Usage of Unsupported Technologies in Websites Worldwide

Pascal Alfadian Nugroho[1], Raymond Chandra Putra[1], Rajasa Cikal Maulana[2], Vinson Tandra[1]
[1]Universitas Katolik Parahyangan. Jl. Ciumbuleuit No. 94. Bandung 40252. Indonesia
[2]PT Bank Rakyat Indonesia. Jl. Jend. Sudirman No.14. DKI Jakarta 10210. Indonesia

ARTICLE INFO

Article history:

Received April 16, 2024
Revised June 27, 2024
Published July 08, 2024

Keywords:

Google BigQuery;
HTTP Archive;
Measurement;
Software Dependencies;
Software Maintenance;
Website;
Website Security

ABSTRACT

Websites using unsupported 3rd party technologies (libraries, frameworks, plugins, etc) are generally not recommended, especially due to security issues that are left unfixed. However, upgrading to supported technologies is also challenging, hence not all web maintainers upgrade their technology dependencies. Measuring the existence of unsupported technologies in the wild may contribute to the sense of urgency in keeping technologies updated. Our research proposed a method to measure the existence of unsupported technologies in international websites, using HTTP Archive as the data source. The contribution from our research is the method as well as the snapshot result from January 2023 data. The method is composed of four steps, namely: identify the list of websites, identify technologies used, group by technology names and retrieve currently supported versions, and compare versions between usage and supported versions. From the January 2023 data, we found several interesting results. One is that the higher the website rank is, the higher the number of supported technologies used. Another finding was that worldwide websites also generally use more supported versions of technologies, compared to Indonesian websites. Further research may be performed for longitudinal analysis of technology support evolution.

Corresponding Author:

Pascal Alfadian Nugroho. Universitas Katolik Parahyangan. Jl. Ciumbuleuit No. 94. Bandung 40252. Indonesia
Email: pascal@unpar.ac.id

1. INTRODUCTION

Most modern websites in service today are built and depend on various underlying technologies, including libraries, frameworks, plugins, or external services. These technologies are often maintained by third parties, i.e., not by the same organization that made and maintains the website in question. For example, https://unpar.ac.id, a website owned by Universitas Katolik Parahyangan, is built on top of PHP (maintained by The PHP Group), WordPress (by WordPress Foundation), MySQL (by Oracle), among other technologies.

In the meantime, more websites are expected to run reliably 24/7, as people's lives are more reliant on them [1]. It changed the way people work [2], study [3], [4], [5], [6], shop [7], as well as changing how governments work [8], [9]. On the other side, cyber-attack frequency is also increasing and has become a lucrative business by itself [10], [11]. Therefore, it is important to ensure that all components or technologies serving as building blocks for the website are properly maintained. The terminology "full stack developer" was then coined to refer to someone who can work with those various technologies to make a website run [12], [13]. A full-stack developer who maintains a website does not only need to understand and maintain the code that he/she built but also the various third-party technologies made and maintained by people from outside his/her organization. For those third-party technologies, ensuring reliability means ensuring that the technologies used are still supported by the respective maintainers. For example, in January 2023, the PHP Group supported PHP versions 8.1 and 8.2. If a certain website is built on top of, for example, PHP version 7.4, the quality of such technology is no longer guaranteed by PHP Group, despite being able to work. From the maintainer side (e.g. The PHP Group), it is impossible to support all versions of their product, especially the older ones, since supporting software requires resources. This is even more challenging for open-source projects, that generally do not directly charge their users for maintenance costs [14], [15], [16], [17], [18].

In this case, "support" means making new improvements and more importantly fixing bugs (including security-related bugs). In the current situation, software can inevitably be released and left as it is without such kind of support. This is because the environment where such software is running is dynamically changing, so the software needs to adapt to those changes. Without support in the form of fixes, the software will degrade, and the involved degradation of quality may be related to security lapses and poses a risk to users of such website [19]. In fact, "Vulnerable and Outdated Components" was in the 6th position in OWASP Top 10 2021, up from 9th position in OWASP 2017 [20].

Therefore, ideally, a web maintainer needs to keep the websites' supporting technologies up to date by regularly upgrading them to the supported versions. For example, a website using PHP version 7.4 should be upgraded to at least version 8.1, to ensure that at least the PHP part of the website is still receiving security fixes. There are challenges, though, to keep these versions updated. Unfortunately, updating technologies brings not only new features and fixes but sometimes also problems [21], [22], [23], [24], [25], [26]. For example, in PHP 8.1 implicit conversion from floating point to integer is no longer valid, hence if such code exists on the website, they need to be changed. Therefore, not all web maintainers keep the technologies supporting his/her website updated to the supported versions. There are several ways proposed to promote updates, such as a silent update mechanism [27] though, but at the expense of unstable running of the software if not taken care of properly.

In this research, we would like to know how many of the world's websites are using technologies supported (and unsupported) by the respective maintainers. This research is a continuation of our previous research [28] that answered the same question but for websites in a particular country, i.e. Indonesia.

Measurement and analysis of the usage of unsupported or outdated technologies have been performed in the past several researches. [29] analyzed 5.6 million websites from HTTP Archive and NVD (National Vulnerability Database) over 18 months to see whether websites are using outdated software, their update behavior, and the impact on security. In that research, "outdated" is defined as using a certain version of software whereas a newer minor or patch version is available from the respective maintainer. For example, using PHP version 7.3 is considered outdated if PHP version 7.4 is already available, even if version 7.3 is still supported. However, using PHP 7.4 is not considered outdated when PHP 8.0 is available, because the major version is different. They also statistically measured if website owners updated their software dependencies over the 18 months. Finally, the versions used are mapped to the NVD database to see the severity of using outdated technologies. The researchers found that among 8.205.923 origins (URLs), 246 different technologies were found and 148 of them have exploitable vulnerabilities. Moreover, [23] measured the "technical lag", i.e. the difference of major/minor/micro versions between the currently used and the supported version as well as the number of days that elapsed since the last release. [30] studied usage of unsupported (more formally as "deprecated") specifically in the field of JavaScript language ecosystem, npm.

Similarly, [31] analyzed 157.2 million webpages from Alexa's Top 1M websites for four years, focusing on the usage of JavaScript libraries and Macromedia Flash. They use the term "vulnerable" for versions that have security vulnerabilities registered in the CVE (Common Vulnerabilities and Exposures) database but are still used. They found that an average of 531.2 days with 25.336 websites of the window of vulnerability; as well as the fact that 13 out of 27 CVE reports have incorrect vulnerable version information, which may mislead security-related tasks such as security updates.

Similar studies [32], [33], [34] looked at websites reliance on more critical third-party services, namely the DNS (Domain Name Service), CDN (Content Delivery Network), and CA (Certificate Authority). Among the findings are: that 89% of the top-100K websites were critically dependent on third-party DNS, CDN, or CA providers and in Africa, the dependencies were largely (92%) concentrated to three providers. However, the services mentioned in those studies are different in terms of criticality, when compared to our study. DNS, CDN, and CA services are required in either real-time or regularly. If a visitor is trying to visit https://unpar.ac.id while the DNS service is not available, the whole website is inaccessible because the visitor is unable to find the correct IP address for unpar.ac.id. If such a website merely uses an unsupported version of PHP, visitors can still visit the same website because the copy of PHP interpreter exists on that server.

Other studies [32], [35], [36] focused on the chains of dependencies. A website's certain dependency may depend on yet another service. [35] found that a single third-party dependency can lead up to eight subsequent requests to another service. They also found that 93% of analyzed websites are embedded by third parties that are in regions that might not be in line with the current legal framework. The implication of this is that one of those services may be disrupted or blocked by the local government, making the main website unusable. An example of this problem is the blocking of https://vimeo.com in Indonesia, while it is used in some websites like https://laracasts.com/.

The lack of support for website dependencies is not always bad. Another research [37] argued that 20% of dependencies affected by security vulnerabilities are not deployed and hence pose no significant danger. Those dependencies were mostly used in build time. Our research detects what is directly accessible from the internet and already filters out technologies that are not deployed.

Smaller-scale website security researches were also performed by earlier research. [38] studied two specific websites in East Java for their security vulnerabilities, exploring several tools like OWASP ZAP, Wapiti, and Nikto. OWASP ZAP was also used in another research [39], as well as Nessus [40], [41]. The security model was also evaluated in another research [42].

Finally, our research in the past [28] measured the unsupported technologies used in 1.439 websites in Indonesia. In general, there are several differences between our previous research compared to the current one:

1. Previous research looked at websites from Indonesia, while the current one looks at all websites in the world.
2. Previous research took URLs from a “top list” (Alexa), while the current one takes data from “existing sources” (HTTP Archive). The benefit of using existing sources, as described in [43], is it can be easily reproducible since all required information was already stored and we simply had to query them, instead of recrawling every time we need the data.
3. Detection of technologies in previous research was done by using the Wappalyzer tool, automated and run from researchers’ computers. Detection of technologies in current research is done by the HTTP Archive team, and we simply queried the technologies used for each website from their database.

The contribution from our research is insights into the state of technologies supported in websites worldwide as of January 2023. We also proposed a method for supportability measurement that can be replicated for different timelines, as long as the raw data is available in HTTP Archive. These two contributions complement the aforementioned research that focused on different areas of website technology supportability. Subsequent sections will be structured as follows. Section 2 (methods) will explain the four steps to acquire and analyze the data mainly using the Google BigQuery technology, as well as some limitations to our method. Section 3 (results and discussion) discusses the results of the analysis. Finally, Section 4 (conclusion) concludes the findings and gives recommendations based on the result.

2. METHODS

The research method is composed of four steps and will be explained in the following subsections. Note that we performed these steps in January 2023.

2.1. Identifying list of websites

We collected our data from the HTTP Archive database, whose data is collected from the Chrome User Experience Report. When a user uses Google Chrome to browse the web and opts into a certain program, Google will anonymously record the website URLs visited and send it to the database. Every 1st day of each month, those URLs are processed by being visited with both Chrome desktop and mobile browsers in the HTTP Archive’s private instances, mostly located in the United States. Based on these visits, extra data is then attached to each website URL. These data include website performance data, technologies used, and other information. Finally, the collected data were then stored in Google’s BigQuery database, which is accessible through the Google Cloud console. BigQuery itself is a technology from Google that allows storage and analysis of data using common SQL syntax, while the Google Cloud console is the interface to interact with various Google Cloud products, including BigQuery.

HTTP Archive data has been used in many research, ranging from measuring usage of newer HTTP protocol versions in the wild [44], conformance of server implementations to the standard [45], as well as the usage of tracking mechanisms [46].

For this research, we use the dataset httparchive.summary_pages.2023_01_01_desktop and httparchive.summary_pages.2023_01_01_mobile, by executing the following query in the console:

```
SELECT url. AVG(rank) AS rank FROM `httparchive.summary_pages.2023_01_01_*` GROUP BY url ORDER by
rank;
```

From such query, we received a result of 16.852.173 URLs, with the first 10 URLs shown in Table 1. While in previous research we could get the exact value of rank (determines the popularity of such a website), HTTP Archive only provides relatively vague rank values, either 1.000, 5.000, 10.000, 50.000, or the 10-multiplies of them.

In previous research, we stored the retrieved URLs locally for further processing. In this current research, we don't need to store the result locally because subsequent steps will again query directly from the HTTP Archive database.

Table 1. First 10 results of identified websites

rank	url
1.000	https://www.pa---nx.com/
1.000	https://www.gov.br/
1.000	https://tecno.servicewebly.com/
1.000	https://bbs.animanch.com/
1.000	https://www.google.co.th/
1.000	https://igram.io/
1.000	https://clever.com/
1.000	https://veja.abril.com.br/
1.000	https://m.gsmarena.com/
1.000	https://www.amazon.co.jp/

2.2. Identify technologies used

The next step is to identify technologies used on those websites. Instead of querying them one by one using the Wappalyzer tool as was done in previous research, we queried the HTTP Archive database. We use the following query to identify them:

```
SELECT url. ARRAY_AGG(STRUCT(app AS name. info AS version)) AS app FROM
`httparchive.technologies.2023_01_01_*` GROUP BY url;
```

With that query, we collected technologies used for each website and showed them in the form of structures. Table 2 shows the result of the first two rows, as an example. It can be seen from the table, that the https://www.samplecodeabap.com/ website uses at least six technologies: Endurance Page Cache, jQuery Migrate, PHP, WordPress, Yoast SEO, and Open Graph. It is also known that they are using jQuery Migrate version 3.3.2 and Yoast SEO version 19.13, but the versions of Endurance Page Cache, PHP, WordPress, and Open Graph are not known.

2.3. Group by technology names and retrieve the currently supported version

From Table 2, it is also implied that the technology "Endurance Page Cache" is used in at least one website. We record technologies that are used on at least one website and store them in one place. The query used to collect these technologies is as follows:

```
SELECT
  app as name.
  COUNT(app) AS num_sites.
  COUNTIF(info="") AS num_unversioned.
  COUNTIF(info!="") AS num_versioned.
  "" AS website.
  "" AS min_supported_version.
  "" AS min_supported_version_eol.
  "" AS supported_version_reference
FROM `httparchive.technologies.2023_01_01_*`
GROUP BY app ORDER BY num_versioned DESC;
```

Notice that the last four columns were filled simply with empty string. This is because we wanted to export the result into a Google Sheet and those columns were meant to be filled manually. The exported result can be seen in Fig. 1. Columns A, B, C, and D are automatically filled in by the SQL query, while the rest (E, F, G, H) were filled manually.

Column A "name" defines the name of the technology. Column B "num_sites" defines the number of websites that are detected using this technology. Column C "num_unversioned" defines the number of websites detected using the technology but the version is unknown. Finally, column D "num_versioned" defines the number of websites using the technology and the version is known. We only fill columns E, F, G, and H when the value of num_versioned is non-zero. This is because in the last step, we will check supportability based on the version number, and if no information on the version number exists then no comparison will be made.

Column E “website” is filled with the website URL of the technology, detected using a simple Google search. For example, the website for PHP is https://www.php.net. At the time of research was performed, the supported version of PHP was version 8.1 or above.

Column F “min_supported_version” is the PHP version supported by The PHP Foundation. We named column F “min_supported_version” because we originally thought that when a certain version number is supported, any versions above it would be supported as well. However, we found that it was not always the case. For example, the maintainer of Drupal supported version 7 and 9 or above, but not version 8. This is usually because some major version number is marked as “Long Term Support” and has a longer supportability period. The values of cells in this column are governed by rules as follows:

1. We put “?” if we could not find what versions are supported for this technology.
2. We put “>= X” if version X or greater is supported, “>= X.Y” and “>= X.Y.Z” applies similarly, but are separated as MAJOR/MINOR/PATCH versions as defined in the Semantic Versioning rule [24]. Other operators like “>” and “=” also apply similarly.
3. We put “,” to separate more than one rule.

We also fill column G “min_supported_version_eol” when we know when the current version will be last supported. For example, PHP 8.1 would be supported until 25 November 2023. Finally, in column “H” we put the URL where we found the information for the former columns.

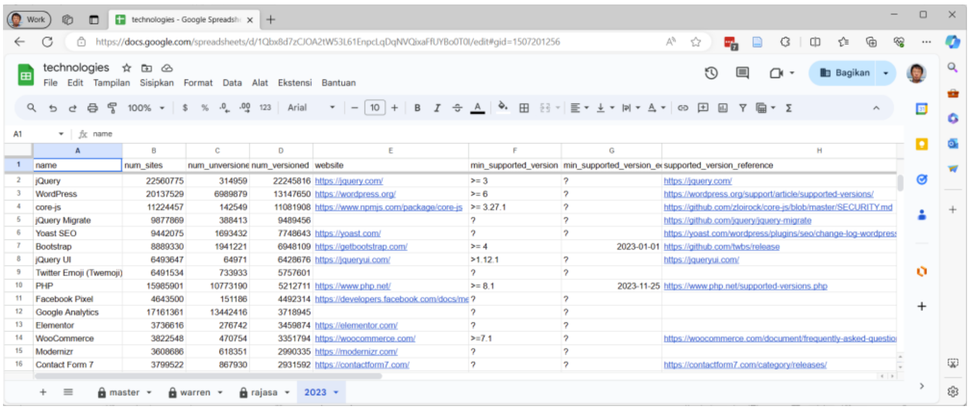

Fig. 1. Technology export result in Google Sheet

Table 2. First two rows of website technology query result

url	app_name	app_version
https://www.samplecodeabap.com/	Endurance Page Cache	
	jQuery Migrate	3.3.2
	...	
	PHP	
	WordPress	
	...	
	Yoast SEO	19.13
	Open Graph	
https://www.fi-lab.com/	PHP	
	jQuery	1.12.4
	...	
	Google Analytics	

2.4. Compare version between current usage and supported version

Finally, for each version detected, we decide whether such a version is supported or not. More precisely, the decision can be one of the following:

1. “Not-versioned” if there is no version data found on the currently detected technology.
2. “Non-conclusive” if it is not possible to determine whether the version found is still supported or not. An example of this case is when min_supported_version is “?”.
3. “Supported” if we are sure that the version is still supported.
4. “Unsupported” if the technology version is no longer supported.

The detection function is implemented in JavaScript using a user-defined function within Google BigQuery. Fig. 2 shows a simplified algorithm (does not change from our previous research [28], while the full source code is available on GitHub. Since we are measuring millions of websites, there are some edge cases that the function needs to handle, especially in the version numbering of technologies.

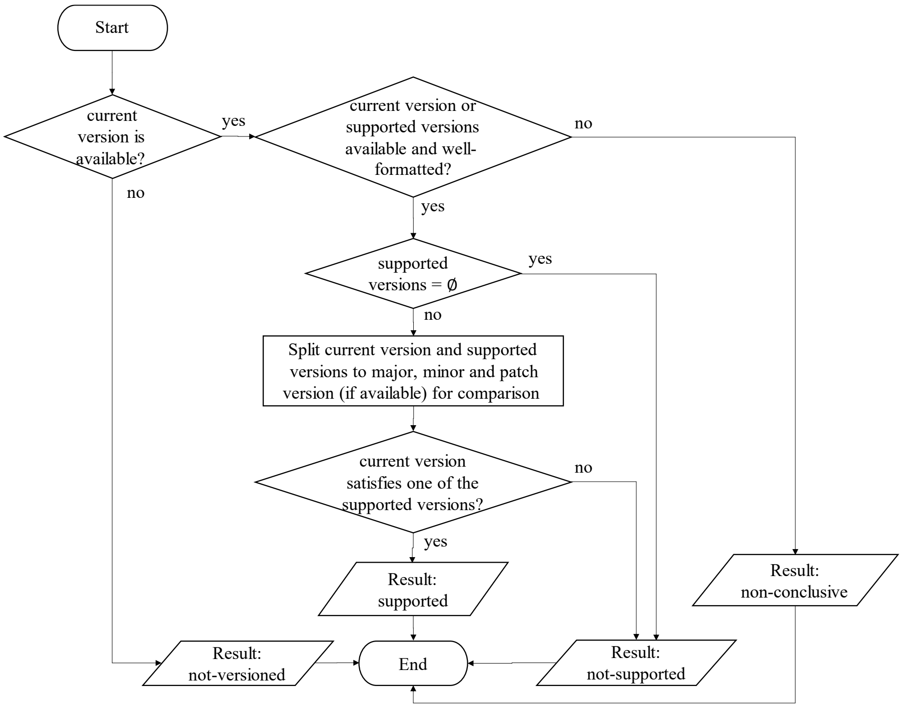

Fig. 2. Simplified algorithm of comparing version

2.5. Limitations

Before proceeding with the results and discussion, we would like to explain some limitations of our method. First, the data source of our research is the Chrome User Experience report, which means that the characteristics of users and the websites visited may be biased [47]. However, as [48] suggested, the HTTP Archive is so far the only option that allows reproduction, as well as offering high repeatability. Secondly, the way we detected the supported versions for each technology was through manual search from search engines and official websites. This method allowed human error to happen and propagate up to conclusions. As far as our knowledge, this is the best method we could do since there is no centralized database available that tracks different types of technologies and the versions supported. Lastly, complexity and ambiguity in version numbering and the comparison may lead to different results when the method is reproduced without access to the exact source code. For example, there may be disagreement on deciding whether Bootstrap (a frontend library) version found "24-55-051" is within the supported version ">= 4" or not.

3. RESULTS AND DISCUSSION

Unlike previous research, we didn't have to take care of timeout problems or erroneous HTTP status codes, since all had been handled by the HTTP Archive. In the following discussions, we will also compare the result we get in this research, with our previous research [28].

In terms of scale, as can be seen in Table 3, we collected 11.639 times more websites than the previous research. This is possible thanks to the large scale of infrastructure provided by HTTP Archive and the Chrome User Experience Report data collection process. In terms of technologies identified, the current research found 40.182 times more technologies, which means 4x more technologies identified per website. The number of different technologies identified was also increased by about 10x compared to previous research.

Table 4 shows the overall technologies count for measurement results, compared to previous research. Numbers shown on the first lines of each cell are marked with (P) and come from previous research. The numbers shown on the second lines of each cell are marked with (C) and come from current research. In terms of the overall measurement result, we still found that most of the technologies reported were not versioned, at 69.09%. This value is slightly less than the previous research at 70.37%, but there is also an increase in non-conclusive results from 11.04% to 15.04%. This means that even when we have more versions detected, we are still unable to decide whether such versions are still supported or not.

On first try. we received a total 512.842.625 technologies. which is a difference of 33.937 technologies compared to 512.808.688 as found in Table 3. It turned out that the technologies table has double entry for technology "Fourthwall". After removing the duplicat, we could get the exact sum. Subsequent queries use the cleaned technology list.

Table 5 shows the number of non-versioned, non-conclusive, unsupported, and supported technologies from the top 10 websites as listed before in Table 1. We also put information on the top 10 measurement results from the previous research. Each cell of the table body is also composed of numbers from previous research (P) and current research (C), for comparison. In general, current results provide more diverse technologies identified.

Table 6 shows the number of unsupported technologies used in a website, grouped by the website rank. The first line in each cell represents data from previous research, while the second line represents data from current research. The numbers shown are the ratio of websites using n unsupported technologies. For example, the number 0.173 in row "1-150" column "n = 2" means that $0.173 \times 150 \approx 26$ websites having ranks ranging from 1 to 150 are using exactly 2 unsupported technologies, based on previous research data. Finally, cyan and purple bars are drawn to represent the numbers visually. While in previous research we found no correlation between website ranks and the number of unsupported technologies, the bars from current research show that up to the 1.000.000th rank, the number of unsupported technologies used increases as the rank increases, marked by steadily decreasing factor on n=0, n=1 and n=2, and steadily increasing factor on n >= 4.

Table 3. Scale comparison with previous research

Result	Previous research	Current research	Scale
Coverage	Indonesia	Worldwide	N.A.
Number of websites	1.439	16.747.881	11.639×
Number of technologies detected	12.762	512.808.688	40.182×
Number of different technologies	331	3.323	10×

Table 4. Overall technologies count for measurement result. compared to previous research

Result	Technologies count	Percentage
Not versioned	8.980 (P)	70.37
	354.309.633 (C)	69.09
Non-conclusive	1.409 (P)	11.04
	77.120.611 (C)	15.04
Unsupported	1.508 (P)	11.82
	49.186.986 (C)	9.59
Supported	865 (P)	6.78
	32.191.428 (C)	6.28
Total	12.762 (P)	100.00
	512.808.688 (C)	100.00

Table 5. First 10 results of measurements. compared to previous research

rank	url	not-versioned	non-conclusive	unsuppor-ted	supported
1	(P) okezone.com	7	0	1	1
1.000	(C) https://www.pa---nx.com/	6	0	1	0
2	(P) google.com	1	0	2	0
1.000	(C) https://www.gov.br/	40	20	8	0
3	(P) tribunnews.com	11	2	0	0
1.000	(C) https://tecno.servicewebly.com	10	2	1	0
4	(P) youtube.com	1	1	2	0
1.000	(C) https://bbs.animach.com/	62	4	4	8
5	(P) grid.id	11	1	0	1
1.000	(C) https://www.google.co.th/	12	0	0	0
6	(P) detik.com	8	1	0	0
1.000	(C) https://igram.io/	48	8	0	4
7	(P) kompas.com	10	2	1	0
1.000	(C) https://clever.com/	24	2	2	1
8	(P) sindonews.com	4	1	1	0
1.000	(C) https://veja.abril.com.br/	138	4	0	4
9	(P) tokopedia.com	5	0	0	0
1.000	(C) https://m.gsmarena.com/	24	0	2	0
10	(P) liputan6.com	11	1	1	0
1.000	(C) https://www.amazon.co.jp/	32	0	4	0

Table 6. Number of unsupported technologies grouped by website rank. compared to previous research

rank	n=0	n=1	n=2	n=3	n>=4
(P) 1-150	0.373	0.387	0.173	0.060	0.007
(C) 1-1K	0.317	0.091	0.068	0.013	0.511
(P) 151-300	0.347	0.367	0.193	0.080	0.013
(C) 1K-5K	0.250	0.069	0.058	0.025	0.598
(P) 301-450	0.393	0.287	0.213	0.067	0.040
(C) 5K-10K	0.250	0.069	0.058	0.025	0.598
(P) 451-600	0.373	0.320	0.147	0.140	0.020
(C) 10K-50K	0.233	0.049	0.041	0.020	0.657
(P) 600-750	0.393	0.387	0.147	0.067	0.007
(C) 50K-100K	0.224	0.044	0.037	0.018	0.678
(P) 751-900	0.453	0.293	0.167	0.053	0.033
(C) 100K-500K	0.206	0.038	0.031	0.014	0.711
(P) 900-1.051	0.433	0.280	0.200	0.067	0.020
(C) 100K-500K	0.193	0.036	0.030	0.013	0.729
(P) 1.051-1.200	0.373	0.307	0.227	0.067	0.027
(C) 1M-5M	0.181	0.058	0. 048	0.025	0.688
(P) 1.201-1.350	0.333	0.380	0.207	0.073	0.007
(C) 5M-10M	0.191	0.107	0.081	0.042	0.579
(P) 1.351-1.500	0.413	0.307	0.193	0.073	0.013
(C) 10M-50M	0.223	0.115	0.077	0.037	0.548

3.1. Maintainer supports per technology

Like previous research, we also look at the top 15 technologies that contain version information in at least 1 website and identify how much of them are used with support from their maintainers. Table 7 shows the result using similar formatting as earlier tables. For example, the technology "jQuery" is sampled from 1.011 websites in previous research and 22.560.775 websites in current research. In worldwide data, more websites are using supported versions of jQuery (0.524) compared to Indonesian data (0.257).

The following subsections discuss the results of several technologies. Unlike in previous research where we could show graphs, in current research, the variation of versions found is just too large to show graphically or in table form here. The full data for more exploration is available on GitHub, while in this paper we present the most interesting results.

3.1.1. Nginx and Apache HTTP Server

Like previous research, only a small portion of Nginx servers are running supported versions. With 1.22.1 as the minimum supported version, we found mostly versions 1.22.1, 1.23.0, 1.23.1, 1.23.2, and 1.23.3 in the supported version groups (among others). There are also nonstandard versions found and detected as supported by our algorithm, such as 10.0, 3.0, and 9.99.9. We assume that those versions are crafted manually to hide the actual version number.

For Apache HTTP Server (simply known as "Apache" in previous research), there are many variations of versions in the supported group, with a minimum supported version of 2.3. We also found unusual version numbers too, such as 5.5.5, 6.6.6, and 7.0, despite those numbers not being listed in the official documentation. In general, less portion of Apache HTTP Server usage is found supported in current research compared to previous research (0.158 vs 0.333), but there is also a larger portion of not-versioned Apache usages in current research (0.782 vs 0.616), which means that more website maintainers worldwide hiding the Apache version, compared to Indonesia website maintainers.

3.1.2. PHP and WordPress

On PHP, there is also a higher percentage of unsupported versions found in the wild (0.314) compared to previous research (0.215). In previous research, we argued that the biggest difficulty in using supported versions of PHP was upgrading from PHP 5.x to PHP 7.x. In this research, the minimum supported version of PHP is 8.1, and this gives more difficulties for PHP developers to upgrade from unsupported to supported versions. Given so many websites are using unsupported PHP versions and the difficulties upgrading, alternatives were made. One of which was Suhosin, an extension for protecting unsupported PHP versions.

For WordPress, the percentage of supported versions is higher in current research (0.420) compared to previous research (0.341). We still see the ease of upgrading in WordPress system as the main driver of WordPress websites using supported versions. Like other technologies, we also found unusual version numbers, such as 999.9, 99.7.4, and 9870.

Table 7. Top technologies used, compared to previous research

num_sites	name	supported	un-supported	non-conclusive	not-versioned
1.011	(P) jQuery	0.257	0.729	0.000	0.014
22.560.775	(C) jQuery	0.524	0.462	0.000	0.014
346	(P) WordPress	0.341	0.118	0.017	0.523
20.137.529	(C) WordPress	0.420	0.233	0.000	0.347
	(P) N.A				
17.161.361	(C) Google Analytics	0.000	0.000	0.217	0.783
591	(P) PHP	0.200	0.215	0.000	0.585
15.985.901	(C) PHP	0.013	0.314	0.000	0.674
478	(P) Nginx	0.010	0.243	0.000	0.747
15.761.927	(C) Nginx	0.018	0.142	0.000	0.840
	(P) N.A				
11.224.457	(C) core-js	0.002	0.985	0.000	0.013
298	(P) jQuery Migrate	0.000	0.000	0.896	0.104
9.877.869	(C) jQuery Migrate	0.000	0.000	0.961	0.039
	(P) N.A				
9.442.075	(C) Yoast SEO	0.000	0.000	0.821	0.179
430	(P) Bootstrap	0.265	0.530	0.000	0.205
8,889.330	(C) Bootstrap	0.288	0.396	0.097	0.218
237	(P) Apache	0.333	0.042	0.008	0.616
8.232.254	(C) Apache HTTP Server	0.158	0.021	0.048	0.782

3.1.3. jQuery, jQuery Migrate, and Bootstrap

JQuery and Bootstrap are two of the most popular JavaScript frameworks [49], while jQuery Migrate is a tool to allows websites still using older jQuery versions to work with newer ones.

On jQuery, we found a significantly higher percentage of supported versions (0.524) compared to previous research (0.257). On jQuery Migrate, like previous research, we were unable to determine the percentage of unsupported/supported versions. This is because there is no information on what the minimum supported version is actually. On Bootstrap, the percentage of supported versions (0.265) is roughly the same as in previous research (0.288). Compared to other technologies studied in this research, Bootstrap has the most variation of versions (14.445 different versions). The unusual version numbers look like Git commit hash but up to the time this paper is finalized, we could not confirm the exact cause. It is also worth noting that starting from Bootstrap version 5, jQuery is no longer a requirement for using Bootstrap.

3.1.4. core-js

Core-js was under the radar in previous research but shows up in our detection in this research. It is a “polyfill” technology. With new browsers released, they came up with new features as well as breaking changes to older scripts. core-js polyfill helps codes written for older browsers to work well with newer browsers, like jQuery Migrate which helps codes written for older versions of jQuery to work with newer versions of jQuery. core-js has a strict policy of only the latest released version is supported, therefore only a small percentage (0.002) of versions are supported. However, the main contributor is facing difficulties in continuing the project, leading to uncertainty about the future of this technology [17].

3.1.5. Google Analytics

Google Analytics, also not detected in previous research, has only two notable versions: UA (Universal Analytics) and 4 (Google Analytics 4). Google dropped support for Universal Analytics on 1 July 2023 and the news was announced months before, so we saw significant numbers of websites already migrating to Google Analytics 4 in our report. As discussed in [50], Google Analytics is widely due to its ease of use to track the statistics of a website.

3.1.6. Yoast SEO

Yoast SEO is a popular WordPress plugin to improve the SEO (Search Engine Optimization) score for WordPress websites. There are many variations of Yoast SEO versions, but unfortunately, we could not find information on what versions are supported.

4. CONCLUSION

In this research. we have constructed a process for measuring the number of unsupported technologies in websites worldwide based on the HTTP Archive database. The process itself is a combination of automated processes using SQL queries as well as manual work for identifying the minimum supported version of technologies. However. the reader should be aware that there were some limitations of this research. as discussed in the Section 2.5.

We have also run the process against the database of 2023_01_01 (January 2023) to see what the state of unsupported technologies at that time is. In general. the top technologies used by international websites are about the same as the technologies used by Indonesian websites. In terms of supported technologies. some technologies are supported more than the Indonesia website counterparts. such as jQuery and WordPress. For Apache HTTP Server. more international websites seem to hide the version numbers. likely for security reasons. We also found that the higher the website rank is. the greater the number of supported technologies used. Due to the difficulties of upgrading technology dependencies to a supported version. it is understandable that websites with lower ranks (arguably having less budget than the higher-rank ones) do not always use the supported version. A general suggestion for library developers is to make upgrading as easy as possible. like what has been successfully done by the WordPress team.

Since we have devised a process for measurement. potential future research may involve longitudinal study. i.e.. repeating the process for several months of data of HTTP Archive. similar to what has been done in [29]. [31]. Further research may also be done to understand the impact of using different methods in collecting data. For example. Google Analytics was not found in the top 10 technologies of the previous research. while in the current research. it was in the 3rd place. Another potential work is to have a sustainable centralized database that records the supported versions for different technologies. This is significantly harder work to do and requires great incentives for people to build. In this research, we have constructed a process for measuring the number of unsupported technologies in websites worldwide based on the HTTP Archive database. The process itself is a combination of automated processes using SQL queries as well as manual work for identifying the minimum supported version of technologies. However, the reader should be aware that there were some limitations of this research, as discussed in the section 2.5.

We have also run the process against the database of 2023_01_01 (January 2023) to see what the state of unsupported technologies at that time is. In general, the top technologies used by international websites are about the same as the technologies used by Indonesian websites. In terms of supported technologies, some technologies are supported more than the Indonesia website counterparts, such as jQuery and WordPress. For Apache HTTP Server, more international websites seem to hide the version numbers, likely for security reasons. We also found that the higher the website rank is, the greater the number of supported technologies used. Due to the difficulties of upgrading technology dependencies to a supported version, it is understandable that websites with lower ranks (arguably having less budget than the higher-rank ones) do not always use the supported version. A general suggestion for library developers is to make upgrading as easy as possible, like what has been successfully done by the WordPress team.

Since we have devised a process for measurement, potential future research may involve longitudinal study, i.e., repeating the process for several months of data of HTTP Archive, similar to what has been done in [29], [31]. Further research may also be done to understand the impact of using different methods in collecting data. For example, Google Analytics was not found in the top 10 technologies of the previous research, while in the current research, it was in the 3rd place. Another potential work is to have a sustainable centralized database that records the supported versions for different technologies. This is significantly harder work to do and requires great incentives for people to build.

REFERENCES

[1] H. Hassani, X. Huang, and E. Silva, "The Human Digitalisation Journey: Technology First at the Expense of Humans?," *Information*, vol. 12, no. 7, Jul. 2021, https://doi.org/10.3390/info12070267.

[2] D. Schilirò, "Digital transformation, COVID-19, and the future of work." Accessed: Mar. 22, 2024. [Online]. Available: https://mpra.ub.uni-muenchen.de/108817/

[3] Y. Lou and Q. Sun, "Over-reliance on database: A case study of using web of science," *Hum. Behav. Emerg. Technol.*, vol. 3, May 2021, https://doi.org/10.1002/hbe2.266.

[4] B. Bogdandy, J. Tamas, and Z. Toth, "Digital Transformation in Education during COVID-19: a Case Study," in *11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 000173–000178, Sep. 2020, https://doi.org/10.1109/CogInfoCom50765.2020.9237840.

[5] Y. K. Dwivedi *et al.*, "Impact of COVID-19 pandemic on information management research and practice: Transforming education, work and life," *Int. J. Inf. Manag.*, vol. 55, p. 102211, Dec. 2020, https://doi.org/10.1016/j.ijinfomgt.2020.102211.

[6] F. J. García-Peñalvo, "Digital Transformation in the Universities: Implications of the COVID-19 Pandemic," *Transformación digital en las universidades: Implicaciones de la pandemia de la COVID-19*, Feb. 2021, Accessed: Mar. 22, 2024. [Online]. Available: https://repositorio.grial.eu/handle/grial/2230.

[7] X. Liu *et al.*, "Cyber security threats: A never-ending challenge for e-commerce," *Front. Psychol.*, vol. 13, Oct. 2022, https://doi.org/10.3389/fpsyg.2022.927398.

[8] T. N. Hai, Q. N. Van, and M. N. Thi Tuyet, “Digital Transformation: Opportunities and Challenges for Leaders in the Emerging Countries in Response to Covid-19 Pandemic,” *Emerg. Sci. J.*, vol. 5, pp. 21–36, May 2021, https://doi.org/10.28991/esj-2021-SPER-03.
[9] J. Iida, “Digital Transformation vs COVID-19: the Case of Japan,” *Digit. Law J.*, vol. 1, no. 2, Aug. 2020, https://doi.org/10.38044/2686-9136-2020-1-2-8-16.
[10] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions,” *Electronics*, vol. 12, no. 6, Jan. 2023, https://doi.org/10.3390/electronics12061333.
[11] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, “A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions,” *ACM Comput. Surv.*, vol. 54, no. 11s, p. 238:1-238:37, Sep. 2022, https://doi.org/10.1145/3514229.
[12] A. Taivalsaari, *et al.*, “Full Stack Is Not What It Used to Be,” *International conference on web engineering*, pp. 363–371, 2021, https://doi.org/10.1007/978-3-030-74296-6_28.
[13] J. Shropshire, J. Landry, and S. Presley, “Towards a Consensus Definition of Full-Stack Development,” *SAIS*, Proc., Mar. 2018, [Online]. Available: https://aisel.aisnet.org/sais2018/17.
[14] A. Nowogrodzki, “How to support open-source software and stay sane,” *Nature*, vol. 571, no. 7763, pp. 133–134, Jul. 2019, https://doi.org/10.1038/d41586-019-02046-0.
[15] X. Tan and M. Zhou, “Scaling Open Source Software Communities: Challenges and Practices of Decentralization,” *IEEE Softw.*, vol. 39, no. 1, pp. 70–75, Jan. 2022, https://doi.org/10.1109/MS.2020.3025959.
[16] S. Butler *et al.*, “Considerations and challenges for the adoption of open source components in software-intensive businesses,” *J. Syst. Softw.*, vol. 186, p. 111152, Apr. 2022, https://doi.org/10.1016/j.jss.2021.111152.
[17] A. Kritikos and I. Stamelos, “A resilience-based framework for assessing the evolution of open source software projects,” *J. Softw. Evol. Process,* vol. 36, no. 5, p. e2597, 2024, https://doi.org/10.1002/smr.2597.
[18] W. Schueller and J. Wachs, “Modeling interconnected social and technical risks in open source software ecosystems,” *Collect. Intell.*, vol. 3, no. 1, p. 26339137241231912, Jan. 2024, https://doi.org/10.1177/26339137241231912.
[19] T. Tervoort, M. T. De Oliveira, W. Pieters, P. Van Gelder, S. D. Olabarriaga, and H. Marquering, “Solutions for Mitigating Cybersecurity Risks Caused by Legacy Software in Medical Devices: A Scoping Review,” *IEEE Access*, vol. 8, pp. 84352–84361, 2020, https://doi.org/10.1109/ACCESS.2020.2984376.
[20] D. Upadhyay, N. Ware, and B. Mahesh, “Evolving Trends in Web Application Vulnerabilities: A Comparative Study of OWASP Top 10 2017 and OWASP Top 10 2021,” *Int. J. Eng. Technol. Manag. Sci.*, vol. 7, Nov. 2023, https://doi.org/10.46647/ijetms.2023.v07i06.038.
[21] K. E. Vaniea, E. Rader, and R. Wash, “Betrayed by updates: how negative experiences affect future security,” i*n Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Toronto Ontario Canada: ACM,* pp. 2671–2674, Apr. 2014, https://doi.org/10.1145/2556288.2557275.
[22] J. M. Haney and S. M. Furman, “User Perceptions and Experiences with Smart Home Updates,” *in IEEE Symposium on Security and Privacy (SP)*, pp. 2867–2884, May 2023, https://doi.org/10.1109/SP46215.2023.10179459.
[23] J. Stringer, A. Tahir, K. Blincoe, and J. Dietrich, “Technical Lag of Dependencies in Major Package Managers,” *in 27th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 228–237, Dec. 2020, https://doi.org/10.1109/APSEC51365.2020.00031.
[24] P. Lam, J. Dietrich, and D. J. Pearce, “Putting the Semantics into Semantic Versioning,” in *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pp. 157–179, Nov. 2020, https://doi.org/10.1145/3426428.3426922.
[25] F. Li, L. Rogers, A. Mathur, N. Malkin, and M. Chetty, “Keepers of the Machines: Examining How System Administrators Manage Software Updates For Multiple Machines,” *presented at the Fifteenth Symposium on Usable Privacy and Security (SOUPS*), pp. 273–288, 2019, Accessed: Mar. 19, 2024. [Online]. Available: https://www.usenix.org/conference/soups2019/presentation/li.
[26] K. Vaniea and Y. Rashidi, "Tales of Software Updates: The process of updating software," *In Proceedings of the chi conference on human factors in computing systems*, pp. 3215-3226, 2016, https://doi.org/10.1145/2858036.2858303.
[27] T. Duebendorfer and S. Frei, “Why Silent Updates Boost Security,” *TIK, ETH Zurich, Tech. Rep*, 2009, https://tik-old.ee.ethz.ch/file/ef72343372ca8659a9ae8a98873167c0/TIKReport302.pdf.
[28] P. A. Nugroho and H. Steven, “Measurement of Unsupported Applications used in Indonesia Popular Websites,” *J. Ilm. Tek. Elektro Komput. Dan Inform.*, vol. 7, no. 1, Apr. 2021, https://garuda.kemdikbud.go.id/documents/detail/2177793.
[29] N. Demir, T. Urban, K. Wittek, and N. Pohlmann, “Our (in)Secure Web: Understanding Update Behavior of Websites and Its Impact on Security,” *In Passive and Active Measurement: 22nd International Conference*, pp. 76–92, 2021, https://doi.org/10.1007/978-3-030-72582-2_5.
[30] F. R. Cogo, G. A. Oliva, and A. E. Hassan, “Deprecation of Packages and Releases in Software Ecosystems: A Case Study on NPM,” *IEEE Trans. Softw. Eng.,* vol. 48, no. 7, pp. 2208–2223, Jul. 2022, https://doi.org/10.1109/TSE.2021.3055123.
[31] K. Lim, Y. Kwon, and D. Kim, “A Longitudinal Study of Vulnerable Client-side Resources and Web Developers' Updating Behaviors,” *in Proceedings of the ACM on Internet Measurement Conference*, pp. 162–180, 2023, https://doi.org/10.1145/3618257.3624804.
[32] A. Kashaf, V. Sekar, and Y. Agarwal, “Analyzing Third Party Service Dependencies in Modern Web Services: Have We Learned from the Mirai-Dyn Incident?,” *in Proceedings of the ACM Internet Measurement Conference,* pp. 634–647, 2020, https://doi.org/10.1145/3419394.3423664.

[33] A. Kashaf, J. Dou, M. Belova, M. Apostolaki, Y. Agarwal, and V. Sekar, "A First Look at Third-Party Service Dependencies of Web Services in Africa," *in Passive and Active Measurement,* pp. 595–622, 2023, https://doi.org/10.1007/978-3-031-28486-1_25.
[34] R. Kumar, S. Asif, E. Lee, and F. E. Bustamante, "Each at its Own Pace: Third-Party Dependency and Centralization Around the World," *Proc. ACM Meas. Anal. Comput. Syst*., vol. 7, no. 1, pp. 1-4, Mar. 2023, https://doi.org/10.1145/3579437.
[35] T. Urban, M. Degeling, T. Holz, and N. Pohlmann, "Beyond the Front Page:Measuring Third Party Dynamics in the Field," *in Proceedings of The Web Conference*, pp. 1275–1286, Apr. 2020, https://doi.org/10.1145/3366423.3380203.
[36] Y. Takata, H. Kumagai, and M. Kamizono, "The Uncontrolled Web: Measuring Security Governance on the Web," *IEICE Trans. Inf. Syst*., vol. E104.D, no. 11, pp. 1828–1838, Nov. 2021, https://doi.org/10.1587/transinf.2021NGP0003.
[37] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vulnerable Open Source Dependencies: Counting Those That Matter," *In Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, pp. 1-10, 2018, https://doi.org/10.1145/3239235.3268920
[38] M. Zaidan, F. Noeraini, Z. Sari, and D. R. Akbi, "Website Vulnerability Analysis of AB and XY Office in East Java," *J. Ilm. Tek. Elektro Komput. Dan Inform*., vol. 9, no. 2, no. 2, Jun. 2023, https://garuda.kemdikbud.go.id/documents/detail/3575344.
[39] D. Priyawati, S. Rokhmah, and I. C. Utomo, "Website Vulnerability Testing and Analysis of Website Application Using OWASP," *Int. J. Comput. Inf. Syst. IJCIS*, vol. 3, no. 3, no. 3, Aug. 2022, https://doi.org/10.29040/ijcis.v3i3.90.
[40] M. A. Muin, K. Kapti, and T. Yusnanto, "Campus Website Security Vulnerability Analysis Using Nessus," *Int. J. Comput. Inf. Syst. IJCIS*, vol. 3, no. 2, no. 2, Jun. 2022, https://doi.org/10.29040/ijcis.v3i2.72.
[41] R. Armando, I. G. A. K. A. Melyantara, R. Elfariani, D. F. A. Latuconsina, and M. Nasrullah, "IT Support Website Security Evaluation Using Vulnerability Assessment Tools," *J. Inf. Syst. Inform*., vol. 4, no. 4, Nov. 2022, https://doi.org/10.51519/journalisi.v4i4.330.
[42] N. A. bt Mohd and Z. F. Zaaba, "A Review of Usability and Security Evaluation Model of Ecommerce Website," *Procedia Comput. Sci*., vol. 161, pp. 1199–1205, Jan. 2019, https://doi.org/10.1016/j.procs.2019.11.233.
[43] N. Demir, M. Große-Kampmann, T. Urban, C. Wressnegger, T. Holz, and N. Pohlmann, "Reproducibility and Replicability of Web Measurement Studies," *in Proceedings of the ACM Web Conference* pp. 533–54, 2022. https://doi.org/10.1145/3485447.3512214.
[44] J. Wendroth, "A Brief Overview on HTTP," *Seminar IITM SS 22, Network Architectures and Services*, 2022, https://doi.org/10.2313/NET-2022-11-1_11.
[45] J. Rautenstrauch and B. Stock, "Who's Breaking the Rules? Studying Conformance to the HTTP Specifications and its Security Impact," *ACM ASIA Conference on Computer and Communications Security (AsiaCCS)*, 2024, https://publications.cispa.de/articles/conference_contribution/Who_s_Breaking_the_Rules_Studying_Conformance_to_the_HTTP_Specifications_and_its_Security_Impact/25826629/1.
[46] N. Demir, D. Theis, T. Urban, and N. Pohlmann, "Towards Understanding First-Party Cookie Tracking in the Field." *arXiv*, Feb. 10, 2022, https://doi.org/10.48550/arXiv.2202.01498.
[47] J. Woo, J. W. Choi, S. Jeon, J. K. Han, H. Kim, and S. S. Woo, "Tale of Two Browsers: Understanding Users' Web Browser Choices in South Korea," *in Financial Cryptography and Data Security*, pp. 3–17, 2020, https://doi.org/10.1007/978-3-030-54455-3_1.
[48] K. Ruth, D. Kumar, B. Wang, L. Valenta, and Z. Durumeric, "Toppling top lists: evaluating the accuracy of popular website lists," *Proc. 22nd ACM Internet Meas. Conf*., pp. 374–387, Oct. 2022, https://doi.org/10.1145/3517745.3561444.
[49] I. Zozas, I. Anagnostou, and S. Bibi, "Trends on Crowdsourcing JavaScript Small Tasks:," i*n Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering, Online Streaming, --- Select a Country ---: SCITEPRESS - Science and Technology Publications*, pp. 85–94. 2022, https://doi.org/10.5220/0011035800003176.
[50] T. Alby, "Popular, but hardly used: Has Google Analytics been to the detriment of Web Analytics?," *in Proceedings of the 15th ACM Web Science Conference,* pp. 304–311, Apr. 2023, https://doi.org/10.1145/3578503.3583601.

BIOGRAPHY OF AUTHORS

Pascal Alfadian Nugroho is a lecturer at Universitas Katolik Parahyangan Bandung. as well as the technical director for PT DNArtworks Komunikasi Visual. coordinating developers on building websites for commercial products. He holds a master's degree in Infocom Security from the National University of Singapore.

Raymond Chandra Putra. is a lecturer at Universitas Katolik Parahyangan. Bandung. He holds a master's degree in Informatics (subspecialis in Software Engineering) from Institut Teknologi Bandung. At UNPAR. he teaches many courses related to web programming and software engineering.

Rajasa Cikal Maulana. Rajasa Cikal Maulana Solihin is a Web Developer at PT. Bank Rakyat Indonesia. Rajasa holds a Bachelor's degree in Informatics Engineering from Universitas Katolik Parahyangan.

Vinson Tandra is a Computer Science graduate from Universitas Katolik Parahyangan Bandung and currently works as Back Office Staff at PT Asuransi Sinar Mas.