E-COMMERCE PRODUCT PRICE TRACKER

[1]Dr. G Madhusudhan, [2]Nitin Gopalakrishna Bhat, [2]Sahana Venkatraman Patgar, [2]Chandan N A, [2]Bharath S V
[1]Assistant. Professor, [2]Student
[1,2]Computer Science and Engineering
[1,2]JSS Science and Technology University, Mysuru, India.

Abstract : Online purchasing is gradually displacing traditional shopping methods in every manner. As a result of pandemic, it has gained traction and has become the new normal. From shoes to food, everything is now available on E-cart. People like to purchase online since there are so many possibilities in each area. Users can notice the price differences between websites and, as a result, the majority people will choose the service with the lowest price. So, in order to stay ahead of the competition, business minds are constantly devising new strategies in order to maximize the profit from each sale. They come up with enticing offers in order to attract more clients. In the E-commerce industry, dynamic pricing is currently the most popular method. Consumer may find it challenging to keep up with the pricing changes that occur every 10 minutes on average. They give the illusion that the goods are on sale on that particular by doing so. We offer a basic price tracker application in this paper that tracks the price of a product and notifies the customer when the price reaches the desired level. It will get the customer pricing from the browser and act in accordance with the customer's wishes.

Index Terms **- E-commerce, web crawler, dynamic pricing, threshold price, mail alert**

I INTRODUCTION

Modern consumers are becoming increasingly aware of the prospects of alternative pricing schemes, which has made their buying decisions more sophisticated, thanks to significant improvements in the Electronic Commerce sector. According to recent research, internet customers are increasingly waiting for the best price offers before making purchases. This research seeks to track down prices and determine whether the offered offers are genuine savings or simply a marketing ploy to encourage sales. Many academics have proposed ideas to automate the bargaining process, but they are inaccurate. The cheapest price in history cannot be considered the best value for a product. The product's true worth fluctuates from day to day. As a result, we give the user the option of setting a price limit for the product. The user will be aware of the product's current price and will set the price based on that information. As a result, instead of waiting eternally for the lowest price, the application will know what price to wait for. Using HTML, CSS, and Bootstrap, we designed an E-cart website. Then we used GoDADDY to host it, and then we used C# and Java to construct an application that scrapes the pricing from the website and notifies the consumer

II LITERATURE SURVEY:

Quanyin Zhu and Sunqun Cao [1] published a study on Web Mining-based Price Forecasting without Complete Data. Web mining can be used to obtain more and more data from many sources, such as e-supermarkets and e-commerce. This paper presents a case study of pricing extraction from internet mobile phone sales. It is investigated how to extract the commodity price in order to obtain the price trend utilizing incomplete data. The forecast algorithm is well described. The PHP routines are intended for implementing the pricing extraction. Experiment confirms its effectiveness and demonstrates that price evolution is important and valuable for online shopkeepers.
Inference: We got the idea that the data extracted will be incomplete, we can use it for our project by taking the remaining required data from the user itself.

A work on Dynamic pricing, different schemes, related research survey, and evaluation was proposed by Ahmed Z. Gabar and Ahmed A. Helal [2]. The dynamic pricing concept is highlighted in this study by surveying its many schemes and related research effort in chronological order. The article concludes with a detailed subjective comparison of dynamic pricing schemes in order to assess their efficacy in various smart grid applications. The key value of the resulting comparison is that it may be used as a baseline for determining the most appropriate dynamic pricing scheme for both utilities and consumers, while taking into account current and future utility infrastructure and operation conditions.
Inference: We understand the idea that we can't have a solid base condition as the infrastructure and market conditions changes regularly. We kept it in our mind while building our project.

Jianxia Chen and Ri Huang [3] proposed a price comparison system based on IOT. The article provides an online product price comparison system that displays all possible product costs for customers. The suggested system, in particular, develops a multithreaded crawler for web information crawling and employs IOT, a prominent full-text search library, for data indexing and retrieval. The trial results showed that the system enhances consumers' shopping efficiency in a flexible and advanced manner.
Inference: It gave us idea that the crawler can be multithreaded instead of just a single thread so that the crawling process continuously run in background.

Nitis Monburinon and Prajak Chertchom [4] presented a paper on utilizing regression models to predict prices. They compared the performance of regression models based on supervised machine learning models for this investigation. Each model was trained using data from a German e-commerce website on the used automobile market. As consequence, with a mean absolute error (MSE) of 3D 0.28, gradient enhanced regression trees provide the best performance. Random forest regression with MSE =3D 0.35 and multiple linear regression with MSE =3D 0.55 were the next steps, followed by random forest regression with MSE =3D 0.35 and multiple linear regression with MSE =3D 0.55, respectively.

Inference: We got to learn that we have to provide the system with the predicted or threshold price. Since we are dealing with discounts, we can't rely on regression model which doesn't consider user's affordability. We need to get that data from user only**.**

III PROPOSED SYSTEM AND ARCHITECTURE:

PROPOSED SYSTEM:

i. Each product will have its original price and discounted price at that instance.
ii. The product can be handled admin via remote location.
iii. The user will add the product to the cart and will set the threshold price.
iv. Set the time interval for fetching data from the website.
v. Dashboard starts fetching prices once the background application is started.
vi. If the price is below threshold, alert will be sent to user.
vii. If the user didn't buy the product, then the discounted price is set as new threshold.
viii. Continue fetching until the user stops the thread.

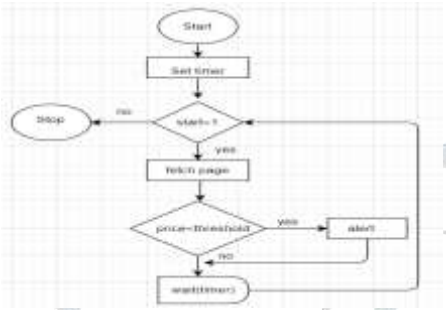

Fig 3.1: Flowchart:

ARCHITECTURE:

We have deployed the website using 3-tier architecture. This separates applications into three logical and physical computing tiers. The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team and can be updated or scaled as needed without impacting the other tiers.

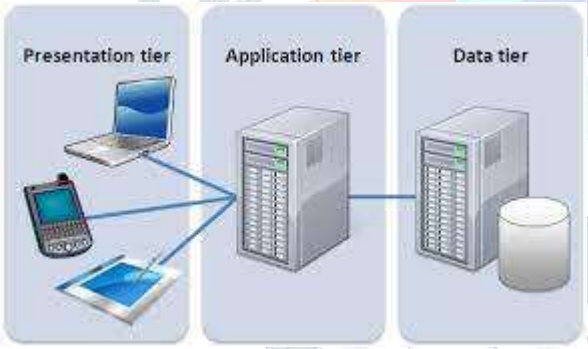

Fig 3.2: 3-tier architecture

PRESENTATION TIER:

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser. Web presentation tier is developed using HTML, CSS and Bootstrap.

APPLICATION TIER:

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier. We have used .NET for this tier implementation.

DATA TIER:

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB or MongoDB. We have developed this tier using MySQL.

IV SYSTEM IMPLEMENTATION:

Designing a website

At the client's end, two pages must be developed. They are

i. Product browsing page.
ii. A shopping cart where you can add things.

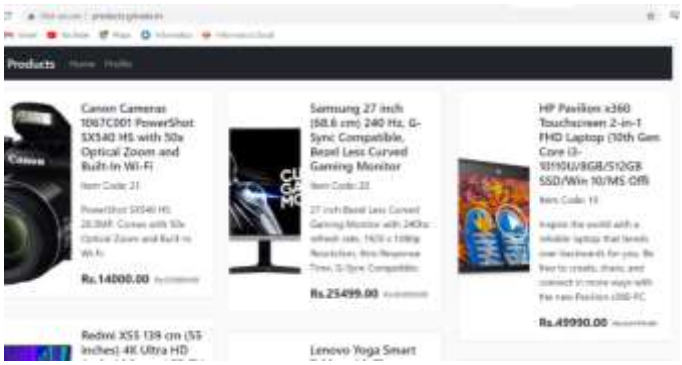

Fig -4.1: Browsing Page

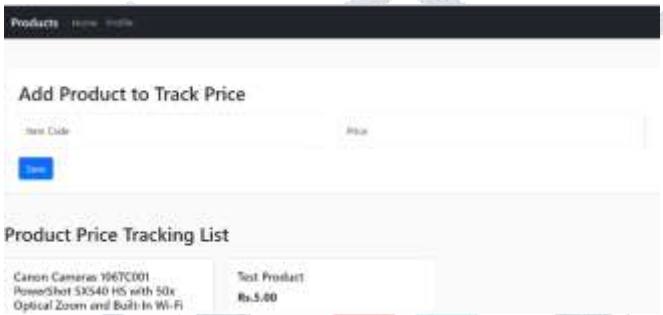

Fig -4.2: Cart for the user

On the server side, we'll need to make an

i. Page for website item administration.

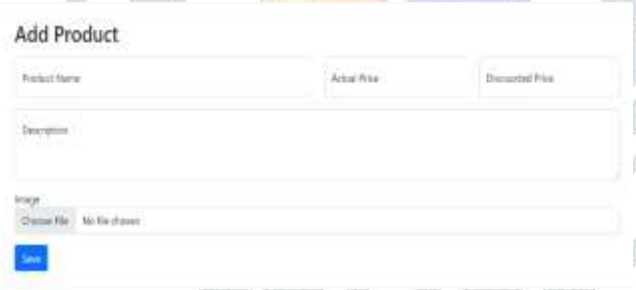

Fig – 4.3: Administration page

Creating database for the website

The database is created with MySQL. In the beginning, Microsoft SQL server is used to create the database on the local PC. We constructed tables to keep track of product and customer information.

Establishing connection between the database and the website

We need to set up an ODBC connection before we can get and store data in the database. We used the Connection string function to make the connection. We first got the connection string by using ODBC data sources. The connection key was then obtained by creating a new data source. Finally, using the DSN file we saved in the previous phase, we were able to extract the field information we need.

Hosting the website

GoDaDDY was used to host the website. To begin, we prepared a zip file containing all of the code that had been developed and tested locally. Then we uploaded the zip file to the GoDaDDY space to which we had subscribed. Then, using WordPress, we built a separate database at the GoDaDDY and utilized HeidiSQL[5] to control the database from any remote place.

Creating a web crawler application

We designed a web crawler application that would operate in the background indefinitely in order to retrieve data from the website in real time. For crawling data from the website, HTML agility pack [6], an open-source software, is employed. We took the website's page source and reduced it down to just the discounted price. The crawler will begin crawling the data from the page after we hit the start button. We can set the interval in which the crawler gathers data with tracker delay timeout. The data will be displayed in the Web crawled data area once it has been fetched. If the price falls below the

threshold, the user id is identified, and an email notice is sent to the person who added the data to his or her cart. For the mail alerts, we used 'System.Net.Mail', which contains the classes for sending mail to SMTP servers.

Fig – 4.4: Web crawler dashboard

V RESULT:

If the current price of the product falls below the customer's threshold price, the user will receive an e-mail alert message and a window will appear indicating that the alert has been delivered.

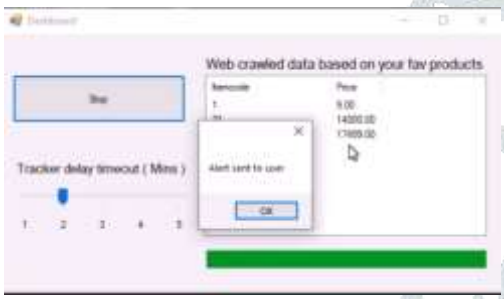

Fig -5.1: Dashboard

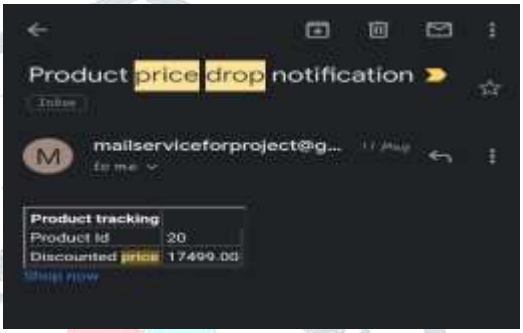

Fig -5.2: Mail alert

VI CONCLUSION AND FUTURE ENHANCEMENTS:

The idea in this article is a real-time program that tracks the website's dynamic price behavior. The customer's pricing will be compared to the current price by crawling the real time price from the website, and if the condition is met, an alert will be issued to the user telling them of the new price and a link to the product on the website.
By hosting the crawler application on the cloud and making it available to all users, the job may be expanded to a larger area. This solution eliminates the need for the crawler to be installed on the local machine. If we use it to track the prices of other websites, it can also be utilized as third-party software.

VII ACKNOWLEDGEMENT:

We'd like to express our gratitude to JSS STU Mysore's Department of Computer Science and Engineering for supporting us throughout our engineering journey. We are grateful to our guides, Dr. G Madhusudhan, Dept. of Computer Science and Engineering, JSS STU, and Dr. H C Vijayalakshmi, Dept. of Computer Science and Engineering, JSS STU, for their patience and ideas throughout our project while enabling us to work in our own way

VIII REFERENCES:

[1] Research on the price Forecast without Complete data based on Web Mining. Published in 10th international Symposium on Distributed Computing and Applications to, Business Engineering and Science (2010).

[2] Dynamic pricing; different schemes, related research survey and evaluation. Published on 9th International Renewable Energy Congress (IREC) in the year 2018.

[3] A price comparison system based on IOT. Published on 8th International Conference on Computer Science and Education (2013).

[4] Prediction of prices for using regression models. Published on 5th International Conference on Business and Industrial Research (2018).

[5] Technical help document – HeidiSQL https://www.heidisql.com/help.php

[6] Documentation | HTML agility pack https://html-agility-pack.net/documentation