# Developing a Social Platform using MERN Stack



SUBMISSION DATE / POSTED DATE

09-12-2022 / 12-12-2022

CITATION

Desai, Krutika; Fiaidhi, Jinan (2022): Developing a Social Platform using MERN Stack. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.21699764.v1

DOI

10.36227/techrxiv.21699764.v1

# Developing a Social Platform using MERN Stack

1st Krutika Desai
*MSc Computer Science*
*Lakehead University*
Ontario, Canada
desaik@lakeheadu.ca

2nd Dr. Jinan Fiaidhi
*Department of Computer Science*
*Lakehead University*
Ontario, CANADA
jinan.fiaidhi@lakeheadu.ca

***Abstract*—This is the era of internet and there are a lot of Social Media platforms out there that facilitate the sharing of a vast variety of user generated content. LinkedIn is a business and employment oriented social platform whereas Twitter is used to share recent news trends and updates. Pinterest is about discovering new content and ideas while Facebook is more about catching up with friends and family. These networks and virtual communities cast a tremendous amount of influence on their users. Here we have a similar content-oriented platform called Social. It is designed and built for the users to connect and share digital content (like text, images and/or gifs) related to community, social, healthcare and welfare services. MERN stands for MongoDB, ExpressJS, ReactJS and NodeJS (the four key technologies that make the stack) and is used to built this fully responsive web application in conjugation with other APIs and tools.**

***Index Terms*—MongoDB, ExpressJS, React, Node, Backend, Frontend, dependencies, Social Media, APIs, SPA (Single Page Application)**

## I. Introduction

Interactive Web 2.0 Internet-based applications make up social media platforms. Simply put, they support the proliferation of online social networks by tying a user's profile to those of other persons or organisations. Through online communities and networks, social media platforms enable the development and dissemination of user-generated content that consists of knowledge, ideas, interests, and other kinds of expression [1].

Social Media is not simply an important tool for communication, connecting people locally and globally, it also lets people to share, create, and spread information. These platforms also cast an immense influence on consumer's purchase habits and decisions through their product reviews, marketing tactics and advertising algorithms. Businesses utilize social media to interact with their target market, build brand awareness, create leads, increase sales, and make money. Businesses may effectively target their advertising because to social networks' exclusive access to members' most sensitive information, including their interests, hobbies, and frequented areas. It allows companies to not only engage with the customers on a personal level but also influence them with the appropriate content, triggering desire and formulating purchase patterns [2].

Here we develop a fully functional and user-friendly social platform that does not include any of the business models or customised advertising. It can be used as a tool for various studies on Influence, social network formations, homophily and much more in an unaltered, content controlled environment.

## II. Objective

MERN Stack is a web development framework that uses MongoDB, ExpressJS, ReactJS and NodeJS as its four main technologies. Performance is much more important now because of the explosive growth in the number of electronic devices with real-time and Internet capabilities. Consequently, the web development sector has experienced rapid expansion. The most popular web development frameworks over the past few years have been based on conventional technologies like Servlets, ASP.NET, and PHP. The performance expectations of today's consumers are not met by these technologies, despite their widespread use and lengthy histories of research and deployment. Recently, the MERN stack was developed because of its consistency and simplicity to address this performance issue [3].

The four technologies employed here in MERN stack are all Javascript based in addition to being high-performance and customizable. Therefore, the backend, frontend, and database can be controlled easily if one knows JavaScript (and JSON), which makes the technology a very popular choice. The goal here is to utilize the fore-mentioned technologies to design and implement a fully functional, user-friendly social platform. This web application will not only demonstrate the deployment of these tools but will also provide as a gizmo for further research along the lines of network formation, online behaviour, social influence and much more.

## III. Literature Review

Facebook and Twitter-like social media platforms are starting to have a significant impact on academic publications as well as corporate practises. Over the past few years, there have been numerous academic study papers on the usage of social media in business. Twitter has been studied to see what role it plays in a range of marketing contexts, from promoting brands and enterprises (Greer and Ferguson 2001) to having an impact on political campaigns (Towner and Dulio 2012). The most widely used social media platform, Facebook, has been studied in a variety of areas, like how it affects a company's or brand's value (Plangger, 2012) to fundraising campaigns (Vericat 2010) [4].

Social media platforms are among the simplest and m effective ways to disseminate information because they are widely used and because access to the internet is so qu and inexpensive. As of March 5, 2020, the top 100 vid on YouTube with the keyword "coronavirus" had had m than 165 million views collectively, with 85% of those vie coming from news channels, according to a recent study Basch et al. This clearly demonstrates how the accessibil influence and reach of Social Media effects all major aspects our life. Social networks work in fascinating manner, grow more complex by the day. Using the concept of Homoph social networks facilitate a formation of clusters betwe similar individuals across the world. These miniature netwo usually have influencers that trigger heard behaviour amc the users and active sharers that stimulate a chain or reactio By examining the sentiment occurring in a given social me content and keeping track of a user's information shar... behaviour, one can eventually decipher customer behaviour and buying habits and by extension even predict and formulate group patterns.

This is how news go viral, products go trending and exactly why social media marketing is equivalent to a gold mine for companies right now. That being said, we now know how social media platforms are the single most effective medium to influence people and promote ideas globally.

## IV. Framework (MERN)

MERN stack is a framework used for creating websites (web app development). MongoDB, ExpressJS, ReactJS, and NodeJS make up its functional components. The specific role of each of these elements while creating a web application are listed below:

- MongoDB: The application data is stored in this document-oriented, No-SQL database.
- NodeJS: This is the JavaScript runtime environment that is used to run the JavaScript code on the machine itself, instead of a browser.
- ExpressJS: It is a framework that sits atop NodeJS and is used to create a website's backend using NodeJS functions and structures. NodeJS was created to run JavaScript on computers, not to create websites, so ExpressJS was created to fill that gap.
- ReactJS: It is a library that Facebook built. It is used to build the UI elements that go into a single page web application's user interface.

As shown in Fig 1, the user interacts with the ReactJS UI components in the front-end of the application, which is situated in the browser. The backend of this application, which is located on a server, is served by ExpressJS, which is built upon NodeJS [7].

A request to change data is sent to the Express server, which is built on NodeJS, after any interaction. When necessary, Express fetches information from the MongoDB database and sends it to the application's front end, where it is shown to the user.

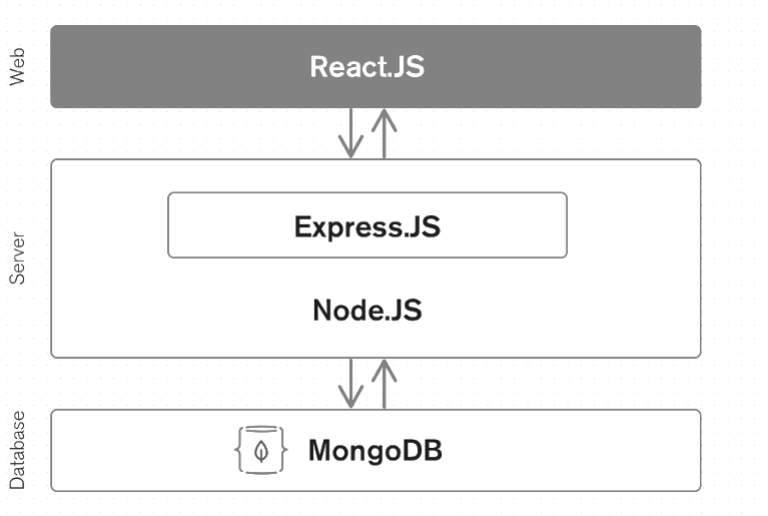


Fig. 1. 3 Tier MERN Architecture.

A single-page web application (SPA) or website interacts with the user and dynamically updates the current web page rewriting the new or modified data from the web server, in contrast to the traditional practise of a web browser loading entirely new pages. The webpage will transition more quickly to boost the appearance of a native app.

As opposed to the traditional way, all essential HTML, JavaScript, and CSS code is either fetched by the browser with a single page load or the required resources are dynamically updated and loaded to the webpage as needed, generally in reaction to user activities. A SPA never refreshes the page [8].

Even using the tools mentioned above, it is difficult to build a high-performance app that is fast, responsive, user-friendly by design and secure, maintaining user integrity and security.

## V. Backend (Server-side)

Building websites and web apps has always been done using server-side rendering, also referred to as back-end web development. When we access a page, we send a request for data to the server, which processes it and sends back a response to the browser.

All the activities required to build an HTML page that the web browser can understand are carried out on the remote server that houses the website or web application when a website renders server-side. This entails processing any required logic as well as information queries from databases for that web application.

While it waits for the distant server to finish processing the request and provide the response, the web browser on the other end sits idle. When a response is sent, web browsers interpret it and show the material on the screen.

## VI. Frontend (Client-side)

Client-side rendering, often known as front-end development, is a new style of site rendering that is employed in contemporary apps. JavaScript, which is now the de facto

standard web language, is used to render the content on your computer as opposed to a distant web server in client-side rendering. In actuality, this indicates that a browser is responsible for generating the HTML output of the web application and that a server is only needed to provide the raw web application. Additionally, it shows that a piece of the presentation logic—the reasoning used to create a web page and display it to the user on the screen—is handled on the client-side.

With the introduction of JavaScript libraries like Angular, React, and Vue, client-side rendering became more common.

## VII. MVC Framework

In direct terms, the backend is the server or, more lately, the cloud, while the frontend is the browser. While the front-end focuses on user interfaces, sound design, and the visual components of developing apps, the back-end is concerned with making websites and web applications render on the client-side. The creation of services that execute business logic and gain access to additional resources, like databases, file servers, cloud services, APIs, and more, falls under the purview of the back-end.

User interfaces, data, and controlling logic are frequently implemented using the software architecture framework known as MVC (Model-View-Controller). It emphasises a division between the business logic and appearance of the application. A better division of labour and better maintenance are made possible by this ”separation of concerns” [9].

- Model: Manages data and business logic.
- View: Handles layout and display.
- Controller: Routes commands to the model and view parts.

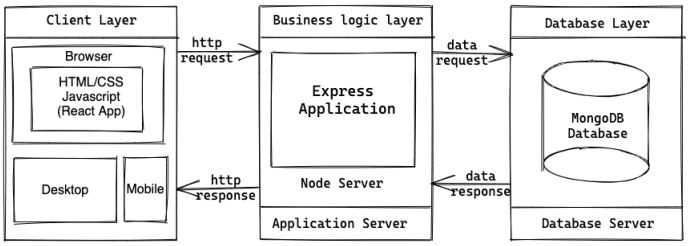


Fig. 2. 3-Tier Architecture

### A. MVC Architecture in MERN Stack

**MongoDB** creates the database layer. The data component is defined by the Mongoose models. This stores all the essential data that the application requires in order to run.

**Express & Node.js** are used to build the Business Logic Tier, and they handle all functional programming (controller). The application server that will serve as a conduit for communication between the client and database is represented by this layer. The React components will be delivered to the user's device by this layer, which will also take user HTTP requests and reply appropriately.

**React** acts as the MVC's ”V,” or view. JavaScript, HTML, and CSS are used to create the Client layer (View), which is built using the ReactJS framework. To access the features of our application, the user will have to interact with this level of the architecture.

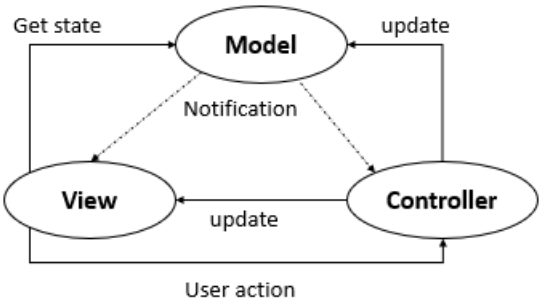


Fig. 3. MVC Architecture

## VIII. Dependencies

A software dependency is a reusable code library or package that allows quick software delivery as it allows developers to building on previous work. The JavaScript runtime environment Node.js uses the Node Package Management, or npm, as its default package manager.It is made up of the npm command-line client and the npm registry, an online repository for free and paid private packages. The client is used to access the registry, and the npm website is used to browse and search for the packages that are currently available [11].

### A. Server-side Dependencies

- **mongodb** - The MongoDB Node.js driver enables Node.js applications to connect to MongoDB and operate with data using an asynchronous API that makes use of Promises or conventional callbacks for interaction [12].
- **mongoose** - A link between MongoDB and the Node.js JavaScript run-time environment is made possible via the object-oriented JavaScript programming package called Mongoose [13].
- **nodemailer** - A Node.js tool called Nodemailer makes sending emails simple. It is a single module with no dependencies, enables embedded attachments in unicode, and has SMTP and OAuth2 authentication for security [14].
- **nodemon** - Nodemon is a programme that facilitates the creation of Node.js-based apps by relaunching the node application whenever a file change in the directory is identified. It is a node wrapper substitute [15].
- **jsonwebtoken** - A concise, URL-safe method of encoding claims that need to be exchanged between two parties is the JSON Web Token (JWT). In a JSON Web Token (JWT), the claims are encoded as a JSON object. This is used as the plaintext of a JSON Web Encryption (JWE) structure or as a payload of a JSON Web Signature

(JWS) structure allowing the claims to be encrypted, digitally signed or integrity protected with a Message Authentication Code (MAC) [16].

- **googleapis** - (Google APIs Node.js Client) is a library for leveraging Google APIs with Node.js. The API endpoints are produced automatically, and authorization and authentication using OAuth 2.0, API Keys, and JWT tokens are supported. For the purpose of user verification, we send emails to newly registered users using Google APIs and OAuth2 playground.
- **express** - Express is a straightforward and adaptable Node.js web application framework that offers a comprehensive selection of functionality for both web-based and mobile applications. It facilitates the creation of a strong API rapidly by providing access to a wide range of HTTP utility methods and middle-ware [18].
- **express-fileupload** - Uploading and downloading files is among the key features of any web app. Here, we'll use the npm package express-fileupload to handle file uploads, and the express library's res.download() function will take care of downloads. The middleware is supplied to the app as express-fileupload [19].
- **dotenv** -Dotenv is a zero-dependency module that loads process.env with environment variables from.env files. The Twelve-Factor App technique is used to store settings in the environment apart from code [20]. We store the unpublished environment variables here like the PORT, the DATABASE_URL, TOKEN_SECRET, BASE_URL, EMAIL (through which verification emails are sent), MAILING_ID MAILING_SECRET (Google cloud API credentials), MAILING_REFRESH MAILING_ACCESS ( OAuth 2.0 Playground Refresher and access tokens), CLOUD_NAME, CLOUD_API_KEY and CLOUD_API_SECRET (for access to Cloudinary to store and access media) that is accessed from the .env file in our code.
- **cors** - A server can specify any origins (domain, scheme, or port) other than its own from which a browser should be able to load resources using the Cross-Origin Resource Sharing (CORS) protocol, which is based on the HTTP header. As a result, we can run the server and client on distinct domains by separating them.
- **cloudinary** - A potent media API is called Cloudinary. The cloudinary package makes it easier to create transformation URLs and offers unique elements and directives that make it simple to incorporate assets into our React application. We can quickly and simply integrate our application with Cloudinary, as well as easily optimise, transform, upload, and manage the assets in our cloud, thanks to the Cloudinary Node SDK [22].
- **bcrypt** - Niels Provos and David Mazières' password-hashing function, bcrypt, which was first presented in 1999 at USENIX, was built on the Blowfish cypher. Bcrypt is an adaptive function that can be slower over time when increased the number of iterations, making it more resistant to brute-force search assaults even with rising computational power [23].

### *B. Client-side Dependencies*

- **axios** - Axios is a promises-based HTTP client library. It facilitates CRUD activities and the delivery of asynchronous HTTP requests to REST endpoints. This REST endpoint/API could be our own backend Node.js server or an external API like the Google API, GitHub API, and so on. The most often API queries are get, post, and delete because we always need to retrieve data to display on our applications as well as add and remove data from and to our API [24].
- **emoji-picker-react** - An emoji picker component for React applications [25]. We provide this for the user to add emojis in texts, comments and posts.
- **file-saver** - The client-side file saving option is File-Saver.js, which is ideal for web applications that produce client-side files [26].
- **formik** - Formik maintains a record of the state of a form and then makes it accessible via props to a few reusable methods and event handlers like handleChange, handleBlur, and handleSubmit. To determine which field needs updating, these handlers look at the name or the id attribute [27].
- **js-cookie** - Cookies are little text files that contain data that are saved on a user's computer. After delivering a web page to a browser, a web server cuts off communication with the client and completely forgets about the user. To address this issue, cookies were created. A straightforward, lightweight JavaScript API for managing cookies is called JavaScript Cookie. It is compatible with all browsers, supports AMD/CommonJS and ES modules, is RFC 6265 compliant, and allows for custom encoding and decoding. It also takes any character [28].
- **react** - React is a JavaScript library used to create user interfaces. Only the functionality required to define React components is contained in the react package. A React renderer, such as react-dom for the web or react-native for native applications, is generally used in conjunction with it. When the data changes, React quickly updates and renders the appropriate components. [29].
- **react-dom** - The server and DOM renderers for React are accessible through this package. It is meant to be used in conjunction with the react package, which contains the basic React components. If necessary, the react-dom package offers DOM-specific functions that can be used as a backdoor to leave the React model [30].
- **react-router-dom** - An npm package called React Router DOM makes it possible to integrate dynamic routing into web applications. It can be utilised to display pages and enable visitor navigation. It is a complete client- and server-side routing library for React. Building single-page applications, or programmes with numerous pages or components but no page refresh since the content is dynamically downloaded from the URL, requires the use of React Router Dom. React Router Dom makes it

possible for this procedure, which is known as routing [31].

- **react-easy-crop** - A free open-source JavaScript library is available under the react-easy-crop package. It supports video formats supported by HTML5 as well as picture formats (JPEG, PNG, and GIF) as a URL or base64 string. It works well for modifying images and movies in front-end applications. It supports drag, zoom and rotate interactions, provides crop dimensions as pixels and percentages and supports any images or video format as url or base 64 string supported in HTML5 [32].
- **react-moment** - It is a lightweight JavaScript date library for parsing, validating, manipulating, and formatting dates [33]. It is a a better way to calculate date and time for our project since we utilize timestamps in several ways throughout. All the post are sorted and displayed by recency and the timeline for each post is visible to every user.
- **react-redux** - The official Redux+JS/TS templates for Create React App, which make use of Redux Toolkit, are the suggested method for beginning new projects with React Redux. React Redux is the name of Redux's official React binding. It enables React components to access Redux Store data so they can update the data by sending actions to the store. Redux assists in project growth by providing a viable means of managing state using a unidirectional data flow design. [34].
- **react-responsive** - A media query module called react-responsive uses CSS media queries as a component or hook for responsive web design. In terms of CSS/Sass styling, we can rearrange our DOM in order to render or remove specific styled elements from the DOM in accordance with the screen resolution/size [35].
- **react-spinners** - Developed with react in mind, React-Bootstrap is a front-end framework. The Spinner Component offers a means of displaying the loading effect. It can be used to display the loading state anytime it's necessary for our application. React-npm-boilerplate is used to bootstrap this package, which consists of a collection of loading spinners built with React.js and based on Halogen [36].
- **redux** - Redux is a design pattern and framework that uses "actions," or events, to manage and update application state.It serves as a central repository for state that must be used across the entire programme, with rules making sure that the state can only be changed in a predictable way. We can manage "global" state with Redux. The state required by numerous components of our application. Redux's patterns and tools help you understand when, when, why, and how the application's state is updated. They also help you understand how those updates will affect the app logic [37].
- **yup** - Yup is a tool for value parsing and validation in JavaScript. We can establish a schema, modify a value to match, check the shape of a value that already exists, or do both. Yes, schema are very expressive and enable for the modelling of intricate, interrelated validations or value transformations. While significantly influenced by Joi, Yup's API is leaner and was developed with client-side validation as its main use-case. Yes, it splits the functions of parsing and validating into different processes. Data is transformed by cast() while input shape accuracy is verified by validate. Each can be carried out either jointly (as with HTML form validation) or individually (such as deserializing trusted data from APIs) [38].

## IX. Application Design

The Social Media Platform we are building here is a fully responsive and functional web application where a user can upload or view image or text based content and connect with other users which by extension will be the content filter for our application. User can view other connected user's uploaded content sorted by recency and can upload content of their own that will be visible to other connected users. The user can like or comment on the posts or feed of other connected users and vice versa. They can send and receive connection requests to and from other users and respond to accept only the ones they approve. The idea of the project is to provide a working model of an actual Social Media Platform deploying a certain set of tools and technologies.

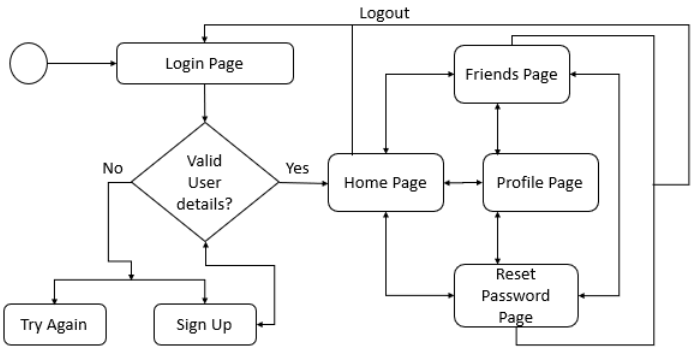


Fig. 4. Application Design - User Navigation DFD

### A. Models

The data in our MongoDB database is abstracted by the models defined in the Node.js server. It is because of this abstraction that we call on Mongoose schemas in order to construct a blueprint of how we want the added data to look and behave. A document data structure that is imposed by the application layer is called a Mongoose schema. Models are more complex constructors that accept a schema and produce a document instance. The Mongoose schema is contained within a Mongoose model. A Mongoose model offers a database interface for creating, querying, updating, removing, and modifying records, whereas a Mongoose schema defines the structure of the document, the default values, their validators, etc. The compiled form of the schema specification that corresponds exactly to a single document

in a MongoDB collection is known as a mongoose model [10].

1) User Model
   The User model or user schema is for all the information required/collected from the user. It requires to have the user's first_name, last_name, username (assigned in the front-end), email, password, gender, birth year (b_year), birth month (b_month), birth day (b_day) and createdAt (date) when the user first signs up. It also saves the other (non-compulsory) information like the Cloudinary links to a user's current profile and cover pictures as strings. A user's friends, followers, following, pending requests, search history, saved posts, all with time stamps. It also has all the details users feed into their bio, namely - the bio itself, other name (if a user has any), job, workplace, high school, college, current city, hometown, their Instagram link, etc. We define a schema and export the model as follows:

$$module.exports =$$

$$mongoose.model(\text{"}User\text{"}, userSchema);$$

   Where the userSchema is defined using mongoose.Schema() and "User" is the model.

2) Post Model
   The Post Model, as the name suggests is for the posts/uploads specifically. In our web application, a user can upload or change his/her profile picture, cover picture, or create a post with either plain text, images, text with images or text on top of one of the give image backgrounds. Hence, for starters, we store the type of the post (profile picture, cover picture, etc.) along with the text (if any else null), images (an array or null if not any), the user by reference, the background (if any else null) and the comments received - the text, image(if any), commented by and the comment timestamp. All the posts also have their timestamps saved.

3) React Model
   Inspired by Facebook's react design, the react model is to save all the likes and reacts a user got on his/her individual posts. It saves the react type (like, love, laugh, sad, angry and amaze). It also stores the post reference of the user who created the post and the user reference of the user who reacted on the post.

### *B. View*

The frontend view defines all the UI elements of a web application where we receive all the data from the Controllers and packages and present it in the browser display. It also has all the templates and data forms and produces a response for the browser. In our Social app, we have defined a login page, home page, user's profile page, a reset password page and a friends page. These pages assemble the several individual components like header, footer, profile picture, error popups, create post popups, cards, input forms and many more with functions that help us handle the get, post, put and other requests while updating profile/cover pictures, uploading images, creating posts, adding/removing/searching friends, accessing their profile, requesting/cancelling/accepting/deleting requests, etc. It has all the design elements like the icons and styles, the UI elements, the request and data handling functions and the reducers and routes handling the fluent back and forth data transactions with the backend. Below is a sample folder structure representing our application:

**Frontend:**

1) Public (icons, images, reacts, etc.)
2) src
   a) Components
      i) Header
      ii) Footer
      iii) Create post
      iv) Post
      v) Error
      vi) Login
         A) Login form
         B) Sign up form
      vii) Bio
   b) Helper functions
      i) Image URL to Blob
      ii) Image cropper
      iii) Close menus/popups on click outside
   c) Functions (to handle requests)
      i) User
      ii) Post
      iii) Image upload
   d) Pages
      i) Home
      ii) Login
      iii) Profile
      iv) Reset Password
      v) Friends
   e) Reducers
   f) Routes (for when logged in and when logged out)
   g) Styles
   h) App.js (Root-component - starting point of our SPA.)
   i) index.css (Universal colors, cards, buttons, etc.)
   j) index.js (Mount/Render the root component)
   k) .env
   l) .gitignore
   m) package.json (With all the scripts, version info and dependencies)

### *C. Controllers*

The Controller handles all the events fired up by the View. The component receives a path, retrieves a JSON answer, and then delivers the object to its child component. The Controller is in charge of page caching, personalised loading screens,

prefetching, and server-side rendering. It processes a request and returns a status code and message for a response. For eg., we try a block of code to activate a user's account when clicked on the activation link. We extract the user Id from the account from which the request has been made and check if it is a valid user from our database. If the account has not been previously activated, and the request is from a valid user, the controller function updates the required fields (in this case verified is turned to true) and

*return res.status(200).json({ message: "Account has been activated successfully." });*

else, if it is a valid user who has already activated their account before, it will do the following:

*return res.status(400).json({ message: "This email is already activated." });*

and if it is not a valid user, it will

*return res.status(400).json({message: "You don't have the authorization to complete this operation."});*

In case an error has occurred during this process,

*res.status(500).json({ message: error.message });*

The backend has other helper functions as well. They help generate the 5 digit activation code, a mailer function to send an email to the user with the activation code, a token generator and other validation functions to check if it is a unique and valid email address, password and username. They also include the middle-ware. Functions that run during the request-response cycle are found in **Middle-ware**. Both the request object (req) and the answer object are accessible to them (res). Middle-ware is executed in the window between a server receiving a request and it sending a response. Utilizing JWT, we use it for authorization (Jason Web Token). Each subsequent request once the user logs in contains the JWT. The middleware then validates this token before allowing the user access to routes, services, and resources. Similarly, when uploading images or media, the middle-ware 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 and the database is updated.

## X. Working on a Real-world Application

### A. Sign-up Process

As per the MVC Architecture, the data is received through the front-end forms, verified and validated by the controller and middle-ware authentication in the back-end and then populated in the database as per the model schema.

**Model**

const userSchema is a mongoose schema for user information.
userSchema =
first_name (required, string),
last_name (required, string),
username (required, unique string),
email (required, string),
password (required, string),
picture (url string),
cover(url string),
gender (required, string),
birth_year (required, string),
birth_month (required, string),
birth_day (required, string),
verified (default : false),
friends (array),
following (array),
followers (array),
requests (array),
search (array with timestamps),
details (bio, other_name, job, workplace, high_school, college, current_city, hometown, instagram, savedPosts (array with timestamps)

**View**

const userInfos = {
first_name = "",
last_name = "",
email = "",
password = "",
birth_year = "",
birth_month = "",
birth_day = "",
gender = ""
}
set user state to userInfos;
get the values from the form as the user populates the fields;
//front-end validation
first_name is required, should be between 2 to 16 characters long and should have only alphabets.
last_name is required, should be between 2 to 16 characters long and should have only alphabets.
email is required and should be a valid email address
password is required and must be 6 to 36 characters long
if all requirements are met, a post request is made passing the data via the controllers, in which case
error = "",
success = data with message
else success = "",
error = response data and message

**Controllers**

try {
//check if the input is a valid email convert the string into lower case and match the format
if not a valid email
return response status 400 and message: "invalid email address";
if the email already exists in the database
return response status 400 and message: "email address already exists";
if first_name, last_name and password are not appropriately long
return response status 400 and message: "must be between *x* and *y* characters long";

password is encrypted using bcrypt and saved
//generate a unique username for every sign-up username = first_name + last_name; while username matches any in the database { username = username + a random number between 0 to 9; } if username is unique, save
a verification email is sent to the user's email for account activation
a jwt token is generated and the user is redirected to the home page.
This token is required for every user activity and/or requests the entire logged in session of that user for authentication purposes.

Similarly, all the requests and responses are handled in a similar fashion, using the Model-View-Control Architecture which keeps all sections of the code separate and sorted, enhances re-usability and readability of the code and help with future modifications, updates, testing and debugging.

## XI. Future Scope

Social Media is an ever growing industry. Information is more valuable than gold. Uncountable tools ranging from NLP, Sentiment Analysis, decrypting patterns, Big Data analysis, micro and macro network studies, human behaviour, impact of influence, triggering heard behaviour and much, much more has branched out ever since the rise of Social Media. The future scope of this particular project has no bounds but restricting our focus towards the short term goals and for better implementation of this project, I would like to work on improving the UI and towards the functionalities this application is offering. It is the first step to build any successful web app. If the User Experience is not rich enough, it will never gain that many participants in order to conduct these further "behind-the-scenes and under-the-hood" studies. As for the long term goals, who's to say, maybe someday we find a way to use the networks to detach people from consumerism instead and to promote and influence more pressing problems as we now know the power Social Media possesses.

## Conclusion

The goal was to build a content oriented social media platform using mern stack. We here, successfully built a social media platform called Social where the user can access, view and generate digital content under the humanitarian genra. The web application is fully functional and responsive and provides great user experience alongside serving a purpose. Social Media today owns the market, casting influence and shaping behavioural and purchase patterns. "The Science of Influence" is a report on how social media influences decision-making, 40% of respondents across generational categories identified social media as having an influence on their travel-related decision-making. Social media was still prominent in other areas, albeit to a lesser extent; 25% of respondents said it was in financial services, 22% said it was in retail, and 21% said it was in healthcare. As a close friend casts much more influence, through the connectivity they offer, social media posts became the most influential source across all generations. We have already witnessed the power of Social Networks. They can alter votes, change preferences, set or banish trends and alter reality altogether with the sheer amount of influence it has over people today. This project uses that same influence to draw and motivate users into indulging their focus towards more significant things like social service, animal welfare, sustainable growth and much more. Social Media casts a domino effect- we see something, we like/buy it, we share it and we reccomend it. Just this time it will be for a cause.

## Acknowledgment

Thanks to Dr. Jinan Fiaidhi, Lakehead University, 955 Oliver Rd, Thunder Bay, ON P7B 5E1, Canada. E-mail: jfiaidhi@lakeheadu.ca The project has emphasized on effective independent re- search and development approaches in computer science including literature exploration, discussion and presenting material, and provided the ability to identify and venture possible new ideas for gaining practical knowledge. I am grateful to have received the opportunity to explore and develop this project under Dr. Fiaidhi's supervision.

## References

[1] Obar, Jonathan A.; Wildman, Steve (2015). "Social media definition and the governance challenge: An introduction to the special issue". Telecommunications Policy. 39 (9): 745–750. doi:10.2139/ssrn.2647377. SSRN 2647377.
[2] Hasan, M. and Sohail, M.S., 2021. The influence of social media marketing on consumers' purchase decision: investigating the effects of local and nonlocal brands. Journal of International Consumer Marketing, 33(3), pp.350-367.
[3] Mehra, M., Kumar, M., Maurya, A. and Sharma, C., 2021. MERN stack Web Development. Annals of the Romanian Society for Cell Biology, 25(6), pp.11756-11761.
[4] Forbes, L. P., Forbes, L. P. (2013). Does Social Media Influence Consumer Buying Behavior? An Investigation Of Recommendations And Purchases. Journal of Business Economics Research (JBER), 11(2), 107–112. https://doi.org/10.19030/jber.v11i2.7623
[5] González-Padilla, Daniel A., and Leonardo Tortolero-Blanco. "Social media influence in the COVID-19 Pandemic." International braz j urol 46 (2020): 120-124.
[6] Basch CH, Hillyer GC, Meleo-Erwin ZC, Jaime C, Mohlman J, Basch CE. Preventive Behaviors Conveyed on YouTube to Mitigate Transmission of COVID-19: CrossSectional Study. JMIR Public Health Surveill. 2020; 6:e18807. Erratum in: JMIR Public Health Surveill. 2020; 6:e19601.
[7] Flanagan, David, "JavaScript - The Definitive Guide", 5th ed., O'Reilly, Sebastopol, CA, 2006, p.497
[8] "Single-page applications vs. multiple-page applications: pros, cons, pitfalls - BLAKIT - IT Solutions". blak-it.com. BLAKIT - IT Solutions. October 17, 2017. Retrieved October 19, 2017
[9] Davis, Ian. "What Are The Benefits of MVC?". Internet Alchemy. Retrieved 2016-11-29.
[10] Matallah, H., Belalem, G. and Bouamrane, K., 2021. Comparative study between the MySQL relational database and the MongoDB NoSQL database. International Journal of Software Science and Computational Intelligence (IJSSCI), 13(3), pp.38-63.
[11] "NPM docs," npm Docs. [Online]. Available: https://docs.npmjs.com/. [Accessed: 01-Nov-2022].
[12] "MongoDB Node Driver". MongoDB. URL: https://www.mongodb.com/docs/drivers/node/current/
[13] "Schemas". mongoose. Url: https://mongoosejs.com/docs/guide.html
[14] "Nodemailer". Nodemailer. URL: https://nodemailer.com/about/
[15] "nodemon". npm. URL: https://www.npmjs.com/package/nodemon
[16] "jsonwebtoken". npm. URL: https://www.npmjs.com/package/jsonwebtoken
[17] "APIs Explorer". Google. URL: https://developers.google.com/explorer-help
[18] "express". npm. URL: https://www.npmjs.com/package/express

[19] ”How to implement file uploading and downloading with Express ?”. Geeks for Geeks. URL: https://www.geeksforgeeks.org/how-to-implement-file-uploading-and-downloading-with-express/
[20] ”dotenv”. npm. URL: https://www.npmjs.com/package/dotenv
[21] ”cors”. npm. URL: https://www.npmjs.com/package/cors
[22] ”Cloudinary Node SDK”. npm. URL: https://www.npmjs.com/package/cloudinary
[23] Provos, Niels; Mazières, David; Talan Jason Sutton 2012 (1999). ”A Future-Adaptable Password Scheme”. Proceedings of 1999 USENIX Annual Technical Conference: 81–92.
[24] ”Axios”. npm. URL: https://www.npmjs.com/package/axios
[25] ”Emoji Picker React (v4)”. npm. URL: https://www.npmjs.com/package/emoji-picker-react
[26] ”FileSaver.js”. npm. https://www.npmjs.com/package/file-saver
[27] ”Formik Docs”. Formik. URL: https://formik.org/docs/overview
[28] ”JavaScript Cookie”. npm. URL: https://www.npmjs.com/package/js-cookie
[29] React. URL: https://reactjs.org/docs/getting-started.html
[30] ”React-DOM”. React. https://reactjs.org/docs/react-dom.html
[31] ”React Router Dom”. npm. https://www.npmjs.com/package/react-router-dom
[32] ”react-easy-crop”. ”npm”. https://www.npmjs.com/package/react-easy-crop
[33] ”react-moment”. npm. https://www.npmjs.com/package/react-moment
[34] ”React Redux”. npm. https://www.npmjs.com/package/react-redux
[35] ”Build Responsive Web Pages With React-Responsive And TypeScript”, OpenReplay. https://blog.openreplay.com/build-responsive-web-pages-with-react-responsive-and-typescript/
[36] ”React Spinners”. npm. https://www.npmjs.com/package/react-spinners
[37] ”Redux”. Redux. https://redux.js.org/tutorials/fundamentals/part-1-overview
[38] ”Yup”. npm. https://www.npmjs.com/package/yup
[39] Fig 1: ”MERN Stack”. Java-T-Point. https://www.javatpoint.com/mern-stack
[40] Fig 2: ”The Ultimate Guide to MERN Stack”. ALT-Campus. https://altcampus.school/guides/the-ultimate-guide-to-MERN-stack