Review Paper on Modern Tools and Current Trends in Web-Development

Pankaj Kumar[1], Karan Singh[2], Ms. Maninder[3]

[1,2] *Student, IV Sem, M.C.A, DAV institute of Engineering and Technology*

[3]*Assistant Professor, M.C.A, DAV institute of Engineering and Technology Jalandhar, Punjab*

***Abstract*—This paper presents the development of a social media platform similar to LinkedIn and Facebook, utilizing MongoDB as the database. The primary goal is to incorporate modern web development tools to create an efficient and user-friendly web application, ensuring both customer satisfaction and ease of development.**

The platform enables users to create accounts, manage their profiles, and update or delete details related to their education and work experience. Additionally, users can create posts, like, and comment on other users' posts. A monolithic architecture is implemented to simplify database management.

To test backend functionality, the Postman API was utilized, while deployment was handled using Git, GitHub, and Heroku. Security and validation were reinforced using NPM packages like encrypt for password encryption and validator for user authentication. The front-end design is made responsive with the help of CSS media queries, ensuring a seamless experience across different devices, including mobile phones, tablets, and personal computers.

I. INTRODUCTION

Since the 1990s, after the invention of the World Wide Web (WWW) by British scientist Tim Berners-Lee, the internet has played a crucial role in global communication. Initially designed for scientists to exchange information, the web was built on three core technologies that continue to serve as its foundation: Hypertext Markup Language (HTML), Uniform Resource Identifier

(URI), and Hypertext Transfer Protocol (HTTP). Over the years, additional programming languages such as Cascading Style Sheets (CSS), Java, Hypertext Preprocessor (PHP), JavaScript, Python, Structured Query Language (SQL), and Angular have emerged to enhance web development. These advancements have focused on improving both web security and developer convenience.

In modern web development, HTML, CSS, and JavaScript remain the three essential coding languages.

Additionally, User Interface (UI) and User Experience (UX) design have gained significant importance, with HTML and CSS playing a key role in crafting visually appealing and user- friendly applications.

Web developers often encounter two primary software architectures: monolithic and microservices. In a monolithic architecture, all components, including authentication, user profiles, posts, database, and server, are bundled into a single application. This structure means that if one part fails, such as the authentication system, the entire application may become non- functional.

In contrast, microservice architecture divides these functionalities into independent applications. For example, authentication, posts, and user profiles exist as separate services.

These services communicate through an event bus, which acts as a centralized communication system. If one service fails, the rest of the application remains unaffected. For instance, if authentication encounters an issue, other non- authenticated features remain accessible.

One of the key benefits of microservices is scalability, as developers can upgrade individual services without modifying the entire application. This flexibility is often achieved using Kubernetes. However, while microservices provide long-term advantages, they require more effort and resources, especially in the early stages when fewer developers manage the database.

On the other hand, monolithic architectures are faster to develop, making them suitable for projects with tight deadlines. Due to their simplicity, they are often preferred in the initial phases of web application development before transitioning to a microservices approach as scalability needs increase.

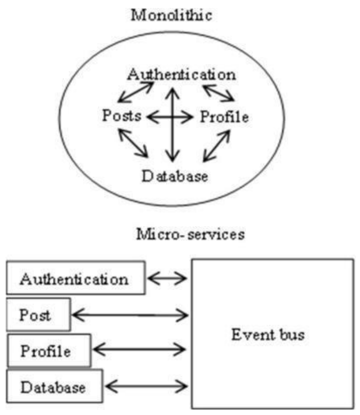

Figure 1. Monolithic and micro-services architecture comparison

Apart from selecting the right database architecture, it is crucial to test an application's functionality to ensure it works as expected. Jest and Mocha are two widely used JavaScript testing frameworks, with Jest being particularly user-friendly, especially for Node.js applications. The latest Jest version is 26.6, and it can be installed using the command: npm i jest@26.6.0 --save-dev

By installing Jest as a development dependency, developers can test their application locally before deployment. Jest provides a simple way to test functions, such as:

test("Success", () => {}); For testing failure conditions: test("Fail", () => {
throw new Error("Fail");
});

Since Jest defines the test() function globally, it can be used anywhere in an application to validate different routes and functionalities.

Web Development Tools and Workflow

This work aims to familiarize developers with essential modern tools required for efficient web development. Efficiency means meeting the latest web technology demands while also utilizing developer- friendly tools to streamline workflow. For instance, a social media platform similar to LinkedIn and Facebook was built using MongoDB as the database. Adobe XD was used for User Experience (UX) design, allowing developers to create a prototype of the application's interface before writing the actual CSS and React components.

This step is crucial, as jumping straight into coding without a design plan can lead to inefficient development.

For styling, SCSS (Syntactically Awesome Stylesheets) was used, which was later compiled into standard CSS using npm packages. Additionally,

Firefox Developer Tools provided a user-friendly environment for working with CSS. The front-end of the application was developed using React, ensuring a dynamic and interactive user interface.

Choosing Monolithic vs. Microservices Architecture

Since the application was still in the development phase, a monolithic architecture was used. This approach is often more suitable for startups, as it is easier to manage with a small development team. As the business grows, transitioning to a microservices architecture can help scale the application more efficiently.

Testing remains an essential part of software development, especially as an application expands. Automated testing using Jest ensures that various functionalities and API endpoints work correctly.

Deployment and Security

To verify back-end functionality, Postman API was used to test the server and REST API responses. For deployment, tools like Git, GitHub, and Heroku were utilized.

Additionally, npm packages such as bcrypt and validator were implemented to encrypt passwords and validate user inputs during authentication, ensuring security best practices.

In summary, a combination of proper planning, modern development tools, testing frameworks, and deployment strategies is essential for building a scalable and efficient web application.

II. RESEARCH METHODS

This section provides an overview of the latest trends in web development. The primary goal is to familiarize readers with essential tools and technologies that modern web developers need to build efficient and scalable applications. Efficiency, in this context, means meeting current web technology requirements while leveraging tools that streamline the development process.

This section covers key technologies such as MongoDB (a NoSQL JSON-based database), Mongoose (an ODM for MongoDB), Node.js (a JavaScript runtime), and React.js (a front-end library for building dynamic user interfaces).

Additionally, it explains why these technologies are widely adopted in the industry and how they contribute to modern web development.

III. MONGOOSE

To interact with MongoDB in Node.js, a popular ODM library called Mongoose is used. Mongoose helps manage data relationships, enforce schema validation, and translate JavaScript objects into MongoDB documents. When defining a Mongoose model, certain guidelines must be followed. Each model requires a name and a schema, which defines the structure of the data. The schema specifies the data types and validations for different fields, ensuring consistency in stored information.

For example, a user model in Mongoose might include fields like name, email, and password, all defined as strings. The "required" keyword acts as a validator, ensuring that essential fields like name and password must be provided before saving a document. This validation feature helps maintain data integrity and security in a MongoDB-based application.

In MongoDB, data is accessed using an Object Data Mapper (ODM), which serves as a bridge between the application and a NoSQL database. An ODM is designed to map objects to database documents, ensuring smooth interaction with MongoDB. One key feature of an ODM is that it automatically assigns a unique ObjectId to each document, allowing easy retrieval of stored data using that specific ID.

IV. NODE.JS

Node.js is a server-side JavaScript runtime environment that enables developers to run JavaScript code outside the browser. It is built on Google's V8 engine, which is also used in Google Chrome to execute JavaScript efficiently. Both V8 and Node.js are primarily written in C and C++, focusing on optimizing performance while minimizing memory consumption. However, while V8 is specifically designed to run JavaScript in the browser, Node.js is meant for executing server-side applications.

One of the key advantages of Node.js is its asynchronous, event-driven architecture, which allows it to handle multiple requests using a single-threaded model. Instead of processing requests sequentially (as in a synchronous system), Node.js places long- running tasks in the background and continues executing other tasks. This is achieved using async/await functions and callbacks, making Node.js highly efficient for handling concurrent operations.

In contrast, traditional synchronous systems like relational databases (RDBMS) such as MySQL process requests one at a time. This can lead to delays as incoming requests must wait for previous ones to be completed before being served. To handle high traffic in such systems, additional hardware resources (like CPU cores and threads) are required, making the solution costly and less scalable.

By combining Node.js with MongoDB, developers can build applications that can handle a large number of users efficiently without needing expensive hardware upgrades. Studies comparing Node.js with traditional web servers like IIS (Internet Information Services) have shown that Node.js performs significantly better in I/O- intensive applications, making it a preferred choice for real-time web applications, APIs, and high-traffic platforms.

V. REACT.JS

Currently, React and Angular are the two most popular JavaScript frameworks for front-end development. React, developed by Facebook, and Angular, maintained by Google, are both open-source frameworks that enable developers to build single-page applications (SPAs). Major platforms such as YouTube, PayPal, Walmart, and Gmail utilize Angular, while Facebook, Instagram, and WhatsApp rely on React.

One of the key differences between the two is how they handle the Document Object Model (DOM). Angular uses the real DOM, meaning that any small change in the application state can trigger a full-page re- render, which can impact performance, especially in applications with large datasets. In contrast, React utilizes a virtual DOM, which optimizes performance by re-rendering only the specific components that have been updated, making it significantly faster than Angular in rendering-intensive applications.

Another major distinction is in the way data binding works.
Angular employs two- way data binding, meaning

that changes in the UI automatically update the underlying data model and vice versa. While this can be useful in some cases, it can also make debugging complex applications more challenging. React, on the other hand, follows a one-way data binding approach, where the application state is stored centrally and updates occur through actions and event dispatchers. This approach provides better control over data flow, making it easier to debug and manage state efficiently.

VI. RESULTS AND DISCUSSION

This section outlines the step-by-step approach taken to develop this project. Throughout the development process, Visual Studio Code (VS Code) was used as the primary code editor due to its user-friendly interface and extensive extension support.

VS Code offers a variety of useful extensions that enhance the coding experience. Some of the key extensions used in this project include:

- Emmet – Helps in writing HTML and CSS faster with shorthand syntax.
- Prettier – Automatically formats the code for better readability.
- Bracket Pair Colorizer – Makes it easier to identify matching brackets.
- Auto Rename Tag – Automatically renames the closing tag when the opening tag is modified.
- Live Server – Launches a local development server for real-time preview.

These extensions improve code efficiency, error detection, and debugging, making VS Code an ideal choice for building this project.

VII. REST API

In this project, three POST request routes were implemented:

1. User Route – Allows users to register and authenticate.
2. Profile Route – Enables users to create and manage their profiles.
3. Post Route – Allows users to create, update, and delete posts.

To authenticate users, an additional authentication route was created. This route enables users to log in by entering their email and password. Once authenticated, users can send POST requests to create or delete their profiles and posts.

The profile and post routes also support GET requests, allowing users to view the profiles and posts of other users. JSON Web Tokens (JWT) were used for authentication, ensuring that only logged-in users could access these protected routes.

To verify the functionality of the REST API, Postman was used for testing. Upon logging in, a JWT token is generated for the user. This token allows access to all protected routes but is valid for only 3600 seconds (1 hour). After this period, the JWT expires, requiring the user to log in again to obtain a new valid token.
This ensures secure access control, preventing unauthorized users from accessing protected routes.

VIII. FRONT-END

The front-end of a web application refers to the user interface (UI) and the client-side rendering of content. This includes CSS for styling and React.js for dynamically rendering web pages. The primary goal is to display data from the backend (Nodemon server) on the frontend (client-side server), allowing users to view posts, comments, and profiles in real-time.

To ensure smooth operation, both the backend server and the frontend server need to be run simultaneously. To simplify this process, the NPM package "concurrently" was used. This package allows both servers to be started with a single command, eliminating the need to manually run them separately.
On the backend, when making HTTP requests using Axios, a proxy is set up in the package.json file using:

```
"proxy": "http://localhost:PORT"
```

This setup allows Axios to send GET, POST, and DELETE requests directly to the backend routes without needing to specify the full backend URL every time.

Axios is a widely used client-side HTTP service in both React and Angular, enabling seamless communication between the frontend and the backend.

IX. CONCLUSION

Web technologies continue to evolve, making it easier for businesses to connect with customers worldwide.

Currently, some of the most popular technologies in web development include MongoDB, Node.js, and React.js.

Why MongoDB?

MongoDB is a non-relational database management system (NoSQL), widely preferred for handling large volumes of data. Unlike traditional relational databases (RDBMS) like MySQL, which store data in tables, MongoDB stores data in JavaScript Object Notation (JSON) format, using a key- value pair structure. This approach makes data storage and retrieval more flexible and efficient.
When a structured schema is needed—similar to relational databases—Mongoose is used. Mongoose provides schema validation and establishes a connection between the backend and the MongoDB database, making data management more efficient.

Backend Development with Node.js

For the backend, Node.js is used. It is a JavaScript runtime environment that enables fast, scalable, and asynchronous server-side operations. Node.js is ideal for web applications that require high-speed data processing and real-time interactions.
Frontend with React.js

On the frontend, React.js is utilized for rendering web pages. One of its key advantages is its Virtual DOM, which updates only the modified components instead of reloading the entire page. This makes React faster and more efficient compared to Angular, which relies on a real DOM and re-renders the entire content whenever changes occur.
Together, MongoDB, Node.js, and React.js provide a powerful and efficient tech stack for building modern web applications.

REFERENCES

[1] M. Cantelon, M. Harter, B. Rajlich, and A. Holowaychuk, *Node.js in Action*, 2nd ed. Shelter Island: Manning Publications, 2017.

[2] R. Wieruch, *The Road to React: Your journey to master plain yet pragmatic React.js*, 2020. [Online]. Available: https://www.roadtoreact.com/

[3] MongoDB Inc., "MongoDB –The Developer Data Platform," [Online]. Available: https://www.mongodb.com

[4] Mongoose, "Mongoose ODM for MongoDB and Node.js," [Online]. Available: https://mongoosejs.com/