Backend Framework Analysis

By Luke Daniel and Luke Farthing

Overview

Introduction

- The purpose of our project was to do an in-depth test of two different web application development frameworks in terms of usability and performance.
- Chosen Frameworks
  - Django (Python)
  - ASP.NET Core (C#)
- Our goal was to develop two web applications, one in Django and one in ASP.NET Core. These web applications would be forum post websites and performance would be based on the GET method (loading the page and it's associated data).
- Grading Metrics
  - Documentation
  - Development Tools
  - Rendering Visuals
  - Back End Code Development
  - Load Testing

Motivation

- We are coworkers and we both frequently work on software applications, more recently becoming involved in web development.
- A constant recurring question for us: What framework should we use?
- Most of the products that we use are Microsoft based: Visual Studio, WPF, ASP.NET Core, etc.
- What other frameworks are out there? What are sufficient metrics/criteria to make decisions on which frameworks would work best for us in a “development”?

Related Work

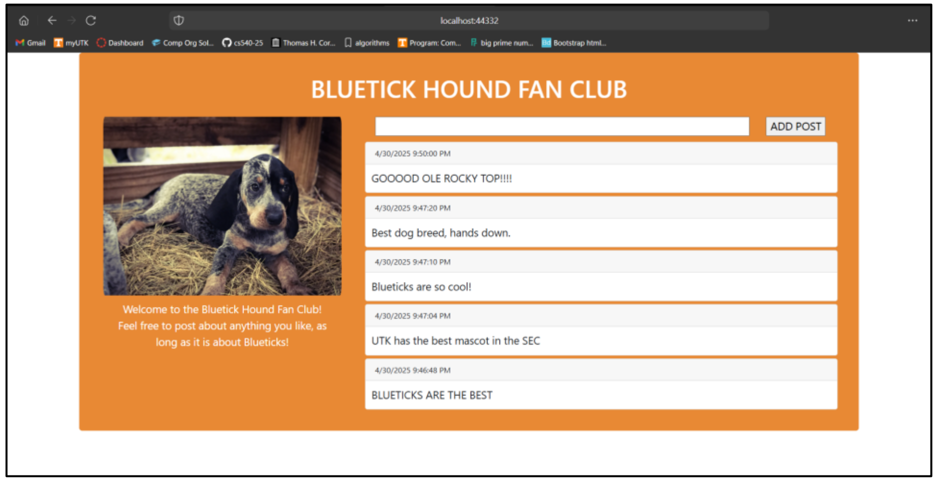
- Literature related to the evaluation of web development frameworks often includes metrics that generate scores based on a mix of factors such as community support, market demand, and some performance metrics.
- For instance, Kaluža et al. propose a list of scoring criteria that is 24 elements long. These elements include things like “high scalability” and “adapted to beginners” alongside other more subjective metrics such as Github and Reddit ratings.
- Raible proposes a similar evaluation rubric that includes many similarities to the one above but also focuses on compatibility with specific software packages.

Research Question

Can we define a set of both quantitative and qualitative metrics that will allow users to effectively evaluate characteristics of one framework against those of another?

How do our two chosen frameworks (Django and ASP.NET Core) stack up to these metrics?

Visuals



BLUETICK HOUND FAN CLUB

Welcome to the Bluetick Hound Fan Club! Feel free to post about anything you like, as long as it is about Blueticks!

4/30/2025 9:50:00 PM
GOOOOD OLE ROCKY TOP!!!!

4/30/2025 9:47:20 PM
Best dog breed, hands down.

4/30/2025 9:47:10 PM
Blueticks are so cool!

4/30/2025 9:47:04 PM
UTK has the best mascot in the SEC

4/30/2025 9:46:48 PM
BLUETICKS ARE THE BEST

Django

- Used HTML and CSS to design pages.
- Utilizes models (classes defined in Python) to read and write data to the view.
- Django templating allows for creating generic base templates for commonly formatted pages while also allowing for extensions that can define specific elements that may be unique to a specific page.

ASP.NET Core

- Uses HTML (with a twist) and CSS to design web pages
- Has the Bootstrap visual library built-in which provides easy-to-use classes that allow grid organization of elements as well as access to a whole library of components.
- Allows for "binding" to backend elements to dynamically display variables to the screen.

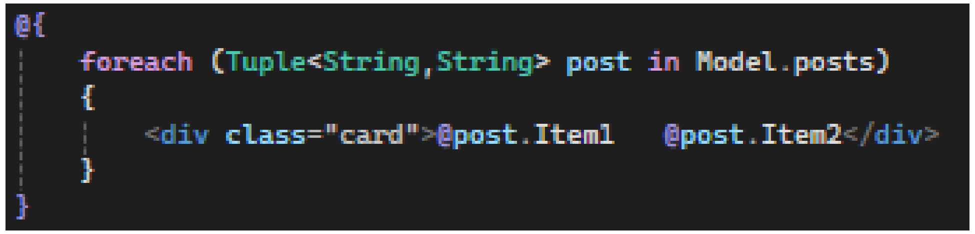
Backend Coding

Django

- Python
- Page functions are written in Python allowing users to utilize a wide range of libraries to manipulate data.

ASP.NET Core

- C#
- Recommended to follow the Model, View, View Model (MVVM) architecture to take full advantage of binding and code organization.
- Allows for C# code to be directly inserted into the html file by using the "@" symbol.



```
@{
    foreach (Tuple<String,String> post in Model.posts)
    {
        <div class="card">@post.Item1   @post.Item2</div>
    }
}
```

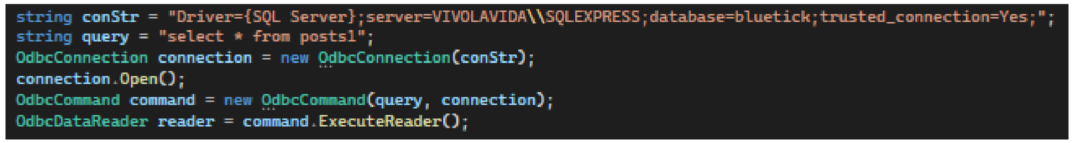
Database Connectivity

Django

- When a project is created, a SQLite database is automatically created to support initial testing.
- A development server can also be utilized to speed up testing and troubleshooting by allowing the developer to quickly deploy locally.
- Many different database types can be used and only require minor changes to the settings file for implementation.

ASP.NET Core

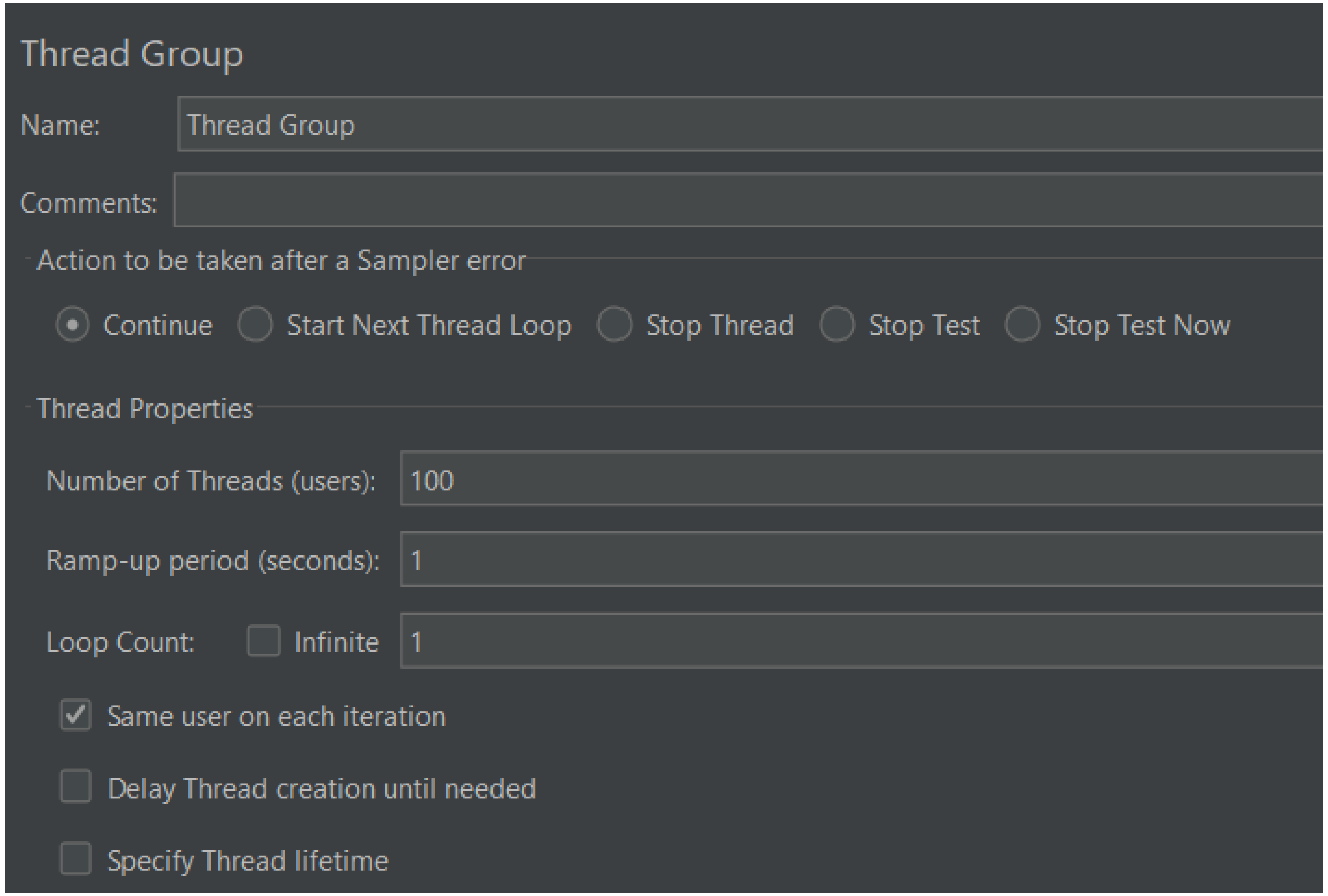
- Must create a database manually
- Set up all required tables manually through a software like SSMS
- Use a package (such as System.Data.Odbc) to connect to the database
- Manually place data in the database through queries run through Odbc connections.



```
string conStr = "Driver={SQL Server};server=VIVOLAVIDA\\SQLEXPRESS;database=bluetick;trusted_connection=Yes;";
string query = "select * from posts1";
OdbcConnection connection = new OdbcConnection(conStr);
connection.Open();
OdbcCommand command = new OdbcCommand(query, connection);
OdbcDataReader reader = command.ExecuteReader();
```

Results – Performance (JMeter)

Setup – Thread Group


Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 100

Ramp-up period (seconds): 1

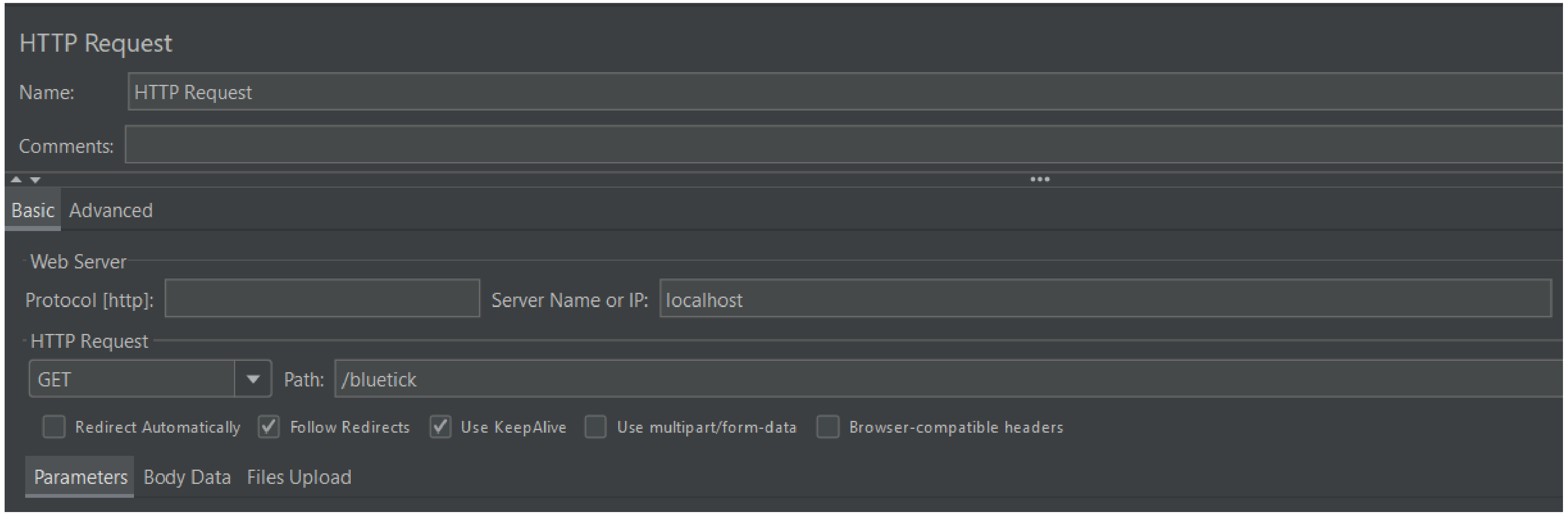
Loop Count: ☐ Infinite 1

☑ Same user on each iteration

☐ Delay Thread creation until needed

☐ Specify Thread lifetime

Setup – HTTP Request



HTTP Request

Name: HTTP Request

Comments:

Basic | Advanced

Web Server

Protocol [http]:

Server Name or IP: localhost

HTTP Request

GET

Path: /bluetick

Redirect Automatically [ ] | Follow Redirects [✓] | Use KeepAlive [✓] | Use multipart/form-data [ ] | Browser-compatible headers [ ]

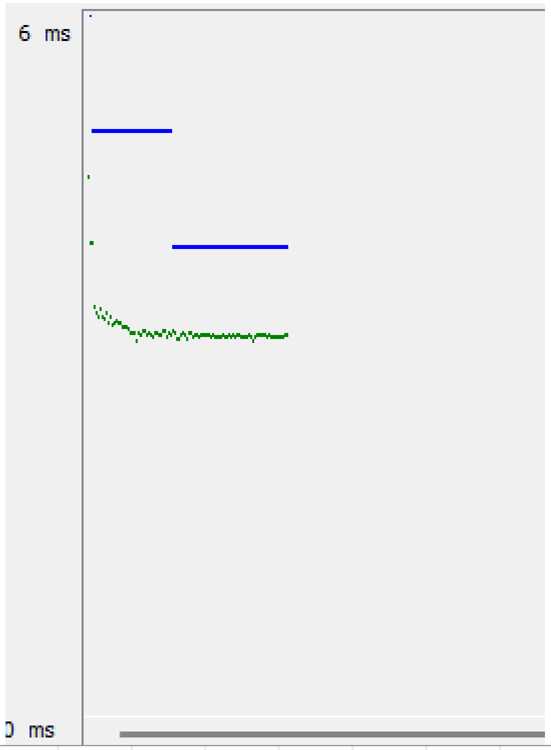
Parameters | Body Data | Files Upload

Test Cases

Users (GET)	Loop Count
100	1
300	2
500	3

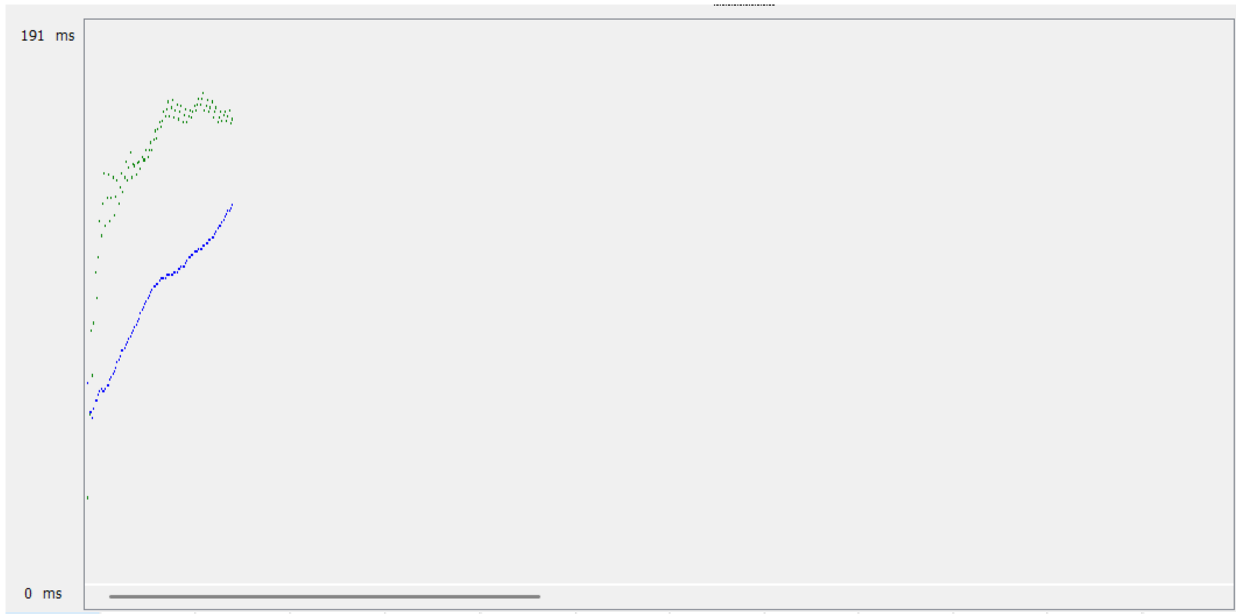
Test Case 1 Results (100, 1)

Django



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received ...	Sent KB/sec
LukePage	100	4	5	6	6	8	3	11	0.00%	100.4/sec	230.02	12.4

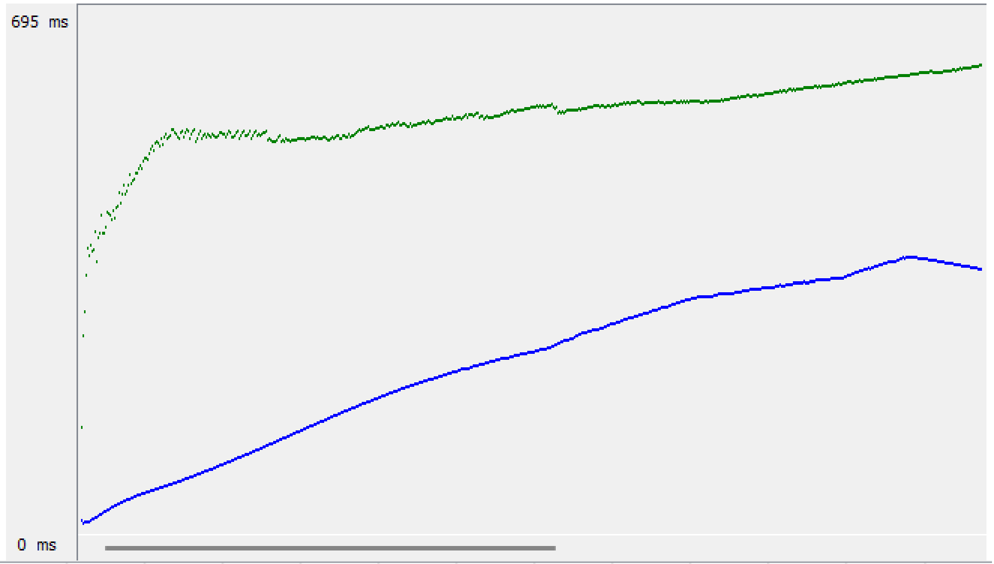
ASP.NET Core



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received ...	Sent KB/sec
HTTP Request	100	129	129	191	204	218	49	227	0.00%	81.5/sec	225.95	9.15

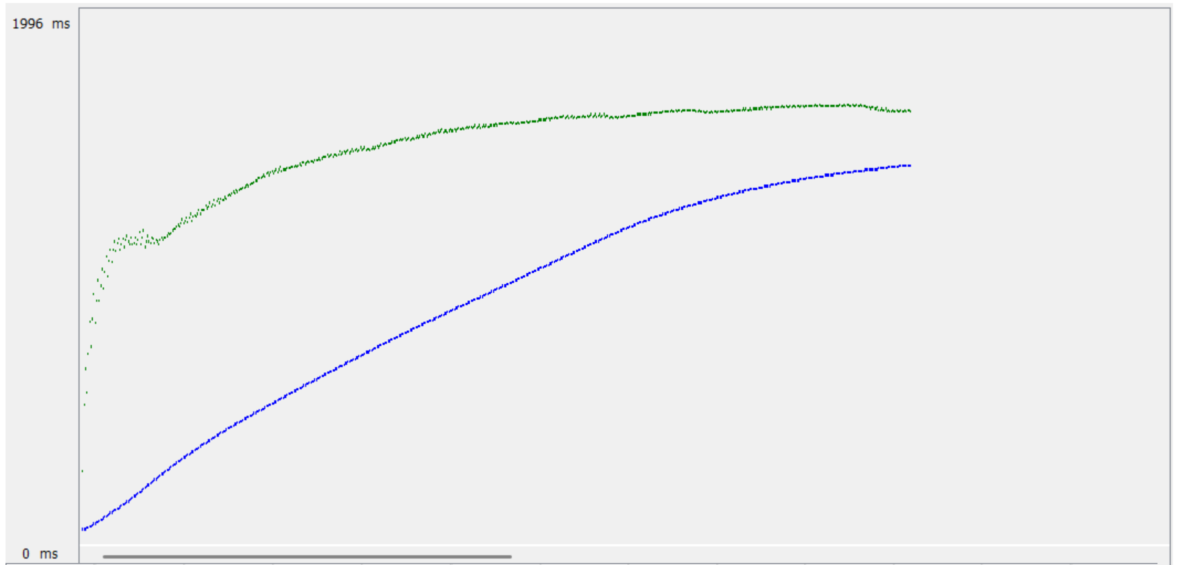
Test Case 2 Results (300, 2)

Django



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received ...	Sent KB/se
LukePage	600	349	305	695	856	873	8	876	0.00%	317.0/sec	726.15	39.3

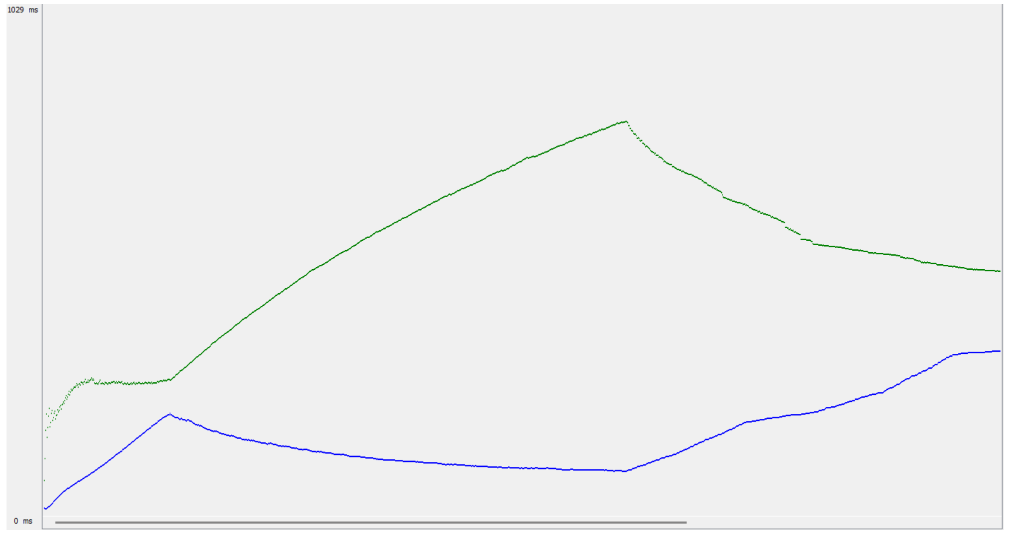
ASP.NET Core



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received ...	Sent KB/sec
HTTP Request	600	1422	1685	1996	2036	2060	48	2068	0.00%	125.9/sec	349.03	14.14

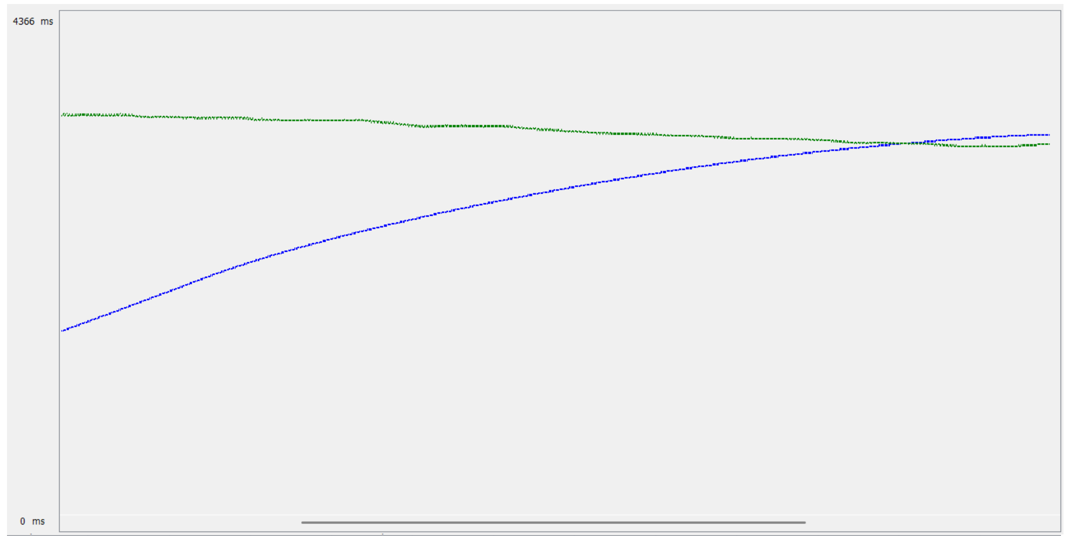
Test Case 3 Results (500, 3)

Django



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
LukePage	1500	337	261	1029	1724	1946	0	2218	36.33%	493.7/sec	1181.11	38.9

ASP.NET Core



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request	1500	3311	3982	4366	4390	4409	54	4426	0.00%	113.2/sec	313.89	12.71

Stress Test Results

Framework	# of Samples	Avg. Response Time (ms)	Error %	Throughput
ASP .NET CORE	100	31	0	96.4/sec
DJANGO	100	4	0	100.4/sec
ASP .NET CORE	600	1421	0	127/sec
DJANGO	600	349	0	317/sec
ASP .NET CORE	1500	2698	0	137.5/sec
DJANGO	1500	337	36.33%	493.7/sec

Limitations

- Our tests assume that the users are at least familiar with the underlying language of each framework.
- While the basic websites developed in this project are as close as possible to each other, some differences in how the sites are hosted may affect load test data.
- Comparing larger numbers of frameworks using this method may be difficult due to the time it takes to develop a basic webpage to test.

Conclusions

- We evaluated each framework based on five categories and made decisions on whether one framework performed better in any category (noted with a ✓ below).

- This evaluation methodology does not make an overall judgement on whether one framework is better than another, Instead, it allows potential users to choose based on what characteristics are most important to them.

	DJANGO	ASP .NET CORE
Documentation/Examples	✓	
Development Tools	✓	
Rendering Visuals		✓
Back End Code Development	✓	
Load Testing		✓

References

- Kaluža, Marin, et al. "A COMPARISON OF BACK-END FRAMEWORKS FOR WEB APPLICATION DEVELOPMENT." Journal of the Polytechnic of Rijeka, vol. 7, no. 1, 13 May 2019.
- Raible, Matt. "JVM Web Frameworks Rating Logic." Google, Google, 6 Dec. 2010, docs.google.com/document/d/1X_XvpJd6TgEAMe4a6xxzJ38yzmthvrA6wD7zGy2Igog/pub.
- I. P. Vuksanovic and B. Sudarevic, "Use of web application frameworks in the development of small applications," *2011 Proceedings of the 34th International Convention MIPRO*, Opatija, Croatia, 2011, pp. 458-462.
- P. Thakur and P. Jadon, "Django: Developing web using Python," *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India, 2023, pp. 303-306, doi: 10.1109/ICACITE57410.2023.10183246.

THANK YOU