iJRASET
International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 Issue: V Month of publication: May 2024

DOI: https://doi.org/10.22214/ijraset.2024.61983

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Research Paper on Social Media Application Using MERN Stack

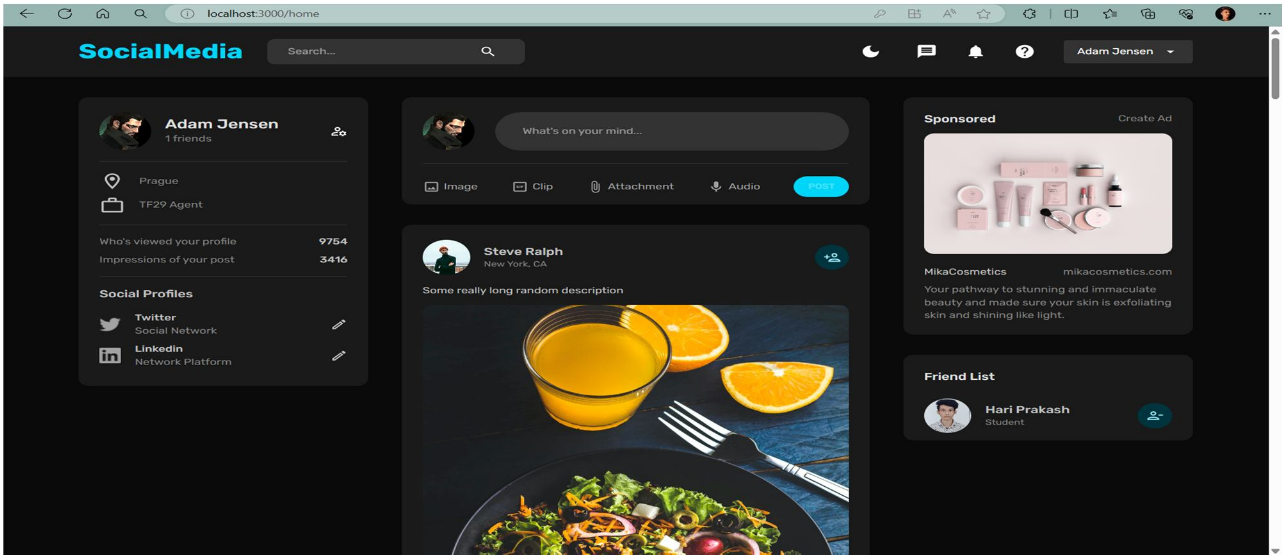
Hari Prakash[1], Manisha Pathak[2]
[1, 2]Computer Science & Madhav Institute of Technology and Science, Gwalior,

Abstract: With rapid increase in the technology it is very essential to understand about emerging computing technology in the world. This research paper will highlight the popular MERN stack which includes four open source technologies i.e MERN (MongoDB, Express, ReactJs, NodeJs). The demand for skilled MERN stack developers is skyrocketing. Companies across industries are recognizing the power and flexibility of this technology stack and are constantly seeking talented individuals who can bring their ideas to life using MERN. We will be discussing MERN Stack Application Social Media App.

Keywords: Social Media Application, MERN Stack, Mongodb, Express, ReactJS, NodeJS, Front-end Development, Backend Development, Single Page Application, Future with MERN Technology

I. INTRODUCTION

Social Media Application is going to be a MERN Application , Dynamic application that interacts with users dynamically by rewriting the current webpage with new web page data instead of the default method of web browser by loading entire new pages. This application will have a login and register system so users can create their account on this platform. It will have three main sections after logging in UserCard, FeedPosts, and FriendsList. Different API calls are done to fetch the data of these sections from the custom backend. You can post, like the posts, make friends and view profiles of any user.



localhost:3000/home

Some really long random description

Your pathway to stunning and immaculate beauty and made sure your skin is exfoliating skin and shining like light.

Fig. 1 Home Page

II. METHODOLOGY

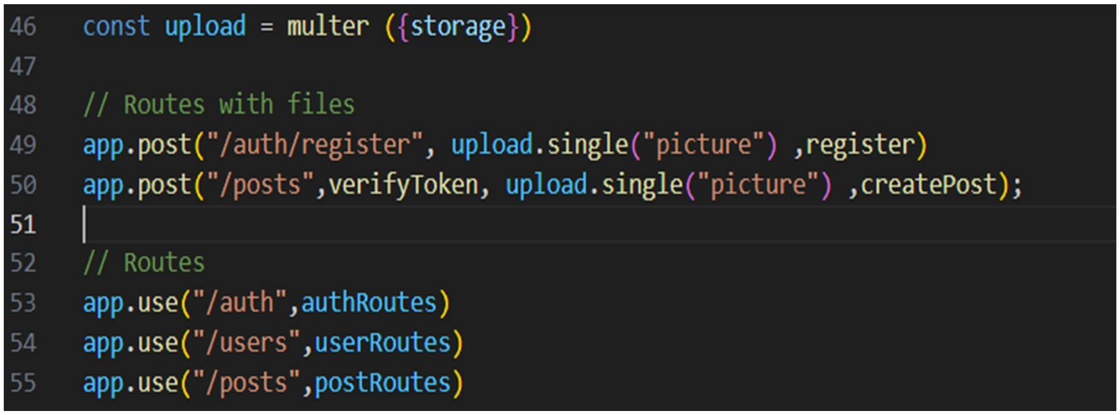
For database storage we used MongoDb, primarily It's open-source and incurs fewer server costs. Open-source is free. NoSQL databases use cheaper servers, so the price of data storage and processing per gig is significantly lower. It allows

1) *Flexible Schema Design:* The document-oriented approach allows non-defined attributes to be modified on the fly. This is a key contrast between MongoDB and other relational databases.
2) *Security Features:* Authentication and authorization are taken into account.
3) *JSON:* Many frontend and API communication systems use JSON. Only using the same protocol for the database makes sense. The schema in MongoDB is simple to use. A specified schema is not necessary to insert data into a NoSQL database, allowing you to modify the data model and formats without causing problems with running programs.

A. *For Backend Development we used NodeJS and ExpressJs.*

Express.js is a popular web development framework for Node.js due to its simplicity, flexibility, and minimalism. It's designed to make building web applications and APIs easier and more efficient by providing a thin layer of fundamental web application features, without obscuring the features of Node.js. Here are some key reasons why developers choose

1) *Express.js:* Simplicity and Minimalism: Express has a very simple design, making it simple to use and learn. Handling HTTP requests, defining routes, and setting up a server are all done effortlessly.
2) *Flexibility and Customization:* It allows you to structure your application based on your preferences, without imposing a strict application architecture.
3) *Middleware Ecosystem:* Express has a large number of middleware functions that can be easily integrated to handle tasks like authentication, logging, and error handling.
4) *Scalability:* It's lightweight and scalable, suitable for both small projects and large-scale applications. Its asynchronous and event-driven architecture can handle many requests efficiently. Routes we have are shown in the figure



```
const upload = multer ({storage})

// Routes with files
app.post("/auth/register", upload.single("picture") ,register)
app.post("/posts",verifyToken, upload.single("picture") ,createPost);

// Routes
app.use("/auth",authRoutes)
app.use("/users",userRoutes)
app.use("/posts",postRoutes)
```

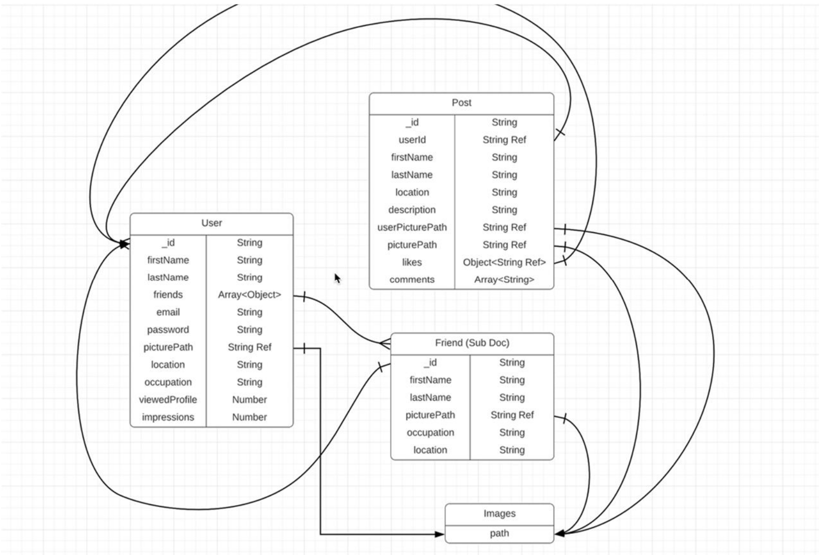
Fig.2 Routes

While registering a user we will need a user profile picture, to upload that picture, also when the user wants to post a picture. Multer Middleware is used

5) *Bcrypt:* Bcrypt is a password-hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher and presented at USENIX in 1999. Bcrypt uses the Blowfish cipher to convert plaintext passwords into a hash, a fixed-length string of characters. This hash is then stored in the database instead of the plaintext password, enhancing security. This method ensures that even if two users have the same password, their hashes will be different due to the unique salt used during the hashing process.
6) *JSON Web Token:* JSON Web Tokens (JWTs) are commonly used for API authentication. They provide a compact and self-contained way to securely transmit information between parties as a JSON object. Whenever a secured route is accessed, an authorization check is there which is done to check whether the user has access or not, with the help of JWT.
7) *Data Modeling:* Data modeling is the process of defining(visually) the structure of a database to represent the data and its relationships accurately, efficiently, and effectively. Several factors are considered during data modeling to ensure that the resulting database meets the requirements of the application and supports its functionalities.

4 data models are designed in the app.

- User Data - It holds the information of registered users.
- Post Data - Whenever a user posts an instance of post is created in the database. Feed posts are displayed through this data.
- Friends Data - It holds friendlist data.
- Picture Path - for user profile.



Post	
_id	String
userId	String Ref
firstName	String
lastName	String
location	String
description	String
userPicturePath	String Ref
picturePath	String Ref
likes	Object<String Ref>
comments	Array<String>

User	
_id	String
firstName	String
lastName	String
friends	Array<Object>
email	String
password	String
picturePath	String Ref
location	String
occupation	String
viewedProfile	Number
impressions	Number

Friend (Sub Doc)	
_id	String
firstName	String
lastName	String
picturePath	String Ref
occupation	String
location	String

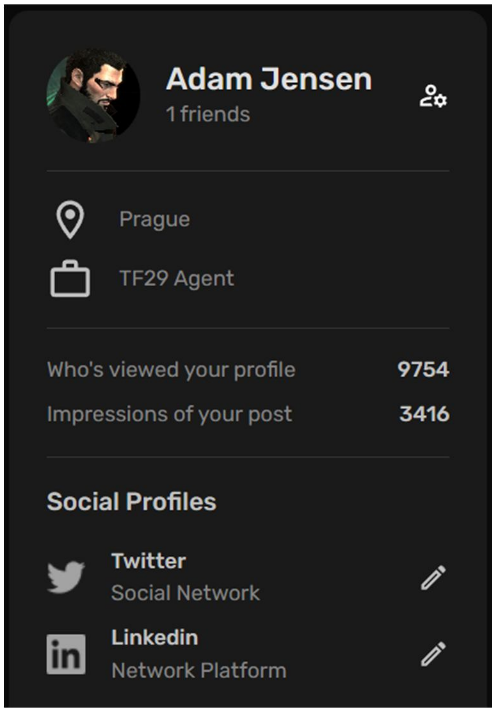
Images
path

Fig.3 Database Modeling

At client side, Login/Register Form Handling is done by Formik is the world's most popular open-source form library for React and React Native. It's designed to simplify the process of building forms by handling repetitive and cumbersome tasks, so you can focus more on your business logic and less on form-related boilerplate.

8) *Intuitive:* Formik uses plain React state and props, avoiding complex subscriptions or observables. This makes debugging, testing, and reasoning about your forms straightforward.
9) *Adoptable:* Since form state is inherently local and ephemeral, Formik doesn't rely on external state management libraries like Redux or MobX. It's easy to incrementally adopt and keeps bundle size minimal. **Battle-Tested Solutions**: Formik provides solutions for input validation, formatting, masking, arrays, and error handling. It's trusted in production by many developers.
10) *Form-Level Validation:* Handle form-level validation and display error messages. Array Fields: Easily manage arrays of form fields.
11) *Custom Validation:* Define your own validation logic.
12) *Internationalization:* Support multiple languages in your forms. Most of the component are used from Material UI library that gives this application a modern look
13) *Material UI:* MUI System provides you with a general collection of wrapper components, such as Box and Container, that are easily changed with the help of the sx prop. With the use of this prop, you can declare styles right within the components themselves compared to having to create large, redundant const definitions with styled-components. Popular companies like Spotify, Amazon, Netflix, Unity, AWS, and Shutterstock opted for Material UI.
14) *Pre-designed Components:* Material-UI provides a comprehensive set of pre-designed and customizable components such as buttons, cards, menus, dialogs, and more. These components are designed following Material Design guidelines, ensuring a consistent and polished look and feel across your application.
15) *Responsive Design:* The responsiveness is taken into consideration when creating Material-UI components. They make it simpler to develop apps that function well on PCs, tablets, and mobile devices without extra effort since they automatically adjust to multiple screen sizes and devices.
16) *Customization:* While Material-UI offers ready-to-use components out of the box, it also allows extensive customization to match your specific design requirements. You can easily override default styles, tweak component behavior, and create custom themes to reflect your brand identity or unique design preferences.
17) *Modularity:* Material-UI components are designed to be modular and composable, meaning you can easily combine them to create complex UI layouts and functionalities. This modularity promotes code reusability and maintainability, as components can be reused across different parts of your application.
18) *Documentation and Examples:* Material-UI provides comprehensive documentation and examples that demonstrate how to use Material-UI components in React applications. The documentation includes clear explanations, code samples, and live demos, making it easy for developers to learn how to use Material-UI effectively in their React projects.

Material-UI integrates seamlessly with React, making it straightforward to incorporate Material Design components and styles into React applications. Here's how Material-UI achieves easy integration with React:


Adam Jensen
1 friends

Prague
TF29 Agent

Who's viewed your profile 9754
Impressions of your post 3416

Social Profiles

Twitter
Social Network

Linkedin
Network Platform

Fig.4 Profile Card

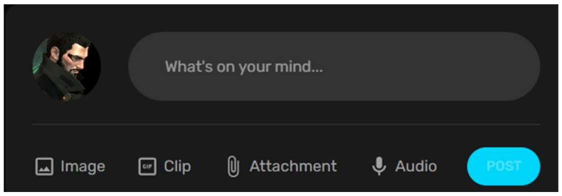

What's on your mind...

Fig. 5 Post Utility

Thanks to MUI for making front end development easier. In essence, the MUI library has not only simplified the process of building beautiful and functional user interfaces but has also raised the bar for what developers can achieve in the realm of web development. Its impact on the industry is undeniable, and its continued evolution promises even greater possibilities for the future of web design and development.

III. FUTURE WORK

My aim is to bring this project to the next level, i'll be working on it in future and add more functionalities such as

1) Commenting
2) Video Uploading in Post
3) Chat System
4) Edit Profile Details

IV. CONCLUSION

The MERN stack has gained popularity because it allows developers to use JavaScript throughout the entire application, both on the client and server sides. This can lead to increased productivity and code maintainability since developers can work with a single language and set of technologies. The SocialMedia application highlights how complex applications can be built using MERN Stack with ease.

The development of our MERN stack application has been a rewarding journey, resulting in a robust and feature-rich platform that fulfills its intended purpose effectively. Throughout the development process, we successfully leveraged the strengths of each technology within the MERN stack to create a dynamic and responsive web application.

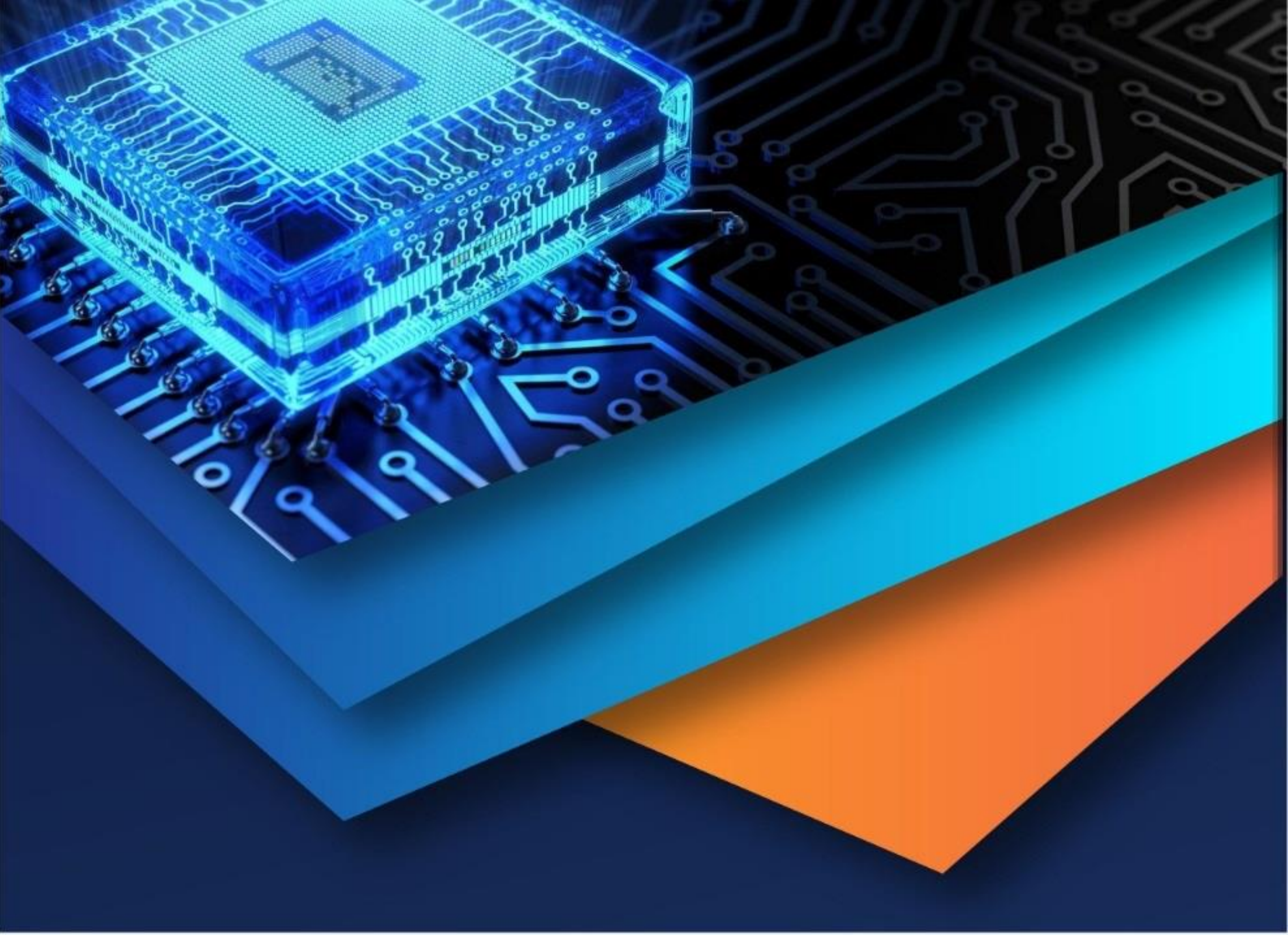
In conclusion, our MERN stack application represents a significant achievement, showcasing the power and versatility of the MERN stack in modern web development. Moving forward, we remain committed to further enhancing and refining our application to meet the evolving needs and expectations of our users.

V. ACKNOWLEDGEMENT

This project work is completed under the guidance Computer Department of Prof. Manisha Pathak and Madhav Institute of Technology of Science, Gwalior

REFERENCES

[1] React The library for web and native user interfaces React
[2] Fast, unopinionated, minimalist web framework for Node.js Express - Node.js web application framework (expressjs.com)
[3] Node.js® , open-source, cross-platform JavaScript runtime environment Node.js — Run JavaScript Everywhere (nodejs.org)
[4] MongoDB: The Developer Data Platform | MongoDB
[5] JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties .JSON Web Tokens - jwt.io
[6] MUI: The React component library you always wanted
[7] Data Modeling - MongoDB Manual v7.0
[8] bcrypt - npm (npmjs.com)



10.22214/IJRASET