**PRICE COMPARISON**

A Synopsis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Samriddhi Mishra

Sahil Saxena

Suchita Yadav



COMPUTER SCIENCE & ENGINEERING

KAMLA NEHRU INSTITUTE OF TECHNOLOGY,SULTANPUR

2019-2020

**INTRODUCTION**

ABOUT PROJECT ( ACKNOWLEDGMENT)

"Compares the price of the product entered by the user from e-commerce sites Paytm and Flipkart."

Its a price comparison based project that compares the prices of searched products and it displays the various prices offered by different online retailers

This project helps you to scrap prices from ecommerce sites like flipkart and paytm and stores it in a django database file For Running this project you need to have

\*Django

\*Python, selenium and other libraries that are mentioned in the code installed in your system.

Technology used:-

\*Selenium

for accessing data from the websites

\*Django

for front end and maintaining database

\*Chrome Web Driver

for manipulating websites and control flow

**Motivation**

Price comparison websites are becoming more popular since Ecommerce revolution. The data and values provide by the price comparison sites helps consumers to save money while shopping online. The ability to check the price of products brings convenience and saves a lot of time. Price comparison is done when the price of the same product is compared in different outlets. There are many online shopping sites which are being used to shop various products online. A particular product has different prices on every different shopping site. We need to check manually the price of the product on every site, which causes inconvenience and consumes a lot of time. There are many price comparison sites which provides products comparison, but there is no site which compares the price of wish list products. Hence, we have proposed a project named Wish list Products Price comparison website which compares the prices of only interested products which user intents to buy. This price comparison website for wish list products will help to compare the price from various e-commerce websites. This system has an advance feature of sending notification when the price changes of a particular product which has been added in the wish list. This Price comparison site is extremely helpful for frequent online shoppers to check prices on different online stores in one place. This system will show you the product prices from different retailers to show you where to buy the product at affordable price. Whenever the price of the product changes it sends notification to the user.

**TECHNICAL DETAILS**

SOFTWARE PARTS

LANGUAGE USED – PYTHON 3.4.3

Python 3.4. 3 (64-bit) Python 64-bit is a dynamic object-oriented programming language that can be used for many kinds of software development. ... Python 64-bit is distributed under an OSI-approved open source license that makes it free to use, even for commercial products

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.

As its name suggests, HTML is a Markup Language which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

FRONT END

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it 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's free and open source. Ridiculously fast

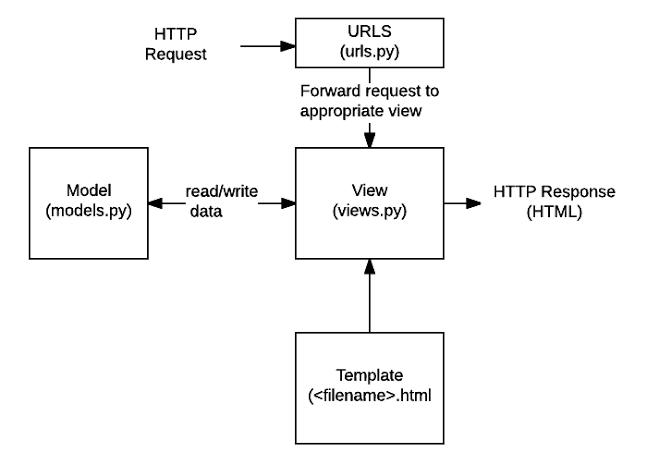
BACK END

Selenium is a portable framework for testing web applications. Selenium provides a playback tool for authoring functional tests without the need to learn a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. The tests can then run against most modern web browsers. Selenium runs on Windows, Linux, and macOS. It is open-source software released under the Apache License 2.0.

SITES NAVIGATOR

ChromeDriver. WebDriver is an open source tool for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user input, JavaScript execution, and more. ChromeDriver is a standalone server that implements the W3C WebDriver standard.

**DFD(Data Flow Diagram):-**

****

**Further Details of Project:-**

The commercial sites used for comparing the results of the search are ‘Flipkart’ and ‘PayTm’. The index page of the interface takes any keyword related to the items available on the two sites, whose prices are to be compared. This keyword is used in selenium code to pop up a new chrome window using the chrome driver. In this window the keyword is searched on both sites and the result after the search appears on the screen, using which we extract all the item names and their respective prices to generate four lists of item names and their prices , a pair for each site. These lists are used to create the records of the information depending on the models created in models.py file. These records are shown on the screen in form of the table using HTML and CSS.

Index page , Accepts the keyword related to search:-

###

<!DOCTYPE **html**>  
<**html lang="en"**>

<**head**>  
 {% load static %}  
 <**meta charset="UTF-8"**>  
 <**title**>COMPARE</**title**>  
</**head**>

<**body style="**text-align:center;background-image:url({%static 'images/pic1.jpg'%})**"**>

<**form action="{% url 'createtable' %}" method="post"**>  
 <**h1 style="**font-family:forte;font-size:100px;text-align:center;color:white;text-shadow:3px 3px rgba(0,0,0,0.4);**"**>Price Comparison</**h1**>  
 <**h2 style="**font-family:forte;font-size:60px;text-align:center;color:white;text-shadow:2px 2px rgba(0,0,0,0.5);**"**>Begin From Here.....</**h2**>  
 {% csrf\_token %}  
 <**input type="text" name ="product" style="**width:400px;height:30px;border:0px;padding:5px;font-size:18px;**"**/>  
 <**input type="submit" value="Compare" style="**width:100px;height:32px;font-family:forte;font-size:20px;background-color:rgba(255,255,255,0.6);border:0px;**"**/>  
</**form**>

</**body**>

</**html**>

###

Logical code written in ‘selenium’ to generate the lists:-

###

**import** selenium  
**from** selenium **import** webdriver  
**from** selenium.webdriver.common.keys **import** Keys  
**from** .models **import** amfp  
  
**def** AmazonItemName(item):  
 amazon\_item = item.split()  
 amz = **''  
 for** i **in** range(len(amazon\_item)):  
 amz = amz + amazon\_item[i] + **'%20'** amz = amz[:-1]  
 **return** amz  
  
  
**def** Sort\_Tuple(tup):  
 lst = len(tup)  
 **for** i **in** range(0, lst):  
 **for** j **in** range(0, lst - i - 1):  
 a=((tup[j][1]).strip()).replace(**','**,**''**)  
 b=((tup[j+1][1]).strip()).replace(**','**,**''**)  
 **if** (a > b):  
 temp = tup[j]  
 tup[j] = tup[j + 1]  
 tup[j + 1] = temp  
 **return** tup  
  
**def** compare(item):  
 driver = webdriver.Chrome(  
 executable\_path=**r"C:\Users\samriddhi mishra\Documents\Projects\price\_comparison\comp\_env\price\_comparison\myapp\chromedriver.exe"**)  
 *# FLIPKART FROM HERE* driver.get(**'https://www.flipkart.com'**)  
 driver.get(  
 **'https://www.flipkart.com/search?q={}&otracker=search&otracker1=search&marketplace=FLIPKART&as-show=on&as=off'**.format(  
 item))  
 flipkart\_price = driver.find\_elements\_by\_xpath(  
 **'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_1vC4OE", " " ))]'**)  
 fp = []  
 **for** i **in** flipkart\_price:  
 fp.append(i.text)  
 flipkart\_name = driver.find\_elements\_by\_xpath(  
 **'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_3wU53n", " " ))]'**)  
 **if** (len(flipkart\_name) == 0):  
 flipkart\_name = driver.find\_elements\_by\_xpath(  
 **'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_2B\_pmu", " " ))]'**)  
 **if** (len(flipkart\_name) == 0):  
 flipkart\_name = driver.find\_elements\_by\_xpath(  
 **'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_2cLu-l", " " ))]'**)  
  
 fn = []  
 **for** i **in** flipkart\_name:  
 fn.append(i.text)  
  
 *# AMAZON FROM HERE* amazon\_item = AmazonItemName(item)  
 *# name = driver.find\_element\_by\_xpath('//input');  
 # name.sendKeys("{}".format(item));  
 # WebElement login = driver.findElement(By.id("SubmitLogin"));  
 # login.click()* driver.get(**'https://paytm.com/shop/search?q={}&from=organic&child\_site\_id=1&site\_id=1'**.format(amazon\_item))  
 amazon\_item = AmazonItemName(item)  
 amazon\_price = driver.find\_elements\_by\_xpath(**'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_1kMS", " " ))]'**)  
  
 amazon\_name = driver.find\_elements\_by\_xpath(**'//\*[contains(concat( " ", @class, " " ), concat( " ", "\_2apC", " " ))]'**)  
  
 ap = []  
 **for** i **in** amazon\_price:  
 ap.append(i.text)  
 an = []  
 **for** i **in** amazon\_name:  
 an.append(i.text)  
**if**(len(fp)<len(fn)):  
 m=len(fp)  
 **else**:  
 m=len(fn)  
 **if** (len(ap) < len(an)):  
 n = len(ap)  
 **else**:  
 n = len(an)  
 ff=[(fn[i], fp[i]) **for** i **in** range(0, m)]  
 aa=[(an[i], ap[i]) **for** i **in** range(0, n)]  
 aa=Sort\_Tuple(aa)  
 ff=Sort\_Tuple(ff)  
   
 **for** i **in** range(n):  
 an[i]=aa[i][0]  
 ap[i] = aa[i][1]  
 **for** i **in** range(m):  
 fn[i]=ff[i][0]  
 fp[i] = ff[i][1]  
  
 **if** (n < m):  
 m = n  
 **else**:  
 m = m  
  
 **for** i **in** range(m):  
 f = amfp(aprod=an[i], aprice=ap[i], fprod=fn[i], fprice=fp[i])  
 f.save()  
  
###

The models.py file creating model to store the information, in form of records:-

###

**from** django.db **import** models  
  
*# Create your models here.***class** amfp(models.Model):  
 aprod = models.CharField(max\_length=100)  
 aprice = models.CharField(max\_length=100)  
 fprod = models.CharField(max\_length=100)  
 fprice = models.CharField(max\_length=100)

###

Showing the result in form of Table on the screen:-

###

<!DOCTYPE **html**>  
<**html lang="en"**>

<**head**>  
 {% load static %}  
 <**meta charset="UTF-8"**>  
 <**title**>Event</**title**>  
</**head**>

<**body style="**text-align:center;background-image:url({% static 'images/pic4.jpg' %});**"**>

<**div id="outer" style="**min-height:750px;padding:10px;**"**>  
<**table style="**width:90%;margin:0 auto;background-color:rgba(0,0,0,0.5);padding:2px;color:white;**"**>  
 <**tr style="**font-family:Forte;color:white;text-shadow:1px 1px rgba(0,0,0,0.3);font-size:30px;height:30px;**"**>  
 <**th style="**border-right:1px solid white;border-bottom:1px solid white;**"**>Flipkart\_Prod</**th**>  
 <**th style="**border-right:1px solid white;border-bottom:1px solid white;**"**>Flipkart\_Price</**th**>  
 <**th style="**border-right:1px solid white;border-bottom:1px solid white;**"**>PayTm\_Prod</**th**>  
 <**th style="**border-bottom:1px solid white;**"**>PayTm\_Price</**th**>  
 </**tr**>  
 {% for ff in f %}  
 <**tr style="**height:50px;padding:2px;**"**>  
 <**td style="**border-right:1px solid white;width:50%;**"**>{{ff.fprod}}</**td**>  
 <**td style="**border-right:1px solid white;**"**>{{ff.fprice}}</**td**>  
 <**td style="**border-right:1px solid white;width:40%;**"**>{{ff.aprod}}</**td**>  
 <**td**>{{ff.aprice}}</**td**>  
 </**tr**>  
  
 {% endfor %}  
  
</**table**>  
</**div**>  
  
</**body**>

</**html**>

###

**Plan Of Work:-**

AUGUST :- Formation of the team and selection of the project that includes work related selenium and Django Framework.

SEPTEMBER:-Worked on the code (using selenium) to fetch the data from the respective sites by searching a keyword related to the product.

OCTOBER:-Worked on Django Framework which includes storing of the information fetched form the sites and displaying it on the screen.

NOVEMBER:-Worked on the appearance of the final result which includes designing of the interface.

**Role of Project Members:-**

Samriddhi Mishra :- Django Framework

Sahil Saxena :- Selenium

Suchita Yadav :- HTML,CSS