Christopher Chen

CS 489 SCH1

4/16/21

Academic Research Paper

**Introduction**

Web application development is the science of creating application programs that are hosted on online servers in the cloud. These application programs run in the web servers, unlike the standard computer-based application programs and software that are installed on computers' local disks to run locally on the operating system (Annarelli et al.). Users must have a web browser with an active internet connection for users to access the web application. This means that web applications do not need to be downloaded, as is the case with other computer-based application software. Instead, they are accessed through a network. This is made possible by using browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge, among others web browsers. Today, the most available web applications are written JavaScript, Cascading Style Sheets (CSS), and HTML5.

Web application development usually has a relatively short development life cycle led by a relatively small program development team (Falzone and Bernaschina). The web applications' actual development is usually divided into two: The Front-end development and the Back-End development (Bukhari et al.). Front-End development is accomplished through what is known as client-side programming. A client is typically a computer software or application such as the web browsers listed above. Programming languages such as HTML, CSS, and JavaScript are what are typically utilized during client-side programming. This means that the source code will be processed by the client web browser on which it is running.

Back-End application development, on the other hand, also sometimes referred to as server-side programming, is the one that activates the client-side. The main purpose of server-side programming is to create the scripts that the web applications will use after they are ready and implemented (Mahmudi et al.). Multiple scripting languages are used to write these scripts, and the most common ones include Python, PHP, Ruby, among many others. Server-side scripting is responsible for the creation of custom end-user interfaces. The source code becomes invisible to the end-user since the webserver processes it.

Web development was easy back then when languages such as HTML and CSS ruled the front end and PHP was the master of backend programming. However, there are now several online platforms and libraries for building interactive user interfaces and their components. Over the years, the web has changed, and so have the programmers developing these robust applications with the latest tech stacks. More emphasis has shifted from the server-side programming to the client-side. And for this reason, this paper focuses on three frameworks on which web application development is done today. These include web application development using React front-end, node back end, and Mongo DB.

**Web Application Development using React Front-End**

React is an open-source JavaScript library mainly known for its efficiency in building compassable interactive user interfaces (Gackenheimer). It is a front-end framework that can also be used as a base for mobile applications with single pages. React has made it possible for programmers to build complex and robust application programs more quickly and iterate immediately since it is a front-end library. Since the introduction of this JavaScript-based user interface library, react has changed the dimensions of web development. Currently, more than a hundred web applications have been developed using React.

**How has React changed web development?**

Before the onset of the server-side revolution brought by the React framework, web development was all about scripting and rendering. There was the use of static HTML pages, which were put in folders and rendered using PHP scripting language. This was not unique back in the day, and neither were the web pages intuitive. That changed with the dawn of JavaScript libraries like React (Gackenheimer). With React, a lot more can now be done with JavaScript that was not possible ten years ago. This change was brought about by the possibility of writing web applications with client-side JavaScript. The development of Single Page Applications (SPA) was made possible by React, which has made programmers write codes on the client-side JavaScript.

React was initially primarily used for rendering views in web applications or Single Page Application programs (Gackenheimer). This allowed the web developers to develop reusable components that are platform-independent on each other; a phenomenon is also known as code reusability. In this phenomenon, the application development process is geared towards reusing existing software. This means that any critical failure in the system was easily re-engineered by the use of pre-existing code. This is one of the revolutions brought about by React in the web development world.

Another exciting feature is called Virtual DOM, a programming concept that has changed the way programmers implement Sever Side Rendering (SSR) without needing to update the whole view each time they carry out system updates. This technique is not moving away from SSR but rather focuses on its implementation whenever need be. React also ensures that there is the option of client-side routing, which primarily ensures a quick navigational experience, especially for the end-user. This means that React can make subsequent navigations behave like Single Page Applications (Gackenheimer). This is to say that React is best to be used in the building of dynamic and robust interfaces that are painlessly interactive and triumph over all other existing frameworks such as Angular. This is made possible by using virtual DOM to update components whenever an end-user interacts with a web application without interfering with other parts of the interface.

**Benefits of using React Front End in Web Application Development**

One of the main concerns that continue to affect programmers today is choosing a user-friendly and easy-to-learn framework. This groundbreaking JavaScript library is the easiest to grasp programming language, especially for developers who are already acclimatized to JavaScript. In that case, a team of the programmer who is well conversant with JavaScript will find it very easy to code using React (Guan et al.). On the other hand, lack of pre-knowledge in JavaScript does not mean one cannot learn to code using React. Unlike Angular, React maintains a smooth learning curve, especially for developers who are beginners.

React allows for code reusability, a programming practice that makes it possible to use existing software components to develop another software. This is possible with React since applications developed in React comprise components that are independent of one another. This component reusability technique has revolutionized the number of time programmers today take in developing application programs generally. Another important benefit of React is that it allows for a good abstraction layer that shields end-users from accessing a web application program's source code. This is a key technique to protect the web application's integrity from being tampered with by potential clients (Guan et al.).

Lastly, React is a well-established JavaScript library with a vibrant ecosystem of developer tools (Guan et al.). This library has a collection of ready-made, customizable charts, graphics, and amazing visualization techniques. In addition to this, React has other components that advocate for the fast development of web applications without reinventing the wheel. An amazing collection of React developmental tools are also available for programmers who go the extra mile to build amazing stuff on their websites.

**Drawbacks of** **Using React for Web Application Development**

One of the drawbacks of using React for Web Application Development is the lack of proper documentation. React technology has accelerated so fast to the extent that there has been no time to make proper documentation. This has caused React developers to develop instructions of their own with every evolving phase, which has led to a lack of standardized documentation. Another limitation of React is its high pace of development. Though this may sound like an advantage to React developers, some developers feel inadequate to relearn new upcoming coding methods (Guan et al.). They consider it a tedious and unworthy job to adopt all the new changes, coupled with continuous updates. The need to always stay updated due to the high development pace is very cumbersome to some developers.

React developers also have time and again complained about the complexity of its unsteady learning curve. This is due to the use of JSX, a syntax used by React developers to allow HTML to interact with JavaScript. This technique's benefits aside, some developers consider this syntax a barrier, claiming it is too complex, especially for new developers.

These drawbacks, however, are still manageable for most professional React developers, and they still consider React the safest tool to use, especially for long-term supported web applications. This is because React offers the most robust interactive user interfaces that are meaningful even to the end-user. In that case, React remains the most appropriate choice for modernizing relevant codebases gradually (Guan et al.).

**React front-end interaction with the node using REST API**

Node is a runtime environment in which software developers can launch both front-end and back-end web application programs using JavaScript. It is an open-source, cross-platform runtime environment where JavaScript code can be executed outside a browser. React Frontend, therefore, interacts with Node, especially when the front-end development is launched in the Node environment. During this launch, the use of a REST API is mandatory. REST API, also known as RESTful API, is an application programming interface (API) designed to take advantage of existing protocols that allow for restful interaction with web services.

During a secure application build with React front-end and node using the REST API, an open-source Identity and Access Management solution application called Keycloak must be used to make it easy to secure these web applications and services with close to no code at all (Ahmed). It is the responsibility of this Keycloak to open up protocol standards during the launching process. This authentication process is thereby responsible for the virtual availability of every available feature that might be used regarding user authentication and authorization.

React front-end interaction with the node using REST API has a series of steps that must be followed to ensure the successful launching of the web applications in the cloud. First, when a user tries to log in to the Front-end of the web application, the React app is activated. The user will then receive authentication requests from the keycloak server. Successful authentication from them by the Keycloak server will guarantee the user access to the web application program, and a JWT (JSON Web Token) will be awarded to them. On the other hand, unsuccessful authentication will redirect a user back to the Keycloak server login page to enter correct login details (Ahmed).

With this JWT token issuance, the Frontend of the web application can now access the back end of the backend REST API by sending a request containing the JWT token. Validation of this token will be done by the Keycloak server for it to communicate with the back-end. The backend server at this point will request accordingly, but only if the JWT token is valid. Otherwise, the backend will respond with error code 401 if the JWT token becomes invalid, leading to the end user's automatic un-authorization.

**Web Application Development Using Node Back-end**

Node is a runtime environment, not a framework nor a library like React. Node is a runtime environment in which software developers can launch both front-end and back-end web application programs using JavaScript. It is an open-source, cross-platform runtime environment where JavaScript code can be executed outside a browser. After introducing Node into the world of programming, it was immediately recognized as the most exciting single piece of software in the current JavaScript engine.

JavaScript has always been the most popular programming language and exhibits universally approved technological advancement in programming history. As an open-source runtime environment, Node is the number one cloud computing technology web hosting solutions provider. Recently there has been wide adoption of server-side JavaScript since most programmers have gotten acclimatized with Node's use.

Web development using node backend is very popular since almost any code editor in the market now supports and has plugins for javascript and Node, meaning programmers are only left to customize their IDE environment to their specific coding needs. The Node backend introduction has made JavaScript escalate up the walls of fame in a direction that was never going to be possible without it. Node backend has made it possible to use JavaScript on the server and develop full-scale enterprise-level web applications.

**The Benefits of using Node Backend in web application Development**

Node has recorded over three million downloads and over 750 new contributors by 2018, as reported by the Node by Numbers survey report. The use Node runtime environment as for backend has several full-stack JavaScript development benefits. These include better efficiency of the web applications developed coupled with overall increased developer productivity. Node backend is a very robust environment that offers developers in the field a huge number of free tools such as visualization technologies and free upgrades (Cantelon et al.).

Conversely, the Node backend offers code reusability, a programming practice that makes it possible to use existing software components to develop another software. This is possible with the Node backend since applications developed in the runtime environment comprise components independent of one another (Cantelon et al.). This component reusability technique has revolutionized the number of time programmers today take in developing application programs generally.

Another important benefit of using Node backend in web application development is its effectiveness in web applications' performance. Node backend has a reputation for developing very robust applications in a short period due to component reusability. The development of these web applications being less time-consuming ensures fast and reliable software. Programmers with a history in coding with JavaScript can take up the server-side programming with minimum effort (Cantelon et al.). The same programming language used on both sides can ensure effective code reusability of code on the front end and the back end. This is achieved by wrapping code into modules and creating new advanced levels of abstraction.

**The Drawbacks** **of using Node Backend in web application Development**

As much as Node backend is considered a very robust runtime environment, its biggest drawback is its inability to process CPU-bound tasks. This problem occurs when a CPU-bound task is issued to the Node backend environment. Node is limited in handling heavy requests, a problem that results in slow processing and overall delay in the event loop, leading to performance bottlenecks. It is for this reason that Node is not recommended for heavy computational tasks.

Another drawback of using Node backend in web application development is the callback hell issue. This happens due to the asynchronous nature of the Node. The callback is a Node technique where functions are executed after each task queued in the system is finished. The callback hell is a situation that arises when all the queued tasks are simultaneously called back for execution, resulting in a negative impact on the quality of code. This happens because these callbacks get nested within other pending callbacks several levels deep, to a point where it is impossible to understand and maintain the code.

Lastly, there is a shortage of experienced professionals in Node despite the common misguided belief that all JavaScript programmers are also good node developers. There is a significant amount of effort that is accompanied by mastering the server-side of JavaScript programming. The same is true also for gaining background knowledge in backend programming. Because of this steep learning curve, the number of well-experienced Node developers is significantly low as opposed to the total number of JavaScript professionals.

**Web Application Development Using MongoDB**

Over time, MongoDB has been considered one of the best tools that serious programmers use during web development. It is a cross-cutting platform that is document-oriented and is majorly known for using JSON-like documents. MongoDB is usually classified as a NoSQL database application program. It is very popular in web applications, however much imperfect a tool it is. But despite its challenges and limitations, MongoDB works perfectly well when it comes to developing robust applications.

The building of web applications using MongoDB involves two key components: A REST API and a Feed grabber. The REST API is created with the help of a MEAN stack (Ahmed). This MEAN stack is responsible for availing MongoDB, which is the database that is to be used for the web application development, availing the JavaScript, which is the framework on which the development takes place during the creation of rich applications as well as availing a server-side JavaScript interpreter. The REST works on a client-server model such that when the client communicates with the HTTP server using HTTP commands, the server replies with JSON codes (Ahmed). This efficient interaction of MongoDB and JavaScript makes the MEAN stack particularly suitable for the development of web applications.

**Benefits of** **Using MongoDB for Web Application Development**

MongoDB is schemaless, meaning a voluminous amount of documents that differ in size, properties, content, and the number of fields. This allows for the free and easy manipulation of data without interfering with its integrity (Satheesh et al.). This means that it is easy to scale out MongoDB when need be.Another important characteristic of MongoDB that makes it best suited for web application development is that the web applications developed are general purpose.

MongoDB supports auto-sharing of data within the web application programs. Being a database application program, developers are tasked with the responsibility of deploying multiple relational databases. MongoDB supports auto-sharing, meaning they can automatically share data across different servers without knowing the web application itself (Satheesh et al.). Lastly, MongoDB is a very useful cross-cutting platform, especially in the field of cloud computing. A programmer can only focus on the writing of codes rather than building very expensive and complex platforms.

**Drawbacks of** **Using MongoDB for Web Application Development**

Despite the many benefits of MongoDB, it houses a few other disadvantages. MongoDB uses high memory for data storage. This is a limiting factor since one of the most paramount factors for every developer is space utilization. The codes in MongoDB are usually very long and heavy, leading to high memory usage. Another limiting factor is the limit for document sizes. MongoDB

Can only execute documents that 16MB and below. Documents above 16MB are considered too heavy for MongoDB to manipulate. On the other hand, highly experienced programmers work on larger document sizes, as large as gigabytes and terabytes of documents in size. It is due to this limitation that most programmers shy away from developing web applications using MongoDB.

Lastly, MongoDB offers no transactional support. Electronic signatures are left behind when a user works on a web application that was originally engineered using MongoDB. This is caused by the fact that it uses its internal memory for storage which does not guarantee any transactional support. This means that it is not suitable for operations that require transactional evidence.

MongoDB is still one of the most widely used NoSQL databases in the modern web application world. Despite its limiting factors, MongoDB is still easy to understand, and unlike other relational databases, it is easy to scale. It is best suited for hierarchical data storage as opposed to other RDBMS. MongoDB is not the right solution for social data, especially where there are links in the documents. This is because it cannot show relationships between the documents, and when the links are the most critical data in the system, MongoDB becomes a limitation.

**Node interacting with MongoDB**

Node and MongoDB are quickly becoming the most robustly recognized technologies in the world of full-stack JavaScript technology. Node is widely known for its ability to build scalable web applications in a short amount of time since it is powered by Google's V8 JavaScript runtime (Satheesh et al.). On the other hand, MongoDB is the perfect choice when it comes to the development of scalable, high-performance open-source solutions. When these two technologies are used together in the development process, subsequent web applications can be built within a short period and launched to the cloud with ease. However, depending on the end-user's needs, MongoDB could at times not be the best solution for the web application that is to be developed. That is why most developers prefer using the two technologies to balance out each technology's cons.

The interaction of Node with MongoDB is extremely popular in the web development community today and for enterprises. Many big companies have now fully embraced the implementation of these two technologies to power their web applications. These include Fortune 500, PayPal, LinkedIn, eBay, Walmart, Yahoo, Microsoft, Uber, etc. Due to Node's synchronous nature, the interaction of these two applications in web development has become a great alternative for high traffic with high input/output applications (Satheesh et al.). Examples of these applications include e-commerce websites alongside mobile applications.

**Data encoding between Front-end and backend.**

When building any web applications, there will be a front-end and a backend part of the application. In simple terms, the front end is what the end-user sees on his computer, while the backend is all the other heavy lifting at the back of the scenes (Guan et al.). The Frontend and the back end communicate with each other through HTTP requests sent from the Front-end to the back end in the event of transmission of data entered by the user. When this data is sent from the Front-end to the backend, the backend must validate the data before eventually storing it in some database or responding to it through codes.

There are two ways in which a front end communicates to the backend of the same web application (Guan et al.). First, a Hypertext Transfer Protocol (HTTP) request sent from the Front-end to the server responds with browser-ready codes from the backend after validation is successful. Once the user can access the website and create, read, update, destroy methods, commonly known as CRUD, that are primarily used to manipulate data with being updated on the backend. This will then be displayed via the Frontend, and in cases where the display is not relevant for the user experience, it will be stored for future reference.

The second technique in which a front end communicates to the backend is through AJAX, a popular choice that is commonly used by high skilled developers for back and forth communication between both sides (Guan et al.). This is achieved through a technique known as Representational State Transfer (REST) coupled with asynchronous communication. There is no need to refresh the entire website to update the site's basic components in this technique.

**The Difference between Front-end and Backend Development**

Front-end and back-end development are two sides of the same coin; however, they are distinct from each other. The proper functionality of a web application relies on these two sides working together as a single unit. Neither the front-end nor the backend is more important than the other, and no web applications can exist with just one side. While the front-end is the client-side of the web application, the user interacts with the backend is the server-side of the web application responsible for the behind the scenes operations (Guan et al.).

Front-end development basically dwells on the visual and interactive side of a web application, the side that the end-user sees. This, however, is not the same as web design. It is the work of a User interface (UI) designer to develop these front aspects of a web application, contrary to the belief that this is achieved by the front-end developers. Using the already available reusable front-end languages, front-end developers take the designs of the User interface designers and transforms them into something functional.

That is to say, web designers are tasked with designing the web application and their corresponding feel (Almqvist). This includes the layout of the web applications as well as the functionality of buttons and touchpoints for the effectiveness of the end-user. On the other hand, front-end developers are tasked with the responsibility of working on the functionality of the web application. This encompasses the engineering behind which the user interface developers' are transformed into a live, interactive website.

The backend, sometimes referred to as the server-side of the web application, is the portion behind the scenes that is not visible to the end-user. Storage and organization of data happen in the backend to ensure effective and efficient deliverables on the client side. In order for Backend developers to make dynamic web applications, they use additional backend components. This ensures that the user input can be modified based on what is in the database (Waschneck et al.). This means that the contents of the web application can be changed by the end-user. This phenomenon differs from that of a static web application since its contents are unchangeable.

**Conclusion**

Web development is a very symbiotic process that requires a balanced interaction between the front end and backend sides of the program for smooth functionality. These two ends both work together, side by side, to provide valuable service and experience to the end-user even though the role for each is different. There are many programming languages, frameworks, and online platforms and libraries that can be used to successfully implement web applications. All these libraries and runtime environments have their strengths and weaknesses. It is, therefore, up to the developer to determine which programming language to use and why in an attempt to come up with the most robust and efficient web application that can satisfy the end-user needs completely.

**Works Cited.**

Ahmed, Satia. *A RESTful API with MongoDB*. 2018.

Almqvist, Frida. “The Fuzzy Front-End and the Forgotten Back-End: User Involvement in Later Development Phases.” *The Design Journal*, vol. 20, no. sup1, Taylor & Francis, 2017, pp. S2524–33.

Annarelli, Alessandro, et al. “Web-Application Development Projects by Online Communities.” *Industrial Management & Data Systems*, Emerald Publishing Limited, 2017.

Bukhari, Syed Shafique Ali, et al. “Improving Requirement Engineering Process for Web Application Development.” *2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, IEEE, 2018, pp. 1–5.

Cantelon, Mike, et al. *Node. Js in Action*. Manning Greenwich, 2014.

Falzone, Emanuele, and Carlo Bernaschina. “Model Based Rapid Prototyping and Evolution of Web Application.” *International Conference on Web Engineering*, Springer, 2018, pp. 496–500.

Gackenheimer, Cory. *Introduction to React*. Apress, 2015.

Guan, Shiqi, et al. “Front-End and Back-End Separation-React Based Framework for Networked Remote Control Laboratory.” *2018 37th Chinese Control Conference (CCC)*, IEEE, 2018, pp. 6314–19.

Mahmudi, Kamal, et al. “Ontology to Relational Database Transformation for Web Application Development and Maintenance.” *Journal of Physics: Conference Series*, vol. 971, no. 1, IOP Publishing, 2018, p. 012031.

Satheesh, Mithun, et al. *Web Development with MongoDB and NodeJs*. Packt Publishing Ltd, 2015.

Waschneck, Bernd, et al. "Unified Front-end and Backend Industrie 4.0 Roadmap for Semiconductor Manufacturing.” *SamI40–2nd International Workshop on Science, Application and Methods in Industry 4.0, Proceedings of the Workshop Papers of i-Know*, 2017, pp. 11–12.