ENHANCING THE E-COMMERCE EXPERIENCE: A WISHLIST BROWSER EXTENSION WITH PRICE-DETECTION

Stanislav Dakov[1, *], Anna Malinova[2]

Abstract. In this article we present the development of a Chrome plugin for price (and other product features) tracking of a wish-listed product for a better e-commerce user experience. We also review some existing plugins that are closely related to the world of e-commerce. We explore the benefits of using such tools and how browser extensions and the microservices backend work together to improve the user experience. The article highlights the multifaceted benefits of web browser extensions, demonstrating their ability to improve the way users navigate and engage with e-commerce browsing.

Key Words: *e-commerce, wish list, price-detection, browser extension, microservices*

Introduction

The rise of e-commerce has revolutionized the way we shop, offering convenience, variety, and accessibility at our fingertips. As online shopping continues to dominate the retail landscape, web browser extensions have emerged as powerful tools that can greatly enhance the e-commerce browsing experience. These extensions provide added functionalities and features that can streamline the shopping process, save money, and ensure a secure and personalized experience. Web browser plugins offer several benefits for e-commerce browsing, enhancing the overall shopping experience, and providing users with additional functionality and convenience.

First, we will make an overview of some existing plugins, which help users to have a better experience in the e-commerce world. In the article, we show our implementation of the Chrome plugin [10] which helps to enhance the user experience. We emphasize the plugin part without a deep dive into the backend. The plugin can observe different plugin prices and notify users when the price has been changed. Users can dynamically select a product from different e-commerce websites. The plugin collects the price and stores it in the database. The user can check what was the previous price and when the price has been changed.

Benefits of web browser plugins for e-commerce browsing

A. Price Comparison and Savings

One of the key advantages of web browser extensions for e-commerce is the ability to compare prices and find the best deals. Price comparison extensions, such as Honey [7] and PriceBlink [8], automatically search for and display alternative prices from different retailers when users browse product pages. These extensions save time and effort by eliminating the need to visit multiple websites to compare prices manually. Additionally, they often provide coupon codes and discounts that can be applied at checkout, maximizing savings for online shoppers [1], [2].

Plugins like Keepa [9] and Invisible Hand provide price-tracking functionality. They monitor the prices of products over time and alert users when the price drops below a specified threshold. This feature helps users make informed buying decisions and purchase products at the optimal time, saving them money.

Price comparison extensions leverage algorithms to scrape pricing information from various online retailers in real-time. They provide users with an overview of the price range for a specific product, allowing them to make informed decisions and choose the most cost-effective option. By accessing a wide range of retailers, these extensions offer users a comprehensive view of the market, ensuring they get the best deal possible [2].

These extensions often provide additional features such as historical price charts and price drop alerts. Users can track the price history of a product and receive notifications when prices drop or reach a desired threshold. This enables users to strategize their purchases and buy products at the most opportune time, maximizing their savings [1].

B. Automatic Coupons and Discounts

Web browser extensions can help shoppers find and apply coupons and discounts seamlessly during the checkout process. Extensions like RetailMeNot [11] and Cently [12] (formerly Coupons at Checkout) automatically scan for available coupons and promotional codes when users reach the checkout page of supported e-commerce websites. With a single click, these extensions apply the best available discounts, ensuring users get the most value for their money without the hassle of searching for codes or deals elsewhere [1].

C. Wishlists and Price Tracking

Keeping track of desired products and monitoring price fluctuations can be a time-consuming task. Web browser extensions offer wishlist and price-tracking features that simplify this process. Extensions like Amazon Assistant [13] and Keepa [9] allow users to add products to their wishlists and receive notifications

when prices drop or reach a desired threshold. These extensions enable users to make informed purchasing decisions by waiting for the optimal time to buy and avoid impulse purchases [4].

Wishlist features provided by browser extensions allow users to create personalized lists of products they are interested in purchasing. Users can add items to their wishlists while browsing e-commerce websites, ensuring they have a centralized place to keep track of their desired products. This eliminates the need to remember or bookmark individual product pages, making it easier for users to revisit their favorite items later [4].

Price tracking features offered by these extensions monitor the prices of products on users' wishlists. They provide timely notifications when prices drop or reach a predefined threshold set by the user. This empowers users to take advantage of price reductions and make purchases when the cost aligns with their budget or desired savings. By leveraging these extensions, users can make well-informed decisions and avoid overpaying for products [4].

Some extensions provide price history charts that display the historical pricing trends of a product. These charts offer insights into price fluctuations over time, allowing users to assess the best time to make a purchase. By observing the pricing patterns, users can determine if a product's price is stable or subject to frequent changes, enabling them to make strategic buying decisions [1].

D. Automatic Form Filling and Secure Payments

Web browser extensions can streamline the checkout process by automatically filling in shipping, billing, and payment information. Extensions such as Autofill [29] and LastPass [14] securely store user details, eliminating the need to manually enter them for each purchase. This not only saves time but also reduces the chances of errors or typos during checkout. Furthermore, some extensions integrate with secure payment services like PayPal, providing an additional layer of security for online transactions [3].

E. User Reviews and Ratings

User reviews and ratings play a vital role in the online shopping experience, helping users make informed purchasing decisions. Web browser extensions offer the convenience of viewing product reviews and ratings directly on e-commerce websites. Extensions like ReviewMeta [24] and Fakespot [25] analyze user reviews for authenticity and provide insights into the overall quality and trustworthiness of product ratings. By utilizing these extensions, shoppers can avoid misleading or biased reviews, ensuring a more reliable shopping experience [5].

F. Enhanced Security and Privacy

Security and privacy are paramount concerns when it comes to e-commerce. Web browser extensions can bolster security and protect users' personal information during online transactions. Extensions such as HTTPS Everywhere [27] and Privacy Badger [28] enforce secure connections and block tracking elements, respectively, safeguarding against potential security threats and data breaches. Additionally, extensions like Blur and Ghostery [21] help users manage their online identities by providing masked email addresses and blocking third-party trackers that collect user data [6].

Wishlist with price detection – plugin development

We have made a Chrome plugin that can collect different information for the specific product. Users can easily and dynamically choose what to obtain from the plugin. In a common situation, the user selects the product price, but the plugin can check for different changes in any content, like description, title, stock value, or product quantity as well.

We will go through an overview of the architecture, functionality, and technologies used.

A. Architecture

Our implementation of the Chrome plugin is a packaged set of HTML, CSS, and JavaScript files that can modify or enhance the functionality of the Chrome browser. The architecture consists of several key components that work together to provide the desired functionality [15]:

- Manifest File - This file is the backbone of a Chrome extension and defines the basic properties of the extension, including its name, version, permissions, and background scripts.
- Background Scripts - JavaScript files that run in the background of the extension, providing additional functionality such as event listeners, API calls, and passing messages between the extension and web pages.
- Content Scripts - JavaScript files that run on specific web pages and change their behavior or appearance. They have access to the web page's DOM and can manipulate it using JavaScript.
- Popup Page - HTML page that appears when the extension icon is clicked. It usually provides a user interface for the extension and allows users to interact with the extension's features.
- Options Page - HTML page that provides a user interface for the extension's options and settings.

- Icon files - Images that represent the extension icon in the browser toolbar, extension manager, and other places.
- Localization Files - JSON files that provide translations for the extension's user interface, allowing it to be used in different languages and locales.

B. Functionality

Our plugin provides seamless authentication to a REST API [20], securely stores the authentication token in the browser, and retrieves data from the API to display it within the plugin's popup interface. This combination of functionalities requires a well-designed architecture to ensure smooth communication with the API, secure token storage, and efficient data retrieval.

We have 4 components:

- Chrome Extension: user-friendly interface, residing within the browser, that acts as a conduit for accessing various services.
- Authentication Service: The gatekeeper that validates users and issues access tokens, ensuring secure communication between the extension and other services.
- Data Retrieval Service: The dynamic data hub is responsible for providing and processing information.
- Worker Service: The service that performs background tasks, offloading intensive computations from the extension.

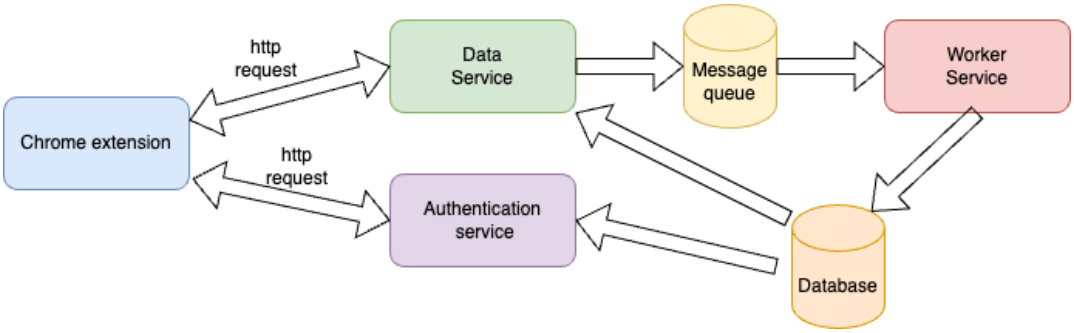

Figure 1. Communication flow

For our implementation, we use two types of communication [16]:

- Synchronous - HTTP protocol [19] for returning sync response.
- Asynchronous – AMQP (Advanced Message Queuing Protocol) protocol [18] for these async actions.

1) Authentication:

When a user clicks on the Chrome plugin icon, it triggers an event that signals the start of the authentication process. This event is usually captured by a background script within the plugin. The background script crafts an HTTP request to the authentication service's endpoint.

Figure 2. Example of an authentication request

To authenticate the plugin with a REST API, the plugin interacts with the API's authentication mechanism. This involves sending authentication credentials, such as a username and password, or obtaining an API key/token. The plugin provides a user interface within the browser to input the necessary authentication details. Once the user submits the required information, the plugin initiates the authentication process by making a secure HTTP POST request to the API endpoint designated for authentication.

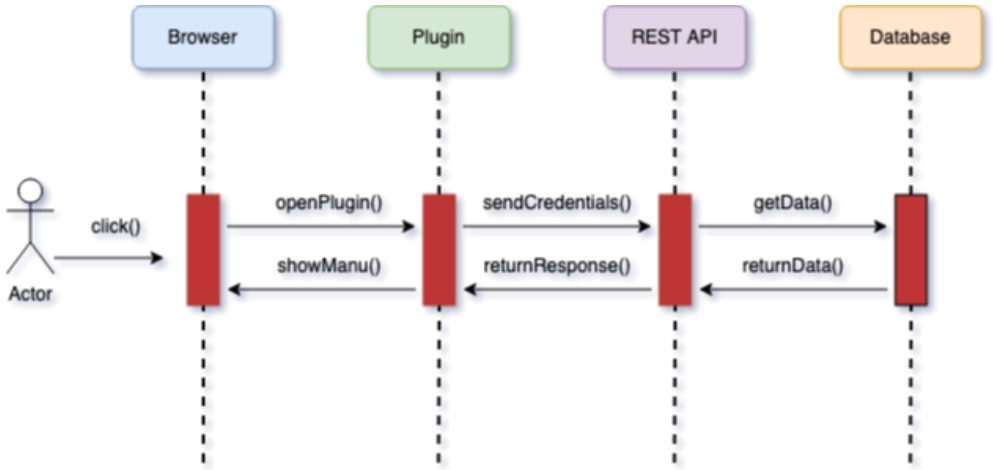

Figure 3. Authentication flow

Upon successful authentication, the REST API returns an authentication token JSON Web Token to the plugin. The plugin's architecture includes a secure and reliable method of storing this token in the browser.

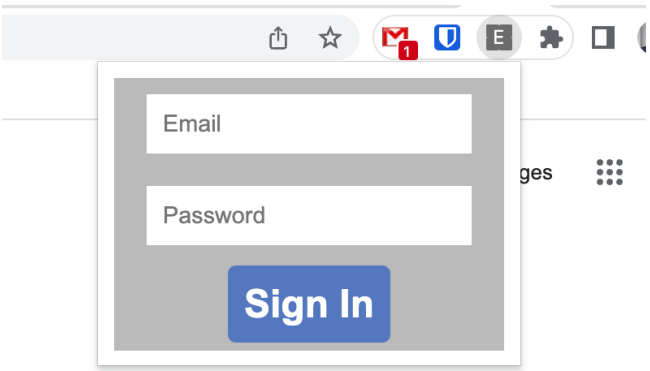

Figure 4. Authentication form

One common approach is to utilize browser storage mechanisms such as local storage or the extension storage API provided by the browser's extension framework. These storage methods ensure that the token remains accessible across browser sessions and tabs while maintaining its security. Our plugin stores the token in the browser's local storage as an encrypted string.

After the authentication token is securely stored, the plugin uses this token to make subsequent requests to the REST API's endpoints. The plugin architecture includes a mechanism to handle these API requests and responses. When the plugin requests the REST API it attaches the authentication token to the request headers, ensuring that subsequent API requests are properly authenticated.

2) Selecting product price:

The plugin provides functionality to select the price of a product by allowing users to interactively highlight the desired element on the screen. This is achieved through a button that, when clicked, initiates the process of marking the element with a visible border upon mouse hover. This button serves as the trigger to initiate the element selection process. Users can easily identify and interact with this button to activate the functionality.

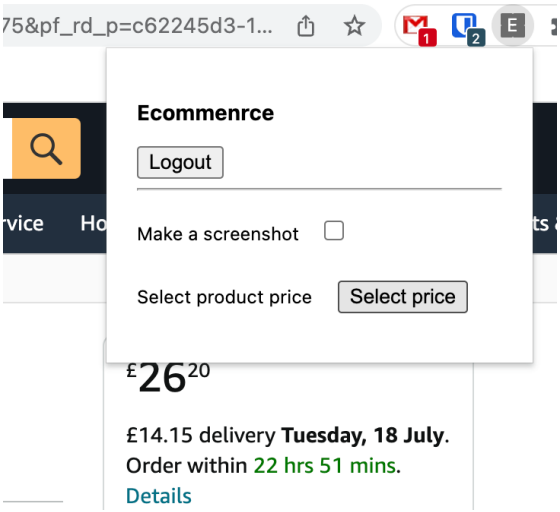

Figure 5. Plugin menu

When users click on the button, the plugin enters an element selection mode. The plugin sends a script to the current active tab and executes it. Then it starts a listener which listens for the message provided by that script. This message contains the unique CSS Path to that element.

As the user moves the mouse cursor over the webpage, the plugin dynamically highlights elements that can potentially represent the price of the product. When the cursor hovers over an element, the plugin visually marks it with a border or overlay, making it easily distinguishable from other page elements. This real-time highlighting allows users to precisely identify and select the element that corresponds to the price.

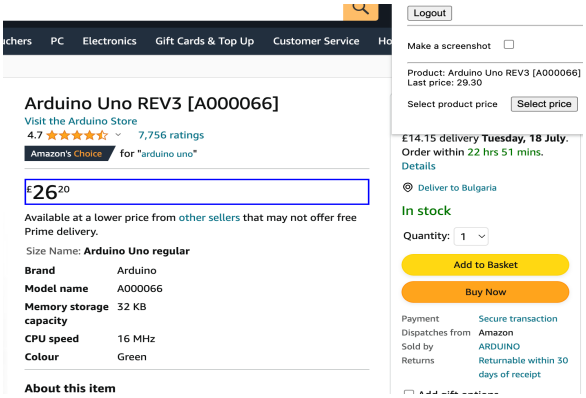

Figure 6. Selecting price

In this mode, the user interface of the plugin may change to reflect the active state, providing visual feedback that element selection is in progress. The plugin is now ready to receive user input to select the desired price element on the screen.

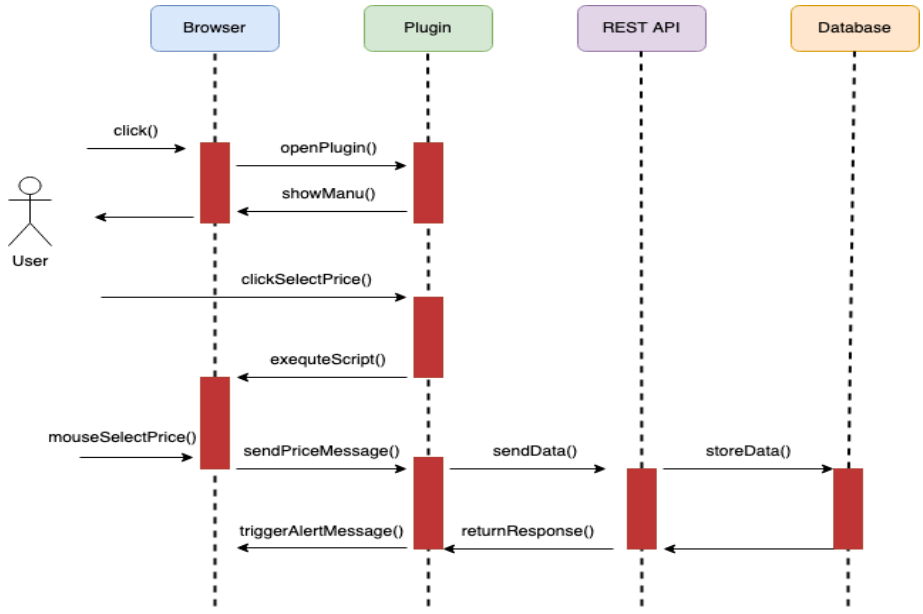

Figure 7. Selecting price task flow

When the user clicks on the selected price, the plugin receives the data and sends it to the REST API. When the data is successfully stored it triggers an alert message.

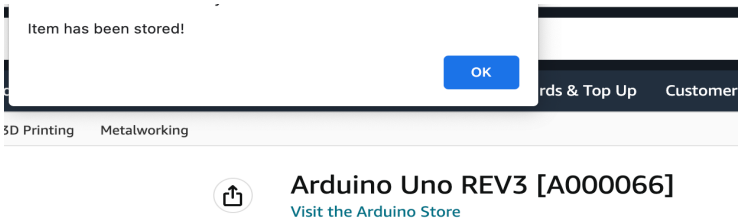

Figure 8. Success message

By integrating a button-triggered selection mode, real-time element highlighting, and secure storage of the selected price element, the plugin streamlines the process of gathering pricing data, empowering users to make informed purchasing decisions while browsing e-commerce websites. The plugin includes a checkbox within its user interface, located on the plugin's page. This checkbox serves as a control for enabling or disabling the screenshot capture feature. Users can interact with this checkbox to toggle the screenshot functionality on or off.

3) Screenshot

In addition to the element selection functionality and price capture, the plugin also offers users the ability to capture a screenshot of the current webpage. This feature is initiated by a checkbox within the plugin's user interface, providing users with the option to capture and save a snapshot of the page at the current moment.

When users check the checkbox, the plugin activates the screenshot capture feature. The user interface provides visual feedback to indicate that the feature is enabled, as a highlighted checkbox.

4) Show product data:

The plugin provides functionality to show the latest price for a current product. When the user goes to a specific product page and clicks on the plugin, the plugin gets the URL of the current active tap, that product is shown and makes a HTTP request to the REST API using the authentication token to retrieve already stored information about this product.

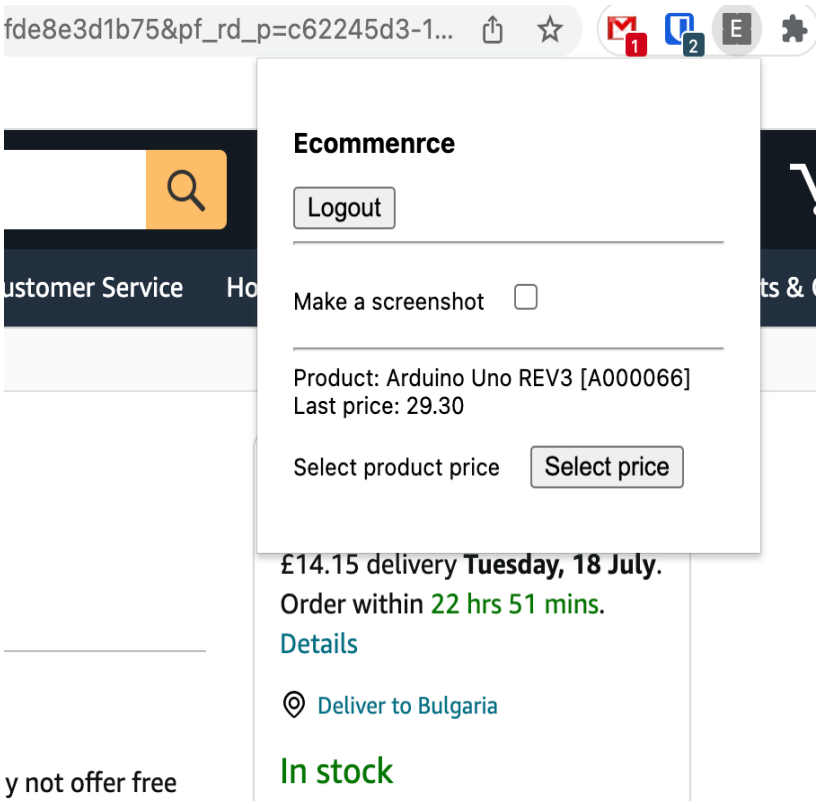

Figure 9. The plugin shows the old price

Once the data is fetched from the REST API, the plugin processes, and formats it to be displayed in the plugin's popup. The popup interface is created using HTML, CSS, and JavaScript, with the retrieved data dynamically rendered within the popup's DOM elements. The architecture includes appropriate event handlers and data binding techniques to ensure a responsive and interactive user experience within the plugin's popup interface.

C. Used technologies

Here are some of the used technologies for creating our browser extension:

1) Frontend, plugin UI:

- HTML is used for creating the structure and content of extension pop-ups, options pages, and other user interface elements.
- CSS is used for styling the extension's user interface to provide a visually appealing and consistent appearance.
- JavaScript is used for implementing interactive behavior, handling events, manipulating the DOM (Document Object Model) [23], and communicating with external APIs [31].

2) Browser-Specific APIs:

Each browser provides a set of browser-specific APIs (Application Programming Interfaces) that enable developers to access and interact with various browser functionalities. For example, Chrome provides the Chrome API,

Firefox provides the WebExtensions API, and Edge provides the Microsoft Edge API. These APIs allow developers to manage tabs, manipulate bookmarks, access browser storage, communicate with web pages, and perform other browser-related tasks. For our plugin, we have used Chrome API.

There are several extension frameworks and libraries that aim to simplify the development process and provide cross-browser compatibility. Examples of popular extension frameworks include:

- Chrome Extension Manifest (for Chrome extensions)
- WebExtensions (supported by Firefox, Edge, and other browsers)
- Safari App Extension (for Safari extensions)

These frameworks offer higher-level abstractions, standardized APIs, and tools that help streamline the development and deployment of extensions across multiple browsers. It's important to consider the browser compatibility and specific requirements of each target browser when selecting the technologies and libraries for your extension development. Additionally, referring to the official documentation and developer resources provided by each browser's extension ecosystem can provide valuable insights and guidelines for creating robust and effective web browser extensions.

3) Backend Technologies

The backend contains 3 microservices [17]:

- Authentication service
- Data service
- Worker service

For authentication and data service, we have used PHP with SQL database [30]. For the Worker service, we used JAVA, and RabbitMQ [22] as a buffer for all tasks. When something has been changed the Worker sends requests to the Telegram bot [26] via Telegram SDK.

Conclusion

Our implementation of the Chrome plugin successfully can help the user to easily track product detail when they have been changed especially the product price. Users can interact with all e-commerce platforms and add any product to be watched. When something has been changed, the user is notified of that change. The plugin successfully makes the user experience better in the e-commerce world.

Acknowledgments

This work is supported by the project FP23-FMI-002 of the Scientific Fund of the Paisii Hilendarski University of Plovdiv, Bulgaria.

References

[1] P. Chatterjee, A. Joshi, *Modeling and analysis of online coupons with reference price effects*, Journal of Marketing Research, 2018, pp.837-853.

[2] S. Das, Y. Kim, *Impact of online product reviews on product price: An empirical investigation*, Information Systems Research, 2018, pp.556-570.

[3] J. Hong, J. Landay, Automatically filling in web forms using content and context, ACM Transactions on Information Systems, 2004, pp.374-413.

[4] W. Moe, P. Fader, *Capturing evolving visit behavior in clickstream data*, Journal of Interactive Marketing, 2004, pp.5-19.

[5] D. Park, J. Lee, I. Han, *The effect of on-line consumer reviews on consumer purchasing intention: The moderating role of involvement*, International Journal of Electronic Commerce, 2007, pp.125-148.

[6] A.Vemuri, H. Topaloglu, *Privacy concerns and factors influencing online transactions*, International Journal of Electronic Commerce, 2015, pp.53-87.

[7] Honey, https://chrome.google.com/webstore/detail/honey-automatic-coupons-r/bmnlcjabgnpnenekpadlanbbkooimhnj.

[8] PriceBlink, https://chrome.google.com/webstore/detail/priceblink-coupons-and-pr/aoiidodopnnhiflaflbfeblnojefhigh.

[9] Keepa, https://chrome.google.com/webstore/detail/keepa-amazon-price-tracke/neebplgakaahbhdphmkckjjcegoiijjo.

[10] Liu L, Zhang X, Yan G, Chen S, *Chrome Extensions: Threat Analysis and Countermeasures*, InNDSS 2012 Feb.

[11] RetailMeNot, https://chrome.google.com/webstore/detail/retailmenot-deal-finder%EF%B8%8F/jjfblogammkiefalfpafidabbnamoknm.

[12] Cently, https://chrome.google.com/webstore/detail/cently-automatic-coupons/kegphgaihkjoophpabchkmpaknehfamb.

[13] Amazon Assistant, https://chrome.google.com/webstore/detail/amazon-assistant-for-chro/epikoohpebngmakjinphfiagogjcnddm.

[14] LastPass, https://chrome.google.com/webstore/detail/lastpass-free-password-ma/hdokiejnpimakedhajhdlcegeplioahd.

[15] Chrome plugin development, https://developer.chrome.com/docs/extensions/mv3/getstarted/development-basics/.

[16] F. Cristian. 1996, *Synchronous and asynchronous*, Commun. ACM 39, 4, 1996, pp. 88–97. https://doi.org/10.1145/227210.227231.

[17] W. Hasselbring and G. Steinacker, *Microservice Architectures for Scalability, Agility and Reliability in E-Commerce*, IEEE. International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden,2017,pp.243-246, doi: 10.1109/ICSAW.

[18] S. Vinoski, *Advanced Message Queuing Protocol,* in IEEE Internet Computing, vol. 10, no. 6, pp. 87-89, Nov.-Dec. 2006, doi: 10.1109/MIC.2006.116.

[19] Fielding R, Gettys J, Mogul J, *RFC2616: Hypertext Transfer Protocol–HTTP*/1.1 [J]. 1999.

[20] Williams, Brad, Tadlock, Justin, Jacoby, John, *REST API*, 2020, 10.1002/9781119666981.ch12.

[21] Ghostery, https://chrome.google.com/webstore/detail/ghostery-%E2%80%93-privacy-ad-blo/mlomiejdfkolichcflejclcbmpeaniij

[22] V. M. Ionescu, *The analysis of the performance of RabbitMQ and ActiveMQ*, 2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER), Craiova, Romania, 2015, pp. 132-137, doi: 10.1109/RoEduNet.2015.7311982.

[23] L. Wood, A. Le Hors , V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, G. Nicol, J. Robie, R. Sutor, C. Wilson, *Document object model (dom) level 1 specification*. W3C recommendation. 1998 Oct 1;1.

[24] ReviewMeta, https://chrome.google.com/webstore/detail/reviewmetacom-review-anal/fjifglfkcaipnmhngbigdebkoikioend

[25] FakeSpot, https://chrome.google.com/webstore/detail/fakespot-fake-amazon-revi/nakplnnackehceedgkgkokbgbmfghain

[26] R. Aisyah, D. Istiqomah, M. Muchlisin, *Developing E-learning Module by Using Telegram Bot on ICT for ELT Course*, In 5th International Conference on Arts Language and Culture (ICALC 2020) pp. 106-111, Atlantis Press, 2021.

[27] HTTPS Everywhere, https://chrome-stats.com/d/gcbommkclmclpchllfjekcdonpmejbdp

[28] Privacy Badger, https://chrome.google.com/webstore/detail/privacy-badger/pkehgijcmpdhfbdbbnkijodmdjhbjlgp

[29] Autofill, https://chrome.google.com/webstore/detail/autofill/nlmmgnhgdeffjkdckmikfpnddkbbfkkk

[30] W. Khan, T. Kumar, C. Zhang, K. Raj, A. Roy, B. Luo, *SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A*

Systematic Literature Review, Big Data and Cognitive Computing, 2023 May 12 7(2):97.

[31] T. Gardjeva, N. Pavlov, A. Rahnev, (2019), *JavaScript report generator*, Proceedings of the Scientific Conference "Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies", 29-30 November 2018, Pamporovo, Bulgaria, 2019, ISBN: 978-619-202-439-0, pp. 73-78.

Stanislav Dakov[1, *], Anna Malinova[2]

[1, 2] Paisii Hilendarski University of Plovdiv,

Faculty of Mathematics and Informatics,

236 Bulgaria Blvd., 4003 Plovdiv, Bulgaria

Corresponding author: stanislav.dakov@uni-plovdiv.bg