WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

The Era of Modern Web Development Using Javascript

Prayag Raj Mathuria[1], Dr. Vishal Shrivastava[2], Dr. Akhil Pandey[3], Prof. Mohit Mishra[4]

[1]B.TECH. Scholar, [2,3]Professor, [4]Assistant Professor
Computer Science & Engineering, Arya College of Engineering & I.T. India, Jaipur
[1]prayag07.mathuria@gmail.com, [2]vishalshrivastava.cs@aryacollege.in,[3]akhil@aryacollege.in, [4]mohitmishra.cs@aryacollege.in

ABSTRACT

Web development has evolved a lot since the 1990s. Nowadays, we use HTML, CSS, and JavaScript as the main building blocks for web apps, and we pay a lot of attention to how things look and work for users. We also consider how to structure our apps, like choosing between big or small pieces. We use tools like Jest to make sure everything works well. And behind the scenes, MongoDB, Mongoose, Node.js, and React.js help make websites fast, adaptable, and interactive. It's a changing field, and these tools are at the front of it all.

1. Introduction

In the 1990s, the digital world changed a lot, and we owe a lot of that to a smart scientist named Tim Berners-Lee. He gave us the World Wide Web, and it wasn't just for scientists; it's the basis for today's internet. Tim introduced three important things: Hypertext Markup Language (HTML), Uniform Resource Identifier (URI), and Hypertext Transfer Protocol (HTTP).

As time moved forward, web development underwent a whirlwind of changes. New programming languages like css, javascript, python, php, sql, Angular, and more joined the party. These languages not only brought robust security measures but also made life easier for programmers.

Fast forward to today, and there are three superstar coding languages that are a must for any web application: HTML, CSS, and JavaScript. These are the foundational pillars of web development. But we can't overlook the rising star - User Interface/User Experience (UI/UX) design. It's all about making things not only functional but visually stunning, and it heavily relies on HTML and CSS. In a world where aesthetics matter, UI/UX design plays a leading role.

If you're a web developer, you've likely heard about "monolithic" and "microservices" architecture. To put it simply, monolithic is like keeping all your things in one place. It's when you put everything - like authentication, posts, profiles, database, servers - in one big pile. But here's the tricky part: if something breaks in one part, it can mess up the whole thing.

Now, let's discuss microservices, the new trend. It's a bit like having a team of superheroes. If one superhero has a problem, the others keep things running. With microservices, if one part of your system has issues, the rest can keep working. Each part has its own space on the server and they talk to each other using something called an event bus, which is like a busy data highway.

When a user logs in, the authentication service tells the event bus. Other apps hear about it and let the user into protected areas. Why use microservices? It's all about growing and making things better one step at a time instead of redoing everything at once. But there's a trick; you need Kubernetes to make microservices work like magic.

Monolithic architecture, on the other hand, can be like speed dating—it's quick and dirty. It suits the early days when you don't have a big team dealing with databases.

But let's stop talking about theory and have a look at Jest

Creating an application is a bit like painting. You don't just start without a plan; it's like painting without knowing what the final picture will be. That's where Adobe XD comes in. It's like a canvas for web developers. You use it to draw the plan for your app's look, making sure it's beautiful. It's a waste of time to write the CSS and React without a plan!

To make your UI ideas real, you can first write the design in SCSS and then turn it into CSS using special SCSS tools. It's a bit like changing a sketch into a digital picture. For creating your UI, Firefox is a good choice, and React is the key player that makes the front-end work.

When you're still in the development phase, a monolithic approach is your ally. It's perfect for startups. But as your business grows, you might consider switching to microservices. Also, testing is like the quality control team. As your app expands, automated testing is a must. Jest is like your trusty sidekick, testing everything to ensure it's all working as it should.

To ensure everything works together harmoniously, you've got Postman API to check your backend server and REST APIs. And when it's showtime, Git and GitHub step in to deploy our website.

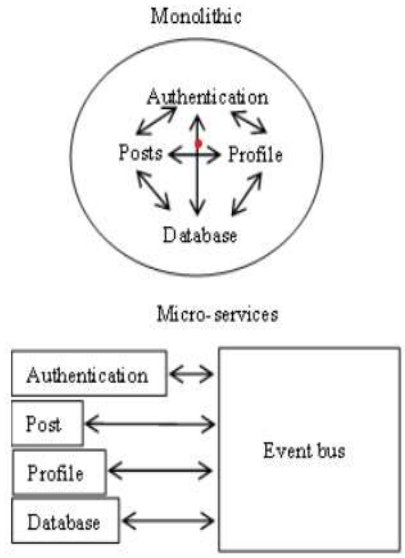

Figure 1. Monolithic and micro-services architecture comparison

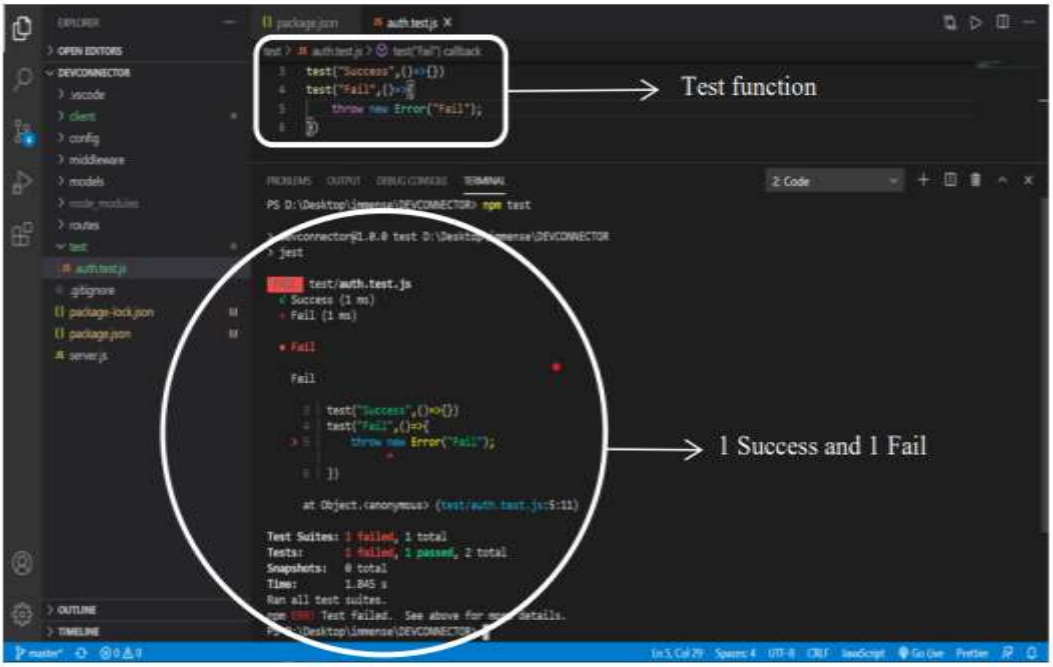

Figure 2. Testing an application using Jest

2. Research Methods

Let's explore the newest developments in web development in this section, along with the resources that can help you become an expert in the field. We're talking about the tools that perfectly fit the needs of modern web technology and make a developer's life smoother.

2.1 MongoDB/Nosql

Speed is a key factor in every application in the fast-paced digital world of today. Your app's response time has a big impact on how the user feels. Scaling up is necessary to reach a wider audience from a business perspective, and this calls for a database that can handle the demand.

Traditional relational databases, managed using structured query language (SQL), have their limitations, especially when dealing with massive datasets. As the user base expands, SQL can struggle to maintain its performance.

Now, let's rewind to 1998 when Carlo Strozzi introduced a groundbreaking concept known as "NoSQL." This kind of data management system doesn't play by the old rules of rigid schemas used in traditional databases (RDBMS). Instead, it embraces flexibility by storing data in a versatile JavaScript object notation format (JSON), eliminating the constraints of fixed tables. This adaptability is what makes NoSQL stand out.

Among NoSQL databases, MongoDB takes the spotlight. It leverages JSON for data storage, making it a star in the world of noSQL databases.

In a captivating comparative study, they put MySQL, a renowned RDBMS system, head-to-head with MongoDB. They conducted tests covering all the usual database operations: inserting, updating, and deleting data. In the first test, they created 500 users, 500 orders, as well as 500 items. The slowest performance for mongoDB. Amazing 0.0474 seconds. In the second test, involving 1000 users and 1000 user requests to modify data, MongoDB demonstrated impressive performance at its slowest, with a response time of 2.3201 seconds.

In the third and final test, they raised the stakes by creating 7,000 users and handling 10,000 user updates. Even in this scenario, MongoDB showed its mettle, with the slowest response time at 23.3427 seconds. The takeawayA standard RDBMS system will take much longer to complete create, read, update, and delete (CRUD) activities as your data amount grows.

2.2 Mongoose

Let's now explore the realm of MongoDB and Mongoose, its reliable friend. MongoDB is a versatile NoSQL database, not just any database. To communicate with it effectively, you need something special – an Object Data Mapper (ODM).

Think of an ODM as a mediator between your app and MongoDB's noSQL world. It acts like a language translator for your data. Every item of data that MongoDB stores has a unique ID that can be used to locate specific information at a later time.

To connect their application to MongoDB, developers frequently use a library known as Mongoose. Mongoose does more than just fetch data; it manages relationships between data, ensures data follows a structure (schema validation), and translates between your code and the database.

When you make a model in Mongoose, there are some rules to follow. You need to describe a plan for your data called a schema and give your model a name. The schema makes sure everything is organized. For instance, if you're making a user model, the schema might say that the password, email, and name must be in text. And that "required" word you see? It's like a guard for your data, ensuring nothing important is missing.

2.3 Node.js

Now, let's have a chat about Node.js. It's the rockstar of server-side JavaScript. It uses Google's V8 engine, the same engine that powers the Chrome browser, to execute JavaScript code. The magic of Node.js lies in its speed and memory efficiency.

Node.js dances to its own rhythm. It follows an asynchronous I/O event model, which means it can handle multiple requests at the same time. It's like having a chef who can juggle multiple orders in a restaurant. All orders are taken, and the chef starts cooking the longest one last, while the other dishes are already on their way to your table. This asynchronous magic is why Node.js can handle several requests simultaneously, thanks to the async & await.

No wonder Node.js, when combined with MongoDB, is lightning fast. Traditional databases work synchronously, and as the number of users grows, they tend to slow down. Waiting isn't good in today's fast-paced world.

To keep up, synchronous systems need more hardware to serve more users, which can be costly. But Node.js, with its asynchronous architecture, doesn't have that problem. It can handle more requests without requiring loads of extra hardware. So, it's a win-win for both speed and cost-effectiveness.

2.4. React.js

Lastly, let's talk about React.js, one of the trendsetters in front-end development. It's like the canvas for artists to create stunning user interfaces and often compared to its cousin, Angular.

Angular, backed by Google, and React developed by Meta, are such open source frameworks that help programmers build single page apps. They power some of the biggest names in the tech world.

The secret to React's power is its usage of a Virtual DOM, which is essentially an optimized, lightweight variant of Angular's true Document Object Model (DOM). When a change happens in the DOM, Angular refreshes the entire thing, but React is smarter. It only updates the part that has changed. Think of it as editing a large whiteboard with lots of drawings; Angular would erase and redraw the whole board for a small edit, while React would simply fix that one spot, making it much faster.

React uses a one way data binding. Instead of the two way used by Angular, React's data flow is more organized and easier to understand.

3. Results and Discussion

Now, let's dive into how we built this project, step by step. For our coding tasks, we turned to Visual Studio Code – a friendly, powerful code editor. It's like the Swiss Army knife of code editors, with lots of handy extensions like Prettier, Auto rename , and Live Server. These extensions aren't just bells and whistles; they're the tools that help us spot bracket pairs, identify errors, and even set up a local server for testing and deployment. In short, they're our trusty sidekicks in this coding adventure.

3.1 Shaping Mongoose Schemas

Our journey began with the crafting of Mongoose schemas, each with its unique role.

The user schemas collected vital details like name, email, password, and the date they joined. The profile schema extended this by allowing users to share information about their company, website, location, skills, experience, and education. This was where users could tell their professional story.

And then, the post schema entered the scene, handling attributes like text, name, and date. Comments were key to keeping track of user interactions with posts. In the mongodb cloud database, we organized 3 folders, for user, profiles and for posts. You can see this in action in Figure 3, where these collection folders reside in the MongoDB cloud.

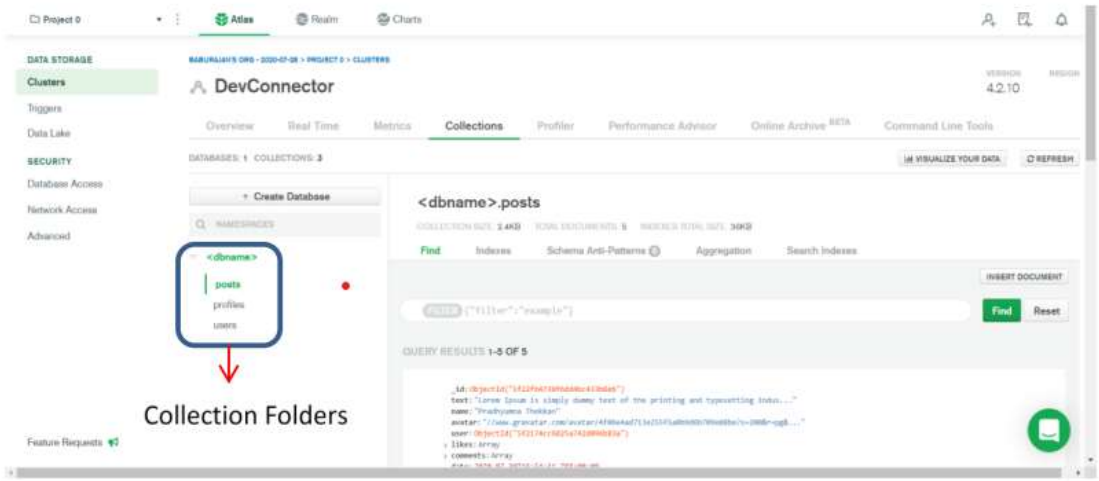

Figure 3. MongoDB collection

3.2 REST APIs

Now, let's delve into the realm of REST APIs, where we built three distinct routes for users, profiles, and posts. These routes were our gateways for user authentication and interaction. Users could log in using their email and password and then take advantage of post requests to create or delete posts and profiles.

To ensure everything ran smoothly, we relied on the Postman API to scrutinize the performance of our rest api. If a user signs in, a jwt was generated, granting them access to all the protected routes. But, like Cinderella's spell, this token had an expiration time set at 3600 seconds (or one hour). Once the clock struck, the JWT lost its powers, and users needed to log in again to receive a fresh, valid JWT for continued access to the routes. See Figure 4 for a visual representation of this process.

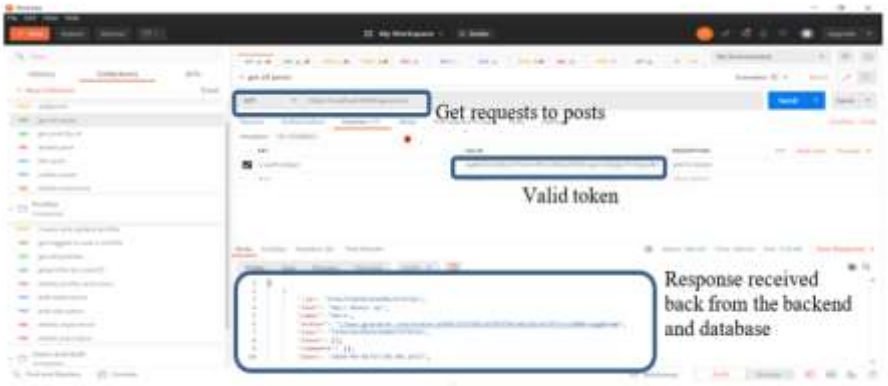

Figure 4. Valid JWT and response from the backend and database

3.3 Front-End

Okay, let's talk about making websites look good. Imagine it as an artist's canvas. We use CSS to style things and React.js to show stuff on the webpage. The ultimate goal? To seamlessly transfer data from the backend server (nodemon) to the client server (node) so that users can enjoy real-time posts, comments, and profiles.

To make this happen, we needed both servers to work together harmoniously. We made our lives easier by using an NPM package called "concurrently."

3.3.1. User-Friendly Design

In web applications, design isn't only about appearances. It's about making things easy for users. How it looks affects how happy customers are, which affects how well a business does. When people are happy, they often tell others about it. That's why we don't just start designing without a good plan. This is where User Experience (UX) design becomes important.

Our useful tool for ux and ui design is from adobe. It's easy to use, feature-packed software available for both macOS and Windows. Adobe XD allows developers to create a prototype of the application's UI, offering a functional and visual representation. Designers can create a sketch roughly, complete with functionality elements ex. pop up window (CSS modals), page links, effects, color schemes, and more. One advantage is that Adobe XD uses feature names that align with CSS properties, making the transition from design to code a seamless process.

In the design section of Adobe XD, we could work on web designs. A handy feature is the "repeat grid," which allows us to replicate a grid cell across the webpage. Components in the design part are used to show recurring elements like sidebars and navbars, which appear on multiple pages. Once declared as a component, you design it once and use it across many pages. In the prototype section, you can create page links and add functionality to modals (pop-up windows).

After our UX design and UI prototype are ready, it's time to write the css. However, traditional css can be rigid. So, we opt for SASS, a scripting language that compiles to css. SASS offers flexibility by allowing us to declare variables for colors, making design changes a breeze. Moreover, we can also write the code in different files and import them using the import command, allowing for organized and reusable components. SASS empowers us to create web designs with greater ease and efficiency, making it the modern CSS solution of choice.

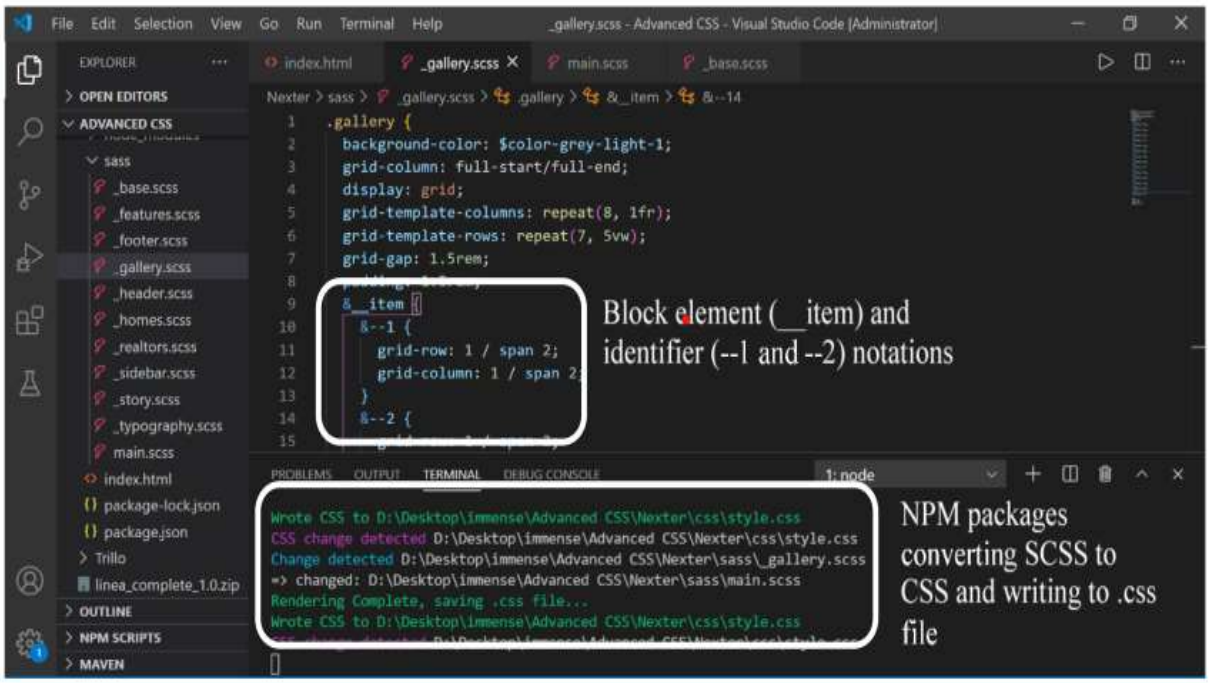

Figure 5. SCSS Code and NPM package

3.3.2. Redux

Now, let's explore Redux, a nifty open-source JavaScript library that plays a pivotal role in handling your app's status or, in simpler terms, its state. Redux is like the conductor of an orchestra, making sure all your front-end components harmonize seamlessly.

In the Redux world, the term "state" might sound a bit imposing, but think of it as the master switchboard of your app's status. It's controlled by the Redux store and can be accessed via the getState() function. This state reflects everything happening in your Redux-powered app, so when something changes in one part of your app, it sends ripples through the whole interface.

Redux makes use of middleware to intercept actions before they reach the reducer, which is like the action handler. Actions act as messengers, carrying data from your app's back-end to the store, allowing the state to know what's going on. For instance, when a user logs in, information flows from back end to front end, connecting the application to a database, and then displaying the data on the front-end. Actions are the couriers making this happen.

Your Redux store is the headquarters where you set up the rules, the initial conditions, and the middleware. If you're curious about what's going on with your app's status, you're in luck - there's a handy Chrome extension for Redux that lets you peek under the hood.

Actions have two main parts: the "type" and the "payload." The "type" is like a label that tells Redux what action to perform. For example, when you want to fetch a user's profile, you use the action type "GET_PROFILE." The "payload" is where you stash the data fetched from the back-end. So, if you're getting a user's profile, the data you bring back becomes the payload.

This payload then makes its way to the Redux store, and from there, it spreads through your app's front-end like wildfire, making changes as it goes. Now, if a user's login attempt fails, the action type switches to "LOGIN_FAIL," and the "authenticated" status becomes "false," indicating that the user's credentials didn't checked out.

When this happens, the back-end sends a message to the front-end through the "LOGIN_FAIL" action, popping up a little notification on the login page. To keep things tidy, this notification disappears after 5 seconds, thanks to a clever trick using JavaScript's setTimeout function.

4. Conclusion

In the fast-moving world of the internet, we see a smooth connection between online stores and customers everywhere. The big names in web tech are MongoDB, Node.js, and React.js, and they're popular for a reason.

MongoDB takes the lead as a non-relational database system, outshining traditional systems like MySQL. It excels in handling vast data volumes, thanks to its JSON data format. Unlike MySQL's rigid tables, JSON's key-value approach makes data management a breeze. Mongoose, our trusted ally, steps in to create data schemas, forming the vital link between the application's back-end and the MongoDB database.

Node.js, our backstage hero, powers the server-side with JavaScript. Its asynchronous nature ensures all user requests are swiftly handled, leaving no one waiting. The combination of Node.js and MongoDB outperforms sluggish RDBMS systems.

Now, meet React.js, the star performer on the front-end. It works its magic by rendering content on the client-side. With its virtual DOM, React.js re-renders only the updated parts, making it a top choice for efficient and visually appealing user experiences.

In the always changing tech world, MongoDB, Mongoose, Node.js, and React.js are making things easier and providing great web experiences. This tech revolution isn't slowing down.

5. References

● Alaoui, K. S., Foshi, J., & Zouine, Y. (2019). "Radio over fiber system based on a hybrid link for the next generation of optical fiber communication." International Journal of Electrical and Computer Engineering, 9(4), 2571-2577. DOI: 10.11591/ijece.v9i4.pp2571-2577.

● Stuart, G. (2017). "HTML5 and the Canvas Element." In Introducing JavaScript Game Development (pp. 3-16). DOI: 10.1007/978-1-4842-3252-1.

● Attardi, J. (2020). "Introduction to CSS." In Modern CSS (pp. 1-15). DOI: 10.1007/978-1- 4842-6294-8.

● Guha, A., Saftoiu, C., & Krishnamurthi, S. (2010). "The Essence of JavaScript." In European Conference on Object-Oriented Programming–Object-Oriented Programming (pp. 126-150). DOI: 10.1007/978-3-642-14107-2_7.