

WEBSITE UPDATE CHECKER USING WEB CRAWLER

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering
by

P. B. S. S. ADITYA REDDY (37110528)

PESALA VENKATA NITISH (37110566)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAA

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **P. B. S. S. Aditya Reddy (37110528)**, **Pesala Venkata Nitish (37110566)** who carried out the project entitled as **Website Update Checker Using Web Crawler** under my supervision from November 2020 to March 2021.

Internal Guide

Dr. L. SUJHELEN, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We **P. B. S. S Aditya Reddy (37110528), Pesala Venkata Nitish (37110566)** hereby declare that the Project Report entitled “**Website Update Checker Using Web Crawler**” done by me under the guidance of **Dr. L. SUJHELEN, M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

SIGNATURE OF THE CANDIDATE

PLACE:

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to **Board of Management of Sathyabama Institute Of Science And Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D.**, and **Dr. L. Lakshmanan, M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. SUJHELEN, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The World Wide Web is the biggest assortment of information today and it keeps expanding step by step. A web crawler is a program from the tremendous downloading of pages from World Wide Web and this interaction is called Web crawling. To gather the site pages from www an internet searcher utilizes web crawler and the web crawler gathers this by web crawling. Because of limits of organization transmission capacity, tedious and equipment's a Web crawler can't download all the pages, it is critical to choose the main ones as right on time as conceivable during the slithering cycle and abstain from downloading and visiting numerous insignificant pages. Pursuing saving time utilizing the most recent innovation is the principle rationale to assemble this site update checker. This venture for the most part centers around distinguishing the progressions that happen in our necessary site moment to moment and it will send a data to mail if any change happens in site or not.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	IV
	LIST OF FIGURES	IX
	LIST OF TABLES	X
1	INTRODUCTION	01
	1.1 OVERVIEW	01
	1.2 PURPOSE OF WEB CRAWLER	02
	1.3 PROBLEM STATEMENT	03
	1.4 OBJECTIVES	03
2	LITERATURE SURVEY	04
3	SYSTEM ANALYSIS	06
	3.1 DRAWBACKS OF EXISTING SYSTEM	06
	3.2 PROPOSED SYSTEM	06
	3.3 SYSTEM REQUIREMENTS	06
	3.3.1 HARDWARE REQUIREMENTS	07
	3.3.2 SOFTWARE REQUIREMENTS	07
4	SYSTEM DESIGN	08
	4.1 PROPOSED WORK	08
	4.2 ARCHITECTURE DIAGRAM	09
	4.2.1 TAKING USER INPUTS	09
	4.2.2 VALIDATION	10

	4.2.3 CONTENT FETCHING	11
	4.2.4 SLEEP	11
	4.2.5 REPETITION OF TASKS	12
	4.2.6 MAIL SENDING	13
	4.3 ALGORITHMS USED	14
	4.3.1 BREADTH FIRST SEARCH ALGORITHM	14
	4.3.2 DEAPTH FIRST CRAWLING ALGORITHM	14
	4.3.3 CRAWLING THE LARGEST SITES FIRST	15
	4.3.4 CRAWLING THROUGH URL ORDERING	15
	4.3.5 BY HTTP GET REQUEST AND DYNAMIC WEB PAGE	15
5	SYSTEM IMPLEMENTATION	17
	5.1 MODULES	17
	5.1.1 WEB DRIVER	17
	5.1.2 DJANGO	17
	5.1.3 EMAIL	19
	5.1.4 TIME	21
	5.1.5 CV2	21
	5.2 RESULTS AND DISCUSSION	22
6	CONCLUSION AND FUTURE WORK	28
	6.1 CONCLUSION	28
	6.2 FUTURE WORK	28
	REFERENCES	29
	APPENDIX	31
	A. SOURCE CODE	31

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
4.2	ARCHITECTURE DIAGRAM	09
5.1	FRONTEND SCREEN	22
5.2	VALIDATING URL	22
5.3	VALID URL	23
5.4	INVALID URL	23
5.5	AFTER FILLING DATA	24
5.6	SUBMITTING REQUEST	24
5.7	PROCESSING REQUEST	25
5.8	AFTER REQUEST PLACED ACKNOWLEDGEMNT WE GET	25
5.9	CHANGES DETECTED	26
5.10	CHANGES NOT DETECTED	26
5.11	FINAL OUTPUT SCREEN	27

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
2.1	LITERATURE SURVEY	05

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Web crawler is a program that acts as an automated script which browses through the internet in a systematic way. The web crawler looks at the keywords in the pages, the kind of content each page has and the links, before returning the information to the search engine. This process is known as Web crawling.

The page you need is indexed by a software known as web crawler. A web crawler gathers pages from the web and then, indexes them in a methodical and automated manner to support search engine queries. Crawlers would also help in validating HTML codes and checking links.

These web crawlers go by different names, like bots, automatic indexers and robots. Once you type a search query, these crawlers scan all the relevant pages that contain these words and turn it into a huge index.

For example, if you are using Google's search engine, then the crawlers would go through each of the pages indexed in their database and fetch those pages to Google's servers. The web crawler follows all the hyperlinks in the websites and visits other websites as well.

So when you ask the search engine for a 'course in software development', it will come up with all the web pages that feature the term. Web crawlers are configured to monitor the web regularly so the results they generate are updated and timely.

Web crawlers have become so important to companies having a strong online presence, and they use it to obtain data like product information, reviews, pricing details and images to ensure they deliver better than what their competitors give. Web crawlers can, thus, make an impact on every aspect of business.

It could be an e-commerce site or a travel-based comparison site, but the presence of web crawlers makes all the difference to the end user. Everywhere businesses are looking for ways to beat their competition trying to provide better quality products at reasonable prices.

1.2 PURPOSE OF WEB CRAWLER

A web crawler is an internet bot that systematically browses world wide web for the purpose of extracting useful information.

The Internet is constantly changing and expanding. Because it is not possible to know how many total webpages there are on the Internet, web crawler bots start from a seed, or a list of known URLs. They crawl the webpages at those URLs first. As they crawl those webpages, they will find hyperlinks to other URLs, and they add those to the list of pages to crawl next.

Given the vast number of webpages on the Internet that could be indexed for search, this process could go on almost indefinitely. However, a web crawler will follow certain policies that make it more selective about which pages to crawl, in what order to crawl them, and how often they should crawl them again to check for content updates.

The relative importance of each webpage: Most web crawlers don't crawl the entire publicly available Internet and aren't intended to; instead they decide which pages to crawl first based on the number of other pages that link to that page, the amount of visitors that page gets, and other factors that signify the page's likelihood of containing important information.

The idea is that a webpage that is cited by a lot of other webpages and gets a lot of visitors is likely to contain high-quality, authoritative information, so it's especially important that a search engine has it indexed – just as a library might make sure to keep plenty of copies of a book that gets checked out by lots of people.

Revisiting webpages: Content on the Web is continually being updated, removed, or moved to new locations. Web crawlers will periodically need to revisit pages to make sure the latest version of the content is indexed.

For revisiting the sites using this bots will be helpful in collecting the contents from them so it will be helpful to everybody who is potentates the data. It makes the

world conscious.

1.3 PROBLEM STATEMENT

To automate the process of checking the updates or changes of a webpage in its contents or by any visual change occurred on the webpage so that it could help an individual who wants to monitor a particular webpage for a period of time in a while.

1.4 OBJECTIVE

- ❖ This project mainly focuses on detecting the changes that occur in the given web page from the requested time to the end of the time that is provided by the individual and with the mentioned period of intervals to check for the changes or updates created by the webpage owner. So if that happens a quick notification will be triggered to the user by sending the mail to the given mail id.
- ❖ This save the time and efforts to check for content changes to that individual who is caring the task manually.
- ❖ This can be used process of deployment of new webpage contents by developers to get updates on what they have updated and to check whether the updates are reflected or not.

CHAPTER 2

LITERATURE SURVEY

Web driver Tool: A Web based Automation Testing Framework

In the year 2014, RigzinAngmo, Monika Sharma [2] discussed about the selenium framework. A framework may be for a set of functions within a system that interrelate the layers of an operating system and the layers of an application subsystem as well as how communication should be standardized at some level of a network. Some frameworks in computer system also include programming interfaces ,programs.

The author proposed approach which utilizes versatile specialists to slither the pages. Versatile specialist is made, sent, at long last got and assessed in its proprietor's home setting. These versatile crawlers are moved to the site of the source where the information dwell to sift through any pointless information locally before shipped it back to the web search tool. These versatile crawlers can diminish the organization load by lessening the amount of information communicated over the organization. Utilizing this methodology channel those website pages that are not adjusted utilizing portable crawlers however recovers just those site pages from the far off workers that are really changed and play out the sifting of non-altered pages without downloading the pages. Their relocating crawlers move to the web workers, and do the downloading of web reports, handling, and extraction of catchphrases. In the wake of packing, move the outcomes back to the focal web crawler

The time shown the way to introduced another model and engineering of the Web Crawler utilizing various HTTP associations with WWW. The various HTTP association is applied utilizing numerous strings and nonconcurrent downloader part so the generally speaking downloading measure is ideal. The client gives the underlying URL from the GUI gave. It starts with a URL to visit. As the crawler visits the URL, it recognizes all the hyperlinks accessible in the site page and attaches them to the rundown of URLs to visit, known as the slither wilderness. URLs from the wilderness is iteratively visited and it closes when it arrives at in

excess of five levels from each home pages of the sites visited and it is cultivated that it isn't

needed to go further than five levels from the landing page to catch the majority of the pages visited by individuals while attempting to recover data from the web.

S.No	Author	Title	Goals	Perspective
1	Olston, C. and Pandey	Recrawl scheduling based on information longevity	Achieved the recrawling concept by and gathered contents on works wide web	This provides the enthusiasm to create the automated content fetcher by using concepts of crawler
2	Junghoo Cho and Hector Garcia-Molina	Parallel crawlers	non linier algorithm to achieve collection of content on the web using a single crawler	To implement the modern crawlers for gather the content non linearly and by using multiple phases of it
3	H. M. Moftah and S. M. Abuelenin	Elastic Web crawler service oriented architecture	Provideing an architecture over the crawling methodology	Implementing an architecture over this can save the time
4	Nataliya Boyko; Oleg Basystiuk; Nataliya Shakhovska	Comparison of Image and Face Recognition softwares	Provides detailed review about cv2 module to compare images	Provides ideas about the grayscaling effect to distinguish between the images
5	M. Cafarella and D. Cutting	Building nutch: Open source search	Searching for content on a webpage	Efficient and effective collection process for crawlers
6	K Usha Manjari; Syed Rousha; Dasi Sumanth; J Sirisha Devi	Extractive Text Summarization from Web pages using Selenium and TF-IDF algorithm	To obtain an overview of the content present in numerous documents, is a time-consuming task. Similarly, searching for specific information online	Fast and most accurate data collection from the web

TABLE 2.1 LITERATRE SURVEY

CHAPTER 3

SYSTEM ANALYSIS

3.1 DRAWBACKS OF EXISTING SYSTEM

There are very efficient existing systems are there in the theorized work but not more than ten percent of them come live to work on the real life to automate the process the following are the few reasons for that

- Updation of protocols on the websites may cause the activity to fail the algorithms
- Caching and cookie techniques can false result when crawling the webpages
- Algorithms can be deviated and fooled by the latest page indexing policies by search engines and web spiders working on them

3.2 PROPOSED SYSTEM

The main theme of this project, is to detect the changes that occur in our required website minute to minute and sending that information to end user via mail.

The implementation phase begins with gathering inputs from user and do the following steps:

- ❖ Validation
- ❖ Content Fetching
- ❖ Sleep
- ❖ Repetition of tasks
- ❖ Mail Sending

After performing all the above steps, we can able to detect the changes that occurred in website.

3.3 SYSTEM REQUIREMENTS

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output

requirements it gives the wants in term of input to produce the required output.

The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

3.3.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented.

- ❖ System : Intel core
- ❖ Speed : 2.4GHZ
- ❖ Hard Disk : 500 GB and Above
- ❖ Ram : 4GB and Above
- ❖ Net Speed : 2Mbps and Above

3.3.2 SOFTWARE REQUIREMENTS

The software requirements are the specification of the system. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

- ❖ Operating System : Windows
- ❖ Coding language : Python
- ❖ IDE : Python IDLE
- ❖ Web framework : Django

CHAPTER 4

SYSTEM DESIGN

4.1 PROPOSED WORK

The working of Web crawler is starting with introductory arrangement of URLs known as seed URLs. They download site pages for the seed URLs and concentrate new connections present in the downloaded pages. The recovered pages are put away and all around listed on the capacity territory so that by the assistance of these records they can later be recovered as and when required.

The separated URLs from the downloaded page are affirmed to know if their connected records have just been downloaded. In the event that they are not downloaded, the URLs are again allotted to web crawlers for additional downloading. This interaction is rehashed till no more URLs are missing for downloading. A large number of pages are downloaded each day by a crawler to finish the objective.

This venture fundamentally centers around recognizing the progressions that happen in our necessary site moment to moment and it will send a data to mail if any change happen in site or not. Mechanizing this undertaking utilizing site update checker will assume a gigantic part to understand what changes happen in site. Utilizing this we can discover changes happen in site either content or by any visual change you can get informed through email in regards to that change following change recognized. This will help a great deal to client as It encourages client to save time.

4.2 ARCHITECTURE DIAGRAM

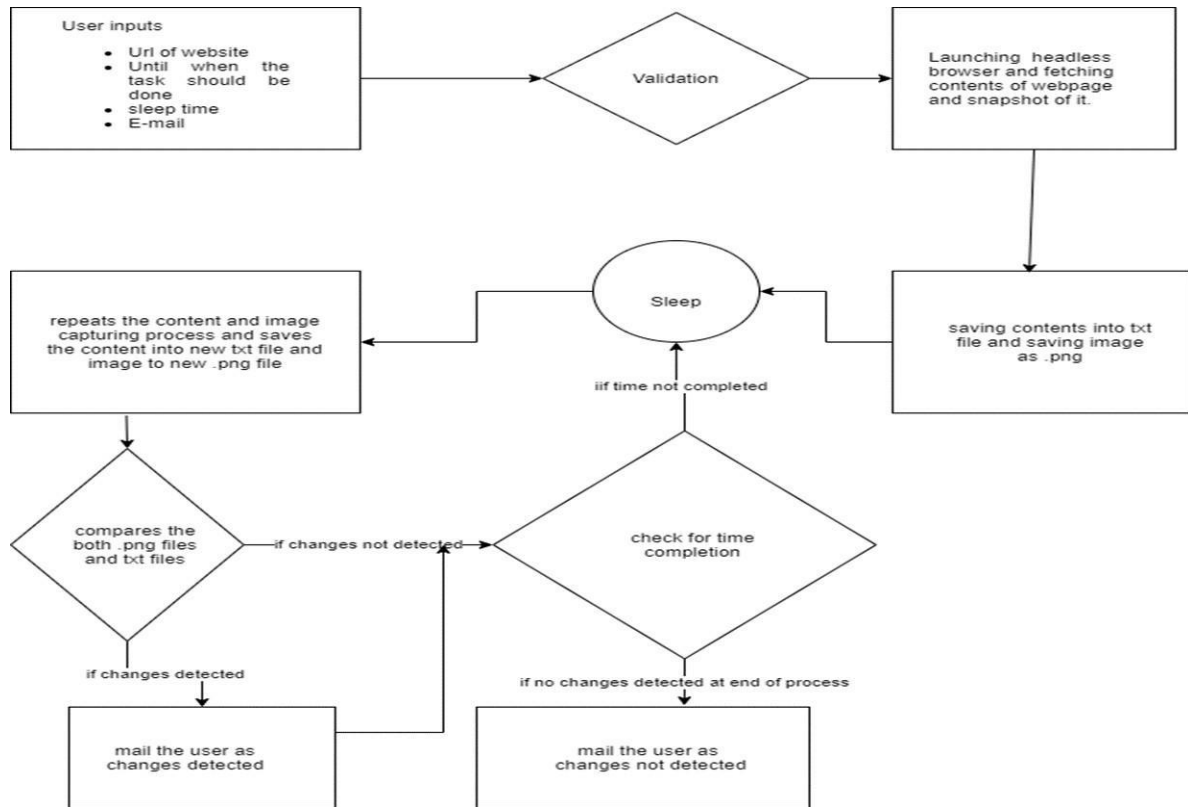


Fig 4.2 Architecture Diagram

4.2.1 TAKING USER INPUTS

First in this project we have to gather the inputs from end user. The various inputs that we have to gather here in this project are URL, Email, Date, Time and Sleep time. Here end user should have to enter the URL of the website on which he wants to track the changes. After entering the URL the URL value will be carried to server by the ajax calls so that the system can verify that the given URL is a working website and also to check web crawler allowed on that website, crawling by using web drivers we can able to connect to that website and conform that . In the next step end user should provide email id.

Here the email id is required because after tracking the changes in the website the user will get a mail to his provided email id regarding changes occurred in website. If any changes occurred in website means user will get mail. That mail consist

what

are the changes that happened in the website which was given by end user. If no changes occurred in the website means then also user will get a mail saying that no changes has been detected in website. End user should also provide the date and time until when the user wants to monitor the particular website. The Sleep time which is provided by the user is needed in seconds so that the particular period to system to take rest to recapture the website once the time period is done.

4.2.2 VALIDATION

Data validation is an essential part of any data handling task whether you're in the field collecting information, analysing data, or preparing to present data to stakeholders. If data isn't accurate from the start, your results definitely won't be accurate either. That's why it's necessary to verify and validate data before it is used.

While data validation is a critical step in any data workflow, it's often skipped over. It may seem as if data validation is a step that slows down your pace of work, however, it is essential because it will help you create the best results possible. These days data validation can be a much quicker process than you might've thought. With data integration platforms that can incorporate and automate validation processes, validation can be treated as an essential ingredient to your workflow rather than an additional step.

Validating the accuracy, clarity, and details of data is necessary to mitigate any project defects. Without validating data, you run the risk of basing decisions on data with imperfections that are not accurately representative of the situation at hand.

While verifying data inputs and values is important, it is also necessary to validate the data data model itself. If the data model is not structured or built correctly, you will run into issues when trying to use data files in various applications and software.

Both the structure and content of data files will dictate what exactly you can do

with data. Using validation rules to cleanse data before use helps to mitigate “garbage in = garbage out” scenarios. Ensuring the integrity of data helps to ensure the legitimacy of your conclusions.

When we accept inputs from user, we need to check whether the data is valid or not. There are three methods to check whether the data is valid or not.

1) TYPE CHECK

Type check is used to validate data type entered by the user on the webpage itself by using javascript and typecheck factors available in HTML5.

For example, It will helps the user to not to enter wrong input type In place of other types like if the wants to enter integer or float values but the given field type name so that the webpage throws the error to the user specifying that you are entering wrong value in place of name.

2) LENGTH CHECK

Length check is used to check the length of a input field so that the user cannot exceed the limit and cause problem to the system.

Length check is mainly used while validating passwords. If we give the maximum password length as 10 then the user cannot able to enter more than 10 characters.

3) RANGE CHECK

Range check is used to check whether a number entered is between two numbers or not.

For example, the admin keeps a constraint that the value entered by user should be in between 10 and 20 means the minimum value entered by user is 10 and maximum value entered by user is 20. If user enters a number below 10 or above 20 means it throws an error.

4.2.3 CONTENT FETCHING

In content fetching firstly we will fetch the content based on the details provided by user. When user enters the URL we have to fetch the content present on that web page.

In additional to content fetching we should also have to save the content present on web page in .txt file and also we have to take the snapshot of content present

on web page. The reason we are saving the content in .txt file.

This will make the system to detect even a small is occurred on behalf of the user and can prevent from not getting loss by any of the necessary information getting lost. This makes the environment more powerful and accurate to get the results.

Later in this process we will make sure that every time the content is fetched it should compared with previous time one by fetching and comparing content from the saved files in to string by linear regression algorithms

And as describes as and talking we moreover takes the snapshot of the webpage at that instance and saving it in .png file is because we have to compare this .png file with finally taken .png file. If any changes detected in two .png files means user will get notified with a mail regarding that change in addition to that we also compare contents of the page to notify the user using mail. If no changes detected means in that case also user will get a mail saying that no changes detected. This process is further discussed in the below sections

4.2.4 SLEEP

sleep() is defined in time() module of python 3. Sometimes, there is a need to halt the flow of the program so that several other executions can take place or simply due to the utility required. sleep() can come handy in such a situation which provides an accurate and flexible way to halt the flow of code for any period of time.

The sleep() function suspends (waits) execution of the current thread for a given number of seconds.

This function is used in this program to give rest for the system and also to avoid blocking from the website as continuous fetching will block the crawler from fetching and snapshotting the website

4.2.5 REPETATION OF THE TASKS

Based on the sleep time entered by user fetching the content again present on web page. The content on the web page will be fetched irrespective of the language type and format type and that will be saved to the .txt file and the file name will be the same as the title of the webpage. After that the system is set for

taking snap shot of whatever content present on that web page in .png file this file will be stored as the file with name as same as the title page.

The images stored and get compared after comparing both images before sleep time. If any changes detected between two .png files means user will get notified with a mail regarding that change so that the user can get use of the information related on that website. If no changes detected means it will check for the requested time until when the user is need to monitor the page in that case also user will get a mail saying that no changes detected.

After completion of sleep time again we have to save the content in new .txt file and also again we have to take the snapshot again. Now we have to compare both the snap shots. Repeat the same process continuously until the time entered by user is completed.

4.2.6 MAIL SENDING

For sending an email to a person or the user about any changes we have a use a email in first place that has to be a service email only to be used for sending mails and there should be no way a user could not replay to the email for this you have to block the incoming mails for this email address

For doing this using gmail create a email id using google account and after that login to the account enable two-factor authentication on that after that you can see an option will appear that says app passwords now enable that so that an hash will be generated now that is the password you have to use for sending mails to the user by using that email address

This process is getting triggered just after the Comparison of both snapshots of the webpages from different times. If any changes detected between the two .png files then this mail will automatically start striking users given email this would help the user to reach his favourite website to strike the changes immediately on that webpage.

If there are no changes detected between the two different times then the system will check for the time weather the given time is obtained in case the given time is not obtained the system will take a stop method by final intimating the user that the system could not identify any changes in the given time by alerting the user with most recent fetch information so that the user can place the new request with new

timelines.

It also can be said like means in that case also user will get a mail saying that no changes detected It will make the user to know that the system is not anymore looking on to that website.

4.3 ALGORITHMS USED

4.3.1 BREADTH FIRST SEARCH ALGORITHM

BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer wise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes.

As the name BFS suggests, you are required to traverse the graph breadthwise as follows:

1. First move horizontally and visit all the nodes of the current layer
2. Move to the next layer.

Expansiveness first calculation work on a level by level, for example calculation begins at the root URL and searches the all the neighbours URL at a similar level. On the off chance that the ideal URL is discovered, the inquiry ends. In the event that it isn't, search continues down to the following level and rehash the cycles until the objective is reached. At the point when all the URLs are examined, however the goal isn't discovered, at that point the disappointment announced is produced. Expansiveness first Search calculation is by and large utilized where the target lies in the depthless parts in a more profound tree.

4.3.2 DEPTH FIRST CRAWLING ALGORITHM

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Here, the word backtrack means that when you are moving forward and there are

no more nodes along the current path, you move backwards on the same path to

find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:

Pick a starting node and push all its adjacent nodes into a stack.

Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.

Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. Profundity first pursuit calculation is a more helpful inquiry what begins at the root URL and cross profundity through the kid URL.

To start with, we move to one side most kid on the off chance that at least one than one youngster exist and cross profound until no more is accessible Here backtracking is utilized to the following unvisited hub and cycles is reimbursed in comparable way. By the utilization of this calculations creators ensures that all the edges, for example all URL is visited once breath. It is proficient for search issues, yet when the kid is enormous then this calculation goes into a boundless circle .

4.3.3 CRAWLING THE LARGEST SITES FIRST

BaezaYates et al "Crawling a Country: Better Strategies than Breadth First for Web Page Ordering" perform tests in approximately 100 million website pages and find that slithering the huge webpage first plan has for all intents and purposes most valuable at that point on-line page significance calculation. The web crawler fined above all else un – slithered site pages to discover high need page for picking a site, and starts with the locales with the enormous number of forthcoming pages.

4.3.4 CRAWLING THROUGH URL ORDERING

Junghoo Cho et al "Proficient Crawling Through URL Ordering" locate that a crawler is to choose URLs and to filter from the line of referred to URLs to discover more significant pages first when it visits prior URLs that have anchor text which is like the driving question or connection distance is likewise short to a page and that

kind of pages to be known significant.

4.3.5 BY HTTP GET REQUEST AND DYNAMIC WEB PAGE

It is a Query based Approach to limit the Web Crawler or arachnid Traffic by utilizing HTTP Get Request and furthermore Dynamic Web Page. As per the creator it is a question based way to deal with educate all reports on the site by web crawler utilizing by Dynamic website page and furthermore HTTP GET Request. Also, crawler download just refreshed site pages after the last visit.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULES

5.1.1 WEB DRIVER

Web Driver is an open-source testing apparatus, which implies it very well may be downloaded from the web without spending anything. Web Driver is a utilitarian testing device for web and furthermore viable with non-useful testing instruments also. In this venture we utilized Web Driver for content getting, page title from a page and for page screen capture catching.

5.1.2 DJANGO

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading

is built with Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Scalable

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication,

reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

5.1.3 EMAIL

The email package is a library for managing email messages. It is specifically *not* designed to do any sending of email messages to SMTP (**RFC 2821**), NNTP, or other servers; those are functions of modules such as `smtplib` and `nntplib`.

The overall structure of the email package can be divided into three major components, plus a fourth component that controls the behavior of the other components.

The central component of the package is an “object model” that represents email messages. An application interacts with the package primarily through the object model interface defined in the `message` sub-module. The application can use this API to ask questions about an existing email, to construct a new email, or to add or remove email subcomponents that themselves use the same object model interface. That is, following the nature of email messages and their MIME subcomponents, the email object model is a tree structure of objects that all provide the `EmailMessage` API.

The other two major components of the package are the parser and the generator. The parser takes the serialized version of an email message (a stream of bytes) and converts it into a tree of `EmailMessage` objects. The generator takes an `EmailMessage` and turns it back into a serialized byte stream. (The parser and generator also handle streams of text characters, but this usage is discouraged as it is too easy to end up with messages that are not valid in one way or another.)

The control component is the policy module. Every `EmailMessage`,

every generator, and every parser has an associated policy object that controls its behavior. Usually an application only needs to specify the policy when

an `EmailMessage` is created, either by directly instantiating an `EmailMessage` to create a new email, or by parsing an input stream using a parser. But the policy can be changed when the message is serialized using a generator. This allows, for example, a generic email message to be parsed from disk, but to serialize it using standard SMTP settings when sending it to an email server.

The email package does its best to hide the details of the various governing RFCs from the application. Conceptually the application should be able to treat the email message as a structured tree of unicode text and binary attachments, without having to worry about how these are represented when serialized. In practice, however, it is often necessary to be aware of at least some of the rules governing MIME messages and their structure, specifically the names and nature of the MIME “content types” and how they identify multipart documents. For the most part this knowledge should only be required for more complex applications, and even then it should only be the high level structure in question, and not the details of how those structures are represented. Since MIME content types are used widely in modern internet software (not just email), this will be a familiar concept to many programmers.

The following sections describe the functionality of the email package. We start with the message object model, which is the primary interface an application will use, and follow that with the parser and generator components. Then we cover the policy controls, which completes the treatment of the main components of the library.

The next three sections cover the exceptions the package may raise and the defects (non-compliance with the RFCs) that the parser may detect. Then we cover the headerregistry and the contentmanager sub-components, which provide tools for doing more detailed manipulation of headers and payloads, respectively. Both of these components contain features relevant to consuming and producing non-trivial messages, but also document their extensibility APIs, which will be of interest to advanced applications.

Following those is a set of examples of using the fundamental parts of the APIs

covered in the preceding sections.

The foregoing represent the modern (unicode friendly) API of the email package. The remaining sections, starting with the Message class, cover the legacy compat32 API that deals much more directly with the details of how email messages are represented. The compat32 API does *not* hide the details of the RFCs from the application, but for applications that need to operate at that level, they can be useful tools. This documentation is also relevant for applications that are still using the compat32 API for backward compatibility reasons.

5.1.4 TIME

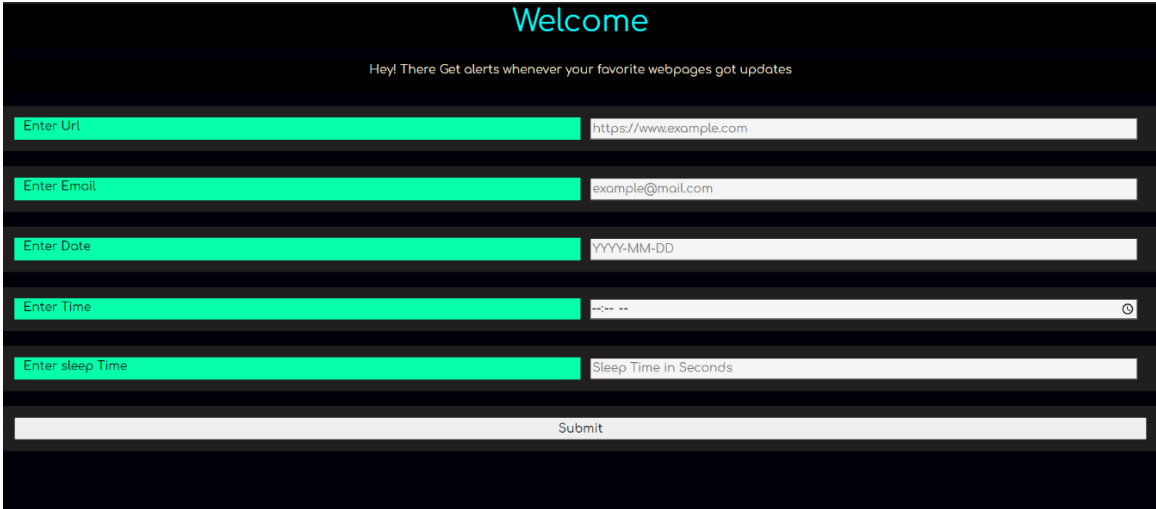
The Python time module provides many ways of representing time in code, such as objects, [numbers](#), and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code. It additionally gives usefulness other than addressing time, such as holding up during code execution and estimating the effectiveness of your code. In this undertaking we utilized TIME module to store the length that entered by end client and up to that time that entered by client we will send the updates that occurred in site through mail to client.

5.1.5 CV2

CV2 is an open source library which is supported by multiple platforms including Windows, Linux, and MacOS, and is available for use in multiple other languages as well; however, it is most commonly used in Python for Machine Learning applications, specifically in the Computer Vision domain.

Apart from its cross-platform support and availability in multiple other computer languages, which allows applications developed in it to be used on different systems, CV2 is also, in comparison to other such libraries, fairly efficient in terms of computations, as it uses vector operations for most of its functions.

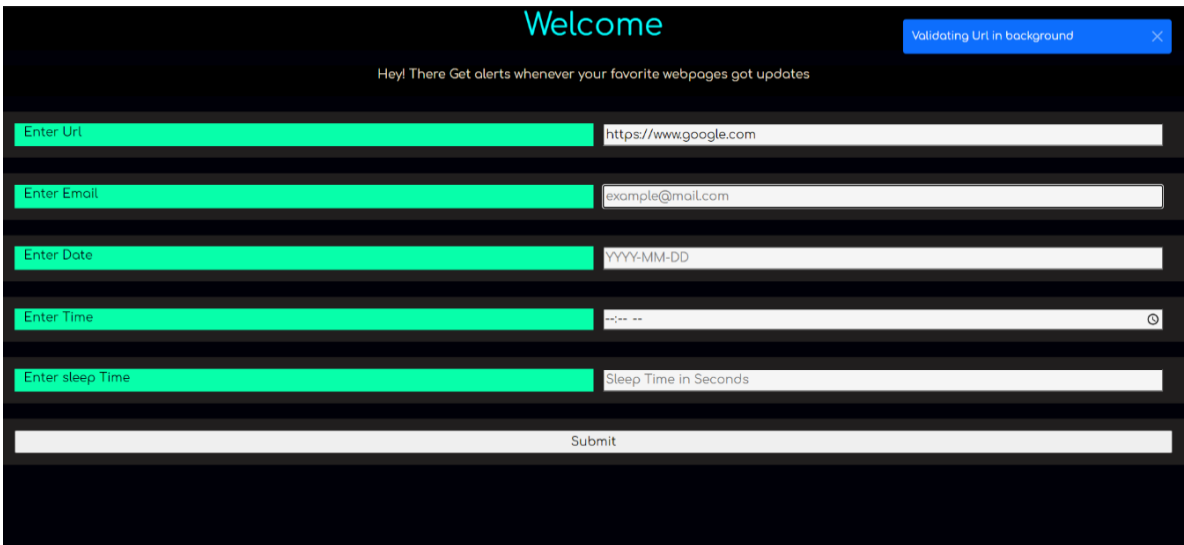
5.2 RESULTS AND DISCUSSION



The screenshot shows a web application interface with a dark background. At the top, the word "Welcome" is displayed in a light blue font. Below it, a message reads "Hey! There Get alerts whenever your favorite webpages got updates". The form consists of five input fields, each with a red label on the left and a white input area on the right. The labels are "Enter Url", "Enter Email", "Enter Date", "Enter Time", and "Enter sleep Time". The corresponding values in the input fields are "https://www.example.com", "example@mail.com", "YYYY-MM-DD", "--:--:--", and "Sleep Time in Seconds". At the bottom of the form is a wide, light gray "Submit" button.

Fig 5.1 FRONTEND SCREEN

This above screen is the user view of the page to submit the request to the system For start monitor the webpage.



This screenshot shows the same web application interface as Fig 5.1, but with updated data. The "Enter Url" field now contains "https://www.google.com". A blue notification banner in the top right corner reads "Validating Url in background" with a close button (X). The other input fields and the "Submit" button remain the same as in the previous figure.

Fig 5.2 VALIDATING URL

The above screen is whenever the url enter is completed by the user it will go to the system for permission verification for running crawlers on the website and also the url format checking.

The screenshot shows a web interface with a dark background. At the top, the word "Welcome" is displayed in a light blue font. Below it, a message says "Hey! There Get alerts whenever your favorite webpages got updates". In the top right corner, a green notification box contains the text "this website can be observed" with a close button. The form consists of several input fields, each with a red label on the left and a white input area on the right. The labels are "Enter Url", "Enter Email", "Enter Date", "Enter Time", and "Enter sleep Time". The corresponding input values are "https://www.google.com", "example@mail.com", "YYYY-MM-DD", "--:--:--", and "Sleep Time in Seconds". At the bottom of the form is a wide white button labeled "Submit".

Fig 5.3 VALID URL

The above screen is if the url given by the user and the website is ready for crawling it.

This screenshot shows the same web interface as Figure 5.3, but with an error state. The "Welcome" title and the "Hey! There" message remain. However, the green notification box is replaced by a red one that says "WE cannot work on this site" with a close button. The input fields for "Enter Url", "Enter Email", "Enter Date", "Enter Time", and "Enter sleep Time" contain the same values as in Figure 5.3: "https://www.youtube.com", "example@mail.com", "YYYY-MM-DD", "--:--:--", and "Sleep Time in Seconds". The "Submit" button is still present at the bottom.

Fig 5.4 INVALID URL

The above screen shows that if the entered url is not valid or the website is not valid for crawling.

The screenshot shows a web application interface with a dark background. At the top, the word "Welcome" is displayed in a light blue font. Below it, a subtitle reads "Hey! There Get alerts whenever your favorite webpages got updates". The form consists of five input fields, each with a red label on the left and a white input area on the right. The fields are: "Enter Url" with the value "https://www.google.com", "Enter Email" with "sairedy3212@gmail.com", "Enter Date" with "2021-03-16", "Enter Time" with "07:47 PM", and "Enter sleep Time" with "30". Below these fields is a wide, orange "Submit" button. The bottom of the form is a solid dark grey area.

Fig 5.5 AFTER FILLING DATA

The above screen is the fully filed form before submitting the request to the system

This screenshot shows the same web application interface as Fig 5.5, but after the form has been submitted. The "Submit" button is now a greyed-out, disabled button. A large, white, semi-transparent loading spinner (a 'C' shape) is centered over the form fields. In the top right corner, a green notification bar with white text says "your request is being processed". The "Welcome" header and subtitle remain the same.

Fig 5.6 SUBMITTING REQUEST

This above screen is showing the user side view after submitting the request.

Welcome

Hey! There Get alerts whenever your favorite webpages got updates

Enter Url

Enter Email

Enter Date

Enter Time

Enter sleep Time

Fig 5.7 PROCESSING REQUEST

This is the screen showing to user until the time is done

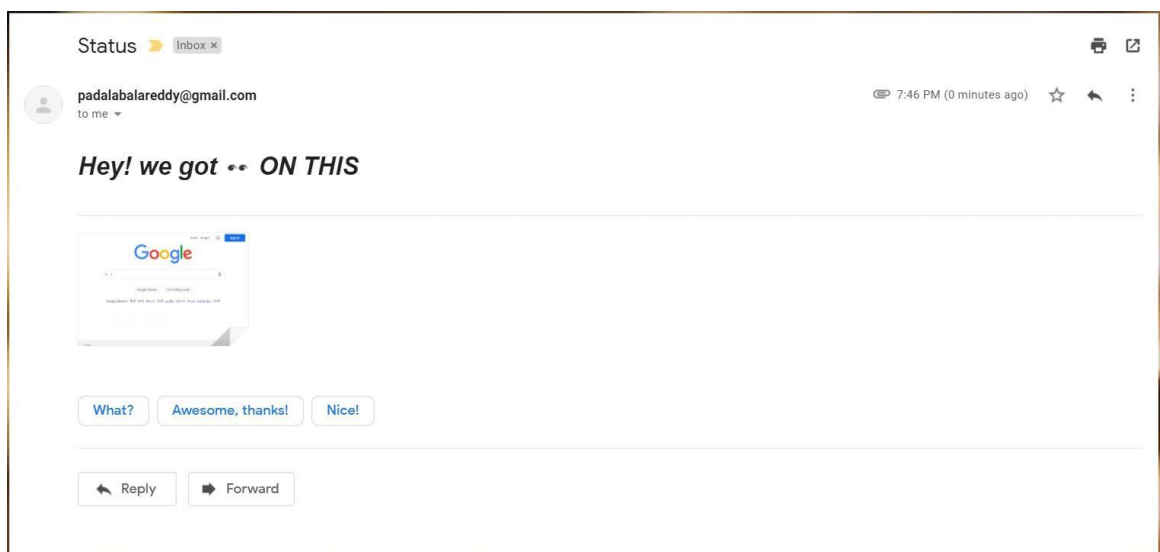


Fig 5.8 AFTER REQUEST PLACED ACKNOWLEDGEMNT WE GET

This is first acknowledgement get to the user mail id as we have started monitoring the website.

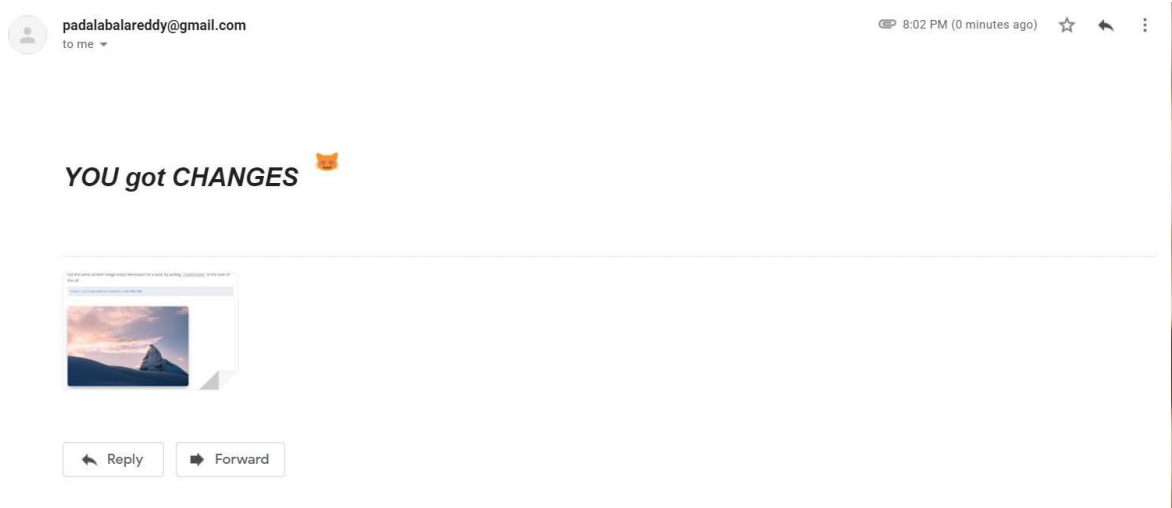


Fig 5.9 CHANGES DETECTED

The above screen is the alert the user got when there is a change detected by the system on the website.

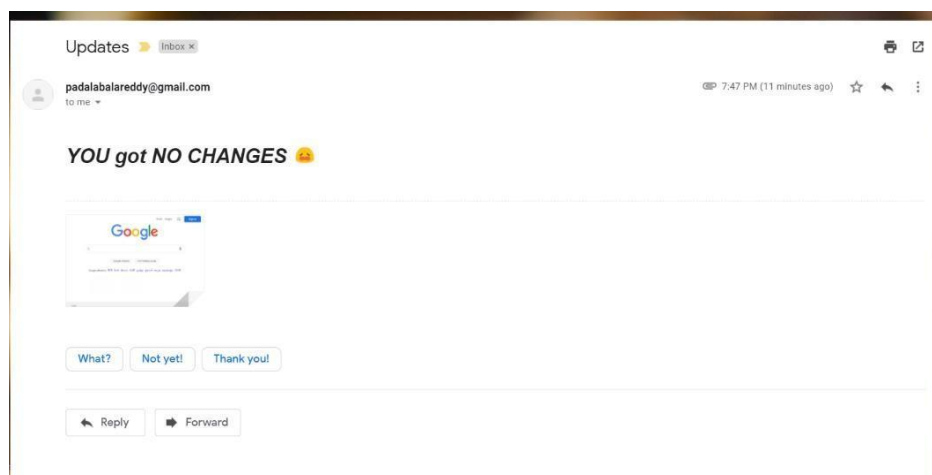


Fig 5.10 CHANGES NOT DETECTED

If there are no changes are detected until time is completed then the user will be notified like this.

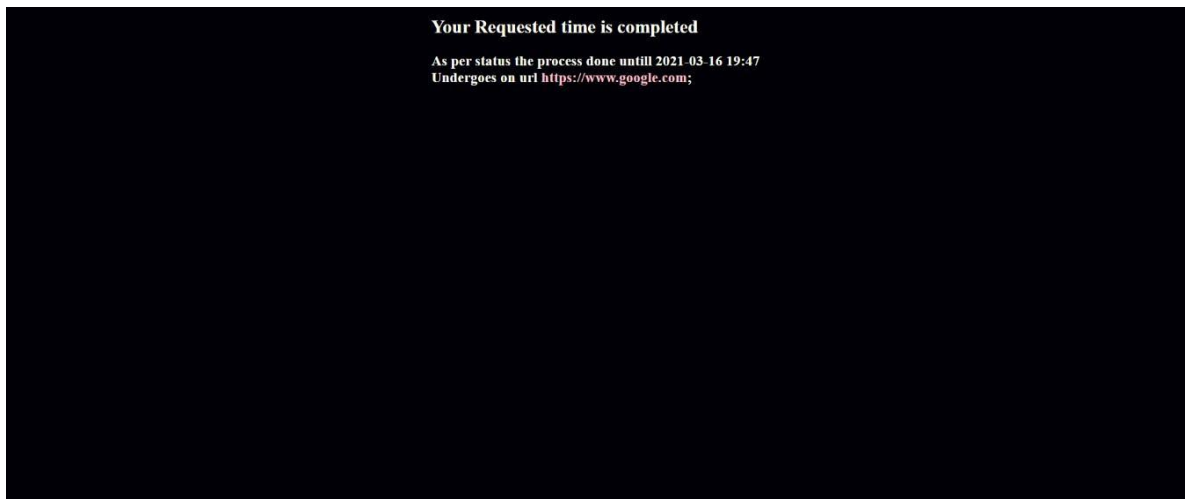


Fig 5.11 FINAL OUTPUT SCREEN

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Web Crawler is consequently crucial data recovery which navigates the Web and downloads web archives that suit the client's need. Web crawlers are intended to recover Web pages and addition them to nearby vault. Crawlers are essentially used to make an imitation of the relative multitude of visited pages which are subsequently handled by a web search tool that will file the downloaded pages that help in speedy inquiries. The client will get an alarm email after location of progress and furthermore get advised after the interaction if there are no adjustments in the recognition. Consequently, we can ready to save time by getting changes straightforwardly to mail without Refreshing the site page so often.

6.2 FUTURE WORK

The future work would add interface and ready notices if there should be an occurrence of blunders.

REFERENCES

- [1] H. M. Moftah and S. M. Abuelenin, "Elastic Web crawler serviceoriented architecture over cloud computing," *Arabian J. Sci. Eng.*, vol. 43, pp. 8111–8126, May 2018.
- [2] J. Yu, M. Li, and D. Zhang, "A distributed Web crawler model based on cloud computing," in *Proc. 2nd Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, vol. 66, 2016, pp. 276–279.
- [3] M. Cafarella and D. Cutting, "Building nutch: Open source search," *Queue*, vol. 2, no. 2, pp. 54–61, 2014.
- [4] F. Ye, Z. Jing, Q. Huang, and C. Hu, "The research and implementation of a distributed crawler system based on Apache Flink," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, vol. 3, 2018, pp. 90–98.
- [5] H. Xu, K. Li, and G. Fan, "An improved strategy of distributed network crawler based on Hadoop and P2P," in *Proc. Int. Conf. Appl. Techn. Cyber Secur. Intell.*, vol. 2, 2019, pp. 849–855.
- [6] L. Barbosa and J. Freire, "Siphoning Hidden-Web Data through KeywordBased Interfaces," *J. Inf. Data Manage.*, vol. 1, no. 1, pp. 133–144, May 2010.
- [7] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, C. Schallhart, and C. Wang, "DIADEM: Thousands of websites to a single database," *Proc. VLDB Endowment*, vol. 7, no. 14, pp. 1845–1856, Oct. 2014.
- [8] K. S. Kim, K. Y. Kim, K. H. Lee, T. K. Kim, and W. S. Cho "Design and Implementation of Web Crawler Based on Dynamic Web Collection Cycle", pp. 562-566, IEEE 2012
- [9] Anbukodi.S and Muthu Manickam.K "Reducing Web Crawler Overhead using Mobile Crawler", *PROCEEDINGS OF ICETECT*, pp. 926-932, 2011
- [10] Bahador Saket and Farnaz Behrang "A New Crawling Method Based on AntNet Genetic and Routing Algorithms", *International Symposium on Computing, Communication, and Control*, pp. 350-355, IACSIT Press, Singapore, 2011

- [11] Andoena Balla, Athena Stassopoulou and Marios D. Dikaiakos (2011), "Real-time Web Crawler Detection", 18th International Conference on Telecommunications, pp. 428-432, 2011
- [12] Song Zheng, "Genetic and Ant Algorithms Based Focused Crawler Design", Second International Conference on Innovations in Bio-inspired Computing and Applications pp. 374-378, 2011
- [13] P. Liakos, A. Ntoulas, A. Labrinidis, and A. Delis, "Focused crawling for the hidden Web," World Wide Web, vol. 19, no. 4, pp. 605–631, Jul. 2016.
- [14] C. Castillo, A. Nelli, and A. Panconesi, "A memory-efficient strategy for exploring the Web," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI Main Conf.)(WI), Dec. 2016, pp. 680–686.
- [15] D. Gunawan, Amalia, and A. Najwan, "Improving data collection on article clustering by using distributed focused crawler," J. Comput. Appl. Informat., vol. 1, no. 1, pp. 39–50, 2017.

Appendix

A. Source code

Functions.py

```
import os
```

```
from drivers import webdriver
```

```
from webdriver_manager.chrome import ChromeDriverManager
```

```
import requests
```

```
import cv2
```

```
import numpy as np
```

```
"""def comp(url,type):
```

```
    options= webdriver.ChromeOptions()
```

```
    options.headless=True
```

```
    driver =webdriver.Chrome(ChromeDriverManager().install(),options=options)
```

```
    driver.get(url)
```

```
    title = driver.title
```

```
    img1 = cv2.imread(title+'.png')
```

```
    img2 = cv2.imread(title+'.png')
```

```
    diff= cv2.subtract(img1,img2)
```

```
    result = np.any(diff)
```

```
print(result)
```

if result :

print("images are same")

else:

cv2.imwrite("results.jpg",diff)"""

#saves content of url to a file

def titlee(url):

options= webdriver.ChromeOptions()

options.headless=True

driver =webdriver.Chrome(ChromeDriverManager().install(),options=options)

driver.get(url)

title=driver.title

print('title fetched from functions')

return title

def contentFetcher(url,type):

options= webdriver.ChromeOptions()

options.headless=True

driver =webdriver.Chrome(ChromeDriverManager().install(),options=options)

driver.get(url)

l=driver.find_element_by_tag_name("body")

```
s=l.text
```

```
f=open(driver.title+type+'.txt','w+',encoding='utf-8')
```

```
f.write(s)
```

```
f.close()
```

```
print('from Functions: content fetcher')
```

```
driver.quit()
```

```
#saves screen shoot of a url
```

```
def capture(url,type):
```

```
    options= webdriver.ChromeOptions()
```

```
    options.headless=True
```

```
    driver =webdriver.Chrome(ChromeDriverManager().install(),options=options)
```

```
    driver.get(url)
```

```
    title=driver.title+type
```

```
    l=True
```

```
    try:
```

```
        s = lambda X: driver.execute_script('return  
document.body.parentNode.scroll'+X)
```

```
        driver.set_window_size(s('Width'),s('Height'))
```

```
driver.find_element_by_tag_name('body').screenshot(title+".png")
```

```
except:
```

```
    l=False
```

```
finally:
```

```
    print('Picture capture from fnctions')
```

```
    driver.quit()
```

```
print("helo fn ")
```

```
print(l)
```

```
return l
```

```
def siteStatus(url):
```

```
    try:
```

```
        request_response = requests.head(url)
```

```
    except :
```

```
        return False
```

```
    status_code = request_response.status_code
```

```
    website_is_up = status_code == 200
```

```
    return website_is_up
```

```
Forms.py
```

```
from django import forms
```

```
from .models import Post
```

```

class Inform(forms.ModelForm):

    class Meta:

        model = Post

        fields = "_all_"

        labels = {

            'url': 'Enter your url',

            'mail': 'Enter Mail',

            'id', 'dt': 'select date',

            'tim': 'select time',

            'slep': 'enter rest time'

        }

        widgets = {

            'url':
forms.URLInput(attrs={'id': 'url', 'onfocusout': 'urlfn()'}, placeholder='https://www.example.com', class_='input')),

            'mail':
forms.EmailInput(attrs={'id': 'mail', 'placeholder': 'example@mail.com', 'class': 'input'}),

            'dt': forms.DateInput(attrs={'id': 'datepicker', 'placeholder': 'YYYY-MM-DD', 'class': 'input'}),

            'tim': forms.TimeInput(attrs={'id': 'time', 'name': 'time', 'type': 'time', 'class': 'timepicker'}),

```

```
'slep':forms.NumberInput(attrs={'id':'slep','placeholder':'Sleep Time in  
Seconds','class':'input'})
```

```
}
```

```
def _init_(self):
```

```
    super(Inform,self).__init__()
```

Index.Html

```
{% load static %}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<link rel="stylesheet"
```

```
href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
```

```
<link rel="stylesheet"
```

```
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.16/themes/base/jquery-  
ui.css">
```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
```

```
beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
```

```
BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqlj61tLrc4wSX0szH/Ev+nYRRuWloIffl"
```

```
crossorigin="anonymous">
```

```
<link rel="preconnect" href="https://fonts.gstatic.com">
```

```
<link  
href="https://fonts.googleapis.com/css2?family=Comfortaa&display=swap"  
rel="stylesheet">
```

```
<link rel="stylesheet" href="{% static 'styles.css' %}">
```

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

```
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
```

```
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
```

```
<title>Index</title>
```

```
<style>
```

```
</style>
```

```
<script>
```

```
    window.addEventListener("load", () => {
```

```
        var a = window.navigator.onLine
```

```
        if(!a){
```

```
            alert("hey chevk internet connection")
```

```
        }
```

```
    });
```

```
$( function() {

    var dateToday = new Date();

    var dates = $( "#datepicker" ).datepicker({

        defaultDate: "1d",

        changeMonth: true,

        numberOfMonths: 1,

        minDate: dateToday,

        onSelect: function(selectedDate) {

            var option = this.id == "from" ? "minDate" : "maxDate",

                instance = $(this).data("datepicker"),

                date =

$.datepicker.parseDate(instance.settings.dateFormat ||
$.datepicker._defaults.dateFormat, selectedDate, instance.settings);

            dates.not(this).datepicker("option", option, date);

        },

        dateFormat: "yy-mm-dd"
```

```
});
```

```
} );
```

```
//setTimeout("location.reload(true)",10000);
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<div id="overlay">
```

```
<div class="hidden spinner-border text-light" role="status">
```

```
<span class="visually-hidden">Loading...</span>
```

```
</div>
```

```
</div>
```

```
<div class="position-fixed top-0 end-0 p-3" style="z-index: 5">
```

```
<div class="toast align-items-center text-white bg-primary border-0"
role="alert" aria-live="assertive" aria-atomic="true">
```

```
<div class="d-flex">
```

```
<div class="toast-body">
```

```
Hello, world! This is a toast message.
```

```
</div>
```

```
<button type="button" class="btn-close btn-close-white me-2 m-
```

auto" data-bs-dismiss="toast" aria-label="Close"></button>

</div>

</div>

</div>

<div class="div1">

<h1>Welcome</h1>

</div>

<div class="div2">

<p style="color: antiquewhite;">Hey! There Get alerts whenever your favorite webpages got updates{{posts}}</p>

</div>

<div class="container">

<form action="adds" method="POST">

{% csrf_token %}

<div class="row">

<div class="col t1">Enter Url</div>

<div class="col t2">{{form.url}}</div>

</div>

<div class="row">

<div class="col t1" onclick="hid()">Enter Email</div>

<div class="col t2">

{{form.mail}}

</div>

</div>

<div class="row">

<div class="col t1">Enter Date</div>

<div class="col t2">{{form.dt}}</div>

</div>

<div class="row">

<div class="col t1">Enter Time</div>

<div class="col t2"> {{form.tim}}</div>

</div>

<div class="row">

<div class="col t1">Enter sleep Time</div>

<div class="col t2">{{form.slep}}</div>

</div>

<!--

Enter url: <input type="url" id='url' name="urls"
placeholder="https://www.example.com" onfocusout='urlfn()' required>

Enter email: <input type="email" name="mail"
id="mail"placeholder="example@domain.com"
required>

enter date: <input type="text" name="date"
id="datepicker" placeholder="YYYY-MM-DD" required>

enter time: <input type="time" name="time" id="" required >

enter sleep time: <input type="text" name="slep" id=""
placeholder="in Seconds"required>-->

<div class="row">

<input type="submit" class="sub" onclick="func()">

</div>

</form>

```
<h1 id="hidden">helloss</h1>
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-b5kHyXgcpbZJO/tY9UI7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous"></script>
```

```
</body>
```

```
<script>
```

```
  //toast
```

```
  var toastElList = [].slice.call(document.querySelectorAll('.toast'))
```

```
  var toastList = toastElList.map(function (toastEl) {
```

```
    return new bootstrap.Toast(toastEl)
```

```
  });
```

```
  //
```

```
  //toast fetcher
```

```
  function
```

```
  show(){
```

```
    toastList.forEach(toast => toast.show());
```

```
  }
```

```
  var cprf= document.querySelector('[name=csrfmiddlewaretoken]').value;
```

```
$.ajaxSetup({

    beforeSend: function(xhr, settings) {

        if (!(/^http:.*/.test(settings.url) || /^https:.*/.test(settings.url))) {

            // Only send the token to relative URLs i.e. locally.

            xhr.setRequestHeader("X-CSRFToken", cprf );

        }

    }

});
```

```
//url fetcher
```

```
function urlfn(){

    var u = document.querySelector('form').elements[1].value;

    $('#toast-body').text("Validating Url in background ");

    show();

    console.log(cprf);

    console.log(u);

    $.ajax({

        headers: {'X-CSRFToken': cprf},

        url: '/ajax',

        type: 'post',
```

```
data:{

    'link':u

},

success: function(response){

    $(".toast-body").text(response.posts);

    if(response.l){

        $(".toast").removeClass('bg-primary');

        $('.toast').addClass('bg-success');

        setTimeout(function(){

            $('.toast').addClass("done");

        }, 2000);

    }

    else{

        $(".toast").removeClass('bg-primary');

        $('.toast').addClass('bg-danger');

    }

    show();

},
```

```
error: function(){
```

```
    $(".toast").removeClass('bg-primary');
```

```
    $('.toast').addClass('bg-danger');
```

```
    $(".toast-body").text("internal sever Error or Url error");
```

```
}
```

```
});
```

```
$(".toast").addClass('bg-primary');
```

```
$('.toast').removeClass('bg-danger');
```

```
$('.toast').removeClass('bg-success');
```

```
}
```

```
function hid(){
```

```
    $('.hidden').removeClass('hidden');
```

```
    document.getElementById("overlay").style.display = "block";
```

```
$(".toast").addClass('top-0 start-50');
```

```
$('.toast').removeClass('end-0');
```

```
$('.toast').text(" your request is being processed");
```

```
show();
```

```
}
```

```
function func(){  
  
    alert("hey, check your mail for updates");  
  
    document.querySelector("form")  
  
    hid();  
  
}  
  
</script>  
  
</html>
```