Web APIs: Features, Issues, and Expectations

A Large-Scale Empirical Study of Web APIs from Two Publicly Accessible Registries Using Stack Overflow and A User Survey

Neng Zhang, Ying Zou, Xin Xia, Qiao Huang, David Lo, Shanping Li

Abstract—With the increasing adoption of services-oriented computing and cloud computing technologies, web APIs have become the fundamental building blocks for constructing software applications. Web APIs are developed and published on the internet. The functionality of web APIs can be used to facilitate the development of software applications. There are numerous studies on retrieving and recommending candidate web APIs based on user requirements from a large set of web APIs. However, there are very limited studies on the features of web APIs that make them more likely to be used and the issues of using web APIs in practice. Moreover, users' expectations on the development and management of web APIs are rarely investigated. In this paper, we conduct a large-scale empirical study of 20,047 web APIs published at two popular and publicly accessible web API registries: ProgrammableWeb and APIs.guru. We first extract the questions posted in Stack Overflow (SO) that are relevant to the web APIs. We then manually analyze 1,885 randomly sampled SO questions and identify 24 web API issue types (e.g., *authorization error*) that are encountered by users. Afterwards, we conduct a user survey to investigate the features of web APIs that users often consider when shortlisting a web API for testing before they adopt it, validate the identified types of web API issues, and understand users' expectations on the development and management of web APIs. From the 191 received responses, we extract 14 important features for users to decide whether to use a web API (e.g., *well-organized documentation*). We also gain a better understanding of web API issue types and summarize 11 categories of user expectations on web APIs (e.g., *documentation* and *SDK/library*). As the result of our study, we provide guidelines for web API developers and registry managers to improve web APIs and promote the use of web APIs.

Index Terms—Web APIs, Empirical Study, User Survey, Stack Overflow

1 Introduction

Web APIs (also known as web services) are loosely-coupled software modules that encapsulate resources (e.g., data, storage, or computing resource), which are accessible via the internet. By reusing web APIs, developers can reduce the time and effort spent on the development, and increase the quality and flexibility of software applications [1]. With the rapid adoption of services-oriented computing and cloud computing technologies, web APIs have become the fundamental building blocks to construct software applications [2], [3]. Many companies, e.g., Microsoft and Amazon, have developed numerous web APIs and built various service-based systems (i.e., distributed software systems built by composing web services [4]). The web APIs are published using the web API marketplace from a particular company (e.g., Microsoft Azure[1] and Amazon AWS[2]) or public web API registries (e.g., ProgrammableWeb[3] (PW), APIs.guru[4], and RapidAPI[5]), enabling third-party companies or developers to access the resources offered by web APIs [5]. Independent developers can also create web APIs and share them using registries. For example, as of November 12, 2019, more than 22,000 web APIs have been registered at PW.

Given a requirement, it is not an easy task for users to find appropriate web APIs from a large number of web APIs published on the internet, especially when many web APIs offer similar resources, e.g., Google Maps[6] and Bing Maps[7]. To facilitate the selection of web APIs, a great amount of research effort has been devoted to improving the performance in service retrieval [6], [7], [8], [9], [10], [11], [12] and service recommendation [2], [13], [14], [15], [16], [17], [18], [19]. However, there are still a few limitations in the current state of arts.

1) There is no systematic study conducted to understand *whether the published web APIs have been widely used?* and *what features of web APIs make them more likely to be used?*
2) Quite often, users may encounter issues when using web APIs. Intuitively, the issues of using web APIs may differ from those happened when using non-web APIs, e.g., Java SE Development Kit (JDK)[8], as web APIs can be only accessed via internet without installing the

- *Neng Zhang is with the School of Software Engineering, SUN Yat-sen University, China. E-mail: zhangn279@mail.sysu.edu.cn*
- *Ying Zou is with the Department of Electrical and Computer Engineering, Queen's University, Canada. E-mail: ying.zou@queensu.ca*
- *Xin Xia is with the Software Engineering Application Technology Lab, Huawei, Hangzhou, China. E-mail: xin.xia@acm.org*
- *Qiao Huang and Shanping Li are with the the College of Computer Science and Technology, Zhejiang University, China. E-mail: {tkdsheep, shan}@zju.edu.cn*
- *David Lo is with the School of Information Systems, Singapore Management University, Singapore. E-mail: davidlo@smu.edu.sg*
- *Xin Xia is the corresponding author.*

1. https://azuremarketplace.microsoft.com
2. https://aws.amazon.com/marketplace
3. https://www.programmableweb.com
4. https://apis.guru/
5. https://rapidapi.com
6. https://www.programmableweb.com/api/google-maps
7. https://www.programmableweb.com/api/bing-maps
8. https://www.oracle.com/technetwork/java/javase

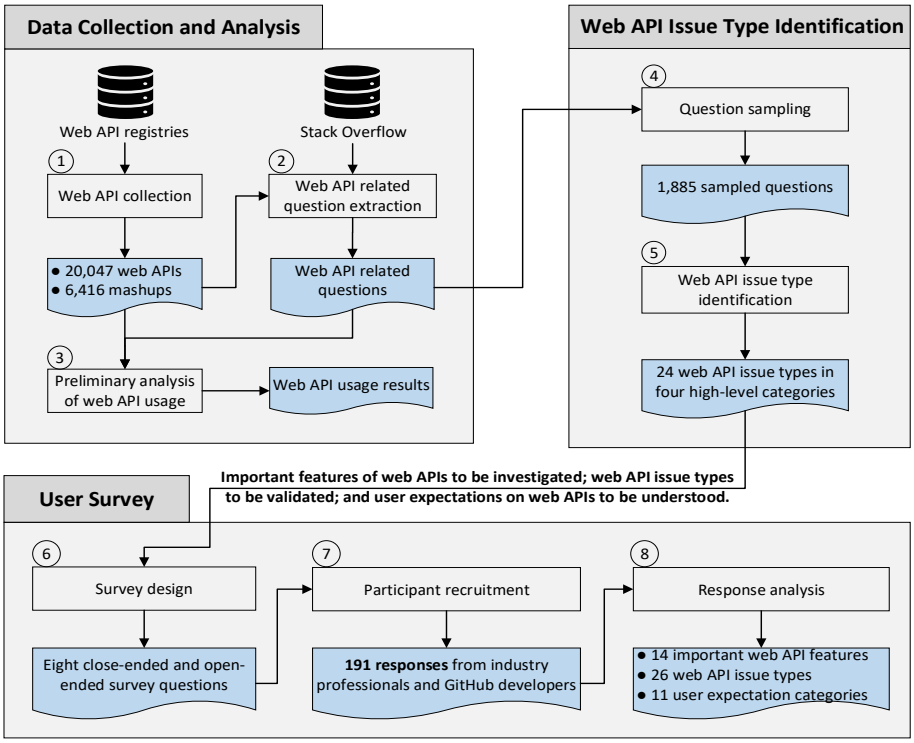

Fig. 1. An overview of our approach.

executable code in a local machine. Although existing work has studied specific kinds of web API issues, e.g., web API changes [20] and documentation reliability [21], *there is a lack of an in-depth investigation of the issues encountered when using web APIs.*

3) *There is almost no study conducted to understand users' expectations (i.e., requirements) on the development and management of web APIs.*

In this paper, we conduct a large-scale empirical study to investigate the usage of web APIs, the issues of using web APIs, and users' expectations on web APIs. We collect 20,047 web APIs from two popular and publicly accessible registries: PW and APIs.guru. Stack Overflow[9](SO), a popular Q&A community for developers, has accumulated millions of questions about various technologies. We propose a heuristic method for extracting SO questions related to the web APIs by leveraging web API names and a set of 26 web API related keywords/phrases that are manually defined (Table 4). We then perform a preliminary study of the web API usage by analyzing the extracted SO questions and the mashups [22] that invoke web APIs from PW. The result reveals that 18.1%-33.5% of the web APIs are used by developers in practice.

We randomly sample 1,885 SO questions related to the web APIs and manually identify 24 types of web API issues (e.g., *authorization error* and *incomplete documentation*) experienced by users. The web API issue types are grouped into four high-level categories: *authorization*, *function*, *documentation*, and *others*.

Finally, we perform a user survey for three purposes: 1) investigating the features of web APIs that are important for users when shortlisting a web API for testing before they adopt it in their applications; 2) validating the identified web API issue types; and 3) understanding users' expectations on web APIs. Our survey is sent to 72 industry professionals who are working at multiple companies and 2,000 Open-Source Software (OSS) developers who have reported issues to web API projects hosted at GitHub[10]. By analyzing 191 received responses, we obtain 14 web API features (e.g., *well-organized documentation* and *standard request/response formats*) and two additional web API issue types that are not discovered from SO questions. Moreover, we summarize 11 categories of user expectations on the development and management of web APIs. As a result, we provide guidelines for web API developers and registry managers to improve the quality of web APIs and promote the use of web APIs.

The major contributions of our study are:

1) We collect a large dataset of 20,047 web APIs from two popular registries: PW and APIs.guru. We propose a heuristic method for extracting SO questions relevant to web APIs.
2) We obtain 14 features of web APIs that are critical for users to shortlist a web API for testing by performing a user survey with industry professionals and GitHub developers.
3) We identify 26 types of web API issues by manually analyzing 1,885 SO questions and the survey responses. The issue types are grouped into four high-level categories.
4) We summarize 11 categories of user expectations on web APIs from the survey responses.
5) We provide guidelines for web API developers and registry managers based on our findings.

9. https://stackoverflow.com

10. https://github.com

Paper Organization. The rest of the paper is structured as follows. Section 2 gives an overview of our approach. Section 3 presents the results of our study. Section 4 discusses the implications for web API developers and registry managers. Section 5 discusses the application of language models in retrieving SO questions for web APIs, the reasons why we present the usage results of web APIs, the uniqueness of our findings to web APIs, and the threats to validity of our study. Section 6 reviews the related work. Section 7 concludes the paper and discusses the future work.

2 Overview of Our Approach

Figure 1 gives an overview of our approach, including three phases: 1) *Data Collection and Analysis* that collects web APIs and mashups from PW and APIs.guru, extracts SO questions related to web APIs, and analyzes the usage of web APIs; 2) *Web API Issue Type Identification* that identifies the types of user reported issues related to web APIs from a set of sampled SO questions; and 3) *User Survey* that investigates the features of web APIs that make web APIs more likely to be used, validates the identified web API issue types, and understands user expectations on web APIs. The materials of this work are released at GitHub: https://github.com/nengz/WebAPIStudy.

2.1 Data Collection and Analysis

In this subsection, we describe the three steps ① - ③ of the data collection and analysis phase shown in Fig. 1.

2.1.1 Web API Collection

We investigate the web APIs published at PW and APIs.guru for the following reasons:

- PW and APIs.guru are two popular and publicly accessible registries that have a large number of web APIs provided by many companies and independent developers.
- Web APIs published at PW and APIs.guru are widely used by researchers in web services community [4], [6], [7], [11], [13], [14], [19], [23], [24], [25], [26], [27], [28], [29].

PW starts from 2005. The web APIs at PW are organized by a number of categories, e.g., *Mapping* and *Financial*. Each web API is assigned to several categories by the API provider and/or registry managers. Despite the assigned categories, PW records other information of a web API, e.g., the API name, a short functional description, and the number of followers. PW also contains a large number of mashups (e.g., more than 7,900 mashups as of November 12, 2019) that are developed by composing the web APIs published at PW. For each mashup, PW records the mashup name, several categories assigned to the mashup, a short functional description, and the related web APIs that are composed by the mashup. We collect 20,175 web APIs and 6,416 mashups from PW.

APIs.guru is built in 2017 and driven by the emerging OpenAPI [30] and GraphQL [31] standards. APIs.guru has two directories of web APIs: the OpenAPI directory and the public GraphQL APIs directory. Both directories are stored at the Github repository of APIs.guru[11]. The specification of each web API in the OpenAPI directory is contained in a YAML file, named as *openapi.yaml* or *swagger.yaml*. The information of web APIs in the public GraphQL APIs directory is either contained in a JSON file, named as *apis.json*, or listed in the *README.md* file of the directory. We download the APIs.guru repository from GitHub and extract the information (e.g., the API name and description) of 1,316 and 69 web APIs from the OpenAPI directory and the public GraphQL APIs directory, respectively.

TABLE 1
Information of web APIs and mashups collected from the two registries: ProgrammableWeb (PW) and APIs.guru.

1. Information collected for web APIs published at PW with an example	
API Name	Amazon DynamoDB
Description	Amazon DynamoDB is a scalable NoSQL database homegrown by the Amazon team. Developers set their write and read capacity to match the number of reads and writes their system needs. ...
Categories	Database, NoSQL
API Portal / Home Page	http://aws.amazon.com/dynamodb/
Docs Home Page URL	http://aws.amazon.com/dynamodb/
Architectural Style	REST
Supported Request Formats	URI Query String/CRUD
Supported Response Formats	JSON
# SDKs	1
# Source Code	0
# Developers	2
# Followers	26
2. Information collected for mashups published at PW with an example	
Mashup Name	Flickr on Dynamo
Description	Downloads entries from the Flickr public photos Atom feed every 2 hours and caches them in Amazon DynamoDB ("zaps atoms into Dynamo"). Why? So you can do zappy fast searches and likes of photos using DynamoDB, ...
Categories	Search, Photos
Related APIs	Flickr, Amazon DynamoDB
3. Information collected for web APIs published at APIs.guru with an example	
API Name	Amazon DynamoDB
Categories	cloud
Description	Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling ...

TABLE 2
The numbers of web APIs from PW that have different architectural styles.

Architectural Style	# Web APIs	Architectural Style	# Web APIs
REST	17,286	Unspecified	289
RPC	1,612	Native/Browser	198
Push/Streaming	354	GraphQL	60
Indirect	328		48

Table 1 lists part of the information collected for web APIs and mashups. We observe several problems with the web APIs: 1) some web APIs from PW have unspecified or special architectural styles (e.g., 'Unspecified' and 'Native/Browser') and may not be real web APIs; 2) some web APIs are created for testing purposes, and such web APIs typically contain the word 'test' in the names or descriptions or have no meaningful words in the descriptions; 3) the names of some web APIs lack spaces or end with web API type keywords/phrases or contain redundant descriptions, e.g., *AzureAnalysisServices*, *Microsoft Azure Search Service*, and *Open Banking - Payments initiation service*; and 4) there are duplicate web APIs[12] with the same names from both registries, e.g., the web API *Amazon DynamoDB* shown in Table 1. The problems can affect other tasks of our study, e.g., extracting SO questions related to the web APIs and analyzing the usage of the web APIs. We process the web APIs as follows.

1) We count the number of web APIs from PW of each archi-

11. https://github.com/APIs-guru

12. We consider two web APIs that have the same name duplicate.

tectural style, as listed in Table 2. REST [32], RPC [33], and GraphQL are popular web API architectural styles, whereas other styles, e.g., 'Native/Browser', are not certainly relevant to web APIs. We filter out 1,217 web APIs that do not use REST, RPC, or GraphQL.

2) We manually examine 906 web APIs that contain 'test' in either the web API names or descriptions. Moreover, we manually check 549 web APIs without meaningful words in the descriptions. We filter out one web API from PW and four web APIs from OpenAPI as such web APIs are likely to be created for testing purposes.
3) We manually examine the names of web APIs and revise a number of irregular names to improve the retrieval of SO questions related to web APIs. We focus on three kinds of web API names: a) the names that end with typical web API type keywords/phrases (e.g., 'api', 'service', and 'web service'), e.g., *Microsoft Azure Search Service*; b) the names that start with well-known companies or platforms (e.g., Amazon and Azure), contain multiple tokens, and have no space in them, e.g., *AzureAnalysisServices*; and c) the names that contain a special string ' - ' (the descriptions after '-' may be redundant), e.g., *Open Banking - Payments initiation service*.
   For a web API name belonging to case a), we remove the web API type keyword/phrase at the end of the name and search results for the modified name using the SO search engine. If there are results returned, we revise the original name to the modified name. For example, *Microsoft Azure Search Service* is revised to *Microsoft Azure Search*.
   For a web API name belonging to case b), we gradually modify the name by adding spaces between the multiple tokens (e.g., *Azure AnalysisServices* and *Azure Analysis Services*) and replacing the specific characters (e.g., '.' and '_') with spaces. We search results for the original name and each modified name using the SO search engine. We revise the original name to the modified name that has the maximum number of search results. For example, *AzureAnalysisServices* is revised to *Azure Analysis Services*. If the revised name matches case a), we further revise it using the steps used for case a). Consequently, we revise *Azure Analysis Services* to *Azure Analysis*.
   For a web API name belonging to case c), we modify the name by retaining the part before ' - ' and search results for the original name and the modified name using the SO search engine. We revise the original name to the modified name if more results are returned for the modified one. For example, *Open Banking - Payments initiation service* is revised to *Open Banking*. If the revised name matches cases a) or b), we further revise it using the steps used for cases a) or b).
   Eventually, we revise 961 web APIs (456 from PW and 505 from APIs.guru). Moreover, we remove two web APIs with meaningless API names from OpenAPI, i.e., *API v1* and *REST API Version 2*.
4) We transform the names of web APIs to lowercase to eliminate the possible inconsistent spellings of web API names. We then filter out the duplicate web APIs. More specifically, we remove 91 duplicates from PW, 32 duplicates from APIs.guru (including 17 duplicates from OpenAPI, 13 duplicates from GraphQL, and two duplicates appearing in OpenAPI and GraphQL), and 166 duplicates that appear in both registries. For the duplicates appearing in both registries, we retain the web APIs from PW and delete those from APIs.guru.

After completing the steps above, we obtain 20,047 web APIs with unique names from both registries. There are 18,866 and 1,347 web APIs with unique names from PW and APIs.guru, respectively.

2.1.2 Web API Related Question Extraction

Users may encounter issues when using web APIs. Since the implementation code of web APIs is generally unavailable, web API forum is used for users to seek help from web API providers. A web API at PW can offer a URL referring to its forum, e.g., the field *API Forum / Message Boards* of web API *Twilio SMS*[13]. However, we find that only 3,225 (i.e., 17.0%) of the 18,957 web APIs from PW (after filtering out the non-web APIs and the web APIs created for testing purposes) have forums. Most of the web APIs do not provide a forum for users to discuss issues. Therefore, web API forums are not suitable to investigate the usage and issues of web APIs.

As a collaborative and active Q&A community, SO has attracted ten million users around the world to discuss technical problems in software development. As of November 12, 2019, SO accumulates 18 million questions and 28 million answers[14]. The tremendous amount of SO data is widely used to conduct empirical studies on various technologies, e.g., mobile development [34], [35], text mining [36], and blockchain [37]. We observe that many SO questions are relevant to web APIs. For example, 60,931 and 24,111 SO questions are tagged with 'google-maps' or 'google-maps-api-3', respectively, indicating that the questions are relevant to the web API *Google Maps*. Therefore, we use the SO data to investigate the usage and issues of web APIs.

We download the official SO data dump released on September 2, 2019. The data dump includes five XML documents: *Posts.xml*, *Comments.xml*, *Badges.xml*, *Users.xml*, and *Votes.xml*. All questions and answers are stored in *Posts.xml*. We extract questions from *Posts.xml*. It is a challenging task to extract questions relevant to our collected web APIs due to the inconsistent mentions for a web API in questions. For example, *Google Maps* are referred to as 'google-maps-api-3', 'google maps', 'Google maps', etc. Recall that we have normalized the names of web APIs by converting them to lowercase. We normalize the mentions of web APIs by processing each question using two steps: 1) converting the title, tags, and body of the question to lowercase; and 2) replacing '-' with a blank character. Based on the transformed results, we can match an API name with its references in SO questions.

We aim to extract the SO questions that mention web APIs. It is an intractable task to examine each of the 18 million questions. To improve the efficiency, we use Lucene [38] (an efficient text search engine that implements BM25) to build the index for all questions, and then use the Lucene search engine to retrieve questions relevant to web APIs. We first retrieve the questions by searching for web API names in the content (including the title, tags, and body) of each question. More than 13.6 million questions are retrieved for

13. https://www.programmableweb.com/api/twilio-rest-api
14. https://data.stackexchange.com/

TABLE 3
Web APIs extracted from four Stack Overflow (SO) questions using two methods. Only the web APIs in italics are correctly extracted from the corresponding questions. *WAKP* is the set of web API related keywords and phrases listed in Table 4.

Question ID	Question Field	Preprocessed Content of the Question Field	Web APIs Extracted based on API Names	Web APIs Extracted based on API Names and *WAKP*
41519641	Title	how do I retrieve a specific range using **google sheets api**	*google sheets*, view, data	*google sheets*
	Tags	android studio, for loop, **google sheets api**		
	Body	i have the following code using the **google sheets api** and got it working perfectly. it sees all my **data** and puts it into the app perfectly via the for loop. it runs through the spreadsheets and puts the **data** in an edit **view**. as you can see, the for loop retrieves the **data** one by one until there is no more **data** inside a specific range ...		
12868753	Title	recommendations for programmatic web searches	*google custom search*, *bing*, search	*google custom search*, *bing*
	Tags	web services, **search**, **bing**, web **search**		
	Body	i was hoping i could use a web service to automatically perform full web **searches** based on keywords ... my first thought was google, and their **google custom search service** looks pretty good ... the only problem is that it has a limit of 100 queries per day. I need more like 1000 ... as i wrote this question i found microsoft's **bing web service** home page. at first blush it looks pretty good. i have a slight preference for google, but am open to microsoft. i would love to hear any advice about using microsoft's api.		
16310966	Title	social framework auto appending access key	*facebook*, form	
	Tags	ios, **facebook**, access token		
	Body	i am attempting to retrieve a list of friends with specific fields using the ios 6 social framework. i am using the url: but the framework seems to be appending the access key to it automagically, and is doing so with a '?' as if it were the first url parameter ... if so, how should i **form** my urlstring so that this is not a problem?		
13188025	Title	insert **string** into **string**?	string	
	Tags	php		
	Body	i have a number of images being uploaded to a site with random file names e.g. is it possible in php to somehow insert immediately before the file extension (.jpg, .jpeg .png or .gif) part of the url another **string** such as 300x200?		

TABLE 4
Web API related keywords and phrases (*WAKP*).

No	Web API Related Keyword or Phrase	No	Web API Related Keyword or Phrase
1	api	14	service
2	webapi	15	webservice
3	web api	16	web service
4	rest api	17	rest service
5	rest webapi	18	rest webservice
6	rest web api	19	rest web service
7	restful api	20	restful service
8	restful webapi	21	restful webservice
9	restful web api	22	restful web service
10	soap api	23	soap service
11	soap webapi	24	soap webservice
12	soap web api	25	soap web service
13	microservice	26	micro service

TABLE 5
The top ten web APIs that have the maximum numbers of SO questions retrieved based on API names and *WAKP*. The web APIs marked with ⋆, ◇, and • are from PW, APIs.guru, and both registries, respectively.

No	API Name	# Retrieved SO Questions
⋆ 1	asp.net web	36,027
⋆ 2	facebook graph	34,639
⋆ 3	google maps	33,767
⋆ 4	youtube	12,225
◇ 5	drive	11,513
⋆ 5	google drive	11,142
⋆ 6	facebook	10,354
◇ 7	data	9,864
⋆ 6	core	7,550
• 8	twitter	6,714

6,323 web APIs. However, many of the questions are irrelevant to web APIs. Table 3 presents nine web APIs extracted from four questions. Only the four web APIs in italics, e.g., *google sheets* and *google custom search*, are indeed discussed in the questions, while the other five web APIs, e.g., *view* and *form*, are incorrectly extracted. We evaluate 384 randomly selected questions, which is a statistically significant sample size considering a confidence level of 95% and a confidence interval of 5%. The result shows that the accuracy is 13.5%. That is, 86.5% of the questions are false positives (i.e., all web APIs extracted from the questions are incorrect), e.g., the question '13188025' shown in Table 3.

Through our observation, some **web API related keywords and phrases** (denoted as *WAKP*) immediately following web API names, e.g., 'api' and 'web api', can be used to improve the accuracy of question retrieval. To build *WAKP*, we collect the questions retrieved for five well-known web APIs: *google maps*, *facebook*, *twitter*, *facebook graph*, and *youtube*. We extract the 1-gram, 2-gram, and 3-gram that follow the five web API names in the questions. An N-gram [39] means a sequence of N consecutive words. A ranked list of the extracted word sequences is generated by sorting them in a descending order by frequency. There are 626,247 word sequences. It is a tremendous workload to manually examine all the word sequences to build a set of *WAKP*. To make the evaluators' workload manageable, we ask the first and the fourth authors of this paper to independently examine the ranked list and build their initial sets of *WAKP* by randomly picking up to 50 interested word sequences. There are 12 different phrases between the two sets of *WAKP*. After discussing the differences, both evaluators build a set of *WAKP* that contain 26 keywords and phrases, as listed in Table 4. REST(ful) [32] and SOAP [40] are two mainstream styles of web APIs [41]. By leveraging *WAKP*, we design a method to retrieve relevant SO questions for a web API:

1) We create a set of search phrases by concatenating the web API name and each element in *WAKP*. For example, the set of search phrases created for *google maps* are { 'google maps api', 'google maps service', ... }. The elements in *WAKP* that contain any word in the web API name are not used to create search phrases in order to avoid unnatural search phrases. For example, the elements in *WAKP* that contain the word 'web', e.g., 'web api' and 'rest webapi', are not used when creating search phrases for the web API name *asp.net web*.
2) We use the Lucene search engine to retrieve the questions that contain any of the search phrases.

Before using the method above, we further process the content of each SO question by replacing the two words 'apis' and 'services' with 'api' and 'service', respectively. We then retrieve 299,701 SO questions that involve 3,018 web APIs. Table 3 shows the web APIs identified from four questions.

TABLE 6
The numbers of web APIs extracted from two data sources (i.e., the mashups from PW and SO questions); and the numbers/percentages or number/percentage ranges of web APIs from PW, APIs.guru, and both registries (denoted as 'both') that have been used by developers.

A Specific Number or Number Range of Web APIs	The Set Expression for Calculating the Number or Number Range	Value of the Set Expression	Percentage or Percentage Range % (Registries)
# Web APIs from PW that are extracted from mashups	$\lvert API_{mas} \rvert$	1,167	6.2% (PW)
# Web APIs from PW that are extracted from SO questions based on API names	$\lvert \hat{API}_{SO}^{PW} \rvert$	5,819	30.8% (PW)
# Web APIs from APIs.guru that are extracted from SO questions based on API names	$\lvert \hat{API}_{SO}^{APIs.guru} \rvert$	616	45.7% (APIs.guru)
# Web APIs from both registries that are extracted from SO questions based on API names	$\lvert \hat{API}_{SO} \rvert$	6,323	31.5% (both)
# Web APIs from PW that are extracted from SO questions based on API names and *WAKP*	$\lvert API_{SO}^{PW} \rvert$	2,695	14.3% (PW)
# Web APIs from APIs.guru that are extracted from SO questions based on API names and *WAKP*	$\lvert API_{SO}^{APIs.guru} \rvert$	406	30.1% (APIs.guru)
# Web APIs from both registries that are extracted from SO questions based on API names and *WAKP*	$\lvert API_{SO} \rvert$	3,018	15.1% (both)
# Web APIs from PW that have been discussed in SO questions	$[\lvert API_{SO}^{PW} \rvert, \lvert \hat{API}_{SO}^{PW} \rvert]$	[2,695, 5,819]	[14.3%, 30.8%] (PW)
# Web APIs from APIs.guru that have been discussed in SO questions	$[\lvert API_{SO}^{APIs.guru} \rvert, \lvert \hat{API}_{SO}^{APIs.guru} \rvert]$	[406, 616]	[30.1%, 45.7%] (APIs.guru)
# Web APIs from both registries that have been discussed in SO questions	$[\lvert API_{SO} \rvert, \lvert \hat{API}_{SO} \rvert]$	[3,018, 6,323]	[15.1%, 31.5%] (both)
# Web APIs from PW that have been used by developers	$[\lvert API_{used}^{PW} \rvert, \lvert \hat{API}_{used}^{PW} \rvert]$	[3,306, 6,215]	[17.5%, 32.9%] (PW)
# Web APIs from APIs.guru that have been used by developers	$[\lvert API_{used}^{APIs.guru} \rvert, \lvert \hat{API}_{used}^{APIs.guru} \rvert]$	[406, 616]	[30.1%, 45.7%] (APIs.guru)
# Web APIs from both registries that have been used by developers	$[\lvert API_{used} \rvert, \lvert \hat{API}_{used} \rvert]$	[3,629, 6,719]	[18.1%, 33.5%] (both)

We observe that although *facebook* in the question '16310966' is missed, no web API is incorrectly extracted from the four questions. We evaluate 384 questions that are randomly sampled from the retrieved questions with a confidence level of 95% and a confidence interval of 5%. The accuracy of retrieval is 92.4%. That is, only 7.6% are false positives, indicating that our refined method can accurately retrieve SO questions related to web APIs. Table 5 lists the top ten web APIs that have the maximum numbers of raised questions. Only 33,767 questions are raised for *google maps*. According to the number of questions tagged with 'google-maps' or 'google-maps-api-3' (i.e., 85,042), at least 60.3% of the questions related to *google maps* are missed by our question retrieval method. This result indicates that the recall of our method is poor. In this work, we ensure the high accuracy of our method for the following reasons. A large number of questions are subsequently sampled from the raised questions to identify web API issue types. A low accuracy would lead to retrieving many questions irrelevant to web APIs and thus waste a significant amount of manual effort.

2.1.3 Preliminary Analysis of Web API Usage

Knowing the usage of web APIs can help web API providers and registry managers optimize the maintenance and management of their developed and managed web APIs. For web API providers, it is easy to know the usage of their web APIs from the monitoring data (e.g., execution logs). In general, the monitoring data of web APIs is unavailable for others, including the registry managers. We attempt to understand the usage of our collected web APIs by leveraging two publicly available data sources: the SO questions related to the web APIs and the mashups from PW. We acknowledge that our results may not reflect all the possible usage of web APIs as web API users may have no record in both data sources. For example, users may encounter few issues of a high-quality web API or report issues in channels other than the two data sources. Even though our results provide a preliminary study of the web API usage, we still think the results are useful for several tasks, e.g., the management of web APIs published at PW and APIs.guru, as described in Section 5.2.

In Section 2.1.2, we retrieve the SO questions related to the web APIs. The entire set of web APIs extracted from the questions are denoted as API_{SO}. The web APIs used by a mashup from PW are recorded in the field *Related APIs* shown in Table 1. We collect the web APIs used by the 6,416 mashups from PW, which are denoted as API_{mas}.

By integrating the web APIs extracted from the two data sources, we obtain a set of web APIs from both registries that have been used as $API_{used} = API_{SO} \cup API_{mas}$. We further distinguish the web APIs from PW and APIs.guru in API_{SO}, which are denoted as API_{SO}^{PW} and $API_{SO}^{APIs.guru}$, respectively. We then obtain a set of web APIs from PW that have been used as $API_{used}^{PW} = API_{SO}^{PW} \cup API_{mas}$, and obtain a set of web APIs from APIs.guru that have been used as $API_{used}^{APIs.guru} = API_{SO}^{APIs.guru}$.

As explained previously, our method for retrieving SO questions related to web APIs has a low recall and may overlook some web APIs, e.g., *facebook* in the question '16310966' shown in Table 3. Some web APIs discussed in SO questions may not be included in API_{SO} if all the questions relevant to a web API are missed. Although the question retrieval method based on web API names has a low accuracy of 13.5%, it may not miss any web API in SO questions, given that there is no problem with the API names and their references in SO questions (except the problems addressed in Sections 2.1.1 and 2.1.2). Therefore, we use the web APIs that are collected from the SO questions raised using API names as a *superset* of the web APIs discussed in SO questions. We denote the superset of a set S as $\hat{S}$. By replacing the sets (except API_{mas}) defined above with the supersets, we obtain six ranges of the web APIs from PW, APIs.guru, and both registries that have been discussed in SO questions or used by developers. For example, the range of web APIs from both registries that have been used is $[API_{used}, \hat{API}_{used}]$.

Table 6 presents the numbers of web APIs extracted from the two data sources, and the number/percentage ranges of web APIs from PW, APIs.guru, and both registries that have been used by developers. $\lvert S \rvert$ is the cardinality of set S. In the 'Percentage' column, '(PW)', '(APIs.guru)', or '(both)' behind a percentage value indicates that the percentage is computed with respect to the number of web APIs from PW, APIs.guru, or both registries, respectively. From the table, we have the following findings:

- Among the 18,866 web APIs from PW, 1,167 (i.e., 6.2%) are extracted from the mashups; 5,819 (i.e., 30.8%) and 2,695 (i.e., 14.3%) are extracted from SO questions using the question retrieval method based on API names and our refined method based on API names and *WAKP*, respectively. Based on the results, 3,306-6,215 (i.e., 17.5%-

32.9%) web APIs from PW have been used by developers.
- Among the 1,347 web APIs from APIs.guru, 616 (i.e., 45.7%) and 406 (i.e., 30.1%) are extracted from SO questions using the two retrieval methods, respectively. Based on the results, 406-616 (i.e., 30.1%-45.7%) web APIs from APIs.guru have been used by developers. The percentages of the web APIs that have been used from APIs.guru are higher than those of the web APIs that have been used from PW, which could be due to three possible reasons: 1) web APIs driven by the emerging OpenAPI and GraphQL standards are gaining greater prevalence in SO; 2) the specifications of web APIs at APIs.guru can be easily obtained from the GitHub repository, while those of web APIs at PW need to be explored from the webpages *API Portal / Home Page* or *Docs Home Page* (Table 1), which hampers the use of web APIs at PW; and 3) APIs.guru is hosted at GitHub, which is a more popular platform than PW, and thus the web APIs at APIs.guru have a higher opportunity to be discovered and used than those at PW.
- Among the 20,047 web APIs from both registries, 3,629-6,719 (i.e., 18.1%-33.5%) have been used by developers based on the analysis of SO questions and the mashups.

According to the results listed in Table 6, 66.5% of the web APIs from PW and APIs.guru have not been used by developers. This result may be frustrating for the web API providers and registry managers. It is necessary to gain some insights on the factors that impact the use of web APIs, so that web API providers and registry managers can take actions to improve web APIs and facilitate the use of web APIs. In the remaining of the paper, we summarize the issues experienced by web API users, investigate the features of web APIs that users often consider when shortlisting a web API for testing before adopting it, and understand users' expectations on the development and management of web APIs.

For the 20,047 web APIs collected from PW and APIs.guru, we analyze the usage of the web APIs by leveraging SO questions and the mashups from PW. We find that 66.5% of the web APIs might not be used by developers. Only 17.5%-32.9% of the 18,866 web APIs from PW are used, while 30.1%-45.7% of the 1,347 web APIs from APIs.guru are used. Our findings can benefit web API users, registry managers, and researchers on several tasks such as obtaining a rough view of the web APIs that have been used and developing better web API retrieval or recommendation approaches, as described in Section 5.2.

2.2 Web API Issue Type Identification

In this subsection, we describe the two steps ④ and ⑤ of the web API issue type identification phase shown in Fig. 1.

2.2.1 Question Sampling

To figure out the types of web API issues encountered by users, we manually analyze a set of SO questions related to web APIs. However, it is not suitable to directly sample 384 questions from the 299,701 questions retrieved using web API names and *WAKP* due to two reasons: 1) the sample size may be insufficient to discover all possible web API issue types; and 2) the sampled questions may focus on the web APIs that have a large number of questions but neglect many other web APIs, as the numbers of questions retrieved for web APIs have a large variation from 1 to 36,027.

To solve the above problem, we divide the numbers of questions retrieved for web APIs into five ranges: [1, 10), [10, 100), [100, 1,000), [1,000, 10,000), and [10,000, ∞)[15]. Table 7 lists the number of web APIs in each range. 'Total' represents the range of the numbers of questions retrieved for all web APIs and can be viewed as a specific range, '[1, ∞)'. For each range, '# Web APIs in the Range' shows the number of web APIs that have the numbers of questions belonging to the range, and '# SO Questions in the Range' shows the total number of questions retrieved for web APIs in the range. We observe that only seven web APIs have more than 10,000 questions (Table 5). 2,154 (i.e., 71.4%) of the 3,018 web APIs have less than ten questions. The sum of the number of questions in the five ranges is 314,244, which is larger than the total number of questions retrieved for all web APIs (i.e., 299,701), since multiple web APIs could be extracted from a question. As listed in Table 3, two web APIs, i.e., *google custom search* and *bing*, are extracted from the question '12868753'. To better understand the variance among the numbers of questions retrieved for web APIs, we calculate five statistics, including *mean*, *standard deviation*, *the first quartile*, *the second quartile* (aka. *median*), and *the third quartile*, of the numbers of questions retrieved for the web APIs in each range. The statistics are listed in Table 7. From the statistics of 'Total', we find that the third quartile is 13.0, indicating that 75% of the numbers of questions retrieved for all web APIs concentrate in a small range, [1, 13]. The mean of 'Total' (i.e., 114.2) is much higher than the third quartile (i.e., 13.0), and the standard deviation of 'Total' (i.e., 1234.5) is much higher than the mean. These results indicate that the last 25% (after the third quartile) of the numbers of questions retrieved for all web APIs have a large variance. From the statistics of the other four ranges (except the range [10,000, ∞) that contains only seven web APIs), the numbers of questions retrieved for the web APIs in each of the four ranges have a similar variance to the variance of 'Total'. That is, 75% of the numbers concentrate in a relatively small range, while the last 25% of the numbers have a large variance.

We sample different sets of SO questions from the questions in each of the five ranges, as listed in Table 7. The sample sizes are statistically significant with a confidence level of 95% and a confidence interval of 5%. In total, we sample 1,885 SO questions that involve 837 web APIs.

2.2.2 Web API Issue Type Identification

We use a card sorting approach [42], [43] to identify web API issue types[16] from the 1,885 sampled SO questions. Two evaluators (i.e., the first and the fourth authors of the paper) are involved in the issue type identification. Since we do not have a set of predefined web API issue types, it is a cumbersome task to identify web API issue types from 1,885 questions because both evaluators need to manually conduct two steps: 1) analyzing the issues reported in each question and annotating each issue with a short description (e.g., a few keywords) that briefly describes the issue; and 2) extracting

15. We determine the five ranges by requiring that the number of questions in each range is large enough for sampling.

16. We refer to a web API issue type as a specific type of issue, e.g., *authentication error*, that happened when using web APIs.

TABLE 7
Five ranges of the numbers of SO questions retrieved for web APIs using web API names and *WAKP* (Table 4). 'Total' represents the range of the numbers of questions retrieved for all web APIs, i.e., '[1, ∞)'. For each range, '# Web APIs in the Range' presents the number of web APIs that have the numbers of questions belonging to the range; and '# SO Questions in the Range' presents the number of questions retrieved for all web APIs in the range. Columns 4-8 present five statistics such as mean and median measured for the numbers of questions retrieved for the web APIs in each range. The last column presents the sample size calculated for the set of questions retrieved for the web APIs in each range, with a confidence level of 95% and a confidence interval of 5%.

Range of # Questions Retrieved for Web APIs	# Web APIs in the Range	# Questions in the Range	Mean	Standard Deviation	The First Quartile	The Second Quartile (Median)	The Third Quartile	Sample Size
[1, 10)	2,154	5,718	2.7	2.1	1.0	2.0	4.0	360
[10, 100)	628	19,729	32.2	23.4	14.0	23.0	43.2	377
[100, 1,000)	189	57,161	316.4	221.0	145.0	242.0	410.0	382
[1,000, 10,000)	40	98,010	2,727.4	2,003.8	1,276.0	1,916.5	3,565.8	383
[10,000,)	7	133,626	21,381.0	11,657.8	11,327.5	12,225.0	34,203.0	383
Total	3,018	299,701	114.2	1,234.5	1.0	3.0	13.0	1,885

TABLE 8
Issue annotations that are manually assigned by two evaluators (i.e., the first and the fourth authors of this paper) to a SO question.

Question	ID	29026067
	Title	paypal api how do i upload tracking numbers
	Tags	paypal
	Body	*i can't find the answer in the paypal api docs.* i have a paypal transactionid. i have a usps tracking number. using paypal api (php), *which api call do i use to tell paypal what the tracking number is for my transaction?*
Extracted Web APIs		paypal
Issue Annotations by the First Author		no answer in docs; ask for an api call
Issue Annotations by the Fourth Author		which api call do i use

web API issue types from their issue annotations. To reduce the workload in manual analysis, both evaluators perform the task in two rounds. In the first round, they build a set of web API issue types from the 360 sampled questions in the range [1, 10) as the questions cover more web APIs than the questions sampled from any of the other four ranges. More specifically, there are 320, 257, 177, 93, and 43 web APIs involved in the questions that are sampled from the ranges [1, 10), [10, 100), [100, 1,000), [1,000, 10,000), and [10,000, ∞), respectively. These numbers are different from the numbers of web APIs listed in Table 7, which can be explained by the fact that a question may involve multiple web APIs, e.g., the question '12868753' listed in Table 3. In the second round, they identify web API issue types from the remaining 1,525 questions based on the issue types created in the first round.

The first round consists of the following three steps:

- **Initial issue annotation.** For each question, the two evaluators independently identify issues from the question and annotate each issue with a short description. Multiple issue annotations are separated by a semicolon ';'. Table 8 shows the issue annotations assigned by both evaluators to the question '29026067'. One evaluator identifies two issues while the other evaluator identifies only one issue from the question. The annotations of an issue from both evaluators have different expressions, e.g., *ask for an api call* and *which api call do i use*.
- **Issue type identification.** We gather all issue annotations of the 360 questions. One evaluator process the issue annotations by 1) grouping the issue annotations that have the same meaning, e.g., *ask for an api call* and *which api call do i use*, and 2) assigning an issue type to each group, e.g., *enquiring about a particular function* is assigned to the group that contains the two issue annotations mentioned above. The other evaluator verifies the grouped issue annotations and the assigned issue types, and provide suggestions for improvement. After incorporating the suggestions, both evaluators reach a consensus on 23 web API issue types.
- **Issue annotation replacement.** We automatically replace the issue annotations of each question with the corresponding issue types. For example, the issue annotations given by the first author listed in Table 8 are separated by ';' and put into two issue annotation groups with the issue types *incomplete documentation* and *enquiring about a particular function*, respectively. As a result, we replace the issue annotations with the two issue types. 72 questions are in disagreement between the evaluators; and the Fleiss Kappa [44] value is 0.78, indicating substantial agreement. Both evaluators discuss the disagreements to reach a common decision.

In the second round, both evaluators independently identify web API issue types from the remaining 1,525 questions that are sampled from the other four ranges (except [1, 10)). They annotate issues with the existing issue types. If there are new issues (that cannot be covered by the 23 issue types identified in the first round), they annotate each new issue with a short description. After the annotation process, we gather all new issue annotations from the questions. We find that only 12 questions have new issue annotations. Both evaluators add one new web API issue type, i.e., *authorization expiration*, to the set of issue types identified in the first round. We then replace the new issue annotations with the new issue type. After the replacement process, there are 196 questions with disagreement between the evaluators. The Kappa value is 0.86, indicating almost perfect agreement and is higher than that of the first round, since the evaluators already have some experience to identify issue types from questions. The evaluators reach a consensus by discussing the disagreements.

It takes approximately 125 hours for the evaluators to complete the two rounds and identify 24 web API issue types from the 1,885 sampled questions.

2.3 User Survey

In this subsection, we describe the three steps ⑥ - ⑧ of the user survey phase shown in Fig. 1.

2.3.1 Survey Design

The goals of our user survey include: 1) understanding the features of web APIs that make them more likely to be used;

2) validating the web API issue types identified from SO questions; and 3) inquiring about users' expectations on web APIs. To achieve the goals, we design eight close-ended and open-ended survey questions, and limited the answer types to Likert-scale and short free-form text based on the guidelines specified by Kitchenham et al. [45]. In particular, we capture the following information from respondents.

Demographics:

- *Number of used web APIs*: 0 / 1-5 / 6-10 / 11-20 / 20+
- *Years of experience using web APIs*: 0-1 years / 1-3 years / 3-7 years / 7-12 years / 12+ years
- *Current country of residence*: _____

We collect the demographic information of respondents[17] to 1) analyze which countries the respondents are from; 2) filter the respondents who have not used any web API (if a respondent chooses '0' as the number of used web APIs, the survey will be terminated); and 3) group the respondents' results by the number of used web APIs and the years of experience using web APIs.

Important features of web APIs:

Given a requirement, users may obtain a set of candidate web APIs using general web search engines (e.g., Google) and/or the search engines of web API registries or marketplaces. Due to the limited time and budget, it is often impossible for users to find the desired web APIs by testing every candidate. Users often need to choose some promising candidates for testing based on the displayed information of web APIs. We create a survey question to ask respondents about *how often they need to choose some promising web APIs for testing their functionality, in order to find appropriate ones from a number of candidates?*: Very Often / Often / Neutral / Rare / Very Rare.

We create another survey question to ask respondents *the top three important features of web APIs when they decide to shortlist a web API for testing*. To design this question, we recruit 20 developers who have experience using web APIs from two large sister IT companies in China: Insigma Global Service (IGS) and Hengtian[18], via email. In the emails, we introduce the purpose of our study to the developers, i.e., inquiring three characteristics of web APIs that they care the most when shortlisting web APIs for testing after they browse the public information of no less than 50 web APIs published at PW, APIs.guru, and RapidAPI[19]. We also ask the developers about their years of experience using web APIs. They have 1-7 years of experience using web APIs, as listed in Table 22. After the developers complete the task of browsing the public information of web APIs, we ask them for the three most important characteristics of web APIs. The two evaluators independently analyze the developers' comments and extract web API features[20] using a card sorting approach. There are two comments with disagreement. The Fleiss Kappa value is 0.89, meaning almost perfect agreement. Both evaluators reach a consensus on 11 features after discussing the disagreements. We list the 11 features as answer options of the survey question. We also allow respondents to specify other features. The entire answer options are as follows.

- ☐ Similar functional description to the requirement[21]
- ☐ Well-organized documentation
- ☐ Well-known API provider
- ☐ Relatively high popularity (e.g., the number of followers and popularity score)
- ☐ Standard request/response formats (e.g., JSON)
- ☐ Support (e.g., API client wrappers or sample codes) for familiar programming languages
- ☐ Easy-to-test endpoints
- ☐ Free trial
- ☐ No charge for use
- ☐ Accessible API forum
- ☐ Compatible license[22] with the existing projects
- ☐ Other (please specify): _____

Validation of web API issue types:

We create two survey questions to validate the 24 web API issue types identified from SO questions.

- *For each of our identified web API issue types explained below, please confirm whether you have encountered an issue of the type when using web APIs.*
  - (I1) ***Authorization*** *- error: An authorization error occurs when creating an API key (or access token) of a web API or requesting a function with the created API key (or access token)*: Yes / No / I don't know
  - ...
- *Please list other issues you have encountered*: _____

The first survey question asks respondents to confirm whether they have encountered each of the 24 issue types. For explanations of the issue types described in the survey question (e.g., the *authorization error* presented above), please refer to Section 3.2. We provide an 'I don't know' option to reduce the impact of respondents' poor understanding of any issue type. The second survey question is to obtain other web API issues that are not identified from SO questions.

Expectations:

17. For the two survey questions that ask for the years of experience using web APIs and the number of used web APIs, we determine the ranges based on the surveys conducted in prior studies [46], [47].

18. IGS is an outsourcing company which has more than 500 employees and mainly does outsourcing projects for Chinese vendors. Hengtian is also an outsourcing company which has more than 2,000 employees and mainly does outsourcing projects for American and European corporations.

19. RapidAPI is another popular web API registry. Unlike PW and APIs.guru that allow researchers to use the web APIs published at them, RapidAPI has relatively strict terms of services on the use of the web APIs published at it. Unfortunately, we do not get the RapidAPI managers' permission on using the web APIs published at RapidAPI. However, the public information of web APIs published at RapidAPI can be freely browsed by people. We ask the recruited developers to browse the web APIs of RapidAPI, so that they could know more about the available information of web APIs.

20. We refer to a web API feature as a characteristic of web APIs, e.g., *well-organized documentation*

21. We list this feature as an answer option for the following reason. Given a query, the web APIs returned by the search engines in web API registries (e.g., PW) may not all satisfy the user's requirement due to two main factors: the existing search engines mainly rely on keyword matching that often suffers from poor performance; and the query may not precisely represent the user's requirement. Therefore, users often need to examine the functional descriptions of the returned web APIs to determine if the web APIs could satisfy their requirements.

22. Similar to the other software applications, web APIs may have licenses for use, e.g., 'MIT License' and 'Apache License 2.0'.

We create an open-ended survey question to collect respondents' expectations on the development and management of web APIs: *Considering the web API issues that you have encountered, what expectations do you have on the development and management of web APIs?* _____

2.3.2 Respondent Recruitment

To conduct our survey, we need to collect a sufficient number of respondents with diverse backgrounds. Similar to the previous work [47], [48], [49], we follow a multi-pronged strategy to recruit respondents from both OSS community and industry.

- **For industry professionals.** We send the survey to the industry professionals in our contacts and receive responses from 21 contacts working in different companies from China and the United States. We encourage the respondents to help us distribute the survey to some of their associates and colleagues in order to recruit more industry professionals from other countries. As a result, we are able to reach 72 industry professionals working in Microsoft, Google, Huawei, Alibaba, Baidu, IGS, Hengtian, and other companies from five countries.
- **For OSS developers.** We target developers at GitHub who have used web APIs. We select ten well-known web APIs, e.g., *google maps* and *facebook graph*, from the top 20 web APIs that have the maximum numbers of SO questions retrieved using web API names and *WAKP*. We then search 50 relatively popular and active GitHub projects (that have more than 1,000 stars and have been updated within one month) related to the web APIs, e.g., *googlemaps/google-maps-services-js*[23] and *mobolic/facebook-sdk*[24]. We obtain 2,000 developers who have reported issues to the projects, and send an email to each of them with links to two versions of the survey (one link is for the English version and the other link is for the Chinese version, as described below).

To increase the response rate, we use an anonymous survey [50]. No identity information is required or gathered from respondents. Since our survey is oriented to web API users, we notify in the email that *If you have never used any web API, please ignore the survey.*

Before sending the survey to potential respondents, we conduct a pilot study with five developers to verify the survey questions from three aspects: 1) *length*: whether there are any survey questions that are too lengthy to read or understand; 2) *clarity*: whether there are any survey questions with unclear expressions; and 3) *bias*: whether there are any survey questions that may imply the answers that we wish to obtain from the respondents. We refine the survey based on the developers' feedback and ask the developers to verify the refined survey again, which results in the final version of our survey in English. The details of the pilot study are described in Appendix A. As Chinese is one of the most spoken languages in the world, we translate the survey from English to Chinese to ensure that respondents from China could understand our survey well. The first author of the paper translates the survey from English to Chinese. The fourth author of the paper then checks the semantic consistency between the English and Chinese versions of the survey and finds three pieces of inconsistent content related to the descriptions of three web API issue types, e.g., the ***function*** - *inconsistent result* (IT10) listed in Table 13. Both authors solve the inconsistencies together to produce the final version of our survey in Chinese. We do not verify the Chinese version of the survey using a pilot study since the content is consistent with the English version that has been verified. Unlike the other tasks performed by the two authors, e.g., the identification of web API issue types from SO questions, we do not measure the inter-rater agreement score (e.g., the Fleiss Kappa value) between both authors for the survey translation task. The reason is that the translation process does not meet the conditions that are used to measure an inter-rater agreement score. More specifically, we do not have an initial survey in Chinese for both authors to independently examine the semantic consistency between the Chinese and English versions of the survey and then discuss any disagreements to reach a consensus.

2.3.3 Response Analysis

We receive 191 survey responses from 35 countries, of which the top three countries are China (with 48.7% responses), the United States (with 11.0% responses), and Germany (with 4.7% responses). Each survey response contains answers to at least one of the five survey questions that do not ask for demographic information. Among the responses, 144 and 47 responses are from the 2,000 GitHub developers and the 72 industry professionals, respectively. We find that 83 emails sent to the Github developers are rejected. Therefore, the overall response rate of our user survey is 191/(2,000-83+72) = 9.6%, which is similar to the online surveys conducted in previous work [46], [51]. We analyze the responses as follows.

- For the survey question used to investigate the important features of web APIs, we collect 11 pieces of free-form feature descriptions specified in the 'Other' field. We then use a card sorting approach [42] to identify features from the descriptions. Two evaluators (i.e., the first and the fourth authors of the paper) first independently identify features from each description. There are two descriptions with disagreement; and the Fleiss Kappa value is 0.75, indicating substantial agreement. By discussing the disagreements, both evaluators reach a consensus on three features in addition to the 11 features listed as answer options of the survey question. For each of the 14 features, we count the number of respondents who consider the feature important when shortlisting a web API for testing. Note that although we only ask for the top three important features, a number of respondents specify more than three features.
- For the two survey questions used to validate the 24 web API issue types identified from SO questions, we collect 28 pieces of free-form issue descriptions from the second survey question. Similar to the second round of issue type identification described in Section 2.2.2, both evaluators independently identify web API issue types from the descriptions. There are disagreements on four descriptions; and the Fleiss Kappa value is 0.83, which means almost perfect agreement. Both evaluators discuss the disagreements and obtain two additional web API issue types. For each of the 26 issue types, we count the number of respondents who have encountered an issue of the type (i.e., the answer to the issue type in the first

23. https://github.com/googlemaps/google-maps-services-js
24. https://github.com/mobolic/facebook-sdk

TABLE 9
Six respondent groups. The three groups APINumS, APINumM, and APINumL are divided by the number of web APIs that have been used by the respondents; and the three groups ExpL, ExpM, and ExpH are divided by the respondents' years of experience using web APIs.

Group Set	Group	# Respondents	Percentage of Respondents
APINumGSet	APINumS	32	17%
	APINumM	68	36%
	APINumL	91	48%
ExpGSet	ExpL	49	26%
	ExpM	106	56%
	ExpH	36	19%

survey question is 'Yes' or such an issue is specified in the second survey question).

- For the survey question used for respondents to express expectations on web APIs, we collect the free-form descriptions given by 114 respondents. Both evaluators use a card sorting approach again to extract expectations from the descriptions. They first independently extract expectations from each description. We observe 18 descriptions with disagreement. The Fleiss Kappa value is 0.83, indicating almost perfect agreement. After resolving the disagreements, both evaluators summarize 11 categories of expectations and a set of expectations specific to each category. We count the number of respondents who express expectations to each category.

We divide the respondents into six groups based on their demographic information to examine whether there are differences among the respondents who have used different numbers of web APIs or have different years of experience using web APIs. The six groups are as follows.

- Respondents who have used a small number (i.e., 1-5) of web APIs (APINumS).
- Respondents who have used a medium number (i.e., 6-20) of web APIs (APINumM).
- Respondents who have used a large number (≥ 20) of web APIs (APINumL).
- Respondents who have low experience (≤ 3 years) using web APIs (ExpL).
- Respondents who have medium experience (i.e., 3-12 years) using web APIs (ExpM).
- Respondents who have high experience (≥ 12 years) using web APIs (ExpH).

Table 9 lists the numbers and percentages of respondents in the six groups. The six groups are further grouped as two group sets: 1) $APINumGSet = \{$APINumS, APINumM, APINumL$\}$ that are divided by the number of used web APIs; and 2) $ExpGSet = \{$ExpL, ExpM, ExpH$\}$ that are divided by the years of experience using web APIs.

We compute the percentages of respondents in each group who express the 14 web API features, 26 web API issue types, and 11 categories of expectations on web APIs. Moreover, we perform multiple comparisons between any two groups in *APINumGSet* or *ExpGSet* using the multiple test approach proposed by Konietschke et al. [52], which is implemented as the function *nparcomp*() in the R package *nparcomp*. For a comparison (e.g., A vs. B), the approach produces a p-value (p) which indicates whether the differences between the two groups are statistically significant (if $p < 0.05$) and a relative contrast effect (e) which indicates whether the observations in group A tend to be larger (if $e > 0.5$) or smaller (if $e < 0.5$) than those in group B. The p-values and relative contrast effects are computed with simultaneous confidence intervals for all comparisons in *APINumGSet* or *ExpGSet*.

As described above, our survey respondents are from different countries. Moreover, there are two kinds of respondents, i.e., OSS developers and industry professionals. According to the study conducted by Zampetti et al. [53], it is not always the case that the answers provided by OSS respondents and industry respondents are comparable. Similarly, the answers provided by respondents from different countries may not be comparable. To verify these problems, we conduct two experiments. First, we perform multiple comparisons between the respondents from China, the United States, and the other 33 countries, with respect to each of the 14 features, 26 issue types, and 11 categories of expectations using the function *nparcomp*() in *nparcomp*. Since there are only a few (i.e., 1-9) respondents from each of the 33 countries except China and the United States, we view the 33 countries as one group. The results show only two of the 153 comparisons have significant differences. Second, we perform comparisons between the OSS and industry respondents with respect to each of the features, issue types, and expectation categories using the function *npar.t.test*() in *nparcomp*. Only five of the 51 comparisons have significant differences. Based on the results from both experiments, the answers provided by the respondents from different countries and by the OSS and industry respondents are comparable in most cases. Therefore, we do not distinguish the respondents in terms of the two properties in this work.

3 Results

In this section, we present the results of our study to answer the following three research questions:

- **RQ1. What features of web APIs do users often consider prior to shortlisting web APIs for testing?**
- **RQ2. What types of web API issues have been encountered by users?**
- **RQ3. What expectations do users have for web API providers to facilitate the use of web APIs?**

3.1 RQ1. What features of web APIs do users often consider prior to shortlisting web APIs for testing?

Motivation. Users may obtain a number of candidate web APIs for a requirement after searching on the internet. Due to the limited time and budget, they need to shortlist a subset of web APIs for testing in order to find appropriate ones to implement the requirement. We want to find out whether users often need to shortlist web APIs for testing and the features of web APIs that users often consider prior to shortlisting a web API.

Approach. We create two questions in our survey. One survey question asks respondents how often they need to shortlist web APIs for testing. The other survey question asks respondents to express the top three important features of web APIs when they decide to shortlist a web API for testing.

Results. As shown in Fig. 2, 126 (= 55+71) of the 191 (i.e., 66.0%) respondents often need to shortlist web APIs for

TABLE 10
Fourteen features of web APIs expressed by the respondents as important for shortlisting a web API for further testing.

Feature ID	Feature	# Respondents	Percentage of Respondents
F1	Similar functional description to the requirement	118	62%
F2	Well-organized documentation	162	85%
F3	Well-known API provider	64	34%
F4	Relatively high popularity	59	31%
F5	Standard request/response formats	113	59%
F6	Support for familiar programming languages	61	32%
F7	Easy-to-test endpoints	72	38%
F8	Free trial	69	36%
F9	No charge for use	68	36%
F10	Accessible API forum	12	6%
F11	Compatible license	33	17%
F12	Acceptable response time	5	3%
F13	Follows REST/GraphQL specification	3	2%
F14	Uses versioning	2	1%

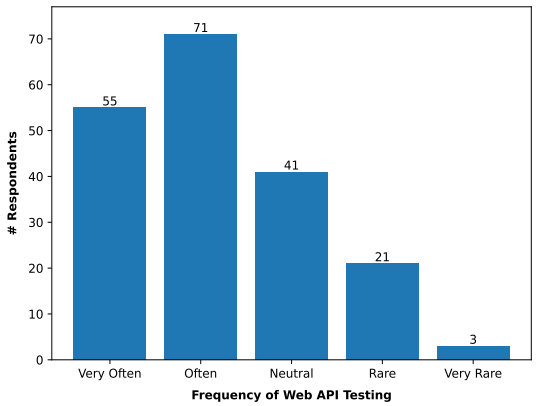

Fig. 2. The numbers of respondents based on the frequencies of shortlisting web APIs for testing.

testing. Table 10 lists 14 important features of web APIs expressed by the respondents. The 'Percentage of Respondents' column presents the percentages of respondents who express the features. From the table, we have the following findings:

- *Similar functional description to the requirement* (F1), *well-organized documentation* (F2), and *standard request/response formats* (F5) are selected by 59%-85% respondents. If a web API has a functional description similar to the requirement, it could be probably used to implement the requirement. A well-organized documentation can help users easily find the functions supported by a web API and how to invoke the functions. For a web API with standard request/response formats, the functions can be easily requested and the results can be easily processed using a standard method, e.g., a JSON library in Python.
- *Easy-to-test endpoints* (F7), *free trial* (F8), and *no charge for use* (F9) are selected by 36%-38% respondents. Before shortlisting a web API, users are interested in checking whether the web API endpoints can be easily tested to see if the results are expected and whether there is no expense on the trial and use of the web API.
- 31%-34% respondents care about whether a web API has a *well-known provider* (F3), *relatively high popularity* (F4), and *support for familiar programming languages* (F6). F3 and F4 can reflect the quality of a web API to some extent; and F6 indicates whether a web API can be used with the user's familiar programming languages.
- The five features F10-F14 are selected by 1%-17% respondents, meaning that only a few users care about whether a web API has the five features, e.g., an accessible web API forum and a REST or GraphQL specification.

Table 11 presents the percentages of respondents in the six groups (Table 9) who express the 14 features. The percentages are different among the three groups in *APINumGSet* or *ExpGSet*. For example, the percentages of *well-known API provider* (F3) and *relatively high popularity* (F4) in APINumS are 44% and 53%, respectively, which are higher than those in APINumM and APINumL. The percentages of F3 and F4 in ExpL (i.e., 39% and 37%, respectively) are also higher than those in ExpM and ExpH. The results can be explained by the following possible reason. The novice users who have used a small number of web APIs or have low experience using web APIs may lack knowledge on how to efficiently test the quality of web APIs through specialized methods, e.g., endpoint testing. Therefore, novice users are more likely to follow a common sense that web APIs from a well-known provider and/or with high popularity generally have high quality. In terms of *accessible API forum* (F10), the percentage in APINumS (resp. ExpL) is higher than those in APINumM and APINumL (resp. ExpM and ExpH). This result means that novice users care more about the accessibility of web API forums, i.e., whether they could contact and get support from web API providers for encountered issues. We also find that *well-organized documentation* (F2) has the highest percentages in all the six groups. This is because documentation is the main resource describing the functionality of a web API. A well-organized documentation can help users easily find a function and how to use the function.

Table 12 presents the results of multiple comparisons between any two groups in *APINumGSet* or *ExpGSet* on each of the 14 features. Each cell shows the p-value (p) and relative contrast effect (e) for a comparison on a feature. '–' means $p \geq 0.05$. We find a number of significant differences between the six group pairs on six features: F2, F4, F8-F10, and F12. For example, in terms of *relatively high popularity* (F4), the differences between the group pair 'APINumS vs. APINumM' are significant with $p < 0.01$ and $e = 0.69$. This result implies that the respondents in APINumS tend to care more about the popularity of a web API than the respondents in APINumM, which could be explained by the reason mentioned above.

TABLE 11
Percentages of the respondents in the six groups (Table 9) who express each of the 14 web API features (Table 10).

Feature ID	APINumS	APINumM	APINumL	ExpL	ExpM	ExpH
F1	62%	60%	63%	71%	60%	53%
F2	94%	97%	73%	88%	85%	81%
F3	44%	28%	34%	39%	34%	25%
F4	53%	16%	34%	37%	31%	22%
F5	69%	51%	62%	65%	54%	67%
F6	31%	25%	37%	35%	29%	36%
F7	22%	38%	43%	39%	35%	44%
F8	25%	54%	26%	55%	28%	33%
F9	44%	32%	35%	57%	30%	22%
F10	16%	4%	4%	12%	6%	–
F11	28%	10%	19%	31%	13%	11%
F12	–	6%	1%	–	5%	–
F13	–	–	3%	–	3%	–
F14	–	–	2%	–	–	6%

TABLE 12
The results of multiple comparisons between any two respondent groups in APINumGSet and ExpGSet (Table 9), with respect to each of the 14 web API features (Table 10). Each cell (p, e) presents the p-value (p) and the relative contrast effect (e) of a pairwise comparison on a feature. '–' means there is no significance (i.e., $p \geq 0.05$).

Feature ID	APINumS vs. APINumM	APINumS vs. APINumL	APINumM vs. APINumL	ExpL vs. ExpM	ExpL vs. ExpH	ExpM vs. ExpH
F1	(–, 0.51)	(–, 0.50)	(–, 0.49)	(–, 0.45)	(–, 0.59)	(–, 0.54)
F2	(–, 0.48)	(<0.01, 0.61)	(<0.001, 0.62)	(–, 0.49)	(–, 0.54)	(–, 0.52)
F3	(–, 0.58)	(–, 0.55)	(–, 0.47)	(–, 0.48)	(–, 0.57)	(–, 0.55)
F4	(<0.01, 0.69)	(–, 0.59)	(<0.05, 0.41)	(–, 0.47)	(–, 0.57)	(–, 0.55)
F5	(–, 0.59)	(–, 0.54)	(–, 0.45)	(–, 0.44)	(–, 0.49)	(–, 0.44)
F6	(–, 0.53)	(–, 0.47)	(–, 0.44)	(–, 0.47)	(–, 0.49)	(–, 0.47)
F7	(–, 0.42)	(–, 0.40)	(–, 0.48)	(–, 0.48)	(–, 0.47)	(–, 0.45)
F8	(<0.05, 0.35)	(–, 0.49)	(<0.01, 0.64)	(<0.01, 0.37)	(–, 0.61)	(–, 0.47)
F9	(–, 0.56)	(–, 0.54)	(–, 0.49)	(<0.01, 0.36)	(<0.01, 0.68)	(–, 0.54)
F10	(–, 0.56)	(–, 0.56)	(–, 0.50)	(–, 0.47)	(<0.05, 0.56)	(<0.05, 0.53)
F11	(–, 0.59)	(–, 0.55)	(–, 0.46)	(–, 0.41)	(–, 0.60)	(–, 0.51)
F12	(–, 0.47)	(–, 0.49)	(–, 0.52)	(<0.05, 0.52)	(–, 0.50)	(<0.05, 0.52)
F13	(–, 0.50)	(–, 0.48)	(–, 0.48)	(–, 0.51)	(–, 0.50)	(–, 0.51)
F14	(–, 0.50)	(–, 0.49)	(–, 0.49)	(–, 0.50)	(–, 0.47)	(–, 0.47)

Among the 191 survey respondents, 66.0% respondents often need to shortlist candidate web APIs for testing in order to find appropriate web APIs to fulfill a requirement. We summarize 14 features of web APIs (i.e., F1-F14 listed in Table 10) that are important for the respondents to shortlist a web API for testing. Some features (e.g., standard request/response formats (F5) and support for familiar programming languages (F6)) provide guidance for web API developers to design better web APIs, such as offering standard request/response formats and implementing web API client wrappers for multiple programming languages. Based on the features, web API developers and managers can also present the information of web APIs important to users to help users make informed decisions on whether to test a web API. A guideline is described in Section 4.2.

3.2 RQ2. What types of web API issues have been encountered by users?

Motivation. Having a good understanding of the web API issues experienced by users can help web API providers develop better web APIs by taking actions to avoid some issues during the development process. Existing work studies only a few kinds of web API issues, e.g., documentation reliability [21]. We attempt to build a comprehensive view of web API issues reported by users.

Approach. We manually identify 24 web API issue types from 1,885 SO questions. We also conduct a user survey to validate the issue types by asking respondents to confirm whether they have encountered issues of each type and express other issues that cannot be covered by the issue types.

Results. Table 13 presents the final set of 26 web API issue types identified from SO questions and the survey responses. We group the issue types into four high-level categories: ***authorization*** (IC1), ***function*** (IC2), ***documentation*** (IC3), and ***others*** (IC4). The two issue types ***authorization*** - *implementation problem* (IT4) and ***others*** - *difficult to sign up* (IT26) are identified from other issue descriptions given by four respondents. For each issue type, we present the number and percentage of SO questions that discuss an issue of the type, and the number and percentage of respondents who have encountered an issue of the type. The value 0% means that the percentage is less than 0.5%.

3.2.1 Explanations of 26 Web API Issue Types

IC1. Authorization. Web APIs often use an authorization mechanism (e.g., HTTP Basic Auth [54] and OAuth [55]) to grant a user access to the functionality. In general, users need to create an API key or access token using the authorization mechanism of a web API, and then use the API key or access token to request the functions. We identify four issue types related to the authorization of web APIs.

IT1 **Error.** An authorization error occurs when creating an API key/access token of a web API or requesting a function with the created API key/access token. An issue

TABLE 13
Twenty-six web API issue types belonging to four categories that are identified from 1,885 SO questions and the survey responses.

Issue Category ID	Issue Category	Issue Type ID	Issue Type	# Questions	Percentage of Questions	# Respondents	Percentage of Respondents
IC1	Authorization	IT1	Error	97	5%	131	69%
		IT2	Expiration	12	1%	129	68%
		IT3	Enquiring about acquisition	73	4%	91	48%
		IT4	Implementation problem	–	–	3	2%
IC2	Function	IT5	Request failure with unknown reasons	315	17%	136	71%
		IT6	Enquiring about a particular function	445	24%	98	51%
		IT7	Change/Deprecation	62	3%	134	70%
		IT8	Limitation	52	3%	158	83%
		IT9	Lack of programming language support	36	2%	110	58%
		IT10	Inconsistent result	95	5%	64	34%
		IT11	Instability	27	1%	83	43%
		IT12	Unexpected result	69	4%	106	55%
		IT13	Unsupported function	31	2%	113	59%
		IT14	Unsatisfactory performance	16	1%	99	52%
		IT15	Implementation problem	4	0%	94	49%
		IT16	Difficult to process result	80	4%	75	39%
IC3	Documentation	IT17	Missing	4	0%	140	73%
		IT18	Incorrect	67	4%	134	70%
		IT19	Incomplete	92	5%	149	78%
		IT20	Out-of-date	6	0%	141	74%
		IT21	Difficult to read/understand	32	2%	143	75%
IC4	Others	IT22	Checking for web API usage	164	9%	74	39%
		IT23	Enquiring about web API implementation	108	6%	63	33%
		IT24	Web API searching	25	1%	92	48%
		IT25	Unsuitable charge	4	0%	56	29%
		IT26	Difficult to sign up	–	–	1	1%

description in the SO question '34215664' is: "*i am getting the below error when i am trying to use the authentication process with twitter api*".

IT2 **Expiration.** The authorization (i.e., an API key/access token) expires after working for a period. An issue description in the SO question '49552348' is: "*i am using amazon cognito identity services in mobile app. the id token will expire after 30 minutes of login*".

IT3 **Enquiring about acquisition.** A user wants to know how to acquire the authorization, e.g., creating an API key/access token. An issue description in the SO question '38684173' is: "*the problem is that we don't have the accesstoken as described in the link. were do i get this accesstoken from?*".

IT4 **Implementation problem.** The authorization mechanism of a web API has implementation problems. This issue type is identified from the other issue descriptions given by three respondents, e.g., "*incorrect authorization implementation*".

IC2. Function. We identified 12 issue types related to the functions of web APIs.

IT5 **Request failure with unknown reasons.** A functional request fails due to unknown reasons, excluding the failures caused by the other identified issue types, e.g., ***authorization*** - *error* (IT1) and ***function*** - *change/deprecation* (IT7). An issue description in the SO question '42730353' is: "*i'm trying to make a dm auto reply, something like a q&a with the twitter api, but maybe i'm screwing something and i can't make it works*".

IT6 **Enquiring about a particular function.** A user wants to know whether a particular function is supported or how to implement a particular function. An issue description in the SO question '32711534' is: "*i'm trying to use the google analytics core reporting api and have the following troubles. i know how to fetch that i want the problem i have is: i want to use a filter only for specific metrics and not for all of them. is this even possible?*".

IT7 **Change/Deprecation.** Functions are changed or deprecated because of web API updates. An issue description in the SO question '15056383' is: "*i've been using a class based on hernan amiune library for the facebook graph api to allow website user to write to their friends walls from my code. with the recent february changes the graph api method to do this is no longer available*".

IT8 **Limitation.** A function has a limited functionality, e.g., the maximum number of requests per day and the number of items per request. An issue description in the SO question '31092862' is: "*i want to get all of my youtube subscriptions using the youtube api. however, google limits the amount of items that get returned in api calls to 50*".

IT9 **Lack of programming language support.** There is lack of support (e.g., a client wrapper or sample code) for a specific programming language. An issue description in the SO question '8971374' is: "*i am using wikipedia api to get result as article. for example to get result on india i'm using action=parse and page=india. can anyone please tell me how to use this using java*".

IT10 **Inconsistent result.** The result of a function is inconsistent (including success or failure) when requesting the function on different platforms (e.g., web browsers and devices) or using different methods (e.g., CRUL [56] and Node.js). An issue description in the SO question '18269085' is: "*the standard geolocation code from google does not work in mobile browsers (android chrome, standard android browser) but in desktop browser it works fine... why?*".

IT11 **Instability.** A function is unstable. For example, a function works for a period but suddenly fails. An issue description in the SO question '49229881' is: "*i'm using odoo v11 in windows localhost and i'm still beginner. recently, i tried to edit posticket and suddenly, the point of sale module stops loading it keep blank page. i have cleared cache for browser and restart odoo service, it worked*".

IT12 **Unexpected result.** The result of a function is unexpected, e.g., missing a required field. An issue description in the SO question '35242742' is: "*the current problem is that the confirmation message, which is sent using twilio*

sms api, is getting segmented into 2 or more parts. this is messing up the confirmation link in the sms".

IT13 **Unsupported function.** A desired function is not supported. An issue description in the SO question '42085202' is: "*the current kik bot api gives very limited privileges and does not allow monitoring group messages*".

IT14 **Unsatisfactory performance.** The performance (e.g., response time and availability) of a function is not satisfactory. An issue description in the SO question '42853484' is: "*since 10 march 2017, i am experiencing the slowness in the luis api*".

IT15 **Implementation problem.** A function has implementation problems or bugs. An issue description in the SO question '46475320' is: "*safari 11 youtube api bug. rapid play pause and failure to autoplay*".

IT16 **Difficult to process result.** The result of a function is difficult to process. An issue description in the SO question '36658752' is: "*i'm using a php wrapper api for ups rating api, not the ups dev kit directly. i am trying to echo the monetaryvalue of the shipping quote contained in this array. ... i'm in a little over my head as i've never work with an array of this complexity*".

IC3. Documentation. We identified five issue types related to the documentation of web APIs.

IT17 **Missing.** The documentation is not found. An issue description in the SO question '52044218' is: "*I'm using ms translation speech and build up custom translator model (customtranslator.ai) however, i cannot find the document of custom translator api*".

IT18 **Incorrect.** The documentation contains incorrect content, e.g., the explanation of a function does not match the actual result, an example code or an instruction does not work, etc. An issue description in the SO question '37195607' is: "*i'm using the slack web api to post messages to a channel in go. i'm trying to support multi line messages in the text field. according to the documentation simply adding a \n should work but it not working*".

IT19 **Incomplete.** The documentation may not contain all user desired or important content, e.g., the explanation of a functional parameter, an example code for requesting a function, etc. An issue description in the SO question '49106614' is: "*i need information about the campaign type but i don't find this field in the api documentation*".

IT20 **Out-of-date.** The documentation does not match the latest web API. An issue description in the SO question '15244280' is: "*does anyone know where i can find the latest documention for bings api ... even their own website has the wrong url in the word docs i have been reading*".

IT21 **Difficult to read/understand.** The documentation is difficult to read or understand, which can be caused by the quality of the documentation or by the mismatch between the description style of the documentation and users' information processing styles. More specifically, when solving problems, females are more likely to use comprehensive information processing styles (i.e., gathering fairly complete information before proceeding), whereas males are more likely to use selective styles (i.e., following the first promising information and then backtracking in depth-first order) [57]. An issue description in the SO question '26522088' is: "*the documentation is pretty vague and i've searched long and hard for examples samples but to no avail*".

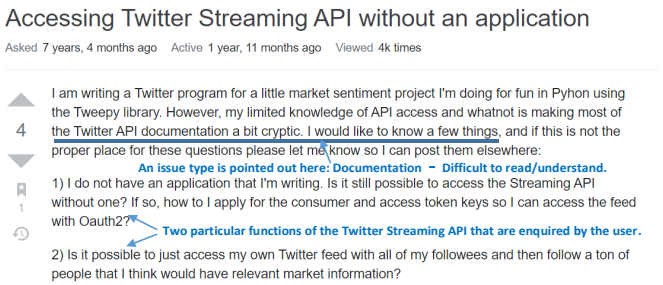

Fig. 3. Part of an example SO question that describes two web API issue types, i.e., IT6 and IT21 shown in Table 13.

IC4. Others. This category contains issue types that do not belong to the four categories explained above. We identify five issue types belonging to this category.

IT22 **Checking for web API usage.** A user wants to know whether a web API can be used in a technical context (e.g., some specific libraries or frameworks). An issue description in the SO question '43244489' is: "*can the paypal adaptive payments api be employed with kinvey mbaas for a mobile ios application written in swift?*".

IT23 **Enquiring about web API implementation.** A user wants to know the implementation of a function. An issue description in the SO question '28205748' is: "*i just want to ask how does the uber appz track realtime? do they request to server every 1 second to updates the position of car so that the car moves?*".

IT24 **Web API searching.** A user needs to search for a web API that can achieve a requirement or an alternative web API. An issue description in the SO question '16694506' is: "*is there a way to access dropbox events page using some api?*".

IT25 **Unsuitable charge.** Web API charges (i.e., pricing) are not suitable, e.g., unwanted functions are required to be paid. An issue description in the SO question '12003277' is: "*i have to pay 110$ per year which is a lot giving the fact that i want to use only 3 fonts. so my question is can't i just buy those 3 fonts and embed anyhow i want?*".

IT26 **Difficult to sign up.** It is difficult to sign up a web API. This issue type is identified from the other issue description given by a respondent: "*signing up for a simple api becoming too hard*".

3.2.2 Discussions of the Top Ten Web API Issue Types Reported in SO Questions

As listed in Table 13, the top ten web API issue types identified from SO questions are {IT6, IT5, IT22, IT23, IT1, IT10, IT19, IT16, IT3, IT12}. We perform an in-depth analysis of the possible reasons why users frequently report the top ten issue types by looking deeper into a number of SO questions that discuss them.

For the issue type ***function*** - *enquiring about a particular function* (IT6), two possible reasons that users frequently report issues of this type are: 1) users may want to implement various or customized functionality using a web API. It is difficult for web API providers to consider all possible use

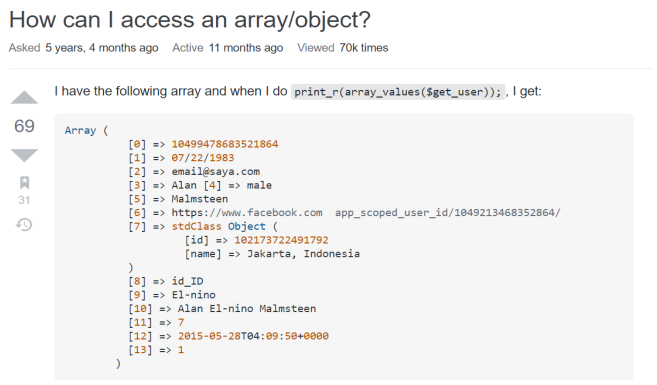

Fig. 4. Part of an example SO question that describes a complex array returned by a web API that is difficult to process.

scenarios of their web APIs and describe the instructions for many different use scenarios in the documentation; and 2) it may not be easy for users to fully examine the documentation (especially when the documentation is not well written or too lengthy) to find a possible function. Figure 3 shows part of an example SO question '16650361' that describes two issue types, i.e., IT6 and ***documentation*** - *difficult to read/understand* (IT21). Users often resort to the millions of developers in SO for the desired functions of web APIs, which may be a good choice to quickly get the answers and avoid wasting too much time and effort.

For the issue type ***function*** - *request failure with unknown reasons* (IT5), since the functions of web APIs are requested via the internet, it is difficult for users to figure out the causes of issues without having the implementation code of web APIs. Therefore, users often ask for help by reporting the encountered issues in SO.

For the issue type ***others*** - *checking for web API usage* (IT22), two possible reasons that users often report issues of this type are: 1) it may be difficult for some users (especially the users unfamiliar with web APIs) to successfully access a web API; and 2) users may want to figure out whether a web API can be used in some technical contexts, e.g., using the web API with some other web APIs, libraries, or frameworks. An example SO question is given in the explanation of IT22.

For the issue type ***others*** - *enquiring about web API implementation* (IT23), the main reason that users frequently enquire about the implementation of web APIs is: when encountering unexpected errors or results of a web API, e.g., IT1 and IT10, users may want to have a better understanding of the implementation details of the web API in order to pinpoint the causes of the issues. For example, users tried to understand how the authorization mechanism or a function is implemented in a web API.

For the two issue types ***authorization*** - *error* (IT1) and ***authorization*** - *enquiring about acquisition* (IT3), the possible reasons that users frequently report issues of the two types include: 1) the authorization mechanism of a web API is not implemented using a standard method (e.g., OAuth), which makes it difficult for users to obtain an API key/access token; 2) the instructions for requesting an API key/access token are incorrect or not clearly described in the documentation; and 3) it is difficult for users who are unfamiliar with web APIs to know how to obtain an API key/access token.

For the issue type ***function*** - *inconsistent result* (IT10), users may request the functions of a web API from different browsers (e.g., Chrome and Internet Explorer) in different platforms (e.g., Android and Windows) or using different languages (e.g., Javacript and Node.js). It is difficult for web API providers to make sure that the results of a function are consistent from various browsers running in different platforms and from various requesting languages.

For the issue type ***documentation*** - *incomplete* (IT19), three possible reasons that users frequently report issues of this type are: 1) the documentation of web APIs may not include all necessary information, e.g., the explanation of a parameter or an example code for using a function; 2) it is difficult for web API providers to describe all possible information interested by various users in the documentation; and 3) the documentation is not well written or too lengthy, which makes it difficult for users to find the desired content.

For the issue type ***function*** - *difficult to process result* (IT16), users often report issues of this type can be due to two possible reasons: 1) the result of a function does not use a standard format, e.g., JSON; and 2) the result of a function is too complex to process. Figure 4 shows part of an example SO question '30680938'. The question shows a complex array returned by the Facebook SDK 4, which is difficult to process for many users (as the question has been viewed 70 thousand times).

For the issue type ***function*** - *unexpected result* (IT12), the main reason that users often report issues of this type is: for many web APIs, the result of a function may not be desired by all potential users. An example SO question is given in the explanation of IT12.

3.2.3 Summary of Findings

From Table 13, we have the following findings:

- For each of the 24 issue types identified from SO questions, it is encountered by at least 29% of the survey respondents. Only two additional issue types, i.e., IT4 and IT26, are expressed by four respondents. The results mean that the issue types identified from SO questions have good coverage of the common web API issues. In the survey question that asks for web API issues outside the 24 issue types identified from SO questions, two respondents express comments on the 24 issue types: "*this is a great list. great job on summarizing!*" and "*those are the most common*". Therefore, we are confident that the 26 issue types identified from SO questions and the survey responses provide a comprehensive list of web API issues experienced by users.
- For the 24 issue types identified from SO questions, the percentages of SO questions (i.e., 0%-23%) are notably different from the percentages of respondents (i.e., 29%-83%). The top ten issue types with the highest percentages of respondents, i.e., {IT8, IT19, IT21, IT20, IT17, IT5, IT7, IT18, IT1, IT2}, are different from those with the highest percentages of SO questions. Only 1% of the SO questions contain the issue type IT2, while 68% of the respondents experienced IT2. The differences between the issue types identified from SO questions and those expressed by the respondents of the survey could be explained by two possible reasons: 1) users may report issues using channels other than SO, e.g., the web API

TABLE 14
Ten of the 26 web API issue types (Table 13) that have been reported in prior studies.

Issue Type ID	Prior Studies that Have Reported Web API Issues of the Issue Type
IT5	[58], [59]
IT7	[20], [26], [29], [60], [61], [62]
IT10	[58], [63], [64]
IT11	[65]
IT12	[58]
IT14	[58], [65]
IT15	[58], [65], [66]
IT18	[21], [64], [67]
IT19	[21], [67]
IT20	[61], [67]

projects at GitHub; and 2) users may not mention every issue in a SO question as some issues are not critical or can be solved by themselves. For example, ***authorization*** - *expiration* (IT2) is a frequently encountered issue type; but in most cases it can be easily solved by refreshing the expired API key/access token (if the refresh method is well documented). When posting a SO question, users may focus on describing more serious issues related to authorization and function while ignoring documentation issues.

After extensive literature search on the existing work on web API issues [20], [21], [26], [29], [58], [59], [60], [61], [63], [64], [65], [66], [67], we find that 16 of the 26 issue types are reported in this work. Table 14 lists the ten issue types, i.e., { IT5, IT7, IT10, IT11, IT12, IT14, IT15, IT18, IT19, IT20 }, reported in prior studies. For example, Wang et al. [20] studied the types of web API changes, e.g., *add method* and *change response format*, during the evolution of web APIs. Li et al. [58] studied the issues of web APIs by leveraging the discussion forums of commercial cloud platforms, e.g., the Amazon outage reports and Netflix technical blogs. They summarized a taxonomy of 15 types of web API failures and faults. The taxonomy includes five failure or fault types that correspond to five issue types reported in this work, i.e., IT5, IT10, IT12, IT14, and IT15. For example, the failure type *unexpected content* corresponds to our issue type ***function*** - *unexpected result* (IT12).

Table 15 presents the percentages of respondents in the six groups (Table 9) who experienced each of the 26 issue types. For 24 issue types (except IT2 and IT26), the percentages of respondents in APINumS are lower than those in APINumM; and for 17 issue types (except IT3, IT4, IT10, IT12, IT18, IT19, IT21, IT22, and IT23), the percentages of respondents in APINumM are lower than those in APINumL. We observe similar results among the other three groups. These findings could be due to the reason that the quality of web APIs is mainly determined by web API providers during the development and maintenance (e.g., update) of web APIs. However, it is difficult to eliminate all possible issues for several factors. For example, web API providers may lack experience or have limited resources (e.g., expense budget) in developing and maintaining web APIs; and various and complex use scenarios of web APIs could be performed by users. Therefore, users may encounter more issues as they have used more web APIs.

Table 16 presents the results of multiple comparisons between any two groups in *APINumGSet* or *ExpGSet* on each of the 26 issue types. Most of the significant differences are observed between the five group pairs (except 'APINumM vs. APINumL') on 13 issue types: IT5, IT8, IT9, IT12, IT14-IT21, and IT25. For example, in terms of ***function*** - *request failure with unknown reasons* (IT5), the differences between the two group pairs 'APINumS vs. APINumM' and 'APINumM vs. APINumL' are significant with p-value < 0.05 and 0.01, respectively. The relative contrast effects for the two group pairs are 0.34 and 0.33, respectively, which imply that as the number of used web APIs increases, users tend to experience more request failures without knowing the reasons. In addition to the explanations above, a supplementary explanation of this result is that since users generally do not have the implementation code of web APIs, it is usually difficult for them to pinpoint the reasons for request failures.

Different from the existing studies that focus on particular web API issues (e.g., web API changes and documentation reliability), we provide a comprehensive list of 26 web API issue types (i.e., IT1-IT26 listed in Table 13) by analyzing 1,885 SO questions and 191 survey responses. The issue types are grouped into four high-level categories: ***authorization***, ***function***, ***documentation***, *and* ***others***. *16 of the 26 issue types (except those listed in Table 14) are reported in our work, such as* ***authentication*** *- error (IT1) and* ***function*** *- difficult to process result (IT16).*
We find that the top ten issue types reported in SO questions are notably different from those expressed by the survey respondents, which can be explained by two possible reasons: 1) users may report issues in channels other than SO; and 2) users may leave out non-critical issues (e.g., documentation issues) to focus on more serious authorization or function issues when posting a question in SO.
The web API issue types can help web API developers understand why the survey respondents express the 11 categories of expectations on web APIs (Table 17). Based on the issue types and expectations, web API developers can improve web APIs by adopting the respondents' suggestions expressed in the expectations to reduce web API issues. The guidelines are described in Section 4.1.

3.3 RQ3. What expectations do users have for web API providers to facilitate the use of web APIs?

Motivation. As mentioned in Section 3.2, web API providers can become more aware of developing better web APIs from our identified web API issue types. For example, from the two issue types ***function*** - *change/deprecation* (IT7) and ***documentation*** - *out-of-date* (IT20), web API providers would be more aware of keeping the documentation in sync with the updates of web APIs and clearly describe the changed/deprecated functions. However, it may be difficult to extract practices from the issue types when web API development and management require more in-depth knowledge (e.g., design, testing, and standardization) and/or rich experience using web APIs. We strieve for deriving useful suggestions on the development and management of web APIs from users.

Approach. In the user survey, we create an open-ended question to collect respondents' expectations on the development and management of web APIs.

Results. Table 17 presents 11 categories of expectations on web APIs that are summarized from the survey responses. We

TABLE 15
Percentages of the respondents in the six groups (Table 9) who have encountered each of the 26 web API issue types (Table 13).

Issue Type ID	APINumS	APINumM	APINumL	ExpL	ExpM	ExpH
IT1	56%	65%	76%	57%	74%	69%
IT2	69%	56%	76%	59%	70%	72%
IT3	34%	51%	49%	39%	47%	61%
IT4	–	3%	1%	–	2%	3%
IT5	44%	75%	78%	45%	78%	86%
IT6	44%	51%	54%	51%	50%	56%
IT7	56%	62%	81%	73%	67%	75%
IT8	66%	78%	92%	67%	87%	92%
IT9	25%	59%	68%	41%	56%	86%
IT10	12%	40%	36%	20%	37%	42%
IT11	22%	41%	53%	29%	44%	61%
IT12	31%	63%	58%	33%	60%	72%
IT13	44%	54%	68%	51%	56%	81%
IT14	9%	49%	69%	27%	55%	78%
IT15	22%	47%	60%	22%	53%	75%
IT16	–	38%	54%	8%	47%	58%
IT17	31%	81%	82%	59%	72%	97%
IT18	31%	82%	75%	51%	70%	97%
IT19	34%	90%	85%	59%	80%	97%
IT20	34%	76%	86%	63%	75%	86%
IT21	44%	81%	81%	59%	80%	81%
IT22	22%	46%	40%	31%	38%	53%
IT23	22%	35%	35%	35%	25%	53%
IT24	31%	46%	56%	53%	44%	53%
IT25	9%	31%	35%	8%	36%	39%
IT26	–	–	1%	–	1%	–

TABLE 16
The results of multiple comparisons between any two respondent groups in APINumGSet and ExpGSet (Table 9), with respect to each of the 26 web API issue types (Table 13). Each cell (p, e) presents the p-value (p) and the relative contrast effect (e) of a pairwise comparison on an issue type. '–' means there is no significance (i.e., $p \geq 0.05$).

Issue Type ID	APINumS vs. APINumM	APINumS vs. APINumL	APINumM vs. APINumL	ExpL vs. ExpM	ExpL vs. ExpH	ExpM vs. ExpH
IT1	(–, 0.46)	(–, 0.40)	(–, 0.44)	(–, 0.58)	(–, 0.44)	(–, 0.52)
IT2	(–, 0.56)	(–, 0.47)	(<0.05, 0.40)	(–, 0.55)	(–, 0.43)	(–, 0.49)
IT3	(–, 0.41)	(–, 0.42)	(–, 0.51)	(–, 0.54)	(–, 0.39)	(–, 0.43)
IT4	(–, 0.48)	(–, 0.49)	(–, 0.51)	(–, 0.51)	(–, 0.49)	(–, 0.50)
IT5	(<0.05, 0.34)	(<0.01, 0.33)	(–, 0.48)	(<0.001, 0.67)	(<0.001, 0.29)	(–, 0.46)
IT6	(–, 0.46)	(–, 0.45)	(–, 0.49)	(–, 0.49)	(–, 0.48)	(–, 0.47)
IT7	(–, 0.47)	(<0.05, 0.38)	(<0.05, 0.40)	(–, 0.47)	(–, 0.49)	(–, 0.46)
IT8	(–, 0.44)	(<0.05, 0.37)	(<0.05, 0.43)	(<0.05, 0.60)	(<0.05, 0.38)	(–, 0.48)
IT9	(<0.01, 0.33)	(<0.001, 0.28)	(–, 0.45)	(–, 0.57)	(<0.001, 0.27)	(<0.001, 0.35)
IT10	(<0.01, 0.36)	(<0.01, 0.38)	(–, 0.52)	(–, 0.58)	(–, 0.39)	(–, 0.48)
IT11	(–, 0.40)	(<0.01, 0.35)	(–, 0.44)	(–, 0.58)	(<0.01, 0.34)	(–, 0.42)
IT12	(<0.01, 0.34)	(<0.05, 0.36)	(–, 0.53)	(<0.01, 0.64)	(<0.001, 0.30)	(–, 0.44)
IT13	(–, 0.45)	(–, 0.38)	(–, 0.43)	(–, 0.52)	(<0.01, 0.35)	(<0.01, 0.38)
IT14	(<0.001, 0.30)	(<0.001, 0.20)	(<0.05, 0.40)	(<0.01, 0.64)	(<0.001, 0.24)	(<0.05, 0.39)
IT15	(<0.05, 0.37)	(<0.001, 0.31)	(–, 0.43)	(<0.001, 0.65)	(<0.001, 0.24)	(<0.05, 0.39)
IT16	(<0.001, 0.31)	(<0.001, 0.23)	(–, 0.42)	(<0.001, 0.69)	(<0.001, 0.25)	(–, 0.44)
IT17	(<0.001, 0.25)	(<0.001, 0.24)	(–, 0.49)	(–, 0.56)	(<0.001, 0.31)	(<0.001, 0.37)
IT18	(<0.001, 0.24)	(<0.001, 0.28)	(–, 0.54)	(–, 0.59)	(<0.001, 0.27)	(<0.001, 0.36)
IT19	(<0.001, 0.22)	(<0.001, 0.25)	(–, 0.53)	(<0.05, 0.60)	(<0.001, 0.31)	(<0.01, 0.41)
IT20	(<0.001, 0.29)	(<0.001, 0.24)	(–, 0.45)	(–, 0.56)	(<0.05, 0.39)	(–, 0.44)
IT21	(<0.01, 0.31)	(<0.001, 0.31)	(–, 0.50)	(<0.05, 0.60)	(–, 0.39)	(–, 0.50)
IT22	(<0.05, 0.38)	(–, 0.41)	(–, 0.53)	(–, 0.54)	(–, 0.39)	(–, 0.42)
IT23	(–, 0.43)	(–, 0.43)	(–, 0.50)	(–, 0.45)	(–, 0.41)	(<0.05, 0.36)
IT24	(–, 0.43)	(<0.05, 0.38)	(–, 0.45)	(–, 0.46)	(–, 0.50)	(–, 0.46)
IT25	(<0.05, 0.39)	(<0.01, 0.37)	(–, 0.48)	(<0.001, 0.64)	(<0.01, 0.35)	(–, 0.48)
IT26	(–, 0.50)	(–, 0.49)	(–, 0.49)	(–, 0.51)	(–, 0.50)	(–, 0.51)

explain each expectation category as follows.

3.3.1 Explanations of 11 Expectation Categories

EC1. Authorization. Users often need to obtain the authorization (e.g., an API key) required by a web API before using the functions. We summarize two specific expectations on the authorization of web APIs from the expectation descriptions specified by ten respondents.

- **Standard implementation.** The authorization mechanism of web APIs should be implemented using standard flows. An expectation description is: "*use of standard authentication flows*".
- **Automatic request.** The authorization should be able to be requested automatically. An expectation description is: "*api key can be requested automatically is nice*".

EC2. Development. This category contains expectations on the development process of web APIs. We summarize eight specific expectations specified by 54 respondents.

- **Use of standard tools.** It should be better to develop web APIs using standard tools, e.g., Swagger [30] and OpenREST [68]. An expectation description is: "*tools such as swagger/openrest should be automated as part of the development process*".
- **Stateless and user-centered design.** The design of

TABLE 17
Eleven categories of the respondents' expectations on web APIs.

Expectation Category ID	Expectation Category	# Respondents	Percentage of Respondents
EC1	Authorization	10	5%
EC2	Development	54	28%
EC3	Documentation	63	33%
EC4	Update	12	6%
EC5	SDK/Library	24	13%
EC6	Monitoring	2	1%
EC7	Performance	13	7%
EC8	Issue support	15	8%
EC9	Easy trial	7	4%
EC10	Pricing	5	3%
EC11	Discoverability	1	1%

web APIs should be stateless and user-centered (i.e., making web APIs easy to understand and use from the perspective of users). Two expectation descriptions are: "*api should be designed for the consumers*" and "*i expect us devs to look at the products through the eyes of users*".

- **Testing.** Web APIs should be thoroughly tested during the development. Two expectation descriptions are: "*thoroughly test the apis before starting to develop any internal function*" and "*i think standardizing something as simple as unit testing will boost the overall quality of apis*".
- **REST support.** Web APIs should support a REST specification. An expectation description is: "*support rest*".
- **Simple, consistent, and explainable results.** The results of functions should be simple and consistent on different platforms and endpoints. Rate limitations should also be explained in the results. Two expectation descriptions are: "*results should be as simple and light as possible*" and "*has consistency across all platforms/endpoints*".
- **Stable.** The functions of web APIs should be stable. An expectation description is: "*api is stable*".
- **Failure control.** Web APIs should provide a control of failures. An expectation description is: "*expect failure and develop fail-first systems with retry possibilities*".
- **Clear error messages.** The error messages of web APIs should be clear for users to understand. An expectation description is: "*good error messages with pointer urls to related entities in the returned data*".

EC3. Documentation. This category contains expectations on the documentation of web APIs. The following specific expectations are summarized from 63 respondents:

- **Unrestricted access.** The documentation should be accessible without restriction. An expectation description is: "*no login required to see documentation*".
- **Well-written.** The documentation should be well-written, such that users can easily find the desired functions and know how to use the functions. An expectation description is: "*documentation should be well-written*".
- **Complete.** The documentation should be complete, i.e., including all information that is necessary for users to understand and use the web API. Two expectation descriptions are: "*documentation should be complete*" and "*fully documents all side-effects of functions*".
- **Consistent with web API implementation.** The documentation should be consistent with the implementation code of the web API. An expectation description is: "*ensuring documentation is generated from code*".
- **Up-to-date.** The documentation should be up-to-date. An expectation description is: "*documentation is updated along with the api*".
- **Include code examples.** The documentation should include code examples on how to use the functions. An expectation description is: "*documentation with examples would be helpful*".

EC4. Update. This category contains expectations on the update of web APIs. Three specific expectations are summarized from 12 respondents, as described below:

- **Update with users' feedback.** Web APIs should be updated according to users' feedback. An expectation description is: "*get feedback then update*".
- **Avoid breaking changes.** Web APIs should avoid breaking changes, e.g., no change of result formats. Two expectation descriptions are: "*hopefully without breaking changes*" and "*the result format must not change (in this cases we must to adapt our code)*".
- **Notify users about changes and deprecations.** Users should be notified about the changes and deprecations. Two expectation descriptions are: "*be public about changes*" and "*notify the users about deprecated functions*".

EC5. SDK/Library. This category contains expectations on the SDKs and libraries that are created for facilitating the use of web APIs. 24 respondents expect standard SDKs and client libraries of web APIs for major programming languages, e.g., Java and Javacript. Example expectation descriptions are: "*sdks for major languages*", "*a standard sdk to use the api*", and "*try writing a library for common programming languages which could easily be imported and used by a developer*".

EC6. Monitoring. Two respondents expect that web APIs should be monitored in case of functional or non-functional issues. An expectation description is: "*putting enough effort into monitoring*".

EC7. Performance. This category contains expectations on the performance of web APIs. 13 respondents expect that web APIs should be highly available, fast, reliable, and secure. Example expectation descriptions are: "*it must be 100% available*", "*to be reliable and quick*", and "*i expect to be able to fully mutate all of my personal data (or the data of my users, assuming they have granted that authorization)*".

EC8. Issue support. 15 respondents expect instant support for web API issues. Example expectation descriptions are: "*fix the issues always*", "*pm has to take notice of the bottlenecks and resolve them asap*", and "*it's not viable not to have any problems, but if they're resolved rapidly, then it's a good experience*".

EC9. Easy trial. Seven respondents expect that web APIs should provide easy trial for users. Two expectation descriptions are: "*providing a sandbox for trying out the api is always appreciated*" and "*easy trial and testing*".

EC10. Pricing. Five respondents expect that the pricing of web APIs should be cheap or reasonable. Two expectation descriptions are: "*cheap prices*" and "*in production something that works well at a reasonable price*".

TABLE 18
Percentages of the respondents in the six groups (Table 9) who express expectations in each of the 11 categories of user expectations on web APIs (Table 17).

Expectation Category ID	APINumS	APINumM	APINumL	ExpL	ExpM	ExpH
EC1	–	–	11%	–	8%	6%
EC2	19%	28%	32%	31%	27%	28%
EC3	34%	32%	33%	24%	37%	33%
EC4	–	3%	11%	12%	1%	14%
EC5	12%	15%	11%	2%	17%	14%
EC6	–	1%	1%	–	–	6%
EC7	–	9%	8%	4%	7%	11%
EC8	6%	9%	8%	10%	8%	6%
EC9	12%	1%	2%	8%	–	8%
EC10	–	1%	4%	4%	3%	–
EC11	–	–	1%	–	1%	–

TABLE 19
The results of multiple comparisons between any two respondent groups in APINumGSet and ExpGSet (Table 9), with respect to each of the 11 categories of expectations on web APIs (Table 17). Each cell (p, e) presents the p-value (p) and the relative contrast effect (e) of a pairwise comparison on an expectation category. '–' means there is no significance (i.e., $p \geq 0.05$).

Expectation Category ID	APINumS vs. APINumM	APINumS vs. APINumL	APINumM vs. APINumL	ExpL vs. ExpM	ExpL vs. ExpH	ExpM vs. ExpH
EC1	(–, 0.50)	(<0.01, 0.45)	(<0.01, 0.45)	(<0.05, 0.54)	(–, 0.47)	(–, 0.51)
EC2	(–, 0.45)	(–, 0.43)	(–, 0.48)	(–, 0.48)	(–, 0.51)	(–, 0.50)
EC3	(–, 0.51)	(–, 0.51)	(–, 0.50)	(–, 0.56)	(–, 0.46)	(–, 0.52)
EC4	(–, 0.48)	(<0.01, 0.45)	(–, 0.46)	(–, 0.44)	(–, 0.49)	(–, 0.43)
EC5	(–, 0.49)	(–, 0.51)	(–, 0.52)	(<0.01, 0.57)	(–, 0.44)	(–, 0.52)
EC6	(–, 0.49)	(–, 0.49)	(–, 0.50)	(–, 0.50)	(–, 0.47)	(–, 0.47)
EC7	(<0.05, 0.46)	(<0.05, 0.46)	(–, 0.51)	(–, 0.51)	(–, 0.47)	(–, 0.48)
EC8	(–, 0.49)	(–, 0.49)	(–, 0.51)	(–, 0.49)	(–, 0.52)	(–, 0.51)
EC9	(–, 0.56)	(–, 0.55)	(–, 0.50)	(–, 0.46)	(–, 0.50)	(–, 0.46)
EC10	(–, 0.49)	(–, 0.48)	(–, 0.48)	(–, 0.49)	(–, 0.52)	(–, 0.51)
EC11	(–, 0.50)	(–, 0.49)	(–, 0.49)	(–, 0.51)	(–, 0.50)	(–, 0.51)

EC11. Discoverability. One respondent expects that web APIs should be able to be discovered by users. The expectation description is: "*there should be discoverability*".

3.3.2 Discussions of the Top Three Expectation Categories

As listed in Table 17, the top three categories of the respondents' expectations on web APIs are *documentation* (EC3), *development* (EC2), and *SDK/library* (EC5). The possible reasons for the categories are explained as follows.

For the category *documentation* (EC3), the documentation of a web API is the main resource offered by the web API provider to help users understand and use the web API. However, users often experience documentation issues (Table 13). The respondents express many expectations on the high-quality documentation, e.g., complete and up-to-date, which could help users easily find the desired functions and solutions to the encountered issues.

For the category *development* (EC2), a possible reason that the respondents express many expectations of this category is: if web API providers could ensure the quality of their web APIs, users could encounter fewer issues when using the web APIs. For example, if web APIs could provide simple, consistent, and explainable results of the functions, users may not encounter some types of issues, e.g., ***function*** *- inconsistent result* and ***function*** *- difficult to process result*.

For the category *SDK/library* (EC5), a possible reason that the respondents express many expectations belonging to this category is: SDKs and libraries can help users easily request the functions of web APIs and avoid many issues by providing high-level functional interfaces or templates.

3.3.3 Summary of Findings

We find that the 11 categories of expectations cover diverse aspects of web APIs, e.g., documentation, development, and pricing, which could be explained by the fact that the respondents experience various issues of web APIs (Table 13).

Table 18 presents the percentages of respondents in the six groups (Table 9) who express expectations belonging to each category. There are differences among the six groups in terms of the percentages. For example, the percentage of the category *easy trial* (EC9) is relatively high (i.e., 12%) in the group APINumS, while those in the other five groups are low (i.e., $\leq 8\%$), indicating that users in APINumS more strongly expect that the functions of a web API can be easily tried out. This result may be due to a possible reason that users with a small number of used web APIs may lack confidence about judging the functionality and performance of web APIs based on the documentation and some other descriptions. They prefer trying out the functions of a web API to see whether the web API can meet their requirements. In terms of the category *SDK/library* (EC5), the percentages in ExpM and ExpH are 17% and 14%, respectively, and are much higher than that in ExpL (i.e., 2%), which could be explained as follows. With relatively high experience using web APIs, users may realize that SDKs and libraries can greatly facilitate the requests of the functions of web APIs, especially when the users need to work with an unfamiliar programming language. Therefore, they will more expect that web APIs can provide SDKs and libraries. Despite the differences, the top two categories with the highest percentages, i.e., *development* (EC3) and *documentation* (EC2), are the same in the six groups.

Table 19 presents the results of multiple comparisons be-

TABLE 20
Correlation between the 26 web API issue types (Table 13) and the 11 categories of user expectations on web APIs (Table 17). Each cell presents the number of respondents who have encountered the corresponding issue type and express expectations in the corresponding category.

	EC1	EC2	EC3	EC4	EC5	EC6	EC7	EC8	EC9	EC10	EC11
IT1	8	40	41	8	13	2	10	4	7	–	1
IT2	6	38	42	10	12	2	10	9	7	–	1
IT3	6	27	32	4	13	–	–	10	2	2	1
IT4	–	3	–	–	2	–	–	–	–	–	–
IT5	8	44	52	8	18	2	11	12	7	1	1
IT6	4	32	49	6	10	–	3	13	6	1	1
IT7	8	47	54	12	12	2	13	15	7	3	1
IT8	10	41	57	10	20	2	13	12	7	5	1
IT9	8	33	44	8	14	2	11	9	7	3	1
IT10	4	13	26	5	6	1	6	4	6	1	1
IT11	6	33	29	6	5	2	8	12	7	2	1
IT12	8	32	44	7	14	2	7	13	7	1	1
IT13	6	41	43	10	7	2	9	15	3	1	1
IT14	8	34	41	9	9	2	13	9	3	3	1
IT15	6	38	37	5	10	2	9	10	3	1	1
IT16	4	25	29	5	7	2	8	10	3	–	1
IT17	6	42	57	12	20	2	11	15	7	3	1
IT18	4	42	53	10	18	2	11	13	7	1	1
IT19	6	48	56	10	20	2	11	15	7	1	1
IT20	8	45	58	12	17	1	10	13	6	3	1
IT21	8	45	63	10	17	1	12	15	6	3	1
IT22	6	16	25	4	12	–	2	4	2	–	1
IT23	4	20	32	8	11	–	2	5	2	–	1
IT24	6	32	38	10	8	1	8	6	3	5	1
IT25	2	22	24	4	6	–	5	2	2	1	1
IT26	–	1	1	–	–	–	–	–	–	–	–

tween any two groups in *APINumGSet* or *ExpGSet* on each of the 11 expectation categories. There are significant differences between the five group pairs (except 'ExpL vs. ExpH') on four categories: EC1, EC4, EC5, and EC7. For example, in terms of *authorization* (EC1), the differences between the two group pairs 'APINumS vs. APINumL' and 'APINumM vs. APINumL' are significant with p-value < 0.01; and the relative contrast effects are 0.45, which implies that users in APINumL tend to have more expectations on authorization. A possible reason for this result is that users with a large number of used web APIs may often encounter authorization issues and find that it is not easy to obtain the authorization. Therefore, they more strongly expect that web API developers implement the authorization using standard flows and enable automatic request of the authorization.

Table 20 presents the correlation between the 26 web API issue types (Table 13) and the 11 expectation categories. Each cell (ITi, ECj) (i = 1-26 and j = 1-11) contains the number of respondents who have encountered the web API issue type 'ITi' and meanwhile express expectations in the category 'ECj'. From the perspective of issue types, the top two expectation categories with the maximum numbers of respondents, i.e., *development* (EC2) and *documentation* (EC3), are the same for 25 issue types (except IT4). For 18 issue types (except IT6, IT7, IT11, IT13, IT14, IT16, IT24, and IT26), the category *SDK/library* (EC5) is in the top three categories. This result could be explained by the reason that SDKs and libraries of web APIs can help users avoid many issues by providing high-level functional interfaces for requesting the functions. The category *issue support* (EC8) has relatively strong correlation with 18 issue types (i.e., EC8 is in the top five categories of those issue types), since an issue support is generally necessary for users to solve encountered issues.

After enquiring the survey respondents about their experienced web API issues, we derive the respondents' expectations for web API developers and managers related to improving web APIs and facilitating the use of web APIs. A set of user expectations on web APIs that belong to 11 categories (i.e., EC1-EC11 listed in Table 17) are summarized by analyzing the survey responses. For example, the authorization is expected to be implemented using standard flows; and the results of functions are expected to be consistent on different platforms. Following the expectations, web API developers and managers can improve web APIs and reduce the web API issues encountered by users. A number of guidelines are described in Section 4.1.

4 Implications

In this section, we provide implications for web API developers (or providers) and registry managers based on the results of our study to improve the practice of web APIs.

4.1 Implications for Web API Developers

1) *Offer well-structured, correct, and complete web API documentation with code examples.* We find that *well-organized documentation* is the most important feature of web APIs when the respondents of our survey decide to shortlist a web API for testing. The web API issue types related to documentation are frequently encountered by the respondents. The documentation is also the most required expectation. Therefore, it is important for web API developers to provide high-quality documentation for web APIs. Even though good documentation is a general practice, some web API developers (especially inexperienced developers) may not have a very good understanding of the requirements of high-quality documentation. Based on our findings, we recommend web API developers to: 1) provide well-structured documentation

by clearly describing different content (e.g., authorization, functions of different topics, and charge of services), so that users can easily find their needed information in the documentation; 2) check the correctness of the documentation, e.g., the parameters and results of functions are consistent with the web API implementation, the instructions are workable, etc.; 3) check the completeness of the documentation, i.e., whether the documentation contains the explanations of every necessary parameter and result field of functions; and 4) include code examples for important functions in the documentation. As expressed by several respondents, the documentation of the web API *GitHub* is a good reference.

2) *Provide simple, explainable, and consistent results of web API functions.* We find that ***function*** - *inconsistent result* and ***function*** - *difficult to process result* are two of the top ten web API issue types reported in SO questions. Some respondents of our survey express that the results of functions should be simple, consistent, and explainable. Based on these findings, we recommend web API developers to: 1) keep the results of web API functions simple without too complex or nested data structures, e.g., the counter-example shown in Fig. 4; 2) make the results of functions explainable, e.g., naming the result fields using meaningful words and including rate limitations in the results; and 3) test the functions on different platforms (e.g., web browsers) using different methods (e.g., CURL and Node.js) to avoid inconsistent results.

3) *Provide clear error messages and failure control.* As listed in Table 13, users may encounter various issues when invoking functions of web APIs. The issues of type ***function*** - *request failure with unknown reasons* are frequently reported in SO questions and experienced by the survey respondents. As expressed by some respondents, web API developers should provide clear error messages and failure control of web APIs. More specifically, web API developers should clearly describe the error messages of functions to help users understand and address the errors, such as the possible causes (e.g., authorization expiration or function deprecation), the related resources in the requests or results, and the possible solutions (e.g., refreshing the API key or using a new function). For unexpected errors (e.g., network problems), web API developers could provide failure control for users, e.g., allowing users to set a time interval to periodically request a function until obtain the results.

4) *Implement authorization using standard flows and enable automatic request.* We find that ***authorization*** - *error* and ***authorization*** - *expiration* are two of the top ten web API issue types that are frequently encountered by the respondents; and authorization errors are often difficult to address as they are reported in many SO questions. As specified by some respondents, web API developers should implement the authorization mechanisms using standard flows (e.g., OAuth) and allow the authorization (e.g., an API key) to be requested automatically, which could help reduce authorization issues.

5) *Create standard SDKs and libraries for major programming languages.* We find that 13% of the respondents expect that web APIs should provide standard SDKs and libraries for major programming languages, e.g., Java, Python, and JavaScript. Web API developers should better create such SDKs and libraries to facilitate the use of web APIs by users who prefer different programming languages. For independent web API developers or small companies with a few developers who are limited by the technical background (e.g., familiar programming languages) and budget, they may not be able to provide support for multiple programming languages. In such cases, web API developers could publish web APIs at publicly accessible registries (e.g., PW) and rely on other developers to create diverse SDKs and libraries for the web APIs.

6) *Make the changes and deprecations public and notify users about them.* We find that ***function*** - *change/deprecation* is one of the top ten web API issue types frequently experienced by the respondents. When updating web APIs, web API developers should make the changes and deprecations public and notify users about them. More importantly, as specified by some respondents, web API developers should better avoid breaking changes, e.g., no change of the result formats. A new version number should also be used to label the updated web API with significant changes, such that users can easily know the current state of a web API.

7) *Put effort on monitoring.* As expected by two respondents, web API developers should put effort on monitoring their web APIs after publishing the web APIs. From the monitoring records (e.g., execution logs), web API developers could design machine learning algorithms to effectively discover functional and non-functional issues of the web APIs and address the issues before users report them.

8) *Provide instant issue support.* Users may encounter various issues during the use of web APIs, e.g., those shown in Table 13. As expected by 8% of our survey respondents, web API developers should provide instant issue support that allow users to report encountered issues and help users solve issues rapidly. Web API developers should also pay close attention to the discussions of their web APIs in SO and the registries where the web APIs are published, in order to proactively discover and address the issues of web APIs.

9) *Allow easy trial.* We find that *easy-to-test endpoints* is an important feature of web APIs that is cared about by 38% of the survey respondents when shortlisting a web API for testing. Seven respondents express expectations on *easy trial* of web APIs. According to these findings, web API developers should provide easy trial for users to test the functions and endpoints of their web APIs, such that users can quickly check whether a web API could be used to fulfil a requirement.

10) *Charge properly.* We find that *free trial* and *no charge for use* are two important features of web APIs that are cared about by 36% of the survey respondents. Five respondents expect that the pricing of web APIs should be cheap and reasonable. It is good to also keep the charges flexible, e.g., allowing users to pay only for their needed functions.

4.2 Implications for Web API Registry Managers

11) *Collect and present the features of web APIs that are often considered by users when shortlisting a web API for testing.* We find 14 important features of web APIs (Table 10) when users decide to shortlist a web API for testing. Web API registry managers should better collect such features of web APIs and present them to users. 11 features can be specified by web API providers, including the functional description,

API provider, documentation URL, pricing, support (e.g., SDKs and libraries) for major programming languages, request/response formats, license, forum URL (if exist), specification types (e.g., REST and GraphQL), and versions. The other three features, i.e., popularity, easy-to-test endpoints, and performance, could be collected or provided as follows. The popularity could be collected by recording the number of followers and the average score rated by users. A testing platform could be developed for users to test endpoints of web APIs. The performance (e.g., response time) could be monitored by periodly calling the functions of web APIs.

12) *Provide a communication channel for users and web API providers.* We find that 83.0% of the web APIs from PW do not provide a forum for users to communicate with the web API providers. Although users can report web API issues by posting questions in Q&A communities such as SO, it should be better for web API registry managers to provide a communication channel for users and web API providers to discuss and solve issues.

5 Discussions

5.1 Can Language Models Help Retrieve More SO Questions Relevant to Web APIs?

In this work, we propose a method for retrieving SO questions related to web APIs based on web API names and a set of web API related keywords/phrases. The method achieves an accuracy of 92.4%, but suffers from a poor recall. As explained in Section 2.1.2, the method is suitable for our study. Recently, some language models, e.g., word2vec [69] and word Inverse Documentation Frequency (IDF) [70], have demonstrated the effectiveness of retrieving similar documents for a query [71], [72], [73]. We further examine whether the language models could improve the recall of our method by retrieving more SO questions for web APIs.

We implement a language model based question retrieval method that combines the word2vec and word IDF models according to the details described by Huang et al. [72]. To build the two language models, we collect a dataset of 599,402 SO questions that contain the 299,701 questions retrieved using our method and another 299,701 questions randomly selected from the set of more than 13.6 million questions retrieved using web API names. We process the collected questions by performing tokenization, stemming, and stop word removal using the NLTK [74] toolkit. We then train the word2vec model using the word2vec algorithm (with default parameter settings) in the Gensim [75] toolkit, and compute the IDF metrics of all words.

We randomly select 20 SO questions relevant to web APIs from the 384 questions sampled for evaluating the accuracy of our method. We retrieve the top 200 similar questions (that are not included in the 299,701 questions retrieved using our method) for each selected question using the language model based method. We then evaluate two different strategies to select similar questions as follows.

1) We evaluate the top 200 questions with the maximum similarities from the 4,000 ($= 20 \times 200$) similar questions retrieved for the 20 questions. We find that the questions are all retrieved for a question related to web API *linkedin*. The result shows that only 70 (i.e., 35%) of the 200 questions are relevant to web APIs.
2) We evaluate the top ten similar questions retrieved for each of the 20 questions. 141 (i.e., 71.9%) of the 196 questions (except four duplicate questions) are relevant to web APIs, which is much better than the first evaluation result. However, we find that the language model based method could not achieve good performance for every input question. Among the 20 selected questions, for the questions related to some web APIs, e.g., *google maps*, the top ten questions are all relevant; while for the questions related to some other web APIs, e.g., *mailchimp*, most of the top ten questions are irrelevant.

Based on the results from both strategies, it is promising to retrieve more SO questions relevant to web APIs using a language model based method. However, the two strategies could collect similar questions irrelevant to web APIs. To use the language model based method, we need to prepare a number of questions relevant to web APIs, e.g., the 20 questions used in the above experiment. The questions relevant to a web API will be used as the input of the language model based method to retrieve other questions that may be relevant to the web API. As described in Section 2.1.2, our method can help find questions relevant to a considerable number of web APIs. However, it still requires a great amount of effort to search for questions relevant to the web APIs that have no question retrieved using our method. We will further investigate the application of language models in retrieving SO questions relevant to web APIs in future work.

5.2 Why Do We Present the Preliminary Analysis Results of Web API Usage?

In Section 2.1.3, we perform a preliminary analysis of the usage of 20,047 web APIs from PW and APIs.guru. The results show that 18.1%-33.5% of the web APIs have been used. Our results are obtained by analyzing SO questions that discuss the web APIs and the mashups from PW that directly invoke the web APIs. The results may not reflect all the possible usage of web APIs in source code. We present the results because they could be beneficial for web API users, registry managers, and researchers on several tasks:

1) Our results help web API users validate the conventional wisdom that most of the web APIs published on the internet may not have been used.
2) Our results could help the registry managers of PW and APIs.guru obtain a rough view of the used web APIs published at their platforms. Based on the results, the registry managers could optimize the resource allocation for the management of web APIs. For example, more resources could be allocated to popularize and maintain the web APIs with higher frequencies of use.
3) Our results could help web API researchers develop better web API retrieval or recommendation approaches. For example, web APIs with higher frequencies of use could be ranked higher in the recommendation list.
4) We understand the usage of web APIs by leveraging the SO and mashups. Web API researchers could further investigate the web API usage using other data sources, e.g., the request code of web APIs in applications. They could then compare the results with our results to examine whether there are significant differences between the two kinds of results.

TABLE 21
Classification of our findings.

Findings	Unique to Web APIs	Not Unique to Web APIs
Web API Features	F5-F10, F12, F13	F1-F4, F11, F14
Web API Issue Types	IT1-IT5, IT8-IT10, IT22-IT26	IT6, IT7, IT11-IT21
Categories of Expectations on Web APIs	EC1, EC5, EC6	EC2-EC4, EC7-EC11
Implications for Web API Developers and Registry Managers	2)-5), 7), 11)	1), 6), 8)-10), 12)

5.3 Are Our Findings Unique to Web APIs?

We obtain four sets of findings, including 14 web API features, 26 web API issue types, 11 categories of user expectations on web APIs, and 12 implications for web API developers and registry managers. It is worth mentioning that some findings are unique to web APIs, while the other findings are applicable to both web APIs and non-web APIs. Whether a finding is unique to web APIs can be judged based on the aspect to which the finding is related. If a finding relates to an aspect common to any APIs, e.g., documentation or popularity, then the finding is applicable to any APIs. If a finding relates to an aspect specific to web APIs, e.g., endpoint or authorization, then the finding is unique to web APIs. We divide the findings into two categories, i.e., unique to web APIs and not unique to web APIs, as listed in Table 21.

5.4 Threats to Validity

Internal validity relates to the bias and errors in our manual analysis. In this study, we perform several manual processes. To retrieve SO questions related to web APIs, we create 26 web API related keywords/phrases (i.e., *WAKP*) from a ranked list of word sequences extracted from the SO questions that contain five well-known web APIs. We identify 24 web API issue types from 1,885 SO questions. For the survey design, we summarize 11 features of web APIs from the comments given by 20 developers. We extract three new features of web APIs, two new web API issue types, and 11 categories of user expectations on web APIs from the free-form text descriptions specified by the survey respondents. To avoid potential subjective bias from a single person, we carefully perform each of the manual tasks with the first and the fourth authors of this paper. The authors first independently build their own results (e.g., *WAKP*, features, issue types, and expectations) from the corresponding data sources, and then discuss the disagreements to reach a common decision.

For the analysis of the usage of web APIs, a threat is that there are problems with the collected web APIs, e.g., some web APIs may not be real web APIs. As described in Section 2.1.1, we address four kinds of problems. For example, we filter out the possible non-web APIs with unspecified or special architectural styles (e.g., 'Native/Browser') and filter out the web APIs that may be created for testing purposes. The steps contribute to retrieving better SO questions related to the web APIs. However, there might be other problems unsolved in this study, which may impact the results.

For the identification of web API issue types from SO questions, we randomly select a statistical sampling of questions from each of the five ranges (i.e., within the range from 360 to 383 for each sample) listed in Table 7. According to the *stratified sampling* method [76], the number of samples selected from a range should be proportional to the number of questions in the range. From this perspective, in our study, the numbers of questions sampled from the ranges are not proportional, which may lead to bias in the identified issue types towards the web APIs with a small number of relevant questions. Moreover, our work may not provide a complete list of web API features that are important to users when they shortlist web APIs for testing. In addition to the 14 features of web APIs listed in Table 10, users may consider the features of competing web APIs with similar functionality when shortlisting a web API.

It is also possible that there might be bias and errors in the survey responses. We make several kinds of effort to alleviate this threat. First, we include an 'I don't know' option in the survey question that asks respondents to confirm whether they have encountered each of the web API issue types identified from SO questions. This option could help reduce the noise data caused by respondents' poor understanding of some issue types. Second, to avoid dishonest answers (e.g., saying what we want to hear or saying what they want us to hear) from respondents, we explicitly state in the survey invitation email that our survey is anonymous and no personal information would be disseminated in the paper. As found by Ong et al. [77], anonymity and confidentiality can help obtain un-biased answers from survey respondents. Third, we conduct a pilot study with five developers to verify the survey questions from three aspects: length, clarity, and anchoring bias. We refine the survey based on the developers' feedback. Fourth, following the advice given by Kitchenham and Pfleeger [45] that proper language medium should be used for intended survey respondents, we translate our survey into Chinese to ensure that respondents from China could understand our survey well. However, there might be two another threats related to the user survey. When recruiting industry professionals, we send the survey to our contacts working in different companies from China and the United States as the contacts in the two countries respond to us. To recruit more industry professionals from other countries, we ask the respondents to help us distribute the survey to their associates and colleagues. The industry professionals recruited from our contacts may have the potential for making the survey results biased. To minimize the potential bias, we only ask our contacts to help us distribute the survey. We do not have any other connections with the rest of the participants. During the answering of the survey, after confirming the 24 web API issue types from SO questions, respondents might be tired of listing any other issues.

External validity relates to the generalizability of our results. We identify web API issue types by analyzing a sampled set of web API related SO questions. Due to the extremely imbalanced numbers of questions raised for web APIs, we divide the web APIs into five ranges by the number of questions and

then randomly sample a relatively large set of 1,885 questions. The sampled questions involve 837 web APIs that contain enough numbers of both frequently used and less frequently used web APIs. Therefore, the 1,885 questions should be suitable for identifying web API issue types. Furthermore, we ask the survey respondents to specify other web API issues that cannot be covered by the issue types discovered from SO questions. By using these strategies, the final set of 26 web API issue types provide a comprehensive view of the issues that have been encountered by users.

The response rate of our survey is 9.6%, which is similar to those of the surveys conducted in recent work [46], [51]. This indicates that our survey does not suffer from a low response rate, and the survey results should be reliable. Our 191 survey respondents are from 35 countries. They are working for different companies (e.g., Microsoft, Google, Alibaba, and Baidu) or have reported issues to web API projects at GitHub. Moreover, the respondents have different numbers of used web APIs and different years of experience using web APIs. The diversity of respondents can help improve the generalizability of our findings obtained from the survey responses.

Construct validity relates to the suitability of the data sources used for the analysis of web API usage and the translation of the English survey into Chinese. We analyze the usage of our collected web APIs by leveraging two publicly accessible data sources: the discussions of web APIs in SO and the mashups that invoke web APIs from PW. Our results may not reflect all possible usage of the web APIs. For example, users may not report their encountered web API issues in the two data sources and may experience few issues when using high-quality web APIs.

We adopt two strategies to alleviate the threat that the translated Chinese survey may be inconsistent with the English version and may pose any information about the answers that we wish to obtain from the respondents. Before translating the survey from English to Chinese, we verify the length, clarity, and bias of the survey written in English using a pilot study with five developers. We refine the survey to address the developers' feedback. The first and the fourth authors of the paper then carefully translate the survey from English to Chinese while keeping the content consistent between the two versions of the survey (Section 2.3.2). The English survey verified by the pilot study and the consistency between the two versions of the survey could help avoid the bias of the Chinese survey. However, the assessment of the consistency between the Chinese and English versions of the survey could be influenced by the personality of the two authors. For example, one author might have a dominant personality and have a stronger influence on the decisions. Moreover, the two authors may potentially have some bias towards our work.

6 Related Work

In this section, we review the related work on web API studies, empirical studies using SO, and surveys of software practitioners.

6.1 Web API Studies

In the past two decades, the studies on web APIs mainly concentrate on proposing approaches to retrieve and recommend web APIs based on user requirements. Most of the existing service retrieval and recommendation approaches can be roughly categorized as keyword-based [4], [10], [12], [78], [79], [80], semantics-based [6], [7], [8], [11], [81], [82], [83], [84], [85], [86], [87], Quality of Services (QoS)-based [9], [15], [17], [88], [89], and network-based [13], [14], [18], [19], [23], [90], according to the descriptive models of web APIs and requirements and the algorithms for measuring similarities between web APIs and requirements.

Despite numerous studies on service retrieval and recommendation, there are a number of studies [20], [21], [26], [29], [58], [59], [60], [61], [63], [64], [65], [66], [67], [91], [92], [93], [94], [95], [96] conducted to investigate the usage of web APIs or the issues happened when using web APIs. For example, Wang et al. [20] summarized 21 change types (e.g., add/change/delete method and add/change/delete response format) of web API evolution by comparing the multi-version documentation of 11 web APIs. Venkatesh et al. [91] studied the concerns of developers when using web APIs. They collected the developers' discussions of 32 popular web APIs from API forums and SO, and then used the Latent Dirichlet Allocation (LDA) [97] model to mine topics from the discussions. Espinha et al. [26] investigated the impact of web API evolution on the client applications that integrate web APIs. By analyzing the commits of ten client projects at GitHub that use four popular web APIs, e.g., Google Maps and Twitter, they identified several issues about web API evolution (e.g., instability and breaking changes) and provided a list of recommendations for web API providers to ease the evolution task for client developers. Hosono et al. [21] performed an empirical study on the reliability of web API documentation by analyzing 67 endpoints of 14 web APIs. They identified four categories of mismatch between the documentation and endpoints: undocumented/dynamic/unreturned keys and type mismatched. Li et al. [58] presented an empirical study of cloud API issues in commercial cloud platforms based on 32 discussion forums. They built a catalogue of the failures and faults related to cloud APIs. Zhou et al. [65] analyzed the faults in industrial microservice systems as well as the debugging practices based on 22 typical cases. They summarized six fault types, e.g., service interaction fault and functional fault. Belkhir et al. [92] conducted a study on the practices of web APIs used in Android apps. They analyzed the specifications of Android REST clients in the literature and built a catalogue of seven practices (e.g., the use of third-party HTTP client).

Oumaziz et al. [93] performed an empirical study to understand how REST services are used in Android apps based on 15 popular services and 500 popular apps. An online survey was conducted to identify the best practices for Android developers. They discovered that Android developers prefer to use a dedicated service library offered by the service provider and identified several important features of service libraries, e.g., complete documentation and vocabulary consistency. Rodriguez et al. [94] investigated whether the principles and guidelines of the REST architectural style are well followed in practice. They analyzed the HTTP calls of REST APIs from more than 78GB of HTTP traffic collected by Italy's biggest mobile internet provider, Telecom Italia. Rapoport et al. [95] analyzed the web requests made by 20 Android apps and presented a tool called *Stringoid* to scan the string concatenation operations in apps. Neumann et al. [96] analyzed 26 technical features of 500 REST APIs, e.g., the degree

of compliance with REST architectural principles and the adherence to best practices (e.g., API versioning). Based on the analysis results and the findings of several technical trends (e.g., widespread JSON support), they provided guidelines for designing higher quality services. Cummaudo et al. [64] studied three popular computer vision services over an 11-month longitudinal experiment. They found that the services behave inconsistently over time, and the documentation contains inconsistent descriptions of the techniques.

The aforementioned works mainly focused on analyzing 1) the usage of web APIs in Android apps or based on HTTP requests or 2) some specific kinds of web API issues (e.g., API changes and documentation reliability) by leveraging a number of web APIs or industrial cases. Although the discussion topics of web APIs that are mined using LDA by Venkatesh et al. [91] could reflect several web API issues, the topics (i.e., a set of keywords) are difficult to indicate the concrete issues. There still lacks a study on the usage of the large number of web APIs available on the internet and a comprehensive view of the web API issues encountered by users. In this paper, we analyze the usage of 20,047 web APIs from PW and APIs.guru by leveraging the SO questions and the mashups from PW. Moreover, we manually identify 26 web API issue types from 1,885 SO questions and the responses of a user survey. As evaluated by the survey respondents, our identified issue types provide a great view of the common issues that happened when using web APIs.

6.2 Empirical Studies using SO

The huge amount of questions and answers accumulated in SO covers various aspects of software development. Many empirical studies [20], [37], [92], [98], [99], [100], [101], [102] have been performed by leveraging SO data. Wan et al. [37] analyzed the discussion topics of blockchain by applying a balanced LDA model to the SO questions related to blockchain. Li et al. [98] conducted an empirical study to determine the needs of developers and understand the challenges faced by the developers when performing software development tasks. Nashehi et al. [99] investigated the factors that make an effective code example through a qualitative analysis of SO posts. Wang et al. [100] analyzed the interactions among developers in SO based on the distributions of questioners and answerers. Vasilescu et al. [101] evaluated the presence of women in SO by comparing their levels and duration of engagement to the male counterparts. In this paper, we analyze the usage of web APIs by leveraging SO questions. We also identify the issue types of using web APIs from a sampled set of SO questions.

6.3 Surveys of Software Practitioners

Survey is a methodology that has been widely used by prior studies to obtain information from software practitioners, e.g., work habits [103], motivation [104], search behavior [105], and perceptions on technologies or tools [46], [47], [48], [51], [106]. Xia et al. [105] summarized 34 search tasks from the search queries collected from 60 developers. Then, they surveyed 235 software engineers to understand the frequency and difficulty of the tasks. Wan et al. [51] used a mixed qualitative and quantitative approach to explore practitioners' perceptions, expectations, and adoption challenges of defect prediction techniques. They collected hypotheses regarding defect prediction from literature and open-ended interviews. Then, a survey was conducted to investigate the hypotheses from practitioners. Zou et al. [46] investigated practitioners' perceptions on the current state of smart contract development and challenges ahead through interviews and a survey. Bao et al. [106] conducted a survey to investigate how developers participate in OSS projects and their opinions on the factors that affect developers being a long-term contributor.

We conduct a survey to investigate the features of web APIs that are important for users to decide whether to test a web API, validate the web API issue types identified from SO questions, and understand user expectations on web APIs.

7 Conclusion and Future Work

We conduct a large-scale empirical study of 20,047 web APIs collected from two popular registries, i.e., ProgrammableWeb (PW) and APIs.guru. We extract the questions from Stack Overflow (SO) that are relevant to the web APIs using a heuristic method. Based on the SO questions and the mashups composed by web APIs from PW, we perform a preliminary analysis of the usage of web APIs. We then manually identify 24 web API issue types by analyzing 1,885 SO questions. A user survey is finally conducted to investigate the web API features that users often consider when shortlisting web APIs for testing, validate the identified web API issue types, and understand users' expectations on the development and management of web APIs. From the survey responses, we obtain 14 important features of web APIs, a better understanding of web API issue types (including two additional issue types identified from the responses), and 11 categories of user expectations on web APIs. According to our findings, we provide guidelines for web API developers and managers to improve web APIs and promote the use of web APIs.

In the future, we plan to develop an automated method to identify web API issue types from SO questions or other discussions of web APIs. By applying the method to the entire set of SO questions related to web APIs, we could perform some in-depth analysis, e.g., the evolution of web API issue types over time. As discussed in Section 5.1, we will also investigate the use of language models in retrieving SO questions relevant to web APIs. Moreover, we will attempt to further investigate the usage of web APIs by collecting their request code and exploring the request code in GitHub projects. We can then examine whether there are differences between the web API usage results obtained by looking at the code and the results obtained in this work.

Acknowledgments

We are grateful for the participants in our study who answered our survey questions and provided insightful comments. This research/project is supported by the National Natural Science Foundation of China (No. 62032025), National Research Foundation, Singapore, under its Industry Alignment Fund - Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] M. Bano, D. Zowghi, N. Ikram, and M. Niazi, "What makes service oriented requirements engineering challenging? a qualitative study," *IET Software*, vol. 8, no. 4, pp. 154–160, 2013.

[2] Y. Hu, Q. Peng, and X. Hu, "A time-aware and data sparsity tolerant approach for web service recommendation," in *2014 IEEE International Conference on Web Services*. IEEE, 2014, pp. 33–40.

[3] W. Song and H.-A. Jacobsen, "Static and dynamic process change," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 215–231, 2016.

[4] Q. He, R. Zhou, X. Zhang, Y. Wang, D. Ye, F. Chen, J. C. Grundy, and Y. Yang, "Keyword search for building service-based systems," *IEEE Transactions on Software Engineering*, vol. 43, no. 7, pp. 658–674, 2016.

[5] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web apis on the world wide web," in *2010 eighth IEEE European Conference on Web Services*. IEEE, 2010, pp. 107–114.

[6] N. Zhang, J. Wang, K. He, Z. Li, and Y. Huang, "Mining and clustering service goals for restful service discovery," *Knowledge and Information Systems*, vol. 58, no. 3, pp. 669–700, 2019.

[7] N. Zhang, J. Wang, Y. Ma, K. He, Z. Li, and X. F. Liu, "Web service discovery based on goal-oriented query expansion," *Journal of Systems and Software*, vol. 142, pp. 73–91, 2018.

[8] Z. Li, K. He, J. Wang, and N. Zhang, "An on-demand services discovery approach based on topic clustering," *Journal of Internet Technology*, vol. 15, no. 4, pp. 543–555, 2014.

[9] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction," *IEEE Transactions on Software Engineering*, vol. 40, no. 2, pp. 192–215, 2014.

[10] S. Meng, W. Dou, X. Zhang, and J. Chen, "Kasr: a keyword-aware service recommendation method on mapreduce for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3221–3231, 2014.

[11] B. Xia, Y. Fan, W. Tan, K. Huang, J. Zhang, and C. Wu, "Category-aware api clustering and distributed recommendation for automatic mashup creation," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 674–687, 2014.

[12] A. Halevy, E. Nemes, X. Dong, J. Madhavan, and J. Zhang, "Similarity search for web services," in *Proceedings of the 30th VLDB Conference*, 2004, pp. 372–383.

[13] F. Xie, J. Wang, R. Xiong, N. Zhang, Y. Ma, and K. He, "An integrated service recommendation approach for service-based system development," *Expert Systems With Applications*, vol. 123, pp. 178–194, 2019.

[14] R. Xiong, J. Wang, N. Zhang, and Y. Ma, "Deep hybrid collaborative filtering for web service recommendation," *Expert Systems with Applications*, vol. 110, pp. 191–205, 2018.

[15] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2015.

[16] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th International Conference on Web Services (ICWS)*. IEEE, 2012, pp. 202–209.

[17] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "Qos prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Networks and Applications*, pp. 1–11, 2019.

[18] K. Huang, Y. Fan, and W. Tan, "Recommendation in an evolving service ecosystem based on network prediction," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 906–920, 2014.

[19] T. Liang, L. Chen, J. Wu, H. Dong, and A. Bouguettaya, "Meta-path based service recommendation in heterogeneous information networks," in *International Conference on Service-Oriented Computing*. Springer, 2016, pp. 371–386.

[20] S. Wang, I. Keivanloo, and Y. Zou, "How do developers react to restful api evolution?" in *International Conference on Service-Oriented Computing*. Springer, 2014, pp. 245–259.

[21] M. Hosono, H. Washizaki, Y. Fukazawa, and K. Honda, "An empirical study on the reliability of the web api document," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018, pp. 715–716.

[22] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[23] B. Bai, Y. Fan, W. Tan, and J. Zhang, "Dltsr: A deep learning framework for recommendation of long-tail web services," *IEEE Transactions on Services Computing*, 2017.

[24] J. Wang, N. Zhang, C. Zeng, Z. Li, and K. He, "Towards services discovery based on service goal extraction and recommendation," in *2013 IEEE International Conference on Services Computing*. IEEE, 2013, pp. 65–72.

[25] N. Zhang, J. Wang, and Y. Ma, "Mining domain knowledge on service goals from textual service descriptions," *IEEE Transactions on Services Computing*, 2017.

[26] T. Espinha, A. Zaidman, and H.-G. Gross, "Web api growing pains: Stories from client developers and their code," in *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, 2014, pp. 84–93.

[27] U. Zdun, E. Wittern, and P. Leitner, "Emerging trends, challenges, and experiences in devops and microservice apis," *IEEE Software*, vol. 37, no. 1, pp. 87–91, 2019.

[28] E. Wittern, A. Cha, and J. A. Laredo, "Generating graphql-wrappers for rest (-like) apis," in *International Conference on Web Engineering*. Springer, 2018, pp. 65–83.

[29] J. Yasmin, Y. Tian, and J. Yang, "A first look at the deprecation of restful apis: An empirical study," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 151–161.

[30] Swagger, "What is openapi?" 2020. [Online]. Available: https://swagger.io/docs/specification/about/

[31] GraphQL, 2020. [Online]. Available: https://graphql.org

[32] REST, 2020. [Online]. Available: https://restfulapi.net

[33] P. A. Ly, C. Pedrinaci, and J. Domingue, "Automated information extraction from web apis documentation," in *International Conference on Web Information Systems Engineering*. Springer, 2012, pp. 497–511.

[34] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 93–96.

[35] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[36] T. Menzies, S. Majumder, N. Balaji, K. Brey, and W. Fu, "500+ times faster than deep learning:(a case study exploring faster methods for text mining stackoverflow)," in *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE, 2018, pp. 554–563.

[37] Z. Wan, X. Xia, and A. E. Hassan, "What is discussed about blockchain? a case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across the stack exchange communities," *IEEE Transactions on Software Engineering*, 2019.

[38] L. Azzopardi, Y. Moshfeghi, M. Halvey, R. S. Alkhawaldeh, K. Balog, E. Di Buccio, D. Ceccarelli, J. M. Fernández-Luna, C. Hull, J. Mannix *et al.*, "Lucene4ir: Developing information retrieval evaluation resources using lucene," in *ACM SIGIR Forum*, vol. 50, no. 2. ACM, 2017, pp. 58–75.

[39] F. Peng and D. Schuurmans, "Combining naive bayes and n-gram language models for text classification," in *European Conference on Information Retrieval*. Springer, 2003, pp. 335–350.

[40] SOAP, 2020. [Online]. Available: https://en.wikipedia.org/wiki/SOAP

[41] W. Jiang, D. Lee, and S. Hu, "Large-scale longitudinal analysis of soap-based and restful web services," in *2012 IEEE 19th International Conference on Web Services*. IEEE, 2012, pp. 218–225.

[42] D. Spencer, *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[43] Q. Huang, X. Xia, D. Lo, and G. C. Murphy, "Automating intention mining," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1098–1119, 2018.

[44] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological Bulletin*, vol. 76, no. 5, p. 378, 1971.

[45] B. A. Kitchenham and S. L. Pfleeger, "Personal opinion surveys," in *Guide to Advanced Empirical Software Engineering*. Springer, 2008, pp. 63–92.

[46] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, 2019.

[47] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?" *IEEE Transactions on Software Engineering*, 2019.

[48] W. Zou, D. Lo, Z. Chen, X. Xia, Y. Feng, and B. Xu, "How practitioners perceive automated bug report management techniques," *IEEE Transactions on Software Engineering*, vol. 46, no. 8, pp. 836–862, 2018.

[49] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 165–176.

[50] P. K. Tyagi, "The effects of appeals, anonymity, and feedback on mail survey response patterns from salespeople," *Journal of the Academy of Marketing Science*, vol. 17, no. 3, pp. 235–241, 1989.

[51] Z. Wan, X. Xia, A. E. Hassan, D. Lo, J. Yin, and X. Yang, "Perceptions, expectations, and challenges in defect prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 11, pp. 1241–1266, 2020.

[52] F. Konietschke, L. A. Hothorn, and E. Brunner, "Rank-based multiple test procedures and simultaneous confidence intervals," *Electronic Journal of Statistics*, vol. 6, pp. 738–759, 2012.

[53] F. Zampetti, G. Fucci, A. Serebrenik, and M. Di Penta, "Self-admitted technical debt practices: a comparison between industry and open-source," *Empirical Software Engineering*, vol. 26, no. 6, pp. 1–32, 2021.

[54] "Http authentication," 2020. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

[55] OAuth, 2020. [Online]. Available: https://oauth.net

[56] CURL, 2020. [Online]. Available: https://curl.haxx.se/docs/manpage.html

[57] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, "Gendermag: A method for evaluating software's gender inclusiveness," *Interacting with Computers*, vol. 28, no. 6, pp. 760–787, 2016.

[58] Z. Li, Q. Lu, L. Zhu, X. Xu, Y. Liu, and W. Zhang, "An empirical study of cloud api issues," *IEEE Cloud Computing*, vol. 5, no. 2, pp. 58–72, 2018.

[59] T. Espinha, A. Zaidman, and H.-G. Gross, "Web api fragility: How robust is your mobile application?" in *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*. IEEE, 2015, pp. 12–21.

[60] ——, "Web api growing pains: Loosely coupled yet strongly tied," *Journal of Systems and Software*, vol. 100, pp. 27–43, 2015.

[61] S. Sohan, C. Anslow, and F. Maurer, "A case study of web api evolution," in *2015 IEEE World Congress on Services*. IEEE, 2015, pp. 245–252.

[62] J. Li, Y. Xiong, X. Liu, and L. Zhang, "How does web service api evolution affect clients?" in *2013 IEEE 20th International Conference on Web Services*. IEEE, 2013, pp. 300–307.

[63] A. Mendoza and G. Gu, "Mobile application web api reconnaissance: Web-to-mobile inconsistencies & vulnerabilities," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 756–769.

[64] A. Cummaudo, R. Vasa, J. Grundy, M. Abdelrazek, and A. Cain, "Losing confidence in quality: Unspoken evolution of computer vision services," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 333–342.

[65] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, 2018.

[66] V. Atlidakis, P. Godefroid, and M. Polishchuk, "Restler: Stateful rest api fuzzing," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 748–758.

[67] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating web apis on the world wide web," in *2010 eighth ieee european conference on web services*. IEEE, 2010, pp. 107–114.

[68] OpenREST, 2020. [Online]. Available: https://sourceforge.net/projects/openrest

[69] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[70] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, pp. 1–37, 2008.

[71] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th international conference on software engineering*, 2016, pp. 404–415.

[72] Q. Huang, X. Xia, Z. Xing, D. Lo, and X. Wang, "Api method recommendation without worrying about the task-api knowledge gap," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 293–304.

[73] N. Zhang, Q. Huang, X. Xia, Y. Zou, D. Lo, and Z. Xing, "Chatbot4qr: Interactive query refinement for technical question retrieval," *IEEE Transactions on Software Engineering*, 2020.

[74] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[75] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.

[76] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," *arXiv preprint arXiv:2002.07764*, 2020.

[77] A. D. Ong and D. J. Weiss, "The impact of anonymity on responses to sensitive questions 1," *Journal of Applied Social Psychology*, vol. 30, no. 8, pp. 1691–1708, 2000.

[78] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1629–1642, 2009.

[79] F. Liu, Y. Shi, J. Yu, T. Wang, and J. Wu, "Measuring similarity of web services based on wsdl," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 155–162.

[80] F. Thung, R. J. Oentaryo, D. Lo, and Y. Tian, "Webapirec: Recommending web apis to software projects via personalized ranking," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 145–156, 2017.

[81] J. M. García, D. Ruiz, and A. Ruiz-Cortés, "Improving semantic web services discovery using sparql-based repository filtering," *Journal of Web Semantics*, vol. 17, pp. 12–24, 2012.

[82] M. Klusch, B. Fries, and K. Sycara, "Owls-mx: A hybrid semantic web service matchmaker for owl-s services," *Journal of Web Semantics*, vol. 7, no. 2, pp. 121–133, 2009.

[83] D. Roman, J. Kopeckỳ, T. Vitvar, J. Domingue, and D. Fensel, "Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis," *Journal of Web Semantics*, vol. 31, pp. 39–58, 2015.

[84] S. Subbulakshmi, K. Ramar, A. Shaji, and P. Prakash, "Web service recommendation based on semantic analysis of web service specification and enhanced collaborative filtering," in *The International Symposium on Intelligent Systems Technologies and Applications*. Springer, 2017, pp. 54–65.

[85] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2011.

[86] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic web service discovery and composition framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 537–550, 2015.

[87] F. Chen, C. Lu, H. Wu, and M. Li, "A semantic similarity measure integrating multiple conceptual relationships for web service discovery," *Expert Systems with Applications*, vol. 67, pp. 19–31, 2017.

[88] Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang, "Context-aware qos prediction for web service recommendation and selection," *Expert Systems with Applications*, vol. 53, pp. 75–86, 2016.

[89] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2012.

[90] W. Xu, J. Cao, L. Hu, J. Wang, and M. Li, "A social-aware service recommendation approach for mashup creation," in *2013 IEEE 20th International Conference on Web Services*. IEEE, 2013, pp. 107–114.

[91] P. K. Venkatesh, S. Wang, F. Zhang, Y. Zou, and A. E. Hassan, "What do client developers concern when using web apis? an empirical study on developer forums and stack overflow," in *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 2016, pp. 131–138.

[92] A. Belkhir, M. Abdellatif, R. Tighilt, N. Moha, Y.-G. Guéhéneuc, and É. Beaudry, "An observational study on the state of rest api uses in android mobile applications," in *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 2019, pp. 66–75.

[93] M. A. Oumaziz, A. Belkhir, T. Vacher, E. Beaudry, X. Blanc, J.-R. Falleri, and N. Moha, "Empirical study on rest apis usage in android mobile applications," in *International Conference on Service-Oriented Computing*. Springer, 2017, pp. 614–622.

[94] C. Rodríguez, M. Baez, F. Daniel, F. Casati, J. C. Trabucco, L. Canali, and G. Percannella, "Rest apis: a large-scale analysis of compliance with principles and best practices," in *International conference on web engineering*. Springer, 2016, pp. 21–39.

[95] M. Rapoport, P. Suter, E. Wittern, O. Lhóták, and J. Dolby, "Who you gonna call? analyzing web requests in android applications," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 80–90.

[96] A. Neumann, N. Laranjeiro, and J. Bernardino, "An analysis of public rest web service apis," *IEEE Transactions on Services Computing*, 2018.

[97] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[98] H. Li, Z. Xing, X. Peng, and W. Zhao, "What help do developers seek, when and how?" in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 142–151.

[99] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming q&a in stackoverflow," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 25–34.

[100] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in stackoverflow," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1019–1024.

[101] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *2012 International Conference on Social Informatics*. IEEE, 2012, pp. 332–338.

[102] M. Soliman, M. Galster, A. R. Salama, and M. Riebisch, "Architectural knowledge for technology decisions in developer communities: An exploratory study with stackoverflow," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. IEEE, 2016, pp. 128–133.

[103] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: a study of developer work habits," in *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, 2006, pp. 492–501.

[104] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel," *Research Policy*, vol. 32, no. 7, pp. 1159–1177, 2003.

[105] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 2017.

[106] L. Bao, X. Xia, D. Lo, and G. C. Murphy, "A large scale study of long-time contributor prediction for github projects," *IEEE Transactions on Software Engineering*, 2019.

TABLE 22
Profiles of the 20 developers involved in our user survey design and the five developers recruited for our pilot study. The seven developers with '–' in the last two columns are not available for the pilot study.

Developer Id	# Years of Experience Using Web APIs	Proficiency in English	Recruited for Pilot Study
D1	1	Poor	No
D2	2	Mediocre	No
D3	2	–	–
D4	2.5	Good	No
D5	2.5	Mediocre	No
D6	3	–	–
D7	3	Mediocre	No
D8	3	Mediocre	No
D9	3	–	–
D10	3	Good	No
D11	3.5	Good	Yes
D12	4	–	–
D13	4.5	–	–
D14	4.5	Mediocre	No
D15	5	Good	Yes
D16	5	Good	Yes
D17	5.5	–	–
D18	6	Good	Yes
D19	6.5	–	–
D20	7	Very Good	Yes

Appendix

A. Pilot Study for Verifying the User Survey

In this appendix, we describe the detailed steps of the pilot study conducted to verify the design of our user survey.

Step 1: Developer recruitment. The first step of the pilot study is to recruit several developers who have experience in using web APIs and are proficient in English, such that the developers can understand and verify the survey questions about web APIs in English. As described in Section 2.3.1, we ask 20 developers with 1-7 years of experience using web APIs (Table 22) for the top three most important features of web APIs when shortlisting a web API for testing. The developers are recruited from two IT companies: IGS and Hengtian. We contacted the 20 developers again via email. In the emails, we introduce our user survey and the purpose of our pilot study and ask for the following information:

- *Availability to participate in the pilot study?* Yes / No
- *Proficiency in English*: Very Good / Good / Mediocre / Poor / Very Poor

We receive responses from the 20 developers, and 13 developers are available to participate in the pilot study. From the 13 developers, we select five developers with more than three years of experience using web APIs and very good or good proficiency in English. The profiles of the five developers are listed in Table 22.

Step 2: Survey verification. We send our initial survey in English to the five developers by email and ask them to verify the survey questions from three aspects: 1) *length*: are there any survey questions that are too lengthy to read or understand? 2) *clarity*: are there any survey questions with unclear expressions? and 3) *bias*: are there any survey questions that may imply the expected answers that we wish to obtain from the respondents? We give the developers one week to check the survey and encourage them to comment on the three aspects.

One week later, we received the developers' feedback. They report that there is no problem with the length of the survey questions, and the survey questions have no bias. However, the developers provide comments on the clarity of four survey questions (except the three survey questions relevant to the demographics and the survey question that asks for other web API issues outside the 24 issue types identified from SO questions). For example, Table 23 presents an ambiguous survey question that asks for the top three important features of web APIs and the comments given by the developers.

Step 3: Survey refinement. We refine the ambiguous descriptions of the survey questions according to the developers' feedback. The last column of Table 23 presents the refined survey question that clarifies the ambiguous question presented in the first column.

Step 4: Refined survey verification. We send the refined survey to the five developers and ask them to verify the survey again. The developers report that the refined survey has no more problems.

After completing the steps of the pilot study, we obtain the final version of our survey in English.

TABLE 23
Example of an ambiguous survey question, the comments given by the five developers of the pilot study, and the refined survey question.

Ambiguous Survey Question	Comments Given by the Developers	Refined Survey Question
When you need to choose some candidate web APIs for testing, what are the top 3 most important features (i.e., characteristics) of a web API that make you decide to test it? ☐ It has a functional summary similar to your requirements ☐ It has a well-known web API provider (e.g., Google) ☐ It has a relatively high popularity (e.g., #followers and popularity score) ☐ It has a well-organized documentation ☐ It has easy-to-test endpoints ☐ It has a free trial ☐ It has no charge for use ☐ It has a support (e.g., a web API client wrapper or sample code) for your familiar programming languages ☐ It follows standard request/response formats (e.g., JSON) ☐ It has a compatible license with your existing projects ☐ It has a web API forum alive Other: _____	D11: 'It has a' is unnecessary and '#' is unclear. D15: The expressions of web API features can be simplified, e.g., 'It has a free trial' -> 'free trail'. D16: No guidance for the last input field 'Other:' D18: The symbol '#' may not be understood by every person. What do you mean about an 'alive' forum? D20: remove 'it has a' and 'your' in the features; replace 'other:' by 'other (please specify):'	When you need to choose some candidate web APIs for testing, what are the top 3 most important features (i.e., characteristics) of a web API that make you decide to test it? ☐ Similar functional description to the requirement ☐ Well-known API provider (e.g., Google) ☐ Relatively high popularity (e.g., the number of followers and popularity score) ☐ Well-organized documentation ☐ Easy-to-test endpoints ☐ Free trial ☐ No charge for use ☐ Support (e.g., API client wrappers or sample codes) for familiar programming languages ☐ Standard request/response formats (e.g., JSON) ☐ Compatible license with the existing projects ☐ Accessible API forum Other (please specify): _____