# A Privacy-Aware Browser Extension to Track User Search Behavior for Programming Course Supplement

Jihed Makhlouf[1,2(✉)], Yutaka Arakawa[2], and Ko Watanabe[3]

[1] Osaka Prefecture University, Osaka, Japan
[2] Kyushu University, Fukuoka, Japan
jihed.makhlouf@m.ait.kyushu-u.ac.jp, arakawa@ait.kyushu-u.ac.jp
[3] University of Kaiserslautern and DFKI GmbH, Kaiserslautern, Germany
ko.watanabe@dfki.de

**Abstract.** There is an abundant and constantly growing amount of information that can be retrieved from online resources. Moreover, the access to such resources is becoming more and more convenient. Yet, finding the exact needed information is not easy, especially for programming search queries. In this paper, we present TrackThinkTS, a privacy-aware browser extension. It tracks users' behaviors when navigating the web. The extension logs various user actions related to tab management, search query, browsing, and clipboard management. The extension is built with a privacy-first mindset. In fact, the users have full control over the registered logs, they can manage, update and export the logs in a completely transparent way. The vision behind this work is twofold. On one hand, we aim to investigate the web search behavior of programming students and detect patterns of a successful search. On the other hand, the objective is to build a knowledge base that will serve as a course supplement for programming students. Therefore, the proposed extension in this paper is one of the building blocks of the whole system. Data collected from this extension will be also synchronized with log data coming from an online IDE used by programming students during the experiment phase.

**Keywords:** Web search · Browser extension · Programming learning · Course supplement

## 1 Introduction

In a digital world where an immense load of data is created every second, it became very important to know where and how to search for the right information in a timely manner. Terms such as "Information Overload" are frequently used to express the phenomena [11]. Moreover, individual web search behavior and abilities became topics of meticulous research [3,5,9,12].


T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 783–796, 2022.
https://doi.org/10.1007/978-3-030-94822-1_51

Additionally, the problem of finding the right information is amplified when searching for programming errors, examples, and code snippets [19]. In fact, traditional general-purpose search engines are not optimized for programming code search [24]. Instead, they are developed to handle natural text requests. Thus, sometimes they fail to recognize the context and semantics of the code search [18]. Several tools were created to address the code search problem like the Google Code Search [7], Source Graph [21], Searchcode [20] and many others. However, some of these tools were either discontinued or became obsolete [18].

Undoubtedly, web searching is an integral part of the software programmer's activities. The purpose of the web search can vary a lot depending on the context. But many studies found that web searching is one of the most frequent activities of software developers [8,19,24]. However, most of the studies about code search were conducted in a professional environment. The participants in these studies had different levels of expertise in programming. Less focus was given to investigate the web searching behavior of programming students.

In fact, during the process of learning programming, whether it is self-learning or in academic settings, students use the search engines for many various purposes. In addition, the recent events related to the COVID-19 pandemic made the situation even worse. Educational institutions had to shift to fully online learning. This reduced considerably the opportunities for students to communicate and collaborate with each other. Hence, they lost the chance to learn from each other. Moreover, the interactions with the professors and teachers lost the qualities of face-to-face communication. In the programming case, teachers used to monitor and help students more effectively. Industry and academia are trying to mitigate the negative effects of the lockdown. Nevertheless, students in general, and programming students in particular rely more than ever on online resources. It is crucial to help them acquire the skills that help them achieve fulfilling web searches. But, a few research works did focus on this sub-population of programming web search. Therefore, in this study, we introduce a tool that will be used to examine the web search behavior of students while they solve programming exercises.

In the current manuscript, we present a browser extension called TrackThinkTS. This extension logs the browser usage while searching and surfing on the web. It was developed with a privacy-first mindset. Users have full access to view, edit and delete entries before exporting the data to CSV format. Along with their full consent, participants in the experiments using this extension have an aggregated view of the stored logs for an easier management.

This extension will serve as the building block for subsequent experiments and analysis that aim to investigate the following research questions:

- Is there any difference in web search skills between successful students and the others?
- To which extent this difference in web search skills influences the learning of the students?
- What are the common patterns of effective web search skills manifested through thought processes?

– What are the best ways of sharing these thought processes with the students through course supplements?

The rest of the paper is divided as follows: In Sect. 2, we will introduce the general context of our work by reviewing the related work about web search in general, and in the context of programming. Section 3 gives more details about the extension, its architecture, and the collected data. In Sect. 4, we will describe how we preserve users' privacy by giving them full control of the logs before exporting them by themselves. Section 5 is dedicated to discussing the vision and the possible use cases for the extension. We will also give an example of an experiment using the extension before concluding.

## 2 Related Work

### 2.1 User Web Search

The internet and the web became an integral part of our daily lives. It is a hub of rich content and various services. Searching on the web is one of the most common activities of internet users [9]. Therefore, understanding web search behavior is a necessary step to provide better services [11].

The approach used during information search on the web depends on several different factors. For example, the user's familiarity with computers, and the internet has a significant effect [9]. Researchers also found that knowledge-based factors, like domain knowledge, cognitive abilities, and even affective states can impact the effectiveness of the web search [9]. Other studies found that demographic elements, including age might also affect the effectiveness of the web search [9,12].

However, aside from these factors, there is a certain knowledge related to web search itself. In fact, there are different strategies that can be used for a successful web search. Studies found that reformulating the search query by removing words, stemming, word substitution or similar changes might lead to more interesting results for the user [10]. Moreover, the mainstream search engines have developed a set of rules using special characters that can alter the behavior of the search engine itself in the process of solving the search query. However, not all users are aware of the existence of such tricks to improve their search capabilities [23]. Therefore, the skills to properly use the search engines and formulate, also eventually reformulate, the search query is essential to maximizing the efficiency of the web search. The culmination of all these factors and skills is described as "Search Expertise" [3,23].

Hence, "Search Expertise" is a topic of interest to many researchers. Some of them investigated the aspects that differentiate between the "advanced" and "non-advanced" search engine users [23]. Other studies tried to find measures of search expertise [3]. Furthermore, a team of researchers from Microsoft explored some ways of transferring the search expertise between users [15].

The search expertise transfer is a very valuable concept that can have many different applications. In fact, it can be handy for the difficult topics of web search

such as programming and code search. By nature, programming-related search queries carry an inherent complexity compared to traditional search engines [18]. Therefore, programming-related search queries constitute a particular sub-topic of research.

### 2.2 Programming Web Search

During the process of software development, developers usually rely on resources available on the web [18]. In fact, studies have found that developers spend 15% of their time on web search [24]. According to the same study, developers spend up to 35% of their time on web search if they are new to the organization or to the project [24]. However, the time spent on web search is not the only factor that influences the web search behavior. In fact, the authors also found that the number of queries varies a lot [24]. This suggests that many developers find the needed information by web navigation rather than web search, which is also a common behavior [5].

However, a high number of web search requests might indicate an ineffective strategy for searching. In addition, many studies pointed out that the traditional search engines are not well adapted for programming-related web searches [18, 19, 24]. When they formulate properly their problems, developers can actually find decent results from their web search queries. This includes queries about debugging, documentation, etc. But, when they use the traditional search engines for code search such as code snippets, the search engines usually fail to deliver the appropriate results [18, 19, 22, 24]. Some of the reasons behind that are the difficulties of search engines to deal with special characters that are heavily present in programming code.

Several tools and services tried to address the code search problems. However, they are still underused. Additionally, several researchers proposed to bring the web search closer to the developer for a seamless experience. They developed an extension for Eclipse IDE that launches a web search query related to the error and exceptions encountered [17]. Then, it fetches the relevant potential solutions to solve it. The whole process is done within the IDE, therefore the user does not have to switch his work environment [17].

While the proposed solutions are helpful and can be handy in many situations, they are more adapted to professionals that are looking quickly to solve the problem at hand. However, students and people who are starting to learn programming in general or a particular programming language do not benefit from these solutions. More adjustments need to be done to help them learn.

### 2.3 Programming Learning Web Search

In the study conducted by Xia et al. [24], they found that in general new recruits and new graduates spend more time engaging in web search and also formulate more search queries compared to more experimented colleagues. Such a behavior was also discussed in other studies conducted in a slightly different context. In fact, in another research work, the authors found that a clear different behavior

for students learning a new programming language compared to professional developers [2].

It is clear that students need tools that provide them extra support during their learning phase of programming. In fact, one of the common ways of programming for newcomers is to look for existing code and change it according to their needs. This sort of behavior is called opportunistic programming [4]. Therefore, it is really important to help the students find the appropriate resources. The browser extension that we are presenting in this paper is a step toward achieving this goal.

## 3 The TrackThinkTS Extension

There were many tools that track the users' web search. However, most of them only capture the pages visited using the URL and provide some analytics and visualization based on that [1,6,14,25]. However, this approach does not grasp the whole web search behavior of the user. In addition, most modern browsers provide a complete API for a sophisticated access to the internal browser state and allow advanced manipulations.

The tool presented in this paper is the continuation of a previously discontinued proof-of-concept by the same name of TrackThink [16]. It was unfinished and suffered from a few problems. The main objective was to gather as much information as possible of the users' actions within the browser when they engage in web search to capture their "thought process". Compared to the previously mentioned tools, it did not rely solely on the history of the web pages and URLs. However, it was not published in the Chrome extension store and was used only by the developers. The UX needed a lot of improvements. Moreover, it was not optimally adjusted for user privacy nor for the user convenience.

Therefore, in the present update, we aim to transform the proof-of-concept into a fully working product and improve several critical aspects such as:

- Drastically improve the user privacy by giving full control to the user on the registered logs.
- Improve the UX and workflow for both the participants and the experiment organizers.
- Publish it in the Chrome store for easier distribution and installation in the students' machines.

Ultimately, TrackThinkTS inherits the name and some concepts of the original unfinished work, but it is fundamentally different in many aspects including the logs generation the storage, and the whole workflow and experiments.

### 3.1 Logs Generation

Many of the previous research works interested in user behavior when searching on the web used a limited set of information. They, gathered the history, represented mainly by URLs, of the web search. However, in our case, we would

like to gather as much data as possible by exploiting the browser capabilities and API. Therefore, along with the visited pages URLs, TrackThinkTS captures several types of events within the browser. Particularly, the interest in gathering tabs management is driven by the nature of the use case. The users have to switch between tabs frequently. They use some tabs to search on the web, then get back to the online IDE where they continue working on the programming exercises. Therefore, tracking systems based on URLs only are not effective in our situation.

**Tabs Management.** There are 4 particular events related to tabs that TrackThinkTS monitors: Tab creation, Tab activation, Tab update, and Tab delete. There are many advantages to using the browser extension Tab API. Tab creation in itself does not carry a lot of information, but can be useful to detect which is the default new tab page of the user and perhaps it can be used to detect advanced users. Tab activation is important to collect data about page switches. Tab update is triggered when a change happens to the page loaded in a tab. It can be a visit to a new page in the same tab, a refresh, or some new content being loaded on the page. This is particularly useful to capture events that do not require a full page reload (e.g. Javascript events). These events would be hard to detect if we rely only on the history and URLs. Finally, Tab delete is triggered when the user closes a Tab.

**Window Operations.** The extension also monitors users' specific actions, namely Scrolling and Clipboard usage. When the user scrolls on a page, we gather the corresponding coordinates and the visible content of the page. Additionally, we collect information about the viewport of the page. This way, we can easily recover which part of the information displayed on the web page was the most helpful. Additionally, if the user copies pieces of text, we detect it and save the copied text.

Every captured event is also appended to some additional information such as the timestamp, and user identifications. In fact, the user is attributed a randomly generated user id and we ask the participants to input their names as well.

### 3.2 Workflow

The development of the TrackThinkTS extension was achieved using the browser extension API. So far, TrackThinkTS supports officially the main chromium-based browsers like Google Chrome, Brave, and Vivaldi. Any other chromium-based browser might be able to use the extension, but we did not perform thorough testing.

Figure 1 shows the workflow of TrackThinkTS. There are two ways of storing the log data. The first method is to use a cloud-based database such as Firebase. This storage option is deactivated by default and only available when the experiment is conducted in a controlled environment (more details in Sect. 5). The second storage option is to save the log data in the browser local storage.

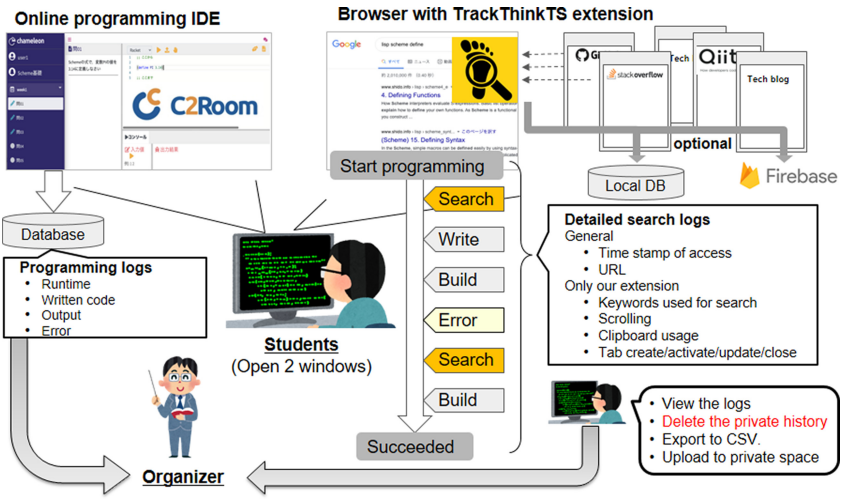


**Fig. 1.** TrackThinkTS workflow.

Later, the logs are aggregated and the user can manage the logs within an integrated dashboard of the extension. After checking the logs the user can export them into a CSV file and submit them to a submission location prepared by the experiment organizers. A proper manual was prepared for the users.[1]

The typical workflow will be as follows. Students will be solving programming problems and have the ability to search the web. They write their code in an IDE and when they are stuck or face an error they turn to the internet to find a solution. Meanwhile, the extension logs most of their actions within the browser.

On the other side, the researchers and experiment organizers can access the logs from 2 locations. If the experiment was conducted in a controlled environment with the cloud database upload activated, then they can access the dataset directly from there. Otherwise, the researchers have to prepare the appropriate structure to store the students' submissions before using them.

## 4 Logs Control

The TrackThinkTS extension logs almost all the users' actions in the browser. Therefore, there is a natural concern about the users' privacy. To address this issue we give the users full control over what they want to submit. There are two levels of control. The first level of control happens within the extension itself, the second level of control occurs when the user exports the logs into CSV and can manage the dataset as it is.

In fact, on the TrackThinkTS configuration page, we display the list of all actions aggregated by URL. That means each row represents a unique URL.

[1] https://ubi-naist.github.io/TrackThink/en/usage.

The row includes the number of actions that have been executed on that page. Figure 2 exposes the TrackThinkTS options page. All the aggregated data logs are displayed in an advanced datatable. Based on this aggregation the users can delete all the logs saved for a particular URL. However, the logs might be too long. So, the users can reorganize and order the entries or apply advanced filters to find specific URLs and remove the corresponding logs. In fact, URLs are unique, however they might share the exact same domain (e.g. google.com), so filtering is useful in such situations.

The options page also allows the participants to input their names, but more importantly, it allows the experiment organizers to apply advanced settings, which are the activation of the cloud upload of logs and the reset of the randomly generated user id. These settings are protected by a password and therefore users cannot tamper with them.

## 5 Use Cases and Vision

### 5.1 Experiments

It is worth noting that the TrackThinkTS browser extension is aimed to be a building block for several studies on web search behavior and course supplement. We will particularly focus on web search behavior in the case of programming learning and assignments. Two experiments have been recently performed using TrackThinkTS. The first experiment was conducted in a controlled environment where the participants have to use a dedicated computer in a reserved room. The computer had all the necessary tools installed, including TrackThinkTS. In this particularly controlled environment, the cloud-based database was used to gather the logs and the participants did have to export manually the logs. This was the setup before the TrackThinkTS extension was accepted for publishing in the Chrome extension store. The second experiment has been conducted on a larger scale after the extension was available in the store and participants could use their own computers. Therefore, the cloud-based logging using Firebase was disabled.

Both experiments consisted of a series of Functional Programming exercises. The participants used an online educational software called C2Room[2] that included an online IDE for programming. Within the C2Room software, they can access the exercises and start solving them using the built-in IDE. Each one of the experiments consisted of 10 exercises using the Scheme programming language with increasing difficulty. Participants were told that they are free to search for online resources when they needed.

### 5.2 Dataset Composition

As explained earlier, the TrackThinkTS browser extension logs different types of events related to the browser usage. Moreover, to solve the programming

[2] https://c2room.jp.

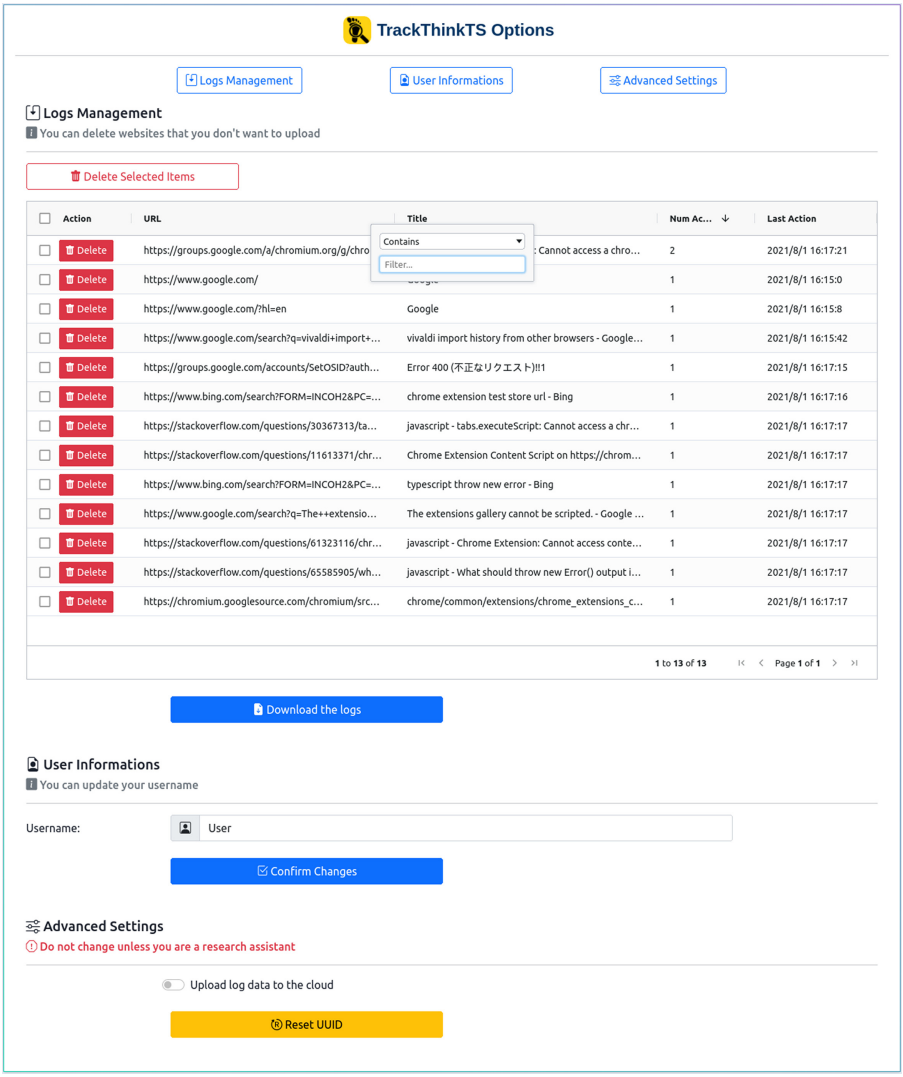


**Fig. 2.** TrackThinkTS options page.

exercises, the students used the C2Room educational software containing an online IDE. We could recover the students' usage logs from the C2Room software as well. Furthermore, we could gather additional information about the students participating in the experiments. Accordingly, the overall dataset is rich and diverse. Each recording in the dataset contains various information depending on the source. More details are provided as follows:

**TrackThinkTS Information.** Each row in the TrackThinkTS dataset represents an action done within the browser and is defined by this set of information:

*Action related:*

- UUID: A unique user ID generated for each user.
- User action: The type of the user action. It can be one of the following: TabCreate, TabActivate, TabUpdate, TabRemove, ClipboardCopy, and WindowScroll.
- Datetime: The timestamp of the action performed.

*Tab related*

- Title: The title of the page where the action happened.
- URL: The URL of the page where the action happened.
- BodyText: The whole content of the page where the action happened.

*Contextual*

- Scroll: Data is not filled here unless the User action is a WindowScroll event. The scroll data contain plenty of information related to the viewport, the scroll speed, the scroll rate, and the document width and height.
- Clipboard: Data is not filled here unless the User action is a ClipboardCopy event. The clipboard data basically contain the copied text.

**C2Room Information.** Each row in the C2Room dataset represents an action defined by this set of information:

*Action related:*

- UID: The user ID set up before the start of the experiment.
- ClassID: The ID of the class to which the user enrolled.
- TaskID: The ID of the task that the student is solving.
- Time: The timestamp of the action performed.
- OP: The operation type that the user has made. It can be one of the following: ConID, MoveTask, StartCompiling, EndCompiling, SubmitCode.

*Data related to the compilation start:*

- Code: It contains the programming code to be compiled.
- Lang: The environment that will handle the compilation of the code.
- stdin: The input to the compilation phase.

*Data related to the end of the compilation:*

- Response: It basically says if the compilation succeeded or failed. Therefore, its values are: success or error.
- Startime: The timestamp when the compilation started.
- stdout: The output of the code after compilation and execution.
- stderr: The error message if the compilation fails.

**Students Information.** The source of this dataset is diverse. Each entry in this dataset represents a student defined by this set of information:

- Personal Information: Such as the Name, Email address, Age, Gender, and the assigned userID
- Years of programming.
- Familiar Programming languages.
- Average number of days per week doing programming.
- Several questions about how they deal with programming errors.
- Feedback about the experiment and the exercises.
- Score: It is possible to recover the students' exam scores in the programming course.

### 5.3 Objectives

Using TrackThinkTS, we want to track the students' search behavior and thought process when they have to solve programming problems. Firstly, we want to investigate the difference in successful web searches between the high-performing students and the others. In fact, based on the diverse studies related to web search behavior [18,23,24], cognitive abilities and skills related to web search expertise have an influence on the success of the web search. Such meta-knowledge and search expertise can be shared and transferred [15]. Therefore, we aim at finding the patterns of search expertise expressed by successful students and transfer them to less successful students by means of course supplements to help them acquire the tools for an independent self-improvement. By using the exam score and the dataset from TrackThinkTS, we can compare the search behavior between successful students and the others. Moreover, in our case, measuring web search expertise is accomplished by finding the successful web searches in the context of programming errors. Accordingly, we also acquire the online resources used by the students to solve their programming errors. Thanks to the fine-grained data gathered by TrackThinkTS, we can store the exact information or code snippet that helped the students solve their errors. If we synchronize this type of data with the stack trace that we recover from the C2Room IDE, it is possible to provide timely help to future programming students.

### 5.4 Data Aggregation and Synchronization

So far, the research experiments were conducted using an online IDE. It is possible to recover the build status and executions states of the students and synchronize them with the web search logs. This allows to capture the successful build and recover accurately which text or page used was the most helpful to the students.

In addition, refined data aggregation using the scroll behavior and timestamp can lead to advanced metrics. Some of these metrics were used to study users browsing strategies [13]. We are interested in using them in students' web search behavior as well.

**Bounce.** A user's interaction is considered to be a bounce when the engagement time is relatively short and they leave (either switch tab or close it) quickly.

**Shallow Engagement.** This type of engagement refers to the user reading less than half of the page's content.

**Deep Engagement.** This engagement implies that the user did read more than half of the page's substance.

**Complete Engagement.** The strongest engagement is achieved when the user reads most of the content and decides to save it either as a bookmark, keep the tab open or save it somewhere else.

## 6 Conclusion

In this paper, we have introduced the TrackThinkTS browser extension. It aims at tracking the users' web search and browsing behavior while emphasizing on keeping the user's privacy. We achieve this by using the browser extension API that allows us to capture events related to tabs management, clipboard usage, and web pages meta-information. Having access to such data raises concerns about privacy. Therefore, we provide an aggregated view of the saved logs for easier management. In addition, users can export the logs, and before submitting them, they can access and manage the raw dataset.

Giving extensive control on the log data to the users might alter the viability of the dataset. To address this issue, we recommended the participants to install different browsers than the ones they usually use in their daily life, and use them to install TrackThinkTS and to proceed with the experiment.[3] In such a case, the users won't need to use the management functionality and remove personal websites and irrelevant URLs. However, even if the students follow this recommendation, there are still a few cases where they need to remove some logs. For example, if a user opens a media website to listen to music while working, he/she has the ability to remove the corresponding logs.

Recently, TrackThinkTS has been accepted in the Chrome extension store and has been used in experiments involving programming students. The objective is to build a system that provides course supplements to students who need support. The course supplement should provide timely suggestions to the students when they face build errors while solving programming problems. But also, the course supplement, eventually, supplies information and meta-knowledge of how to search for the appropriate solution.

**Acknowledgment.** This work was partially supported by JST CREST "Behavior change and harmonious collaboration by experiential supplements" (JPMJCR16E1).

[3] https://ubi-naist.github.io/TrackThink/en/.

## References


1. Bae, J., Setlur, V., Watson, B.: GraphTiles: a visual interface supporting browsing and imprecise mobile search. In: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI'15, pp. 63–70. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2785830.2785872
2. Bai, G.R., Kayani, J., Stolee, K.T.: How graduate computing students search when using an unfamiliar programming language. In: Proceedings of the 28th International Conference on Program Comprehension (2020)
3. Bailey, E., Kelly, D.: Developing a measure of search expertise. In: Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval, CHIIR'16, pp. 237–240. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2854946.2854983
4. Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., Klemmer, S.R.: Two studies of opportunistic programming: interleaving web foraging, learning, and writing code, pp. 1589–1598. Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1518701.1518944
5. Dehghani, M., Jagfeld, G., Azarbonyad, H., Olieman, A., Kamps, J., Marx, M.: On search powered navigation. In: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR'17, pp. 317–320. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3121050.3121105
6. ud Din, I., Khusro, S., Ullah, I., Rauf, A.: Semantic history: ontology-based modeling of users' web browsing behaviors for improved web page revisitation. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) CoMeSySo 2018. AISC, vol. 860, pp. 204–215. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-00184-1_19
7. https://googleblog.blogspot.com/2011/10/fall-sweep.html . Accessed 26 Aug 2021
8. Hora, A.: Googling for software development: what developers search for and what they find. In: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pp. 317–328 (2021). https://doi.org/10.1109/MSR52588.2021.00044
9. Hsieh-Yee, I.: Research on web search behavior. Libr. Inf. Sci. Res. **23**(2), 167–185 (2001)
10. Huang, J., Efthimiadis, E.N.: Analyzing and evaluating query reformulation strategies in web search logs. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM'09, pp. 77–86. Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1645953.1645966
11. Hölscher, C., Strube, G.: Web search behavior of internet experts and newbies. Comput. Netw. **33**(1), 337–346 (2000)
12. Kim, J., McNally, B., Norooz, L., Druin, A.: Internet search roles of adults in their homes, pp. 4948–4959. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3025453.3025572
13. Liu, C., Liu, J., Wei, Y.: Scroll up or down?: using wheel activity as an indicator of browsing strategy across different contextual factors. In: Nordlie, R., Pharo, N., Freund, L., Larsen, B., Russel, D. (eds.) Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval, CHIIR 2017, Oslo, Norway, 7–11 March 2017, pp. 333–336. ACM (2017). https://doi.org/10.1145/3020165.3022146

14. Morris, D., Ringel Morris, M., Venolia, G.: SearchBar: a search-centric web history for task resumption and information re-finding. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'08, pp. 1207–1216. Association for Computing Machinery, New York (2008). https://doi.org/10.1145/1357054.1357242
15. Morris, M.R., Moraveji, N., Morris, D.: Supporting the social transfer of web search expertise. In: CHI 2010 Workshop on the Next Generation of HCI and Education. ACM, April 2010. https://www.microsoft.com/en-us/research/publication/supporting-social-transfer-web-search-expertise/
16. Nagano, K., Arakawa, Y., Yasumoto, K.: TrackThink: a tool for tracking a thought process on web search. In: Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers, UbiComp'17, pp. 681–687. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3123024.3129267
17. Rahman, M.M., Roy, C.: SurfClipse: context-aware meta-search in the IDE. In: 2014 IEEE International Conference on Software Maintenance and Evolution, pp. 617–620 (2014)
18. Rahman, M.M., et al.: Evaluating how developers use general-purpose web-search for code retrieval. In: Proceedings of the 15th International Conference on Mining Software Repositories, SR'18, pp. 465–475. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3196398.3196425
19. Sadowski, C., Stolee, K.T., Elbaum, S.: How developers search for code: a case study. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 191–201. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2786805.2786855
20. https://searchcode.com/ . Accessed 26 Aug 2021
21. https://sourcegraph.com . Accessed 26 Aug 2021
22. Stolee, K.T., Elbaum, S., Dobos, D.: Solving the search for source code. ACM Trans. Softw. Eng. Methodol. **23**(3), 1–45 (2014). https://doi.org/10.1145/2581377
23. White, R.W., Morris, D.: Investigating the querying and browsing behavior of advanced search engine users. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07, pp. 255–262. Association for Computing Machinery, New York (2007). https://doi.org/10.1145/1277741.1277787
24. Xia, X., Bao, L., Lo, D., Kochhar, P.S., Hassan, A.E., Xing, Z.: What do developers search for on the web? Empir. Softw. Eng. **22**(6), 3149–3185 (2017)
25. Xu, L., Fernando, Z.T., Zhou, X., Nejdl, W.: LogCanvas: visualizing search history using knowledge graphs, pp. 1289–1292. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3209978.3210169