Components of Mean Stack for Web Development

Akash Kumar Gupta[1] Raman Chauhan[2]
[1,2]Student
[1,2]Master of Computer Application
[1,2]Uttaranchal University, Dehradun, India

Abstract— In recent years, web applications have become increasingly important in the industry. Web applications make it easier for organisations to grow and manage their operations, helping them to achieve their goals faster. A web application may involve a variety of tasks, including product requirements, design, coding, and testing with frameworks, databases, and operating systems. This research focuses on the web stack, specifically the MEAN stack. A Web stack is the software required for Web development. A Web stack consists of an operating system (OS), a programming language, database software, and a Web server. Throughout history, traditional stacks have been used as the development model for the majority of online applications. The purpose of this research is to plan and construct an email framework as part of a Telemedicine application using MongoDB, Express, Node.JS, and Angular. These unique aspects' benefits are growing, and they can be combined into a single stack. The four components of the MEAN stack (MongoDB, Express, Angular, and Node.Js) are examined in this paper, as well as how they function together and the benefits they provide as a whole in web development. In order to better understand how these four MEAN stack web development breakthroughs work, this article also shows the work process and worker design in detail. The primary focus of this research is on the responsibilities of these four breakthroughs in the MEAN stack, as well as how they are famously implemented in present times.

Keywords: Web Development, MEAN, MongoDb, Express, Angular, Node.js, LAMP

I. INTRODUCTION

MongoDB, Express, Angular, and Node.JS (MEAN) are four programming components designed by Haviv (2015) and Alfred (2014). Regardless of the abbreviation, JavaScript is at the heart of these four categories and the foundation of current application development. Mean is a full-stack JavaScript creation that will aid modern designers in speeding up the structure interactivity of web and mobile apps. It's tough and simple to keep up with.

However, any frontend engine of its type, such as EJS or REACT.JS, can be used in place of AngularJS. JADE is the default layout motor for AngularJS in most circumstances. JavaScript Object Notation is used to communicate amongst all components of the MEAN stack (JSON). As a result, no interpretation is required on the part of the consumer, the worker, or the JSON record Object. Using the same language throughout the stack has several advantages, including better efficiency, increased profitability (Chris, 2015), and decreased memory usage, which is critical in every code execution measure.

Because each segment understands JSON, including the MongoDB data set, which understands the twofold interpretation of JSON, such as BSON, it is also simple to update information fast and concurrently in MEAN. MongoDB allows you to store and query BSON without changing the structure or meaning of the data. The adaptability of MongoDB is one of the most significant advantages of employing it in this inquiry. Figure 1 depicts the JSON language across the M.E.A.N Stack. The Node Environment performs operations and sends requests to the MongoDB database worker using JavaScript, while the Web worker (Node.JS) sends the resulting HTML page to the frontend.

Frontend < - >JSON< - > Backend < - > JSON< - >Database

AngularJS< - >JSON< - >Node.JS/Express.JS< - >BSON< - >MongoDB (NoSQL)

Rather than being a single innovation, the web is a collection of web systems, programming dialects, builds, and libraries. According to the experience of the last 25 years, the LAMP stack, the.NET stack, and a wide range of other systems and apparatuses were employed in the production of various types of web applications. The fundamental issue with these stacks, however, is that each level needs an information base that frequently exceeds the capabilities of a single designer, which has a disadvantage, such as necessitating development organisations to continue developing with diverse talents. As a result, efficiency suffers and sensitivity to unforeseen dangers increases.

A Linux or Windows Operating System (OS), a Web worker (Apache), a data set (RDBMS - MySQL), and a programming language for rendering dynamic HTML pages, such as PHP, PERL, or Python, make up the Linux/Windows Apache MySQL and PHP (L/WAMP) stack. However, these web development advancements that were set up as a single step were not planned to work together at first (David, 2002).

This investigation focuses on the Mean stack, which is a full JavaScript innovation stack that is being worked on by a huge community of designers and promoters to ensure that each component is always progressing to higher levels, as well as relevant documentation (Onodi, 2016). The innovation includes a NoSQL data set, a web worker system, a web customer structure, and a worker stage. The strength of the Mean stack is dependent on the use of JavaScript as the major programming language (Elrom, 2016).

The MEAN stack is an open source collaborative project that connects the advancement of the three-level web MVC design and uses JavaScript as the programming language throughout. The features of Node.JS are similar to those of Apache, but it is more user-friendly. Rather than forwarding the programme to a separate webserver, Node.JS is remembered for the application and utilised to run it using the current version of the web worker and some run-time conditions. Chris (2015) separated the application into two parts: the customer side and the labour side.

The consumer side is made up of the frontend, which gets data from the worker side. The worker side model uses Node.JS to transmit the assets to the aid requester. Node.JS has been continually evolving and improving in general. The worker side stage, as demonstrated by Perrenourd, allows designers to construct their own web worker as well as quick, customised apps using JavaScript (2015). Node.JS is powered by the V8 chrome motor, an open source JavaScript motor created by Google Corporation. Node.JS is a hybrid of JavaScript and C++, with JavaScript accounting for 40% of the code and C++ accounting for 60%.

To help with increased execution, throughput, and adaptability, Node.JS uses event-driven engineering and a non-impeding I/O strategy. This is distinct from a simultaneousness concept based on strings. Node.JS is the greatest choice for data-driven apps and ongoing application development since it can handle a wide range of business relationships. This research presents a hypothetical scientific classification and scenario for evaluating MEAN stack innovation in the advancement of web application frontend and backend. It also lists the advantages that engineers should be aware of when comparing the LAMP stack to other stacks. As part of the investigation, a messaging application was constructed and deployed on the Heroku (PaaS) platform, demonstrating the power of MEAN stack innovation in the production of speedy, proficient, and adaptable web apps.

II. COMPONENTS OF MEAN STACK

A. *MongoDB*

MongoDB, an open source V8-based database, was established by Dwight Merriman and Eliot in 2009. MongoDB's name comes from its enormous scalability. The NoSQL database outperforms traditional databases in terms of scalability and has a BSON-like data architecture with dynamic schemas. MongoDB's popularity has grown as a result of its ability to provide developers with the flexibility they require when working with complex data, as well as the capacity to scale queries significantly more than Relational Database Management Systems (RDBMS) (Rick, 2010). MongoDB makes a lot of transformation logic easier to understand (Yadav et. al., 2019).

It's a popular document-oriented database that uses NoSQL and has persistent storage. BSON is a data transfer standard that allows you to send data from a database to a server and subsequently to a client (Ihrig & Bretz, 2015). It also expresses its model through codes. MongoDB uses aggressive caching techniques and, when possible, overwrites updated entries to provide speedy and efficient updates (Dirolf, 2010). The most serious flaw is that it is not readable.

Mongo Lab (mlab) was used to show how to administer MongoDB databases on the cloud. With impedance mismatch difficulties, this substantial improvement has exhibited a higher level of control than relational database management systems (RDBMS). MongoDB is often used in cloud contexts and gives more benefits than traditional databases due to the lack of native libraries and drivers. MongoDB, like other NoSQL databases, uses object representations rather than storing items in several tables with complex representations or schema.

MongoDB, like other NoSQL databases, may be clustered and scaled to handle increased application load. High-availability data is provided by its REST-enabled Hypertext Transfer Protocol (HTTP) APIs. The Mongoose model connects MongoDB and NodeJS, providing the essential data structure while preserving data storage and retrieval flexibility. Mongoose is an Object Data Modeling (ODM) library for MongoDB.

B. *ExpressJS*

ExpressJS is a server-side web application framework that aids developers in arranging web applications using the MVC paradigm. Everything from routing to handling requests and views is handled by it. Express.JS makes it simpler to develop safe, modular, and fast Node.JS apps that are more efficient, reliable, and have fewer duplications. Express.JS stores sessions in memory and organises web applications by hiding the majority of Node.inner JS's workings. Modular HTML template engines, expanding the response object to allow various data format outputs, routing system, enhanced project structure, proper application configuration, and dividing its logic into multiple modules are some of its characteristics. T.J Holowaychuk created this framework within the JavaScript ecosystem to protect node.JS and make it easier to create speedier Single Page Applications (SPA).

C. *Angular.JS*

Angular.JS, or simply Angular, is a frontend framework for creating single-page applications (SPAs) that load in a single page. It continued to streamline as a result of the utilisation of Ajax call techniques and multiple page loading. With Angular, two-way data binding between views and models is possible. This is useful for developers in terms of timetables, and HTML makes it much easier to place tasks on a template (Adam and Colin, 2014).

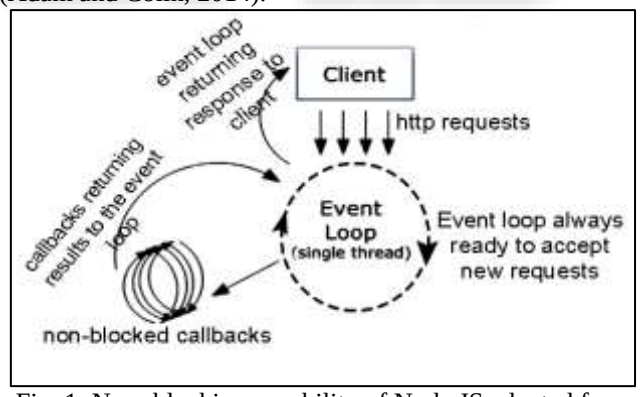

Fig. 1: Non -blocking capability of Node.JS adapted from Mathieu (2017)

D. *Node.JS*

Node.JS is not a framework; rather, it is a very low-level programme, similar to the C programming language that is used in the development of client-side and server-side applications that demand quick and efficient delivery, such as real-time and server applications. Node.JS executes JS code by translating it into machine language and optimising it through sophisticated methods such as code in line, copy elision, and so on, using Google Chrome's super-fast highly optimised V8 execution engine in a JIT (Just in Time) compilation way. The following are some of the features of

node.JS: single thread, demonstrating concurrent execution of instructions without deadlock, Node.js shows non-blocking to optimise throughputs and has a main loop that listens to events and then initiates a call back function to handle concurrent requests up to ten thousand connections and more (Kenneth, 2015). Its code methodology is thought to be critical in preventing users from wasting time by allowing them to do other things while waiting for lengthy activities to finish. NodeJS has a strong community and a long history of rapid growth. It contains extensive libraries in its Node Package Manager (NPM) enabling community sharing and publication of open-source materials (Fhala & Chrispinus, 2017). Two essential characteristics of the development component are shown in the diagram (Fig. 2). The server is run with Node.JS, which is single threaded, asynchronous, and uses an event-driven design. This property is one of the key reasons why Node.JS is so quick, accurate, and resource-efficient. The blocking I/O model is used by several well-known web servers like as Apache, Ngnix, and IIS. This approach handles requests and avoids concurrency concerns by using many threads. As a result, when a new connection is made, it will be served by a distinct thread with a set amount of RAM. Because threads must wait for I/O while the server processes and retrieves data, blocking I/O adds complexity and overhead to the server.

Node uses a single processing thread and asynchronous I/O to handle all requests, using the power of JavaScript. There is no latency for I/O or related context switching processes due to asynchronous or non-blocking I/O architecture. Rather than giving each connection its own thread, Node connects them all to the same thread and keeps an event loop going. The loop searches for new events and passes them on to certain handler routines, while NodeJS continues to run other processes and calls these handler methods to finish the execution of these other events (Yadav and Chandra, 2018).

III. EASE OF USE AND IMPLEMENTATION

JavaScript started out as a simple script that was meant to be run by the browser. On the other hand, JavaScript is now widely used. Smartphones, computers, Raspberry Pis, and a multitude of other technological breakthroughs all contain it. JavaScript is non-blocking, which gives it an advantage over other languages. A single non-blocking thread in JavaScript is more efficient than threads in other languages.

JSON is the industry standard for data exchange between the four layers. Because JSON is native, there is no need to parse it. JavaScript can easily consume JSON because it is a lightweight format. The most frequent and effective approach to use the MEAN stack is to utilise express to construct a Restful API, and then angular to handle client-side routes, taking full advantage of Angular's SPA capabilities. Only when data from the database is necessary will the application be needed to use the API. This method can be used to apply and execute the majority of business logic on the client side.

On the Express side of things, app handlers are utilised to handle requests and offer responses. These handlers accept the request and, with the help of the middleware, start the request-response cycle. Express handles user administration, authentication, session management, and CRUD operations on MongoDB. Technology development may be inhibited if learning is too difficult and the costs outweigh the advantages. These four technologies, however, work together in the MEAN stack; for example, an express response object can be sent immediately to Angular without the requirement for parsing. MEAN.io and MEAN.js are two popular Node packages that come pre-compiled with all four technologies and may be utilised right away. Because some portions of the integration are already automated out of the box, this makes it much easier for developers (Praveen and Chandra, 2017).

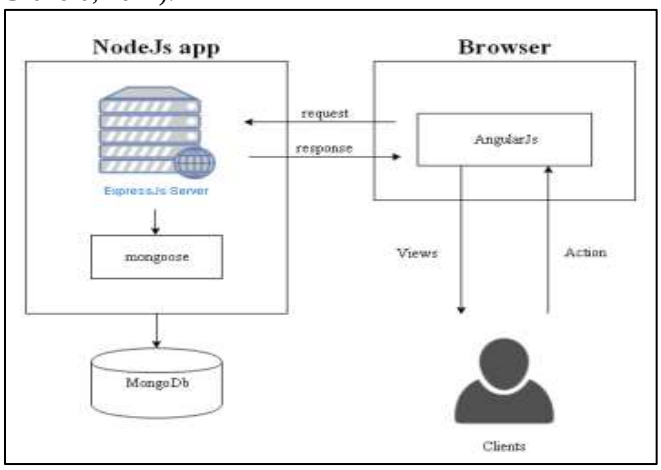

Fig. 2: Workflow of MEAN stack.

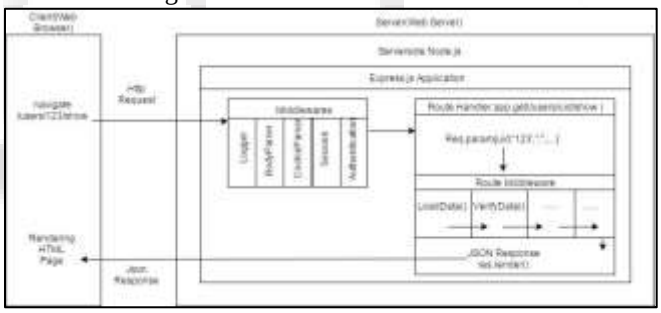

Fig. 3: Server architecture of Express.js

IV. ADVANTAGES OF USING MEAN STACK IN WEB DEVELOPMENT

MEAN has an advantage in designing web applications for a variety of reasons. They are as follows:
Unlike relational database management systems (RDBMS), MongoDB is a NoSQL BSON styled document-oriented database that requires neither conversion nor relational models. Because it was designed for the cloud, it enables better flexibility an accommodating layer for storing data dynamically (Grover, 2015). It solves the impedance mismatch problem that RDBMS suffers from due to their rigid structure (Peter, 2017) and has massive performance consequences while processing queries.

- Unlike the frontend of LAMP, which can operate with more languages and components than the backend, MEAN works with JavaScript technologies and delivers complete frontend development with less difficulty.
- The MEAN application is run on a Node.JS server. It's an event-driven I/O server-side JavaScript environment

that can handle data-intensive and real-time tasks better than Apache, IIS, and Ngnix.
- The LAMP stack and comparable stacks limit the operating system to a specific version. There are no such limitations in the MEAN stack. MEAN allows node.JS to run on any operating system.
- Mean is completely open source and has a strong community behind it.
- MEAN data transfer format is JSON, and having the same language throughout the stack ensures consistency.

V. LIMITATIONS OF USING MEAN STACK IN WEB DEVELOPMENT

- Converting any task from LAMP to MEAN stack isn't easy because it necessitates rewriting the current JavaScript code.
- MEAN stack isn't appropriate for projects with large source code because, as a group grows in size, it becomes more difficult to keep track of and analyse the code in JavaScript because it can be difficult to tell which lines of code are for the front-end and which are for the back-end.
- MongoDb isn't recommended for storing and organising large amounts of data. Working with it effectively may necessitate the assistance of a skilled professional.
- Because Node.js requires a dedicated application interaction to execute, it will be difficult to transport the entire MEAN application on shared hosting.
- Because JSON lacks mapping and namespace capabilities, it isn't nearly as reliable as XML for migrating data across different frameworks.
- Engineers should receive appropriate training in mongo because it differs significantly from a traditional social data store. There are no connections, thus information dissemination should take this into account. Many developers miss this and copy and paste the present social data base into MongoDb, languishing over it.
- Node.js runs on a single string and eliminates concerns with concurrency. Regardless, if simultaneity is essential, the software engineers must deal with it.
- Compared to classic battle-tested libraries like PHP, JAVA, and others, the development level of Node and npm packages is minimal. It's difficult to find new solid bundles because anyone can create one in the npm library. Several bundles have stabilised as the years have passed, but it's possible that it's still looking for new dependable bundles.

VI. CONCLUSION

LAMP and its variants have been a true innovation that has resulted in a slew of beneficial applications all around the world, and this cannot be overstated. In today's technology, JavaScript, on the other hand, is a trailblazer in web application development. Another cutting-edge and powerful creation that will soon disrupt the web development phases is the MEAN stack, which is named for its components. Any stack can be utilised depending on the developer's needs, however the MEAN stack has proven to be a viable choice for applications that need to be fast, versatile, and consistent. The non-obstructive approach of Node.JS made dealing with simultaneity simple. With the use of the JSON-BSON information design, the MongoDB report prepared No-SQL data set has exhibited greater adaptability and versatility in arising innovations. Engineers needed a simple system, therefore Express.JS was designed on top of Node.JS. AngularJS was created by Google to better the MVC technique for constructing SPAs and to quickly build strong intuitive UIs.

REFERENCES

[1] T.Swathi, Y. Yashaswini, S. Suraj and B. N. Kiran, "Ease of MEAN Stack in Web Development", International Journal of Current Trends in Engineering & Research (IJCTER), vol. 2, no. 5, pp. 2117-219, May 2016.
[2] A.Leff and J. T. Rayfield, "Web-application development using the Model/View/Controller design pattern", Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference, pp. 118-127, 2001.
[3] I.K. Chaniotis, K.-I. D. Kyriakou and N. D. Tselikas, "Is Node.js a viable option for building modern web applications? A performance evaluation study", Computing, pp. 1-22, 2014.
[4] Parker, S. Poe and S. V. Vrbsky, "Comparing NoSQL MongoDB to an SQL DB", Proceedings of the 51st ACM Southeast Conference on - ACMSE'13, 2013.
[5] T. Fielding and R. N. Taylor, "Principled design of the modern web architecture", Proceedings of the 22Nd International Conference on Software Engineering, pp. 407-416, 2000.
[6] HTML5 differences from HTML4, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/html5-diff/, accessed 5. 7. 2010
[7] S.Tilkov and S. Vinoski, "NODE. JS: Using JavaScript to build high performance network programs", IEEE Internet Computing, vol. 14, no. 6, pp. 80-83, November-December 2010.
[8] Kai Lei, Yining Ma, Zhi Tan. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js. Computational Science and Engineering (CSE), IEEE 17th International Conference on; 2014: p.661-668.
[9] HTML5, A vocabulary and associated APIs for HTML and XHTML, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/html5/, accessed 5. 7. 2010
[10] JavaFX, http://javafx.com/, accessed 5. 7. 2010
[11] WHATWG Wiki FAQ, http://wiki.whatwg.org/wiki/FAQ, accessed 5. 7. 2010
[12] HTML Microdata, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/microdata/, accessed 5.7. 2010.
[13] A. Trivero: Abusing HTML 5 Structured Client-side Storage, http://trivero.secdiscover.com/html5whitepaper.pdf, accessed 5.7.2010
[14] Web Designer's Checklist, http://www.findmebyip.com/litmus/, accessed 5. 7. 2010
[15] SQLite Home Page, http://www.sqlite.org/, accessed 5. 7. 2010

[16] WebSimpleDB API, W3C Working Draft, 29.9.2009, http://www.w3.org/TR/WebSimpleDB/, access. 5.7. 2010
[17] Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Working Draft, 15. 12 2009, http://www.w3.org/TR/wai-aria/, accessed 5. 7 2010
[18] HTML5, Web Development to the next level, html5rocks.com, http://slides.html5rocks.com/, accessed 12. 7. 2010
[19] Cascading Style Sheets, Current Work http://www.w3.org/Style/CSS/current-work, accessed 5. 7. 2010
[20] Offline Web Applications, W3C Working Group Note, 30. 5 2008, http://www.w3.org/TR/offline-webapps/, accessed 5. 7. 2010
[21] W. Peng, J. Cisna: HTTP cookies – a promising technology Online Information Review, Vol. 24, No. 2, 2000 [14] Web Storage, Editor's Draft, 15. 6. 2010, http://dev.w3.org/html5/webstorage/, accessed 5. 7. 2010
[22] A. Trivero: Abusing HTML 5 Structured Client-side Storage, http://trivero.secdiscover.com/html5whitepaper.pdf, accessed 5.7.2010
[23] Web Designer's Checklist, http://www.findmebyip.com/litmus/, accessed 5. 7. 2010 [17] SQLite Home Page, http://www.sqlite.org/, accessed 5. 7. 2010
[24] Indexed Database API, W3C Working Draft, 5. 1. 2010, http://www.w3.org/TR/IndexedDB/, accessed 5. 7. 2010
[25] WebSimpleDB API, W3C Working Draft, 29.9.2009, http://www.w3.org/TR/WebSimpleDB/, access. 5.7. 2010
[26] Web Workers, Editor's Draft, 25. 6. 2010 http://dev.w3.org/html5/workers/, accessed 5. 7. 2010 [24] The Web Sockets API, W3C Work. Draft, 22. 12. 2009, http://www.w3.org/TR/websockets/, accessed 5. 7. 2010 [25] T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web, Scientific American Magazine, 2001
[27] Praveen, S, Chandra, U. NoSQL products: IT giants perspectives. Int J Computation Intell Res 2017; 13: 2125–2133.
[28] Dharminder Yadav, Umesh Chandra , " Modern Technologies of Big Data Analytics: Case study on Hadoop Platform " , International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 6, Issue 4, 2018, pp. 044-050.
[29] Dharminder Yadav, Himani Maheshwari and Umesh Chandra, "Big Data Hadoop: Security and Privacy", Proceedings of the 2nd International Conference on Advanced Computing and Software Engineering Applications, 2019.

