output until the end where it is echoed (i.e. printed onto a file stream).
- $xml_Open is filled with the XML schema information in line 3
- The variable $basedir holds the base directory information. Dot denotes the current directory.

```
function getdepth($fn)
{
  return (($p = strpos($fn, "/")) === false) ? 0 : (1 +
getdepth(substr($fn, $p+1)));
}
```

Figure 10 Code Block II

- The function getdepth returns the depth of the deepest directory using recursion.

```
function printlink($fn){
                    $url_open = "\t<url>\n";
                    echo $url_open;
                    $loc_open = "\t\t<loc>";
                    echo $loc_open;
                    echo curPageURL().$fn;
                    $loc_close = "</loc>\n";
                    echo $loc_close;
                    $lmod_open = "<lastmod>";
                    echo $lmod_open;
                    echo date( "m-d-y", filemtime($fn));
                    $lmod_close = "</lastmod>\n";
                    echo $lmod_close;
                    $cfreq_open = "<changefreq>\n";
```

Figure 11 Code Block III

- The function printlink prints the skeleton of each URL as an XML node in the intermediate XML file used for transferring information.

```
// main function that scans the directory tree for Web
pages
function crawldir($basedir){
   if ($handle = @opendir($basedir)) {
      while (false !== ($fn = readdir($handle))){
         if ($fn != '.' && $fn != '..')
         { // ignore these
            $dir = $basedir."/".$fn;

            if (is_dir($dir))
              {
               crawldir($dir); // recursive call to this
function
              }
              else
              { //only consider .html etc. files
                  if
(preg_match("/[^.\/].+\.(htm|html|php)$/",$dir,$fname
))
```

Figure 12 Code Block IV

- This is the main function that scans the directory structure and passes the information to printlink function.

- **Windows Forms**

  The detailed explanation of Windows form is described in the following section.

7.4 Forms Explained

Windows forms are the initial steps and direct implementation towards end users. Principles like good color selection font selection and validation are implemented in the forms. A tab bar is used for easy navigation throughout the application.

The forms are divided into five different sections

- Intro
- Settings
- Schedule
- Analysis
- Ping

7.4.1 Intro Tab

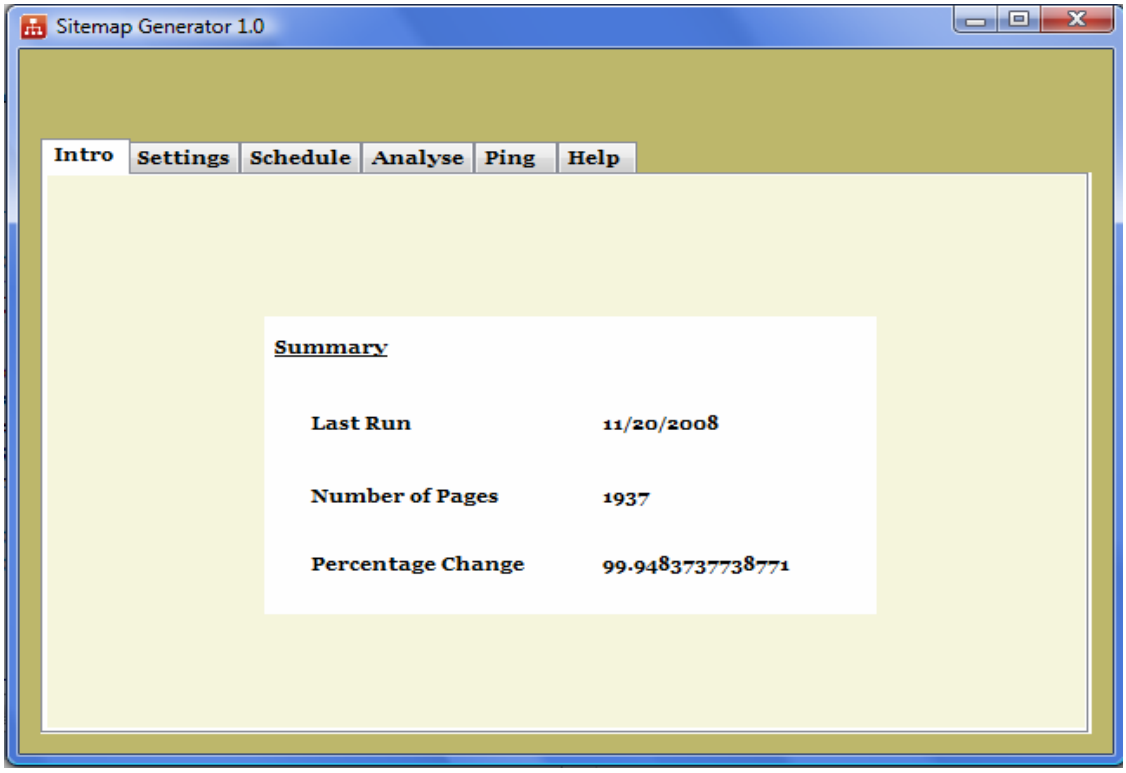


Figure 13 Intro Tab Page

The introduction form has summary information such as last run, number of pages scanned and percentage change etc.

- **Last run**

  The information in this field is the date part of the DateTime.Now function in C#. The time part is stripped to make it look simple.

- **Number of Pages**

  This field shows the number of pages that are included in the sitemap that is generated. This is the count of the number of URL set nodes in the sitemap file.

- **Percentage change**

  Percentage change is the change in percentage of the number of pages scanned now to that of the previous time. This information is used to trigger an event

where if the percentage change is more than a certain percentage, say 20%, the program triggers a sendEmail function which sends an email to the webmaster. The email has recommended information which usually is to recommend increasing the frequency of sitemap generation.

7.4.2 Settings Tab

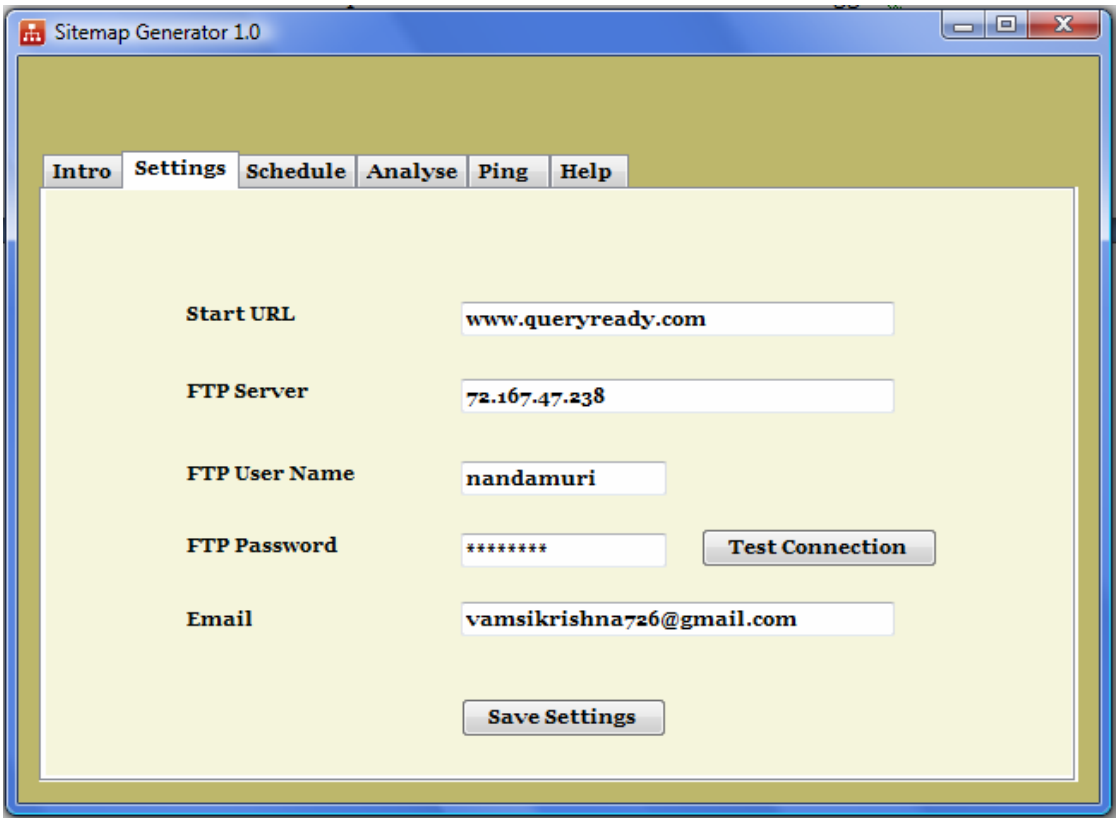


Figure 14 Settings Tab Page

The settings tab contains five main fields including

- Start URL
- FTP Server
- FTP User Name
- FTP Password
- Email

- **Start URL**

  The start URL field is the starting directory level from which the website should be scrolled. This can be a home directory or an intermediate directory. The

information is used while sending the Web request method in .Net to PHP helper file.

- **FTP Server**

  FTP Server field contains the server IP address or Web address of the FTP server. This information is used by the FTP class files in .Net to establish a connection to the FTP server.

- **FTP User Name & FTP Password**

  FTP access is very critical because if made public can be very critical and can do irrevocable damage to a website. Proper encryption and decryption algorithms should be implemented for hiding this data.

- **Email**

  Email information is used to dispatch email when the percentage increase in number of pages exceeds 20%. This information can also be used to dispatch other maintenance related information if the end user opts for.

7.4.3 Schedule Tab

The schedule tab contents are

- "Generate Sitemap Now" button
- "Schedule" button
- "Save Schedule" button
- "Cancel Schedule" button
- Interval selection numeric step-up-down box
- Schedule grid

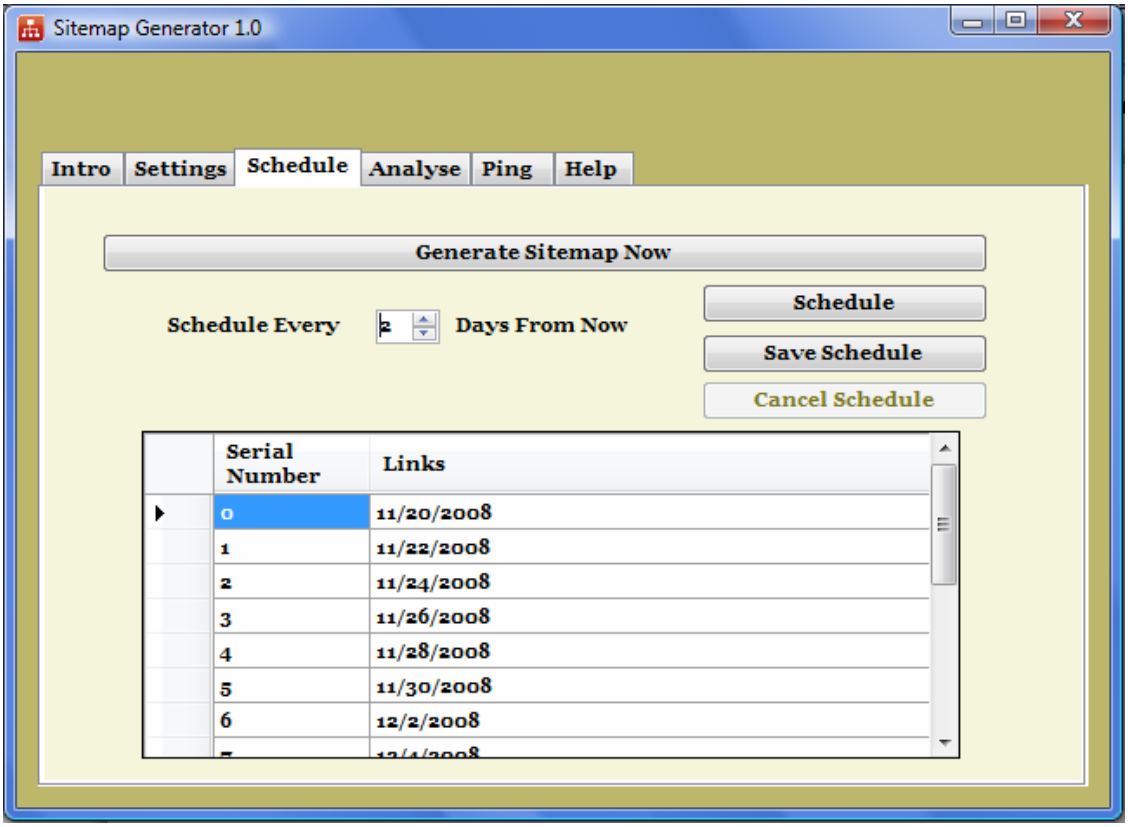


Figure 15 Schedule Tab Page

- **Generate Site Map Now Button**

  This button is used to generate sitemap and upload the file without any schedule at any given point.

- **Schedule button**

  The schedule button is clicked to activate schedule grid and interval selection input boxes.

- **Save schedule and Cancel Schedule buttons** are self explanatory as their name specifies their functionality.

- **Interval selection box**

  When the interval selection box value changes, then the schedule items on the schedule grid will update. These new values are saved or cancelled when appropriate buttons are clicked.

7.4.4 Analyze Tab

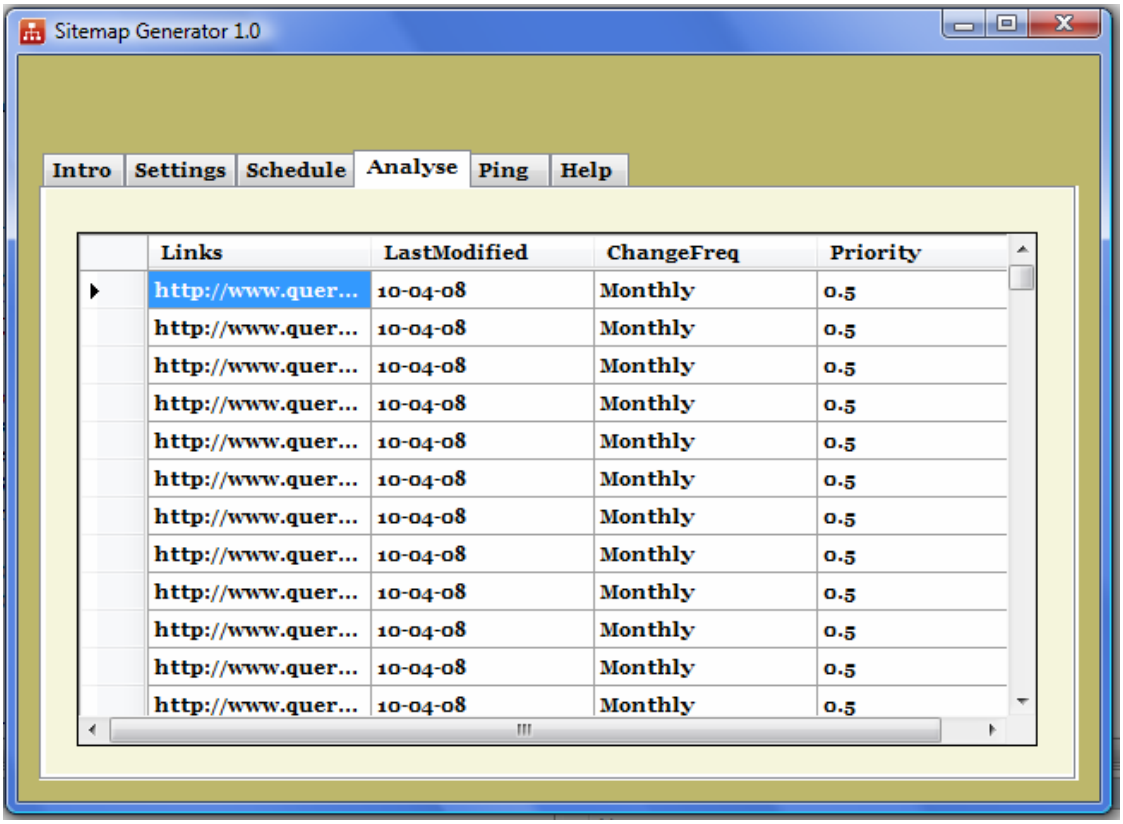


Figure 16 Analyze Tab Page

The analyze tab contains a grid which shows the sitemap of a particular site. End-user can browse through the grid to analyze the current output and can make changes to the settings.

7.4.5 Ping Settings Tab

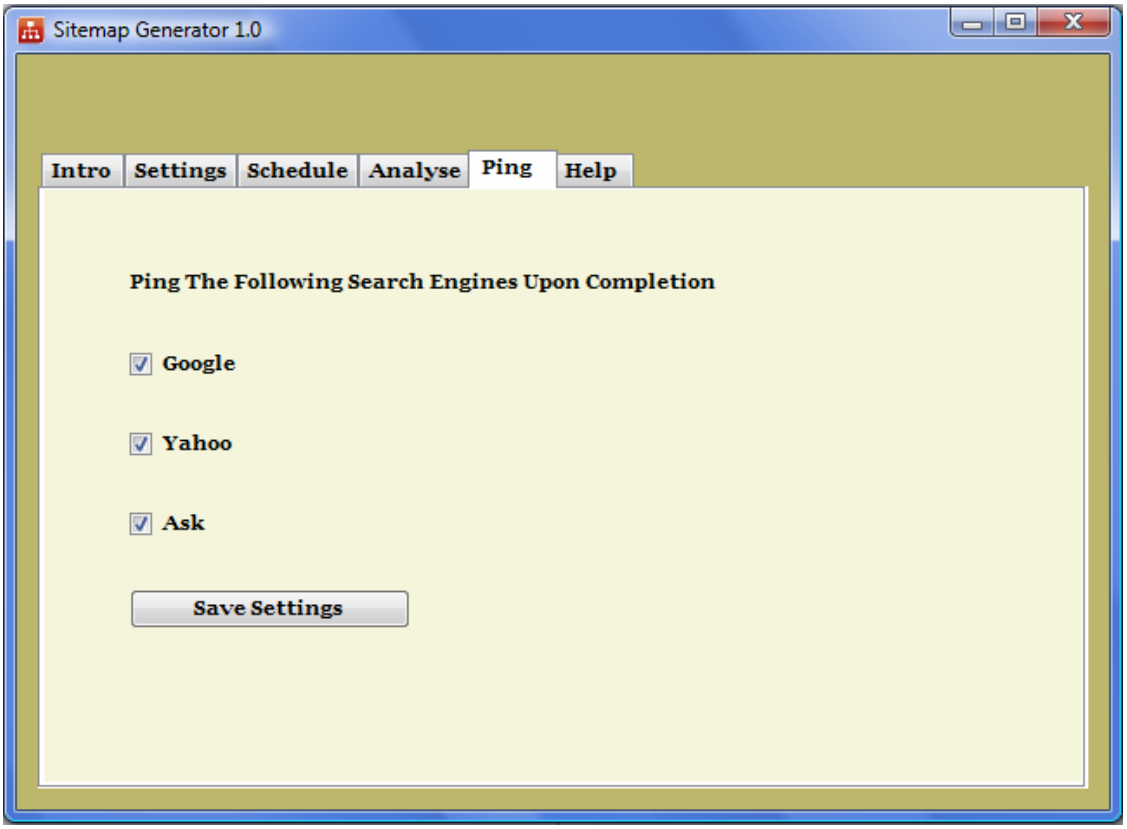


Figure 17 Ping Tab Page

The Ping Tab Settings menu is a simple form with check boxes. Depending on the settings selected the configuration file is updated and used to follow a particular URL calls to Ping search engines regarding the change.

The ping mechanism is done by URL calls specified in the respective API (Application Programming Interface) sections.

Ask.com:http://submissions.ask.com/ping?sitemap=http://www.domain.com/sitemap.xml
Google:http://www.google.com/Webmasters/sitemaps/http://www.domain.com/sitemap.xml
Yahoo: http://search.yahooapis.com/SiteExplorerService...com/sitemap.xml

The domain.com in the above URLs should be replaced with the actual domain name from the application settings area. Then a Web request is made from .Net application depending on the Search Engine ping options.

- **FTP Class**

  The FTP class file is used from an already existing resource. The FTP class file contains two main methods

  1. Upload
  2. Test Connection

- **Upload Method**

  Upload method takes a file name as a input format and uploads it to the specified folder using the FTP connection details. Upload method users streams for file transfer.

- **Test Connection Method**

  Figure 18 shown on the next page explains the code for Test Connection module which gets activated when user clicks on “Test Connection” button on the Settings tab of the program. Exception handling is used to avoid program crash. Exception handling is achieved by using try-catch keywords.

```
public void Upload(string filename)
        {
            FileInfo fileInf = new FileInfo(filename);
            string uri = "ftp://" + ftpServerIP + "/" + fileInf.Name;
            FtpWebRequest reqFTP;

            // Create FtpWebRequest object from the Uri provided
            reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" +
ftpServerIP + "/" + fileInf.Name));

            // Provide the WebPermission Credintials
            reqFTP.Credentials = new NetworkCredential(ftpUserID,
ftpPassword);

            // By default KeepAlive is true, where the control connection is
not closed
            // after a command is executed.
            reqFTP.KeepAlive = false;

            // Specify the command to be executed.
            reqFTP.Method = WebRequestMethods.Ftp.UploadFile;

            // Specify the data transfer type.
            reqFTP.UseBinary = true;

            // Notify the server about the size of the uploaded file
            reqFTP.ContentLength = fileInf.Length;

            // The buffer size is set to 2kb
            int buffLength = 2048;
            byte[] buff = new byte[buffLength];
            int contentLen;

            // Opens a file stream (System.IO.FileStream) to read the file
to be uploaded
            FileStream fs = fileInf.OpenRead();

            try
            {
                // Stream to which the file to be upload is written
                Stream strm = reqFTP.GetRequestStream();

                // Read from the file stream 2kb at a time
                contentLen = fs.Read(buff, 0, buffLength);

                // Till Stream content ends
                while (contentLen != 0)
                {
                    // Write Content from the file stream to the FTP Upload
Stream
                    strm.Write(buff, 0, contentLen);
                    contentLen = fs.Read(buff, 0, buffLength);
                }

                // Close the file stream and the Request Stream
                strm.Close();
                fs.Close();
                MessageBox.Show("Your file is uploaded successfully!");
            }
```

Figure 18 Code Snippet Test Connection

8. Testing

8.1 What is The Objective of Testing

The main aim of testing process in the project is to find out the critical and hidden issues. Finally application should meet all software requirements.

Below mentioned testing approaches are used to validate the testing process for this application.

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Installation Testing

8.2 Unit Testing

Unit testing is a process which will be used to test the unit of source code or single program or modules. This testing mostly done by development team and after completion of this testing other testing process follows.

Project is developed as three main modules, FTP module, PHP module, and Interface module. In this approach each module is tested separately, by checking each function individually to verify expected functionality is achieved or not by giving expected input values.

FTP module is tested by giving correct FTP authentication and also Wrong FTP authentication inputs. PHP module is tested on three different Web servers of different page count. Interface module is tested by giving different XML inputs. Code Walk through is done to eliminate unnecessary comment lines, blank spaces and to follow correct naming conventions.

8.3 Integration Testing

Integration Testing is a process which is done by combining modules together. In this testing test cases are written which covers the flow from one module to another module. This testing is mainly used to test the functionality of the application when two or more modules are combined together.

In this testing each of the modules FTP, PHP and Interface are validated by combining together. Test cases are drawn for all the modules and tested for integrity. Flow of variables from one module to another is tested in this approach. Information is passed in XML format from PHP module to Interface module. XML is verified to be valid by using online tools.

8.4 Functional Testing

The objective of this testing process is use to verify the functionality of the application. This testing will cover all functionality related test cases including positive and negative. Functionality testing is one of the important types in testing phases.

Functionality testing is achieved by testing correctness of each user actions related to the current system. A user action is something like a click event on a button, grid populating, xml file generation etc.

8.5 System Testing

Objective of system testing is to verify whether that application is meeting system requirements or not. In this testing process whole application will be tested at once.

System testing covers all the modules along with their interoperability. Test cases are written to cover all the flows, user actions and functionalities of the whole application.

8.6 Installation Testing

Installation testing is done by installing the program in different operating systems like Windows Vista, XP etc.

Perquisites to install this application

- It should be Windows operating system
- It may or it may not have .Net framework 3.5

So in this testing process required to test whether this application is able install successfully in all Windows operating systems like Windows Vista, XP etc. And also if .Net framework 3.5 is not available in any Windows based system it should install respective dependencies and proceed further to complete the process of installation.

8.7 Test Cases

Following test cases are executed to make sure the system validates for all the system requirements.

Test Case ID	Test Case Description	Steps	Expected Value	Pass /Fail
FTP_01	Verify whether FTP authentication Data is stored successfully in configuration file or not.	1. Run application exe. 2. Open settings panel. 3. Input FTP details. 4. Click “Save Settings” button. 5. Open configuration file.	FTP details to be stored to the configuration file.	Pass

FTP_02	Verify whether user is able to test FTP connection status using valid information.	1. Run application exe. 2. Open settings panel. 3. Input valid FTP details. 4. Click “Test Connection” Button.	FTP module should return a alert box with Message “FTP connection successful”	Pass
FTP_03	Verify whether user is able to test FTP connection status using invalid information.	1. Run application exe. 2. Open settings panel. 3. Input invalid FTP Details 4. Click “Test Connection” Button.	FTP module should return a alert box with Message “FTP connection Failed”	Pass
INT_01	Verify weather email is dispatched to the end users email account.	1. Run application for second time. 2. Verify the “Percentage Change” value from intro tab.	Email to be dispatched to the end users email account.	Pass

INT_02	Verify creation of robots.txt if not already present.	1. Delete robots.txt from Web server. 2. Run application exe. 3. Go to settings tab. 4. Input valid FTP details. 5. Click “Save Settings” tab. 6. Verify Web server files for robots.txt.	Robots.txt file should be created freshly.	Pass
INT_03	Verify functionality of “Generate Sitemap Now” button.	1. Run application exe. 2. Go to “Schedule” tab. 3. Click “Generate Sitemap Now” button. 4. Check “Temp” folder in C folder. 5. A valid XML file should appear with data populated.	XML file should be generated automatically and available for analysis pane.	Pass
INT_04	Verify “Ping Functionality”.	1. Run application exe. 2. Go to Ping Panel. 3. Click on the check boxes appropriately. 4. Click “Save Setting Button”	Use program like Live Wire to test the call to URLs related to search engine notification.	Pass

		5. Click “Generate Sitemap Now” button.		
PHP_01	Verify PHP file sitemapper.php	1. Pass values to sitemapper.php file using GET method. 2. Verify page count in xml generated.	When given correct values the php file should return XML with all the page information.	Pass

Table 6 Test Cases Combined Table

9. Conclusion

Sitemap Generator 1.0 is designed to make the process of sitemap generation automated and easy for novice webmasters. A lot of manual work in designing and deploying sitemaps is eliminated. The same work can be done using existing systems in Linux environment but the need for Windows based system is taken care by designing this system.

10.1 Future Enhancements

As time constrains limits the specifications of the current project some of the functionality is left for future enhancement.

- Priority based on relative importance of a page with respect to others can also be generated automatically.
- Detailed email communication with the end user can be achieved.
- Much more sophisticated scheduling considering time along with date will also be an improvement.
- Building directory structure too on the client side may be implemented for further efficiency.
- WCF may be used to improve the User Interface look and functionality.
- Advanced authentication modules can be added to bring in much needed security.

BIBLIOGRAPHY

[1] Microsoft Live Wire is a program By Microsoft to monitor network traffic, http://www.makeyougohmm.com/20060909/3766/.

[2] Martin Gogolla and Cris Kobryn (editor) "UML 2001 – The Unified Modelling Language", Proceedings of the 4th International conference on the Unified Modeling Language, Springer, October 2001.

[3] Visual Studio 2005 is a IDE application developed by Microsoft to Implement .Net Frame work 2.0, http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[4] Thuan L. Thai, Hoang Q. Lam, .Net Overview, ".Net Framework Essentials", Third Edition, O'Riellys, August 2000, Pages 1-9.

[5] XML Valuator, "Validating XML", http://www.peachpit.com/articles/article.aspx?p=31286&seqNum=12, 2000.

[6] GZip.org, "Introduction to GZip", http://www.gzip.org/#intro.

[7] Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre, Introduction to PHP, "Programming PHP", O'Reilly, 2006, Pages 1-9.

[8] Auditmypc.com, "Sitemap Generator", http://www.auditmypc.com/free-sitemap-generator.asp

[9] James D. Foxall, Microsoft Intermediate Language, "Sams Teach Yourself Visual Basic 2005 in 24 Hours", Publisher Sams, 2005, Pages 509 – 510.

[10] PHP Regular Expression, http://www.phpclasses.org/browse/package/3504.html.

[11] William R. Stanek, Getting to Know Windows Vista, "Introducing Microsoft Windows Vista", Microsoft Press, 2006, Pages 3-17.

[12] Select Business Solutions, "What is the Waterfall Model", http://www.selectbs.com/glossary/what-is-the-waterfall-model.htm, 2006.

GLOSSARY

- **Sitemap** – A hierarchical visual model of the pages of a website. Sitemaps help users navigate through a website that has more than one page by showing the user a diagram of the entire site's contents. Similar to a book's table of contents, the sitemap makes it easier for a user to find information on a site without having to navigate through the site's many pages. Also, in SEO, a sitemap can make it easier for a search engine spider to find all a site's pages.[1]

- **FTP File Transfer Protocol -** Short for File Transfer Protocol, the protocol for exchanging files over the Internet. FTP works in the same way as HTTP for transferring Web pages from a server to a user's browser and SMTP for transferring electronic mail across the Internet in that, like these technologies, FTP uses the Internet's TCP/IP protocols to enable data transfer.[2]

Citations

[1] http://www.Webopedia.com/TERM/S/site_map.html
[2] http://www.Webopedia.com/TERM/F/FTP.html

APPENDIX

sitemapper.php

```
<?php
header("Content-type: text/xml");
$xml_Open  = "<?xml version=\"1.0\"?>\n";
$xml_Open .= "<urlset>\n";
echo $xml_Open;
$xml_close = "";
$basedir = ".";

// function to count depth of directory
function getdepth($fn){
  return (($p = strpos($fn, "/")) === false) ? 0 : (1 + getdepth(substr($fn, $p+1)));
}

// function to print a line of html for the indented hyperlink
function printlink($fn){
                    $url_open = "\t<url>\n";
                    echo $url_open;
                    $loc_open = "\t\t<loc>";
                    echo $loc_open;
                    echo curPageURL().$fn;
                    $loc_close = "</loc>\n";
                    echo $loc_close;
                    $lmod_open = "<lastmod>";
                    echo $lmod_open;
                    echo date( "m-d-y", filemtime($fn));
                    $lmod_close = "</lastmod>\n";
                    echo $lmod_close;
                    $cfreq_open = "<changefreq>\n";
                    echo $cfreq_open;
                    $cfreq_close = "</changefreq>\n";
                    echo $cfreq_close;
                    $prio_open = "<priority>\n";
                    echo $prio_open;
                    echo '0.5';
                    $prio_close = "</priority>\n";
                    echo $prio_close;
                    $url_close = "\t</url>\n";
                    echo $url_close;
          }

function curPageURL() {
```

```
 $pageURL = 'http';
 if ($_SERVER["HTTPS"] == "on") {$pageURL .= "s";}
 $pageURL .= "://";
 if ($_SERVER["SERVER_PORT"] != "80") {
  $pageURL .=
$_SERVER["SERVER_NAME"].":".$_SERVER["SERVER_PORT"].$_SERVER["RE
QUEST_URI"];
 } else {
  $pageURL .= $_SERVER["SERVER_NAME"].$_SERVER["REQUEST_URI"];
 }
$pageURL = str_replace("sitemapper.php", "", $pageURL);
return $pageURL;
}

// main function that scans the directory tree for web pages
function crawldir($basedir){
   if ($handle = @opendir($basedir)) {
      while (false !== ($fn = readdir($handle))){
         if ($fn != '.' && $fn != '..'){ // ignore these
            $dir = $basedir."/".$fn;

            if (is_dir($dir)){
               crawldir($dir); // recursive call to this function
            } else { //only consider .html etc. files
               if (preg_match("/[^.\/].+\.(htm|html|php)$/",$dir,$fname)) {
                  printlink($fname[0]);
               }
                                                                  }
         }
      }
      closedir($handle);
   }

}
// function call
crawldir($basedir);
$xml_close .= "</urlset>\n";
echo $xml_close;
?>
```

App.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="FTPServerName" value="" />
    <add key="FTPServerUserID" value="" />
    <add key="FTPServerPassword" value="" />
    <add key="StartURL" value="" />
    <add key="LastRun" value="N/A"/>
    <add key="NoPages" value="1"/>
    <add key="PingGoogle" value="true"/>
    <add key="PingYahoo" value="true"/>

    <add key="PingAsk" value="true"/>
    <add key="Email" value="vamsikrishna726@gmail.com"/>
    <add key="ClientSettingsProvider.ServiceUri" value=""
/>

  </appSettings>
  <system.web>
    <membership
defaultProvider="ClientAuthenticationMembershipProvider">
      <providers>
        <add name="ClientAuthenticationMembershipProvider"
type="System.Web.ClientServices.Providers.ClientFormsAuthen
ticationMembershipProvider, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" serviceUri="" />
      </providers>
    </membership>
    <roleManager defaultProvider="ClientRoleProvider"
enabled="true">
      <providers>
        <add name="ClientRoleProvider"
type="System.Web.ClientServices.Providers.ClientRoleProvide
r, System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" serviceUri=""
cacheTimeout="86400" />
      </providers>
    </roleManager>
  </system.web>
</configuration>
```

AppconfigFileSettings.cs

```
using System;
using System.Collections.Generic;

using System.Text;
using System.Configuration;
using System.Xml;

namespace SitemapGen
{
    class AppConfigFileSettings
    {
        public void UpdateAppSettings(String KeyName,
String KeyValue)
        {
            XmlDocument XmlDoc = new XmlDocument();

XmlDoc.Load(AppDomain.CurrentDomain.SetupInformation.Config
urationFile);
            foreach (XmlElement xElement in
XmlDoc.DocumentElement)
            {
                if (xElement.Name == "appSettings")
                {
                    foreach (XmlNode xNode in
xElement.ChildNodes)
                    {
                        if (xNode.Attributes[0].Value ==
KeyName)
                        {

                            xNode.Attributes[1].Value =
KeyValue;
                        }
                    }
                }
            }

XmlDoc.Save(AppDomain.CurrentDomain.SetupInformation.Config
urationFile);
        }
        public string GetAppSettingItem(String KyeName)
        {
            XmlDocument XmlDoc = new XmlDocument();
```

```
XmlDoc.Load(AppDomain.CurrentDomain.SetupInformation.Config
urationFile);
            foreach (XmlElement xElement in
XmlDoc.DocumentElement)
            {
                if (xElement.Name == "appSettings")
                {
                    foreach (XmlNode xNode in
xElement.ChildNodes)
                    {
                        if (xNode.Attributes[0].Value ==
KyeName)
                        {

                            return
(xNode.Attributes[1].Value);
                        }
                    }
                }
            }
            return "";
        }
    }
}
```

Sitemapper.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.IO;
using System.Net;
using System.Collections;
using System.Threading;
using System.Net.Mail;
using System.Text.RegularExpressions;
using System.IO.Compression;

namespace SitemapGen
```

```
{
public partial class FormMain : Form
    {
Thread timerThread;
ArrayList time = new ArrayList();
    public FormMain()
        {
            InitializeComponent();
            scheduleSave.Enabled = false;
            cancelSchedule.Enabled = false;
            CheckForIllegalCrossThreadCalls = false;

        }



        private void btn_TestConnection_Click(object
sender, EventArgs e)
        {
            SitemapGen.FTPProcess ftpProcess = new
SitemapGen.FTPProcess(txt_ftpServer.Text, txt_ftpUser.Text,
txt_ftpPassword.Text);
            if (ftpProcess.TestConnection())
            {

                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                try
                {
                    WebRequest request =
HttpWebRequest.Create("http://www.queryready.com/sitemapper
.php");
                    request.Method = "HEAD"; // Just get
the document headers, not the data.
                    request.Credentials =
System.Net.CredentialCache.DefaultCredentials;
                    // This may throw a WebException:
                    using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
                    {
                        if (response.StatusCode ==
HttpStatusCode.OK)
                        {
                            MessageBox.Show("FTP Connect
Information is correct!", "FTP Status");
                        }
                    }
```

```
                }
                catch (WebException ex)
                {
                    // Cast the WebResponse so we can check
the StatusCode property
                    HttpWebResponse webResponse =
(HttpWebResponse)ex.Response;
                    // Determine the cause of the
exception, was it 404?
                    if (webResponse.StatusCode ==
HttpStatusCode.NotFound)
                    {
                    MessageBox.Show("Initial Setup
Sucessfull, Necessary Files Uploaded");
                    }
                        else
                    {
                    // Handle differently...
                    MessageBox.Show(ex.Message);
                    }

                }
            }
            else
            {
                MessageBox.Show("FTP Connect Information is
incorrect!.", "FTP Status");
            }
        }

        private void sitemapRun()
        {
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            XmlDocument xml1 = new XmlDocument();
            DataSet ds1 = new DataSet();
            ds1.ReadXml("http://"+
config.GetAppSettingItem("StartURL") + "/sitemapper.php");

ds1.WriteXml("c:\\SitemapGenerator\\Temp\\sitemap.xml");
            XmlDocument doc1 = new XmlDocument();

doc1.Load("c:\\SitemapGenerator\\Temp\\sitemap.xml");
            XmlNodeList nodeList1;
            XmlNode root1 = doc1.DocumentElement;
            nodeList1 = root1.SelectNodes("/urlset/url");
```

```
            foreach (XmlNode urlset1 in nodeList1)
            {
                DateTime d1 =
DateTime.Parse(urlset1.ChildNodes[1].InnerXml);
                DateTime d2 = DateTime.Today;

                TimeSpan diff;
                diff = d1 - d2;
                if (Math.Abs(diff.Days) > 365)
                {
                    urlset1.ChildNodes[2].InnerXml =
"Yearly";
                    urlset1.ChildNodes[3].InnerXml = "0.2";

                }

                if (Math.Abs(diff.Days) > 30 &&
Math.Abs(diff.Days) < 365)
                {
                    urlset1.ChildNodes[2].InnerXml =
"Monthly";
                    urlset1.ChildNodes[3].InnerXml = "0.5";
                }
                if (Math.Abs(diff.Days) > 7 &&
Math.Abs(diff.Days) < 30)
                {
                    urlset1.ChildNodes[2].InnerXml =
"Weekly";
                    urlset1.ChildNodes[3].InnerXml = "0.7";
                }
                if (Math.Abs(diff.Days) < 7)
                {
                    urlset1.ChildNodes[2].InnerXml =
"Daily";
                    urlset1.ChildNodes[3].InnerXml = "0.9";
                }

            }

            foreach (XmlNode urlset1 in nodeList1)
            {

resultDataGrid.Rows.Add(urlset1.ChildNodes[0].InnerXml,
urlset1.ChildNodes[1].InnerXml,
urlset1.ChildNodes[2].InnerXml,
urlset1.ChildNodes[3].InnerXml);
            }
```

```
            config.UpdateAppSettings("LastRun",
DateTime.Now.ToShortDateString());

            double changePer = (((ds1.Tables[0].Rows.Count
- Convert.ToDouble(config.GetAppSettingItem("NoPages")))/
ds1.Tables[0].Rows.Count)*100);
            label16.Text = changePer.ToString();
            config.UpdateAppSettings("NoPages",
ds1.Tables[0].Rows.Count.ToString());
            if (Math.Abs(changePer) > 30)
            {
                sendtheEmail();
            }
            label12.Text =
config.GetAppSettingItem("LastRun");
            label13.Text =
config.GetAppSettingItem("NoPages");
            notify();
        }

    private void notify()
    {
        try
        {
            String noteMsg = "Site map Generated and
Following Search Engines are Sucessfully Notified\n";
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            if (config.GetAppSettingItem("PingAsk") ==
"true")
            {
                WebRequest req1 =
HttpWebRequest.Create("http://submissions.ask.com/ping?site
map=http://" + config.GetAppSettingItem("StartURL") +
"/sitemap.xml");
                req1.GetResponse();
                noteMsg = noteMsg + "Ask.com\n";
            }
            if (config.GetAppSettingItem("PingGoogle") ==
"true")
            {
                WebRequest req2 =
HttpWebRequest.Create("http://www.google.com/webmasters/sit
emaps/ping?sitemap=http://" +
config.GetAppSettingItem("StartURL") + "/sitemap.xml");
```

```
                req2.GetResponse();
                noteMsg = noteMsg + "Google.com\n";
            }
            if (config.GetAppSettingItem("PingYahoo") ==
"true")
            {
                WebRequest req3 =
HttpWebRequest.Create("http://search.yahooapis.com/SiteExpl
orerService/V1/updateNotification?appid=YahooDemo&url=http:
//" + config.GetAppSettingItem("StartURL") +
"/sitemap.xml");
                //req3.GetResponse();
                noteMsg = noteMsg + "Yahoo.com\n";
            }
            MessageBox.Show(noteMsg, "Ping Notification",
MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString(), "Ping Error");
        }
    }

        private void sendtheEmail()
        {
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            string emailid =
config.GetAppSettingItem("Email");

            SendEmail("SiteMapGenerator",
"tipsytoff@gmail.com", "splash32", "Customer", emailid,
"Sitemap Generator Alert", "Hello Customer, The Schedule
you have selected seems to be out of range with respective
to your site page changes. We recommend you to change the
schedule. Thank you");
        }
        private static void SendEmail(string yourName,
string yourGmailUserName, string yourGmailPassword, string
toName, string toEmail, string subject, string body)
        {
            MailAddress from = new
MailAddress(yourGmailUserName, yourName);
            MailAddress to = new MailAddress(toEmail,
toName);
```

```
            MailMessage message = new MailMessage(from,
to);
            message.Subject = subject;
            message.Body = body;
            SmtpClient smtpClient = new
SmtpClient("smtp.gmail.com", 587);//Edit this, if you like
to use another
            NetworkCredential nc = new
NetworkCredential(yourGmailUserName, yourGmailPassword);
            smtpClient.EnableSsl = true;
            smtpClient.UseDefaultCredentials = false;
            smtpClient.Credentials = nc;
            smtpClient.Send(message);
        }


        private void FormMain_Resize(object sender,
EventArgs e)
        {
            if (FormWindowState.Minimized == WindowState)
                Hide();
        }

        private void notifyIcon1_MouseDoubleClick(object
sender, MouseEventArgs e)
        {
           Show();
           WindowState = FormWindowState.Normal;

        }

        private void numericUpDown1_ValueChanged(object
sender, EventArgs e)
        {
            scheduleSave.Enabled = true;
            scheduleGrid.Visible = true;
            scheduleGrid.Rows.Clear();

            DateTime dt = new DateTime();
            dt = DateTime.Now;


            for (int i = 0; i <= 14; i++)
            {
                DateTime date =
dt.AddDays(Convert.ToDouble(numericUpDown1.Value*i));
```

```
scheduleGrid.Rows.Add(i,date.ToShortDateString() );
            }
        }

        private void btn_GenerateSiteMap_Click(object
sender, EventArgs e)
        {
            AppConfigFileSettings config = new
AppConfigFileSettings();

            panel2.Visible = false;
            scheduleGrid.Visible = false;
            sitemapRun();
            SitemapGen.FTPProcess ftp = new
SitemapGen.FTPProcess();

ftp.uploadFile(config.GetAppSettingItem("FTPServerName"),
"C:\\SitemapGenerator\\Temp\\sitemap.xml",
config.GetAppSettingItem("FTPServerUserID"),
config.GetAppSettingItem("FTPServerPassword"));

CompressFile("C:\\SitemapGenerator\\Temp\\sitemap.xml");

ftp.uploadFile(config.GetAppSettingItem("FTPServerName"),
"C:\\SitemapGenerator\\Temp\\sitemap.xml.gz",
config.GetAppSettingItem("FTPServerUserID"),
config.GetAppSettingItem("FTPServerPassword"));
        }

        private void toolStripMenuItem1_Click(object
sender, EventArgs e)
        {
        this.Show();
        this.WindowState = FormWindowState.Normal;
        this.ShowInTaskbar = true;

        }

        private void toolStripMenuItem2_Click(object
sender, EventArgs e)
        {
            this.Close();
        }

        private void toolStripMenuItem3_Click(object
sender, EventArgs e)
```

```
        {
            this.WindowState = FormWindowState.Minimized;
            this.ShowInTaskbar = true;
        }

        private void FormMain_Load(object sender, EventArgs
e)
        {

            panel2.Visible = false;
            scheduleGrid.Visible = false;
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            label12.Text =
config.GetAppSettingItem("LastRun");
            label13.Text =
config.GetAppSettingItem("NoPages");


        }

        private void tabControl1_KeyDown(object sender,
KeyEventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs
e)
        {
            panel2.Visible = true;
        }

        private void button4_Click(object sender, EventArgs
e)
        {
            //Weather to Ping Google
            if (checkBox1.Checked == false)
            {
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingGoogle",
"false");
            }
            else
            {
```

```
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingGoogle",
"true");
            }

            //Weather to Ping Yahoo
            if (checkBox1.Checked == false)
            {
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingYahoo",
"false");
            }
            else
            {
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingYahoo",
"true");
            }

            //Weather to Ping Ask
            if (checkBox1.Checked == false)
            {
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingAsk",
"false");
            }
            else
            {
                SitemapGen.AppConfigFileSettings config =
new SitemapGen.AppConfigFileSettings();
                config.UpdateAppSettings("PingAsk",
"true");
            }
        }


    private void startSchedule()
    {
        while (true)
        {
            DateTime today = new DateTime();
            today = DateTime.Now;
            foreach (Object timevalue in time)
```

```
            {
                if(today.ToShortDateString()==
timevalue.ToString())
                {

MessageBox.Show(today.ToShortDateString(), "Time Found");
                    sitemapRun();
                    refreshGrid();
                }

            }
            time.Remove(today.ToShortDateString());
        }
    }
    public void refreshGrid()
    {

        scheduleGrid.Rows.RemoveAt(0);
        scheduleGrid.Refresh();
    }

        private void button1_Click(object sender, EventArgs
e)
        {

            for (int j = 0; j < 15; j++)
            {

time.Add(scheduleGrid.Rows[j].Cells[1].Value);
            }
            MessageBox.Show("Schedule Saved", "Schedule
Saved Sucessfully");
            timerThread = new Thread(new
ThreadStart(startSchedule));
            timerThread.Start();
            cancelSchedule.Enabled = true;
            scheduleSave.Enabled = false;
        }


        private void cancelSchedule_Click(object sender,
EventArgs e)
        {
            cancelSchedule.Enabled = false;
            scheduleGrid.Visible = false;
            panel2.Hide();
            timerThread.Abort();
```

```
            MessageBox.Show("Thread Suspended", "");
        }

    private void btn_Save_Click(object sender, EventArgs e)
    {
        validateValues();
        SitemapGen.FTPProcess ftp = new FTPProcess();
        ftp.uploadFile(txt_ftpServer.Text,
System.Windows.Forms.Application.StartupPath +
"\\sitemapper.php", txt_ftpUser.Text,
txt_ftpPassword.Text);
        StreamWriter streamWriter;
        streamWriter =
File.CreateText("c:\\SitemapGenerator\\Temp\\robots.txt");
        streamWriter.WriteLine("User-agent: *\nDisallow:
/wp-admin/");
        streamWriter.Close();
    }
    public static void CompressFile(string path)
    {
        FileStream sourceFile = File.OpenRead(path);
        FileStream destinationFile = File.Create(path +
".gz");

        byte[] buffer = new byte[sourceFile.Length];
        sourceFile.Read(buffer, 0, buffer.Length);

        using (GZipStream output = new
GZipStream(destinationFile,
            CompressionMode.Compress))
        {
            Console.WriteLine("Compressing {0} to {1}.",
sourceFile.Name,
                destinationFile.Name, false);

            output.Write(buffer, 0, buffer.Length);
        }

        // Close the files.
        sourceFile.Close();
        destinationFile.Close();
    }
    public void createDirectory()
    {

        if
(!System.IO.Directory.Exists("c:\\SitemapGenerator\\Temp"))
```

```
        {
            string currentDir = @"c:\SitemapGenerator";
            //Create a new subfolder under the current
active folder
            string newPath =
System.IO.Path.Combine(currentDir, "Temp");
            // Create the subfolder
            System.IO.Directory.CreateDirectory(newPath);
        }
    }
    public void validateValues()
    {
        String valid = "true";
        String errMsg = "";

        if (txt_startURL.Text == "")
        {
            valid = "false";
            errMsg = errMsg + "Blank Start URL Field\n";
        }

        if (txt_ftpServer.Text == "")
        {
            valid = "false";
            errMsg = errMsg + "Blank Server Field\n";
        }
        if (txt_ftpUser.Text == "")
        {
            valid = "false";
            errMsg = errMsg + "Blank User Name Field\n";
        }
        if (txt_ftpPassword.Text == "")
        {
            valid = "false";
            errMsg = errMsg + "Blank Password Field\n";
        }
        if (txt_email.Text == "")
        {
            valid = "false";
            errMsg = errMsg + "Blank Email Field\n";
        }
        else
        {
            if (!(isEmail(txt_email.Text)))
            {
                valid = "false";
```

```
                errMsg = errMsg + "Oops! Invalid Email
Address! Please enter a valid one!\n";
            }
        }
        if (valid == "true")
        {
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            config.UpdateAppSettings("StartURL",
txt_startURL.Text);
            config.UpdateAppSettings("FTPServerName",
txt_ftpServer.Text);
            config.UpdateAppSettings("FTPServerUserID",
txt_ftpUser.Text);
            config.UpdateAppSettings("FTPServerPassword",
txt_ftpPassword.Text);
            config.UpdateAppSettings("Email",
txt_email.Text);
            createDirectory();
            MessageBox.Show("FTP Settings Saved\n" +
"Initial Setup Successfull\nFiles Uploaded", "FTP Settings
Saved", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show(errMsg,"Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    public static bool isEmail(string inputEmail)
    {
        string strRegex = @"^([a-zA-Z0-9_\-\.]+)@((\[[0-
9]{1,3}" +
              @"\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-
]+\" +
              @".)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";
        Regex re = new Regex(strRegex);
        if (re.IsMatch(inputEmail))
            return (true);
        else
            return (false);
    }

    private void txt_ftpServer_Leave(object sender,
EventArgs e)
    {
        if (!txt_ftpServer.Text.StartsWith("ftp://"))
```

```
            txt_ftpServer.Text = "ftp://" +
txt_ftpServer.Text;
    }




    }
}
```

FTPProcess.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;


namespace SitemapGen
{
    class FTPProcess
    {
        string ftpServerIP = "";
        string ftpUserID = "";
        string ftpPassword = "";

        public FTPProcess()
        {
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            ftpServerIP =
config.GetAppSettingItem("FTPServerName");
            ftpUserID =
config.GetAppSettingItem("FTPServerUserID");
            ftpPassword =
config.GetAppSettingItem("FTPServerPassword");
        }

        public FTPProcess(string server, string userid,
string password)
```

```
        {
            SitemapGen.AppConfigFileSettings config = new
SitemapGen.AppConfigFileSettings();
            ftpServerIP = server;
            ftpUserID = userid;
            ftpPassword = password;
        }

        //public void Upload(string filename)
        //{
        //    FileInfo fileInf = new FileInfo(filename);
        //    string uri = "ftp://" + ftpServerIP + "/" +
fileInf.Name;
        //    FtpWebRequest reqFTP;

        //    // Create FtpWebRequest object from the Uri
provided
        //    reqFTP =
(FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" +
ftpServerIP + "/" + fileInf.Name));

        //    // Provide the WebPermission Credintials
        //    reqFTP.Credentials = new
NetworkCredential(ftpUserID, ftpPassword);

        //    // By default KeepAlive is true, where the
control connection is not closed
        //    // after a command is executed.
        //    reqFTP.KeepAlive = false;

        //    // Specify the command to be executed.
        //    reqFTP.Method =
WebRequestMethods.Ftp.UploadFile;

        //    // Specify the data transfer type.
        //    reqFTP.UseBinary = true;

        //    // Notify the server about the size of the
uploaded file
        //    reqFTP.ContentLength = fileInf.Length;

        //    // The buffer size is set to 2kb
        //    int buffLength = 2048;
        //    byte[] buff = new byte[buffLength];
        //    int contentLen;
```

```
        //    // Opens a file stream (System.IO.FileStream)
to read the file to be uploaded
        //    FileStream fs = fileInf.OpenRead();

        //    try
        //    {
        //        // Stream to which the file to be upload
is written
        //        Stream strm = reqFTP.GetRequestStream();

        //        // Read from the file stream 2kb at a
time
        //        contentLen = fs.Read(buff, 0,
buffLength);

        //        // Till Stream content ends
        //        while (contentLen != 0)
        //        {
        //            // Write Content from the file stream
to the FTP Upload Stream
        //            strm.Write(buff, 0, contentLen);
        //            contentLen = fs.Read(buff, 0,
buffLength);
        //        }

        //        // Close the file stream and the Request
Stream
        //        strm.Close();
        //        fs.Close();
        //        MessageBox.Show("Your file is uploaded
successfully!");
        //    }
        //    catch (Exception ex)
        //    {
        //        MessageBox.Show(ex.Message, "Upload
Error");
        //    }
        //}

        public void uploadFile(string FTPAddress, string
filePath, string username, string password)
        {
            //Create FTP request
            FtpWebRequest request =
(FtpWebRequest)FtpWebRequest.Create(FTPAddress + "/" +
Path.GetFileName(filePath));
```

```
            request.Method =
WebRequestMethods.Ftp.UploadFile;
            request.Credentials = new
NetworkCredential(username, password);
            request.UsePassive = true;
            request.UseBinary = true;
            request.KeepAlive = false;

            //Load the file
            FileStream stream = File.OpenRead(filePath);
            byte[] buffer = new byte[stream.Length];

            stream.Read(buffer, 0, buffer.Length);
            stream.Close();

            //Upload file
            Stream reqStream = request.GetRequestStream();
            reqStream.Write(buffer, 0, buffer.Length);
            reqStream.Close();

        }

        public bool TestConnection()
        {
            StringBuilder result = new StringBuilder();
            FtpWebRequest reqFTP;

            try
            {
                //reqFTP =
(FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" +
ftpServerIP + "/"));
                reqFTP =
(FtpWebRequest)FtpWebRequest.Create(new Uri(ftpServerIP +
"/"));
                reqFTP.UseBinary = true;
                reqFTP.Credentials = new
NetworkCredential(ftpUserID, ftpPassword);
                reqFTP.Method =
WebRequestMethods.Ftp.ListDirectory;
                WebResponse response =
reqFTP.GetResponse();
                StreamReader reader = new
StreamReader(response.GetResponseStream());
                //MessageBox.Show(reader.ReadToEnd());
                string line = reader.ReadLine();
                while (line != null)
```

```
                {
                    result.Append(line);
                    result.Append("\n");
                    line = reader.ReadLine();
                }

result.Remove(result.ToString().LastIndexOf('\n'), 1);
                reader.Close();
                response.Close();

//MessageBox.Show(response.StatusDescription);
                return true;
            }
            catch (Exception ex)
            {
                return false;
            }
        }

    }

}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace SitemapGen
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FormMain());

        }
    }
```