Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 13, Number 1, Spring 2023, pp. 65-78
Received: 06/28/2023, Revised: 05/01/2024, Accepted: 08/26/2024

Challenges in Developing Progressive Web Applications: An Empirical Study Using Stack Overflow

Mohammadreza Abbasnezhad [1], Amir Jahangard Rafsanjani [2*] and Amin Milani Fard [3]
1- Department of Computer Engineering, Yazd University, Yazd, Iran.
2*- Department of Computer Engineering, Yazd University, Yazd, Iran.
3- Department of Computer Science, New York Institute of Technology, Vancouver, BC, Canada.
[1] abbasnezhad.m.r@stu.yazd.ac.ir, [2*] jahangard@yazd.ac.ir, and [3] amilanif@nyit.edu

Corresponding author's address: Amir Jahangard Rafsanjani, Faculty of Computer Engineering, Yazd University, Yazd, Iran.

Abstract- Progressive Web Apps (PWA) are a new web development paradigm that is being used in a variety of web apps. With this new paradigm, web apps can provide features similar to native apps, blurring the line between websites and native apps. However, new challenges may arise during the development of PWAs. Understanding the issues that developers may encounter while developing PWAs is necessary due to their widespread interest. These issues have not yet been extensively identified and analyzed in the existing research. Therefore, this research aims to fill this knowledge gap by conducting an empirical study of PWA-related questions on Stack Overflow (SO), a well-known website for developers. Our process involves obtaining a list of PWA tags and extracting 8,654 questions related to PWAs from SO. To comprehend the issues being discussed by PWA developers on SO, we use topic modeling. After that, we analyze their characteristics (such as the challenges' popularity and difficulty). Our research shows that developers encounter eleven challenges, including the Service Worker lifecycle. Our study deepens our understanding of PWA development issues and provides recommendations for researchers, educators, and developers to promote PWA adoption.

Keywords- Progressive Web Application; Empirical Study, PWA development challenge, LDA topic modeling, Stack Overflow.

I. INTRODUCTION

Progressive Web Applications (PWA) represent a novel approach to web app development. PWAs are regular websites that have extra features, like the ability to work offline. When compared to traditional web apps, the benefits of PWAs are evident: they can be launched from the device's home screen (like native apps), they load immediately regardless of network connectivity, and they support push notifications [1]. PWAs are another example of cross-platform development [2], [3]. Many businesses have done well with PWAs and have shared their success stories along with their technical improvements [4].

PWAs enhance user experience, reliability, and performance, but their development may pose challenges for developers. In fact, because it takes time to become acquainted with a new technology, it may naturally offer new hardships to developers in its early stages. PWAs use new HTML5 and JavaScript features such as Service Workers (SW) [5]; consequently, their development challenges may differ from those of traditional web apps [6]. From this point of view, it is interesting to look at the problems developers face when making PWAs. Prior research [7]–[9] examined the security and privacy

concerns associated with PWAs. They also investigated the energy efficiency of PWAs on mobile devices [2], [3], [10], [11]. However, an in-depth analysis of the challenges encountered when developing PWAs is lacking.

Stack Overflow (SO) is a popular question-and-answer community for developers where developers frequently post questions about their problems in order to receive assistance from community experts. SO now has over 100 million monthly visitors, 21 million registered users, and over 23 million questions to discuss technical issues in software development [12]. As a result, it is anticipated that the topics discussed on SO will reflect the various challenges faced by the development community. In many fields, such as Mobile apps [13] and Big data [14], it has been a common practice for researchers to comprehend the interests and difficulties of developers through SO questions.

None, however, looked at the difficulties in developing PWAs. This study fills in a gap in the literature by looking into the questions that developers on SO have asked about PWAs. In other words, our study collects and analyzes SO questions pertaining to PWAs in order to comprehend the potential issues and challenges that developers may face when developing PWAs. From now on, the term "topic" refers to the challenges, issues, or difficulties that developers confront and discuss on SO. In short, the following research questions were the object of our study:

- **RQ1: PWA challenges.** What PWA topics are developers asking about? This RQ aims to find specific problems that developers face when developing PWAs by looking at the topics and issues that developers talk about.
- **RQ2: Popularity level.** How popular are PWA challenges for developers? This RQ aims to identify the PWA issues that developers are most likely to encounter.
- **RQ3: Difficulty level.** How difficult are PWA challenges for developers? This RQ seeks to determine which PWA development challenges are the most difficult.

We take the following actions to respond to our RQs: We extract 8,654 SO questions related to PWAs by developing a set of tags related to PWAs based on heuristic metrics employed in related research [15]–[17]. Latent Dirichlet Allocation (LDA) [18] is a topic modeling method that we use to answer PWA questions. As a consequence, we identify problems with PWAs that developers encounter. In order to measure the popularity and difficulty of these topics, we employ well-known metrics that have been proposed and utilized in prior research [13], [14], [19].

Our research yielded the following results: **(1) PWA challenges.** We discovered that PWA developers inquire about 11 topics (for example, SW Lifecycle). **(2) Popularity level.** PWA issues (such as SW Lifecycle) are among the most popular topics. **(3) Difficulty level.** Some PWA issues, such as Installation, are among the most difficult topics. In addition to the identified PWA topics in SO, the evolution of PWA topics in SO is discussed. In addition, we compare the popularity and difficulty of PWA topics to other mature software engineering domains. Our findings indicate that the PWA community requires additional effort to attain the same level of maturity as comparable software engineering fields.

We believe that as PWAs become more popular, their difficulties will intensify. We anticipate that this research will yield insights that will help mitigate these issues in the future. This study's findings benefit developers, educators, and researchers by informing them of the current state of PWA development issues in practice. In other words, this study can help web developers prepare for similar challenges and enable educators and researchers to provide more targeted assistance to PWA developers.

Our research specifically leads to the following contributions:

- We perform an empirical study on SO to investigate PWA-related questions. To the best of our knowledge, this is the first study to look at PWA question characteristics on SO. We list 11 issues with PWA development. We conduct a series of analyses on the extracted challenges, such as the popular and difficult topics, to illustrate the growth and trends of PWA developer discussions.
- Based on our findings, we present actionable implications for researchers, educators, and developers to help them better select when and where to focus their efforts on learning or building PWAs.

The rest of this paper is structured as follows: Section 2 provides fundamental PWA concepts. Section 3 discusses the work that is related to our study. The fourth section explains our methodology. Section 5 contains the results of our empirical investigation. The sixth and seventh sections discuss our results and their implications, respectively. Section 8 finishes with a summary of our work and future research.

II. BACKGROUND

PWAs are a new approach to web app development that was initially announced in 2015 [20]. In fact, PWAs are a new way of thinking about the web that builds on standard

web tools like HTML5, CSS3, and JavaScript. In addition to a website, PWAs provide an extended set of features that facilitate a native user experience for web apps. A PWA differs from a standard website in that it is available offline, has an installation procedure, push notifications [1] and background sync [21]. As PWAs are enhanced websites, they can be executed on mobile devices and desktop computers, regardless of platform. The PWA is therefore seen as a cross-platform development technique [2], [3]. The web manifest file [22] and SW [5] implementations serve as the foundation for the majority of the PWAs' enhanced functionalities.

A SW is a piece of JavaScript code that runs concurrently with the main page, enabling persistent background processing, and interacts with the rest of the PWA in an event-driven manner. When a PWA is viewed for the first time, the SW is loaded and installed in the background.

The PWA's network calls can be intercepted by a SW, which can then use caching to make the app available offline. For instance, as depicted in Figure 1, a SW can either load resources from the cache [23] or acquire them via the network based on network conditions. These programmable interfaces significantly enhance both online and offline web browsing experiences. Additionally, installed PWA can manage push notifications.

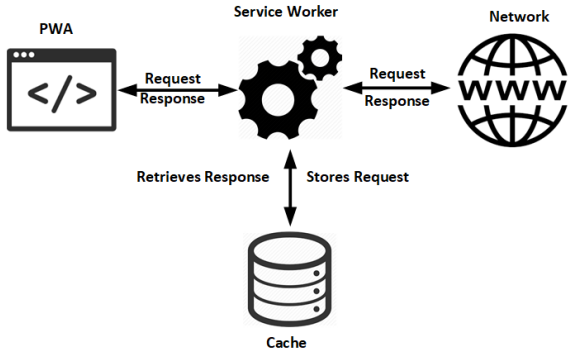


Fig. 1: An illustration of SW as a proxy

PWAs have proven to be beneficial to many enterprises [4]. For instance, Tinder's new PWA reduced load times from 11.91 seconds to 4.69 seconds [24]. In addition to a 44% increase in user-generated ad revenue, Pinterest's new PWA has led to a 40% increase in time spent on the site [25]. With their new PWA, Twitter saw a 65% increase in the number of pages per session and a 75% increase in the number of Tweets [26].

As we can see, PWAs, as a new technology, involve several new concepts, such as SW, which may cause developers difficulty while developing PWAs. PWAs have received a lot of attention; however, these development issues have not yet been taken into account. By focusing on PWA developers' questions on SO, we seek to present a systematic analysis of the difficulties in developing PWAs in this study.

III. Related Work

The research relevant to our work is comprised of PWA-related studies and studies that use data from Stack Overflow to comprehend the interests and challenges of developers.

A. Progressive web app studies

The studies on PWAs by Birn-Hansen et al. [27] and Majchrzak et al. [28] have concentrated on how PWAs can serve as a technically viable alternative to standard web and native apps. They focused on PWAs with multiple platform support, an improved user experience, and better maintenance.

The energy usage of PWAs on mobile devices has been the subject of numerous studies. An empirical study evaluating the effect of SWs and caching status on the energy usage of PWAs has been published by Malavolta et al. [10], [11]. Their research did not yield enough statistical proof to support the possibility that caching or SWs may affect PWAs' energy efficiency. Furthermore, Huber et al. [2], [3], compared the energy efficiency of PWAs to other mobile development methodologies based on interaction scenarios with user interface rendering. According to their research, PWAs are a viable alternative to other cross-platform development strategies in terms of energy efficiency.

There are several studies that target the security of PWAs. Lee et al. [7] are the first to propose novel attacks on the Cache and Push notification APIs in PWAs. Chinprutthiwong et al. [8] also discovered a new XSS vulnerability, which allows web attackers to compromise a benign SW during the SW registration process. Elsewhere, Chinprutthiwong et al. [9] have examined how the IndexedDB [29] and Push notification API can be used by XSS attackers to leverage a benign SW. test

To the best of our knowledge, our study can be considered a step further with respect to the research related to PWAs by studying PWA development challenges. In fact, the popularity of PWAs in academia and business has prompted us to carefully analyze the problems and difficulties PWA developers face when implementing PWAs.

Fauzan et al. [6] conducted a comprehensive examination of PWAs with the objective of building a

thorough understanding through existing research. They searched for and organized literature on PWAs for their study. While Fauzan's paper offers a broad review, categorizing existing literature into thematic groups such as security, ours uniquely focuses solely on the development challenges of PWAs. We diverge by zooming in on the specific hurdles encountered during PWA development.

B. Studies using Stack Overflow

SO has a lot of questions and answers that cover many different parts of software development. Many empirical studies have been conducted using SO data.

Yang et al. [30] investigated the challenges that developers have encountered when utilizing Git commands. They conducted a study based on questions about Git commands on SO. Their research uncovered the most popular and difficult Git commands. Abdellatif et al. [31] investigated the difficulties of creating Chatbots. They gathered questions from Chatbot developers on SO for their study. Then, they utilized the LDA model [18] to extract topics and challenges from the questions. They highlighted the most difficult and popular Chatbot development challenges.

Chakraborty et al. [16] used the LDA method on SO's questions about Swift, Go, and Rust, which are all new programming languages. They revealed the challenges encountered by developers working with these languages. Han et al. [32] used LDA to conduct a study on SO to identify the topics discussed among developers concerning three deep Learning frameworks (Tensorflow, PyTorch, and Theano). Kamienski and Bezemer [33] analyzed question-and-answer websites geared toward game development, such as questions about game development from SO. Using LDA, they classified the game development topics discussed. Uddin et al. [19] used LDA to analyze IoT-related discussions on SO to identify IoT topics and provide insights into their popularity and difficulty. Similarly, research has been conducted in the following areas of software engineering: mobile apps [13], big data [14], security [34], and concurrency [17].

More specifically, Scoccia et al. [15] explored the difficulties that developers face while implementing desktop web apps. In addition, Bajaj et al. [35] investigated the typical challenges encountered by web developers. However, previous research has not looked into PWA-related questions on SO. We believe that by analyzing PWA-related questions, our study complements previous work in SO.

IV. METHODOLOGY

This study's primary objective is to discover the most popular and difficult PWA issues or challenges that developers have been discussing. In this section, we will describe in detail the steps taken in accordance with previous studies [13], [14], [31] to attain this study's primary objectives.

Figure 2 illustrates the study's methodology. Our methodology consists of (1) Data collection, which extracts SO questions related to PWAs based on the identification of PWA-related tags; (2) Data preprocessing, which preprocesses the SO questions; (3) PWA topic identification, which uses LDA to identify the issues related to PWA development; and (4) Topic analysis, which analyzes identified issues to determine their popularity and difficulty. The steps that comprise our methodology are detailed in the subsections that follow.

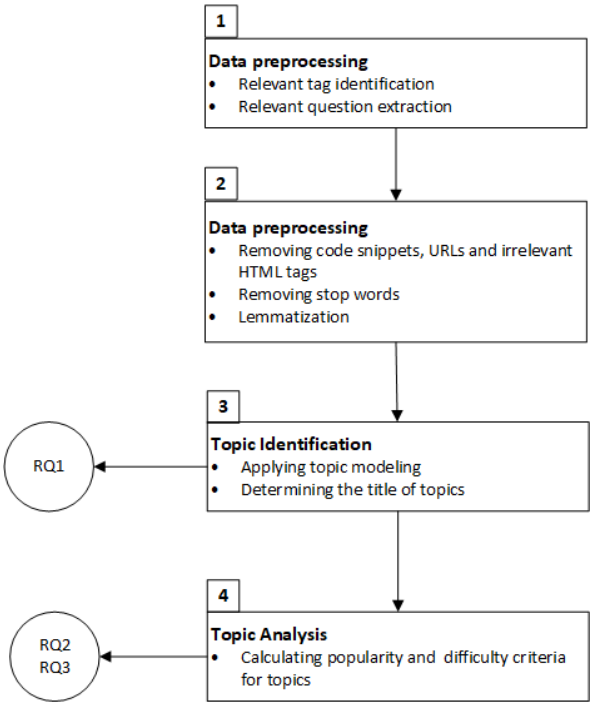


Fig. 2: An overview of the methodology.

A. Data Collection

We turn to studying the developers' questions on SO to achieve our goals because of its rich dataset and application in relevant investigations in other disciplines. One method to access SO data is via the Stack Exchange API[1], which allows programmatic access to all Stack Exchange websites, including SO. The API, for example, allows users to obtain questions, answers, comments, and

[1] https://api.stackexchange.com/

badges from SO. SO does not have any fine-grained topic information about PWAs. Therefore, we must first identify the SO questions that pertain to PWAs. In this section, we will explain how we discovered the tags related to PWAs in order to extract PWA-related questions from SO.

B. Relevant tag identification

We use the methodology from earlier work [15]–[17] to find the most pertinent PWA-related tags. The objective of this technique is to choose tags that are pertinent to PWAs while minimizing dataset noise. The following is the detailed procedure:

Since this study is primarily concerned with PWA (also known as "progressive web apps" or "pwa"), we use T_i = {"progressive-web-apps", "pwa"} as our initial tag set. It is worth noting that all SO questions are designated as Q_{SO}. Then, we retrieve all questions from SO whose tags match at least one tag from T_i (these questions are denoted as Q_i). We do not add any additional tags in this first phase to avoid generating noise, as this will detect other PWA-related tags. Second, we extract all candidate tags that coexist with the T_i tags in the Q_i questions (denoted by T_c). Following that, we employ two heuristic measures to generate a larger set of PWA-related tags.

The first metric is the Tag Relevance Threshold (TRT), which measures the relevance of a tag to PWA-tagged questions. This measure investigates how many of a tag's questions are about PWA compared to how many questions are for that tag as a whole. In particular, the TRT is calculated using:

$$TRT = \frac{\#\ of\ questions\ with\ tag\ t\ in\ Q_i}{\#\ of\ questions\ with\ tag\ t\ in\ Q_{SO}} \quad (1)$$

We can delete irrelevant tags from our set by utilizing the TRT. Additionally, we employ a second metric called the Tag Significance Threshold (TST), which assesses the prominence of a particular tag in PWA-tagged questions. This metric is calculated by comparing the total number of PWA questions for a given tag to the total number of PWA questions obtained using T_i. TST is measured specifically using:

$$TST = \frac{\#\ of\ questions\ with\ tag\ t\ in\ Q_i}{\#\ of\ questions\ in\ Q_i} \quad (2)$$

If a tag's TRT and TST are both above a certain threshold, we think it is important and related to the PWA questions. We experimented with various TRT and TST values and discovered that TRT = 0.001 and TST = 0.1 produce a highly meaningful collection of tags. It is crucial to note that our thresholds are consistent with prior research that employed the same approach [14], [16], [17], [19]. Finally, we use the chosen TRT and TST criteria to identify our tag set, which includes {"progressive-web-apps", "pwa", "service-worker", "workbox", "angular-service-worker", "angular-pwa", "workbox-webpack-plugin", "service-worker-events", "sw-precache", "background-sync", "next-pwa", "pwabuilder", "beforeinstallprompt", "sw-toolbox", "push-api", "cacheapi", "web-manifest", "web-app-manifest", "cachestorage", "web-share", "a2hs", "service-worker-config", "vue-pwa"}.

C. Relevant question extraction

We use the PWA-related tag set we obtained to extract the questions that will comprise our PWA dataset throughout this research. We get this information by querying all questions on SO that use one of the tags in our tag set; this is done via the Stack Exchange API. This procedure resulted in a dataset including 8,654 PWA questions and their associated metadata. Our dataset was compiled in April 2023, and it contains questions until January 2023.

We collected both the question's textual content (including its title and body) and its metadata (including its identifier, creation date, view count, score, and accepted answer identifier). It is important to remember that despite the fact that a question may receive multiple responses, the person who posed the question is the only person who can label a response as accepted, provided it satisfactorily resolves the original questioner's question. As a result, depending on the questioner, questions may or may not have an accepted answer.

D. Data Preprocessing

Before we use the topic modeling technique, we remove information that is not important. In our analysis, we concentrate on the title and content of the questions. If an accepted answer body exists for a question, we additionally consider it.

Many questions contain source code (marked by the <code></code> tag). Our study does not require source code fragments; thus, they were eliminated. Also, the HTML tags (e.g., <p> or <a>) and URLs were removed as they are irrelevant to our analysis and may add unnecessary data. Then, we perform stopword removal using the NLTK stopword corpus [36]. That is the process of deleting words often used in English, such as 'is', 'of', and 'at', which do not significantly change the semantics of a sentence but might potentially generate noise and impede the process of differentiating between topics. In addition, we lemmatize the terms in order to map them to their root (for example, "development" is mapped to "develop"). Following these procedures, we have a preprocessed dataset that can be directly fed into the topic modeling method.

E. Topic Identification

We describe step 3 of our methodology for RQ1, which is depicted in Figure 2, in this subsection. To identify the subjects that PWA developers discuss on SO, we employ the LDA modeling technique [18], which has been extensively utilized in software engineering studies [13], [14], [37]. Based on the word frequencies and co-occurrences in the questions, LDA categorizes the questions in our dataset into a set of categories. The probability that a question will be related to a certain topic is indicated by a set of probabilities that LDA assigns to each question (one for each topic). The topic of a question—or, more precisely, the topic that dominates the question—is the one that has the highest probability for that particular question. In our methodology, we employ the MALLET LDA implementation [38].

The number of topics must be specified as an input parameter to LDA. We use topic coherence to estimate the number of topics. One of the metrics often used to quantify the quality of extracted topics is topic coherence [39], which has been proven to be substantially linked with human understandability [40]. We conducted experiments with a number of topics to identify the one with the highest coherence score. In fact, we experiment with a varying number of topics, ranging from 5 to 25e in increments of 1, and compute the coherence metric across all output topics. Then, we choose the final number of topics by choosing the candidate value that has the best coherence score. We define the number of topics as 11 based on this technique.

The LDA does not produce meaningful names for the topics it creates; instead, it produces a list of keywords for each topic. For the purposes of our analysis, we must be able to understand the logic that links the LDA algorithm's found questions in each topic and describe it using a descriptive term. Topic titles are assigned based on a list of the top words most frequently occurring in each topic, as well as manually inspecting the ten most relevant questions for a topic (i.e., the ones with the highest probability of belonging to the topic), as well as a randomly selected sample of ten questions for a topic. In this way, we can better understand the shared rationale behind the topic's questions and gain new insights.

F. Topic Analysis

In this phase, we leverage metrics from previous work [13], [14], [19] to assess the popularity of PWA topics, i.e., RQ2. The first metric is the average number of views on the topic's questions. Since there are many more visitors than registered users on SO, views from both registered users and non-users are important. The second metric is the average score on questions pertaining to the topic. A PWA with a greater number of views and a higher score is, intuitively, more popular. Overall, these measurements show how interested the community is by telling us how often a certain question comes up. In other words, the assumption is that if a topic is extremely popular, then the issue is widespread among developers. We compute these metrics based on the associated attributes of the questions, namely the number of views and the score.

We initially compute the two popularity metrics for the topics in order to determine the popularity of topics similar to [19]. The topic's metric value is then normalized by dividing it by the sum of the metric values for all the topics. We finally average the two normalized metrics to determine the topic's popularity.

In addition, we use two metrics from earlier work [13], [14], [34] to assess the difficulty of PWA topics, namely RQ3. We calculate the percentage of questions with no accepted answers for each PWA topic. The topics with fewer accepted answers are regarded as more difficult. In addition, the median time in hours for questions to receive an accepted answer is measured. This metric takes into account the time the acceptable answer was created, rather than the time it was tagged as accepted. A question becomes harder the longer it takes to acquire an accepted answer and be correctly answered. We applied the same method used to determine topic popularity with its two metrics to determine the topic's difficulty.

V. RESULTS

We list the answers to our three research questions in this section.

A. RQ1: PWA Challenges

Table 1 displays the 11 resulting topics, as well as the top keywords generated automatically by the LDA algorithm. It also displays the percentage of questions that fall under each topic. As shown in the table, developers inquire about various aspects of PWA development, and the percentage of questions varies by topic. Using examples selected from the SO, we will now elaborate on the emergent topics in greater depth.

1. **Debugging and Testing.** This topic comprises 8.76% of the questions in our dataset. These questions are about solving issues with debugging and testing PWAs. An issue description in the SO question with id "34160509" is: "I want to test service workers but I have a virtual host setup and I can't seem to be able to enable https on localhost."

TABLE 1: PWA DEVELOPMENT TOPICS DISCUSSED ON SO

Topic No.	Topic name	Keywords	% Questions
1	Debugging and Testing	debug, devtool, test, localhost, fail, site, access, issue, console, error, chrome, firefox,	8.76
2	Serving Requests	request, response, fetch, match, intercept, network, return, url, post, cache, offline, online	9.43
3	Updating	update, version, refresh, reload, change, browser, time, cache, check, load, force	5.62
4	SW Lifecycle	serviceworker, event, trigger, register, sw, handler, listener, install, activate, console	9.07
5	Frameworks	angular, ngsw, build, project, file, config, ionic, module, command, cli, react, create, build, workbox	14.51
6	Push Notification	push, notification, message, receive, send, client, server, subscribe, permission, endpoint	7.77
7	Manifest File	file, manifest, json, scope, url, root, html, folder, path, start_url, register	5.59
8	Cross-platform Development	safari, ios, android, open, support, add, window, redirect, login, native, feature, desktop	17.58
9	Background Sync	background, sync, time, browser, connection, request, open, queue, task, start, close, send	2.47
10	Offline Data	data, store, cache, indexeddb, save, access, localstorage, offline, file, image, load, asset	12.12
11	Installation	install, add, icon, chrome, screen, android, prompt, homescreen, browser, button	7.08

2. **Serving Requests.** The SW receives all PWA requests, including HTTP headers and URLs, and then leaves it up to the developer to decide how to handle them. The developer has the option of sending a request to the network, responding with a previously cached response, or creating a new response. Various questions may arise for developers during the procedure described above, accounting for 9.43% of the questions in our dataset. "I want to intercept the response to a given fetch request before that response is passed back to the page," says an issue description in the SO query with id "68642309."

3. **Updating.** 5.62% of the questions fall under this topic, which is related to difficulties updating PWAs. After PWA is published, some consumers utilize it in their browsers, while others install it on their devices. It is critical to follow recommended practices when updating PWA to avoid issues. In reality, there are numerous processes involved in updating PWAs, including updating PWA data, assets that are already cached on devices, the SW file and its dependencies, or manifest information. Following recommended standards for updating presents difficulties for developers. The issue description in the SO question with id "51748241" is as follows: "How can we refresh the client if service worker file is being updated on the server and client is not being reloaded?"

4. **SW Lifecycle.** The way in which SWs are installed is determined by their lifecycles, which is a different process from the installation of PWAs. SW registration begins the lifecycle. After that, the install event is triggered. The installation of the SW occurs in the background and does not require permission from the user. The SW does not yet have control of the PWA after the installation is complete. The activation event will occur when the SW is prepared to take control of the PWA. Despite this, there is no guarantee that the page that was used to register the SW will remain handled. By default, the SW will not take over until the user reloads the website or reopens the PWA and accesses the same page. Questions regarding these steps account for 9.07% of all questions posed to developers. The following is an excerpt from question with id "40902060" from SO: "I have attached install event to service worker. But Install event fired before register event is completed"

5. **Frameworks.** When creating PWAs and conventional websites, developers have comparable possibilities. To create a PWA, web frameworks such as Angular or React are commonly used. The questions in this category are about constructing PWAs with various development frameworks, specifically about special configurations and features in those frameworks. For instance, a developer posed a question with the id "50330155" and the title "Angular Cli not generating angular-cli.json" on SO. This group is the second largest in our study, accounting for 14.51% of the questions in our dataset. This demonstrates that developers rely largely on frameworks while creating PWAs.

6. **Push Notification.** This topic consists of 7.77% of the questions. Push notifications are used to deliver content from the server to the PWAs. SWs are responsible for their operations. Developers must first obtain permission to send Push notifications. Following that, developers must subscribe to a server that will return the data to the PWA. Developers enable the SW in PWA to receive data from the server, display it to users via

notifications, and handle user events associated with notifications. In the process mentioned above, various questions may be facing developers. An example of a question under this topic is a question with id "51881973", which is "I need to show a push notification once a week, even when the user is not on the page."

7. **Manifest File.** The web manifest file is a JSON file that contains information about a PWA. It is required for the PWA to be downloaded and shown to the user, similarly to a native app. A typical manifest file contains the name of the PWA, the icons that the PWA should use, and the URL that should be opened when the PWA is launched. In our dataset, the Manifest File topic is discussed in 5.59% of questions, which indicates developers have various questions about this file. For example, a developer expressing their problem in a question with the id "45412014", and the title "How do I set the start_url of a manifest.json to be the root of the site?".

8. **Cross-platform Development.** PWAs are available for Android, iOS, and Windows using the same codebase. Moreover, PWAs have enabled features previously only found in natively developed apps, such as installability, offline availability, and push notifications. Therefore, PWAs can be considered a cross-platform development approach. This poses questions for developers, as 17.58% of the questions fall into this topic in our dataset (the largest topic). An instance of this topic is "Given a business use case that needs to delivered to its users via mobile, what questions do we need to ask of a use case to help understand if the use case is a better candidate for Native, Hybrid or PWA?" in question with id "58160072".

9. **Background Sync.** The background sync enables PWAs to postpone server synchronization tasks to their SW for later handling while the PWA is offline. In other words, it can be used to send requests in the background if they cannot be sent while the PWA is running. Thus, the background sync enables users to continue using PWAs and performing actions offline. Background Sync is the topic of 2.47% of the questions in our dataset. An issue description in the SO question with id "64303059" is: "Is there anyway to know service worker has finished all background - sync?".

10. **Offline Data.** Storage management is required for the PWA to create a solid offline experience. IndexedDB and Cache are useful tools in this regard. Developers need to know how to use these tools to handle offline data, so they have a lot of questions. For instance, a developer may post a question with the id "34416537" and the topic "Not able to access indexeddb from service worker" to explain the issue they're having. This is the third largest topic, accounting for more than 12% of the total.

11. **Installation.** This topic accounts for 7.08% of the questions. Installation capability is a feature that enables a user to install a PWA, which is then added as a shortcut to their home screen, allowing them to access it with a single tap. Developers must meet installing criteria and effectively handle installing events from browsers to provide users with an in-app install experience. The following is a sample question pertaining to the installing process; the question id is "72791417", and the issue is: "I want to achive feature for install/open button with PWA application".

B. RQ2: Popularity level

Table 2 displays the popularity of the PWA topics as well as the average number of views and scores earned by developers on SO. SW Lifecycle, Debugging and Testing, Installation, and Updating are the most popular PWA topics on SO. Table 2 demonstrates that the most viewed topics by PWA developers are related to the topic SW Lifecycle. The topic Debugging and Testing contains the questions with the highest scores. The high popularity of SW Lifecycle, Debugging and Testing can be attributed to the fundamental nature of these topics; i.e., any newcomer will seek out these questions before creating their first PWA. Table 2 demonstrates that Offline Data, Serving Requests, and Background Sync are less popular than other topics.

C. RQ3: Difficulty level

Table 2 displays the difficulty of the PWA topics as well as the percentage of questions with no accepted answers ("% w/o acc.") and the median time to get an accepted answer ("Hrs to acc."). The most difficult topics include Installation, Debugging and Testing, and Frameworks. On the other hand, questions concerning the SW Lifecycle and Background Sync are typically simpler for developers to answer.

VI. DISCUSSION

The correlations between topic popularity and difficulty, the evolution of PWA topics, and a comparison of PWA topics to other established software engineering fields are covered in this section.

A. Popularity and difficulty correlation

Table 3 shows that the topic SW Lifecycle is more popular and less difficult. An observation like this could lead to the conclusion that there are links between the popularity and difficulty of PWA topics. To investigate this, we compute four correlations between our PWA topics' popularity and difficulty metrics. We used the

TABLE 2: THE DIFFICULTY AND POPULARITY OF PWA TOPICS.

Topic No.	Topic name	Avg. View	Avg. Score	Popularity	% w/o acc.	Hrs to acc.	Difficulaty
1	Debugging and Testing	2358.78	3.51	1.18	0.66	22.95	1.25
2	Serving Requests	1563.07	2.06	0.73	0.60	10.87	0.82
3	Updating	2377.68	3.18	1.12	0.59	11.63	0.84
4	SW Lifecycle	4022.64	3.36	1.52	0.59	8.85	0.75
5	Frameworks	1775.87	2.29	0.82	0.66	21.93	1.22
6	Push Notification	2509.06	2.82	1.08	0.72	13.80	1.01
7	Manifest File	2681.80	2.50	1.06	0.59	15.19	0.95
8	Cross-platform Development	1716.93	2.69	0.88	0.71	16.05	1.07
9	Background Sync	1316.63	2.39	0.74	0.59	8.16	0.72
10	Offline Data	1417.59	2.15	0.71	0.63	16.81	1.03
11	Installation	2723.17	2.97	1.16	0.69	24.68	1.33
	Average	2223.93	2.72	1.00	0.64	15.54	1.00
	Max	4022.64	3.51	1.52	0.72	24.68	1.33
	Min	1316.63	2.06	0.71	0.59	8.16	0.72

Spearman Rank Correlation Coefficient [41] because it doesn't assume that the data are spread out evenly. Table 3 displays the results in x/y format, where x represents the coefficient and y represents the p-value. Table 3 shows that no corelation between popularity and difficulty is statistically significant (all p-values are greater than 0.05). In other words, difficult topics are not always popular among developers, and vice versa.

B. The evolution of the PWA topics

We calculate the number of questions about PWA topics per year to assess the evolution of PWA issues. Figure 3 shows that all PWA topics have been discussed over the years, and none of them have ever stopped. This indicates that developers are interested in learning more about the numerous PWA development activities represented by the various topics. However, questions regarding Cross-platform Development, Frameworks, and Offline Data have remained the most common over time. This is consistent with real-world observations, as one of the primary advantages of PWAs is their offline accessibility and the ability to develop an app with a singular codebase while still being able to use it on multiple platforms. Even though these topics were popular at first, they are not as popular as they used to be. On the other side, discussions on Debugging and Testing have become more significant recently.

TABLE 3: CORRELATIONS OF POPULARITY AND DIFFICULTY OF PWA TOPICS.

coefficient/p-value		Popularity	
		View	Score
Difficulty	% w/o acc.	0.01/ 0.97	0.05/ 0.89
	Hrs to acc.	0.15/ 0.65	0.14/ 0.69

C. PWA Compared to Other SE Fields

Discussions regarding PWA only began in 2015. We set out to study how the PWA domain relates to other software engineering domains as a new and emerging field. To accomplish this, we examine the metrics reported in the papers that have been investigated similarly in the past. We evaluate PWA metrics (such as difficulty and popularity) in relation to those of other domains: chatbots [31], IoT [19], mobile apps [13], big data [14], security [34] and concurrency [17]. Table 4 displays metrics associated with domain popularity: (1) The total number of questions examined in the study; (2) the average number of views; and (3) the average score. Additionally, the table displays two metrics related to topic difficulty: (1) Percentage of questions without accepted answers; (2) Median number of hours per topic to accept an answer

From the number of analyzed questions, there is a greater number of PWA questions than Chatbot questions (8.6K vs. 3.8K) but it is less than that of the other domains (Big data 125K, Security 94K, and Mobile 1.6M). These numbers show that the PWA is an emerging paradigm.

With regard to the other popularity metrics, PWA-related topics have average views and scores comparable to mobile and security, and to security and concurrency, respectively. This observation can be explained by the fact that PWA is a relatively new field compared to mobile and security. In fact, it could imply that PWAs are crucial from the standpoint of developers.

With respect to the difficulty metrics, PWA shows values similar to IoT, Big data and Chatbot for one metric (% w/o accepted answer). However, PWA has the highest median time (in hours) to get an accepted answer (15.5 for PWA, 14.8 for Chatbot, and 3.3 for Big data). It implies that questions related to PWA need longer time to receive an accepted answer. In other words, it is more challenging to answer questions related to PWA than other fields. This corroborates with the emerging nature of the PWA topic and indicates that much needs to be done to put the PWA development community on pair with other mature fields, such as Security, Concurrency, Mobile with median time less than one hour.

VII. IMPLICATIONS

The results of our study can assist managers, educators, researchers, and developers in choosing more effectively where to concentrate their efforts. In this section, we discuss the implications of our study's results for enhancing the practice and education of PWAs. To better show how popular and hard the PWA topics are, we follow previous work [15], [19], [31] and show in Figure 4 a bubble plot that places the topics based on their average score, which shows how popular they are, and the median number of hours it takes to get an accepted answer, which shows how difficult they are. The size of the bubble indicates the total number of questions associated with a specific topic.

Our research shows that PWA developers discuss concepts like SW Lifecycle, Offline Data, and Installation, which set PWA development apart from traditional web app development. Furthermore, we notice that the topics cover a variety of components and stages of PWA development (for example, Updating, Debugging, and Testing). This demonstrates that developers are confronting a variety of issues and are looking for information on various stages of the PWA development process.

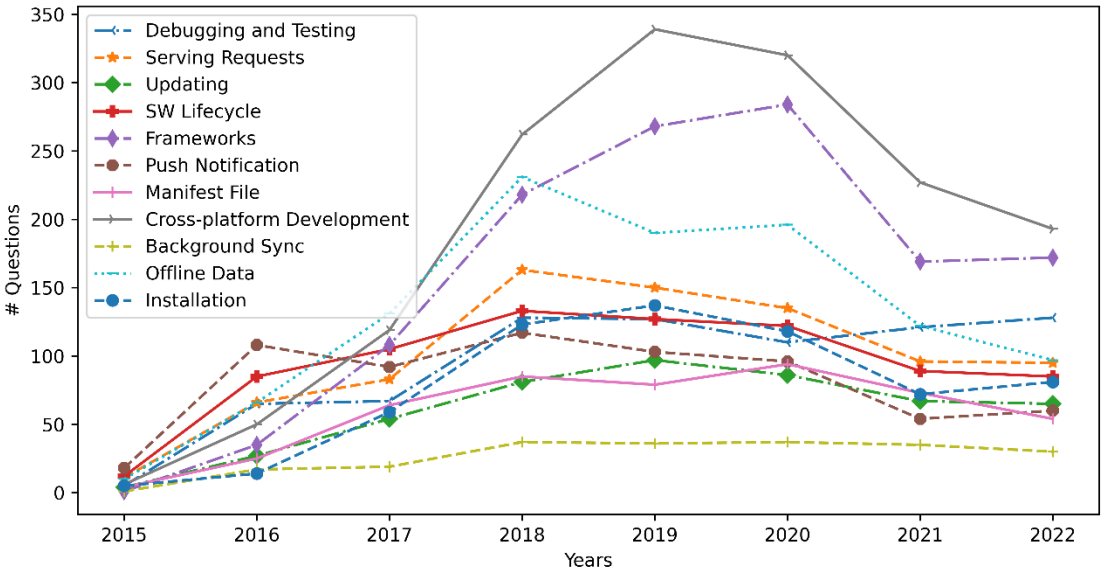


Fig. 3: The evolution of the number of questions by topic overtime.

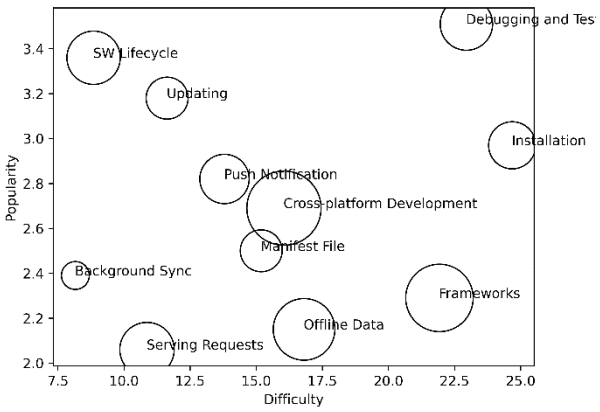


Fig. 4: PWA topics' popularity vs. difficulty

Given the popularity of PWAs on SO, it is obvious that developers need to become more knowledgeable and skilled in creating PWAs. Those with experience in PWAs are well represented in the pool of available software development talent. It has been shown that it is hard to get used to a new field [42]. Developers have the ability to educate themselves on the frequent problems that others have encountered, allowing them to learn about the best practices used by their peers. The results of our research can be used by developers as a checklist for improving their proficiency in PWA in a more directed manner. We have also guessed how long it will take for someone to answer a question and how often they will answer. Since the response time is so long, it is recommended that new PWA developers learn the basics of PWA development in order to increase their chances of finding and fixing their own problems.

We discovered that different PWA topics have varying degrees of difficulty. Some are rare, and others are popular. Knowing this, instructional efforts in these difficult areas can be prioritized. According to Fig. 4, a developer interested in PWA may choose to make a tradeoff between the popularity and difficulty of PWA topics by beginning their learning with the more popular and less difficult SW Lifecycle topic before moving on to the less popular and more difficult Frameworks topic.

Our topics can serve as a guide for instructors as they plan their own courses relating to PWA. The Frameworks topic has a number of discussions about how to set things up correctly and what the best practices are. Because of this, educators can introduce their students to the different frameworks that are already out there and discuss the best practices and standards that should be used during the PWA development process. Additionally, a PWA instructor may choose to allocate more content and teaching time to the topic of Debugging and Testing, which is a more challenging topic than the topic of Background Sync, which is a simpler topic. It is essential for educators to conduct hands-on training programs and take a problem-solving approach while teaching PWAs. This is because PWA topics tend to be difficult.

The most unexpected finding is that, while academic research on PWA security is increasing, there is little discussion on SO about the security implications of PWA. Therefore, developers should put more effort into making their PWAs secure in order to reduce the likelihood of security problems.

Our research serves as either a starting point or a point of comparison for future research. In other words, we comprehend the PWA development challenges by utilizing SO questions. Using additional data sources, such as Github, PWA researchers could investigate the development challenges of PWAs in greater depth. Then, they could compare their results to ours to see if there are any big changes between the two sets. In addition, researchers can investigate PWA learning approaches by surveying developers from both academia and industry to determine how developers learn to develop PWAs, such as through instructional teaching or self-learning via the Internet and documentation. By being aware of this, it is possible to find methods to support them through the learning process in practice, such as by creating well-structured and comprehensive PWA tutorials and documentation with code examples that aim to reduce the difficulties of developing PWAs.

PWA challenges have a high share of queries without accepted answers and the highest median time to acceptable responses, indicating that developers encounter issues with PWAs more frequently. Managers of software can take this into account by giving PWA development more resources (like more time to work on it). Similar to this, a software manager may select more experienced developers to handle PWA development.

Table 4: Comparison of popularity and difficulty between different fields

Type	Metric	PWA	IoT	Big Data	Chatbot	Security	Mobile	Concurrency
Popularity	# Posts	8,654	53,173	125,671	3,890	94,541	1,604,483	245,541
	Avg View	2,224	1,320	1,560	512	2,461	2,300	1,641
	Avg Score	2.7	0.8	1.4	0.7	2.7	2.1	2.5
Difficulty	% w/o acc.	64	64	60	68	48	52	44
	Hrs to acc.	15.5	2.9	3.3	14.8	0.9	0.7	0.7

The decrease in the number of new questions in various topics (Fig. 3) may imply that SO already knows the answer to the majority of the PWA questions or issues. From this perspective, researchers can determine the maturity of PWAs by analyzing their footprint on SO. That can help developers decide whether or not to move to PWAs. In fact, the use of PWAs is strongly correlated with their level of maturity. That means the vast majority of the PWAs' questions have been resolved. The majority of questions have already been answered in this manner, so it is not necessary for the developers to ask a new inquiry. PWA maturity is hard to define. Nonetheless, researchers have the option of employing an indirect method to measure it. In other words, the method of [43] says that they can look at all kinds of interactions to figure out how mature PWAs are. That shows when there are enough knowledge to help PWAs on SO.

SO should refine their strategy to support the growth of PWAs, which are challenging for developers (Table 3). The SO's current approach is to focus on developers who are experts in their fields. To support PWAs, however, they should encourage developers of all levels to respond to questions. It backs up the conclusions of [43], in which they proposed that SO replace the current question-oriented approach with an answer-oriented policy.

More mature software technologies have benefited from empirical and qualitative research on bugs and fault characterization [44]. However, this study looked at the problems of PWA development. As a result of the high difficulty and widespread interest in the Debugging and Testing topic, researchers are able to describe and classify the faults that were found during the creation of PWAs. Such research on the categorization of PWA faults can assist in the improvement of PWA development.

The fact that PWA topics are hard can mean that the general experience of the PWA community is getting worse. Researchers can investigate the backgrounds of developers who contribute to PWAs to determine if the PWA community comprises a few experts or many active developers on SO. In fact, a high number of questions without accepted answers can happen when experienced users stop using the community and when new, less experienced users join. People with more experience quit using the community, but new people did not get enough experience to take their place. Other studies [45] also found that experienced users helped the SO communities by lowering the number of questions that did not have answers. Researchers can discover if experienced users have stopped helping in the PWA community. Like in [45], they can look at community experience to see how user experience has changed over time.

The existence of the Cross-platform Development issue with the highest percentage of questions appears to contradict one of the primary benefits of PWAs: the ability to develop an app in a single codebase while still being used on various platforms. In fact, developers frequently encounter issues when creating PWAs for multiple platforms, as not all PWA behaviors are standardized across browsers and platforms. In a cross-platform development environment, software managers and developers can make more informed decisions when selecting the technologies for their projects with the help of this guide. In other words, PWAs are not the best way to make an app work on more than one device.

VIII. CONCLUSIONS

This study is based on the need to reveal what developers are interested in and what problems they face when developing PWAs. To achieve this, we used a set of tags to extract PWA-related questions from the most prominent online developer forum, Stack Overflow. The Latent Dirichlet Allocation topic modeling technique generated the topics that PWA developers discussed. We found 11 main topics (such as Service Worker Lifecycle). The analysis of these topics yielded numerous crucial insights regarding the PWA-related interests of developers, including the most popular and difficult issues. We have also investigated how the issues facing developers regarding PWAs have evolved over time. The purpose of our study is to provide researchers, educators, and developers with actionable implications that underscore best practices and promising research avenues in PWAs.

We feel that this study can be enhanced in the future by incorporating data from other development communities, such as GitHub. In addition, we intend to conduct a structured survey with both open source and industry software developers. The survey will investigate their general and particular difficulties when developing PWAs. This survey will supplement and validate our Stack Overflow research.

REFERENCES

[1] "Push API." https://w3c.github.io/push-api/ (accessed Mar. 02, 2023).

[2] S. Huber, L. Demetz, and M. Felderer, "PWA vs the Others: A Comparative Study on the UI Energy-Efficiency of Progressive Web Apps," in *Web Engineering*, M. Brambilla, R. Chbeir, F. Frasincar, and I. Manolescu, Eds., Cham: Springer International Publishing, 2021, pp. 464–479.

[3] S. Huber, L. Demetz, and M. Felderer, "A comparative study on the energy consumption of Progressive Web Apps," *Inf. Syst.*, p. 102017, 2022, doi: https://doi.org/10.1016/j.is.2022.102017.

[4] "PWA Stats." https://www.pwastats.com/ (accessed May 05, 2022).

[5] "Service Workers Nightly." https://w3c.github.io/ServiceWorker/ (accessed May 05, 2023).

[6] R. Fauzan, I. Krisnahati, B. D. Nurwibowo, and D. A. Wibowo, "A systematic literature review on progressive web application practice and challenges," *IPTEK J. Technol. Sci.*, vol. 33, no. 1, pp. 43–58, 2022.

[7] J. Lee, H. Kim, J. Park, I. Shin, and S. Son, "Pride and Prejudice in Progressive Web Apps: Abusing Native App-like Features in Web Applications," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, in CCS '18. New York, NY, USA: Association for Computing

Machinery, 2018, pp. 1731–1746. doi: 10.1145/3243734.3243867.

[8] P. Chinprutthiwong, R. Vardhan, G. Yang, and G. Gu, "Security Study of Service Worker Cross-Site Scripting.," in *Annual Computer Security Applications Conference*, in ACSAC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 643–654. doi: 10.1145/3427228.3427290.

[9] P. Chinprutthiwong, R. Vardhan, G. Yang, Y. Zhang, and G. Gu, "The Service Worker Hiding in Your Browser: The Next Web Attack Target?," in *24th International Symposium on Research in Attacks, Intrusions and Defenses*, in RAID '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 312–323. doi: 10.1145/3471621.3471845.

[10] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic, "Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps," in *IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems*, 2017, pp. 35–45. doi: 10.1109/MOBILESoft.2017.7.

[11] I. Malavolta, K. Chinnappan, L. Jasmontas, S. Gupta, and K. A. K. Soltany, "Evaluating the impact of caching on the energy consumption and performance of progressive web apps," in *IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*, in MOBILESoft '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 109–119. doi: 10.1145/3387905.3388593.

[12] "All Sites - Stack Exchange." https://stackexchange.com/sites (accessed Aug. 26, 2022).

[13] C. Rosen and E. Shihab, "What are mobile developers asking about? A large scale study using stack overflow," *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1192–1223, 2016, doi: 10.1007/s10664-015-9379-3.

[14] M. Bagherzadeh and R. Khatchadourian, "Going Big: A Large-Scale Study on What Big Data Developers Ask," in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, in ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 432–442. doi: 10.1145/3338906.3338939.

[15] G. L. Scoccia, P. Migliarini, and M. Autili, "Challenges in Developing Desktop Web Apps: a Study of Stack Overflow and GitHub," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, May 2021, pp. 271–282. doi: 10.1109/MSR52588.2021.00039.

[16] P. Chakraborty, R. Shahriyar, A. Iqbal, and G. Uddin, "How do developers discuss and support new programming languages in technical Q&A site? An empirical study of Go, Swift, and Rust in Stack Overflow," *Inf. Softw. Technol.*, vol. 137, 2021, doi: 10.1016/j.infsof.2021.106603.

[17] S. Ahmed and M. Bagherzadeh, "What Do Concurrency Developers Ask about? A Large-Scale Study Using Stack Overflow," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, in ESEM '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–10. doi: 10.1145/3239235.3239524.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[19] G. Uddin, F. Sabir, Y.-G. Guéhéneuc, O. Alam, and F. Khomh, "An empirical study of IoT topics in IoT developer discussions on Stack Overflow," *Empir. Softw. Eng.*, vol. 26, no. 6, p. 121, 2021, doi: 10.1007/s10664-021-10021-5.

[20] "Progressive Web Apps (Chrome Dev Summit 2015)." https://developers.google.com/web/shows/cds/2015/progressive-web-apps-chrome-dev-summit-2015 (accessed May 27, 2023).

[21] "Background Synchronization API - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Background_Synchronization_API (accessed Jun. 26, 2023).

[22] "Web Application Manifest." https://w3c.github.io/manifest/ (accessed Sep. 05, 2023).

[23] "Cache - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Cache (accessed May 27, 2023).

[24] "A Tinder Progressive Web App Performance Case Study | by Addy Osmani | Medium." https://medium.com/@addyosmani/a-tinder-progressive-web-app-performance-case-study-78919d98ece0 (accessed May 21, 2022).

[25] "A Pinterest Progressive Web App Performance Case Study | by Addy Osmani | Dev Channel | Medium." https://medium.com/dev-channel/a-pinterest-progressive-web-app-performance-case-study-3bd6ed2e6154 (accessed May 21, 2022).

[26] "Twitter Lite PWA Significantly Increases Engagement and Reduces Data Usage." https://developers.google.com/web/showcase/2017/twitter (accessed May 21, 2022).

[27] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," in *Web Information Systems and Technologies*, T. A. Majchrzak, P. and Traverso, K.-H. and Krempels, and V. and Monfort, Eds., Springer International Publishing, 2018, pp. 64–86. doi: 10.1007/978-3-319-93527-0_4.

[28] T. A. Majchrzak, A. Biørn-Hansen, and T.-M. Grønli, "Progressive Web Apps: the Definite Approach to Cross-Platform Development?," in *51st Hawaii International Conference on System Sciences*, T. Bui, Ed., Hilton Waikoloa Village, Hawaii, USA: ScholarSpace / {AIS} Electronic Library (AISeL), 2018, pp. 1–10. doi: 10.24251/HICSS.2018.718.

[29] "IndexedDB API - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API (accessed May 27, 2022).

[30] W. Yang, C. Zhang, M. Pan, C. Xu, Y. Zhou, and Z. Huang, "Do Developers Really Know How to Use Git Commands? A Large-Scale Study Using Stack Overflow," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, Apr. 2022, doi: 10.1145/3494518.

[31] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, "Challenges in Chatbot Development: A Study of Stack Overflow Posts," in *Proceedings of the 17th International Conference on Mining Software Repositories*, in MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 174–185. doi: 10.1145/3379597.3387472.

[32] J. Han, E. Shihab, Z. Wan, S. Deng, and X. Xia, "What do Programmers Discuss about Deep Learning Frameworks," *Empir. Softw. Eng.*, vol. 25, no. 4, pp. 2694–2747, 2020, doi: 10.1007/s10664-020-09819-6.

[33] A. Kamienski and C.-P. Bezemer, "An empirical study of Q&A websites for game developers," *Empir. Softw. Eng.*, vol. 26, no. 6, p. 115, 2021, doi: 10.1007/s10664-021-10014-4.

[34] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts," *J. Comput. Sci. Technol.*, vol. 31, no. 5, pp. 910–924, 2016, doi: 10.1007/s11390-016-1672-0.

[35] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining Questions Asked by Web Developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, in MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, pp. 112–121. doi: 10.1145/2597073.2597083.

[36] "NLTK's list of english stopwords · GitHub." https://gist.github.com/sebleier/554280?permalink_comment_id=2838837#gistcomment-2838837 (accessed Aug. 26, 2022).

[37] C. C. Silva, M. Galster, and F. Gilson, "Topic modeling in software engineering research," *Empir. Softw. Eng.*, vol. 26, no. 6, p. 120, 2021, doi: 10.1007/s10664-021-10026-0.

[38] "MALLET homepage." http://mallet.cs.umass.edu/ (accessed Jan. 14, 2023).

[39] S. Syed and M. Spruit, "Full-Text or Abstract? Examining Topic Coherence Scores Using Latent Dirichlet Allocation," in *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2017, pp. 165–174. doi: 10.1109/DSAA.2017.61.

[40] M. Röder, A. Both, and A. Hinneburg, "Exploring the Space of Topic Coherence Measures," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, in WSDM '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 399–408. doi: 10.1145/2684822.2685324.

[41] "Spearman Rank Correlation Coefficient," in *The Concise Encyclopedia of Statistics*, New York, NY: Springer New York, 2008, pp. 502–505. doi: 10.1007/978-0-387-32833-1_379.

[42] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. P. de Vries, "Moving into a New Software Project Landscape," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, in ICSE '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 275–284. doi: 10.1145/1806799.1806842.

[43] I. Srba and M. Bielikova, "Why is Stack Overflow Failing? Preserving Sustainability in Community Question Answering," *IEEE Softw.*, vol. 33, no. 4, pp. 80–89, Jul. 2016, doi: 10.1109/MS.2016.34.

[44] A. Mazuera-Rozo *et al.*, "Taxonomy of security weaknesses in Java and Kotlin Android apps," *J. Syst. Softw.*, vol. 187, p. 111233, 2022, doi: https://doi.org/10.1016/j.jss.2022.111233.

[45] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos, "Analysis of the reputation system and user contributions on a question answering website: StackOverflow," in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, Aug. 2013, pp. 886–893. doi: 10.1145/2492517.2500242.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۳، شماره ۱، بهار ۱۴۰۳، صفحه ۶۵-۷۸
دریافت: ۱۴۰۲/۰۴/۰۷، بازنگری: ۱۴۰۳/۰۲/۱۲، پذیرش: ۱۴۰۳/۰۶/۰۵

چالش‌های توسعه برنامه‌های کاربردی وب پیشرو: یک مطالعه تجربی با استفاده از استک اورفلو

محمدرضا عباس‌نژاد[۱]، امیر جهانگرد رفسنجانی*[۲]، امین میلانی فرد[۳]
۱- دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
*۲- دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.
۳- انستیتوی فناوری نیویورک پردیس ونکوور، ونکوور، کانادا.
[1]abbasnezhad.m.r@stu.yazd.ac.ir, [2*]jahangard@yazd.ac.ir, and [3]amilanif@nyit.edu

* نشانی نویسنده مسئول: امیر جهانگرد رفسنجانی، یزد، صفاییه، بلوار دانشگاه، دانشگاه یزد، دانشکده مهندسی کامپیوتر.

چکیده- برنامه‌های کاربردی وب پیشرو یک نگرش جدید در توسعه وب هستند که در انواع برنامه‌های کاربردی وب مورد استفاده قرار می گیرند. این نگرش جدید باعث می‌شود که برنامه‌های کاربردی وب، ویژگی‌هایی مشابه با برنامه‌های کاربردی بومی ارائه دهند و مرز بین وب‌سایت‌ها و برنامه‌های کاربردی بومی از بین برود. با این حال، چالش‌های جدیدی ممکن است در توسعه برنامه‌های کاربردی وب پیشرو به وجود آید. به دلیل علاقه روزافزون به برنامه‌های کاربردی وب پیشرو، درک چالش‌هایی که توسعه‌دهندگان ممکن است در هنگام توسعه این برنامه‌های کاربردی با آنها مواجه شوند، ضروری است. این چالش‌ها هنوز در تحقیقات موجود مورد شناسایی و تحلیل قرار نگرفته‌اند. بنابراین، هدف این پژوهش از میان بردن این کاستی در دانش موجود با انجام یک مطالعه تجربی بر روی سوالات مربوط به برنامه‌های کاربردی وب پیشرو در استک اورفلو، یک وب‌سایت مشهور برای توسعه‌دهندگان، است. فرآیند ما شامل به دست آوردن لیستی از برچسب‌های مربوط به برنامه‌های کاربردی وب پیشرو و استخراج ۸۶۵۴ سوال مربوط به آنها از استک اورفلو است. برای درک چالش‌هایی که توسط توسعه‌دهندگان برنامه‌های کاربردی وب پیشرو در استک اورفلو بحث می‌شود، از مدلسازی موضوعات استفاده می‌کنیم. پس از آن، ویژگی های آنها (مانند محبوبیت و دشواری چالش ها) را تجزیه و تحلیل می‌کنیم. بررسی ما نشان می‌دهد که توسعه‌دهندگان با یازده چالش از جمله «چرخه عمر Service Worker» مواجه هستند. این پژوهش درک ما را از چالش‌های توسعه برنامه‌های کاربردی وب پیشرو عمیق‌تر می‌کند و توصیه‌هایی را برای محققان، آموزش‌دهندگان و توسعه‌دهندگان به منظور گسترش استفاده از فناوری برنامه‌های کاربردی وب پیشرو ارائه می‌دهد.

واژه‌های کلیدی: برنامه‌های کاربردی وب پیشرو، مطالعه تجربی، چالش‌های توسعه PWA، مدلسازی موضوعات LDA، استک اورفلو.