



Automatic Website Content Change Detection and Notification Using Image Processing

Bhoomin Tanut, Chitthiwat Ratchathian, Natthawut Matkang, and Surin Petchthai^{1*}

¹ Department of Computer Science and Technology, Faculty of Science and Technology, Kamphaeng Phet Rajabhat University, Kamphaeng Phet 65000, Thailand

Abstract

This research develops a web application called WatchNSend that allows users to automatically monitor websites and detects changes in website content using image processing techniques. The application is built in three modules. First, in the Setup Module, the user creates a monitoring job and specifies how they would like their website monitored. An Area of Interest (AOI) is selected by the user within the displayed webpage, and other job details are also set, such as the desired monitoring frequency. The application then saves the AOI image and job details for later use. The second module is the job module. This is where all job specifications and all job update information are stored. Also in this module, a timer controls mechanisms that monitor the specified update intervals of all jobs and initiate the third module accordingly. The third module is the comparison module. Here the application automatically collects a current copy of the job's AOI and then compares this new version of the AOI with the previous version, using two methods called the edge calculation and the overlay calculation. If content changes inside the AOI have reached a threshold set by the user, the application automatically notifies the user via e-mail. Results from multiple evaluations show that WatchNSend can monitor websites, detect and analyze changes, and notify the webmaster of the changes accurately and efficiently. WatchNSend is a reliable, robust, and easy-to-use tool that can save users time while keeping them current on website changes.

Keywords: Automatic Website Monitoring, Digital Image Processing, Image-based Monitoring, Change Detection

Article history: Received 20 January 2024, Revised 15 August 2024, Accepted 01 October 2024

1. Introduction

Websites are indispensable tools that both the government and private sectors rely on for collecting, processing, and distributing their organization's information [1]. Two basic types of webpages are static webpages and dynamic webpages [2]. Static webpages remain as they are. They do not update automatically or in response to user actions. Static webpages are built with HTML and CSS and reside on a server. When a static webpage is requested by a user, the HTML file is sent by the server to their browser, which opens the page for viewing. In contrast, a dynamic webpage is developed not

only with HTML but also with one or more programming languages such as PHP, C#, JavaScript, Ruby, or Python, and a dynamic webpage can update automatically in response to input from users or other data. A user can personalize a dynamic webpage, which can include interactive components such as a shopping cart or a form to fill out. Data for a dynamic webpage is stored in a database on the server and can be requested at any time, enabling the webpage to update dynamically according to user input or changes in the underlying data.

*Corresponding author; e-mail: surin_p@kpru.ac.th

Website content can change at any time, and many sorts of professions and fields need to be able to monitor various websites for changes. Timely knowledge of new updates to websites can be important or even critical [3]. In order to monitor one or more websites conveniently, users or organizations can avail of software that automatically performs Change Detection and Notification (CDN). As the name suggests, a CDN system automatically detects changes in webpages and notifies the user when there is a change [3]. Previous research in the field of website content change monitoring can be divided into two groups. The first group of CDN systems detects changes to text and HTML element tags. The second group of CDN systems detects changes by using image processing. An example of the first type of CDN system is when M. C. Shobhna [4] conducted exploratory research on webpage change detection systems between 2004 and 2013 using various methods such as BIODIFF, XML Tree Diff, CH Diff, and CX Diff, Level Order Transversal, Optimized Hungarian, Hashing Based Algorithm, Node Signature Comparison, Tree Traversing, and Document Tree Based Approach. Soobbha and Manoj found that a document tree-based approach gave the best performance. Their method relies on converting the content on a webpage to an XML structure. The XML structure generates nodes from the opening and closing tags of the HTML statements. When the system wants to

monitor the website, it compares the parameters for each element. Regarding the second group of CDN systems, those that detect changes by using image processing, a review of current CDN systems by Mallawaarachchi, V. et al. [3] highlighted two examples from this group that are easy to use and flexible to the needs of users: Wachete [5] and VisualPing. Both systems offer a variety of monitoring options, including the ability to monitor multiple webpages, to monitor various types of content, to set a threshold percentage of changes that triggers notification, to adjust the frequency of monitoring, and to decide the frequency of notifications.

E. Fenton reported on the top 5 most popular Change Detection and Notification (CDN) tools for 2023 [6]: Visualping.io [7], Fluxguard [8], Sken [9], Pagescreen [10], and OnWebChange [11]. Fees charged for these services vary, depending on the number of monthly checks desired and other specific features. Limited free trials are also offered. The systems mentioned above are all provided from outside Thailand. In Thailand, there is currently no domestic developer or provider of this service. M. C. Shobhna [4] compared the specific features and detection modes of currently available CDN tools, including those mentioned by E. Fenton [6], and the specific features and detection modes of those five services are shown in Table 1.

Table 1. Comparison of functionality and detection modes of some popular CDN tools

CDN Tool	Functionality					Detection Mode		
	MSP	MMP	SSD	FICS	Notify	Visual	Text	HTML
Visualping.io [7]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Fluxguard [8]	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Sken [9]	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Pagescreen [10]	Yes	Yes	Yes	Yes	Yes	Yes	No	No
OnWebChange [11]	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

SSD=Server side detection, MMP=Monitor multiple pages, MSP= Monitor a single page, FICS = Fixed Interval checks, and Notify = Email notification.

As of the end of March 2023, there were 354.0 million domain name registrations across all top-level domains (TLDs) [12], and the use of dynamic websites has only increased, including frequently updated websites such as news websites, online shopping websites, travel and accommodation websites, Wikipedia pages, and social media. When these webpages update information, users may want to be informed about changes to webpages immediately.

The current study develops a web application called WatchNSend that uses image processing techniques to detect changes in the content of a website and then sends notification of the changes via e-mail to the webmaster in order to reduce time spent monitoring the website in person. In the process, this study will disseminate the techniques involved in creating such an image-based system for those who want to develop website change detection and change

notification. This research can also serve as a guideline for the development of other sorts of related systems, such as GPS location monitoring systems using online maps, online appointment calendar change monitoring systems, and online retail product promotion change monitoring systems.

2. Materials and methods

The WatchNSend web application developed here for automatic website change detection and notification utilizes a variety of image processing techniques. The application is written mainly in PHP [13], with some JavaScript and HTML as well. There is a client side and a server side. Figure 1 shows an overview of the processes that happen on each side. The three modules shown (Setup, Job, and Comparison) are described in detail in Sections 2.1–2.3.

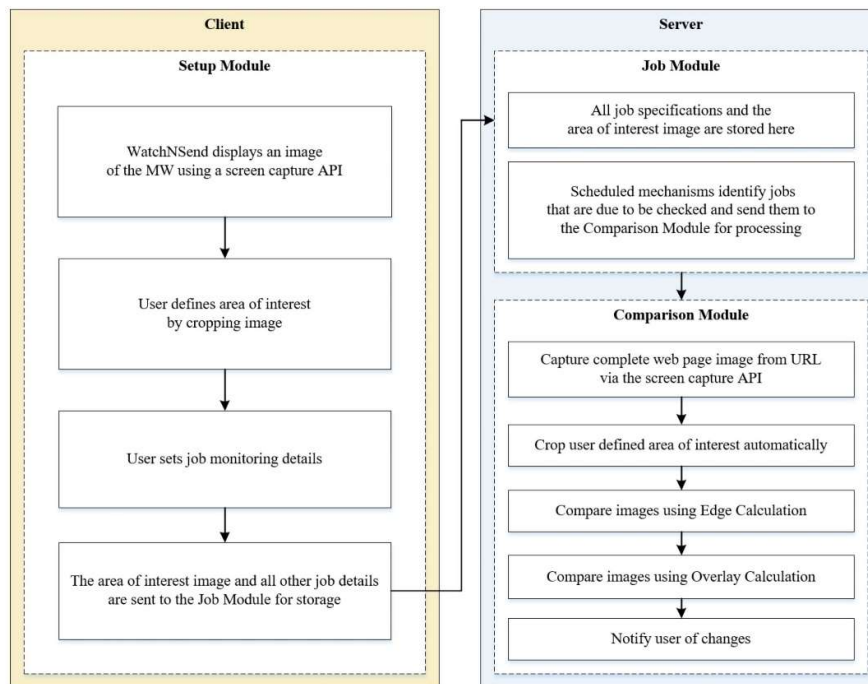


Figure 1. This overview shows the main processes inside the WatchNSend web application. MW=monitored webpage, i.e., the webpage that the user wishes to monitor.

2.1 Setup Module

The Setup Module contains the steps for receiving job details from the user on the client computer. Each monitored webpage (MW) or portion thereof is one job. Users can add, edit, and check jobs in their job list using the WatchNSend website. Here are the steps.

2.1.1 WatchNSend displays an image of the MW using a screen capture API (Application Programming Interface).

The user is prompted to enter the URL of the MW. WatchNSend then uses the specified URL to go capture a current image of the MW. (A description of how this image is captured using a screen capture API appears in Section 2.3.1.)

2.1.2 The user defines the area of interest by cropping the MW image.

When the MW image appears in the display area, an adjustable rectangular selection tool will also be automatically activated for the user to indicate the Area of Interest (AOI). The AOI is the specific part of the image that the user would like to monitor. It can be designated to be anything from a small portion of the full image to the full image itself. The size and position of the selection tool are easy to adjust. The coordinates of the AOI chosen by the user are captured using JavaScript's Cropper command [14] and then saved as part of the data for the current job.

2.1.3 User sets job monitoring details

After selecting the AOI, the user specifies the frequency to check for changes, the threshold of change within the AOI at which the user would like to receive notification, and the e-mail address to which notifications should be sent. This threshold set by the user is referred to in this study as the notification threshold. The frequency to check for changes can be set to every 5, 15, or 30 minutes, daily, weekly, or monthly [15]. If the frequency is daily, the user can select the hour that they would like to check for changes. If the frequency is weekly, the user can select the day and hour to check. If the frequency is monthly, the user can select the date and hour to check.

2.1.4 Store the AOI image and all job details

Once the job monitoring details are complete, WatchNSend will immediately save a current copy of the AOI image as well as all the job details. This information will be used later to schedule checks for MW changes and to assess those changes.

2.2 Job Module

The Job Module resides on the server, and it has two functions: job information storage and identification of jobs that are due to be checked so that they can be sent to the Comparison Module for processing. Each of these functions is explained in detail here.

2.2.1 Job information storage

The Job Module is the storage area for all data on jobs, including all the job specifications that were inputted during the Setup Module, the original AOI image, and all information collected each time the MW is checked for changes. The first fetching of the AOI is initiated in the Setup Module when the new job is created, and all MW checks that follow are initiated by the timer function here in the Job Module.

2.2.2 Identifying jobs that are due to be checked and sending them to the Comparison Module

A system of timed mechanisms identifies jobs that are due to be checked. There are seven of these mechanisms, one for each possible frequency that users can specify for a job to be checked. Because these seven mechanisms are dispatched on a schedule to perform a designated task, they are called "JobFlights" here, like flights being dispatched from an airport. The seven JobFlights used in this study are shown in Figure 2. The 24-hour schedule by which the JobFlights depart, i.e., are run, is shown in Figure 3.

The Seven JobFlights in WatchNSend
JobFlight05 leaves every five minutes to query jobs that are checked every 5 minutes
JobFlight15 leaves every fifteen minutes to query jobs that are checked every 15 minutes.
JobFlight30 leaves every thirty minutes to query jobs that are checked every 30 minutes.
JobFlight60 leaves every sixty minutes to query jobs that are checked every 60 minutes.
JobFlightDay leaves every sixty minutes to query jobs that are checked daily.
JobFlightWeek leaves every sixty minutes to query jobs that are checked weekly.
JobFlightMonth leaves every sixty minutes to query jobs that are checked monthly.

Figure 2. These are the seven mechanisms called JobFlights that identify jobs due to be checked. The last three job flights in the list leave every hour in order to check for and respond to the user's preferred hour for checking changes.

24 Hour Schedule	JobFlight 05	JobFlight 15	JobFlight 30	JobFlight 60	JobFlight Day	JobFlight Week	JobFlight Month
00:00	Run #1	Run #1	Run #1	Run #1	Run #1	Run #1	Run #1
00:05	Run #2						
00:10	Run #3						
00:15	Run #4	Run #2					
00:20	Run #5						
00:25	Run #6						
00:30	Run #7	Run #3	Run #2				
00:35	Run #8						
00:40	Run #9						
00:45	Run #10	Run #4					
00:50	Run #11						
00:55	Run #12						
01:00	Run #13	Run #5	Run #3	Run #2	Run #2	Run #2	Run #2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
23:00	Run #276	Run #93	Run #47	Run #24	Run #24	Run #24	Run #24
23:05	Run #278						
23:10	Run #279						
23:15	Run #280	Run #94					
23:20	Run #281						
23:25	Run #282						
23:30	Run #283	Run #95	Run #48				
23:35	Run #284						
23:40	Run #285						
23:45	Run #286	Run #96					
23:50	Run #287						
23:55	Run #288						

Figure 3. This is the 24-hour schedule by which the JobFlights depart, i.e. are run.

Operation of the JobFlights is managed by two things on the server: the Task Scheduler [16], which is part of Windows, and a set of seven PHP code scripts, one for each JobFlight type. The task scheduler is able to run the code scripts automatically in the background without opening a browser or any other program.

The seven JobFlight scripts vary by nature, with the most complex being the script for JobFlightMonth, because the user can select the monthly date and hour to check the job.

JobFlightMonth's actions will be explained here as an example, but the other six JobFlights operate similarly with their respective selection criteria. As mentioned above, JobFlightMonth leaves once an hour. Here is what happens. Every hour on the hour, the Task Scheduler automatically runs the PHP script specifically for JobFlightMonth. The script performs two important actions. The first action is to build a recordset of all job records that meet the criteria shown in Figure 4.

Recordset Criteria of JobFlightMonth
The job must be currently active.
The job must be specified to check monthly.
The job must be specified to check on the current date of every month.
The job must be specified to check on the current hour.

Figure 4. JobFlightMonth will pick up all records that meet these criteria.

Any records that enter the recordset defined in Figure 4 are due to be checked now. The second action of the JobFlight script is to take any records that have just been collected in the recordset and process them in the Comparison Module shown in Figure 1. The code that carries out the steps of the Comparison Module is here in the JobFlight script. If the JobFlight did not encounter any records to collect, the Comparison Module steps are not called.

2.3 Comparison Module

The job records delivered by each JobFlight need to have the current version of their monitored webpage (MW) captured and compared to the previously captured version. This happens in the Comparison Module, and the details of each step are described here.

2.3.1 Capture complete current webpage image from URL via screen capture API

This procedure relates to both capturing the initial webpage image of the MW during job initiation in the Setup Module and to capturing the current webpage image of the MW when a JobFlight has determined that the job is due to be checked for changes. WatchNSend utilizes a screen capture API to collect a current, complete screenshot of the user's monitored webpage. The underlying reason for using a screen capture API is that the application needs more than just a normal display of the MW as it would usually appear on a monitor. Many webpages are longer than the monitor display and use scrollbars to make the complete webpage accessible. An image of the full webpage that is complete and without any scrollbars will be referred to here as a Complete Webpage Image (CWI). Capturing a CWI is not a straightforward programming task because it requires installation of Selenium

software [17], which is also not a minor task. Various providers simplify this process by offering screen capture API services that collect and deliver CWIs for a small fee. Each API provider has different capabilities and different pricing. Three important performance criteria when selecting a screen capture API provider for an application like WNS are: the completeness of the webpage delivered, the accuracy of the webpage delivered, and the speed of delivery. Since WatchNSend uses PHP as its primary language for development, it also requires an API that supports PHP. Requests to the screen capture API are made through the GET or POST methods of PHP.

2.3.2 Crop user-defined area of interest automatically

When an updated, complete webpage image is acquired from the screen capture API, it is cropped automatically to the AOI using the stored coordinates of the AOI from the job record. JavaScript's Cropper command is used for this purpose as before. The AOI cropped from a current version of the web page is referred to as the Current Image (CI), while the previously stored AOI is referred to as the Previous Image (PI).

2.3.3 Perform the edge calculation and the overlay calculation on the current image and previous image.

Whenever a webpage is checked for updates, both the CI and the PI are sent to feature extraction algorithm 1 to perform the edge calculation and the overlay calculation. Each of these two calculations compares the CI with the PI and provides a percent change between the two images. The two calculations complement each other because the edge calculation is best at detecting actual text

changes, while the overlay calculation is best at detecting changes in presentation of the text and changes in color. Because these two calculations are used at the same time and

because they share some variables, they appear together in Algorithm 1. The steps in Algorithm 1 are described in detail here.

Step 1: Convert the CI and the PI to grayscale. These will be called the Grayscale Current Image (GCI) and Grayscale Previous Image (GPI).

Step 2: Find all the edge pixels of the GCI and GPI using the Canny Edge Detection algorithm [18]. These will be called the Edge (pixels of the) CI and PI (ECI and EPI)

Step 3: Count the number of edge pixels in the ECI and EPI. These will be called the NECI and NEPI.

Step 4: Perform the Edge Calculation using Equation 1 [19].

$$\text{Edge Calculation result} = \frac{|\text{NEPI} - \text{NECI}|}{|\text{NEPI}|} \times 100 \quad (1)$$

Step 5: The Overlay Grayscale Image (OGI) is a construct that reflects changes between the GCI and the GPI. Create the OGI by calculating the OGI value of every pixel using Equation 2 [20].

$$\text{OGI}_{(x,y)} = (\text{abs}(\text{GCI}_{(x,y)} - \text{GPI}_{(x,y)}) + \text{abs}(\text{GPI}_{(x,y)} - \text{GCI}_{(x,y)})) \% 255 \quad (2)$$

If the intensity value of a given pixel has changed between the GPI and the GCI, the intensity of that pixel in the OGI will be greater than 0. If not, it will be 0.

Step 6: Use Otsu's thresholding method [21] to calculate the Otsu threshold value of the OGI. The result will be between 0 and 1, which corresponds to the grayscale pixel value range between 0 and 255.

Step 7: Convert the OGI image to a binary image (BOGI) using the Otsu threshold value calculated in the previous step.

Step 8: Count the number of white pixels in the BOGI (WBOGI).

Step 9: Perform the Overlay Calculation using Equation 3.

$$\text{Overlay Calculation result} = \frac{\text{WBOGI}}{\text{M} \times \text{N}} \times 100 \quad (3)$$

Where: M is the Area Of Interest Image height

N is the Area Of Interest Image width

2.3.4 Select the calculation results

After the Edge Calculation and Overlay Calculation have been performed, their results, which both estimate the Percent Change (PC) between the CI and the PI, are compared, and the higher of the two numbers [22] is selected

to represent the PC for this MW check and referred to as the MWPC.

2.3.5 Notify user of changes

Once the MWPC is obtained, its value is compared with the Notification Threshold

originally specified by the user in the job record. If the MWPC is greater than or equal to the Notification Threshold, an email is sent to the user to notify them of changes. Whether or not the Notification Threshold is met, the following items are stored in the job history record after each MW check: the time of the MW check, the CI, the Edge Calculation result, the Overlay Calculation result, and the MWPC.

2.4 Experiments

2.4.1 Screenshot API service provider comparison test

The purpose of this test is to determine the best-performing screenshot API service from among those that are available at any given time, based on the three previously mentioned performance criteria: the completeness of the webpage delivered, the accuracy of the webpage delivered, and the speed of delivery. In addition to these three performance criteria, the Screenshot API service needs to be able to handle two basic kinds of webpage banners [23]: scrolling banners (simple banners that move with the rest of the webpage and can scroll out of sight) and static banners (banners that are fixed in position and do not move in response to scrolling). Six different screenshot API providers were tested, and a total of ten URLs were used for testing. Five of them have scrolling banners: <https://www.wikipedia.org/>, <https://www.kpru.ac.th/index.php>, <http://omayo.blogspot.com/>, https://www.diald.nu.ac.th/th/clinic_calendar.php, <https://store.steampowered.com/?l=thai>, The other five have static scrolling banners: <https://stackoverflow.com/>, <https://www.w3schools.com/>, <https://www.lazada.co.th/shop/skechers>, <https://shopee.co.th/m/super-brand-day>, and <https://www.deviantart.com>. The test was run on an internet connection measured with Google Fiber [24] as having a download speed of 216.8 Mbps and an upload speed of 53.2 Mbps. The speed of actual image captures during the test was measured using PHP commands to record the time before and after the image capture. Test scores for the three

performance criteria were normalized to values in the range of 0- 100 using the Min- Max Normalization Method [25] shown in Equation 4.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} * 100 \quad (4)$$

In this normalization method, the highest score received during the test becomes 100 and the lowest score received becomes 0. All other scores fall between this minimum and maximum, and the proportions between the original scores are maintained. Results of the three performance criteria are normalized in this way so that they share a common unit and can be conveniently compared.

2.4.2 The AOI processing time test

The purpose of this test is to time how long the application takes to crop an incoming Screenshot API image down to the AOI and then perform both the Edge Calculation and Overlay Calculation on the AOI. To run the test, a timer is started in the application code when the MW image is received, and the timer is stopped right after the results of the two calculations are complete. This test used 70 sample images of the same starting size and cropped them in 7 different patterns (representing different size AOIs). Each cropping pattern was applied to 10 images.

2.4.3 The comparison module test

The purpose of this experiment is to test the functionality of the comparison module and the ability of the Edge Calculation and Overlay Calculation to accurately detect content changes in the AOI. The experiment consists of 4 tests, each with an original image and four progressively changing update images, as shown in Figure 5. Each update image was compared to the original image of its test. In test 1, the amount of text changes. In test 2, the background color changes. In test 3, the text color changes. In test 4, the sequence of the text changes.

Test	Original	Update Images			
1	WatchNSend WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend	WatchNSend	
	(A)	(B)	(C)	(D)	(E)
2					
	(F)	(G)	(H)	(I)	(J)
3	WatchNSend WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend WatchNSend WatchNSend	WatchNSend WatchNSend WatchNSend WatchNSend
	(K)	(L)	(M)	(N)	(O)
4	WatchNSend WatchNSend WatchNSend WatchNSend	dneSNhctaW WatchNSend WatchNSend WatchNSend	dneSNhctaW dneSNhctaW WatchNSend WatchNSend	dneSNhctaW dneSNhctaW dneSNhctaW WatchNSend	dneSNhctaW dneSNhctaW dneSNhctaW dneSNhctaW
	(P)	(Q)	(R)	(S)	(T)

Figure 5. The image dataset used for the four tests that evaluate the functionality of the Comparison Module. In test 1, the amount of text changes. In test 2, the background color changes. In test 3, the text color changes. In test 4, the sequence of the text changes.

3. Results

3.1 Screenshot API service provider comparison test

The results of this test are divided into two figures. Figure 6 shows the results from scrolling banner webpages, and Figure 7 shows the results from static banner webpages.

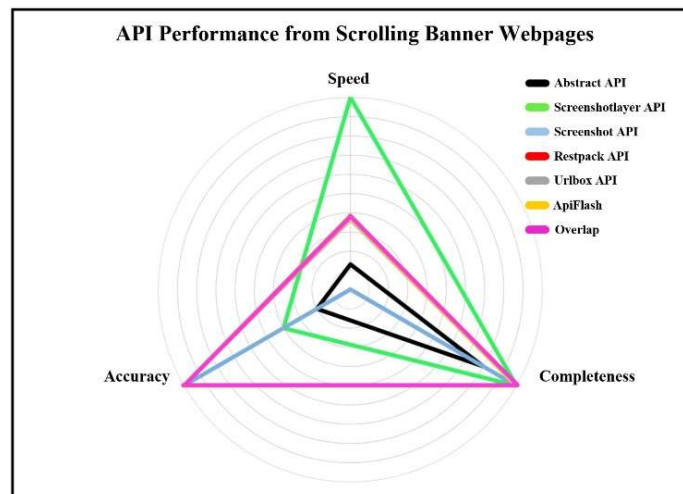


Figure 6. Results of the API performance test using scrolling banner webpages. Scores for the three criteria range from 0% (center point) to 100% (outermost ring). The last three providers in the list (Restpack, Urlbox, and ApiFlash) fully overlap as the pink triangle.

Figure 6 shows the three test criteria results (speed, accuracy, and completeness) displayed visually as a triangle for each provider. In the completeness category, Screenshotlayer, Screenshot API, Restpack, Urlbox, and ApiFlash all scored 100%, followed by Abstract API at 80%. In the accuracy category, Screenshot API, Restpack, Urlbox, and ApiFlash all scored 100%, but Screenshotlayer and Abstract API both had issues displaying Thai language content correctly. Finally, the average speed performance of Screenshotlayer was the best, followed by Abstract API, Restpack, Urlbox, ApiFlash, and Screenshot API.

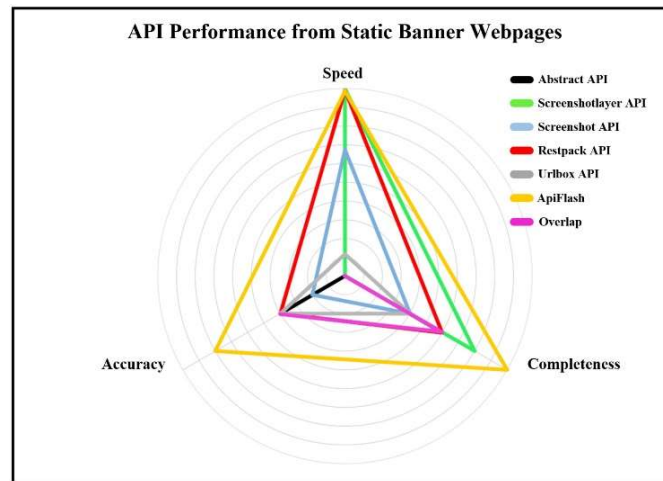


Figure 7. Results of the API performance test using static banner webpages. Scores for the three criteria range from 0% (center point) to 100% (outermost ring). The providers shown in black, green, and red (Abstract API, Screenshotlayer, and Restpack) share overlapping segments in the two pink lines.

Figure 7 shows that when using static banner webpages, the completeness criteria was led by ApiFlash with a score of 100%, followed by Screenshotlayer at 80%, and then the others. In the accuracy category, ApiFlash again performed best, this time with a score of 80%, with other providers trailing. The average speed score performance of Screenshotlayer was the best, followed by Restpack. After considering the results from both scrolling and static banner webpages, ApiFlash was chosen as the API provider for WatchNSend because ApiFlash can handle both types of banners and it scores an average of 86.01% on the three performance criteria.

3.2 The AOI processing time test

The results of this experiment to determine the time required to crop an incoming image to the AOI and then perform both the Edge Calculation and Overlay Calculation are shown in Figure 8.

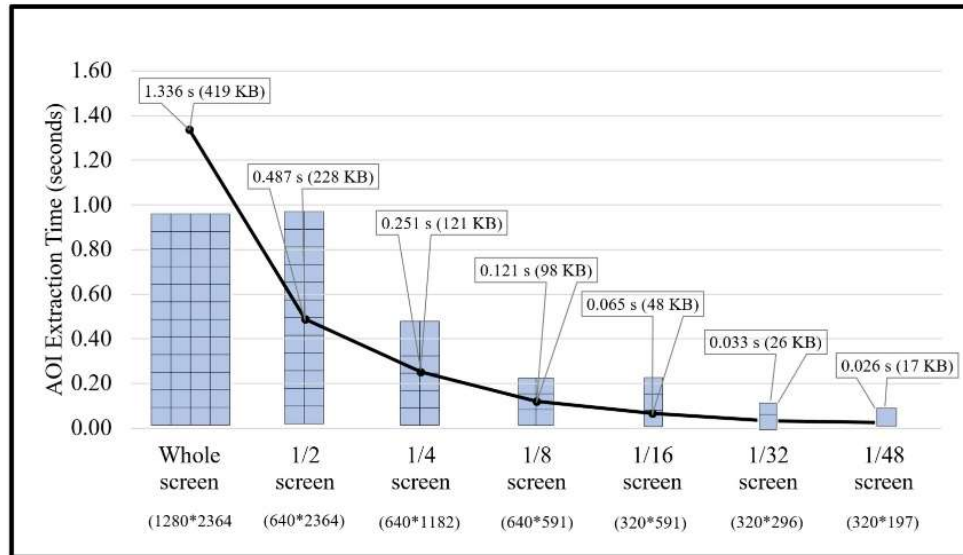


Figure 8. The time required to process various size Areas Of Interest (AOI)

These results show that the time required to process an AOI corresponds to the size of the AOI. The larger the AOI, the longer the processing time. The largest size took 1.336 seconds to process and the smallest took 0.026 seconds. The average processing time across all 7 sizes was 0.331 seconds, indicating that WatchNSend can process an average of 181 images per minute.

3.3 The Comparison Module test

The results of this test to investigate WatchNSend's ability to detect the four kinds of progressive image changes from Figure 5 using a combination of the Edge Calculation and Overlay Calculation are shown in Table 2.





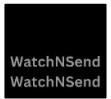




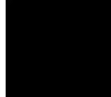






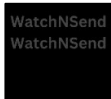
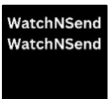

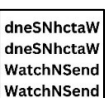




Table 2. The results of the Edge Calculation and Overlay Calculation for four kinds of progressive image changes from Figure 5, along with the Monitored Webpage Percent Change (MWPC)

Test	PI and CI Comparison	Edge Calculation Result	Overlay Calculation Result	MWPC
1	A compared to B	25.00	4.52	25.00
	A compared to C	49.99	9.05	49.99
	A compared to D	74.99	13.57	74.99
	A compared to E	99.98	18.10	99.98
2	F compared to G	Inf (0)	25.21	25.21
	F compared to H	Inf (0)	50.08	50.08
	F compared to I	Inf (0)	75.13	75.13
	F compared to J	Nan (0)	100.00	100.00
3	K compared to L	2.95	4.53	4.53
	K compared to M	5.91	9.07	9.07
	K compared to N	8.86	13.60	13.60
	K compared to O	11.81	18.14	18.14
4	P compared to Q	1.16	4.42	4.42
	P compared to R	2.32	8.84	8.84
	P compared to S	3.49	13.26	13.26
	P compared to T	4.65	17.69	17.69

Remember that whichever result from the Edge Calculation or Overlay Calculation is larger, that result is designated as the Monitored Webpage Percent Change (MWPC). In Table 2, the MWPC results for Test 1 and Test 2 are very tidy. They reflect that the progression of changes to the original image is approximately 25%, 50%, 75%, and 100%. The image differences in Test 3 and Test 4 were more difficult for the Edge Calculation and Overlay Calculation to evaluate, however the MWPC results for these two tests still reflect a steady and proportional increase, which is good. Users

of the application can be notified of these types of webpage changes by selecting one of the lower Notification Thresholds. In general, a high result for the Edge Calculation occurs when text changes on the MW. On the other hand, a high result for the Overlay Calculation generally occurs when the background color, text color, or text orientation change. Examples of the image results from each stage of the Edge Calculation and Overlay Calculation for the four Comparison Module tests are shown in Table 3.

Table 3. Examples of the image results from each stage of the Edge Calculation and Overlay Calculation for the four Comparison Module tests

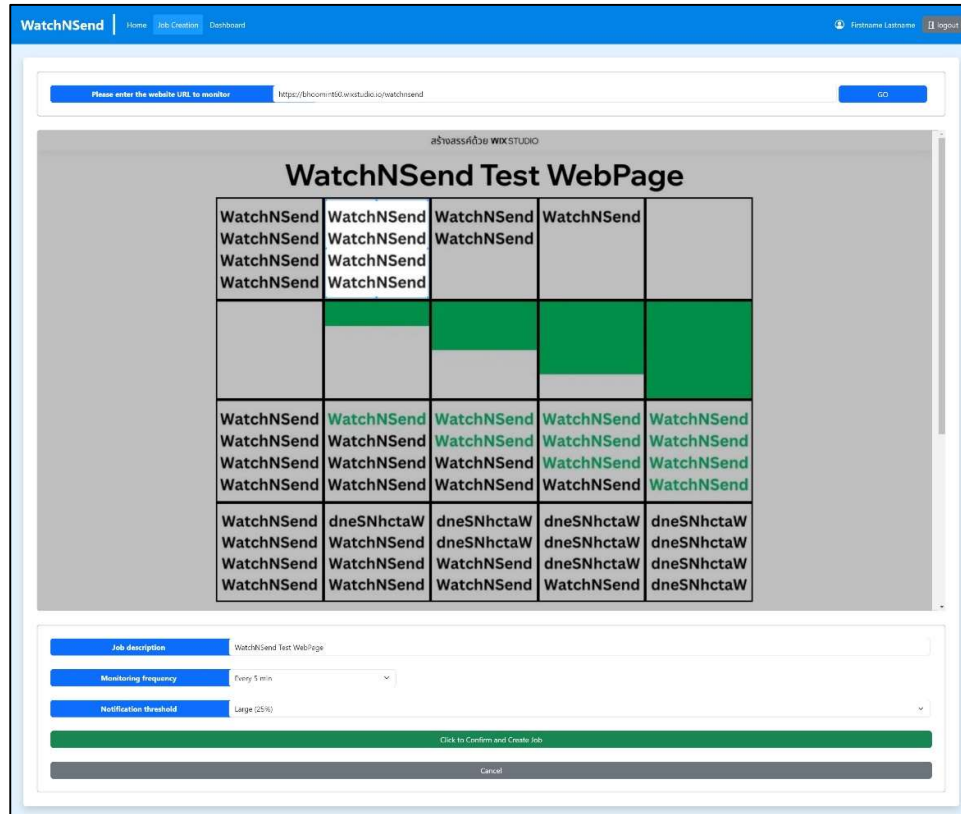
Compared Image Pair from Figure 5	GPI	GCI	EPI	ECI	OGI	BOGI
A and C						
F and H						
K and M						
P and R						

GPI = Grayscale image converted from the color PI, GCI = Grayscale image converted from the color CI, EPI = Edge of the PI by Canny edge detection, ECI = Edge of the PI by Canny edge detection, OGI = Overlay Grayscale Image, and BOGI = the binary image from the OGI image using the threshold value.

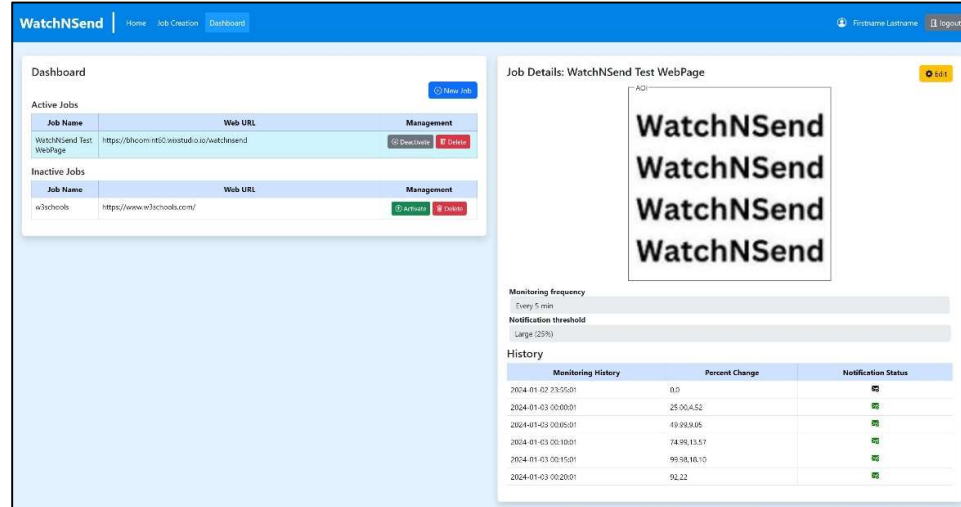
3.4 Application Development

WatchNSend was developed into a web application with which users can easily create jobs to monitor and then view or update current or past jobs. There are two screens, and they are shown in Figure 9. Figure 9(A) is the Job Creation screen, where users can enter new jobs and specify the area of interest to monitor, along with other job details. Figure 9(B) is the dashboard screen where users can see an

overview of current and past/inactive jobs. By clicking on one of the jobs in the list on the left, the user can view the details of that specific job on the right and initiate changes if desired. Also in the dashboard job details is the monitoring history of the selected job, along with access to current and past notifications of webpage changes, which pop open when the notification icon is clicked (not shown).



(A)



(B)

Figure 9. Sample screenshots of WatchNSend's (A) Job Creation screen and (B) Dashboard screen.

4. Discussion

Beyond its ability to accurately and efficiently detect webpage content changes, WatchNSend is also able to perform functions

and detection modes on par with commercially available webpage monitoring tools, as shown in Table 4.

Table 4. The functionality and detection mode of WatchNSend compared with some other CDN tools

CDN Tool	Functionality				Detection Mode			
	MSP	MMP	SSD	FICS	Notify	Visual	Text	HTML
WatchNSend	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Visualping.io [7]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Fluxguard [8]	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Sken [9]	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Pagescreen [10]	Yes	Yes	Yes	Yes	Yes	Yes	No	No
OnWebChange [11]	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

SSD=Server side detection, MMP=Monitor multiple pages, MSP= Monitor a single page, FICS = Fixed Interval checks, and Notify = Email notification

The six tools shown use different combinations of detection modes, but the one detection mode that most all of the tools have in common is the visual mode. The exact analytical methods of the other tools shown are assumed to vary; however, this information is proprietary and not available. Through the current research, the authors wish to disseminate a full example of how such a tool can be created, including robust scheduling and accurate detection. WatchNSend utilizes visual detection. Text detection and HTML tag detection predate the advent of visual detection, which functions broadly in this application to accurately detect changes to a webpage's text, its background color, its text color, or the direction of the text. Possibilities for future extension of this research include exploring novel algorithms for change detection beyond edge and overlay calculations, enabling detection of changes in HTML structure, delivering trend analysis to users, and conducting user studies to evaluate their experiences with WatchNSend.

5. Conclusion

The most distinguishing features of WatchNSend are its Edge Calculation, Overlay Calculation, JobFlights scheduling method, and use of Screenshot API. These features are written and combined using PHP, CSS, and JavaScript, which integrate well with the database, creating a flexible and user-friendly tool that can accurately perform an average of 181 website update checks per minute. WatchNSend's non-proprietary methods hold their own against commercial products, making WatchNSend an economical and expandable application for any individuals or businesses

that need to monitor webpages in a convenient, reliable way.

6. Acknowledgements

The authors would like to thank the Research and Development Institute of Kamphaeng Phet Rajabhat University as well as the Computer Vision and Human Interaction Technologies Laboratory at Kamphaeng Phet University (KPRU Vision Lab) for their kind assistance with this project. Thank you also to Paul Freund of Naresuan University Writing Clinic (DIALD) for editing this manuscript.

7. References

- [1] Altulaihan, E. A., A. Alismail and M. Frikha. A survey on web application penetration testing. *Electronics* 2023, 12, 1229. <https://www.mdpi.com/2079-9292/12/5/1229>.
- [2] Xing, Y., J. Shell, C. Fahy, T. Xie, H. Kwan and W. Xie. Web xr user interface research: Design 3d layout framework in static websites. *Applied Sciences* 2022, 12, 5600. <https://www.mdpi.com/2076-3417/12/11/5600>.
- [3] Mallawaarachchi, V., L. Meegahapola, R. Madhushanka, E. Heshan, D. Meedeniya and S. Jayarathna. Change detection and notification of web pages: A survey. *ACM Computing Surveys (CSUR)* 2020, 53, 1-35. <https://dl.acm.org/doi/pdf/10.1145/3369876>.
- [4] Shobhna, M. C. A survey on web page change detection system using different approaches. *International Journal of Computer Science and Mobile Computing* 2013, 2, 294-99.

- <https://ijcsmc.com/docs/papers/June2013/V2I6201391.pdf>.
- [5] Wachete - monitor web changes. Available online: <https://www.wachete.com/>. (accessed on 20 January 2024).
- [6] Fenton, E. Best 5 free website change monitoring software 2023. Available online: <https://visualping.io/blog/best-free-website-change-detection-monitoring-tools/>. (accessed on 20 January 2024).
- [7] Visualping - website change detection and alerts. Available online: <https://visualping.io/>. (accessed on 20 January 2024).
- [8] Fluxguard - monitor website changes with chatgpt. Available online: <https://fluxguard.com/>. (accessed on 20 January 2024).
- [9] Sken - monitor website changes. Available online: <https://sken.io/>. (accessed on 20 January 2024).
- [10] Pagescreen - automated website change detection, monitoring and alerts. Available online: <https://pagescreen.io/>. (accessed on 20 January 2024).
- [11] Onwebchange - track web page changes and get notified. Available online: <https://onwebchange.com/>. (accessed on 20 January 2024).
- [12] Verisign - domain name industry brief: 354.0 million domain name registrations in the first quarter of 2023 Available online: <https://blog.verisign.com/domain-names/verisign-q1-2023-the-domain-name-industry-brief/>. (accessed on 20 January 2024).
- [13] Odeh, A. Analytical and comparison study of main web programming languages - asp and php. TEM Journal 2019, 8, 1517-1522. https://www.temjournal.com/content/84/TEMJournalNovember2019_1517_1522.pdf.
- [14] kaduru, N. - Cropper js. Available online: <https://codepen.io/Narendrakaduru/pen/oWevXY>. (accessed on 20 January 2024).
- [15] Meegahapola, L., R. Alwis, E. Nimalarathna, V. Mallawaarachchi, D. Meedeniya and S. Jayarathna. Detection of change frequency in web pages to optimize server-based scheduling. Presented at 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer), 2017; pp. 1–7. <https://doi.org/10.1109/ICTER.2017.8257791>.
- [16] Setareh and Mehdi. A new method of scheduling tasks in cloud computing. Revista Publicando. 2018, 5, 227–45. <https://revistapublicando.org/revista/index.php/crv/article/view/1661>.
- [17] Selenium - selenium automates browsers. Available online: <https://www.selenium.dev/>. (accessed on 20 January 2024).
- [18] Canny, J. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8. 1986, 32, 679–98. <https://doi.org/10.1109/TPAMI.1986.4767851>.
- [19] Bansilal, S. The application of the percentage change calculation in the context of inflation in mathematical literacy. Pythagoras. 2017, 38, 1362–1363. <https://doi.org/10.4102/pythagoras.v38i1.314>.
- [20] Niitsuma, H. and T. Maruyama. Sum of absolute difference implementations for image processing on fpgas. In Proceedings of the International Conference on Field Programmable Logic and Applications, 2010; pp. 167–70. <https://doi.org/10.4102/pythagoras.v38i1.314>.
- [21] Otsu, N. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics. 1979, 9, 62–66. <https://doi.org/10.1109/TSMC.1979.4310076>.
- [22] Jin, L., L. Zhang and L. Zhao. Max-difference maximization criterion: A feature selection method for text categorization. Frontiers of Computer Science. 2023, 17, 171337. <https://doi.org/10.1007/s11704-022-2154-x>.
- [23] Peker, S., G. G. Menekse Dalveren and Y. İnal. The effects of the content elements of online banner ads on visual attention: Evidence from an-eye-tracking study. Future Internet 2021, 18, 657–658. <https://www.mdpi.com/1999-5903/13/1/18>.
- [24] Google alerts - monitor the web for interesting new content. Available online: <https://www.google.com/alerts..> (accessed on 20 January 2024).

[25] Islam, M. J., S. Ahmad, F. Haque, M. B. I. Reaz, M. A. S. Bhuiyan and M. R. Islam. Application of min-max normalization on subject-invariant emg pattern

recognition. Transactions on Instrumentation and Measurement. 2022, 71, 1–12.
<https://doi.org/10.1109/TIM.2022.3220286>.