Blog Cum Social Media Platform on a Local Server using Raspberry Pi

Submitted By:

Swapnesh Sanjeev Srivastava

School of Computing Science & Engineering

Galgotias University Greater Noida, Uttar Pradesh

Swapnesh_srivastava.bcamscse@galgotia galgotiasuniversity.edu.in

Rohan Kumar

School of Computing Science & Engineering

Galgotias University Greater Noida, Uttar Pradesh

Rohan_kumar.scsebca@galgotiasuniver sity.edu.in

Rajeev Kumar Singh

School of Computing Science & Engineering

Galgotias University Greater Noida, Uttar Pradesh

Rajeev_singh.scsebca@galgotiasunivers ity.edu.in

Guide: *SUMAN DEVI*

ABSTRACT — **Several studies have shown how to utilize Social Media for various activities to improve your overall business, reach, acting, art and etc. But Social Media is a hindrance to studies and people look at Social Media as an escape while they are studying. This thought process has impacted people negatively and due to the massive amount of entertainment available at a single click people tend to procrastinate a lot while being in a study session or are giving up on knowledge. The Paper proposes a reliable platform which not only has accommodated every aspect of Social Media but has also removed the procrastinating effects of it and also included a different aspect of gaining knowledge through Articles and Blogs.**

This paper shows how you can build such tools to eradicate all the harm which comes from a typical Social Media platform and convert it into a platform for gaining knowledge. This platform is built on MERN Stack and has utilized Node Package Manager for covering all the necessary tasks which also includes security by using Json Web Token.

To promote a privacy conscious mentality and also for the safety of each and every user we are setting up our whole project on our own Server using Raspberry Pi.
This will help us in securing the data with the most cost effective way and will also ensure independence from any third party app for hosting. This also lay down a base for our future additions for our platform. We will thoroughly explain each and every step to be taken to deploy the site on your own server. This will help a lot of users or companies to establish their own Social Media for better and secure communication.

Result — This platform provides an efficient way to engage and refresh people by also giving them hard to find knowledge at a single click.

It provides GUI interface build with React.js that gives an amazing User Experience. This platform utilizes the knowledge and experience of a user to help other users. On this platform you are an Author and also a Reader which make this an knowledge sharing platform which promotes both writers and readers

I. INTRODUCTION

We are trying to make a Social Media Platform which covers all the pleasures of likes and follows from a typical social media platform but it only revolves around Blogs and Articles about the different types of Topics or Genre. Because of which you are enjoying those pleasures while educating or learning something new through each and every post.

We are working on blog cum social media website, it majorly performs four tasks:

- First, helps Authors to create an account and post their blogs in a convenient and attractive format.

- Second, it also helps users to view others blogs and then users can like, comment, share and also follow the Author.

- Third, it contains Badges for Authors to show their expertise, it will be given to Authors who post content only regarding a particular topic i.e., who stick to their niche and have a certain amount of posts regarding that topic to promote Experts and not Generalized Writing.

- Fourth, it will also contain Groups and tags to categorize Authors, Posts and other Readers.

In conclusion it is a Social Media Platform which helps writers and experts to hone their skills while sharing their knowledge and enjoying all the Likes, Comments

and Followers and on the other hand helps Readers to get knowledge and guidance from Experts of their Fields.

II. TECHNOLOGIES USED

A. MERN Stack

We decided to use MERN Stack for this project so that we can achieve all the goals without worrying about the website load time, slow speed, extra computational power to build this site or unnecessarily difficult UI/UX components. As we use React for our frontend, Node & Express for backend & MongoDB for our Database.

The MERN stack is an amalgamation of MongoDB, Express JS, React JS & Node JS. This combination allows developers to create complete websites (back-end and front-end). With the MERN stack, we use JavaScript on the client side and Node.js on the server side.

The MERN stack was created to give full stack developers the ability to develop a site from start to finish, without having to know or use another skill. Developers can thus create a website or a web application from start to finish, and can manage both the front-end and the back-end. The MERN stack provides the possibility of mastering both the algorithmic and logical part of the backend, as well as the design, user experience and animation components of the front end part.

Below is a diagram that will help you visualize how we distinguish the two blocks that form the MERN stack, with the front-end on the left and the back-end on the right.

1. Developing the back-end with the MERN Stack

MongoDB, Node.js and Express are dedicated to developing the back-end of web

applications. This corresponds to database management, scripts, html documents, HTTP requests, etc.

With the MERN stack, the developers create URLs, such as application / users / create.

On these URLs, they then create, read and modify the data that is stored and retrieved in

the MongoDB database.

These URLs represent functions, with HTTP calls as the originators. The data is sent via

the requests and the server in return will be responsible for modifying the database and sending everything back in JSON format (a format that is very practical because it is readable by JavaScript, the language used by all the technologies making up the MERN stack).

At this point, it's time to start discussing the second block of the MERN stack: the front-end.

2. Developing the front-end with the MERN Stack

React's role is to execute HTTP requests. With React, developers make Ajax calls. This allows them to set up dynamic data downloads without the need for reloading the page. As a result, the web application is made to be much faster than average.

Ajax is really interesting because it acts in an invisible way. The user has the impression

that the data displayed has always been present, whereas thanks to Ajax, it has just

been downloaded. When, for example, you write a comment at the bottom of an article on a site and the site does not reload, it's thanks to Ajax.

B. Raspberry Pi

Raspberry Pi 4 Model B is the most recent item in the mainstream Raspberry Pi scope of PCs. It offers momentous speeds up, sight and sound execution, memory, and network contrast with the earlier age Raspberry Pi 3 Model B+, while holding in reverse similarity and comparable power utilization. For the end client, Raspberry Pi 4 Model B gives work area execution similar to passage level x86 PC frameworks.

This current item's key highlights incorporate an elite 64-cycle quad-center processor, double presentation support at goals up to 4K by means of a couple of miniature HDMI ports, hardware video interpret at up to 4Kp60, up to 8GB of RAM, double band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, also, PoE capacity (through a different PoE HAT add-on).

The double band wireless LAN and Bluetooth have secluded consistency accreditation, permitting the board to be planned into finished results with fundamentally decreased consistency

testing, improving both expense and time to showcase.

III. SOFTWARE DEPENDENCIES

A. *Version Control*:

GitHub is Git, an open source project started by Linux creator Linus Torvalds. Matthew McCullough, a trainer at GitHub, explains that Git, like other version control systems, manages and stores revisions of projects.

Although it's mostly used for code, McCullough says Git could be used to manage any other type of file, such as Word documents or Final Cut projects. Think of it as a

filing system for every draft of a document. Some of Git's predecessors, such as CVS and Subversion, have a central "repository" of all the files associated with a

project. McCullough explains that when a developer makes changes, those changes are made directly to the central repository. With distributed version control systems like Git, if you want to make a change to a project you copy the whole repository to your own system. You make your changes on your local copy, then you "check in" the changes to the central server. McCullough says

this encourages the sharing of more granular changes since you don't have to connect to the server every time you make a change.

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

B. *Hosting*

Heroku is known for running apps in dynos – which are really just virtual computers that can be powered up or down based on how big your application is. Think of dynos as malleable building blocks for running your app.

If you want to process more data or run more complex tasks, you are going to need to add more blocks(what is called scaling horizontally) or increase the size of the blocks (what is called scaling vertically). Heroku then charges you a monthly fee based on the number of dynos that you have and the size of each dyno.

Although Heroku charges you by the dyno, they aren't actually hosting your app. In fact, the entire Heroku platform, as well as every app built on Heroku is deployed to Amazon Web Services(AWS).

C. *Packages:*

Axios (0.21.0) : Axios is a promise-based HTTP client that works both in the browser and in a node.js environment. It basically provides a single API for dealing with XMLHttp requests and node's http interface. Besides that, it wraps the requests using a polyfill for ES6 new promise syntax

Bcryptjs (2.4.3) : bcryptjs lets you hash your passwords means it converts your password to a random string. Bcrypt is an adaptive password hashing function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computational power. In cryptography, a **salt** is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. **Salts** are used to safeguard passwords in storage. Similarly, The salt is incorporated into the hash (as plaintext). The **compare** function simply pulls the salt out of the hash and then uses it to hash the password and perform the **comparison**.

Express (4.17.1) : Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

Express Validator (6.6.1) : Express Validator is a set of Express.js middleware that wraps validator.js, a library that provides validator and sanitizer functions. Simply said, Express Validator is an Express middleware library that you can incorporate in your apps for server side data validation.

Gravatar (1.8.1) : Gravatar (Global Recognized Avatar) is a world wide platform for linking a user profile to all kinds of websites using an email address. The Gravatar API is very simple to use. A short explanation is enough to get ready to use it. First up you have to retrieve the email address of a user (from an input field of a HMTL form for example). Next the email address needs to be hashed with the well-known MD5 algorithm. The hash value will then be used to receive information about the user. And that's it. You can call one of the following API functions via HTTP with a GET request and receive the avatar or other details about the user

Json Web Token (8.5.1) : JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

Mongoose (5.10.7) : Mongoose is an Object Data Modelling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB. Mongoose provides an incredible amount of functionality around creating and working with schemas. Mongoose currently contains eight Schema Types that a property is saved as when it is persisted to MongoDB.

Request (2.88.2) : The request module is used to make HTTP calls. It is the simplest way of making HTTP calls in node.js using this request module. It follows redirects by default. It is easy to get started and easy to use. It is a widely used and popular module for making HTTP calls.

Nodemon (2.0.4) : nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. Nodemon does not require any additional changes to your code or method

of development. nodemon is a replacement wrapper for node. To use nodemon, replace the word node on the command line when executing your script.

Raspberry Pi OS : Raspberry Pi OS is a free working framework dependent on Debian, upgraded for the Raspberry Pi hardware. Raspberry Pi OS accompanies more than 35,000 bundles: precompiled software packaged in a pleasant organization for simple establishment on your Raspberry Pi.

Raspberry Pi OS is a local area project under dynamic turn of events, with an accentuation on improving the soundness and execution of whatever number Debian bundles as could be expected under the circumstances.

IV. SECURITY

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA**.**

Although JWTs can be encrypted to also provide secrecy between parties, we will focus on *signed* tokens. Signed tokens can verify the *integrity* of the claims contained within it, while encrypted tokens *hide* those claims from other

parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

Here are some scenarios where JSON Web Tokens are useful:

1. Authorization: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.

2. Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed— for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

Therefore, a JWT typically looks like the following. xxxxx.yyyyy.zzzzz

Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.

Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.

1. Registered claims: These are a set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject), aud (audience), and others.

2. Public claims: These can be defined at will by those using JWTs. But to avoid collisions they should be defined in the IANA JSON Web Token Registry or be defined as a URI that contains a collision resistant namespace.

3. Private claims: These are the custom claims created to share information between parties that agree on using them and are neither registered or public claims.

Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

The signature is used to verify the message wasn't changed along the way, and, in the case of tokens signed with a private key, it can also verify that the sender of the JWT is who it says it is.

Putting all together

The output is three Base64-URL strings separated by dots that can be easily passed in HTML and HTTP environments, while being more compact when compared to XML-based standards such as SAML.

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned. Since tokens are credentials, great care must be taken to prevent security issues. In general, you should not keep tokens longer than required.

You also should not store sensitive session data in browser storage due to lack of security.

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema

This can be, in certain cases, a stateless authorization mechanism. The server's protected routes will check for a valid JWT in the Authorization header, and if it's present, the user will be allowed to access protected resources. If the JWT contains the necessary data, the need to query the database for certain operations may be reduced, though this may not always be the case.

If the token is sent in the Authorization header, Cross- Origin Resource Sharing (CORS) won't be an issue as it doesn't use cookies.

1. The application or client requests authorization to the authorization server. This is performed through one of the different authorization flows. For example, a typical OpenID Connect compliant web application will go through the /oauth/authorize endpoint using the authorization code flow.

2. When the authorization is granted, the authorization server returns an access token to the application.

3. The application uses the access token to access a protected resource (like an API).

Do note that with signed tokens, all the information contained within the token is exposed to users or other parties, even though they are unable to change it. This means you should not put secret information within the token.

As JSON is less verbose than XML, when it is encoded its size is also smaller, making JWT more compact than SAML. This makes JWT a good choice to be passed in HTML and HTTP environments.

Security-wise, SWT can only be symmetrically signed by a shared secret using the HMAC algorithm. However, JWT and SAML tokens can use a public/private key pair in the form of a X.509 certificate for signing. Signing XML with XML Digital Signature without introducing obscure security holes is very difficult when compared to the simplicity of signing JSON.

JSON parsers are common in most programming languages because they map directly to objects.

Conversely, XML doesn't have a natural document-to- object mapping. This makes it easier to work with JWT than SAML assertions.

Regarding usage, JWT is used at Internet scale. This highlights the ease of client-side processing of the JSON Web token on multiple platforms, especially mobile.

V. FEASIBILITY STUDY

Key points of a feasibility study of this Project

Technical capability: Our team learned to code HTML, CSS & JS in our 2nd Semester & we'll be learning React.JS, Express.JS, Node.JS & MongoDB while working on the project. So, this project will not only be a working phase but also will be a Learning Phase of our Team. We have utilized quite a lot of resources online to build the required skill set needed for this project. Due to COVID-19 there were a lot of Webinars which provided us proper platform

to learn these languages & Framework.

Budget: Each & Every tool used in this project is Open Source and we have hosted our site on Heroku which is free. But we have used Raspberry Pi which costs around 8000 rupees. Which if we compare with the outcome we will come to know that we have signed a cost effective deal. We wanted to prove to the developers out there that projects don't always need hefty investments.

Legality: No legal agreements are required for the process of this software however as far as companies are concerned they can formulate a Term and Condition before installation of the software.

Risk: There is no major risk involved with this project.

Operational feasibility: Social Butterfly is a common term used nowadays and people are getting more and more inclined towards Social Networking without any benefits, this software would outcome those challenges. This project will use the urge of people for using social media for their own benefit by giving then the exciting articles along with the social media.

Time: The time that is allotted by the University is more than enough to complete this project and as the University has already

instructed us to break it down in 2 Phases it would be easier and efficient for us students to make it.

VI. CONCLUSION

Social Media is a bug which affects many lives. It holds enormous power. So, why don't we bend this power for our own good? Why don't we utilize it's addiction for betterment of future? Why don't we teach people with the help of Social Media?

This project gave birth to the revolution we all needed. CodeMerx not only utilizes the power of social media to attract people but it also helps them in learning something new everyday by just a mere click. MERN Stack is the present and future Web Development.

And this project utilizes the MERN Stack to its full potential.

We won't stop developing this project. We'll continue building it in a hope that we can teach and bring change to the society with use of CodeMerx and also change the bad reputation of Social Media to a good one.

VII. REFERENCE

[1] Auth0. *Jsonwebtoken Package*. [Online]. Available from: https://www.npmjs.com/package/jsonwebtoken/ [Accessed 20th October 2020]

[2] Rafal Golawski, Emily Morehouse, Matt Zabriskie, Nick Uraltsev. *Node Package Manager: Packages*. [Online]. Available from: https://www.npmjs.com/ [Accessed 20th October 2020]

[3] Ali Alhaddad. *MERN Stack.* [Online]. Available from: https://medium.com/@alialhaddad/ [Accessed 20th October 2020]

[4] *Raspberry Pi Documentation.* [Online]. Available from: www.raspberrypi.org/documentation