COMPARISON BETWEEN SYMFONY, ASP.NET MVC, AND NODE.JS EXPRESS FOR WEB DEVELOPMENT

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Xiaoli Mao

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2018

Fargo, North Dakota

North Dakota State University
Graduate School

Title

COMPARISON BETWEEN SYMFONY, ASP.NET MVC, AND NODE.JS EXPRESS FOR WEB DEVELOPMENT

By

Xiaoli Mao

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Jun Kong

Dr. Zhili Gao

Approved:

April 17, 2018
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

Web development technologies have been developing tremendously in recent years. While new technologies such as Node.js come with various frameworks, traditional technologies such as ASP.NET and PHP are also being used with new frameworks to meet the increasing requirements. In this paper, three server-side frameworks, namely ASP.NET MVC 5, Symfony for PHP, and Node.js Express, are compared in terms of development, performance, and other aspects. For the development comparison, a hospital information management system was developed and it was found that Symfony is the most friendly to new developers. For the performance comparison, benchmark testing was used with two scenarios and it was discovered that ASP.NET MVC 5 has the best performance in a Windows environment. Finally, the security, support, and industry uses were compared, and it was revealed that while Node.js Express has the most support, the ASP.NET MVC 5 framework is the most widely used for enterprise-level websites.

ACKNOWLEDGEMENTS

I would like to thank my Advisor Dr. Simone Ludwig for her constant encouragements and patience on me. I would also like to thank my former Advisor Dr. Wei Jin for her inspiration. Most importantly, I would like to thank my family, thank you for your endless love and support.

DEDICATION

To Karena.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
ASP	Application Service Provider
DQL	Database Query Language
EF	Entity Framework
HTTP	Hyper Text Transfer Protocol
HQL	Hibernate Query Language
IIS	Internet Information Services
I/O	Input/Output
IDE	Integrated Development Environment
JSP	Java Server Pages
LAMP	Linux, Apache, MySQL, PHP
LINQ	Language Integrated Query
MVC	Model-View-Controller
ORM	Object-Relational Mapping
PHP	Personal Home Page
URL	Uniform Resource Locator

1. INTRODUCTION

1.1. Introduction

Web applications play a very important role in people's life and has been the most important media to spread information. With the popularity of data mining and cloud computing, the new trends of web development in recent years are large scale, high concurrency and vast amount of data processing. Some new technologies on web development have become more popular, especially on the server side, such Python Django and Node.js. In the meanwhile, traditional technologies, such as ASP.NET and PHP, are still playing a huge role in web development. Figure 1 shows that the two most used web application server-side languages are still PHP and ASP.NET.

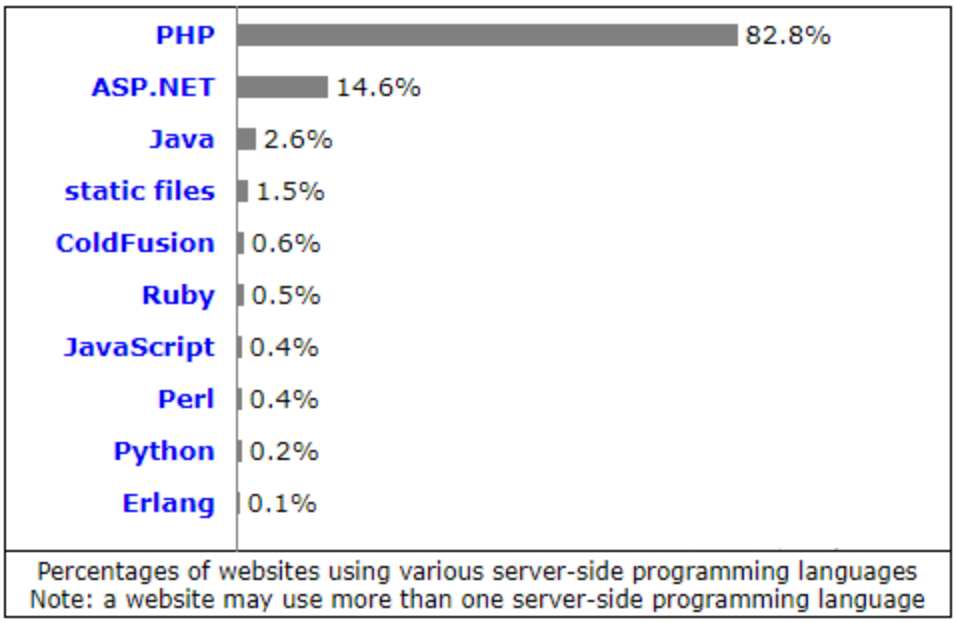

Figure 1. Percentage of Server-Side Language Usage [1]

1.2. Motivation and Contribution

For both ASP.NET and PHP, many new and powerful frameworks are being used in web development. The use of frameworks can significantly reduce repeated code in a project and also make the process easier and short-lasted. The three most popular ASP.NET frameworks are Web Forms, MVC, and Web Pages. Some other popular PHP frameworks include Zend, CakePHP, Symfony, and Laravel.

In this project, three popular frameworks, namely ASP.NET MVC 5, Symfony for PHP, and Node.js Express, are selected for comparison in terms of their development, performance, and other aspects. For the development comparison, a website of hospital information management system was developed with these three frameworks to compare the three server side technologies. For the performance comparison, benchmark testing was used to compare the performance of these three frameworks. In other comparisons, several aspects including security, support, and usage in industry were compared between these three frameworks. The conclusions made from the comparison of these three server-side technologies will help web developers to select more appropriate frameworks for different projects.

1.3. Organization of Paper

The rest of this paper is organized as follows: Chapter 2 discusses related knowledge, Chapter 3 describes the hospital information management system, Chapter 4 details the comparisons and analyzes the results, and Chapter 5 draws conclusions of the paper and briefs future work.

2. RELATED KNOWLEDGE

2.1. Introduction

2.1.1. ASP.NET MVC

ASP.NET MVC 5 was released in 2013. The MVC (Model-View-Controller) pattern, originally named Thing-Model-View-Editor, was first introduced in the 1970s for graphical user interface, and has been widely adopted for web development. The brief explanation of MVC is introduced as follows [2]:

- Model: Application data and behavior
- View: the HTML markup that display to the user
- Controller: manages the relationship between the View and the Model.

2.1.2. Symfony

Symfony is a high-performance PHP framework for web development and an open source project. The first generation of Symfony was released in 2005. This project used the latest version 3.3. Symfony uses the MVC pattern to build website as well. It can also use third party libraries to enhance its performance, which include Doctrine for database management and Twig for front-end development.

2.1.3. Node.js Express

JavaScript, a popular language in recent years, has been widely used in web development, mostly in the front-end. Node.js Express was released as a server-side language around 2009 with the purpose of providing an event-driven, non-blocking I/O model and also a lightweight and efficient framework. Node.js is primarily written in C, C++, and JavaScript.

2.2. Related Work

There have been numerous studies related to the evaluation and analyses of web development and web server technologies. While some of them compared the web development frameworks in several aspects, other studies focused on comparing the web server performance.

Titchkosky et al. [3] evaluated the impact of three different dynamic content technologies (Perl, PHP, and Java) on web server performance. The authors tested the achievable performance of static content workload, dynamic content generation workload without database, and dynamic content generation workload with database, respectively. The results showed that the overheads of dynamic content generation reduced the peak request rate supported by a web server up to a factor of eight. In addition, the authors found that the Java server technologies typically outperformed both Perl and PHP in dynamic content generation.

Swales et al. [4] compared three dynamic web programming technologies (JSP, ASP, and ASP.NET) in terms of their performance in multimedia distribution. The authors used three dynamic web technologies to build a testing application to distribute multiple images from an Oracle 9i database. The results overturned previous research that stated that JSP did not outperform ASP. In addition, the results showed that JSP and ASP both significantly outperform ASP.NET which was newly deployed at that time.

Different types of frameworks come out in recent years, especially for open source projects such as PHP frameworks and Python frameworks. Prokofyeva and Boltunova [5] analyzed various PHP frameworks including CakePHP2, CodeIgniter, Symfony2, Yii, and PhalconPHP, and compared them using several criteria. Besides the general comparison of those frameworks, the authors also provided a simple performance test between these frameworks. The results showed that PhalconPHP and Symfony2 had very similar routing

capabilities, but Symfony2 provided more choices of configuration file formats. In addition, the Phalcon template was friendlier to programmers who were new to template engines, and the performance of Phalcon significantly exceeded Symfony2.

New web technologies such as Python and Node.js have become more popular. Lei and Ma [6] compared and evaluated the performance of web development technologies in PHP, Python, and Node.js. The authors used benchmark tests and scenario tests, and the results showed that Node.js was able to handle many more requests in a certain time than PHP and Python. This paper clearly concluded that Node.js was an ideal fit for I/O intensive websites due to its lightweight property and efficient. While PHP could be used in small and middle scale web applications, Python-web was very suitable for large web architectures and was also very developer-friendly.

Crawford and Hussain [7] compared four leading server-side scripting technologies, namely PHP, Django, Ruby on Rails, and Node.js, in five aspects. The five comparison attributes were ease of getting started, availability of help and support, popularity, availability of development tools and packages, and performance. The results showed that PHP was the simplest technology to get started and in setting up a local server environment. Node.js was proved to be more difficult to beginners since it had some difficult syntax. For help and support, it was also easy to get help on PHP from integrated guides and documentation. For the popularity, PHP was absolutely the most popular server-side technology because it had been over ten years. However, the growth trend of Node.js was better than the other technologies. For development tools and package management systems, the research showed that Node.js provided almost three times the amount of packages compared to the other technologies. The results showed that the latest version of PHP had the most support and might take some work to

utilize Microsoft SQL Server. To compare the performance of each web script technologies, the authors tested Vapor (Swift), Ruby on Rails (Ruby), Laravel (PHP), Lumen (PHP), Express (JavaScript), Django (Python), Flask (Python), Spring (Java), Nancy (C#), and Go. It turned out that Go and Express had the most efficient performance. The results of the comparison of four most popular frameworks is shown in Figure 2.

	Node.js	**PHP**	**Django**	**Rails**
Getting Started	Very good	Excellent	Very good	Good
Help And Support	Good	Excellent	Excellent	Excellent
Popularity	Very Good	Excellent	Fair	Very Good
Development Tools and Package Management Systems	Excellent	Good	Fair	Very Good
Environments	Excellent	Poor	Good	Good
Integrations with Databases	Excellent	Very Good	Fair	Very Good
Performance	Excellent	Fair	Very Good	Fair

Figure 2. Comparison of Four Most Popular Frameworks [7]

3. HOSPITAL WEBSITE

The website used in this project is designed to be used by a hospital to manage patient appointments.

3.1. Detailed Design

In this project, there are two types of anticipated users, which are front desk staff and doctors. The main purpose of the front desk staff is to manage appointments, including search, check in, edit, and cancel. The pipelines of the front desk include checking in an appointment from the upcoming appointments list or search result, and canceling and editing an appointment using the edit button in the upcoming appointments list or from the search result. The detailed design of the front desk is shown in Figure 3.

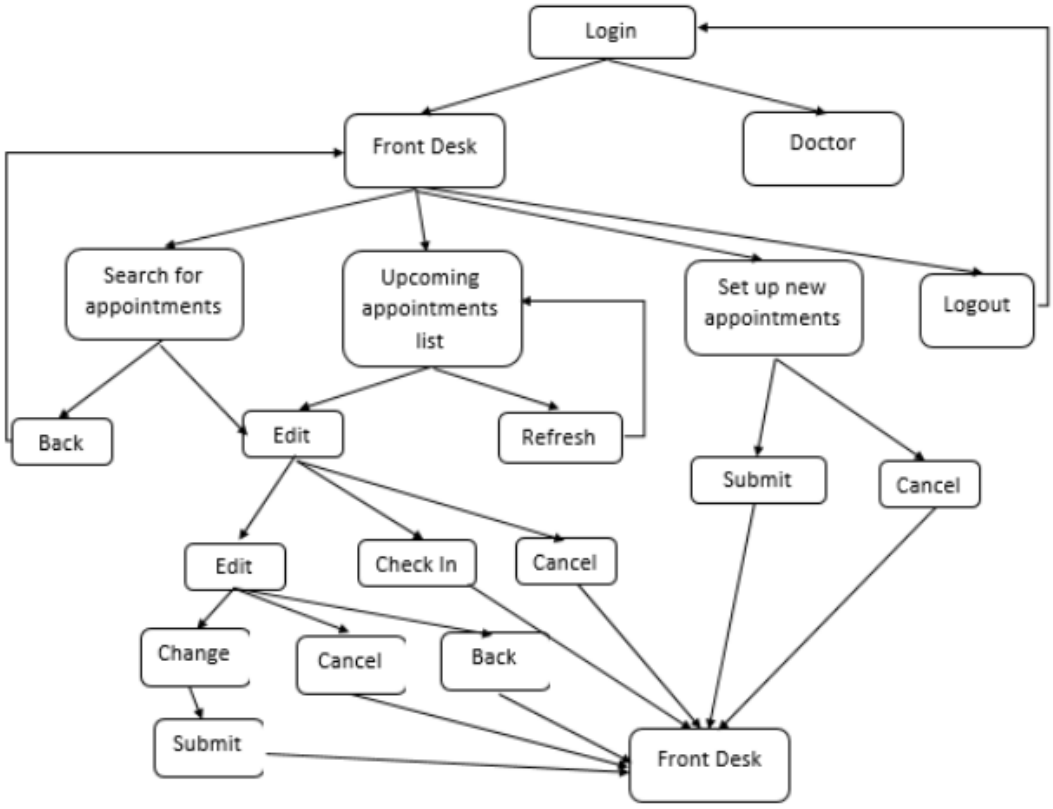

Figure 3. Functionality of Front Desk Staff Users

The detailed design of doctors' functionality is shown in Figure 4. The pipelines of a doctor include searching for the patient list, writing comments for appointments and viewing patient history using the edit button in the upcoming appointments list, and searching for schedule by date.

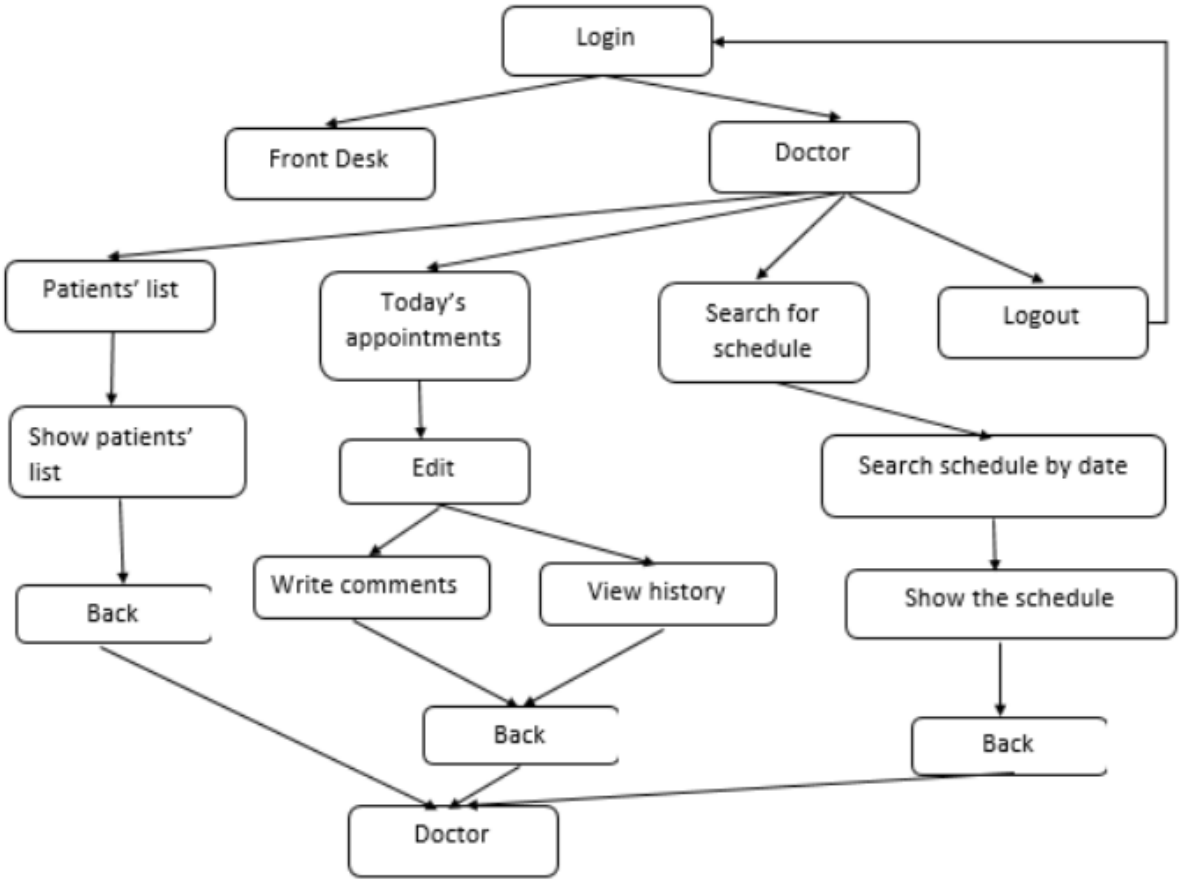

Figure 4. Functionality of Doctor Users

3.2. Database

The purpose of the database is to store appointment information as well as staff and doctor accounts. In this project, two unrelated tables are used to store this information. In the staff and doctor accounts table, four properties are used: ID (Primary key), User Name, Password, and Department, as displayed in Figure 5.

ID	uname	pword	dept
1	user01	USER01	checkin
2	user02	USER02	checkin
3	user03	USER03	checkin
4	doctor01	DOCTOR01	Doctor

Figure 5. Database: Table of Staff Information

In the appointment table, eleven properties are used: ID (Primary key), First Name, Last Name, Date of Birth, Gender, Appointment Date, Appointment Time, Problem, Status, and Comments, as presented in Figure 6.

ID	LName	FName	DOB	AptDate	AptTime	Gender	Doctor	Problem	Status	comments
2	Last002	First002	1992-02-12	2017-04-19	08:30:00	Male	Doctor02	problem description	Checked	problem solved!
9	Last005	First005	1992-10-10	2017-04-18	16:30:00	Male	doctor01	problem 2	Checked	solved
10	Last005	First005	1992-10-10	2017-05-01	16:45:00	Male	Doctor01	problem 2	Not checked	
11	Last004	First004	1995-12-12	2017-04-19	16:00:00	Female	Doctor02	this is a problem	Checked	*NULL*
6	Last006	First006	1999-03-10	2017-04-19	15:00:00	Female	Doctor02	here give the problem	Checked	*NULL*

Figure 6. Database: Table of Appointments Information

3.3. Implementation

The project was implemented in ASP.NET MVC 5, Symfony 3.3 and Node.js Express separately. This section explains the main pages.

3.3.1. Login Page

In the Login Page, as exhibited in Figure 7, a correct combination of username and password will direct to either the front desk staff page or the doctor page. If the combination is incorrect, an alert message will show up.

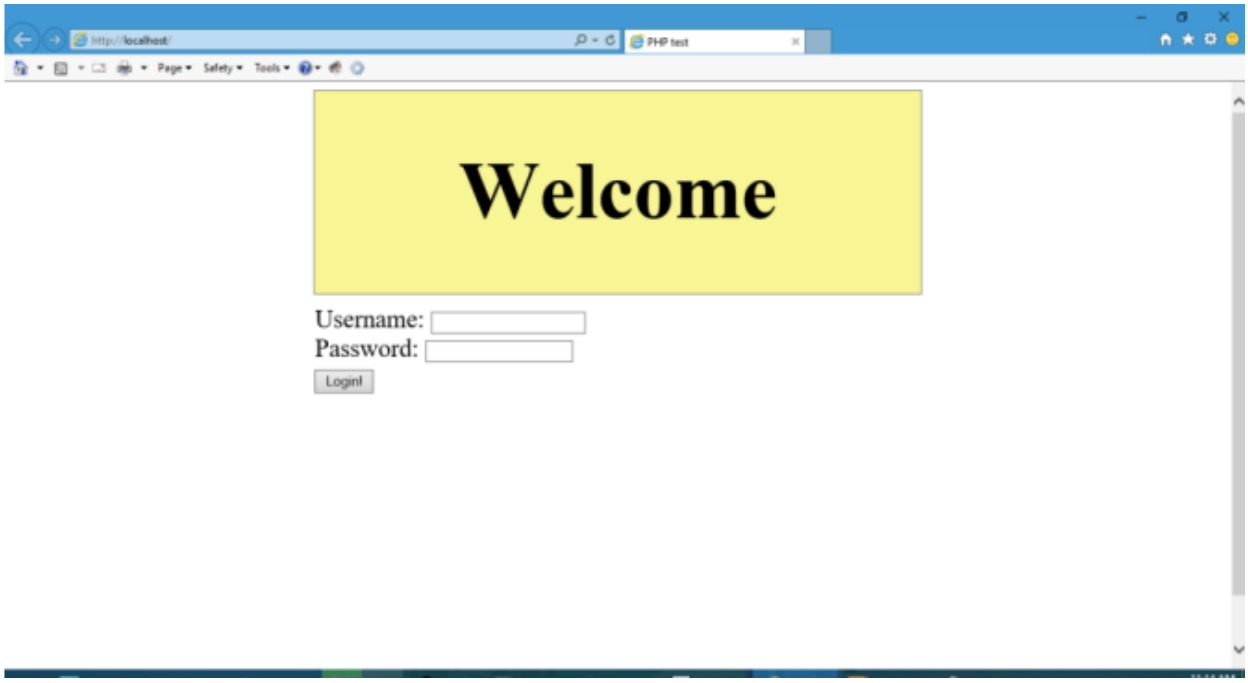

Figure 7. Login Page

3.3.2. Front Desk Page

The Front Desk Page is used by the front desk staff to set up, search, check-in, and edit appointments. The toolbar includes three buttons: set up, search and logout. The upcoming appointments table shows the remaining appointments in the current day. Each appointment in the upcoming appointments table can be modified by clicking the edit button, as presented in Figure 8.

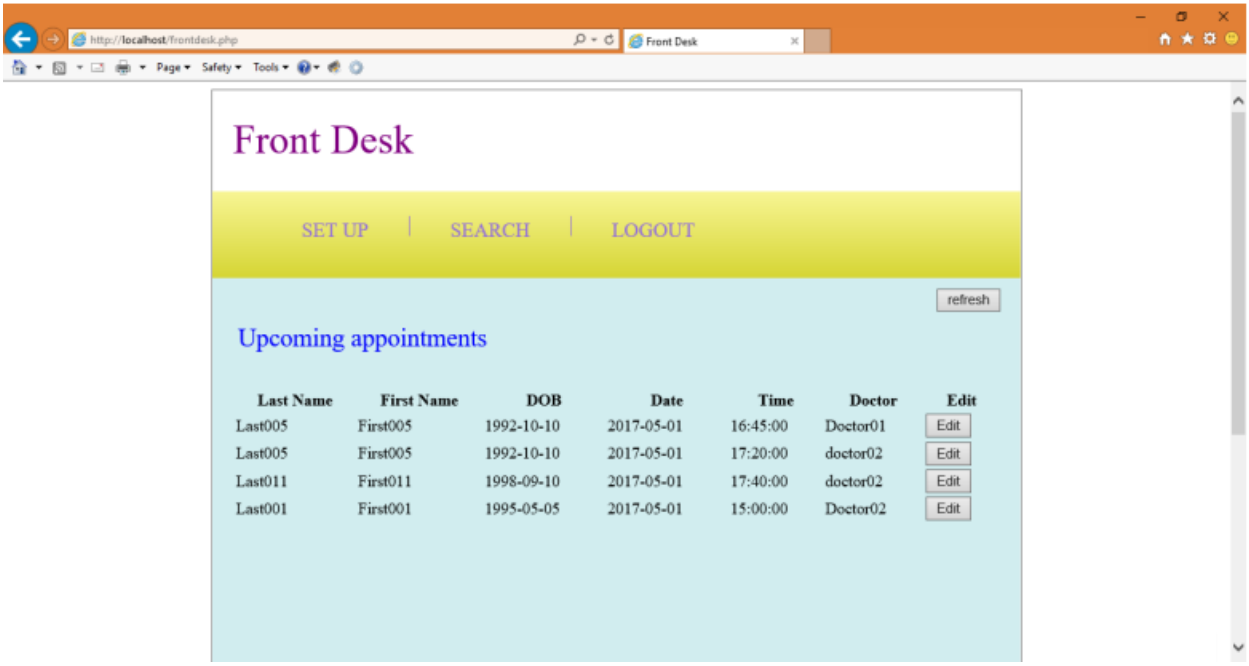

Figure 8. Front Desk Page

3.3.3. Edit an Appointment Page

In the Edit an Appointment Page, appointments can be edited. Updated appointments will be stored in the database after submission, and the page will go back to the front desk page. When clicking the cancel button, this appointment will be canceled and removed from the database. The close button will ignore any change of the appointment and go back to the front desk page, as shown in Figure 9.

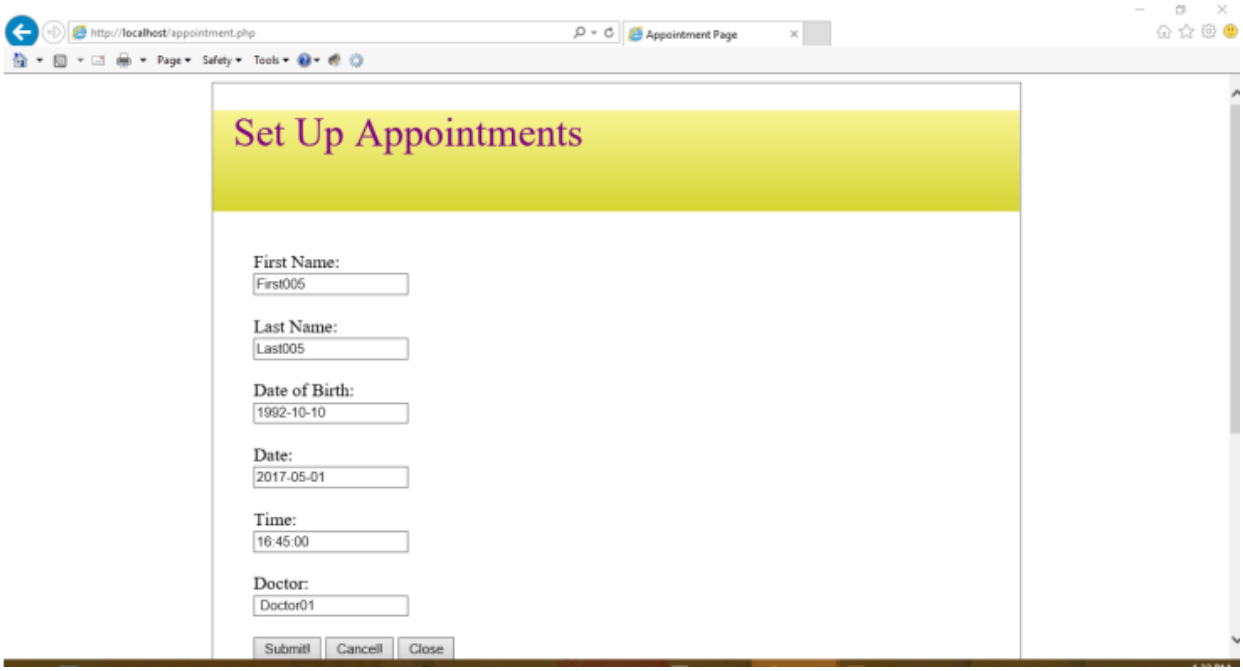

Figure 9. Edit an Appointment

3.3.4. Doctor's Page

The Doctor's Page will allow a doctor to view the patient list and medical history, check schedule, and write comments. The toolbar includes three buttons. The table below the toolbar shows a list of patient appointments in the current day. The edit button after each appointment will direct to the Doctor's Edit Page, as illustrated in Figure 10.

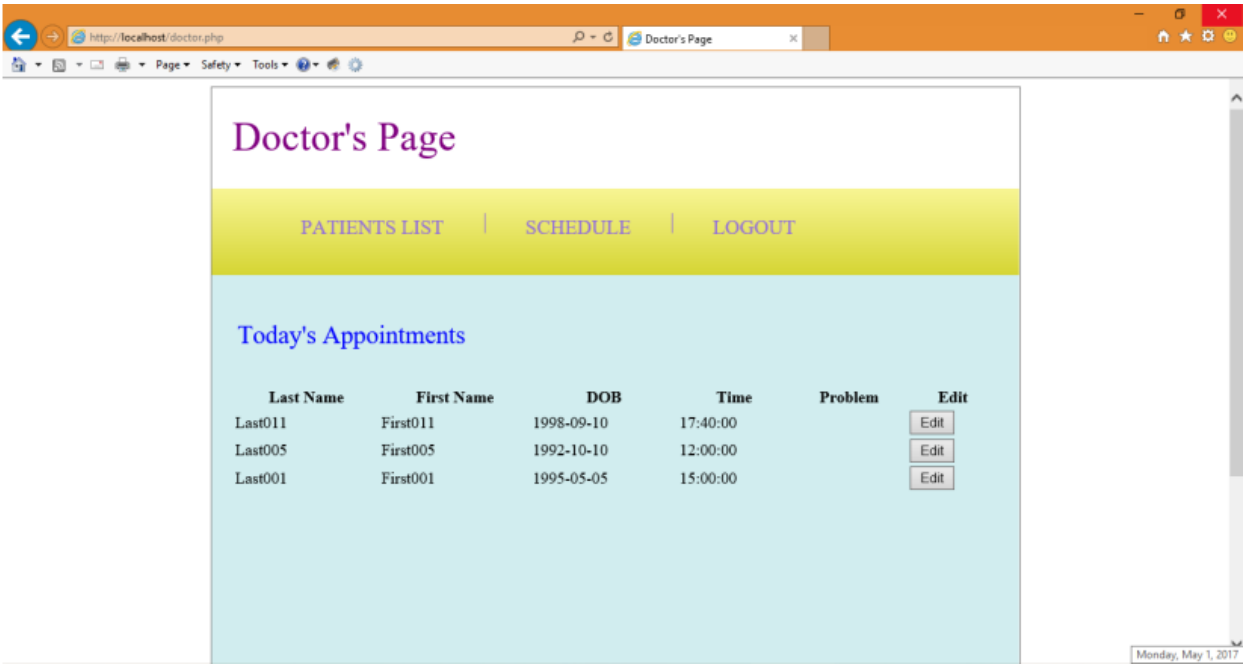

Figure 10. Doctor's Page

3.3.5. Doctor's Edit Page

After clicking the edit button of today's appointments on the Doctor's Page, the Doctor's Edit Page will display. In this page, the doctor can view the history of this patient and write comments of this appointment, as displayed in Figure 11.

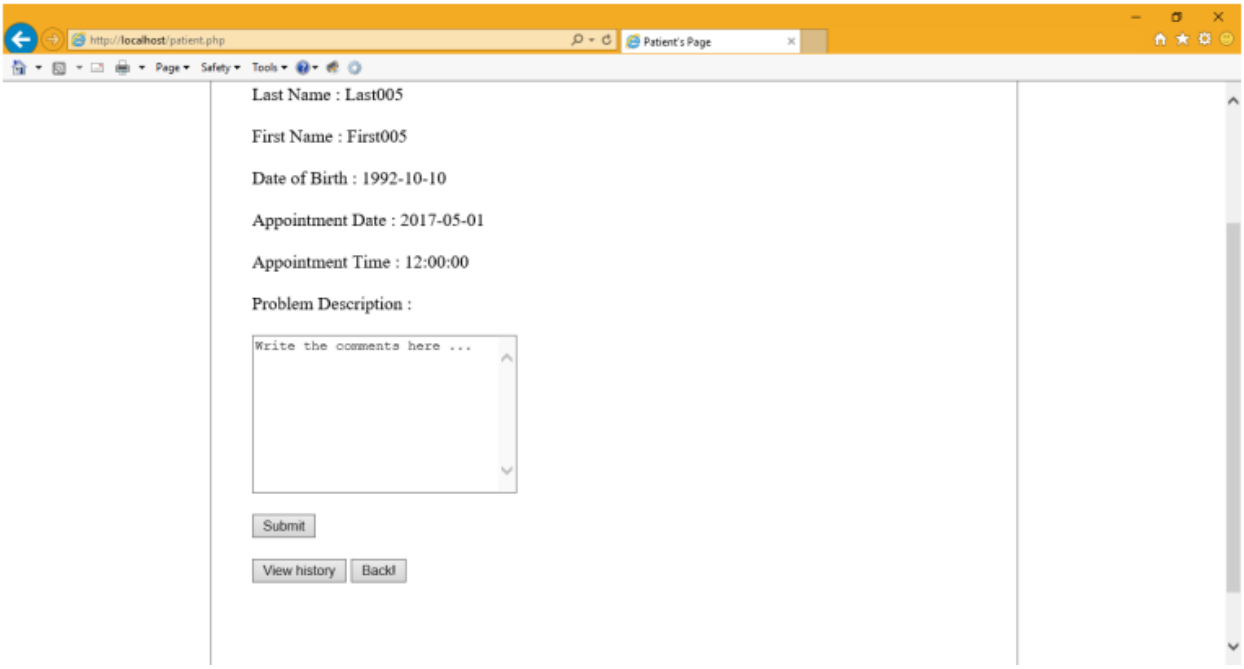

Figure 11. Doctor's Edit Page

4. COMPARISON

Choosing the most suitable technology to build the website is always a challenging task for developers in a company. Many aspects should be considered before selecting the technology and framework, such as cost, development cycle, employee skill level, performance, etc. In this paper, ASP.NET MVC 5, Symfony, and Node.js Express will be compared in three aspects, namely development comparison, performance comparison, and other comparisons.

4.1. Development Comparison

4.1.1. Working Environment

4.1.1.1. ASP.NET MVC

For ASP.NET MVC, the best working environment is in Windows and Visual Studio, of which the latest version is 2017. Once Visual Studio 2013 or newer has been installed, it is ready to start the development work. The web server used in ASP.NET MVC, IIS (Internet Information Services), has already been included in Visual Studio. Therefore, setting up the working environment for ASP.NET MVC is very simple.

4.1.1.2. Symfony

For Symfony, although it can be used in Windows, Linux, and Mac OS, the primary working environment for Symfony is LAMP (Linux, Apache, MySQL, and PHP). Therefore, to set up the working environment of Symfony, Apache, MySQL, and PHP should be installed before installing Symfony. As a result, the developer needs to know some basic knowledge about Linux.

4.1.1.3. Node.js Express

Node.js is also popular in Windows, Linux, and Mac OS. Since one feature of Node.js is lightweight due to npm, a package manager, installing Node.js is quite simple for both Windows and Linux, similarly to the installation of other regular software.

4.1.2. Developing Language

4.1.2.1. ASP.NET MVC

ASP.NET MVC provides a lot of features to make developers' work easier. Some of those features are explained with more details below.

For the front-end development, ASP.NET MVC uses HTTP Helper to assist the developers to build the front-end coding. For example, forms are a very typical function in a web application to collect or display information in the database, and therefore building forms is also a very common task in web development. The HTTP Helper makes the code like the natural language and thus more readable. Figure 12 demonstrates the code of building a form.

```
@using (Html.BeginForm("EditAction","FrontDesk",FormMethod.Post))
{
    @Html.HiddenFor(model => model.ID)
    <table>
        <tr>
            <td>@Html.DisplayNameFor(model => model.patient_lastName)</td>
            <td>@Html.EditorFor(model => model.patient_lastName)</td>
        </tr>
```

Figure 12. HTTP Helper Example

A form in a web page that relates to a table in the database to show and collect information is a frequently used scenario in web development. With the help of scaffolding, a Controller can be easily built with actions such as Index, Edit, Details, and Delete, as well as their view pages for a given database table. The selection of scaffolding controllers is shown in Figure 13.

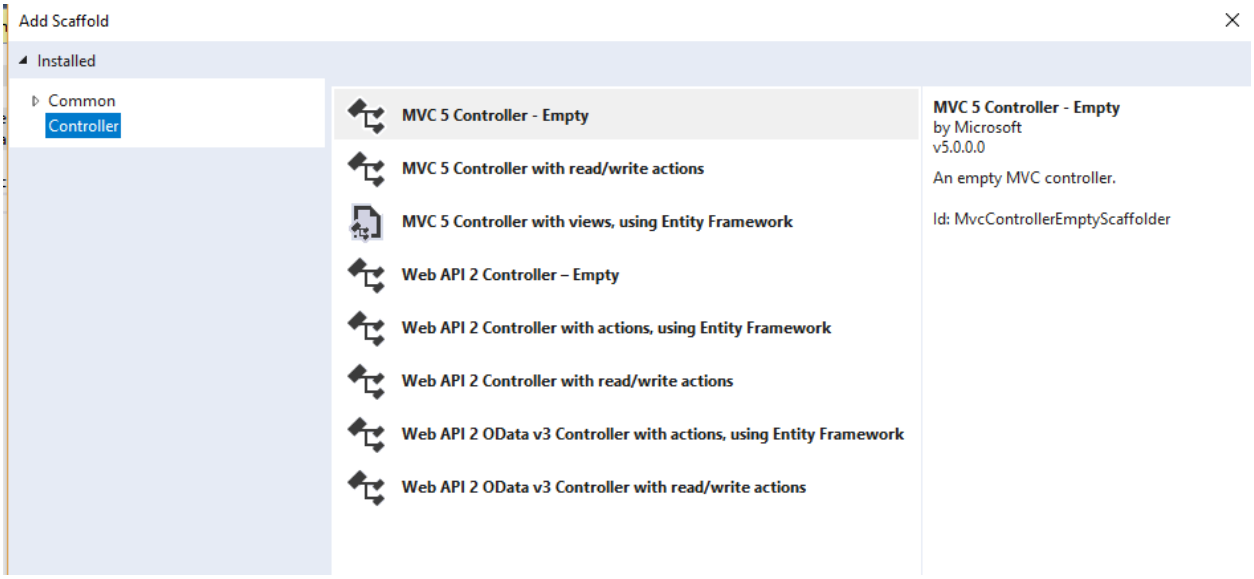

Figure 13. Scaffolding Example

4.1.2.2. Symfony

Twig is used in Symfony front-end development to help developers make the code more conviniently and readable. Figure 14 illustrates that the table will show the information obtained from the controller, in a list.

```
<table class="table table-bordered">
  <tr>
    <th>Last Name</th>
    <th>First Name</th>
    <th>DOB</th>
    <th>Date</th>
    <th>Time</th>
    <th>Doctor</th>
    <th>Check In</th>
    <th>Edit</th>
  </tr>
  <tr>
     {% for apt in apts %}
     <td>{{ apt.patientLastname }}</td>
     <td>{{ apt.patientFirstname }}</td>
     <td>{{ apt.patientDob|date('Y-m-d') }}</td>
     <td>{{ apt.aptDate|date('Y-m-d') }}</td>
     <td>{{ apt.aptTime|date('H:i:s') }}</td>
     <td>{{ apt.doctorLastname }}</td>
     <td>
         <a href="{{ path('checkin', {'id': apt.id}) }}" class="btn btn-primary btn-sm square">Check In</a
     </td>
     <td>
         <a href="{{ path('edit', {'id': apt.id }) }}" class="btn btn-primary btn-sm square" role="button"
     </td>
     </tr>
     {% endfor %}
  </tr>
</table>
```

Figure 14. Twig Example

Figure 15 and Figure 16 demonstrate another example of form. In this project, the most frequent actions are search, modify, and create new appointments via forms, and all data in the forms will communicate with the appointments table in the database. Therefore, a form format can be built based on a given table in the database in Symfony.

```php
class AppointmentType extends AbstractType
{
    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('patientLastname')
                ->add('patientFirstname')
                ->add('doctorFirstname')
                ->add('doctorLastname')
                ->add('aptDate')
                ->add('aptTime')
                ->add('desscription')
                ->add('patientDob')
                ->add('status')
                ->add('gender')
                ->add('result');
    }
```

Figure 15. Form Example in Server-side

```
<div id="inner">
   {{ form_start(editApt) }}
   {{ form_widget(editApt) }}
   {{ form_end(editApt) }}
</div>
```

Figure 16. Form Example in Front-side

4.1.2.3. Node.js Express

Node.js uses Express as the front-end language. It uses <% %> to include non-html language and could be more difficult to use for beginners.

4.1.3. Connection to Database

4.1.3.1. ASP.NET MVC

ASP.NET MVC uses a framework to manage the database. The default database framework used in ASP.NET MVC 5 is Entity Framework (EF). EF is an object-relational mapping (ORM) framework that manages the operations between the program and the database. The architecture of ORM is shown in Figure 17.

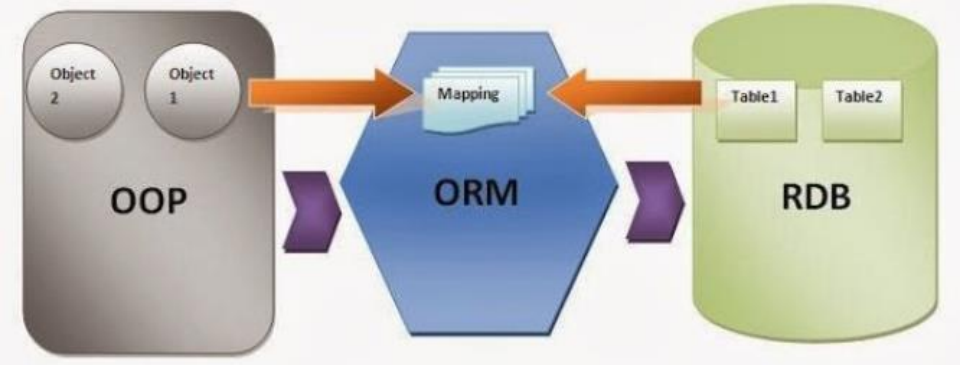

Figure 17. ORM Structure [8]

EF in ASP.NET MVC 5 supports database-first, model-first and code-first styles of development. The default database used in ASP.NET is SQL Server, but it also supports other popular databases. In this project, MySQL is used for all three technologies. After installing support references obtained from MySQL, connecting ASP.MET MVC and MySQL is visualized by following the instructions. When the connection is completed, the database-based model code will be generated automatically with a diagram. The information of a connected database is shown in Figure 18.

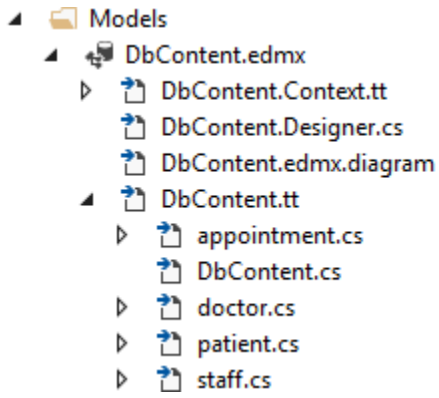

Figure 18. Model Example in ASP.NET MVC

ASP.NET MVC uses Language Integrated Query (LINQ) to query the database. In each table-based files, Data Annotation and Validation can be used to create useable properties, such as DisplayName in the HTML file or to create a new variable, an example of which is shown in Figure 19.

```
public string doctor_firstName { get; set; }
public string doctor_lastName { get; set; }
[DisplayName("Doctor")]
public string doctorName
{
    get
    {
        return doctor_firstName + " " + doctor_lastName;
    }
    set { }
}
[DisplayName("Date")]
public System.DateTime apt_date { get; set; }
```

Figure 19. Data Annotation

4.1.3.2. Symfony

It is very simple to connect to a database in Symfony. The database variables need to be set up first in the parameters.yml file, including database host, port, username, password, etc., and then the database is connected.

Symfony also uses a database framework, Doctrine, to manage the communication with the database. Doctrine is another ORM architecture framework. The query language used in Doctrine is Doctrine Query Language (DQL), which is very similar to Hibernate Query Language (HQL), an object-oriented SQL language which operates on specific objects and their properties instead of tables and columns as in the original SQL language.

4.1.3.3. Node.js Express

Connecting to a database in Node.js is similar to Symfony - setting the properties of a database in a file, such as IP, Username, Password, etc. Node.js does not use a database framework, and therefore the query language in Node.js is the original SQL language which operates directly on tables and columns.

4.1.4. Summary

For a developer, PHP is the easiest programming language to learn while ASP.NET MVC provides features that are most convenient to use. Node.js, as the newest technology among the three, has considerably more advantages over the other two counterparts, such as its fast speed, lightweight with npm, strong computing capabilities, etc.

4.2. Performance Comparison

4.2.1. Introduction

Benchmark test was used to evaluate the performance of ASP.NET, Symfony, and Node.js Express. Considering the different requirements of websites, three scenarios were designed to compare the performance between the three frameworks, which were:

1) Send a string to front-end: this scenario is a basic module in a web server to evaluate the I/O operation of the web application.

2) Calculate the Fibonacci value: this scenario was used to compare the calculation performance of these three frameworks. The formula of n^{th} Fibonacci F(n) is F(n) = F(n-1) + F(n-2).
3) Select items in a database: this scenario was used to compare the operation performance of the three frameworks with a database by retrieving items from a table.

AppachBench (Ab), a tool in Apache, was used to perform the benchmark test. AppachBench is able to make requests in local web server, where the response time is the processing time, which does not include the data transmission time through the Internet.

4.2.2. Hardware

The specifications of the hardware environment used in the test were:

- Operating system: Windows 10 Pro
- Processor: Intel Core i5-3210M 2.5 GHz
- RAM: 8GB
- System type: 64-bit operating system

To ensure the consistency of testing results, the computer was restarted before each framework testing and only the Integrated Development Environment (IDE) and Chrome was run.

4.2.3. Testing Process

In the Benchmark testing, the number of total requests was set at 200, and the number of users was set at 10, 20, 40, 60, 80, and 100, respectively, for each test. The values of mean requests per second and mean time per request in milliseconds (ms) were then compared.

4.2.3.1. Sending a String to Front-end

Table 1 and Table 2 reveals that the performance of ASP.NET was almost 100 times better than Node.js and was over 1,000 times better than Symfony. When the number of total requests was set at 200, the performance of ASP.NET MVC was quite stable, at around 3,200 requests per second. For Node.js Express, the highest number of requests per second was about 505 when the number of users was 40. The performance of Symfony was not as well as the other two frameworks. The bar charts in Figure 20 and Figure 21 show the comparison of sending a string to the front-end between three frameworks. It is noted that the charts are not able to show all results as visible bars due to their low values.

Table 1. Results of Sending a String in Mean Request per Second

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	3199.85	399.98	2.40
20	3199.90	387.86	2.38
40	3200.36	504.92	2.48
60	3199.85	474.05	2.49
80	3199.85	457.12	2.50
100	3199.80	457.12	2.45

Table 2. Results of Sending a String in Mean Time (ms) per Request

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	3.125	25.001	4159.117
20	6.250	51.565	8409.882
40	12.499	79.221	16130.588
60	18.751	126.569	24130.950
80	25.001	175.008	31946.168
100	31.252	218.760	40840.740

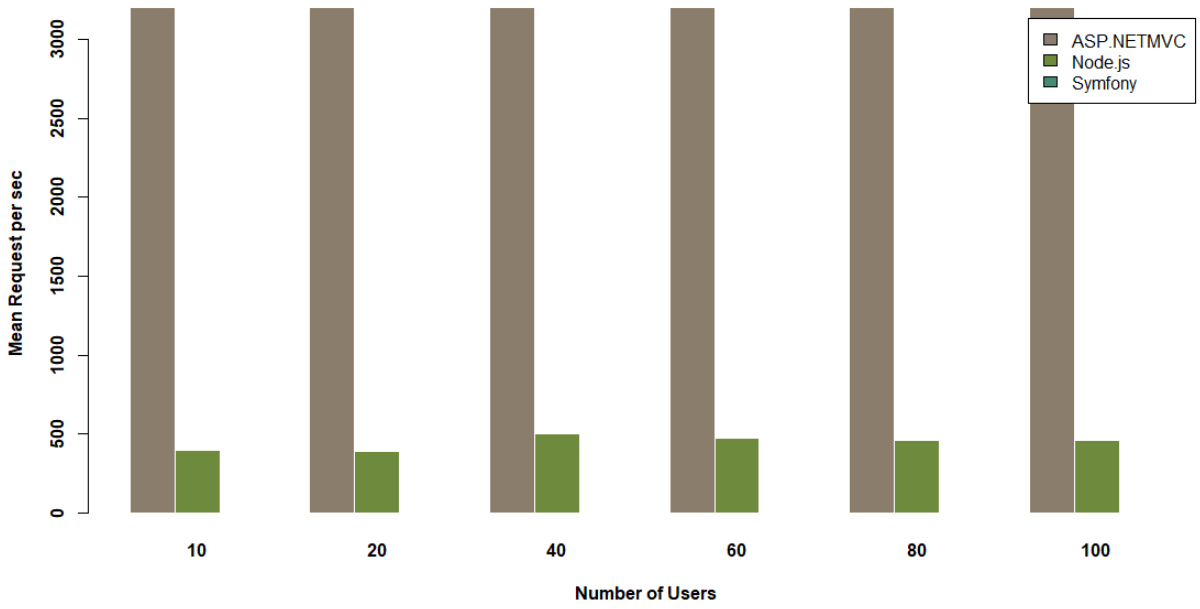

Figure 20. Bar Chart of Sending a String in Mean Request per Second

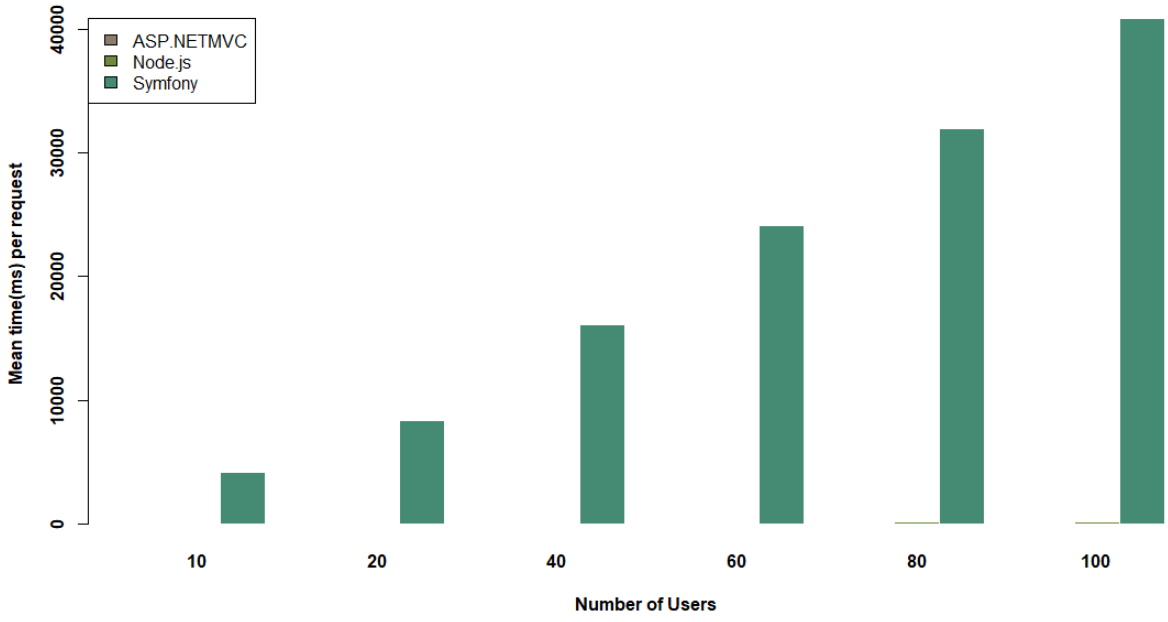

Figure 21. Bar Chart of Sending a String in Mean Time (ms) per Request

4.2.3.2. Calculating the Fibonacci Values

This model is to calculate the value of F(10). The results in mean request per second and mean time per request are displayed in Table 3 and Table 4, and the bar charts of the comparison between three frameworks in calculating F(10) are shown in Figure 22 and Figure 23. It can be seen that ASP.NET MVC still has the best performance in calculating F(10) while Symfony still has the worst performance. In addition, although the performance comparison of calculating F(10) between the three frameworks did not vary much compared to the performance comparison of sending a string, the performance of ASP.NET MVC was less stable in calculating F(10).

Table 3. Results of Calculating F(10) in Mean Requests per Second

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	2133.20	429.29	2.43
20	3199.90	511.98	2.44
40	3199.95	581.79	2.44
60	3199.90	575.98	2.41
80	4266.58	609.50	2.39
100	3199.50	609.50	2.39

Table 4. Results of Calculating F(10) in Mean Time (ms) per Request

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	4.688	20.313	4100.730
20	6.250	39.064	8205.867
40	12.500	68.753	16416.012
60	19.751	104.170	24921.731
80	18.750	131.256	33408.957
100	31.250	164.069	41786.412

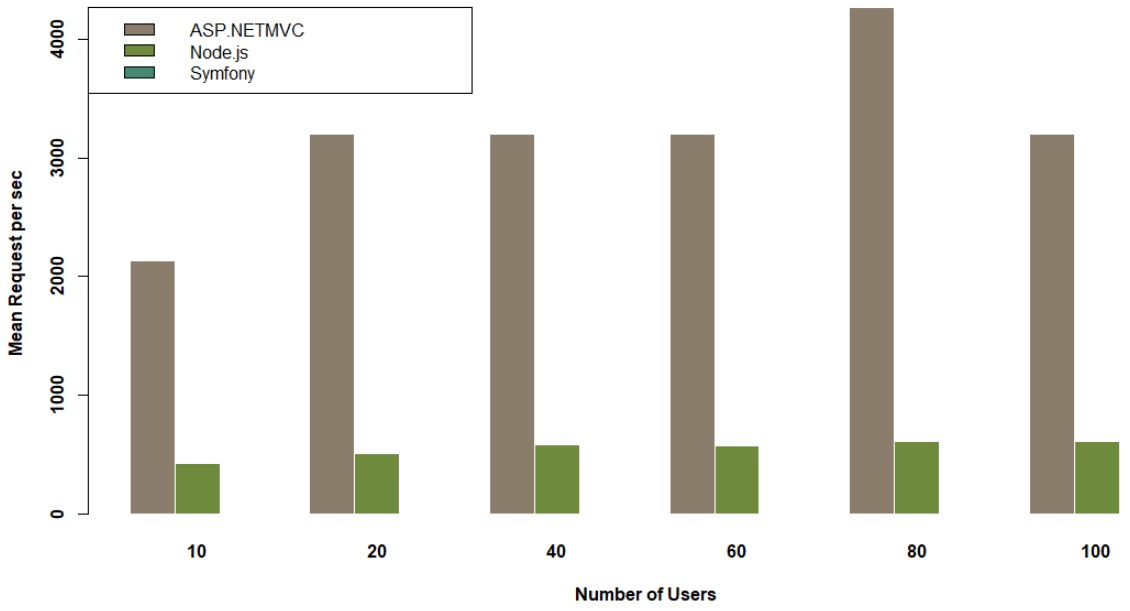

Figure 22. Bar Charts of Calculating F(10) in Mean Requests per Second

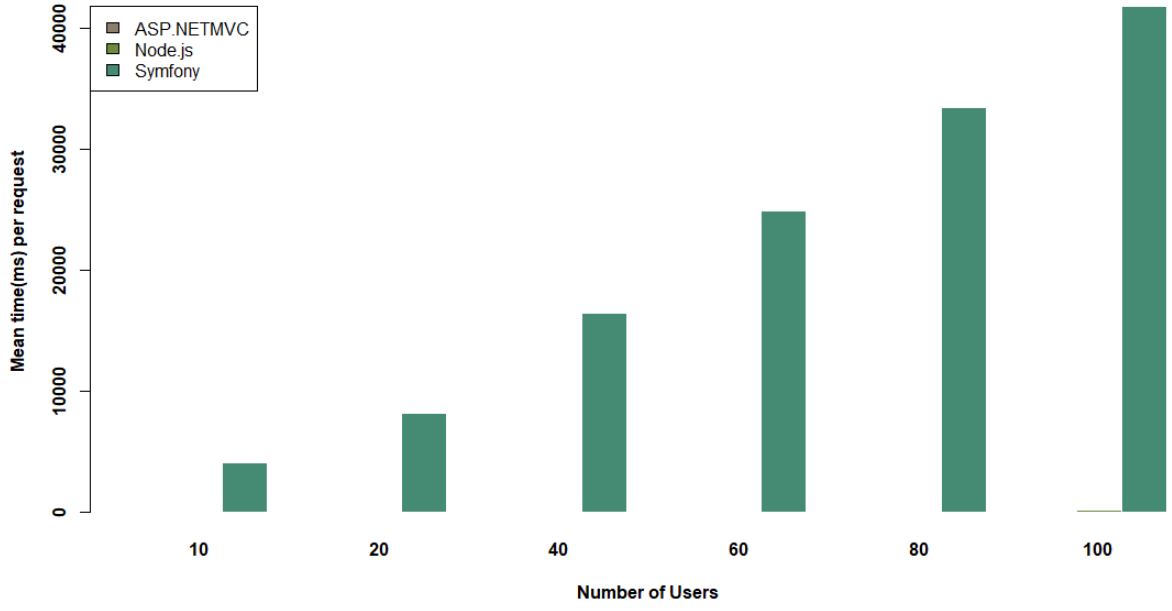

Figure 23. Bar Charts of Calculating F(10) in Mean Time (ms) per Request

In order to test the relationship between the complexity of calculation and the performance of ASP.NET MVC and Node.js, F(30), F(50), and F(100) were also calculated, and the results are shown in Table 5 and Table 6. It can be observed that with the increase of calculation complexity, the performance did not vary much for both ASP.NET MVC and Node.js Express. Figure 24 and Figure 25 visualize such results in mean request per second for ASP.NET MVC 5 and Node.js Express, respectively, when calculating F(10), F(30), F(50), and F(100).

Table 5. Results of F(30), F(50), and F(100) for ASP.Net MVC

User	F(30)		F(50)		F(100)	
	Mean req/sec	Mean ms/req	Mean req/sec	Mean ms/req	Mean req/sec	Mean ms/req
10	2559.90	3.906	3199.95	3.125	3381.59	2.952
20	4266.67	4.688	3199.90	6.250	3331.28	6.004
40	3199.80	15.501	2739.61	14.601	3276.70	12.207
60	4299.58	14.063	3199.85	18.751	3387.65	17.711
80	3199.85	25.001	3199.90	25.001	3223.73	24.816
100	2559.97	39.063	3199.49	31.255	3199.90	31.251

Table 6. Results of F(30), F(50), and F(100) for Node.js Express

User	F(30)		F(50)		F(100)	
	Mean req/sec	Mean ms/req	Mean req/sec	Mean ms/req	Mean req/sec	Mean ms/req
10	573.42	17.439	441.36	22.657	474.05	21.095
20	752.91	26.564	511.98	39.064	492.28	40.627
40	673.65	59.378	511.98	78.129	511.98	78.127
60	673.65	89.066	556.50	107.818	533.32	112.504
80	730.24	109.554	581.79	137.506	581.79	137.506
100	752.91	132.817	639.97	156.258	581.82	171.875

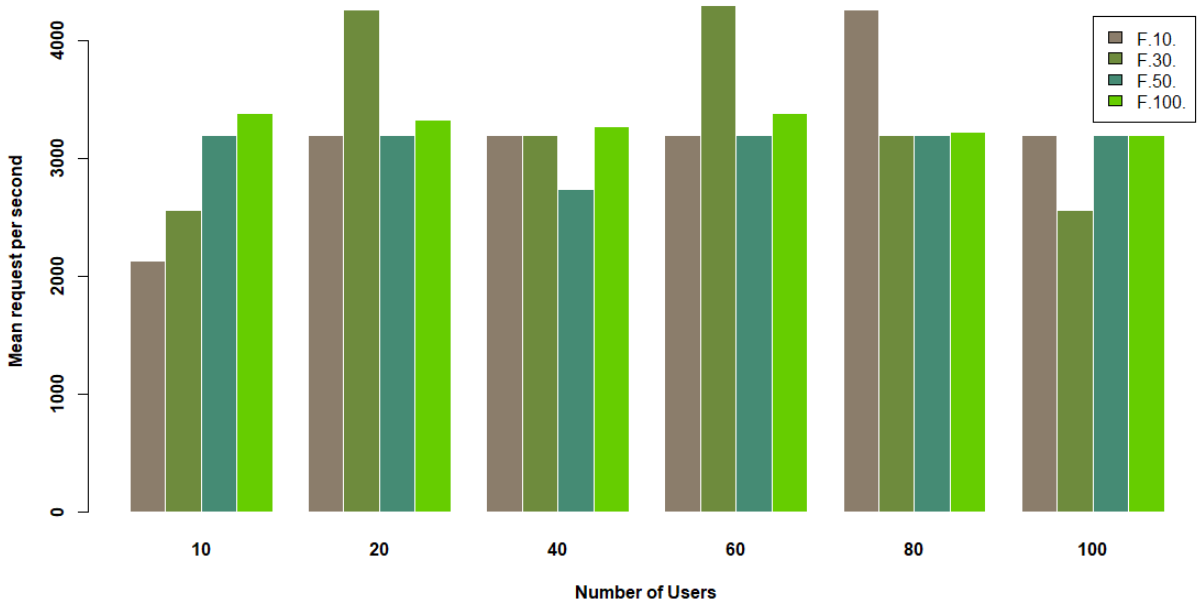

Figure 24. Bar Charts of Fibonacci Values in Mean Request per Second for ASP.NET MVC 5

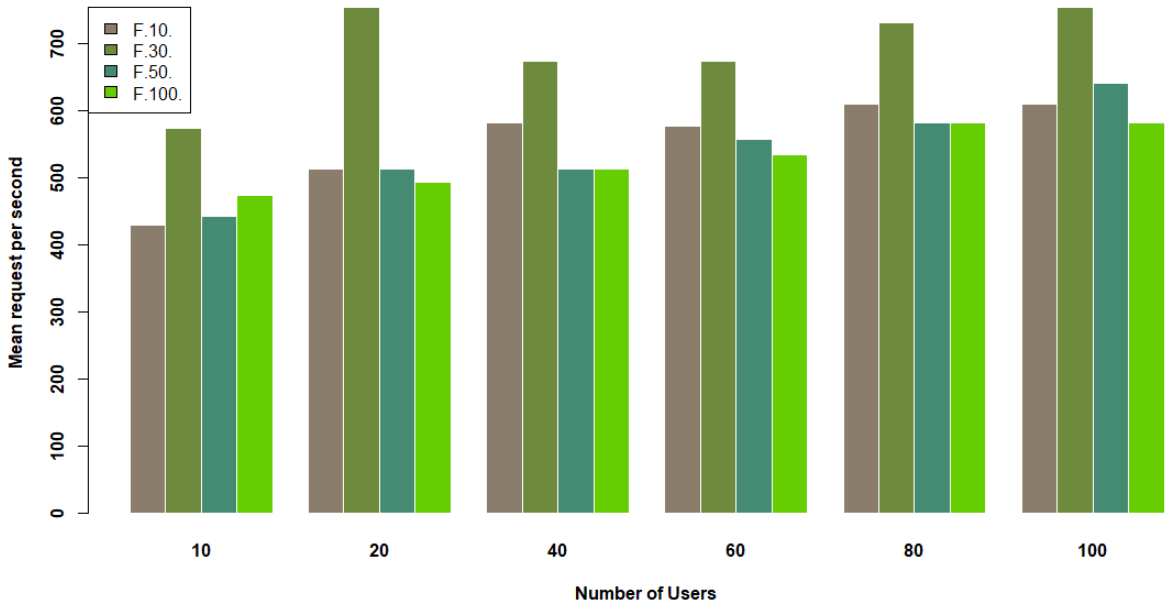

Figure 25. Bar Charts of Fibonacci Values in Mean Request per Second for Node.js Express

4.2.3.3. Selecting Items in Database

Operations with the database is one of the most important tasks in dynamic web applications. In this scenario, selecting 15 rows of data in a 50-row by 5-column table were tested with the three frameworks. The results in Table 7 and Table 8 show that ASP.NET MVC 5 had the best performance on operations with the database. In addition, its performance of selection operations in the database was better than both sending a string to the front-end and calculating a Fibonacci value. The performance of Node.js Express in selection operations in the database did not vary much comparing to the other two scenarios. The performance of Symfony was still the poorest among the three frameworks. Figure 26 and Figure 27 visualize the comparison between the three frameworks. It was also found that the performance of Symfony in selection operations in the database was worse compared to the other two scenarios, dropping from around 2.5 to 0.66 requests per second.

Table 7. Results of Selecting Items in Database in Mean Request per Second

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	6400.20	492.29	0.66
20	4266.48	581.79	0.64
40	4266.39	581.79	0.66
60	4266.30	583.01	0.66
80	6399.80	639.96	0.66
100	4266.58	639.96	0.66

Table 8. Results of Selecting Items in Database in Mean Time (ms) per Request

Number of Users	ASP.NET MVC 5	Node.js Express	Symfony
10	15.620	20.313	15230.632
20	4.688	34.377	31029.856
40	9.376	68.753	60976.155
60	14.064	102.915	91134.397
80	12.500	125.008	120865.168
100	23.438	156.260	151278.730

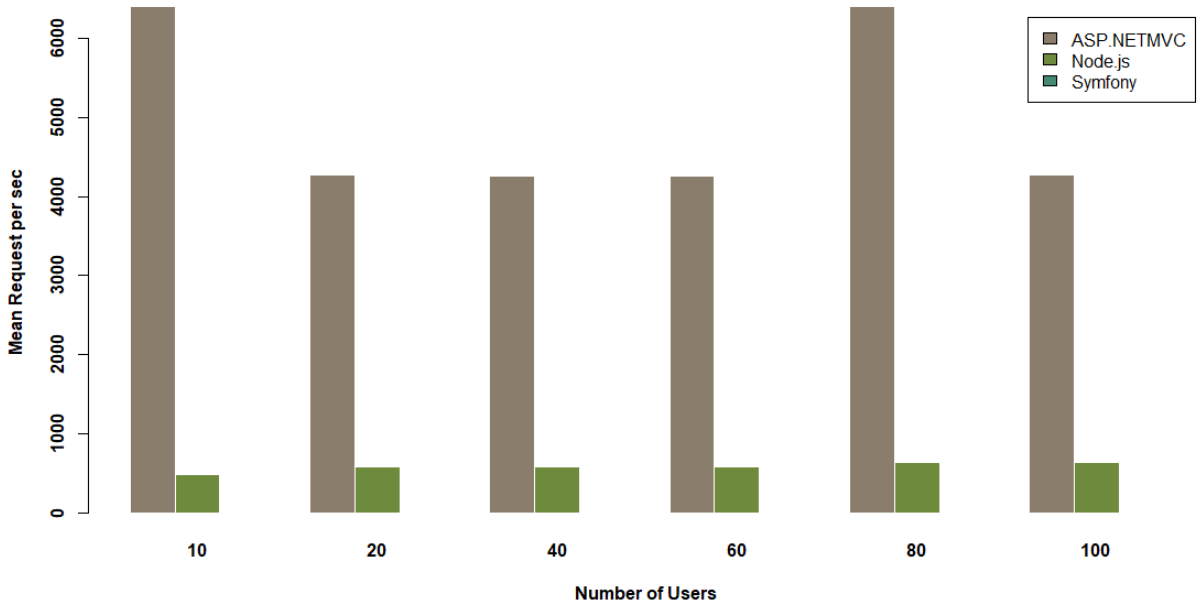

Figure 26. Bar Chart of Selecting Items in Database in Mean Request per Second

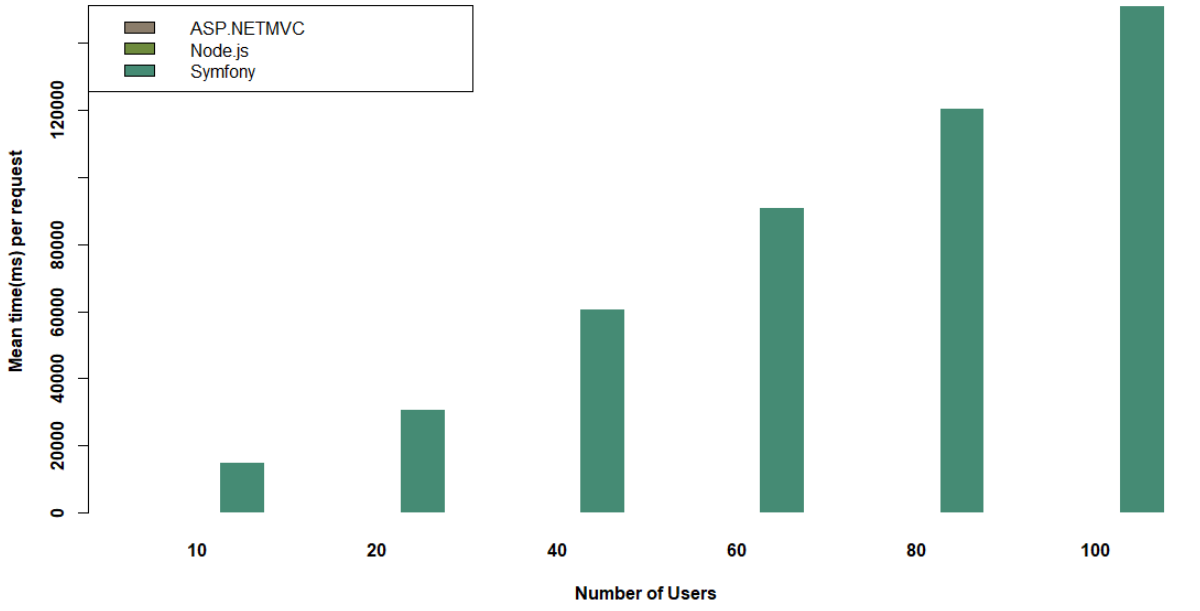

Figure 27. Bar Chart of Selecting Items in Database in Mean Time (ms) per Request

4.2.3.4. Testing in Linux

Since Node.js and Symfony are usually used in a Linux environment, the benchmark tests were also performed in Linux for these two frameworks to test if their performance was the same as in a Windows environment.

To build a Linux system, Ubuntu 16 was installed on a virtual machine, VMware Workstation 14 Player. Generally speaking, the Linux system running in a virtual machine will be assigned with less hardware resources than the Windows system in the physical machine.

When comparing the mean request per second between Linux in Table 9 and Windows in Table 1 and Table 3, it can be concluded that the performance of Symfony in Linux is around four times better than that in Windows in first two scenarios: from between 2.38 and 2.50 to between 8.22 and 9.99 for sending a string to the front-end and from between 2.39 to 2.44 to between 9.12 and 10.65 for calculating F(10). In the selecting items in database scenario, its performance improved dramatically from 0.66 to around 9.50, almost 15 times better. Figure 28 shows the bar chart comparison of sending a string to the front-end for Symfony between Windows and Linux.

Table 9. Symfony Performance in Linux System

User	Sending a String to Front-end		Calculating F(10)		Selecting Items in Database	
	Mean request/sec	Mean ms/request	Mean request/sec	Mean ms/request	Mean request/sec	Mean ms/request
10	9.86	1013.614	9.38	1065.838	8.88	1126.641
20	9.21	2172.775	9.87	2025.807	9.03	2213.729
40	9.49	4212.807	9.12	4386.007	9.70	4124.195
60	9.99	6006.207	10.48	5724.946	9.52	6302.802
80	9.99	8009.036	10.30	7754.323	9.50	8417.177
100	8.22	12159.690	10.65	9385.867	9.95	10046.745

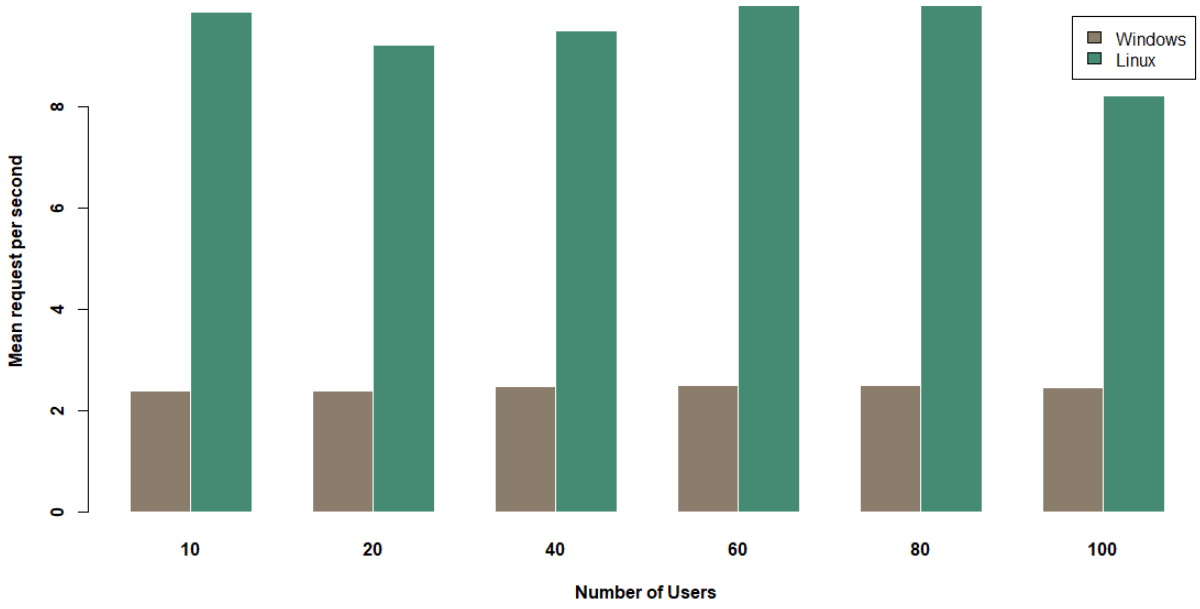

Figure 28. Comparison of Sending a String to Front-end for Symfony between Windows and Linux

Node.js Express was tested for the three scenarios in Linux and it was found that although its performance also improved, the improvement was not as much as that of Symfony. Table 10 shows the performance results of Node.js Express in Linux, and Figure 29 visualizes the comparison of sending a string to the front-end using Node.js Express between Windows and Linux.

Table 10. Node.js Express Performance in Linux

User	Sending a String to Front-end		Calculating F(10)		Selecting Items in Database	
	Mean request/sec	Mean ms/request	Mean request/sec	Mean ms/request	Mean request/sec	Mean ms/request
10	470.42	21.257	756.78	13.214	515.79	19.388
20	545.65	36.654	978.79	20.433	608.42	32.872
40	601.38	66.514	1073.12	37.275	598.40	66.845
60	594.48	100.928	1060.18	56.595	614.03	97.714
80	641.41	124.724	1153.41	69.360	648.88	123.289
100	659.41	151.651	1088.36	91.881	642.54	155.634

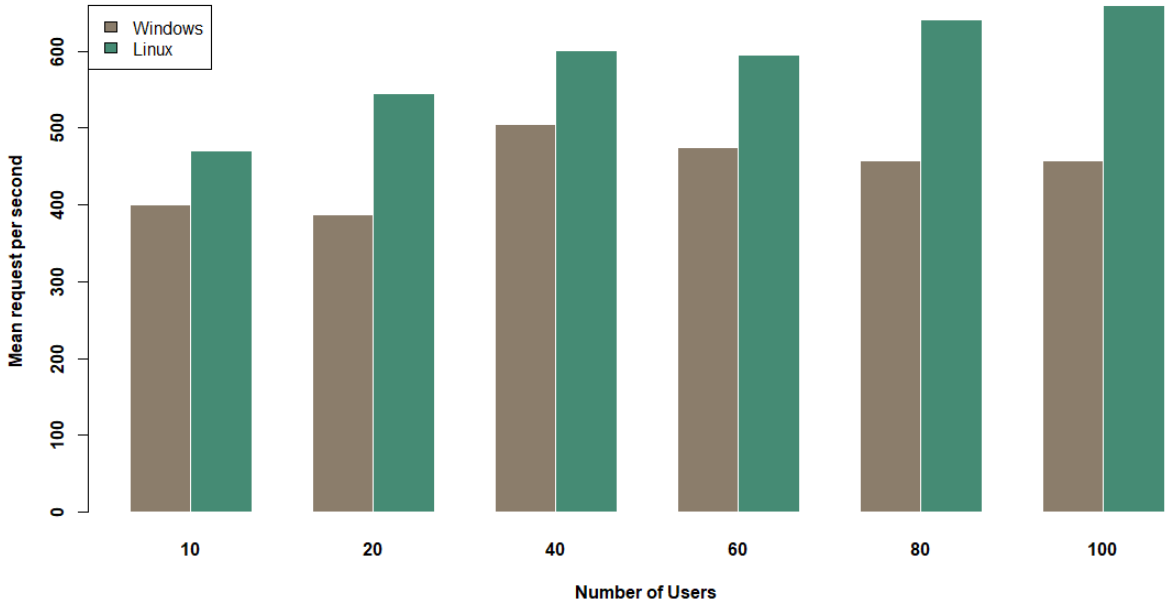

Figure 29. Comparison of Sending a String to Front-end for Node.js between Windows and Linux

4.3. Other Comparisons

4.3.1. Security

4.3.1.1. ASP.NET MVC

Security has always been one of the most important parts for a web portal. ASP.NET MVC framework provides some features to improve the security of websites, such as using the Authorize action filter on a controller to identify if the user is authorized or not [2].

The first security policy in ASP.NET MVC is using the Authorize attribute to require a login. This step can be implemented in Controller actions, as shown in an example in Figure 30.

```
 [Authorize]
public ActionResult Buy(int id)
{
    var album = GetAlbums().Single(a => a.AlbumId == id);

    //Charge the user and ship the album!!!
    return View(album);
}
```

Figure 30. Authorize Attribute in ASP.NET MVC

In addition, if Authorization is set to global, some URLs can still be accessible to non-registered users by using the AllowAnonymous attribute, as shown in an example in Figure 31.

```
// GET: /Account/Login
[AllowAnonymous]
public ActionResult Login(string returnUrl)
{
        ViewBag.ReturnUrl = returnUrl;
        return View();
}
```

Figure 31. AllowAnonymous Attribute in ASP.NET MVC

4.3.1.2. Symfony

The security in Symfony is mainly managed in the security.yml file. The original security.yml file is demonstrated in Figure 32.

```yaml
# To get started with security, check out the documentation:
# https://symfony.com/doc/current/security.html
security:
    encoders:
      AppBundle\Ent\Staff:
        algorithm: bcrypt

    # https://symfony.com/doc/current/security.html#b-configuring-how-users-are-loaded
    providers:
        db_provider:
          entity:
              class: AppBundle:Staff
              property: username

    firewalls:
        # disables authentication for assets and the profiler, adapt it according to your nee
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        main:
            anonymous: ~
            form_login:
              login_path: login
              check_path: login
            # activate different ways to authenticate

            # https://symfony.com/doc/current/security.html#a-configuring-how-your-users-will
            #http_basic: ~

            # https://symfony.com/doc/current/security/form_login_setup.html
            #form_login: ~
```

Figure 32. Security.yml in Symfony

The main job of the security.yml file is to set user Authentication and Authorization. Symfony first uses Firewalls to verify the user (Authentication), and then uses Access Control to check the user's permission to access specific URLs (Authorization), which is illustrated in Figure 33 [9].

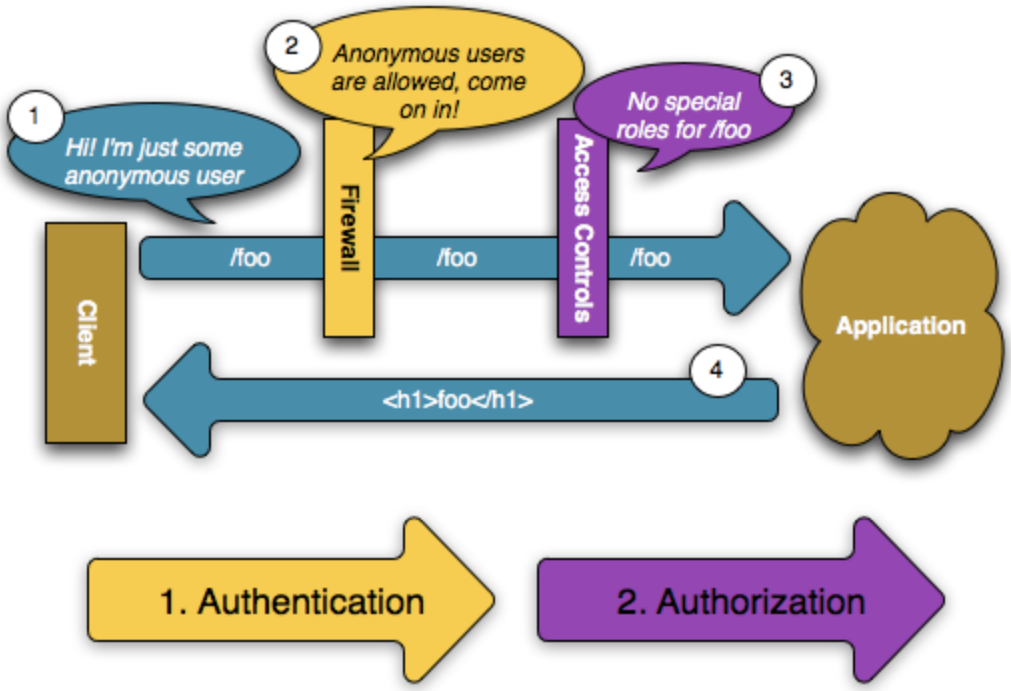

Figure 33. Authentication and Authorization in Symfony [9]

The firewalls in the security.yml file are used to configure how the website users will be authenticated. Typical authentication methods include using a login form, HTTP basic authentication, or an API token [10]. Access_control is then used in routes to ensure only users who have logged in can access the URL.

In addition, Symfony provides a Security component to improve the security of websites. The Security component allows features including authentication using HTTP basic, digest authentication, etc. It also allows developers to implement their own authentication strategies.

4.3.1.3. Node.js Express

There are some security packages in Node.js besides the regular security methods such as cookies. Helmet, one of these security packages, is able to protect the website from some well-known web vulnerabilities by setting HTTP headers appropriately [11]. Two common uses of

Helmet are to validate user input, one of the most important things to secure the web application, and to secure regular expressions [12].

4.3.2. Support

All three technologies have large communities of developers to get support from. In most situations, solutions can be found from these communities when encountering problems.

The open source Node.js is used by many third-party vendors and is the most popular language according to GitHub [13], as shown in Figure 34, while developers in ASP.NET MVC are usually dedicated developers, the community support of Node.js is stronger than ASP.NET MVC. Since Symfony is not the most popular PHP framework among many others, its support from the communities is not as strong as the other two technologies.

Figure 34. Ten Most Popular Languages in the Number of Pull Requests [13]

4.3.3. Industry Uses

Indeed.com, a job hunting website, was used in this paper to identify the uses of these three frameworks in the industry. This survey focused on companies in the Minneapolis area in Minnesota and the Sioux Falls area in South Dakota, as shown in Table 11.

When exploring job descriptions in Indeed.com, the technology used in the companies can be identified by searching for key words including ASP.NET, MVC, PHP, and Node.js. PHP and Node.js were used as key words instead of Symfony and Express which are studied in this paper due to the fact that most job descriptions did not specify the framework to be used in projects.

In the Minneapolis area, 34 companies were identified to be using these three technologies for web development (ASP.NET MVC, PHP, Node.js), among which 24 were using ASP.NET, six were using PHP, three were using Node.js, and one was using both PHP and Node.js. In the Sioux Falls area, eleven companies were found related to web development with the three technologies, among which eight were using ASP.NET MVC, two were using PHP, and one was using both PHP and Node.js.

Table 11. Survey on Usage of the Three Technologies

Technology	**Number of Companies Using the Technology**	
	Minneapolis Area	Sioux Falls Area
ASP.NET MVC	24	8
PHP	7	3
Node.js	4	1
Total	34	11

5. CONCLUSIONS AND FUTURE WORK

In this paper, three web server-side programming technologies were compared in terms of their development, performance, and other aspects.

In the development comparison, a hospital information management system was developed with the three different technologies, and were then compared in four aspects: setting up the environment, sever-side programming, front-end programming, and their connection to database.

When comparing the performance of these three technologies, benchmark testing was used, and the results showed ASP.NET MVC had the best performance among the three frameworks in Windows environment. In addition, it was discovered that Symfony and Node.js Express performed better in Linux environment than in Windows environment.

The three frameworks were further compared in terms of their security, support, and usage in industry. While these three frameworks have different methods to keep the web application secure, ASP.NET MVC and Symfony have similar security policies. With the support from Microsoft, ASP.NET MVC is more suitable for enterprise-level websites, whereas Symfony is more suitable for websites of small business, startups and independent software vendors. On the other side, Node.js fits better for websites that are computation-intensive or require high speed and performance.

Future work will be focused on other web development related technologies, such as Apache, IIS and Docker.

6. REFERENCES

[1] W3Techs. *Usage of Server-Side Programming Languages for Websites*. URL: https://w3techs.com/technologies/overview/programming_language/all, accessed March 6, 2018.

[2] Galloway, J., Wilson, B., Allen, K.S., and Matson, D. (2014). *Professional ASP.NET MVC 5*. Wrox, Hoboken, NJ, ISBN: 9781118794753.

[3] Titchkosky, L., Arlitt, M., and Williamson, C. (2003). A performance Comparison of Dynamic Web Technologies. *ACM SIGMETRICS Performance Evaluation Review*, 31(3), 2-11.

[4] Swales, D., Sewry, D., and Terzoli A. (2003). A Performance Comparison of Web Development Technologies to Distribute Multimedia across an Intranet. Proceedings of *2003 Southern Africa Telecommunication Networks and Applications Conference*, Fancourt, George, September 7-10, 2003.

[5] Prokofyeva, N. Entity Framework (EF). Entity Framework (EF). Entity Framework (EF). (2017). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. *Procedia Computer Science,* 104, 51-56.

[6] Lei, K., Ma, Y., and Tan, Z. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. Proceedings of *2014 IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, China, December 19-21, 2014, 661-668.

[7] Crawford, T. and Hussain, T. (2017). A Comparison of Server Side Scripting Technologies. Proceedings of the *15th International Conference on Software Engineering Research and Practice*, Las Vegas, NV, July 17-20, 2017, 69-76.

[8] Hideghety, B. (2016). *Get Know Your ORM - Avoid Bad Habits*. URL: https://www.linkedin.com/pulse/get-know-your-orm-avoid-bad-habits-balazs-hideghety/, accessed March 6, 2018.

[9] NewLifeClan. (2014). *Chapter 13: Security*. URL: http://www.newlifeclan.com/symfony/archives/215, accessed March 6, 2018.

[10] Symfony. *Security*. URL: https://symfony.com/doc/current/security.html, accessed March 6, 2018.

[11] Express. *Production Best Practices: Security*. URL: https://expressjs.com/en/advanced/best-practice-security.html, accessed March 6, 2018.

[12] Nemeth, G. (2017). *Node.js Security Overview*. URL: https://nemethgergely.com/nodejs-security-overview/, accessed March 6, 2018.

[13] GitHub. (2017). *The State of the Octoverse 2017*. URL: https://octoverse.github.com/, accessed March 6, 2018.