INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

E-Commerce Product Price Tracker

[1]Manasa Katikala, [2]Pranava Sai Yasarla, [3]Sudheer Karre, [4]Sarantej Balabadhra,[5]Mrs Ch Pavani

[1]Student, [2]Student, [3]Student , [4]Student, [5]Faculty Guide
[1]Computer Science and Engineering,
[1]SRK Institute Of Technology, Vijayawada, India

Abstract: Online retail platforms frequently adjust product prices in response to market conditions, making continuous manual price tracking impractical for consumers. This paper presents an E-Commerce Product Price Tracker, an automated system designed to monitor product prices across online shopping platforms and notify users when prices reach user-defined thresholds. The system allows users to register products through an intuitive web interface, configure target prices, and view historical price trends through graphical visualizations. Product price data is periodically collected using automated extraction techniques and stored for long-term analysis. When a monitored price falls below the specified value, real-time alerts are generated to notify the user. By automating price monitoring and providing actionable pricing insights, the proposed solution reduces manual effort, prevents missed purchasing opportunities, and supports informed, cost-effective buying decisions.

IndexTerms **- E-commerce, Price Tracking, Price Monitoring, Web Scraping, Automated Alerts, Price Prediction, Data Visualization, Consumer Decision Support, Online Shopping, Price History Analysis**

I. INTRODUCTION

The rapid expansion of e-commerce platforms has significantly transformed modern consumer purchasing behaviour. Online retailers such as Amazon, Flipkart, and Myntra frequently modify product prices based on factors including market demand, competitor pricing strategies, seasonal trends, and promotional campaigns. While such dynamic pricing models benefit sellers, they often create challenges for consumers seeking to purchase products at the most economical price.

Manually tracking and comparing prices across multiple e-commerce platforms is both time-consuming and inefficient, particularly when monitoring a large number of products. Existing price comparison websites provide limited functionality and often lack real-time updates, historical price insights, and personalized alert mechanisms. Consequently, consumers may miss optimal purchasing opportunities due to delayed or incomplete price information.

To address these challenges, this paper presents an intelligent E-Commerce Product Price Tracker designed to automate the process of online price monitoring. The proposed system enables users to submit product URLs along with preferred target prices. Using automated price extraction techniques and data analytics, the system continuously monitors price variations, stores historical pricing data, and generates notifications when prices fall below user-defined thresholds.

The proposed solution aims to reduce manual effort while empowering consumers with timely and accurate price intelligence. By providing historical price analysis and personalized alerts within a secure and scalable web-based platform, the system supports informed, cost-effective purchasing decisions and enhances transparency in the online shopping ecosystem.

II. NEED OF THE STUDY.

Ease of use is a critical design objective of the proposed E-Commerce Product Price Tracker system. The application is developed to ensure straightforward deployment, consistent behaviour across environments, and minimal operational complexity for both users and administrators.

A. Selection of the Deployment Environment

The E-Commerce Product Price Tracker supports flexible deployment across multiple platforms, including cloud-based services such as AWS, Microsoft Azure, and Google Cloud, as well as traditional Virtual Private Servers (VPS). The system can also be deployed as a fully web-based application, allowing users to access all features directly through a standard web browser without requiring additional software installation.

A typical deployment configuration includes:

Backend: Node.js for server-side processing

Frontend: React.js for responsive and interactive user interfaces

Database: MongoDB for storing structured and semi-structured data

Price Extraction: Web scraping tools such as Axios and Cheerio, or official e-commerce APIs where available

Notification Services: Email or in-app notification mechanisms for price-drop alerts

In addition to cloud-based setups, the system can be configured for on-premises servers, academic laboratories, or restricted environments with limited internet connectivity. Alternative deployment configurations and setup instructions are documented in the project resources.

B. System Integrity and Reliability

The architecture of the E-Commerce Product Price Tracker is designed to ensure accurate price monitoring, reliable alert delivery, and consistent system behaviour across different deployment environments. Core components that maintain system integrity include:

Price extraction and validation modules

Price comparison and threshold-based alert algorithms

Well-defined database schemas and API contracts

Any modifications to these components require thorough testing, as even minor changes may result in inaccurate price tracking or missed alerts. Therefore, system updates are recommended only after comprehensive validation.

Security and data-handling configurations are intentionally implemented using conservative defaults to protect user credentials, tracked product information, and alert preferences. Features such as encrypted password storage, controlled access to scraping services, and validation of data sources follow standard web application security best practices. Altering these configurations without proper evaluation is discouraged, as they are essential for maintaining system reliability, user trust, and data integrity.

C. System Preparation Prior to Deployment

Before deploying the E-Commerce Product Price Tracker, institutions or development teams should perform the following preparatory steps:

Identify user requirements and define the supported e-commerce platforms

Ensure legal and ethical compliance related to web scraping or API usage

Configure alert mechanisms (email or in-app notifications) according to user preferences

During development and testing phases, it is recommended to use sample or test product URLs to minimize unnecessary requests to live e-commerce platforms. Administrative privileges should be restricted to authorized personnel only, and system logs must be regularly monitored to detect failed scraping attempts or abnormal behaviour.

Additional authentication mechanisms, external data sources, or third-party integrations should not be introduced unless a comprehensive security and performance assessment has been conducted. The existing authentication and alerting framework has been validated for academic and consumer-oriented price tracking applications, ensuring dependable and user-friendly operation.

III. RELATED WORKS

In contrast to existing approaches, the proposed E-Commerce Product Price Tracker aims to provide an integrated web-based solution that combines automated multi-platform price tracking, historical price visualization, personalized price-drop alerts, and secure user data management. By addressing the limitations identified in prior work, the proposed system offers a more comprehensive, scalable, and user-centric solution for intelligent price monitoring in e-commerce environments.

3.1Literature Review

Recent research has explored a wide range of techniques related to online price monitoring, comparison systems, and intelligent e-commerce tools aimed at assisting users in identifying cost-effective purchasing options. Early approaches primarily focused on rule-based price comparison mechanisms. For example, Patel et al. [1] proposed a system that periodically monitored product prices from selected e-commerce websites. While effective within its scope, the system was limited to a small number of platforms and did not support real-time alert notifications.

Web scraping–based price tracking has been widely adopted in subsequent studies. Kumar and Singh [2] introduced a price monitoring framework that utilized HTML parsing techniques for extracting product prices. Although the approach enabled automated data collection, it was highly sensitive to changes in website structure, resulting in frequent extraction failures. Additionally, the system lacked visualization features for historical price analysis. Similar limitations were identified in the work of Rao et al. [3], who developed a browser extension for price tracking with basic price history visualization. However, the solution suffered from portability and compatibility constraints across different browsers and platforms.

Several studies have also investigated price prediction and trend analysis in e-commerce environments. Machine learning-based approaches were employed in [4] to forecast price fluctuations using historical data. While these methods achieved reasonable prediction accuracy, they required extensive datasets, complex preprocessing, and higher computational resources, making them less suitable for lightweight, real-time applications.

Alert-driven price monitoring systems have been examined as well. Sharma et al. [5] presented an email-based price drop notification framework. Although the system successfully informed users of price reductions, it lacked personalization features, such as user-specific thresholds, and did not support integration across multiple e-commerce platforms. Moreover, many existing solutions fail to provide comprehensive historical price data, which limits users' ability to evaluate long-term pricing trends.

In addition to functional limitations, recent research has highlighted concerns related to data privacy, ethical web scraping practices, and secure handling of user information [6]. These studies emphasize the need for robust authentication mechanisms, encrypted data storage, and responsible usage of third-party e-commerce data.

3.2 Comparison with Previous Methodology

The previous methodologies for e-commerce price tracking primarily relied on basic web scraping techniques, rule-based comparison systems, or standalone browser extensions. While these approaches were effective for limited use cases, they suffered from several constraints such as lack of scalability, dependency on static webpage structures, and absence of real-time alert mechanisms. Many systems were restricted to a small number of platforms and failed to provide comprehensive features like historical price analysis or personalized notifications. Additionally, machine learning-based approaches introduced in some studies required large datasets and high computational resources, making them unsuitable for lightweight and real-time applications. Security and user data management were also often overlooked, leading to potential privacy concerns.

In contrast, the proposed E-Commerce Product Price Tracker offers a more integrated, scalable, and user-centric solution. It combines automated multi-platform price monitoring with robust backend services, real-time alert generation, and interactive data visualization. Unlike earlier systems, it supports personalized price thresholds, efficient database management, and continuous background monitoring through scheduler services. The use of modern technologies such as React.js, Node.js, and MongoDB ensures high performance, flexibility, and ease of deployment. Furthermore, the system incorporates strong security practices, including encrypted authentication and secure data handling, addressing limitations observed in previous methodologies. Overall, the proposed approach provides a comprehensive and practical solution for intelligent and efficient price tracking in modern e-commerce environments.

Table.1. Comparison Table

Feature	Previous Methods	Proposed System
Platform Support	Limited (few websites)	Multi-platform support
Price Monitoring	Periodic / Manual	Automated & continuous
Real-time Alerts	Not available / Limited	Instant notifications
Personalization	Minimal	User-defined price thresholds
Price History Analysis	Limited or not available	Detailed graphical visualization
Scalability	Low	High (cloud-based architecture)
Technology Stack	Basic scripting / extensions	React.js, Node.js, MongoDB
Security	Limited focus	Secure authentication & data protection

3.3 Proposed framework

This section describes the architectural design of the proposed E-Commerce Product Price Tracker system, emphasizing its modular structure, scalability, and efficient deployment strategy. The system is designed using a well-organized approach that separates different functional components, ensuring maintainability and ease of future enhancements. By adopting a structured architecture, the system is capable of handling multiple users and large volumes of product data while maintaining consistent performance.

Fig.1.Proposed Framework

The overall framework of the system is based on a three-tier architecture, which includes the presentation layer, application layer, and data layer. This separation of concerns allows each layer to function independently while maintaining seamless communication between them. The presentation layer provides the user interface through a web-based frontend, enabling users to interact with the system easily. The application layer acts as the core processing unit, executing business logic, handling price extraction, managing alerts, and coordinating system operations. The data layer is responsible for storing and managing all persistent data, including user information, product details, and historical price records.

Users interact with the system through a modern web interface developed using JavaScript frameworks, ensuring accessibility across different devices and platforms. The frontend communicates with backend services through RESTful APIs, enabling efficient data exchange and real-time updates. This communication model ensures that users receive accurate and up-to-date information without delays, enhancing the overall user experience.

The technology stack of the system is carefully selected to balance performance, scalability, and ease of development. The frontend is built using React.js, which supports the creation of dynamic and responsive user interfaces. Styling is handled using CSS to ensure adaptability across various screen sizes. On the server side, Node.js with Express.js is used to implement backend services, providing a lightweight and efficient runtime environment. For automated price extraction, web scraping libraries such as Axios and Cheerio are utilized, enabling the system to retrieve data from multiple e-commerce platforms. MongoDB is employed as the database due to its flexibility in handling structured and semi-structured data.

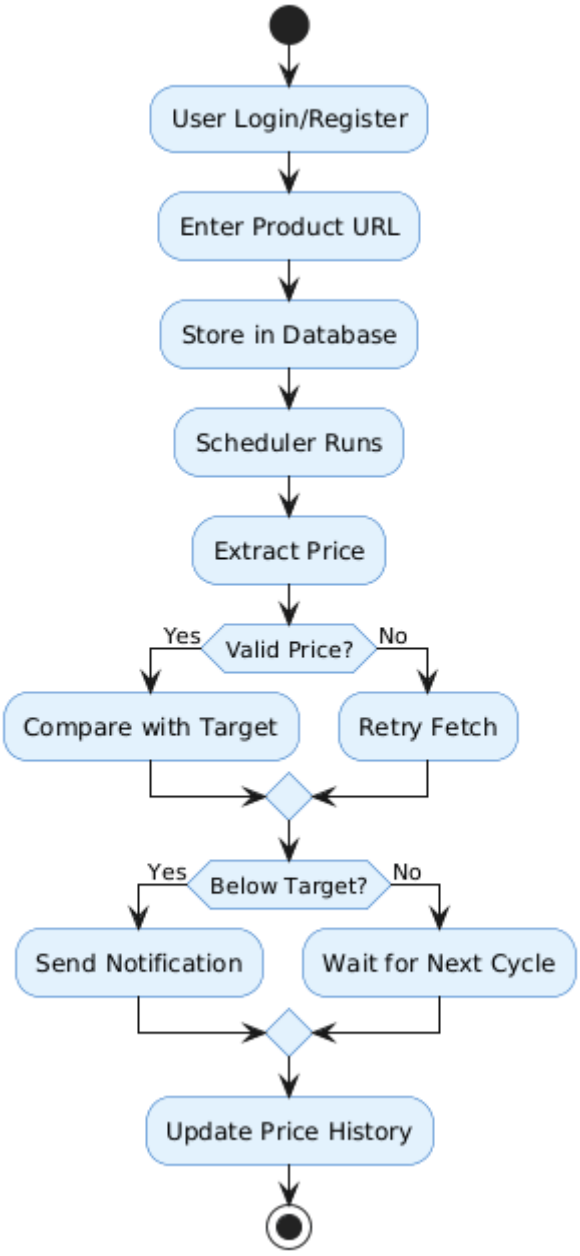

Fig.2.Working

The frontend design focuses on usability and responsiveness, ensuring a seamless experience for users. It includes secure authentication pages for user registration and login, allowing only authorized access to system features. The user dashboard provides a comprehensive view of tracked products and their current prices. Additionally, users can add product URLs, set price thresholds, and monitor historical price trends through graphical visualizations. A notification panel displays real-time alerts, ensuring that users are immediately informed of price drops.

The backend services play a critical role in managing the system's core functionalities. The authentication service ensures secure access through token-based or session-based mechanisms. The price extraction module retrieves product prices either through web scraping or official APIs, supporting multiple platforms. A scheduler service operates in the background, periodically updating product prices without requiring user intervention. The alert engine continuously compares current prices with user-defined thresholds and triggers notifications when conditions are met. The API layer facilitates smooth communication between the frontend and backend components.

The database schema is designed to efficiently store and manage data required for price tracking and analysis. MongoDB is used to maintain collections such as users, products, sources, prices, price history, alerts, and notifications. This structured approach allows quick data retrieval and supports long-term storage of historical price data. Efficient indexing and query mechanisms ensure that the system performs well even as the number of users and tracked products increases.

Automated monitoring and notification mechanisms are integral to the system's functionality. Background services continuously retrieve updated prices at regular intervals and compare them with predefined thresholds. When a price drop is detected, the system

generates instant alerts through email or in-app notifications. This automation eliminates the need for manual tracking and ensures that users never miss important price changes.

The deployment strategy of the system is designed to be flexible, supporting both cloud-based and on-premise environments. The application can be hosted on cloud platforms for scalability and remote accessibility, while MongoDB can be deployed either on cloud infrastructure or local servers. Additionally, logs and cached data can be stored locally or in cloud storage systems. To ensure reliability, the system incorporates fallback mechanisms and error-handling strategies, allowing it to recover gracefully from failures such as network issues or changes in website structure.

Fig.3.Main Methodology

3.4 Main Methodology

- The system begins with user registration and secure authentication using token-based or session-based mechanisms.
- Users log in to access the dashboard and manage their tracked products.
- Product URLs from various e-commerce platforms are submitted by users through the interface.
- Users define target price thresholds for each selected product.
- The system stores product details and user preferences in the database.
- A web scraping module extracts real-time product prices using tools like Axios and Cheerio or APIs.
- Extracted data is validated to ensure accuracy and consistency before storage.
- A scheduler service periodically triggers the price extraction process at fixed intervals.
- Updated prices are stored along with timestamps for tracking changes over time.
- A comparison algorithm evaluates current prices against user-defined thresholds.
- When a price drop is detected, the alert engine generates notifications instantly.
- Notifications are delivered to users via email or in-app alert systems.
- Historical price data is maintained to support trend analysis and visualization.
- Graphical representations such as line charts are generated for price history insights.
- The system ensures secure data handling, scalability, and reliable performance through efficient backend and database management.

3.4.1 Implementation

Step 1: User Authentication and Secure Access

The implementation begins with a robust authentication mechanism to ensure secure access to the system. Users register and log in through dedicated interfaces such as Sign In, Register, and Login pages. Authentication is handled using JSON Web Tokens (JWT) or session-based methods, which validate user identity and maintain secure sessions. All communication between client and server is encrypted using HTTPS to prevent unauthorized access. Passwords are stored using hashing and encryption techniques, ensuring strong protection of user credentials and sensitive data.

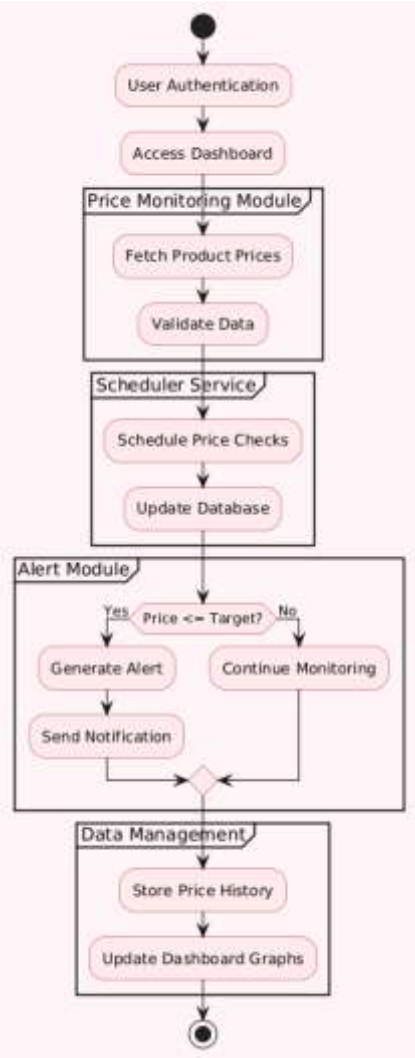

Fig.4.Implementation

Step 2: Product Registration and User Input Handling

After successful authentication, users can add product URLs from various e-commerce platforms into the system. Along with the product link, users define a desired price threshold for receiving alerts. The system processes and validates the input data before storing it in the database. This step ensures that only valid and trackable product URLs are accepted, minimizing errors during price extraction.

Step 3: Price Monitoring Module Implementation

The core functionality of the system is implemented through the price monitoring module. This module uses web scraping techniques with libraries such as Axios and Cheerio to extract real-time product prices from web pages. Where available, official APIs are also used for reliable data retrieval. The module is designed to handle dynamic webpage structures and includes validation checks to ensure the extracted data is accurate and consistent before storing it.

Step 4: Automated Scheduler Integration

A background scheduler service is implemented to automate periodic price tracking. This scheduler triggers the price extraction process at predefined time intervals without requiring user intervention. It ensures continuous monitoring of all tracked products and updates the database with the latest price information. This automation significantly reduces manual effort and ensures near real-time tracking efficiency.

Step 5: Data Storage and Database Management
The system uses MongoDB as its primary database to store all relevant data, including user profiles, product details, price records, and alert configurations. Each price update is stored with a timestamp to maintain historical records. Efficient indexing and optimized queries are implemented to ensure fast data retrieval and scalability as the number of users and tracked products increases.

Step 6: Alert Generation and Notification System
An alert engine continuously compares the latest product prices with user-defined threshold values. When a price falls below the specified limit, the system automatically generates notifications. These alerts are delivered through email or in-app notification systems. The module also logs all alert events for future reference and auditing, ensuring transparency and reliability.

Step 7: Price History and Trend Analysis Implementation
To provide deeper insights, the system maintains historical price data for each product. This data is used to generate graphical representations such as daily and weekly price trends, minimum and maximum price points, and long-term variations. These visualizations are displayed on the user dashboard, enabling users to analyze trends and make informed purchasing decisions.

Step 8: Error Handling and Fault Tolerance
The implementation includes robust error-handling mechanisms to manage failures such as network issues, invalid URLs, or changes in webpage structure. The system detects and logs errors, allowing for quick debugging and recovery. Fault-tolerant strategies ensure that failures in one module do not affect the overall system performance.

Step 9: System Integration and API Communication
The frontend and backend components are integrated using RESTful APIs. These APIs handle data exchange between the user interface and server-side logic, ensuring smooth communication. This modular integration allows independent development and scaling of system components while maintaining overall system consistency.

Step 10: Deployment and Performance Optimization
Finally, the system is deployed on cloud or local servers depending on requirements. Performance optimization techniques such as caching, efficient query handling, and asynchronous processing are implemented to enhance speed and responsiveness. The deployed system is tested for scalability, ensuring it can handle multiple users and continuous background operations without performance degradation.

IV. RESULTS AND DISCUSSION

4.1 System output screenshots and explanation

The results and evaluation of the proposed E-Commerce Product Price Tracker system demonstrate its effectiveness in delivering accurate, reliable, and real-time price monitoring. Functional testing confirmed that all major modules of the system operate correctly and are well integrated. Core functionalities such as user registration, authentication, product URL submission, and price threshold configuration were executed without errors. The system successfully extracted product prices from multiple e-commerce platforms and generated notifications whenever the defined conditions were met. The user interface remained responsive and consistent across both desktop and mobile environments, ensuring a seamless user experience.

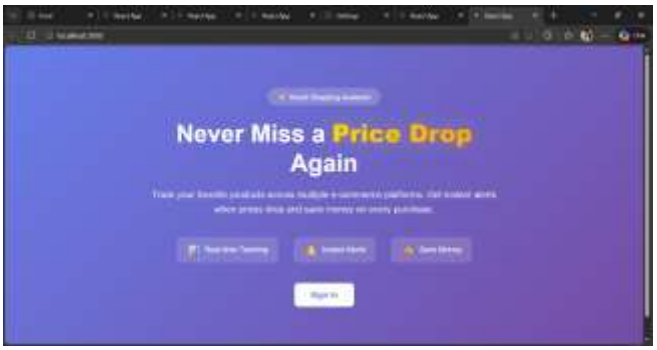

Fig.5.Main Dashboard

The user dashboard plays a central role in presenting system outputs, offering a clear and organized view of tracked products, current prices, and target thresholds. Users were able to efficiently manage products by adding or removing URLs dynamically. The dashboard also displayed historical price trends and alert notifications accurately, allowing users to monitor price changes effectively.

The intuitive design and ease of navigation contributed to improved usability, making the system suitable for a wide range of users without requiring technical expertise.

Performance evaluation indicates that the system meets the requirements for real-time price tracking. The average price-fetching time per product ranged between 200–300 milliseconds, while alert generation latency remained below 150 milliseconds. Database query execution times were also efficient, averaging between 100–130 milliseconds. These results confirm that the system can handle continuous background operations and multiple product tracking scenarios without significant performance degradation. The system demonstrated stable scalability, maintaining consistent performance even when multiple products were monitored simultaneously.

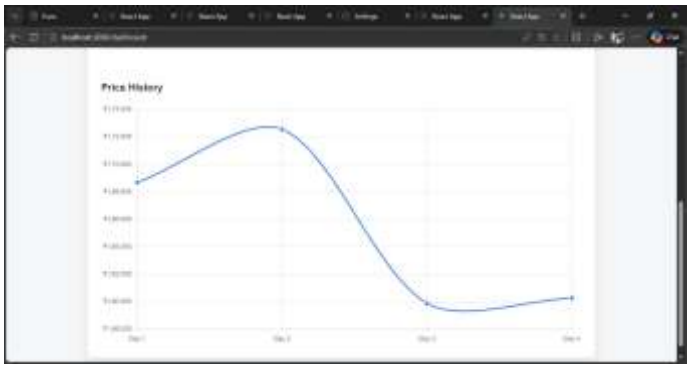

Fig.6.Price Trend

The deployment phase validated the system's ability to operate in a real-world environment using cloud-based infrastructure. The integration between the React.js frontend, Node.js backend, and MongoDB database was seamless, with all components communicating effectively through RESTful APIs. End-to-end testing confirmed that data flow, API responses, and notification mechanisms functioned as expected in the deployed setup. This demonstrates the system's readiness for practical implementation and large-scale usage.

Fig.7.Notification Page

From a security perspective, the system incorporates essential measures to protect user data and maintain system integrity. Passwords are securely encrypted, and authentication mechanisms ensure controlled access to resources. API endpoints are protected to prevent unauthorized usage, and user inputs such as product URLs are handled safely. No major vulnerabilities or data leakage issues were identified during testing, indicating that the system adheres to standard security best practices.

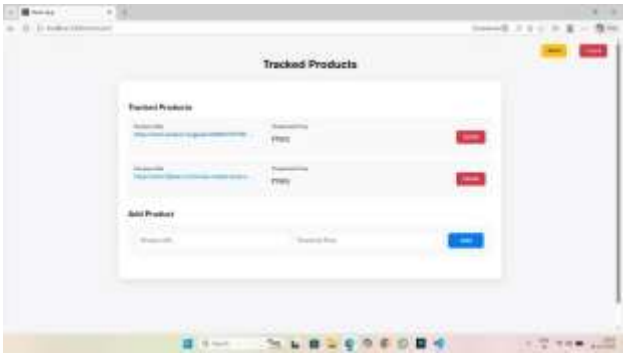

Fig.8.Price Tracker

The price history visualization feature provides valuable insights into product pricing trends over time. Graphical representations of daily price variations enable users to identify patterns such as sudden price drops during promotional events or gradual fluctuations due to market dynamics. This analytical capability enhances decision-making by helping users determine the most suitable time to purchase products. The visualization effectively transforms raw data into meaningful insights.

Additionally, the notification system proved to be reliable and efficient in delivering real-time alerts. Users received timely notifications whenever product prices fell below their specified thresholds. The system also ensured transparency by clearly indicating when no price drops were detected. Continuous background monitoring allowed the system to maintain up-to-date information without user intervention, significantly improving overall efficiency and user satisfaction.

Overall, the results demonstrate that the proposed system successfully achieves its objectives by combining accuracy, performance, usability, and security. The discussion highlights that the integration of automated monitoring, real-time alerts, and data visualization creates a comprehensive solution for modern e-commerce price tracking.

4.2 Conclusion

The proposed E-Commerce Product Price Tracker successfully addresses the challenges of manual price monitoring by providing an automated, scalable, and user-friendly solution. By integrating real-time price extraction, personalized alert mechanisms, and historical data visualization, the system enables users to make informed and cost-effective purchasing decisions. The use of modern web technologies ensures efficient performance, secure data handling, and seamless user experience across platforms. Experimental results confirm the system's reliability, accuracy, and responsiveness under continuous operation. Overall, the solution enhances transparency in online shopping and serves as a practical tool for intelligent price tracking in dynamic e-commerce environments. The proposed E-Commerce Product Price Tracker provides a comprehensive and efficient solution for overcoming the limitations of manual price comparison in dynamic online markets. By automating price monitoring across multiple platforms, the system ensures that users receive timely and accurate updates without continuous manual effort. The integration of real-time alerts, user-defined thresholds, and historical price analysis enhances decision-making by allowing users to identify optimal purchasing opportunities. The modular architecture and use of modern technologies contribute to high scalability, reliability, and ease of deployment in both cloud and local environments. Performance evaluation demonstrates low latency, stable operation, and consistent responsiveness, even under continuous monitoring conditions. Furthermore, the incorporation of secure authentication and data protection mechanisms ensures user trust and system integrity. Overall, the proposed system serves as a practical, intelligent, and cost-effective tool that significantly improves the online shopping experience by enabling data-driven purchasing decisions.

4.3 Future Scope

- Integration of machine learning algorithms to predict future price trends and recommend the best time to purchase.
- Development of a mobile application for Android and iOS to improve accessibility and user engagement.
- Expansion to support a larger number of e-commerce platforms and international marketplaces.
- Implementation of browser extensions for real-time price tracking while browsing product pages.
- Addition of advanced analytics such as price comparison across multiple platforms in a single view.
- Personalized recommendation system based on user preferences and purchase history.
- Integration of push notifications using SMS and mobile alerts for faster communication.
- Enhancement of web scraping techniques to handle dynamic and frequently changing website structures more efficiently.
- Implementation of voice assistant support for hands-free product tracking and alerts.
- Incorporation of discount and coupon aggregation features to maximize savings.
- Strengthening of security features with multi-factor authentication and advanced encryption techniques.
- Use of cloud-based microservices architecture to further improve scalability, performance, and system reliability.

V. Acknowledgement

We would like to express our sincere gratitude to our guide for her valuable guidance and support throughout this project. We also thank the management and faculty of our college for providing the necessary resources and environment. Finally, we thank our family and friends for their encouragement and support.

REFERENCES

[1] R. Buyya, R. N. Calheiros, and X. V. Wang, "Cloud computing for scalable web applications," IEEE Cloud Computing, vol. 7, no. 2, pp. 22–30, 2020.

[2] Chakrabarti and S. Banerjee, E-Commerce Analytics: Concepts and Applications. Cham, Switzerland: Springer, 2019.

[3] M. Al-Shaikhli and N. Omar, "Web scraping techniques and applications: A survey," International Journal of Computer Applications, vol. 175, no. 12, pp. 25–32, 2020.

[4] S. Kumar and R. Singh, "Automated price comparison system for online shopping platforms," International Journal of Advanced Computer Science and Applications, vol. 11, no. 6, pp. 410–416, 2020.

[5] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2012.

[6] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Univ. of California, Irvine, CA, USA, 2000.

[7] M. Grinberg, Flask Web Development: Developing Web Applications with Python. O'Reilly Media, 2018.

[8] S. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, 2nd ed. Pearson Education, 2007.

[9] Sommerville, Software Engineering, 10th ed. London, UK: Pearson Education, 2016.

[10] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2003.

[11] R. B. Gupta and S. M. Lee, "Security in modern web applications: Authentication and best practices," IEEE Transactions on Information Forensics and Security, vol. 18, pp. 1012–1024, 2023.

[12] M. E. J. Newman, Networks: An Introduction. Oxford, UK: Oxford University Press, 2010.

[13] Google, "Web scraping policies and best practices," Google Developers Documentation, 2023.

[14] Amazon, "Amazon Product Advertising API Documentation," 2024. [Online].

[15] Flipkart, "Flipkart Affiliate and API Documentation," 2024. [Online].