Creating a Connected Campus: A MERN-Based Social Media App for College Students

MAANIL LAAD[1], DR. VASUDHA BAHL[2]

[1] *Information Technology, Maharaja Agrasen Institute of Technology, Delhi Maharaja Agrasen Institute of Technology, Delhi*

[2] *Project Mentor, Information Technology, Maharaja Agrasen Institute of Technology, Delhi Maharaja Agrasen Institute of Technology, Delhi*

Abstract- Social media has become an integral part of modern communication, with billions of users worldwide. In this study, we present the design and development of a social media app built using the MERN stack (MongoDB, Express.js, React, and Node.js). We evaluate the performance and scalability of our app, and discuss the benefits and challenges of using the MERN stack for this type of project. Through user testing and feedback, we demonstrate that our app effectively meets the needs of a diverse user base and provides a more personalised and enjoyable social media experience.

Indexed Terms- MongoDB, Express, Node, React, JavaScript.

I. INTRODUCTION

Social media apps can be particularly important for college students, as they can help them stay connected with their peers and communities, share information and resources, and access support and resources. They can help college students stay connected with their peers, both within their academic program and across the wider college community. This can be particularly important for students who are living away from home or who are seeking to build a social network on campus. College students can use social media apps to share information and resources, such as study materials, job openings, and event announcements.

This can help students stay informed and connected to opportunities on campus. College students may also use social media apps to access support and resources, such as mental health resources, tutoring services, and student organisations. These resources can be particularly important for students who are facing challenges or seeking additional support during their college experience.

In this project we aim to create a user-friendly social media platform with MERN (MongoDB, ExpressJS, ReactJS, Node JS) stack. We aim to address the short comings of existing social media platforms and provide a more enjoyable and engaging experience for users. In the following sections, we will discuss the motivation and design considerations behind our app, as well as the benefits and challenges of using the MERN stack for this type of project. Our research also demonstrates the comprehensive approach for effectively testing this application.

II. LITERATURESURVEY

There have been a number of past studies on the development and use of social media apps built with the MERN stack (MongoDB, Express.js, React, andNode.js). Here are a few examples:

“Balasubramani,S.(2019). Building a social network with MERN stack.”This study presents the design and development of a social networking site using the MERN stack, with a focus on providing a user-friendly and responsive interface. The authors evaluate the performance and scalability of the app and discuss the benefits and challenges of using the MERN stack for this type of project.

“Moon, J. (2019). Building a social media application with the MERN stack.” .This case study presents the development of a social media app using the MERN stack, with a focus on implementing real-time features such as live chat and notifications. The authors discuss the design considerations and technical challenges involved in building the app, as well as the benefits of using the MERN stack for this type of project. [2]

"A Comparative Study of MERN Stack and MEAN Stack for Social Networking Sites" (2018): [3] This study compares the MERN stack with the MEAN stack (MongoDB, Express.js, AngularJS, and Node.js) for building social networking sites. The authors present a detailed comparison of the two stacks in terms of performance, scalability, and maintenance, and discuss the benefits and drawbacks of each.

III. METHODOLOGY

A. The MERN Architecture

MERN is a stack of technologies used to build modern web applications. It consists of four main components: MongoDB, a NoSQL database that stores data in a JSON-like format; Express, a web application framework for Node.js that provides a robust set of features for building web applications and APIs; React, a JavaScript library for building user interfaces that can be easily maintained and scaled; and Node.js, a JavaScript runtime built on Chrome's V8 JavaScript engine that allows developers to build scalable network applications. The MERN stack is designed to provide a complete end-to-end solution for building web applications, allowing developers to use JavaScript for both the front-end and back-end of their applications. This makes it easier to develop, test, and maintain web applications, as well as making it easier for developers to switch between front-end and back-end tasks.

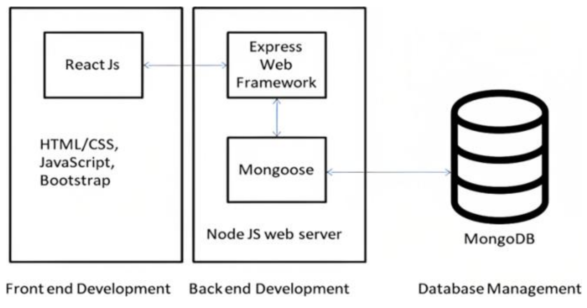


Figure 1: MERN

Architecture Here's how the MERN stack fits together:

The front-end of the application is built using React, which handles the rendering of components and UI elements.

The back-end of the application is built using Node.js and Express. Node.js is used to execute JavaScript on the server, and Express is used to build the API that the front-end will communicate with. The application's data is stored in a MongoDB database, which can be accessed and modified using the API built with Express.

Overall, the MERN stack provides a powerful and flexible way to build modern web applications, and is a popular choice among developers. [4]

B. Application Design

The social media platform we are creating here is a fully functional and responsive web application where a user can publish or see material that is either word or image-based and communicate with other users.

Other users will act as our application's content cens or by extension. The user can upload content that will be available to other connected users and see content that has already been uploaded by other connected users, which is organised by recentness. [5] The user and other connected users can both like or comment on each other's posts or feeds. They have the ability to initiate and respond to connection requests from and to other users, accepting only those they find acceptable. The project's goal is to present a functional model of a real social Media platform. [6]

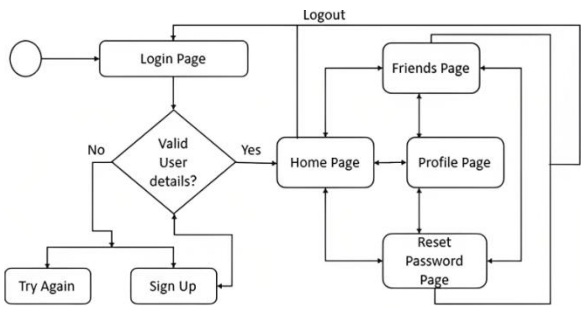


Figure 2: Application Design

1) Models

The models defined in the Node.js server abstract the data in our Mongo DB database. This is the reason. In order to create a blueprint of how we want the extra data to look and behave, we call abstraction on Mongoose schemas. A Mongoose schema is a document data structure that is mandated by the application layer. Models accept a schema and generate a document instance. They are more complicated constructors. A Mongoose model contains the Mongoose schema. A Mongoose model provides a database interface for adding, updating, deleting, and changing records, whereas a Mongoose schema specifies the document's structure, default values, validators, and other details.

- User Model:

All the data necessary or gathered from the user is contained in the User model or User Schema. It is necessary to have the first and last names, a front-end-assigned username, an email address, a password, the user's gender, birth year, month, and day, as well as the created At (date) when the user first joins up. Additionally, it saves as strings the additional (optional) data, such as the Cloudinary URLs to a user's current profile and cover photos. Time stamps are included for a user's friends, followers, following, pending requests, search history, and saved postings. It also includes all the information users enter for their bio, including the bio itself, any alternate names they may have, their job, place of employment, their high school or college, their present city, their hometown.

- Post Model

The Post Model is designed primarily for posts and uploads, as the name implies. A user can upload or alter their profile picture, cover photo, or post in our online application with either plain text, images, textthatincludesimages,ortextthatisshownontopofoneo ftheprovidedimagebackgrounds.Therefore,asastarting point,westorethepost'stype(profilepicture, cover picture, etc.) along with the text (if any, otherwise null), images (an array or null if not any), the user by reference, the background (if any, otherwise null), and the comments received—the text, image (if any), commented by, and the comment timestamp. The time stamps of each post are also saved.

- React Model

There act concept, which was inspired by Facebook's react design, is intended to save all the likes and reactions a user receives on specific articles. The respond kind (such as, love, laugh, sad, furious, and amazed) is saved. Additionally, it saves the user Reference for the user who made the post as well as the user reference for the user who commented on it.

1) View

All of the user interface (UI) components of a web application are defined by the frontend view, which is where we receive all of the data from the controllers and packages and show it in the browser. Additionally, it generates a response for the browser and has all the templates and data fields. We have established a login page, home page, user profile page, password reset page, and friends page in our social app. These pages combine the several separate elements, such as the header, footer, profile image, error pop-ups, create post popups, cards, input forms, and a variety of other things with features that assist us in handling get, post, put, and other requests while updating profile/cover pictures, uploading images, creating posts, adding/removing/searching friends, accessing their profiles, requesting/cancelling/accepting/deleting requests, etc. The UI components, request and data processing routines, reducers and routes handling the fluid back and forth data transactions with the backend, icons and styles, and all other design aspects are present.

2) Controllers

All events triggered by the View are handled by the Controller. After receiving a path, the component gets a JSON response, then sends the object to its child component. The Controller is in charge of custom loading screens, page caching, server-side rendering and prefetching. After processing a request, it responds with a status code and message. For instance, when a user clicks the activation link, we try a piece of code to activate their account. We take the user ID from the account where the request was made and verify our database to see if it is a valid user. The JWT (Jason Web Token) is used to grant access to routes, services, and resources. When uploading images or media, the middleware checks the format and file-size after receiving the request and only if the conditions are met, the controller goes through with processing the request. Other functions include a mailer function to send an email to the user with the activation code.

A. Testing

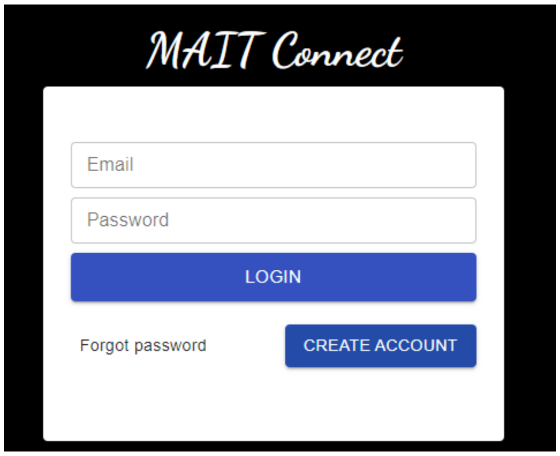
System testing is a type of software testing that focuses on evaluating the end-to-end functionality of a system. It is usually performed after the individual components of the system have been tested and integrated together. The goal of system testing is to verify that the system as a whole meets the specified requirements and functions correctly. Unit testing is a software testing technique that involves testing individual units or components of a software application in isolation from the rest of the application. The goal of unit testing is to validate that each unit of the software application is working as intended and meets the specified requirements.[6]Unit tests are typically small, focused test cases that test a specific feature or behaviour of a unit. They are usually automated and run as part of the development process, and they are designed to be fast and easy to run. Integration testing is a software testing technique that involves testing the interactions between individual units or components of a software application. The goal of integration testing is to validate that the units or components of the software application work together as intended and meet the specified requirements. It is typically performed after unit testing and before system testing. It is designed to test the integration points between the units or components, as well as the functionality of the application as a whole. Validation testing is a type of software testing that focuses on evaluating the accuracy and correctness of the software. It is typically per formed during the later stages of the software development life cycle, after the software has been implemented and integrated. The goal of validation testing is to ensure that the software meets the specified requirements and functions as intended. [7]

Black Box testing is a method for testing an application without having any knowledge of its internal workings. The tester lacks access to the source code and is unaware of the system architecture. Typically, a tester will use a black box test to interact with the user interface of the system by providing inputs and evaluating outcomes without being aware of the location or the method used to process the inputs. [8]

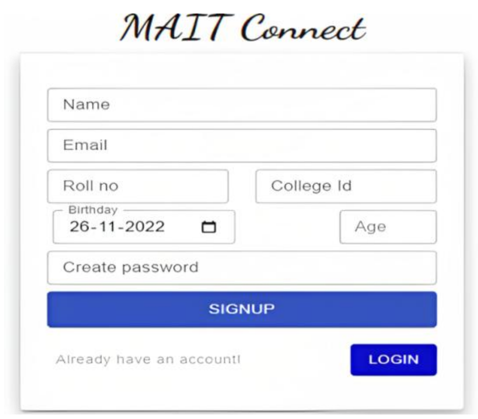
White box testing (also known as glass box testing or structural testing) is a type of software testing that focuses on the internal structure and implementation of a software application. It is based on the assumption [9] that the tester has complete knowledge of the code and design of the software being tested. The goal of white box testing is to validate that the software is correctly implemented and meets the specified requirements. It involves testing the individual components and functions of the software to ensure that they are working as intended and that they are correctly integrated with the rest of the application. [10]

IV. RESULTS

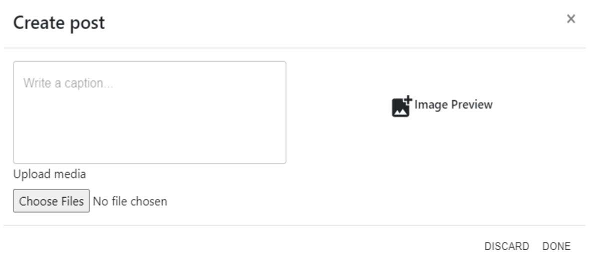
a) Login Page



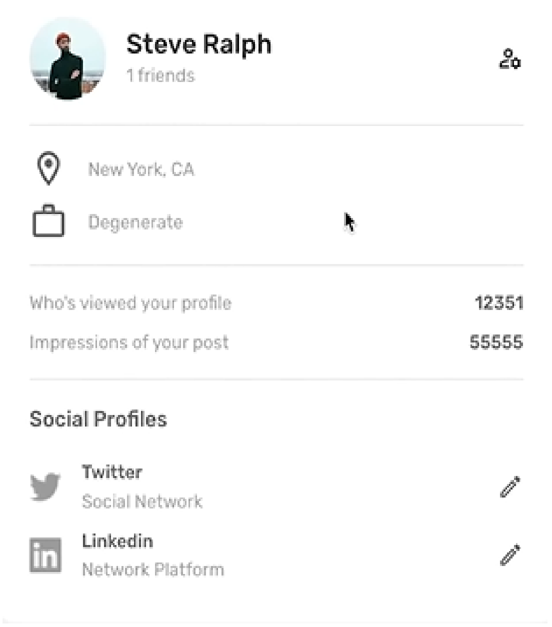
b) Registration page



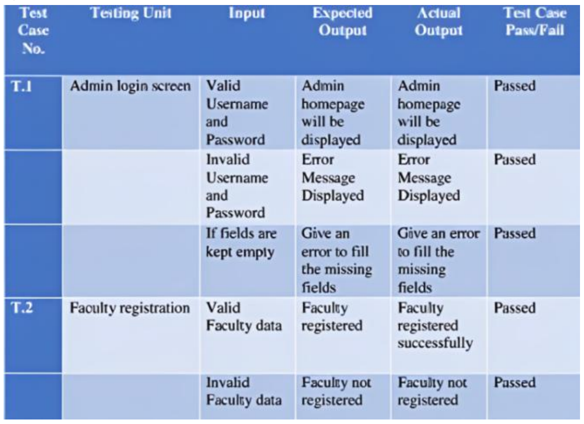
c) Create Post



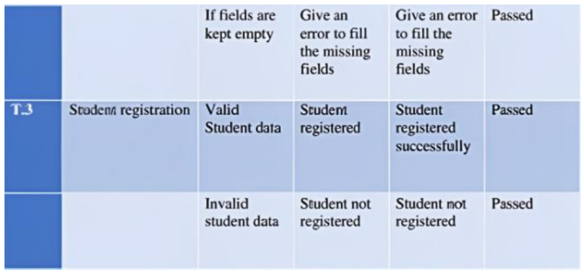
User Profile



d) Test results



Test Case No.	Testing Unit	Input	Expected Output	Actual Output	Test Case Pass/Fail
T.1	Admin login screen	Valid Username and Password	Admin homepage will be displayed	Admin homepage will be displayed	Passed
		Invalid Username and Password	Error Message Displayed	Error Message Displayed	Passed
		If fields are kept empty	Give an error to fill the missing fields	Give an error to fill the missing fields	Passed
T.2	Faculty registration	Valid Faculty data	Faculty registered	Faculty registered successfully	Passed
		Invalid Faculty data	Faculty not registered	Faculty not registered	Passed



		If fields are kept empty	Give an error to fill the missing fields	Give an error to fill the missing fields	Passed
T.3	Student registration	Valid Student data	Student registered	Student registered successfully	Passed
		Invalid student data	Student not registered	Student not registered	Passed

e) Analysis of Results

We created our social media app with functions like creating posts, uploading photos, writing text-based posts, comment, like etc with the help of MERN stack.

We also tested the app on various components like correct password authentication and user registration, responsive design, cross browser testing. There are four pages: a registration page for registration of users, a login page for who already registered a home page to post images, videos or messages, and a profile page to see all the posts that users share with friends. The potential benefits and drawbacks of using the MERN stack: The researches have considered the advantages and disadvantages of using the MERN stack to build the social media app, and have identified best practices or challenges encountered during the development process.

CONCLUSION

This paper includes the implementation of the MERN stack and its characteristics in order to develop an end-to-end social networking web application. Each technology's background ideas, principles, and important techniques are covered in this article. Also discussed are the benefits of such technologies and how they may be used to create a coupled backend and front end application with a NoSQL database engine.

By outlining the steps for creating the social media application, the possibility of applying the above-mentioned ideas to a real-world situation is proved. All of the project's goals are accomplished, and the outcomes are generally good. Following that, detailed implementation methodologies for the social networking application were created.

FUTURE SCOPE

There are many potential directions for future research and development of social media apps built with the MERN stack. Some possible areas of focus could include:

Improving the performance and scalability of the app: As social media apps often need to handle large volumes of data and traffic, there is always a need to optimize their performance and scalability. This might involve exploring new technologies or techniques for optimizing the MERN stack, such as using server less architectures or optimizing database queries.

Enhancing the user experience: Social media apps are often used for extended periods of time, and user satisfaction and engagement can be influenced by factors such as the app's usability, design, and features.Futureresearchcouldexplorewaystoimprovetheuserexperience of social media apps built with the MERN stack, such as by adding new features or refining the interface. Investigating the impact of social media apps on society: Social media apps can have both positive and negative impacts on individuals and society. Future research could examine the ways in which social media apps built with the MERN stack influence user behavior, relationships, and communication, and could identify strategies for minimizing negative impacts and maximizing positive ones.

Developing new security and privacy features: As social media apps often handle sensitive personal information, there is a need to ensure their security and privacy. Future research could explore ways to enhance the security and privacy of social media apps built with the MERN stack, such as by implementing new authentication mechanisms or data protection techniques.

Investigating the use of artificial intelligence and machine learning: Social media apps are increasingly using artificial intelligence and machine learning techniques to personalize content, improve performance, and enable new features. Future research could explore the potential of these technologies in the context of social media apps built with the MERN stack, and could identify new applications and opportunities for their use.

These are just a few examples of the many potential directions for future research and development of social media apps built with the MERN stack. The specific focus of future research will depend on the needs and priorities of the app developers, users, and the broader field of software engineering.

REFERENCES

[1] Balasubramani, S. (2019). Building asocial network with MERN stack. Retrieved from https://www.codementor.io/@sivasubramani/building-a-social-network-with-mern-stack-p80v0hfsf

[2] Moon, J. (2019). Building a social media application with the MERN stack. Retrieved from https://medium.com/@joonmoon/building-a-social-media-application-with-mern-stack-8e34c9b1d84.

[3] A Comparative Study of MERN Stack and MEAN Stack for Social Networking Sites (2018).

[4] Building a social media app using MERN stack (Part 2). Retrieved from https://www.freecodecamp.org/news/building-a-social-media-app-using-mern-stack-part-2-9f2b2c61f84f/

[5] Mernio. (n.d.). Building a social network with the MERN stack. Retrieved from https://mern.io/documentation/tutorials/building-a-social-network.

[6] Bartlett, J. (2018). Testing a MERN stack app. Retrieved from https://www.twilio.com/blog/testing-a-mern-stack-app

[7] Beeman, N. (2019). MERN stack end-to-end testing. Retrieved from https://medium.com/@nbeeman/mern-stack-end-to-end-testing-d6f5c541fdd6

[8] Testing a MERN stack application. Retrieved from https://www.freecodecamp.org/news/testing-a-mern-stack-application-part-1-d6f5c541fdd6

[9] Testing a MERN stack application (Part 2).

Retrieved from https://www.freecodecamp.org/news/testing-a-mern-stack-application-part-2-9f2b2c61f84f

[10] Moon, J. (2019). Testing a MERN stack application. Retrieved from https://medium.com/@joonmoon/testing-a-mern-stack-application-8e34c9b1d84