International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 02 | February - 2024 SJIF Rating: 8.176 ISSN: 2582-3930

# COMPARATIVE ANALYSIS OF MERN, MEAN AND MEVN STACKS IN WEB DEVELOPMENT

**Jayant Baid[1], Sapna Gupta[2]**

[1,2]*Department of ITE, Maharaja Agrasen Institute of Technology*

**Abstract -** Choosing a technology stack is a critical decision that affects how software projects progress in the ever-changing field of web development. Regarding web development, a technology stack is a set of tools, frameworks, and programming languages that are utilized in the construction of an application's front end and back end. In addition to making the development process easier, these stacks provide the groundwork for the finished product's performance, scalability, and maintainability. JavaScript-based stacks are becoming serious contenders as businesses and developers search for frameworks that provide efficiency, scalability, and flexibility. Starting with a thorough examination and comparison of three popular JavaScript stacks, this research paper examines MEAN (MongoDB, Express, Angular and Node.js), MERN (MongoDB, Express, React and Node.js) and MEVN (MongoDB, Express, Vue.js and Node.js).

***Keywords*:** Web Development, MERN Stack, MEAN Stack, MEVN Stack, ReactJS, AngularJS, VueJS

## 1. INTRODUCTION

In the dynamic field of web development, the choice of a technology stack plays a crucial role in shaping the course of software projects. A technology stack, in the context of web development, refers to a combination of programming languages, frameworks, and tools used to build the front-end and back-end of an application[1]. These stacks not only facilitate the development process, but also lay the foundation for the performance, scalability, and maintainability of the final product. As companies and developers look for frameworks that offer flexibility, scalability, and efficiency, JavaScript-based stacks have become key contenders. This research paper begins with a comprehensive study and comparative analysis of three well-known JavaScript stacks: MEAN, MERN and MEVN.

The evolution of web development frameworks has led to a variety of technology stacks, each tailored to specific challenges and preferences. MEAN, MERN and MEVN are characterized by their unique combinations of frontend and backend components and have received widespread attention and acceptance in the developer community. To make informed decisions in the development landscape, it is important to understand their individual characteristics, strengths and potential trade-offs.

The goal of this research is to provide a detailed study of the architectural foundations, component interaction, and development environments of each stack. By exploring the intricacies of database integration, with a particular focus on MongoDB paired with AngularJS, React and Vue.js, the paper aims to uncover nuances that impact application design and performance. In addition, an in-depth analysis of the front-end frameworks (AngularJS, React, Vue.js) and the server-side

framework (Express.js) forms the core of our comparative study.

As we navigate the specifics of these technology stacks, considerations such as scalability, performance optimizations, and community support are at the forefront. Real-world case studies are examined to provide practical insights into the application and success stories associated with each stack. The culmination of this analysis is summarized in recommendations tailored to specific use cases, providing developers and decision makers with a guide for navigating the complex landscape of technology stack selection.

Looking forward, the paper will also explore possible trends and considerations that could influence the development of these stacks. By completing this research, readers will have a nuanced understanding of the MEAN, MERN, and MEVN stacks, empowering them to make informed decisions tailored to their project needs and development preferences.

## 2. DECODING THE MERN STACK

The MERN stack, consisting of MongoDB, Express.js, React and Node.js, represents a powerful and cohesive set of technologies in the field of web development. MongoDB serves as a foundational NoSQL database and provides a flexible and scalable solution for storing Data in JSON-like documents[2]. Its ability to handle unstructured or semi-structured data, coupled with horizontal scaling capabilities, has contributed to its widespread adoption, particularly in applications that require dynamic and evolving data structures.

Express.js, the server-side framework of the MERN stack, is a minimalist and flexible web application framework based on Node.js. Express.js simplifies the process of developing robust and scalable server-side applications by providing essential functionality and middleware support for processing HTTP requests and responses[3, 8]. The unbiased nature of Express.js allows developers to tailor their architecture and tool selection to the specific needs of their projects.

React, a JavaScript library developed and maintained by Facebook, takes on the role of the frontend library in the MERN stack. React is known for its declarative and efficient approach to building user interfaces and promotes a component-based architecture. This design paradigm allows developers to create modular and reusable UI components, promoting code maintainability and scalability. The virtual DOM implemented by React contributes to improved performance by efficiently updating the actual DOM, ensuring a seamless user experience in single-page applications[4].

Node.js, which serves as a JavaScript runtime, unifies the language across the MERN stack and allows developers to use JavaScript for both frontend and backend development[3]. Node.js is event-driven and designed to efficiently handle asynchronous I/O operations and plays a central role in the architecture of the stack[1]. This unification not only simplifies the development process, but also makes code easier to reuse and maintain as developers can leverage their existing skills across the stack.

The integration of these components through the MERN stack offers significant advantages in web development. Its commitment to full-stack JavaScript enables seamless collaboration between frontend and backend teams, promoting code consistency and shortening the learning curve[9]. React's component-based architecture improves code modularity and maintainability, contributing to a more scalable development process. Additionally, the stack benefits from a robust community that ensures ongoing support, a wealth of libraries, and readily available solutions to common development challenges. The scalability of MERN applications is further enhanced by MongoDB's horizontal scaling capabilities, making it well suited to handling increased data volumes in dynamic and evolving projects[5]. Understanding the intricacies of each component within the MERN stack provides developers with the knowledge necessary to make informed decisions and effectively leverage the stack's strengths in various web development projects.

## 3. DECODING THE MEAN STACK

The MEAN stack has become a popular open-source web development stack consisting of the MongoDB database, the Express web application framework, the AngularJS frontend framework, and the Node.js runtime environment. MEAN has enabled full-stack JavaScript development, allowing one programming language to be used across the entire technology stack.

MongoDB has been used as the default database for MEAN stack applications due to its flexibility and scalability as a document database. Instead of requiring predefined schemas, MongoDB used JSON-like documents to store hierarchical data without restricting the structures in advance. Thanks to its easy replication features and horizontal scaling, MongoDB is well suited to processing large amounts of real-time data[8].

The Express web framework was used in the MEAN stack to facilitate server-side web application logic and API endpoint processing in Node.js backend environments. Middleware capabilities, routing, and a wide range of third-party plugins have enabled faster development of services and business logic required to connect the database and frontend layers in full-stack MEAN apps[8].

AngularJS provided an MVC-based frontend JavaScript framework for the MEAN stack, focused on dynamic views and two-way data binding[4]. Built-in directives and dependency injection as well as comprehensive testing tools have made AngularJS a complete client-side framework for developing complex, reactive user interfaces that connect seamlessly with backend Express and MongoDB components within MEAN's JavaScript environment.

## 4. DECODING THE MEVN STACK

The MEVN stack has become a popular open-source, full-stack JavaScript framework consisting of MongoDB, Express.js, Vue.js, and Node.js technologies. This pure JavaScript stack provides developers with a streamlined way for end-to-end JavaScript application development, helping them use one language for server, database, and frontend.

The document-oriented database MongoDB serves as MEVN's flexible and scalable database layer. JSON-like data structures, called BSON documents, have been used by MongoDB to store data without the need to define rigid schemas in advance. Easy horizontal scaling, replication capabilities, and native processing of unstructured data make MongoDB well-suited to processing large volumes of rapidly changing, real-time data that is commonplace in modern web and mobile apps built with MEVN.

Express.js handles the backend application logic and web server functions and has been used as a minimalist web framework for Node.js server environments within the MEVN stack. Useful features such as routing, middleware capabilities, and a plugin ecosystem have enabled faster server-side development with Node.js for API definition and real-time data streaming to clients.

Vue.js has emerged as a front-end framework for MEVN over React/Angular due to its phased adoption style. The Virtual DOM architecture and reactive components have made it significantly easier to perform partial upgrades or integrate Vue into existing projects compared to other frontend options. The result is powerful front-end development that easily connects to back-end services.

## 5. COMPREHENSIVE ANALYSIS

The database layers for the stacks have some similarities, but also important performance differences. The document-based storage model used by MongoDB has given all three stacks flexible schemas and high scalability as unstructured data grows over time. Additionally, native processing of JSON-like data enables seamless integration with Node and JavaScript frameworks. While older MEAN stack deployments experienced network delays when scaling databases, updated connection processing in the latest MERN and MEVN stacks using MongoDB has resulted in higher query and indexing throughput capabilities.

In terms of backend implementation, the Express web application framework for Node.js has usage across all three stacks, enabling clean model/view/controller-based organization and rapid API endpoint development. While Express itself has changed little across stacks and performs equally well, there are some differences in the way it has been integrated. For the MERN stack, the bindings for connecting the React frontend to Node/Express have been optimized through custom libraries. This has enabled greater efficiency in linking user interactions with server functions. The tighter coupling enables better performance for frequent frontend-to-backend requests that are common in modern web apps.

The approaches to render and update views taken by React, Vue, and AngularJS have fundamental differences in the use of

virtual and real DOMs. React and Vue, used by the MERN and MEVN stacks, implemented a virtual DOM in which views were first rendered to an in-memory representation before efficiently propagating batches of changes to the real DOM. This different process has led to immense performance improvements, but also to upfront coding effort. Meanwhile, the MEAN stack using AngularJS relied solely on the real DOM[6], which was easier to code but suffered from performance degradation in highly dynamic UIs. Benchmarks between Vue and React have shown that Vue maintains 60fps animations better, while react can stall with larger state mutations due to its recursive diffing approach.

In summary, the MERN, MEAN and MEVN stacks offer unique combinations of technologies that reflect the diversity of choices available to developers. MongoDB provides end-to-end data storage flexibility across stacks, while Express.js and Node.js contribute server-side functionality and unify the development environment. The choice of frontend frameworks React in MERN, AngularJS in MEAN and Vue.js in MEVN introduces different approaches that meet the evolving needs of web development in different scenarios. Each stack has found its niche: MERN emphasizes versatility, MEAN focuses on bidirectional data binding, and MEVN stands out for its simplicity and adaptability.

**Table-1**: Comparison of Mean, Mern, and Mevn Stacks

| Features | MERN | MEAN | MEVN |
|---|---|---|---|
| Frontend Framework | React | Angular | Vue |
| Server-Side Rendering | Yes | Yes | No |
| Learning Curve | Easy | Moderate | Easy |
| Ecosystem | Large | Large | Small |
| Development Speed | Fast | Fast | Slow |
| Flexibility | High | Limited | High |

## 6. SCALABILITY AND PERFORMANCE

The scaling of the MEAN stack is inherently limited by AngularJS's client-side rendering approach for single-page applications. As more users access the application, client-side processing introduces bottlenecks that cannot be mitigated as easily as server-side rendering. In addition, the AngularJS change detection mechanism must traverse the entire component tree every time data changes, which quickly slows down as complexity increases.

In contrast, both MERN and MEVN enable much better scalability using a virtual DOM localized to patches of components required by React and Vue updates[4]. This prevents unnecessary calculations in unaffected sections. Additionally, react was designed from the ground up to handle large amounts of data through normalized, flattened data requirements, leaving far fewer scalability limits. Finally, the advanced server-side rendering capabilities available in Next.js for React also improve MERN's ability to intelligently distribute processing across the server and client.

The performance of the MEAN stack depends heavily on the code quality and architecture of the AngularJS application, which is entirely based on the frontend[4]. Any non-optimized data processing, tracking, or UI updates result in direct performance delays for users. ANGULARJS's dirty checking digest cycle is also resource intensive because each cycle requires expressions to be evaluated across all components.

Alternatively, the virtual DOM implementations used by React and Vue allow them to minimize expensive DOM manipulations by comparing component trees and applying only differential updates. This enables excellent performance when building complex high-frequency user interfaces in MERN and MEVN. Additionally, React Fiber introduced incremental matching and priority-based rendering, achieving up to five times throughput for optimal use of CPU capacity. Next.js provides MERN server-side rendering capabilities that enable first-time page loading without JavaScript and improve time-to-first. While MEAN itself has performance limitations, the MERN and MEVN stacks have significant improvements at the frontend and backend levels.

In summary, MERN, MEAN and MEVN are three viable options for web application development, but they have different strengths and weaknesses in terms of scalability and performance. Choosing the best stack depends on the needs and preferences of developers and users[1].

## 7. OPTIMIZATION TECHNIQUES

The scalability and performance of each stack depends on how well the front-end framework is integrated with the back-end components and how well the application is optimized. Some of the optimization techniques that can be applied to any stack are:

**MERN:** Using Next.js, a framework that enables server-side rendering and client-side rendering for React.js applications, providing features such as code splitting, prefetching, and caching. Code splitting is a technique that breaks code into smaller pieces that are loaded on demand, reducing initial load time, and improving performance. Prefetching is a technique that fetches the data and resources for the next page in advance, reducing latency and improving the user experience. Caching is a technique that stores data and resources in the browser or server, reducing network requirements and increasing speed[7]. Using Redux, a library that manages application state and reduces the number of API calls. Redux is a technique that centralizes the application's data and logic in a single storage, making development and debugging easier. Redux also reduces the number of API calls by eliminating unnecessary data retrieval and updates. Using Webpack, a module bundler that optimizes code size and quality. Webpack is a technique that bundles the code and its dependencies into a single file, minimizing HTTP requests and improving performance. Webpack also optimizes code quality by applying plugins and loaders such as minification, compression and transpilation.

**MEAN:** Use of Angular Universal, a platform that enables server-side rendering and client-side rendering for Angular.js applications, providing features such as pre-rendering, lazy loading, and caching. Pre-rendering is a technique that generates static HTML pages for the application's routes at build time, reducing server load and improving SEO[10]. Lazy loading is a technique that loads the application's modules and components only when they are needed, reducing initial loading time, and improving performance. Caching is a technique that stores data and resources in the browser or server, reducing network requirements and increasing speed. Using RxJS, a library that handles asynchronous data streams and reduces network latency. RxJS is a technique that uses observables, operators, and schedulers to manipulate and manage the data streams, simplifying development and testing.

RxJS also reduces network latency by using techniques such as debounce, throttling, and retries. Using Gulp, a task runner that automates development workflow and optimizes code performance. Gulp is a technique that uses plugins and tasks to perform various operations on code such as: concatenation, minimization, compression and transpilation.

**MEVN:** Using Nuxt.js, a framework that enables server-side rendering and client-side rendering for Vue.js applications, providing features such as automatic routing, meta tags, and transitions. Automatic routing is a technique that generates the application's routes based on the file structure, reducing configuration, and improving development. Meta tags are a technique that adds metadata and SEO information to HTML pages, improving the visibility and ranking of the application. Transitions are a technique that adds animations and effects to page changes, improving user experience and engagement. Using Vuex, a library that manages application state and reduces data redundancy. Vuex is a technique that centralizes the application's data and logic in a single storage, making development and debugging easier. Vuex also reduces data redundancy by using techniques such as mutations, actions, and getters. Using Parcel, a module bundler that optimizes code size and speed. Parcel is a technique that bundles the code and its dependencies into a single file, minimizing HTTP requests and improving performance. Parcel also optimizes code speed by using techniques such as zero configuration, hot module replacement, and code splitting.

## 8. CONCLUSIONS

The paradigm of using JavaScript in full-stack web applications was made possible by the MEAN stack, but analysis suggests that MERN or MEVN might be a better option for most contemporary use cases. MERN has a market share and facilitates the development of complex, high-performance user interfaces. MEVN is becoming more well-known as a lightweight but powerful alternative.

All stacks benefit from the versatility and large community of JavaScript while providing access to the React, Vue and Angular ecosystems. However, react outperforms the frameworks in terms of rendering efficiency and native compilation. These aspects make MERN ideal for demanding web and mobile applications. Vue within MEVN promotes code reusability across projects and retains many React features.

With increased dynamism in individual technologies reflected in current industry employment trends, MERN is receiving more attention as it gets closer to MEAN installations. MERN oversees new initiatives that call for hybrid features, real-time dashboards, and SPA interfaces. However, MEVN provides more straightforward yet comparable features to satisfy the demands of quicker prototyping.

In conclusion, all three choices are feasible, but MERN offers the best combination of performance, scalability, and community support for enterprises. Developers looking for employment opportunities or who do not need advanced frameworks might make better use of MEVN or MEAN depending on their use case. However, the analysis clearly shows that mastering the MERN stack is an advantageous web development skill in the industry's competitive hiring environment.

## ACKNOWLEDGEMENT

### A. Ethical Approval

The manuscript is not be submitted to any other journal for simultaneous consideration. The submitted work is original and is not published elsewhere in any form or language. No slicing is practiced. No data, text, or theories by others are presented as if they were ours. Proper acknowledgments to other works is given and no copyrighted material is used in the study. No animals or humans were harmed.

### B. Funding Details

Not applicable.

### C. Conflict of Interest

The authors declared that they have no conflict of interest.

## REFERENCES

[1] Sagar Patel. (2023): MEAN vs MERN vs MEVN Stacks: What's the Difference. Groovy Web.
[2] Bhavyaa; Suhani, Gupta; Vaishali. (2021): Comprehensive Study of MERN Stack - Architecture, Popularity and Future Scope. International Journal of Scientific Research in Computer Science, Engineering and Information Technology.
[3] Sumangla A. Bafna; Pratiksha D. Dutonde; Shivani S. Mamidwar; Monali S. Korvate; Prof. Dhiraj Shirbhare. (2022): Review on Study and Usage of MERN Stack for Web Development. International Journal for Research in Applied Science and Engineering Technology.
[4] Prati Jain; Manjunath C R. (2021): Comparative analysis of MEAN stack and MERN stack. International Journal of Emerging Technologies and Innovative Research.
[5] Mianji Johnsson (2020): The building of the webpages: The comparison study of MERN and MEVN. Jonkoping University, School of Engineering.
[6] Maitray Gadhavi (2023): Full-Stack vs MEAN Stack vs MERN Stack: Decide Your Best Stack. Radixweb.
[7] Ah Zau Marang (2018): Analysis of web performance optimization and its impact on user experience. KTH Royal Institute of Technology.
[8] Rimal A. (2019): Developing a Web Application on NodeJS and MongoDB using ES6 and Beyond. Metropolia University of Applied Sciences.
[9] Monika Mehra; Manish Kumar; Anjali Maurya; Charu Sharma; Shanu. (2021): MERN Stack Web Development. Annals of the Romanian Society for Cell Biology.
[10] Bakwa Dunka Edim Emmanuel Yinka Oyerinde. (2018): Simplifying Web Application Development Using-Mean Stack Technologies. ResearchGate.