International Journal Software Engineering and Computer Science (IJSECS) 3 (3), **2023, 424-436**
Published Online December 2023 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN : 2776-4869, E-ISSN : 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v3i3.1850.

Comparative Analysis of PHP Frameworks for Development of Academic Information System Using Load and Stress Testing

Abdurrahman Niarman *
Department of Informatics Management, Universitas Islam Negeri Mahmud Yunus Batusangkar, Tanah Datar Regency, West Sumatra Province, Indonesia.
Email: aabniarman@uinmybatusangkar.ac.id

Iswandi
Department of Informatics Management, Universitas Islam Negeri Mahmud Yunus Batusangkar, Tanah Datar Regency, West Sumatra Province, Indonesia.
E-mail: iswandi@uinmybatusangkar.ac.id

Argi Kartika Candri
Faculty of Computer Science and Telecommunications, Wrocław University of Science and Technology, Wybrzeże Stanisława Wyspiańskiego st. 50-370 Wrocław, Poland.
E-mail: 268894@student.pwr.edu.pl

Received: 24 November 2023; Accepted: 4 December 2023; Published: 20 December 2023.

Abstract: There are many programming languages available for the developers to pick when they want to make a web services project such as Java JSP, Python, ASP.Net, PHP and many others. The reason that writer chose PHP as its research topic is that PHP is still being one of the most promising languages especially for developing web services. Several PHP Frameworks are now available on the internet promising that their framework can do better than others. Some popular PHP frameworks are Laravel, Symfony, CakePHP, Yii, and CodeIgniter. Performance is the most aspect to be concerned because of the market's demands entailing them to do so. The next concern that developer would likely need to decide is whether to go with Pure PHP or to go with PHP Framework. To answer those concerns, this research has been conducted in which a module is taken from an academic information system and will be developed into three similar and equivalent web applications by using Laravel, CodeIgniter, and Pure PHP. The author's aim in conducting this research is to obtain more in-depth information regarding the use of the pure PHP programming language and several frameworks such as CodeIgniter and Laravel. There are many statements stating that using the PHP programming language is no longer feasible, even though in reality using PHP for developing web-based applications is still the choice of many programmers based on statistics from w3tech. The results of research conducted by the author show that the PHP programming language is still very reliable in handling load and stress tests. The pure PHP programming language provides slightly better performance results than the CodeIgniter and Laravel frameworks. Hence, the use of pure PHP or frameworks can be determined from the level of needs of the application development to be created.

Keywords: Comparative Analysis; PHP Frameworks; Academic Information System; Load and Stress Testing; CodeIgniter; Laravel.

1. Introduction

There are many programming languages available for the developers to pick when they want to make a web services project such as Java JSP, Python, ASP.Net, PHP and many others. The reason that writer chose PHP as its research topic is that PHP is still one of the most promising languages especially for developing web services. Based on the data that w3tech.com collected up to May 2018, PHP is still the most used server-side programming language by having 83,4% shares [1]. PHP: Hypertext Preprocessor is a popular scripting language that is often associated with web development [2][3], even though it can be also used for other purposes. PHP was first introduced officially in November 1997, and it could not perform as it could perform today but just to create a simple dynamic web application. At first, PHP did not support Object-Oriented Programming until it was added to the system in PHP 3 and furthermore improved in PHP 4 [4].

Even though PHP is not as famous as any other programming language especially when creating web services, in fact, PHP is the most used server-side programming language by May 2018. PHP is used by 83.4% of all websites which the server uses PHP as its programming language. Based on the percentage that we have here, it means that PHP is still worth considering when we want to start developing a project specifically web services projects [1][5].

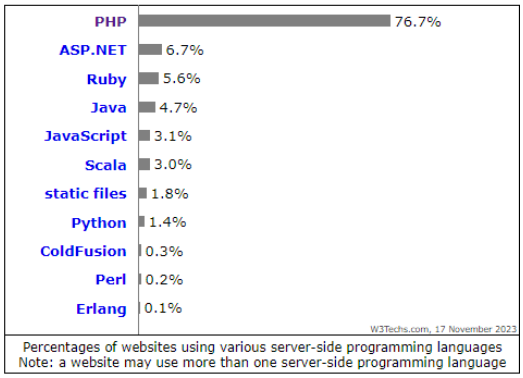

Figure 1. Percentage of websites using various server-side programming language

The PHP programming language is one of the languages supported by most servers today, so developing a web-based application using PHP will become easier for programmers. Server customization is one of the many jobs that must be completed by application developers, and by using PHP, programmers do not need to make too many adjustments to the server. By default, PHP is configured on most servers today, unlike other programming languages which require programmers to install or adjust here and there on the server side [6].

PHP is also very dynamic for various modern application development needs such as machine learning. PHP can be combined with other programming languages such as Python. PHP can be used as server-side and Python can be used as machine learning so that web-based applications displayed to users become more interactive [7]. The large community is also one of the reasons why PHP remains popular among programmers for developing web-based applications. In almost all social media, we can find a community of programmers who use pure PHP, CodeIgniter frameworks, Laravel or other frameworks. We can also study the documentation on the official website easily because it is accompanied by actual examples of its use.

Web-based applications provide several general benefits that neither desktop applications nor mobile applications can provide. Web-based applications can be accessed by anyone via any device and any operating system as long as the device has a browser and internet network. The capabilities of the device and the internet bandwidth used will have an impact on the smooth use of web-based applications. Many Indonesian government applications are web-based, such as Sinta, Sister, Pddikti and many others. Increasingly advanced technological developments are narrowing the gap between native applications and web-based applications in terms of executing a command. Many web-based applications use the Single-Page-Application (SPA) concept which allows each command to be executed by the server without having to force the browser to reload the page.

For the developers, most of the time, performance is the most aspect to be concerned because of the market's demands entailing them to do so. The next concern that developer would likely need to decide is whether to go with Pure PHP or to go with PHP Framework. Both Pure PHP and PHP Framework have their own pros and cons [8]. In the case of PHP Framework, the developer will be faced with several PHP Frameworks that he can choose to start developing their project. To answer those concerns, this study has been conducted in which a module is taken from an academic information system and will be developed into three similar and equivalent web applications by using Laravel, CodeIgniter, and Pure PHP [9][10].

First, a Pure PHP will be developed by using Object-Oriented Programming orientation. Pure PHP will also be using object-relational mapping (ORM) called ActiveRecord to get an equivalent comparison because the rests have their own built-in ORM. After that, Laravel and CodeIgniter version of this small web application will be developed. After the developing process has been done, all three systems will be run to analyze its performance based on its load and stress testing. The result of the data collected from the executions will be used to conclude this study [11].

2. Research Method

This study is conducted in three phases of development, as it requires to develop web application both in Pure PHP, CodeIgniter, and Laravel. As the study case of this study is an academic information system, the system will be huge and hard to compare one to another. Because of that, we decided to take only one module and develop the exact same module in the three applications equivalently. The equivalent here means that each application will use the same template, but it will use its own built-in functions in certain tasks. In order to get the dataset to be measured in the analysis part of this

study, there are a lot of measurements that can be taken into consideration. This academic information system is going to be developed into three equal applications with Pure PHP, CodeIgniter, and Laravel. Equal here means that the built-in functions will be used which are ready to use without involving too many third-party libraries. Basically, there is one big difference between these three applications, which is Object-relational mapping (ORM). Laravel has a built-in ORM called Eloquent while CodeIgniter uses a modified ActiveRecord ORM. Both Laravel and CodeIgniter use the same database pattern called Active Record while in some other cases Data Mapper database pattern is mostly used in other ORMs [12].

While CodeIgniter and Laravel have built-in ORM in it, the Pure PHP application will not have one as it will be developed from scratch top to bottom. As the word 'equal' earlier means that the application should be developed as equally as possible and ORM is a necessary library that will help to decrease developing time. Another reason why ORM is important in this study is that thousands of data will get involved and it will be a better idea to use ORM in this case. In the analysis part of this study, we will also see how the ORM helps Pure PHP execute all the processes. ActiveRecord version one is the ORM that is used to be coupled with the Pure PHP application. This ORM follows the same design pattern as CodeIgniter's ORM and Laravel Eloquent ORM which is the Active Record database pattern. This ORM supports PHP 5.3 or later and MySQL or MariaDb database which are exactly the requirements that will be used and implemented in this study [13].

There are two aspects that are going to be used as data set to conclude this study, such as the load and stress testing. The data set will be gathered into tables and visualized into graphs so that the reader can understand better how each application performs. To achieve the result of each experimental attribute as explained earlier, we need to use some built-in functions and third-party applications. The load and stress testing will be conducted by using JMeter which is an application to load test functional behavior and measure performance. JMeter is made by Apache Software Foundation, and it is specifically made for testing Web Applications. JMeter is an open-source application so that it can be used by programmers to test their web performance for free. It also can-do load and performance test in many different server/protocol types [14]. In this experiment, JMeter will be used to perform the load and stress testing to the local server. This performance testing is necessary to determine how the web application performs under heavy load and to analyze the overall performance. Load Testing is modeling the expected usage by simulating multiple users accessing the web services concurrently and stress testing is pushing the simulation to the web server with a high load capacity until it starts responding slowly and making errors. The purpose of performing stress testing is to find the maximum load the web server can handle.

2.1. System Requirement

Hardware and software specifications are necessary to get a proper result in the form of data sets. This study will be conducted by using a workstation with the specifications presented in Table 1.

Table 1. Hardware Specification

Name	Detail
Processor	Intel Core i5 12500H
Memory	16 GB 2394 MHz DDR5
Graphics	Intel Iris® Xe Graphics
Storage	SSD 512 GB NVME

The workstation is plugged into the power station anytime the experiment will be conducted to get the maximum performance, because there is a possibility of performance decreasing when the battery is not in a good condition and the workstation might suddenly lose its power when the stress testing is run for the experiment. The local server experiment that is used to run the applications is XAMPP version 8.0.28. XAMPP is a part of Apache Friends non-profit project with a mission to build an easy to install distribution for developers to get into the world of Apache. XAMPP includes MariaDB, PHP, and Perl. MariaDB is a fork of MySQL, the database structure, and indexes of MariaDB are the same as MySQL and it allows us to switch from one to another without the needs to alter the application as the data and data structure are the same [15]. One of the reasons XAMPP moved to MariaDB is because Oracle acquired MySQL and it affects the licensing of MySQL. Here is the detail specification of the server:

Table 2. Software Specification

Name	Detail
Server	XAMPP 8.0.28
PHP Version	8.0.28
Database	MariaDB 10.4.28
Browser	Google Chrome
Extension	JMeter

2.2. Data Sampling

Because only a module will be taken from a huge system of academic information system for this study, furthermore the actions that the user can do are limited only to the aspects that will be used later in the experiment. In this case, there is only one user which can do some actions. A module that will be used for this experiment is a student module since it has all the points that are going to be tested for load and stress testing later. There is no special treatment as to why this module was chosen as the experimental module, if the selected module has enough features and is in accordance with the experimental aspects, then the module can be used. User will have access to student module in which has some features such as ;1) loading the data from database to the application; 2) generating 1000 users to the database 3) editing all data in the database at once; 4) deleting all data in database at once; 5) exporting all data from database to csv; 6) uploading photo to server and searching certain letter.

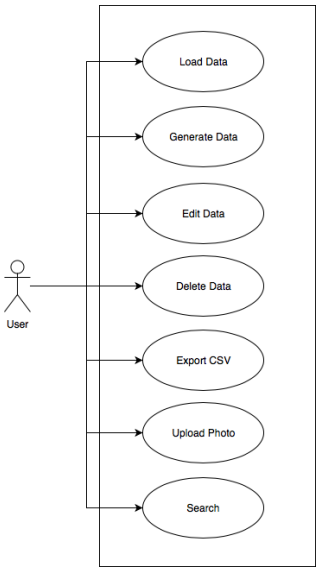

Figure 1. Use Case Diagram of the System

An academic information system is categorized as a big system, as it must handle not only the features but also the huge number of users that will use the system [16]. This study is focusing on a simple academic information system for senior high school, and we will use only a module from the system. Because there is only one module that will be analyzed in this study, therefore the table that is used is also only one. Here is the structure of the table that will be used in this study:

Table 3. The Table Structure

Name	Type	Extra
nim	Int(11)	Primary, Auto_increment
name	Varchar(40)	
gender	Enum('M','W')	
date_birth	Date	
place_birth	Varchar(30)	
cd_religion	Varchar(10)	
photo	Varchar(100)	
id_studygroup	Int(11)	

In order not to distract the performance of each application, later, each application will have its own database with the exact same structure as in the table above. There are some attributes which are the foreign key to other tables such as cd_religion and id_studygroup, but because only one module and one table will be used in this study, the attributes are just left to stay in the table as the complementary.

2.3. Measuring the Performance

The server will be tested with such a configuration that will simulate a few users accessing a certain page and the number of users will gradually increase from 50, 100, 200, 300, 500, 600, 1000, 1500 and 2000. The threads in the range of 50 – 500 are considered as load testing in this experiment as it will not give too much pressure to the server while the threads between 600 and 2000 will be considered as stress testing as it will give much more pressures to the server.

Table 4.Category of Performance Testing with respect to the number of threads

Number of Threads	Category
50	Load Testing
100	
200	
300	
500	
600	Stress Testing
1000	
1500	
2000	

1) Load and Stress Testing

To run this experiment, a test plan will be made in Apache JMeter containing Thread Group which will record the Thread Properties such as the Number of Threads (users), Ramp-Up Period (in seconds) and Loop Count.

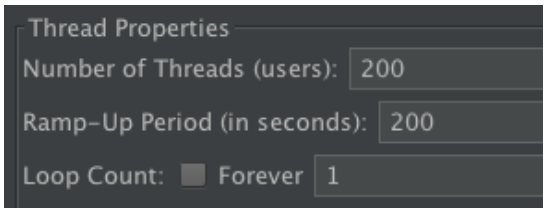

Figure 2. Configuration of Thread Group in JMeter

The picture above can be described as:

a. Number of Threads : Number of users that connect to the target server is 200 users.
b. Ramp-Up Period : The gap between executing one user and other users is 200 seconds in total.
c. Loop Count : Number of times to execute the testing is only one time.

If we put the configuration above into words, it will be: This experiment will be run one time with a total of 200 users. Each user will be run 1 second after another user has run. It is because the Ramp-Up Period has a value of 200, meaning that the delay between one user and other users is 200 users/200 seconds = 1 second. The thread group will also have an element called HTTP Request Defaults which contains the server's name or IP and the port used in our workstation to access the localhost. There is also a controller called Recording Controller which will save all the HTTP request elements in it. There are also two listeners, they are View Results in Table and Summary Report which View Results in Table will record all the executing processes in Detail and Summary Report will summarize the results from View Results in Table.

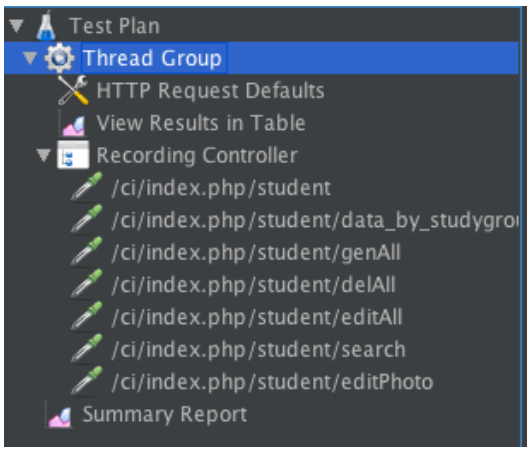

Figure 3. Complete JMeter Configuration Used in The Experiment

2) Execution Time

Execution time is used in this study to get to know the time needed for the application to execute a certain function. PHP already has a built-in function that can be used to calculate the time needed by placing some codes in the beginning and at the end of the code that will be counted its execution time. The function that is used to calculate this execution time is 'micro time' that will return current Unix timestamp with microseconds [17]. The way that we used to calculate the elapsed time is by calculating the differences between the starting time and the ending time of codes which are placed in

between. The result of this calculation will be automatically in seconds' format. By comparing the time elapsed of each application, we could analyze how each application performs in certain functions.

3) Memory Usage

When comparing the performance of applications, the memory used by the application to execute certain functions is necessary. By doing this, how the application performs could be analyzed and could improve the performance of the application if later, we find out that the application took more memories than what we expected which also means there is something wrong that affects the performance of the application. In this study, PHP's built-in function will be used to get the value of how much memory used by an application to execute certain code. The built-in function that is used is '*memory_get_usage(false)*'. The function can be set as True and as False, in this case, we are going with false because PHP theoretically will reserve a large amount of memory and it will return this amount of reserved memory to us if we set the function as True. Whilst it will return to us the actual amount of memory that is used in certain code when we set the function to False.

3. Result and Discussion

3.1 Results

An application, specifically an academic information system, theoretically should be able to handle a heavy request to the server by the users. This is this load and stress testing are performed in this study to simulate the peak request to the server where the application is placed in. Load testing will simulate a normal situation when the request to the server is not more than 500 users, and the stress testing will simulate when the request to the server is more than 500 to 2000 requests. Because the load and stress testing are not only measuring one page but seven pages, it means if we simulate with 2000 threads or users, it will put 14000 requests to the server which really is putting much pressure on the server and system. Finally, we could analyze how each application handles the light and heavy requests.

3.1.1 Execution Time on the local server

CodeIgniter overpowers others in almost every aspect used for this experiment except for the upload photo part where Pure PHP works outstanding by taking only 0.34 milliseconds to execute the function. Pure PHP comes second after CodeIgniter even though it has a problem to generate 1000 data by taking almost 590 milliseconds to execute the function while CodeIgniter took only 17 milliseconds and Laravel which comes as third took only 122 milliseconds to execute the function. Despite its flaw when generating 1000 data, Pure PHP could handle other aspects easily by reaching the execution time less than 36 milliseconds in every aspect. Whilst Laravel, which comes third, has a problem with loading and generating data with execution time reaching almost 130 milliseconds. The good news is Laravel still could perform well in other aspects even though its execution time is still below the execution time of Pure PHP and CodeIgniter which perform much better than Laravel. As we can see in the table below, Pure PHP and Laravel have a problem loading the data while CodeIgniter could load the data with no effort if we compare the execution time that they have. The fastest system is CodeIgniter by having the execution time in the range of 3 – 17 milliseconds only.

Table 5. Result of Execution Time

System	Load 1000 Data	Generate 1000 Data	Delete All Data	Edit All Data	Upload Photo	Export CSV	Search
Pure PHP	35.29	589.85	9.34	8.32	0.34	5.75	3.20
CodeIgniter	8.32	17.26	3.68	4.12	10.00	4.34	3.07
Laravel	126.00	122.36	12.74	13.14	0.79	26.33	14.98

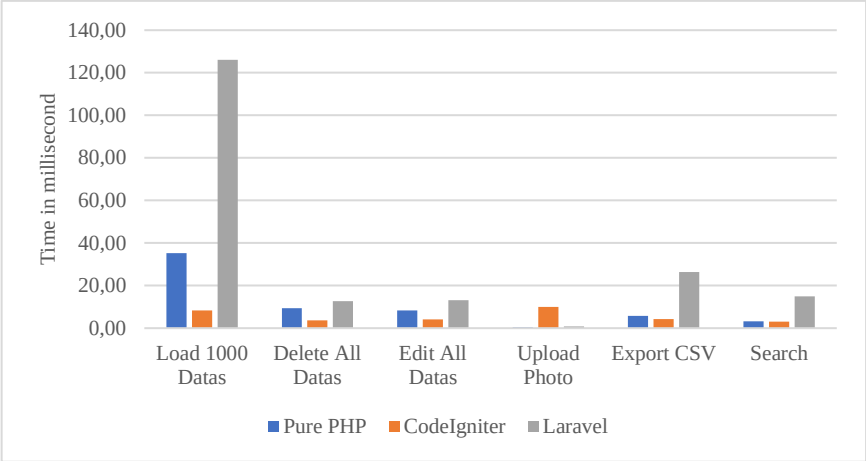

Figure 4. Result of Execution Time

3.1.2 Memory Usage on the local server

In this memory usage experiment section, Pure PHP excels in many aspects compared with other systems. It takes only around 2 Mb to execute the Search function while CodeIgniter needs 3.5 Mb and even more Laravel needs almost 9.5 Mb to execute this function. Pure PHP also excels in Loading, Deleting, Updating, Uploading, Exporting CSV which means that it performs very well by maintaining the memory consumed to execute a certain function. CodeIgniter comes second after Pure PHP by taking slightly more memory space when executing the functions but is still acceptable as it does not take more than 1 Mb with what Pure PHP has. CodeIgniter performs excellently when generating the data with only 2.9 Mb needed to execute the function. Based on the data collected for the execution time, CodeIgniter also performs astonishingly well when generating the data by taking only 17 milliseconds while other systems take more than 100 milliseconds. Laravel comes last by having the memory usage in the range of 6.7 Mb to 11.9 Mb, even the lowest number that Laravel achieved is still bigger than the highest number than other systems achieved. Laravel performs the worst in this part of the experiment as the memory needed to execute certain functions is much bigger than what other systems need, it even sometimes requires twice to thrice in the memory space. Loading and generating a high amount of data, in this case, 1000 data, are always a problem for Laravel as it needs much more time to execute and much more memory as well. Pure PHP comes first in this experimental part considering its lightweight system comparing to other systems which has a much more complex system as a framework, but CodeIgniter performs incredibly well by taking less than 1 Mb than what Pure PHP needs to execute the functions considering its complexity and hundreds of functions running in the background.

Table 6. Result of Memory Usage

Systems	Load 1000 data	Generate 1000 data	Delete All Data	Edit All Data	Upload Photo	Export CSV	Search
Pure PHP	4.00	3.34	2.26	2.25	2.12	2.66	2.00
CodeIgniter	4.92	2.90	2.82	2.82	2.91	3.43	3.56
Laravel	12.00	11.41	7.55	7.55	6.78	8.62	9.37

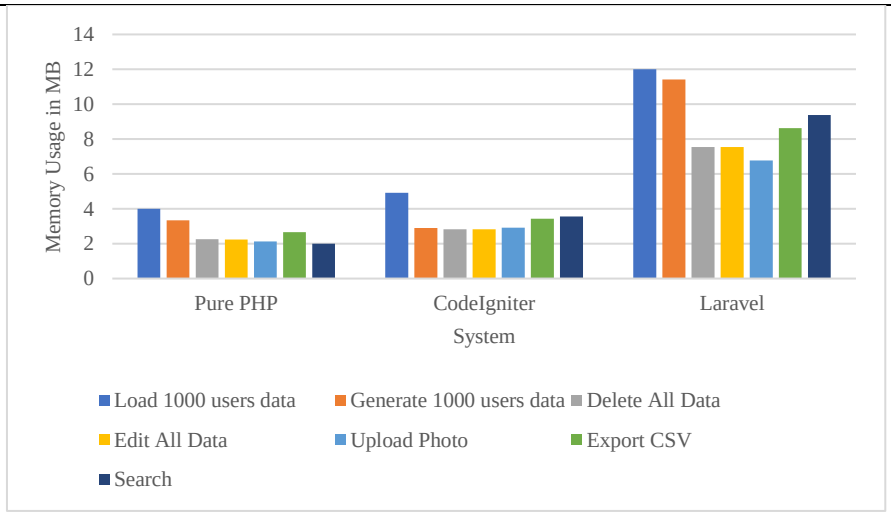

Figure 5. Result of Memory Usage

3.1.3 Performance test result on the local server

The table below shows the result of the performance test running on JMeter with a different number of users as the thread group [18]. The table also shows the total number of samples executed, the average response time shown in second, standard deviation shown in second and the throughput showed in kb/second.

Table 7. Pure PHP performance test result on the local server

Thread Group	Samples	Average (Sec)	Std. Deviation (Sec)	Throughput (kb/s)
50 Users	350	91	179.92	7.049
100 Users	700	92	183.92	7.019
200 Users	1400	94	187.76	7.005
300 Users	2100	95	191.74	7.006
500 Users	3500	86	170.59	7.004
600 Users	4200	85	169.20	7.003
1000 Users	7000	86	171.77	7.001
1500 Users	10500	76	156.65	7.000
2000 Users	14000	77	158.29	6.999

Table 8. CodeIgniter performance test result on the local server

Thread Group	Samples	Average (Sec)	Std. Deviation (Sec)	Throughput (kb/s)
50 Users	350	35	17.19	7.105
100 Users	700	36	17.28	7.049
200 Users	1400	35	17.97	7.023
300 Users	2100	35	17.25	7.015
500 Users	3500	33	16.76	7.009
600 Users	4200	33	17.09	7.008
1000 Users	7000	33	17.99	7.004
1500 Users	10500	35	19.65	7.002
2000 Users	14000	34	18.05	7.001

Table 9. Table 7. Laravel performance test result on the local server

Thread Group	Samples	Average (Sec)	Std. Deviation (Sec)	Throughput (kb/s)
50 users	350	125	204.98	7.043
100 Users	700	138	185.43	7.009
200 Users	1400	116	171.39	7.006
300 Users	2100	113	119.37	7.003
500 Users	3500	102	123.67	7.004
600 Users	4200	101	77.00	7.003
1000 Users	7000	104	99.96	7.001
1500 Users	10500	101	151.47	6.999
2000 Users	14000	106	79.60	6.998

3.1.4 Comparison of the average response time

1) Load Testing

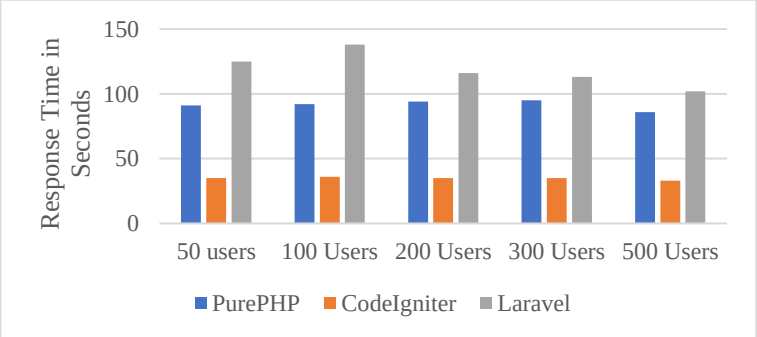

Figure 6. Load Testing - Comparison of Average of Response Time

The comparison was conducted by running the performance test five times with a different number of threads or users. As we can see in the graph above, the trend that happens is the average response time increases as the number of users increases. Even though it does not apply on Laravel as it decreases as the number of users increases, especially when the number of the users is more than 200 threads or users. Theoretically, the number of average response time is supposed to

get higher as the number of user increases, but the error in HTTP request could be the one who is responsible for the changing of the result. Pure PHP and CodeIgniter perform much better than Laravel, especially CodeIgniter with only 36 seconds at its highest achieved average response time. CodeIgniter is stable with the result as it stands on the range of 35 to 36 seconds resulting CodeIgniter as the best competitor in this case. Pure PHP, on the other hand, is also doing well as it really performs as what the theory says, the higher number of the user then the higher average response of time achieved.

2) Stress Testing

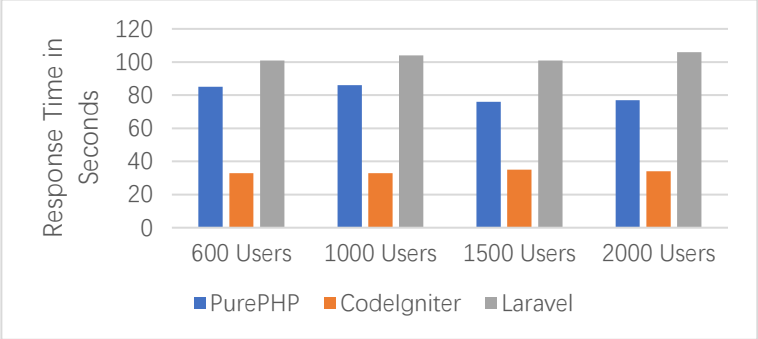

Figure 7. Stress Testing - Comparison of Average of Response Time

The stress testing was conducted by running the performance test four times with a different number of threads or users. CodeIgniter still leads the result by achieving the average response time much lower than other competitors. The highest response time that CodeIgniter achieved was 35 seconds with 1500 threads. Pure PHP performs a little bit better as the threads increase. The average response time that Pure PHP achieved with 1500 users and 2000 users is better than when the users were 600 and 1000. It means that Pure PHP could handle the heavy load that was pushed to the server and even maintain the performance to become better. While Laravel is vice versa, it does not show any sign that it performs better or worse when the number of threads increases. Laravel is stuck with the average response time in between 100 to 120 seconds.

3.1.5 Comparison of the standard deviation

1) Load Testing

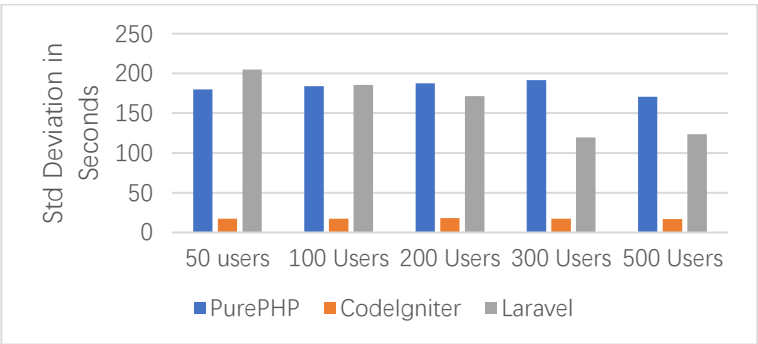

Figure 8. Load Testing - Comparison of Standard Deviation

The graph above shows the comparison of standard deviation between the competitors in this study. The graph shows how many exceptional cases were found when running the performing test which deviated from the average value of the receiving time. The lesser the value meaning the better its system performs as it means that the system could have better consistency of time pattern. As the graph above shows us that CodeIgniter excels in every situation set on the performance test, whilst Pure PHP and Laravel are competing by having almost similar results when the number of the users is equal or less than 200 users. Laravel performs much better than Pure PHP when more extreme stress condition is applied to the test, meaning that Laravel could handle more stress than Pure PHP when the system is pressed by the high number of users requesting to access the system.

2) Stress Testing

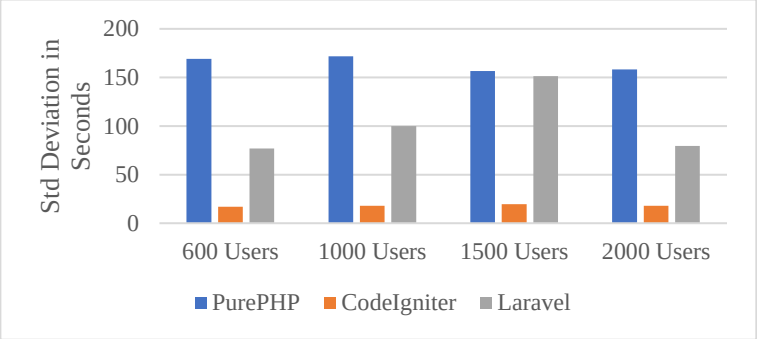

Figure 9. Stress Testing - Comparison of Standard Deviation

The graph above shows stress testing of the comparison of standard deviation between the competitors in this study. It is obvious that CodeIgniter is still leading in this comparison as like in the load testing. The gap between CodeIgniter and other competitors is just too wide for other competitors to catch up CodeIgniter. Pure PHP, shown as the blue bar in the graph above, is consistent in its position by having the standard deviation in the range of somewhere 170-190. The increasing of users does not give much effect to its standard deviation result. In the other hand, Laravel is giving a good result which is proving that Laravel could handle the heavy load pushed to the server. The result that Laravel achieved when the threads increase is much better than when the threads are much less. Even though the performance of Laravel is not as stable as other competitors, it still gives a better performance when the request to the server increases which also means that Laravel is suited to the bigger project.

3.1.6 Comparison of the throughput

1) Load Testing

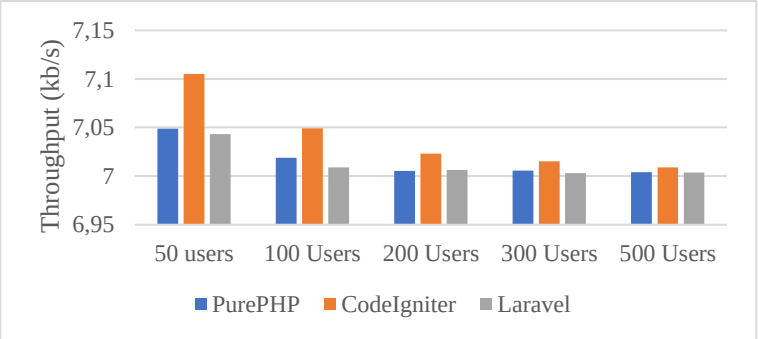

Figure 10. Load Testing - Comparison of Throughput

The graph above tells us that CodeIgniter has a higher value in all conditions compared with other competitors. Theoretically, the higher value of the Throughput then the better the system is. The result of this performance test was recorded as kilobyte per second. The trend that happens based on the graph above is the more users requesting the page then the lower throughput that the system could achieve. Throughput tells us how many requests per second can be handled by the system, for example is the result of CodeIgniter when having 50 users. CodeIgniter has a throughput of 7.10501 which means that it can handle approximately 7.105 per second. Laravel performs slightly better than Pure PHP in this case, but the more stress put into the system, the harder it is for Laravel to cope with it. Laravel could only perform almost the same with Pure PHP when the number of users is more than 100.

2) Stress Testing

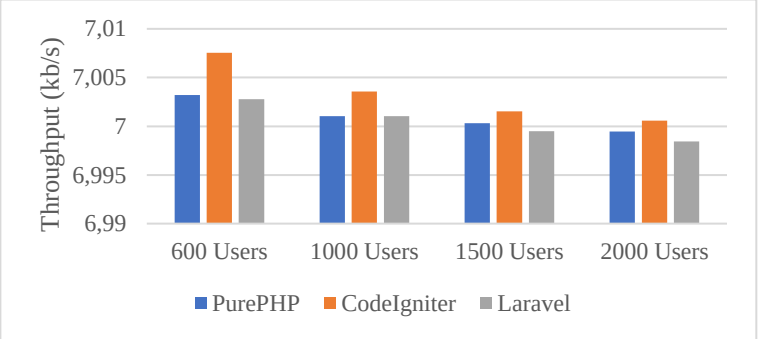

Figure 11. Stress Testing - Comparison of Throughput

In the stress testing, CodeIgniter still performs better than Pure PHP and Laravel. CodeIgniter is stable in its position, it does not matter how many the threads are. CodeIgniter could provide users with more content under heavy loads by having a little bit higher throughput than others, whilst Pure PHP and Laravel are stagnant in their position. The throughput that Pure PHP and Laravel achieved are almost similar in every condition with Pure PHP usually having slightly better results than Laravel.

3.2. Discussion

The research focuses on evaluating the performance of different systems in handling load and stress testing for academic information systems. The main goal is to assess how well the system handles varying levels of requests to the server. Load testing simulates normal situations with up to 500 users, while stress testing simulates heavier loads with more than 500 to 2000 requests. This study used seven different functions for testing, including loading data, generating data, deleting data, editing data, uploading photos, exporting CSV files, and searching. The results show that CodeIgniter performs very well in most aspects, with Pure PHP showing excellent performance in the photo upload function. Although Pure PHP has the disadvantage of generating 1000s of data, it consistently handles other functions efficiently with an execution time of less than 36 milliseconds. Laravel, on the other hand, has difficulty loading and generating data, resulting in higher execution times and memory usage. In terms of memory usage, Pure PHP excels in many aspects, requiring less memory to run functions compared to CodeIgniter and Laravel. CodeIgniter comes in second, with slightly higher but still acceptable memory usage. Laravel exhibits the highest memory usage, often requiring more memory than other systems, especially when handling large amounts of data. Performance test results show that CodeIgniter consistently achieves lower average response times across different numbers of users compared to Pure PHP and Laravel. However, it is worth noting that Laravel performs better as the number of users increases, whereas other systems may show increased response times. When checking the standard deviation, CodeIgniter again outperforms its competitors, showing better consistency in response times, especially when loading. Pure PHP and Laravel have similar results when the number of users is equal to or less than 200 users. However, when stress conditions increase, Laravel shows better response time stability. Regarding throughput, CodeIgniter consistently achieves higher throughput values, indicating its ability to handle more requests per second. Laravel performs slightly better than Pure PHP but struggles to cope with heavier loads compared to CodeIgniter. CodeIgniter emerged as the best performing system in the study, with consistently lower response times, better standard deviations, and higher throughput values. Pure PHP excels in certain functions but faces challenges in data generation. Laravel performs well with a larger number of users but shows higher memory usage and response time under load. Each system has its own strengths and weaknesses, so it is important to consider the specific requirements of academic information systems when selecting the most appropriate framework.

4. Related Work

In the study conducted by Das and Saika (2016), an extensive survey was undertaken to assess various PHP frameworks, including CodeIgniter, Laravel, and Cake PHP. The objective was to compare the distinctive features and advantages offered by each of these frameworks [19]. The analysis led to the conclusion that CodeIgniter, owing to its adherence to the MVC architectural pattern, exhibits enhanced security and user-friendliness when compared to the core PHP. Additionally, Cake PHP was found to be competitive in the context of blog sites, while Laravel showcased the implementation of PHP5 concepts, which greatly facilitated the development process. In a research paper authored by Ripunjit Das and Dr. Lakshmi Prasad Saikia (2016), The focus centered on a comparative analysis between Procedural PHP and the CodeIgniter and Laravel Frameworks [20]. The research methodology involved the creation of three identical simple web blog applications, each executing CRUD operations. Performance comparison was conducted by evaluating the execution time and memory usage of each application. The findings indicated that Laravel outperformed its counterparts, particularly as the scale of the application increased. While Plain PHP demonstrated respectable

performance, it was deemed more suitable for smaller projects. In contrast, CodeIgniter and Laravel were recommended choices for larger-scale projects. Furthermore, in a study carried out by Abutaleb, Tamimi, and Alrawashdeh (2021), the focus was on the evolution of PHP MVC frameworks, with a particular emphasis on frameworks that adhere to MVC design patterns, including CodeIgniter, CakePHP, Symfony, and Laravel [21]. The research incorporated performance assessments using Apache Benchmark, which involved directing requests to a page containing the "Hello World" string with a benchmark configuration of (ab –c 200 –n 50000). After a comprehensive comparative analysis of various aspects, the researchers concluded that Laravel emerged as the optimal choice for deploying next-generation PHP-based web applications.

5. Conclusion

Pure PHP is the lightest system out of three systems that are used in this experiment, it is because it does not have any system cores built in it. We as a developer must build the system from zero unlike other competitors which only need some configurations and then ready to use. Even though it takes a little bit more time in the beginning, it will be easier later as the developer knows exactly what they are developing without the need to open the manual every time like when we are using a framework. Pure PHP proves itself as a good choice as well with good results along the experiment time. It sometimes is head-to-head to CodeIgniter and sometimes heads to head to Laravel as well. Pure PHP outperforms other competitors in the memory used by the system. The difference is not that much with CodeIgniter though, Pure PHP just slightly needs less memory and obviously calls only a few functions, unlike the competitors which are frameworks that usually will have a deeper stack trace.

we conclude that CodeIgniter outperforms Pure PHP in this study case (Academic Information System) regarding the results that it achieved along the experiment time even though Pure PHP in some parts outperforms CodeIgniter, but it is because CodeIgniter has more complex system than Pure PHP and despite being more complex, the gap between Pure PHP and CodeIgniter is very narrow. The distance between the results obtained by PHP and the two frameworks which is not too far apart gives the implication that the use of the framework is still highly recommended with several features embedded in it. Pure PHP can be used if the programmer plans to develop small to medium scale applications or creates his own framework. Meanwhile, the use of the framework can be used on a medium to high level application scale. Using a framework will also help programmers to be able to synergize with each other in developing applications because it is wrapped in the framework ecosystem.

Pure PHP doesn't have to load functions when running certain modules making it a very light-weight choice. This also has the impact of using the least resources when compared to frameworks which are equipped with so many functions in them. Using pure PHP will take up the least number of resources on the server side when compared to frameworks. But we also must be willing not to get so many conveniences in other aspects when using a framework, for example on the security side. Laravel is the choice that requires the most resources when compared to the other two. But Laravel also provides the most features and is currently the most stable PHP framework and still frequently provides the latest updates. Laravel also has the most complete ecosystem if we want to use it on an enterprise scale. Meanwhile, CodeIgniter provides the option of a framework that has quite complete features but is still light on the server side. CodeIgniter remains the right choice for small to medium scale application development.

References

[1] W3techs, 2023. Usage statistics of server-side programming languages for websites. Accessed: Nov. 18, 2023. Available at : https://w3techs.com/technologies/overview/programming_language

[2] Maharani, D., 2017. Perancangan Sistem Informasi Akademik Berbasis Web Pada Sekolah Islam Modern Amanah. *Jurnal Manajemen Informatika Dan Teknik Komputer*, *2*(1), pp.27-32.

[3] Abeysinghe, S., 2009. *PHP team development: easy and effective team work using MVC, agile development, source control, testing, bug tracking, and more*. Packt Publishing Ltd.

[4] Wikipedia, 2023. PHP. Accessed: Nov. 18, 2023. [Online]. Available: https://en.wikipedia.org/wiki/PHP

[5] Sotnik, S., Manakov, V. and Lyashenko, V., 2023. Overview: PHP and MySQL Features for Creating Modern Web Projects. Available at: www.ijeais.org/ijaisr

[6] Welling, L. and Thomson, L., 2003. *PHP and MySQL Web development*. Sams Publishing.

[7] Suroji, S. and Hittalamani, V., 2023. ADDICTION ANALYSIS AND SOLUTIONS USING PHP AND MACHINE LEARNING. *International Research Journal of Modernization in Engineering Technology and Science*, Available at: www.irjmets.com

[8] Jain, T. and Jain, N., 2019, March. Framework for Web application vulnerability discovery and mitigation by customizing rules through ModSecurity. In *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)* (pp. 643-648). IEEE. DOI: https://doi.org/10.1109/SPIN.2019.8711673.

[9] Solanki, N., Shah, D. and Shah, A., 2017. A Survey on different Framework of PHP. *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), 6*(6), pp.155-158.

[10] Joshi, A. and Wanjale, K., 2021. PHP Frameworks in Web Application Development. *International journal of creative research thoughts (IJCRT).*

[11] Nordeen, A., 2020. *Learn CodeIgniter in 24 Hours*. Guru99.

[12] Andre Pratama, 2023. *Laravel Uncover*. Duniailkom.

[13] Leisalu, O., 2009. *Comparative Evaluation of Two PHP Persistence Frameworks* (Doctoral dissertation, University of Tartu).

[14] Apache, 2023. Apache JMeter. Accessed: Dec. 01, 2023. Available at: https://jmeter.apache.org/

[15] Agustin, F., Kurniawan, H., Yusfrizal, Y. and Ummi, K., 2018, August. Comparative analysis of application quality between appserv and xampp webserver using ahp based on iso/iec 25010: 2011. In *2018 6th International Conference on Cyber and IT Service Management (CITSM)* (pp. 1-5). IEEE. DOI: https://doi.org/10.1109/CITSM.2018.8674345.

[16] Deni, D.K. and Ferida, F.Y., 2023. Usability Testing Penggunaan Menu Kartu Hasil Studi Di Website Sistem Informasi Akademik Universitas Teknologi Yogyakarta. *Jurnal Teknologi dan Manajemen Industri Terapan*, *2*(I), pp.41-52. DOI: https://doi.org/10.55826/tmit.v2iI.57.

[17] PHP, 2023. PHP Micro Time. Accessed: Dec. 01, 2023. Available at: https://www.php.net/manual/en/function.microtime.php

[18] Tiwari, V., Upadhyay, S., Goswami, J.K. and Agrawal, S., 2023, April. Analytical Evaluation of Web Performance Testing Tools: Apache JMeter and SoapUI. In *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 519-523). IEEE. DOI: https://doi.org/10.1109/CSNT57126.2023.10134699.

[19] Solanki, N., Shah, D. and Shah, A., 2017. A Survey on different Framework of PHP. *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), 6*(6), pp.155-158.

[20] Das, R. and Saikia, L.P., 2016. Comparison of procedural php with codeigniter and laravel framework. *International Journal of Current Trends in Engineering & Research*, *2*(6), pp.42-48.

[21] Abutaleb, H., Tamimi, A. and Alrawashdeh, T., 2021, July. Empirical Study of Most Popular PHP Framework. In *2021 International Conference on Information Technology (ICIT)* (pp. 608-611). IEEE. DOI: https://doi.org/10.1109/ICIT52682.2021.9491679.