CS 687 Computer Science Capstone

**Comparison between Full Stack Architecture:**

**Typescript/ Javascript vs Python**

Tuan Khai Tran, [t.k.tran@cityuniversity.edu](mailto:t.k.tran@cityuniversity.edu)

Advisor: Dr.Sam Chung [chungsam@cityu.edu](mailto:chungsam@cityu.edu)

MS in Computer Science

City University of Seattle

Nov 20th, 2018

**Abstract**

When human needs become greater and greater, more creative ideas and inventions were born to meet these needs. That is why information technology was developed at such a strong and rapid level. On top of that, being a software developer means they will need to learn during their life as technologies and trends change frequently in the software field. More and more tools, frameworks were provided with the intention to improve the performance and efficiency of an old task so we all should be ready to face sudden changes in order to ensure productivity. One major representation of this is web development and the way it has done has changed a lot over the years. It started from static web pages with HTML, CSS to more interactions with Javascript. Then we have frameworks, technology stacks that basically provide a sample architecture to build an application from head to toe. This thesis aims to research and analyze the topic of Full Stack Architecture. The main goal is to study the differences in terms of structure, performance and use cases of full-stack models using Javascript (MEAN) and Python (Django).

**Keywords:** Full stack, Python, JavaScript, TypeScript, Django, MEAN, ExpressJs, NodeJs, MongoDB, Angular

# TABLE OF CONTENTS

[TITLE](#__RefHeading___Toc208803836) 1

[ABSTRACT 2](#__RefHeading___Toc208803837)

[TABLE OF CONTENTS 3](#__RefHeading___Toc208803838)

[PROBLEM IDENTIFICATION 4](#__RefHeading___Toc208803839)

[PURPOSE 4](#__RefHeading___Toc208803840)

[OVERVIEW 4](#__RefHeading___Toc208803840)

[APPROACH 5](#__RefHeading___Toc208803842)

[LITERATURE REVIEW 5](#__RefHeading___Toc208803849)

[DATA COLLECTION 8](#__RefHeading___Toc208803850)

[FINDINGS 8](#__RefHeading___Toc208803852)

[CONCLUSION 12](#__RefHeading___Toc208803853)

[FUTURE WORK 12](#__RefHeading___Toc208803854)

1. **PROBLEM IDENTIFICATION**

Once upon a time, web development confines with HTML, CSS, and Javascript for front-end and PHP on the backend. However, there are more and more approaches for developers to build web applications nowadays with the rise of technology including more languages, more tools, frameworks along with the development of cloud computing, Software as Service, Infrastructure as Server and etc. It is unarguable that these technologies provide an easier, more efficient and more effective application development process. Nevertheless, this can create more confusion for software developers on which technology and architecture to use. In addition, with the existence of too many available tools, each of the tools may require slightly different skills and knowledge. Choosing a correct architecture from the beginning will have a tremendous impact that allows constant iteration, rapid prototyping, better performance, maintenance and robustness (Adhikari, 2016). Therefore, it is important to dig down to this area, see what are some pros, cons, and differences between some full-stack frameworks.

1. **PURPOSE**

The purpose of this paper is to provide somewhat a direction and guideline on which technology stack to use depended on different circumstances through analyzing some of the most popular frameworks and architectures nowadays (MEAN and Django).

1. **OVERVIEW**

In order to make a complete application, writing code alone is not enough. We will need to store information, which means we have to put it into the database. After completing the code, we must find a way to deploy it somewhere to run the code. A complete program must not only have code but also must have the operating system platform and several accompanying software(web server, database). Assembling all these together will form something we call a technical stack. The Technical Stack, also known as the solution stack, is a collection of software or technologies that work together to form a platform for applications to work (Bulatovych, 2019). A stack is usually made up of these components:

* Frontend
* Web Server
* Database Server
* Backend

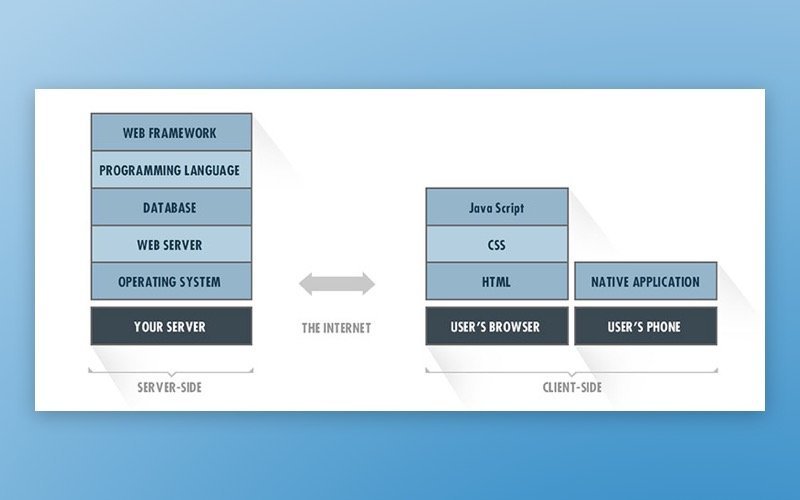


Figure 1: Typical app tech stack(Bulatovych, 2019)

Back in the day, we have the very first and popular stack that most website used, LAMP which consist of Linux, Apache web server, MySQL and PHP/Python/Perl. However, a stack that has been gaining popularity over the last few years is the MEAN stack, which is the combination of MongoDB, Express, Angular, and Nodejs. One of the main reason for the rise of this stack is that the way it works involves around NodeJs and its capability of building a whole web application with just Javascript (Ganguly, 2019)

Another popular option for web development is using Python with the Django framework. From the sound of it, this does not seem like a stack because we do not see like 4 components combined together like MEAN because Django technically is a Python web framework for both frontend and backend. It can be said that Django is an equivalent of Express in MEAN.

Besides the obvious difference between using Django and MEAN is that Django uses Python while MEAN uses JavaScript, Django is designed to support relational database as opposed to MongoDB which is a non-relational database. Some organizations reportedly use Django in their tech stacks include Pinterest, Instagram, Coursera and Udemy (Rubio, 2017).

It is also worth to note that the same web application results can be produced just by any web framework and programming language (Rubio, 2017). But what matters and will vary are the time we spend to create the solution, prototype, maintenance, adding new features, testing, debugging, cost, scale, and many other things.

1. **APPROACH**

This paper will use a qualitative approach to gather all resources, case studies and information needs to analyze the subject matter. It will then look into each different component inside a stack and compare possible framework solutions within that component. It will then attempt to conclude and propose suitable choices for different circumstances along with its reasoning.

1. **LITERATURE REVIEWS**

A comparative analysis of Nodejs by Chhetri 2016 gave a comprehensive introduction on NodeJs capability as well as carried out an investigation of Nodejs’s performance by implementing several server load tests with Apache Servers running PHP. The paper summarizes some features of Nodejs are asynchronous, single thread, event-driven, non-blocking I/O model. JavaScript can be used from end to end not just client-side thanks to Nodejs. The way Nodejs can achieve high concurrency is different than conventional web servers due to its non-blocking I/O model. For legacy models, they tend to block requests when there is currently an I/O operation that occurs. Nodejs’s features of asynchronous programming, using event loops and event-driven I/O is for preventing this from happening which makes it possible to serve multiple requests even with just a single thread model. This also results in the small overhead of Nodejs. Besides, this model also gives an outstanding advantage to Nodejs which makes it become a high-performance web server with more scalability and less cost. Another benefit that is mentioned by Rubio is a simple development environment. He stated that Node can simplify the process of downloading packages, setting up environments for its application ready to build. Setting up a development environment for a traditional web server is tedious, time-consuming and prone to errors because of stuff like dependencies version, packages’ location, etc. (Chhetri, 2016)

The research also conducts a test between Node and Apache Servers on their capability to handle a high level of requests and concurrency on both Windows and Ubuntu environments. They found that the response time for Nodejs is significantly lower than Apache even if they tremendously increase the number of requests. The results from the test are shown in Figure 2.

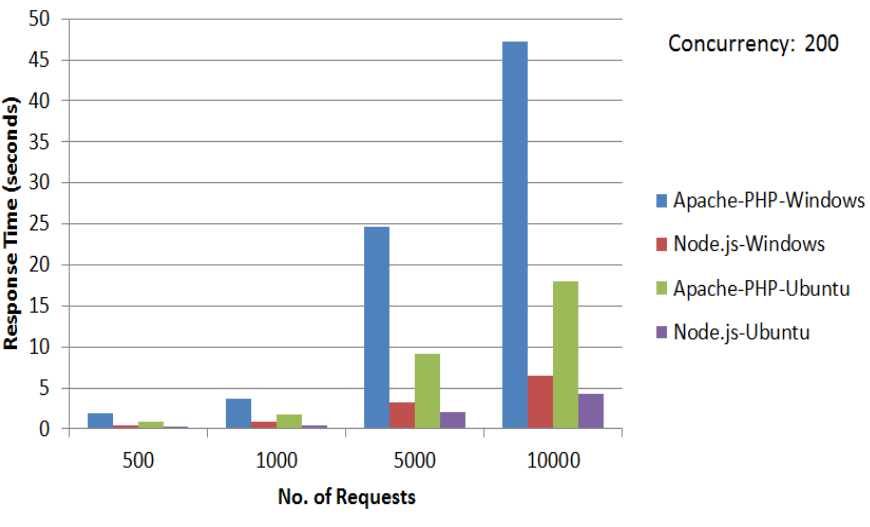


Figure 2: Response Time Graph (Chhettri, 2016)

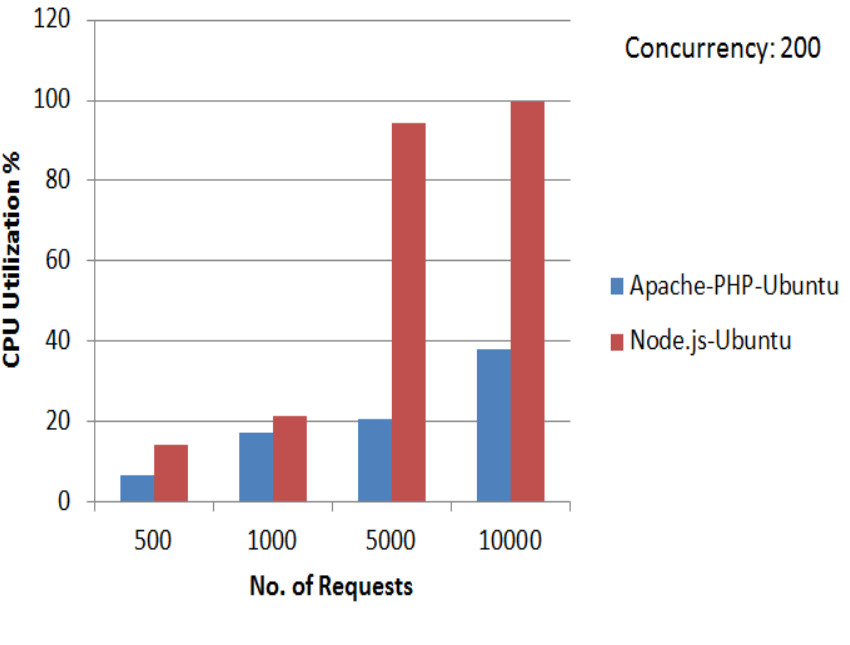


Figure 3: CPU Utilization Graph (Chhetri, 2016)

However, Nodejs fell short in CPU Utilization as it eats up about double the CPU resources compared to Apache. Figure 3 shows us the CPU usages of the two.

The paper concluded some of the limitations of Nodejs. First, it is good at handling a big number of requests but poor in handling heavy server-side computation. Second, Node does not have very good integration with the relational database. Although NoSQL is very popular and has its own strengths but so does regular SQL. Therefore, an immature relational database integration might not be suitable for some organizations (Chhetri, 2016).

Crawford and Hussain, 2016 did a comparison between several server-side scripting technologies in terms of development tools, package management system, integration with database and performance. Targeted technologies that they mentioned in their study are Nodejs, PHP, Django and Ruby on Rails. They found that Nodejs has an overwhelming number of packages compared to the others with approximately 345,000 packages and they are all managed by the Node Package Manager (npm) which can be installed via the command line. More packages do not necessarily mean good because this can be the sign of lots of developers are interested in Nodejs or there are lots of missing features in the original Nodejs so it has to use packages to make it up. They also noted that PyPi, the package manager of Django does not have full and comprehensive documentation for all the packages, and this can cause troubles for developers when they want to use the tool.

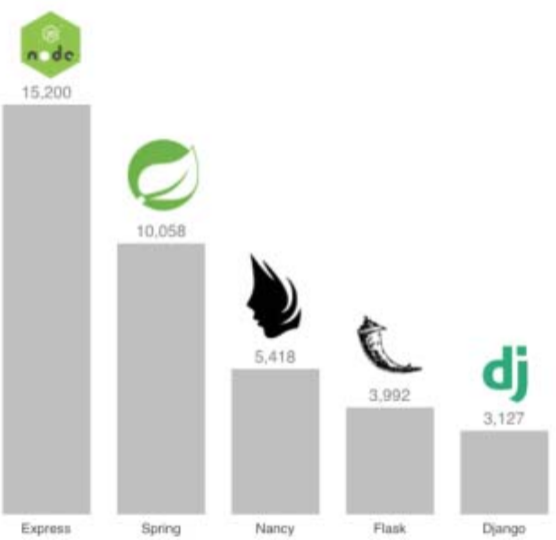


Figure 4: Plain text requests/second performance comparison (Crawford & Hussain, 2016)

In terms of performance, the paper showed the number of requests per second that each of the technologies can handle. In this test, they took Express into account as a representation for Nodejs. The results demonstrated that Express is dominant compared to its rivals. It can handle about 3 times more requests than the second place, Django in every kind of requests include plain text requests, JSON requests and database requests (Crawford & Hussain, 2016).

Maksimov wrote a master’s thesis on the performance of MongoDB, the most widely used NoSQL database and PostgreSQL, the most widely used relational database in 2015. Maksimov implemented a benchmark between the two database technologies using pg\_nosql\_benchmark. This is a great asset to this capstone project because MEAN works with and NoSQL whereas Django supports mostly SQL databases. The study pointed out that PostgreSQL performs around 2 to 3 times faster than MongoDB in selecting, loading and inserting data. MongoDB also consumed about one third more disk space compared to PostgreSQL. However, it is also important to aware that in order to get the best result each tool should be implemented in different scenarios. He suggested that NoSQL or MongoDB, in particular, is suitable when:

* High availability because setting replica servers and recovering from a node failure are easy and fast
* There is a need for data sharing because MongoDB provides an easy built-in solution for that.
* There is no stable schema for the database.
* NoSQL is also more scalable because these databases are designed to use with low-cost commodity hardware so its easier to scale horizontally.
* Data set is going to be big.

In contrast, some use cases for when it may be better to consider SQL DBMS are:

* There is a high load of inserting and updating.
* We have data consistency
* When there is a possibility to use complicated queries. If tables have a relationship with other tables so sometimes, we may want to join tables and retrieve all crucial information at once.
* RDBMS database is the appropriate option to solve ACID (atomicity, consistency, isolation, and durability) problems (Maksimov, 2016).

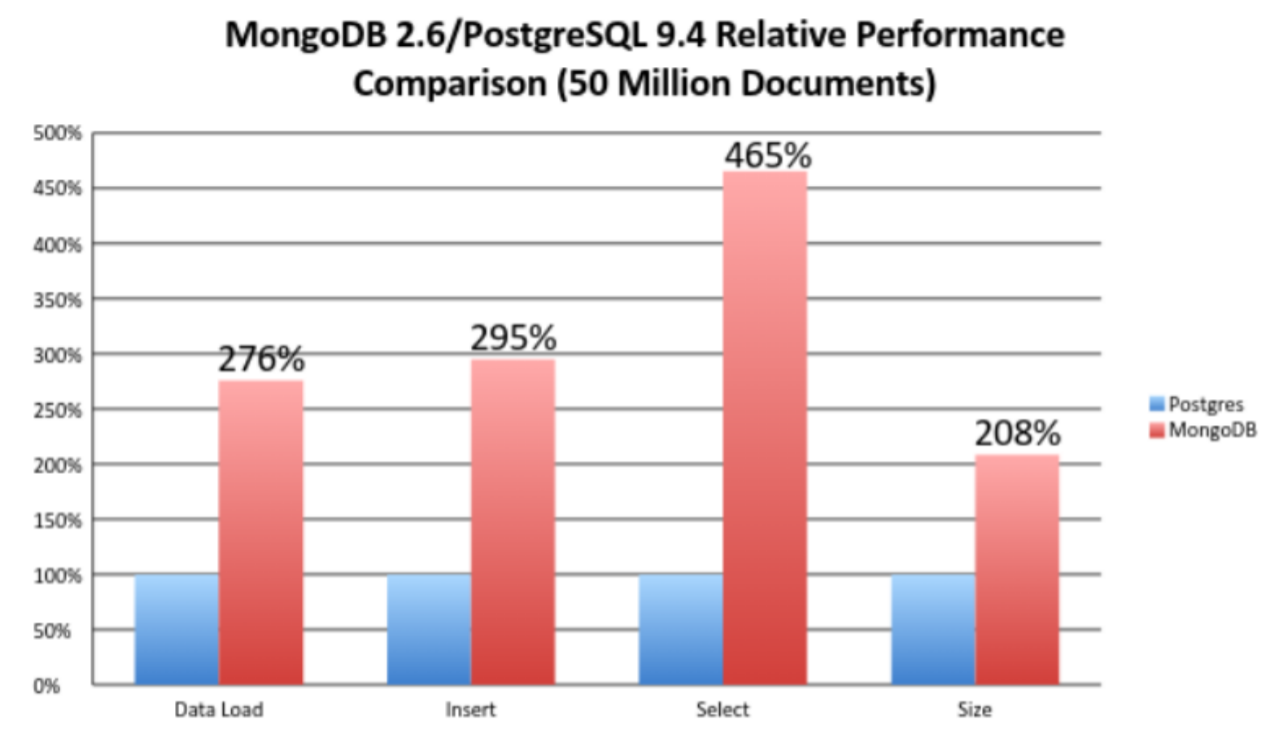


Figure 5. MongoDB 2.6 and PostgreSQL 9.4 Relative Performance Comparison (Maksimov, 2015).

Kaluzza, Troskot, and Vukelic mentioned about the trend of modern web application along with some popular front-end frameworks. The specific trends that this paper discussed are Multi-Page Application (MPA) and Single Page Application (SPA) and how frameworks can help to achieve SPA and MPA for web development. Multi-page application is a traditional way of developing web, which is about retrieving new pages form the server when receiving an HTTP request. That means every HTTP request from the client will generate a new page to show the result of that request. In contrast, Single Page Application refers to a web page that only has one page, but it behaves like a shell for other parts of the front end. In other words, page components or subsequence parts will be dynamically loaded into the homepage without actually reloading the whole web page.

Some advantages of SPA over MPA are that SPA will provide quicker response time due to no more reloading pages, faster data transfer and the ability to support offline because the connection to the server is not necessary all the time (Kaluzza, Troskot, and Vukelic. 2018). However, single-page applications have trouble with search engine optimization (SEO) which is about achieving a high rank in search engines because the crawling process in SEO is depended on the basis of URLs and page changes, but SPA applications only have one initial page. In return for faster loading pages, it may have slower initial loading time. Some other drawback is that the client-side must enable JavaScript and the development process for this type of application might be harder than the traditional approach. The paper also analyzed React, Vue and Angular as the 3 most popular front-end frameworks nowadays. Their research showed that Angular seems to have better utilities for developing single-page applications compared to the other two frameworks but is not suitable for multi-page applications due to the size, complexity and optimization capacity (Kaluzza, Troskot, and Vukelic. 2018).

1. **DATA COLLECTION**

Qualitative approach is used in this project to establish the accuracy, fairness as well as originality of the paper. Studies and comparison analysis of related materials in term of specification, performance, and statistics are collected for an objective point of view. However, each framework was created with a certain use case in mind, so a qualitative research is also needed to ensure that the statistic is not bias.

1. **FINDINGS**

**7.1 Performances**

Nowadays, when it comes to technology in general, and building websites, applications in specific. Performance is always one of the most important factors to evaluate the quality of a product. Users will likely turn away if a web page takes more than 3 seconds to load. Several seconds may not be a big difference as it sounds but in the world of software, it can make a big difference in sales. Therefore, it is common sense to take a deep dive into this aspect of MEAN stack and Django. Performance is judged by their process speed or has fast can they execute the code. It seems like Nodejs beat Django in performance for many reasons. Django is a Python framework and Nodejs is based on JavaScript and as a programing language, Python is already slower than JavaScript due to its dynamic and versatile nature in return of speed. When we take Nodejs into the discussion the difference in speed becomes even bigger due to the introduction of Chrome’s V8 engine which boosts the JavaScript runtime a bunch.

Researchers found that a major factor contributes to Nodejs performance is its components consist of a single thread, event loop, and asynchronous non-blocking I/O (Chhetri, 2016). I/O operation means data running from one place to another like memory to the hard drive, server to browser, etc. Non-blocking I/O, in a simple way, means that tasks that are involved in the input/output do not have the right to block other jobs. In computer processes, I/O tasks are usually the slowest. They are much slower than computation. For that reason, if I/O can run in such a way that it does not delay other computer tasks, the whole computing system will be less stagnant, or in other words, faster. The 2 ideas of non-blocking I/O and asynchronous combine together mean that events will occur in order of precedence as queued in programming.

However, if there is an event that has to wait for IO, then it will wait without preventing other tasks to be executed in the queue. However, there are pieces of evidence suggest that the drawback of Nodejs lies in heavy computation tasks. As shown in the study of Chhetri, Nodejs takes way more CPU usage compared to Apache PHP (Chhetri, 2016). However, the problem occurs whenever there is a heavy computation request, Nodejs would set up a block on other requests on the thread, causing an overall delay (Chrzanowska, 2017). Therefore, it is debatable on which one has better performance because in the normal case where we do not have heavy computing tasks then Nodejs will perform better than Django but if there are lots of heavy computations then Django may surpass Nodejs. Nevertheless, in recent updates, the “worker thread” module was introduced as an experimental feature designed to deal with heavy computations (Riva, 2019). Using this module, developers can spawn additional threads for parallel processing to take care of requests that block the event loop. As a result, it is safe to conclude that Nodejs' performance will still superior to Django.

**7.2 Ease to use**

One criterion that developers also usually look at is the ease to use. A major reason that made the name of MEAN full-stack so popular and widely used is the fact that software engineers can build the whole website from start to end with only JavaScript or TypeScript. In the field of web development, knowledge on HTML CSS and JavaScript is necessary because they have always been the technologies to build the interface of a website until now. On top of that, full-stack developers are also required to know a server-side language to take care of the server and behind the scene tasks. The languages of choices can be Java, PHP, Ruby, etc. With Nodejs, it allows JavaScript to be used also in the server-side so the problem of the multi-language paradigm is solved. Another potential strength of this is that it may unlock the connection between server-side and client-side developers. This helps prevent code duplication, fasten development time and many other benefits (Friesen, 2019). Nonetheless, Python, in general, is considered to have better and easier to understand syntax compared to any other language. This nature of Python allows writing code much less verbose but more extensible and modular, which also improves the readability and facilitates the cooperation of teamwork to focus on completing business logic rather than scaffolding code base. As a result, for people who do not have experience in JavaScript, Python or Django will be less challenging to absorb.

**7.3 Packages management**

As a matter of fact, when it comes to framework and programming languages, their efficiency is actually depended a lot on the number of libraries and modules available to help to address certain tasks. With the help of modules, this allows developers to share, use ready-made modules or reuse their own code and they do not have to write everything from scratch anymore. Therefore, another factor may have a great impact on the ease of development process is their library management system. In this category, we have Pip (Pip install python) and NPM (Node Package Manager). Both Python and Javascript provide tremendous libraries especially NPM. It is actually the biggest repositories of software packages. The reason for this is that JavaScript has a large community and NPM focuses on encouraging users to add new packages, so we have countless, prebuilt solutions for specific problems. (Crawford and Hussain, 2016). However, the Node Package Management registry seems to be quantity but not quality driven. While their core products are well tested, stable and well documented but the others are poorly structured (Humphrey, 2019). On the other hand, despite a smaller number of packages, Python or Django has a very rich standard library for server-side development that comes with it. This is also known as “Batteries included” meaning most of the tools and modules which are usually required for common cases are included when we install Django. Some that worth to mention are middleware, authentication, strong password checking, Http libraries and etc (Django Stars, 2019). This is not the case for Nodejs where developers need to have lots of experiences and knowledge about their packages in order to import and use appropriately.

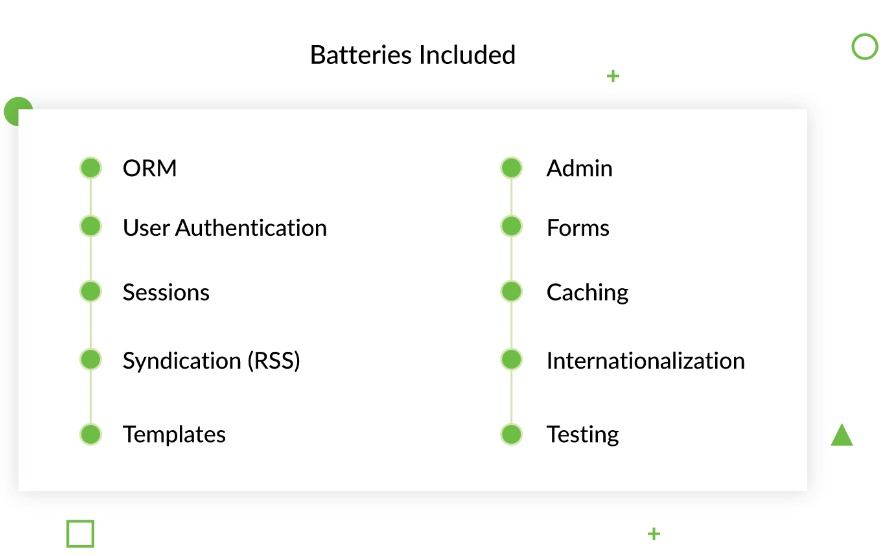


Figure 6. Batteries Include (Django Stars, 2019).

**7.4 Security**

Because Django has many necessary features that are already automatically implemented, it is perceived as having better security than Nodejs at first glance. First, let take a look at one of the most if not the most common cyberattack which is SQL injection. SQL Injection is a technique that attacks an application's database system (including web, mobile or desktop applications, etc.) through exploiting security holes for the purpose of retrieving, altering or even destroying the application's database system (DB Networks, 2016). This is considered the most common attack because the database is used almost for all technologies and also it is common for an application to have one or two pages where it requires the user to input in a certain form. To deal with this, Django has Queryset API, which is a module that can parameterize queries to avoid injection automatically (Boyer, 2018). Another type of attack that Django can take care of by default is Cross-site Scripting (XSS). Django templates can escape almost all characters and symbols that are usually seen in an XSS attack. The Python web framework also implements good user authentication, authorization system, and user, permissions management system by default. Any password in a Django application will also be hashed by default with the Password-Based Key Derivation Function (PBKDF2) (Boyer, 2018). Another security feature that comes with Django is CSRF protection. CSRF stands for Cross-site Request Forgery which is a technique that trick users into executing undesired actions on a web application (Owasp, 2018). In Django, developers can easily get away with this by just add {csrf\_token} tag inside form elements. Overall, Django has many implementations already in place to take care of common security issues.

On the other hand, Nodejs is minimalistic. It only comes with just enough stuff to make a server run (Boyer, 2018). Therefore, the security of applications built with Nodejs or MEAN stack depends a lot on external modules designed for the specific security problem. Luckily, as mentioned above, Nodejs has the biggest community with over 300,000 packages. However, to pick and understand how to use which modules can be another challenge for developers since not all of Nodejs modules are well written and documented. In contrast, a strong community and a huge number of libraries also bring risks. There are in fact many fake modules out there that contain malicious code. Many cases of phishing attacks occurred because developers use the wrong NPM packages. Attackers deployed NPM modules that look like legit thing with all nice documents and similar name but they actually contain malicious code (Tal, nd).

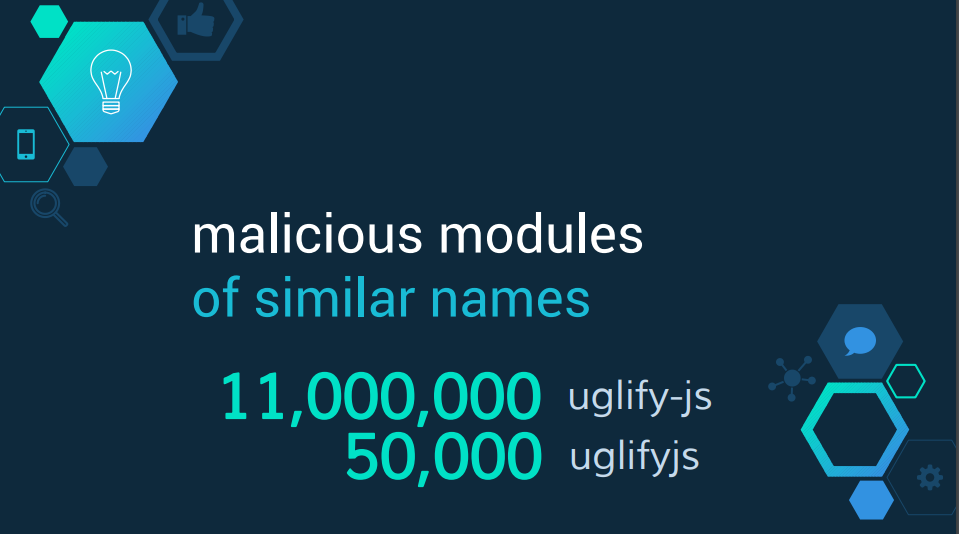


Figure 7: Phishing attack with similar name modules (Tal, L. nd)

**7.5 Single Page Application**

Single page application term is a rising term in the trend of web development so it is fair to consider this feature when it comes to web development frameworks. Many websites have applied this technique. Using Single-page Application (SPA) will be very effective to save bandwidth, as well as for better user experience. To be specific, the single-page app will make the web load faster and all the information will be contained inside one page, which makes it a great advantage. SPAs can support rich client-side functionality without requiring reloading the page when a user takes action or navigates. SPA can load faster, operations will respond faster because it does not have to reload the page fully. In addition, it also supports updating, saving form data or documents without the user pressing the “submit” button. SPAs can support rich client-side behaviors, such as drag and drop, much easier than traditional applications (Joseph, 2015). At the moment, some frameworks that can be used in order to achieve a single page application include Angular, React, Backbone, Ember and etc. Therefore, the MEAN stack seems to be a better choice for this kind of web application because it integrates Angular. According to Kaluzza, Troskot and Vukelic in 2018, Angular is the best front end framework for SPA development. Several reasons for why Angular becomes the most suitable framework for SPA are Angular does not have dependencies compared to other frameworks. Second is routing and navigation between pages in Angular is also simpler and direct. The two-way data binding mechanism of Angular enables the view to get updated whenever the model gets updated. And last but certainly not least, Angular also provides the ease for unit testing due to dependency injection and its CLI (Kaluzza, Troskot, and Vukelic, 2018). This is actually very significant because unit testing is the key component to making sure applications do not break. With that in mind, it is inevitable that a single page application has asserted its position in the web development market. Django is originally designed as primary a backend framework so using Django alone will not be able to obtain SPA. Django does have its own kind of front end framework which is Template Language but it is very bare-boned in functionality. Therefore, when it comes to front-end, it is safe to say that Django alone cannot provide good versatilities and features compared to the MEAN stack. One idea to address this is to integrate any JavaScript front end framework and let Django serve only as a backend service.

Overall, the findings can be wrapped up in the below table

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Django** | **MEAN** |
| Performances | * About 3 times slower than Nodejs. * Slow in general compared to other languages | * Fast * Handle concurrency requests very well * Bad for CPU utilization * There are module to address the CPU usage issue |
| Ease to use | * Better and easier to understand syntax. * Better readability * Easier to learn * Lots of necessary stuffs are already implemented | * Only one language for the whole website. * Depended a lot on modules |
| Packages management | * Strong community * Batteries included * Well documented | * Strong community * Huge number of packages * Only major ones have good documentation |
| Security | * Better security in general because lots of security features are automatically included such as authentication, middleware, strong password check, CSRF protection, XSS, SQL injection and etc | * Nodejs is minimal so there are lots of security holes * Need lots of experiences and understanding of its packages to set a strong security * High amount of libraries also comes with malicious and phishing packages. |
| Single page application | * Cannot achieve Single Page Application * The default template language for front end lacks of features * Can use with Angular to address this | * Very suitable for Single Page Application because of Angular |
| Scalability | * Even more scalable | * Scalable |

1. **CONCLUSION**

Both Django and MEAN stack are open-sourced, free to use and very sufficient to build a web application. Some of the advantages that worth mentioning of Nodejs are their high speed, high performance, huge libraries, huge community, great for building API and is wonderful at handling concurrent requests. Django, on the other hand, pays attention to readability so it is much easier to learn, provided “battery included” which leads to faster development and better security. Both tools have fairly good scalability and performance. While Django seems to have the advantage of scalability, Node.js has the advantage of performance. Django is highly scalable because the cache memory of applications can be easily done by tools like MemCache. NGINX can also be used to handle successful data migration even when the data becomes stronger. Node.js can allow multiple server requests and concurrent operations so it is great for collaborative applications like video conferencing, document or drawing editing, collectively known as Real-Time Application. Node's ability to handle heavy I / O operations and tamper with data between the front end and back end also makes it a great choice for SPAs (Single Page Application)

Besides, you should choose Django when development time has high priority. With a rich library of modules available, Django is very powerful in RAD (Rapid Application Development). With Python, we do not need to search for many modules and utility packages for the application. Python's simple and concise syntax makes writing code very fast.

1. **FUTURE WORKS**

For future works, I want to dig down deeper into the 2 frameworks and build some different kinds of the application so that I have a better insight on their features, strengths, and weaknesses. My interest is machine learning and I would like to know how well these frameworks can handle prediction models and big data.

1. **REFERENCES**

Joseph, J 2015. Single Page Application and Canvas Drawing. International Journal of Web & Semantic Technology (IJWesT) Vol.6, No.1. Retrieved from: <https://arxiv.org/ftp/arxiv/papers/1502/1502.03530.pdf>

Tal, L. nd. Node.JS Security Done Right. Owasp.org. Retrieved from: <https://www.owasp.org/images/f/fa/AppSecIL2016_NodeJS-Security_LiranTal.pdf>

Owasp. 2018. Cross-site Request Forgery (CSRF). Retrieved from: <https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)>

Boyer, J. 2018. How Secure are popular web frameworks? Comparing Options. Retrieved from: <https://hunter2.com/how-secure-are-popular-web-frameworks-comparing-options?fbclid=IwAR0z8kspHnad6AXkyfn7DyqiuwwJklYis2E6aUhsZ_FbXEVUvjJXFP4ohhQ>

DB Networks. 2016. SQL Injection Attacks: Detection in a Web Application Environment. Retrieved from: <https://www.dbcybertech.com/pdf/sql-injection-detection-web-environment.pdf>

Shaikh, N. 2019. Top Predictions for Web Application Development Trends in 2019. Retrieved from: <https://www.creative-tim.com/blog/web-development/top-predictions-web-application-development-trends-2019/>

Adhikari, A. 2016. Web Application Development with MEAN. Helsinki Metropolia University of Applied Sciences. Retrieved from: <https://pdfs.semanticscholar.org/547a/f8ea205a8cc7c02506f58c9599a447da07a7.pdf>

Chhetri, N. 2016. A Comparison Analysis of Node.js(Server-Side JavaScript). Culminating Projects in Computer Science and Information Technology. Retrieved from: <https://pdfs.semanticscholar.org/7b1e/b886dc76fa2d4312ec7aae745f69482fdc13.pdf>

Guardia, C. 2016. Python Web Frameworks. O’Reilly Media. Retrieved from: <https://theswissbay.ch/pdf/_to_sort/O%27Reilly/python-web-frameworks.pdf>

Riva, M. 2019. Going Multithread with Node.js. Retrieved from: <https://itnext.io/going-multithread-with-node-js-492258ba32cf>

Patil, S. Pinyoununtapong, E & Zhang, Y. 2016. React vs AngularJs. The Innovation University Institute of Technology. Retrieved from: <https://web.stevens.edu/softwareanalytics/students-projects/ComparisonStudyoftwoJavaScriptFramework.pdf>

Django Stars. 2019. Top 14 Pros of using Django for Web Development. Retrieved from: <https://djangostars.com/blog/top-14-pros-using-django-web-development/>

Friesen, J. 2019. The Incredible Advantages of Full Stack JavaScript. Retrieved from: <https://medium.com/@jacobfriesen/the-incredible-advantages-of-full-stack-javascript-ba8a2d42626f>

Kaluzza, M. Troskot, K & Vukelic, B. 2018. Comparison of front-end frameworks for web applications development. Retrieved from: <https://hrcak.srce.hr/file/294389>

Humphrey, J. 2019. Pros & Cons of Node.js (Nailing you coding inverview). Retrieved from: <https://simpleprogrammer.com/pros-cons-of-node-js/>

Brown, E. 2014. Web Development with Node and Express. O’Reilly Media, Inc. Retrieved from: <http://www.vanmeegern.de/fileadmin/user_upload/PDF/Web_Development_with_Node_Express.pdf>

Maksimov, D. 2015. Performance Comparison of MongoDB and PostgreSQL with JSON types. Tallinn University of Technology. Retrieved from: <https://digi.lib.ttu.ee/i/file.php?DLID=2983&t=1>

Chrzanowska, N. 2017. Why to Use Node.js: Pros and Cons of Choosing Node.js for Back-end Development. Retrieved from: <https://www.netguru.com/blog/pros-cons-use-node.js-backend>

Mardan, A. 2014. Express.Js Guide. Leanpub. Retrieved from: <https://pepa.holla.cz/wp-content/uploads/2016/08/Express.js-Guide.pdf>

Vainikka, J. 2018. Full-stack web development using Django REST framework and React. Metropolia University of Applied Sciences. Retrieved from: <https://www.theseus.fi/bitstream/handle/10024/146578/joel_vainikka.pdf?sequence=1&isAllowed=y>

Crawford, T & Hussain, T. 2016. A Comparison of Server Side Scripting Technologies. Kutztown University of Pennsylvania, Retrieved from:<https://csce.ucmss.com/cr/books/2017/LFS/CSREA2017/SER3291.pdf>

Mimi Gentz. NoSQL vs SQL [online]. Microsoft Azure official website; 24 June 2016 URL: <https://azure.microsoft.com/en-us/documentation/articles/documentdbnosql-vs-sql/>

Rathanam, L. 2018. DynamoDB vs. MongoDB: Which is the better NoSql Option?. Retrieved from: <http://techgenix.com/dynamodb-vs-mongodb/https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/06/Paper.pdf>

Fernando Monteiro. Learning Single-Page Web Application Development. Packt Publishing; 2014

Gleb, B & Vlad, V. 2019. Technology Stack for Web Application Development. Rubygarage. Retrieved from: <https://rubygarage.org/blog/technology-stack-for-web-development>

Ganguly, S. 2019. Most Popular Technology Stack To Choose From: Full Stack Vs. MEAN Stack Vs. MERN Stack In 2019. Hackermoon. Retrieved from:https://hackernoon.com/most-popular-technology-stack-to-choose-from-full-stack-vs-mean-stack-vs-mern-stack-in-2019-d12c0a17439a

Solovei, V. Olshevska, O & Bortsova, Y. The Difference Between Developing Single Page Application and Traditional Web Application Based On Mechatronics Robot Laboratory Onaft Application <https://journals.onaft.edu.ua/index.php/atbp/article/view/874/950>

Bulatovych, D. 2019. Choosing a Tech Stack for the Full-Cycle Web Application Development. Yalantis. Retrieved from: <https://yalantis.com/blog/tech-stack-for-web-app-development/>

Holmes, S. 2016. Getting MEAN with Mongo, Express, Angular, and Node. Manning Publications Co. ISBN: 9781617292033. Retrieved from: <https://www.programmer-books.com/wp-content/uploads/2018/08/Getting%20MEAN%20with%20Mongo,%20Express,%20Angular,%20and%20Node.pdf>

Rubio, D. 2017. Beginning Django: Web Application Development and Deployment with Python. Apress. ISBN:9781484227862. Retrieved from: https://cityusea.skillport.com/skillportfe/assetSummaryPage.action?assetid=RW$39622:\_ss\_book:135425#summary/BOOKS/RW$39622:\_ss\_book:1354