COURSE TITLE: INTERNET PROTOCOLS

INSTRUCTOR: DR. MUHAMMAD SHAHZAD

SHOP EASY

December 3, 2018

Group : 3

Aneesh Bhagwat (200199173)
Nishank Satish (200204876)
Shelmon Lewis (200204302)
Viplove Rakheja (200205854)

<u>Work Breakdown Chart:</u>

Component	**Component Weightage**	**Aneesh Bhagwat**	**Nishank Satish**	**Shelmon Lewis**	**Viplove Rakheja**
High-Level Design	*0.2*	25	35	20	20
Client Side and Web Page Development	*0.3*	10	10	35	45
Server-Side Code	*0.3*	30	10	35	25
Report and Analysis	*0.2*	35	55	5	5
Per Student Aggregate Contribution:		25*0.2 + 10*0.3 + 30*0.3+ 35*0.2 =24	35*0.2+ 10*0.3+ 10*0.3+ 55*0.2 = 24	20*0.2+ 35*0.3+ 35*0.3+ 5*0.2 =26	20*0.2+ 45*0.3+ 25*0.3+ 5*0.2 =26

I. INTRODUCTION

With the advent of digital shopping, the reliance on going to shops and buying products has steadily dropped and more and more people today are using Amazon, Walmart, Target online stores for their day to day shopping. Our project aims at making online shopping easier and more economical by comparing results from multiple online vendors and providing the customer with a consolidated search result. In most of the cases, the customer does not care about the online vendor and instead wants the cheapest possible deals for the product. Our project would reduce the hassle of having to check multiple online vendors before deciding to order a particular product.

II. DESIGN

In our project, we have shown how we can compare results for a particular product on Amazon and Walmart and it will return results for the best prices for a particular item on both the websites.
The various parts of our project design are:

- *User Input:*

  The User input is the product user wants to search for. This is given in the form of a string to the GUI which in our case is the Webpage.

- *Webpage:*
  This is our GUI (Graphical User Interface) which basically takes user input for the product user wants to be searched. It also shows the end result which gives us a comparison on both websites (Amazon and Walmart in our case).

- *Server Side:*
  This is the vendor side (Amazon and Walmart, in our case). This receives the product request from the user as a string literal and searches its database for the product. Their servers have already been optimized to show the best possible results for any entered user product so we don't touch this part. These vendors return the results to us through the HTML Script of the Webpage.

We have shown below what our project does in very basic terms.

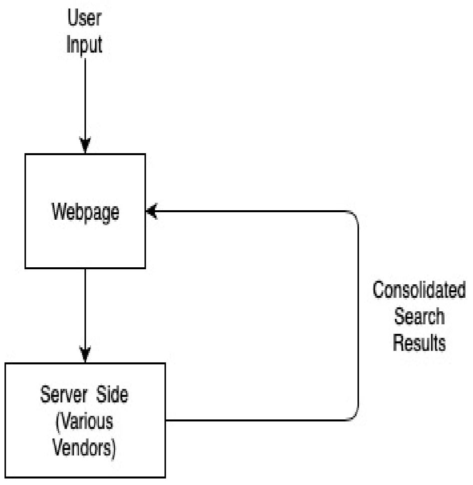


Fig. 1: *Functional Block Diagram*

- *Search Results:*
  This is the final result we get from the two vendors which helps us decide which vendor to choose for buying the product. The result thus obtained is in terms of the HTML script. And based on the HTML script, the user searched products are displayed on our webpage.

We have implemented an easy design so that it is possible for any user to effectively use our code/ steps and add functionalities as per their convenience.

Our webpage is easy to understand and does not have too many complications, thus making it easier even for people not familiar with computers to use it.

The search results don't just return 1 product since people often want to have options for a single product too. So we return a few options using Amazon and Walmart's sorting algorithms to effectively give 3-4 products making it easier for the customer to decide the correct product.

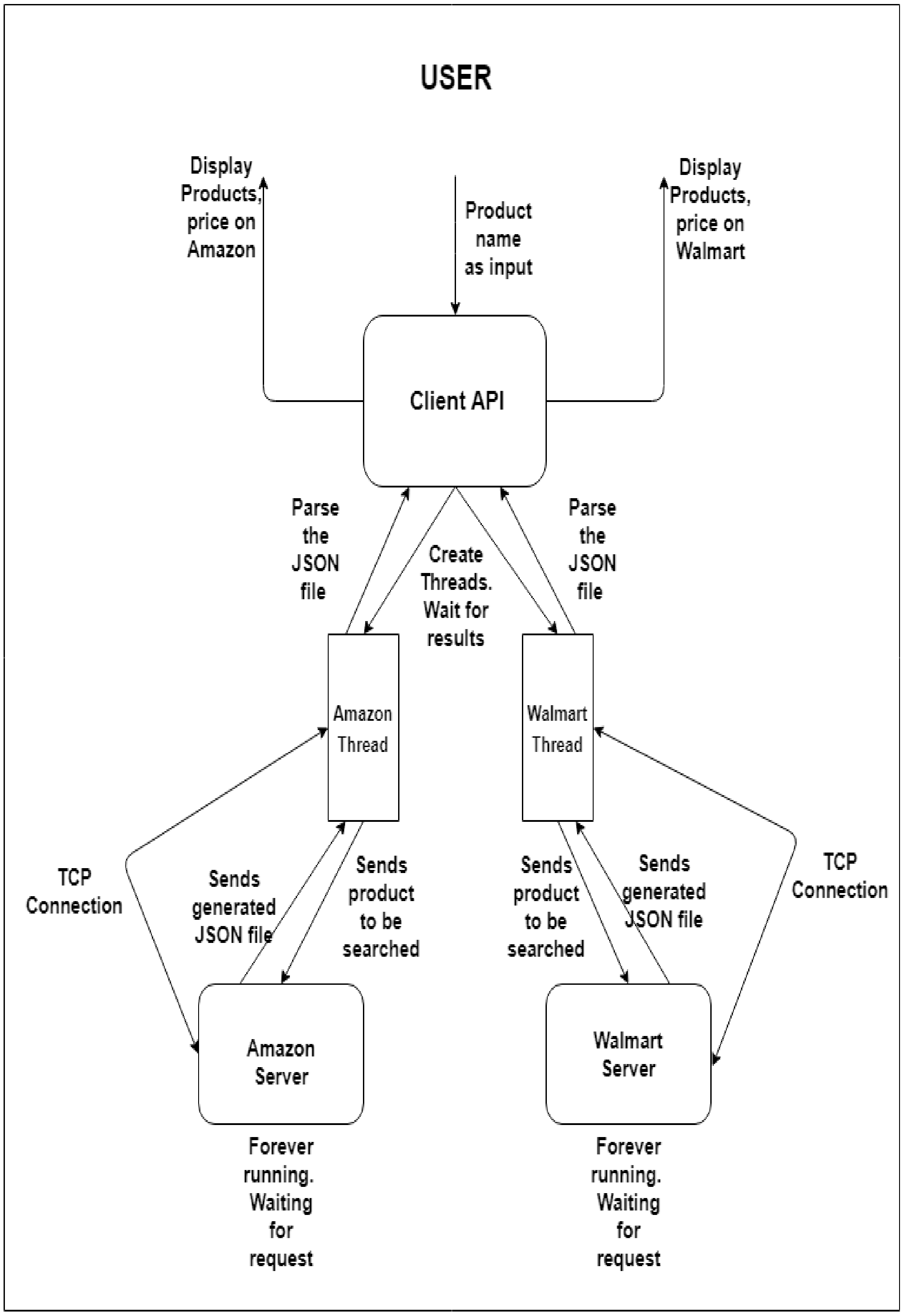

USER
Display Products, price on Amazon
Product name as input
Display Products, price on Walmart
Client API
Parse the JSON file
Create Threads. Wait for results
Parse the JSON file
Amazon Thread
Walmart Thread
TCP Connection
Sends generated JSON file
Sends product to be searched
Sends product to be searched
Sends generated JSON file
TCP Connection
Amazon Server
Walmart Server
Forever running. Waiting for request
Forever running. Waiting for request

Fig. 2: *Flowchart*

III. IMPLEMENTATION

Our Project works in two parts:

- Client Side
- Server Side (2 different Machines)

The client-side implementation is basically a webpage that accepts the product information as string inputs from the user. This string is basically the product whose price we want to search on both Amazon and Walmart servers to help us decide the best option. The Webpage is designed using html and allows the user to type in the product name and press "*Search*"

The Webpage is created on the local host with port number "*8000*".

The product name typed on the webpage is sent to the client backend code as a string. The client backend code which is in python, creates two threads, one for Amazon and the other for Walmart in parallel. We give the same string value (user product) as input to both the threads. These two threads are used to setup TCP connection with the servers (Amazon and Walmart servers) so that both the severs receive the request at the same time.

The Amazon and Walmart server files need to be running before the requests are sent. The two servers run continuously waiting for a request on separate terminals. The servers side code accepts inputs from clients and searches for the appropriate webpage from the respective website (Amazon or Walmart). It then parses the product ID's and the prices for the top 4-5 products that match the search keyword. Using the product ID's, the links for each product is constructed and also the respective prices are parsed from the webpage and added into a data structure. All these details for all the products that best match the search keyword are written into a .JSON file. Both of the servers generate its own .JSON file. Hence, once the threads created, they establish connection with its respective server and sends the input string and waits for the server to process and return the generated .JSON file. Once both the .JSON files are received by the backend python code using the already established TCP connection, the contents in them are displayed so that the user can look at the prices of the top few best matching products in Walmart and Amazon and can make an informed decision as to which site to buy the product from and thus from the webpage itself the user can navigate to the site to order the product.

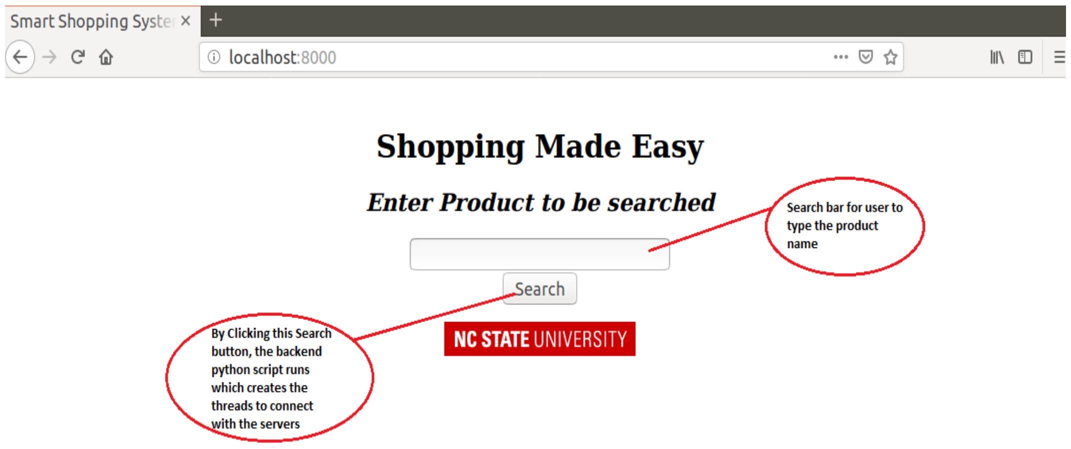

Smart Shopping Syste
localhost:8000
Shopping Made Easy
Enter Product to be searched
Search
NC STATE UNIVERSITY
Search bar for user to type the product name
By Clicking this Search button, the backend python script runs which creates the threads to connect with the servers

Fig. 3: *Webpage Layout*

IV. EXPERIMENTS

As mentioned in the Implementation section, the project consists of three separate units. We experimented with a simple product. The experiment setup consists of the following steps :

A. The Amazon server needs to be running before any requests come along. The Amazon server code is run on one terminal separately that continuously keeps running waiting for any client requests.
B. The Walmart server needs to be running before any requests come along. The Walmart server code is run on one terminal separately that continuously keeps running waiting for any client requests.
C. In order to setup up the application, the user first needs to open a terminal and try to setup the application by typing the command : python -m CGIHTTPServer. Once the connection is made, then the user can open any web application like google chrome or Mozilla .etc. and type the url as : localhost:8000 and prese enter so that the application is setup successfully and the user will be able to see the HTML page that is shown in figure 3.
D. In order to make the search, the client needs to type in the items (a particular item if needs more appropriate results) in the search bar and then press the search button.
E. Once the 'Search' keyword is pressed, the backend python code of the API runs with the inputted data, establishes a TCP connection with the Amazon and Walmart servers running on the other terminals and sends the input string.
F. Once the connection is setup with both the servers, then Servers tries to fetch the product details from the Amazon and Walmart servers and puts the results in their own .JSON files, which is then returned to the Client over the TCP connection.
G. Once the .JSON files are received, the client side backend python code parses the received .JSON files and fetching the product name, price and its URL which is then printed in the new tab of the web browser for both Amazon and the Walmart website.

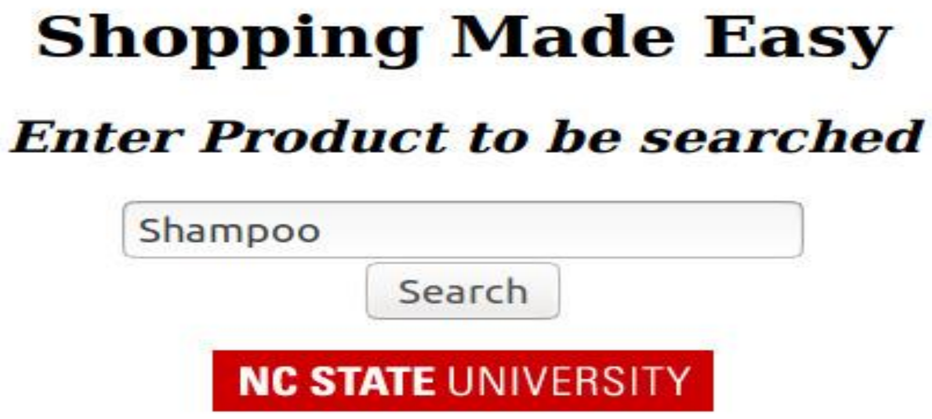
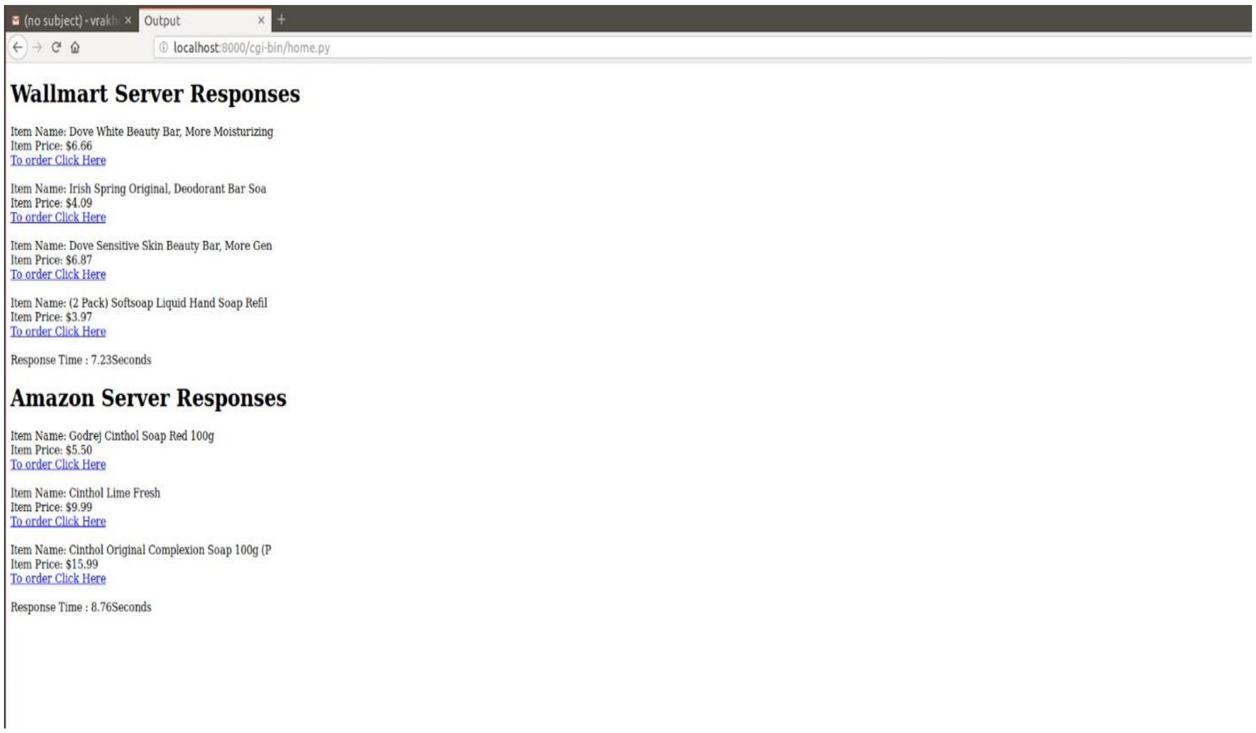


Fig. 4: *Shampoo Search*



Wallmart Server Responses

Item Name: Dove White Beauty Bar, More Moisturizing
Item Price: $6.66
To order Click Here

Item Name: Irish Spring Original, Deodorant Bar Soa
Item Price: $4.09
To order Click Here

Item Name: Dove Sensitive Skin Beauty Bar, More Gen
Item Price: $6.87
To order Click Here

Item Name: (2 Pack) Softsoap Liquid Hand Soap Refil
Item Price: $3.97
To order Click Here

Response Time : 7.23Seconds

Amazon Server Responses

Item Name: Godrej Cinthol Soap Red 100g
Item Price: $5.50
To order Click Here

Item Name: Cinthol Lime Fresh
Item Price: $9.99
To order Click Here

Item Name: Cinthol Original Complexion Soap 100g (P
Item Price: $15.99
To order Click Here

Response Time : 8.76Seconds

Fig. 5: *Shampoo Search Results*

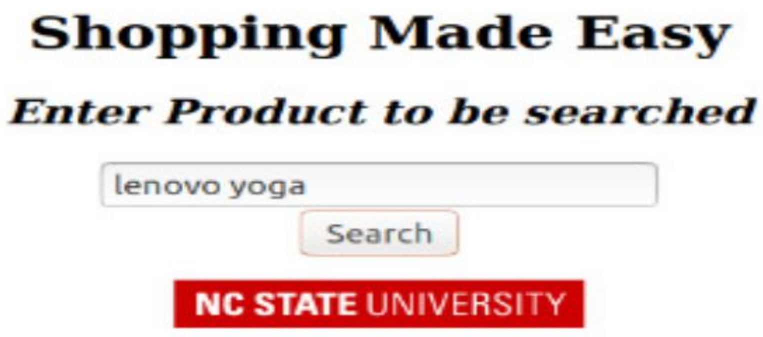
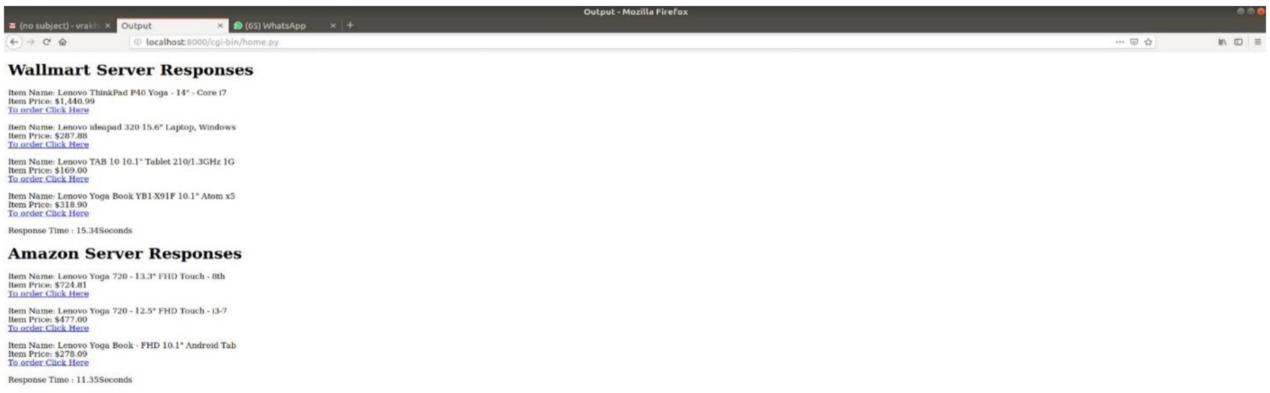


Fig. 6: *Lenovo Yoga Search*



```
Wallmart Server Responses

Item Name: Lenovo ThinkPad P40 Yoga - 14" - Core i7
Item Price: $1,440.99
To order Click Here

Item Name: Lenovo ideapad 320 15.6" Laptop, Windows
Item Price: $287.88
To order Click Here

Item Name: Lenovo TAB 10 10.1" Tablet 210/1.3GHz 1G
Item Price: $169.00
To order Click Here

Item Name: Lenovo Yoga Book YB1-X91F 10.1" Atom x5
Item Price: $318.90
To order Click Here

Response Time : 15.34Seconds

Amazon Server Responses

Item Name: Lenovo Yoga 720 - 13.3" FHD Touch - 8th
Item Price: $724.81
To order Click Here

Item Name: Lenovo Yoga 720 - 12.5" FHD Touch - i3-7
Item Price: $477.00
To order Click Here

Item Name: Lenovo Yoga Book - FHD 10.1" Android Tab
Item Price: $278.09
To order Click Here

Response Time : 11.35Seconds
```

Fig. 7: *Lenovo Yoga Search Results*

V. <u>RESULTS AND FUTURE SCOPE</u>

As we saw above, we conducted experiments using our project. We searched the webpage for shampoo, cinthol soap, Lenovo yoga and through the results, we see that the project works efficiently. However, we also noticed that it takes a small amount of time for results to be printed once the user inputs the search string on the webpage. This is because it takes a large amount of time to transfer the .json file over TCP connection from server to the client. Another reason for this is that the client waits for both the .json files before printing the results to the html file.

Since we have achieved a very high efficiency rate with our project, at this time latency is the only cause for future work. Another aspect where we could improve in the future is by adding more websites although the code for that would largely remain consistent. Also, if needed the web results can be made more fancier by displaying the image along with the product details. Also, since the we page can only be displayed when both the threads have done computing the values, because of which the client needs to wait till both the servers have responded with their corresponding strings. This can be improved in future by creating two separate python scripts along which will be invoked when the search button is pressed and will be displaying the results as soon as available on different webpages (But that wont be much of easy shopping as the user will still have to navigate on multiple tabs).

VI. <u>REFERENCES</u>

(1) https://www.pfsense.org/about-pfsense/
(2) https://www.w3schools.com/php/php_intro.asp
(3) https://dev.mysql.com/doc/connector-python/en/
(4) https://www.promptcloud.com/blog/how-to-buil d-price-comparison-website
(5) https://codereview.stackexchange.com/questions/53749/walmart-scraping-crawling-code
(6) https://docs.aws.amazon.com/polly/latest/dg/examples-python.html
(7) https://stackoverflow.com/questions/5607551/how-to-urlencode-a-querystring-in-python
(8) https://ieeexplore-ieee-org.prox.lib.ncsu.edu/document/744915