Project Report: Backend Framework Analysis

Daniel, Luke Morris Farthing, Luke Alexander

***Abstract*—**

INTRODUCTION: While web developers have a wide variety of options when choosing a backend framework, there is a need for a streamlined method for conducting a comprehensive comparison of different options.

OBJECTIVES: The primary objective of this project was to use a set of quantitative and qualitative metrics to evaluate multiple backend frameworks with the purpose of highlighting strengths and weaknesses for potential users to consider when tasked with choosing a framework.

METHODS: Two popular web development frameworks, Django and ASP.NET Core, were selected for evaluation. Five key framework characteristics (availability of documentation, quality of development tools, how to render visuals, ease of back-end code development, and performance in load testing) were identified for comparison. To test each of these, a basic webpage was designed, and an instance was implemented using each framework.

RESULTS: After creating instances of the webpage with both frameworks, the project team was able to draw several conclusions on the advantages that each offers to developers. Apache JMeter was used to perform load testing and revealed that the ASP.NET Core instance handles higher levels of concurrent users with much fewer errors than the Django instance, which gave it the edge in the performance category. However, the easy-to-use development server and automated database support provided by Django had a noticeable and positive effect on total development time.

CONCLUSION: The project team concluded that this evaluation methodology provided a comprehensive view of the strengths and weaknesses of each framework. ASP.NET Core showed more resilience in load testing and would probably be the choice for large applications that experience heavy traffic. On the other hand, if a developer prioritizes low development time, Django's tools and easy testing interface might make it a better choice. By focusing on independently evaluating each of the five identified framework characteristics, the project team was able to produce a detailed list of results that can be easily referenced no matter the priorities of a specific project.

I. Introduction

The number of tools that a prospective web developer must consider when choosing a backend framework for a website can be daunting. Backend frameworks play a pivotal role in a host of different server-side application functions such as database interaction and user authentication. While the selection of a backend framework may have obvious impacts on the software development process depending on how easy it is for each project team member to use, there may also be characteristics that have significant effect on speed, performance, and the overall security of the application.

Related work that was reviewed in the literature survey centered around project teams developing scoring rubrics that focused on producing a cumulative score to compare framework quality on a comprehensive basis. In most cases, a predetermined set of criteria was defined and applied to each framework with varying methods for scoring each one. The type and number of criteria vary wildly from one study to the next and often oftentimes include requirements that are specifically applicable to a project team's intended use case for a web application or give points for having support for a specific set of libraries or plugins [4]. Many other framework evaluation methods also incorporated inputs such as ratings from online forums such as Github, Reddit, and others [5]. These varying evaluation methodologies and scoring criteria combinations indicate that there is room for an improved and simplified system.

The objectives of this project were to define a set of quantitative and qualitative metrics to use when comparing backend frameworks and then to apply tests in line with those metrics to a selected pair of frameworks. In addition, the goal of this approach was not to score the metrics on a cumulative basis but rather to evaluate each one individually so that any prospective user could apply their own weighting to each category depending on their individual preferences and the specific needs for any application under development. This will allow the proposed evaluation method to be used in many more cases than just with the specific frameworks used in this project.

The results of this project will enable any interested party to make informed decisions on which of the two examined web development frameworks (Django or ASP.NET Core) to use when starting a new project. By defining and implementing a basic webpage, each framework was evaluated to determine not just whether they can perform well under various degrees of load testing, but also on how easy they are to use and how fast they can be picked up by new users. This information can be leveraged to better select tools that will increase the speed, efficiency, and effectiveness of web development efforts in the future. This testing methodology can also be applied to any other instance where a comparison of web development frameworks is warranted.

II. METHOD

A. Parallel Webpage Development

In an effort to perform a like-for-like test of an application developed in two different frameworks (Django and ASP.NET Core), two web pages with extremely similar features and

design were created. Both projects were to make use of the framework it was developed in and all of the tools available in it. The page being developed would be a simple forum post website, the “Bluetick Hound Fan Club”.

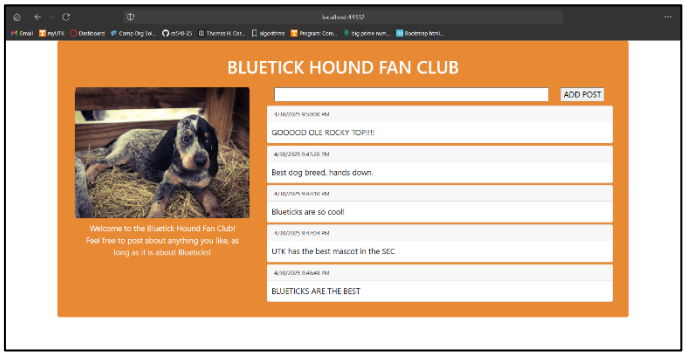

Fig. 1. View of Webpage Format Used for Testing.

Both implementations of the web page take text entry, record datetime, store data in a database, and display the posted data to the screen.

B. Django and ASP.NET Core [2]

1) Django

Django is a web development framework based in Python. Neither of the members of the project team knew much about Django before the execution of this project. However, it was the expectation of the project team that Django (in comparison to ASP.NET Core) would mostly be much more welcoming to new users and have many easy-to-use tools available, at the cost of performance. Django is the most popular Python web application framework and is widely used by many popular sites [6].

2) ASP.NET Core

ASP.NET Core is a web development framework based in C#. Both members of the project team had previous experience with this framework in their full-time jobs.

C. Evaluation Metric

The evaluation metrics chosen for this project are based on general categorizations of framework characteristics valued by the project team members when deciding which framework to use in the development of web applications. Each of these five metrics is described with some detail in the following subsections.

1) Documentation/Examples

The documentation/examples category focuses on the resources readily available to aid in the development of a project in each framework. Lack of documentation for a framework (due to how new or niche a framework is) could leave a developer stranded with little to no options for moving forward. However, if a framework has plenty of documentation, then most problems can be solved by simply searching online. If the most common issues of a framework are well-documented and have been solved before, development of new projects and becoming familiar with a new framework becomes a relatively straightforward task.

2) Development Tools

Development tools needed for a project depend heavily on what kind of app is being developed. For a forum post website that needs to be load tested, manipulating a database and deploying the application to a server are two functions that must be performed by each framework. What tools do each framework have that make implementing database connection and server deployment easier on the developer?

3) Rendering Visuals

The rendering visuals category focuses on the how simply the frontend development of the page can be carried out. Even though (in this project) both chosen frameworks use HTML and CSS, how do they differ from each other in terms of getting the right page layout and content that the developer desires to display? Some frameworks simply give the user access to HTML and CSS and expect the user to format everything manually. On the other hand, there are many frameworks that have built-in libraries that make visual design of a web page a much simpler task.

4) Backend Code Development

The backend code development category focuses on the data processing of the page and how the developer must go about receiving information from users and displaying responses back to the page. In web apps, data is not as easy to maintain, and it usually disappears very quickly due to the fact that web pages are almost always run remotely on a server and its priorities are to run fast and use up the least amount of memory possible. If a framework is extremely obtuse and data is hard to manage, developers may run into roadblocks when trying to retain data, which may prolong development time.

5) Load Testing

The load testing category is extremely important when it comes to evaluating the validity of a web development framework, especially if that framework must maintain functionality and speed when under increased load. [1] Load testing can be done in many ways with various degrees of accessibility and capability depending on the breadth of what needs to be tested. An easy-to-use and readily available strategy is Apache JMeter [10]. . [1]

TABLE 1: JMeter Test Plan

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

a) Thread Group

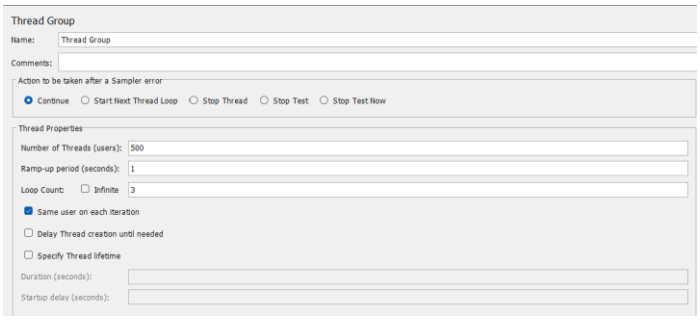

Fig. 2. Thread Group Setting Page in JMeter.

A "Thread Group" is Apache JMeter's way of simulating a user. The settings that are manipulated for testing purposes in this project are the "Number of Threads (users)" setting and the "Loop Count" setting. The number of users setting is how many users will be involved in the entire test which becomes more computationally intense as that number is increased. The loop count setting determines how many times the test will loop through. For example, in the picture above, there are 500 users being created with a loop count of 3. Therefore, during the test there will essentially be a total of 1,500 users trying to access the web page.

b) HTTP Request [8]

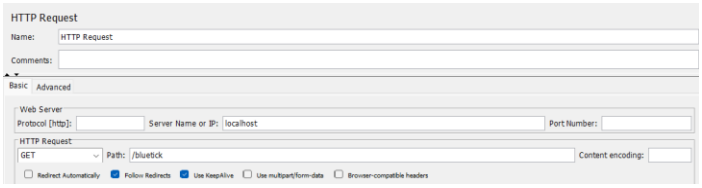

Fig. 3. HTTP Request Setup in JMeter.

Every user (thread group) in Apache JMeter can be assigned many different types of tasks. For this project, each user will be performing an HTTP Request to GET the web page. Since testing is being conducted locally, the HTTP Request will be performed on a subdirectory of localhost.

c) Displaying Results

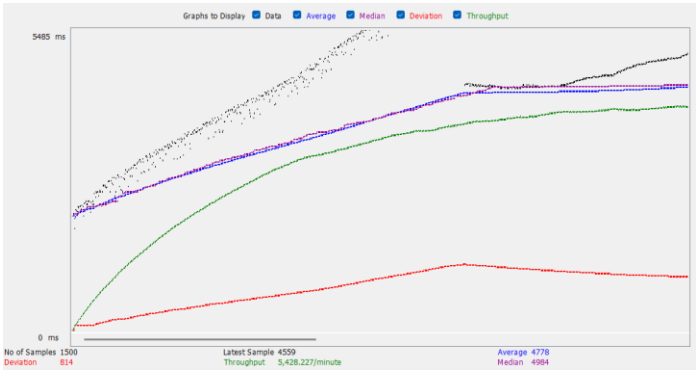

Fig. 4. Test Results in JMeter.

Apache JMeter gives the user multiple options when it comes to displaying the data from the test that has been performed. Pictured above is the "Graph Results" option. This option gives the user easy-to-understand visual of what happened to response time as well as throughput as the test was conducted. However, this graph is somewhat limited in the amount of data it displays when compared to the other data displays that JMeter provides ("Aggregate Report" for example).

III. RESULTS

A. Parallel Webpage Development

The results of the JMeter load tests performed on the Django and ASP.NET webpage instances can be found below in Table 2. These tests were performed in line with the test plan shown in Table 1.

TABLE 2: JMeter 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

B. Full Comparison Matrix

After developing each of the webpages and completing the JMeter tests, the project team compiled their experiences and data to make determinations on which framework ranked better in each of the five selected categories. It was decided to limit the scoring to a binary choice between the two and to not generate an overall cumulative score.

TABLE 3: Criteria Comparison Table

	DJANGO	ASP .NET Core
Documentation/Examples	X	
Development Tools	X	
Rendering Visuals		X
Code Development	X	
Load Testing Performance		X

C. Results Explanation

1) Documentation

Since ASP.NET Core is a relatively newer framework and is preceded by a very similar, but different, framework (ASP.NET), finding help or documentation online is more difficult than it is for Django.

Django is a very common and highly touted framework for its ease-of-use and accessibility. There are plenty of forum posts online and there is no confusion between Django and a predecessor framework.

2) Development Tools

ASP.NET Core expects the developer to handle most aspects of the application manually, this includes connecting to a database and deploying the app to a local server. Although there a tools that make the execution of these tasks straightforward, there is much more setup involved. To connect to a database and manipulate it, a toolkit (System.Data.Odbc for example) must be used, a connection string must be made, and all SQL queries must be written manually by the developer. Additionally, there are many settings and applications that need to be downloaded to create databases as well as enable a PC to allow custom web apps to be deployed to it.

Django expects much less setup from the developer in terms of getting a test environment up and running. By running a few commands from the command line, Django will create and update a database for you as well as deploy to its own version of a test environment. These features allow Django users to rapidly perform testing and troubleshooting without much other setup.[7]

3) Rendering Visuals

Both Django and ASP.NET Core employ HTML and CSS files to develop the front end of their applications. However, ASP.NET comes with Bootstrap built-in. Bootstrap gives the developer immediate access to a lot of different visual elements as well as an easy way to configure the page layout using a grid system.

4) Code Development

ASP.NET Core has access to many coding libraries as well as having the functionality of Razor pages which allows the developer to insert C# code directly into HTML. However, retaining data between pages and refreshes can be extremely difficult. It takes much more effort to maintain the information that you need in ASP.NET Core than it is in Django, making it much easier for a developer (especially if they are new to web development) to implement the functionality of their web app.

5) Load Testing Performance

ASP.NET Core performed slower in each test case compared to Django. However, Django did not handle the increase in users and GET requests well. In the third test case (500 users, 3 loop count), Django was producing an error 36.33% of the time. For many large-scale applications, Django may not be the best fit if reliability under load is highly valuable to the developer.

IV. Threats to Validity

While the webpages developed in this project were intentionally created to be as similar as possible, there were a couple of noticeable differences between the two. While both were hosted locally on the same hardware, the pages were hosted on different types of servers. The ASP.NET Core instance was hosted using IIS, while the Django instance was hosted locally using the waitress Python library (pure-Python WSGI server). The project team does not anticipate that the performance results were greatly affected by these choices, but further work may be needed to verify this theory.

Another point to note is that the evaluation metric proposed in this paper is probably most effective if those who use it are proficient in the underlying languages associated with each framework. For instance, the project team had significant experience developing with both C# and Python which helped limit the amount of development time that was spent on getting up to speed on syntax and other language specific characteristics. While this may limit which frameworks can be compared, it may serve as first stage for evaluation.

V. Conclusions

The two frameworks, Django and ASP .NET Core, showed different reasons why they are both legitimate choices for developers depending on what the priorities are for a specific project. Neither showed any major weaknesses and have been used in a variety of large applications. As shown in our results, it was evident that Django's tools and easy to use default database structure made it an obvious choice for projects that prioritize fast development time and a streamlined troubleshooting process. On the other hand, we saw that ASP .NET Core is a better choice for larger applications that experience high levels of concurrent users.

The results found when developing and testing the two instances of the web pages were valuable tools for highlighting

the strengths and weaknesses of each framework being compared. The five categories of investigation defined above allowed the project team to get a comprehensive overview of each framework by leveraging personal experience and open-source software testing tools. This combination of insights that mixes developer experience with hard testing makes this evaluation method versatile and able to be used in other framework comparisons.

VI. References

[1] "Performance Testing: Types, Metrics and How To." BrowserStack, 17 Jan. 2025, www.browserstack.com/guide/performance-testing.

[2] "Top 7 Backend Development Frameworks [2025 Updated]." GeeksforGeeks, GeeksforGeeks, 10 Dec. 2024, www.geeksforgeeks.org/frameworks-for-backend-development/.

[3] 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.

[4] Raible, Matt. "JVM Web Frameworks Rating Logic." Google, Google, 6 Dec.2010,docs.google.com/document/d/1X_XvpJd6TgEAMe4a6xxzJ38yzmthvrA6wD7zGy2Igog/pub.

[5] 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.

[6] 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.

[7] Nanodano. "Run Python WSGI Web App with Waitress." DevDungeon, 8 Aug. 2020, www.devdungeon.com/content/run-python-wsgi-web-app-waitress.

[8] Templin, Reagan, et al. "Introduction to IIS Architectures." Microsoft Learn, 16 Feb. 2023, learn.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture.

[9] S. Pradeep and Y. K. Sharma, "A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications," *2019 Amity International Conference on Artificial Intelligence (AICAI)*, Dubai, United Arab Emirates, 2019, pp. 399-403, doi: 10.1109/AICAI.2019.8701327.

[10] R. Abbas, Z. Sultan and S. N. Bhatti, "Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege," *2017 International Conference on Communication Technologies (ComTech)*, Rawalpindi, Pakistan, 2017, pp. 39-44, doi: 10.1109/COMTECH.2017.8065747.