Blog Website Design

Matthew Thibault

CS628 Full Stack Web-Dev Application, STC (School of Technology and Computing), City University of Seattle

[thibaultmatthew@cityuniversity.edu](mailto:thibaultmatthew@cityuniversity.edu)

**Abstract**

Weblogs (blogs) are a popular means for recording the thoughts and ideas of users. Many platforms have used the blog content format to successfully engage their user base and audiences. The goal of this project is to create a simple but effective blogging website that allows users to create, edit, and delete posts and add a variety of content such as images to their posts. Each user should have full control over their own content while having only viewing rights to other user’s content. The backend for this project should store the posts and user data.

**Keywords:** MERN, React, MongoDB, Node.js, Express, REST, CRUD

1. **INTRODUCTION**

The use of blogs began in the mid 1990’s and slowly grew in popularity as the internet grew in prominence. In the early days, a number of smalltime internet journals cropped up as individuals sought a way to record their thoughts and ideas. In those early days blogs were almost entirely text based and might only contain links to other topics that the author found interesting. Since then, though, blogs have exploded in popularity. They are used by all manner of individuals to share their ideas with audiences big and small.

The point of a blog is to serve as a website that provides discrete portions of information. Those could take the form of new articles, short stories, diary entries, cooking recipes or anything in between. In their simplest form they are a journal of sorts, allowing their users to record their thoughts in an informal way. However, the amount of refinement that can be applied to blogs on popular platforms such as WordPress is significant, and it can transform the blogging experience for both the author and the readers.

Today it is estimated that as much as 83% of internet users read blog posts. Relatively short snippets of information are favored now and more people get their news and entertainment through short-form media than ever before.

1. **LITERATURE REVIEW**

Blogging platforms were much more common in the late 90s and 2000s. That is also where they have seen the majority of their innovation. In order to gain an understanding of what makes a good blogging platform it is useful to look through the history of some of the most popular ones and see how they have evolved over time.

WordPress was initially created in 2003 and began as a fork of a previous blogging platform called b2/cafelog. The idea behind WordPress was to make a more user-friendly and flexible platform for blogging and content management. By 2004, WordPress had started to gain popularity, particularly among bloggers. The platform was open source, which allowed developers to contribute their own code. This led to a rapidly growing ecosystem of users and developers. Also in 2004, the Themes feature was introduced, allowing users to change the look and feel of their WordPress sites easily. In the mid-2000s, WordPress evolved from a blogging tool into a full-fledged content management system (CMS). It was also around this time that plugins were added to the platform, greatly improving the range of functionality. From there it continued to expand into the CMS direction.

Blogger was launched in 1999 by Pyra Labs. The platform was one of the first easy-to-use tools that allowed users to create and publish blogs without requiring any technical knowledge, thereby becoming one of the first platforms to make blogging accessible to a wide audience. Several years later in 2003, Google acquired Pyra Labs and Blogger which allowed the platform to benefit from the improved infrastructure and resources. During the early 2000s, Blogger grew rapidly in popularity as it became the go-to platform for individual bloggers. It introduced many features that are still common in blogging today including customizable templates and community features. The former allowed users to easily change the appearance of their blogs using pre-designed templates, while the latter allowed users to easily follow and interact with other blogs. Its popularity began to decline in the 2010s due to competition, unlike WordPress it did not expand into CMS. Instead, it was integrated into other Google tools like Google Analytics and Adsense.

From the 2010s onwards, blogging slowed down and many blogging platforms were forced to adapt to the changing internet scene. The increased compute power and internet infrastructure allowed more dense forms of media, namely video, to dominate over the simpler blogs of the previous decade. Combining blogs and video together was the natural evolution and vlogging began to grow. YouTube, which was created in 2005 was designed as a platform for users to upload and watch a variety of user-generated videos. By 2010, YouTube had cemented itself as the dominant video-sharing platform, with millions of people uploading content. The culture of vlogging began to grow rapidly as creators started getting millions of views and has continued to grow all the way to the present day.

1. **METHODOLOGY**

The form that blogs can take varies quite widely depending on the capabilities of the platform and the goals of the author. However, there are some common aspects, some simple, some not, that are core to the medium. First, and most importantly the blog must be able to accept, store, and display many articles. Each article should be an element of its own and should be interactable. How the individual articles are stored and how they are interacted with and retrieved are design choices. Articles should have an owner who has editing and deletion permissions (as well as other possible permissions such as hiding or privating for instance), while all other users should only have read access.

Originally blogs were focused almost entirely on written content, however as internet speeds have improved and the internet has matured, it’s become much more common to provide a mixture of content containing images and video. A good blogging platform should be able to accept and store other forms of content (at least images) and display them as a part of the article that they are a part of. Other images may be included in more generalized areas such as a part of a navigation bar or set as a background to be used across the entire domain, which brings us to the next point.

To make a blog a wholistic application rather than a loose collection of articles by one or more authors, it is desirable to organize the individual elements. The most basic form of organization is to give each user their space (domain) that they can use to manage their own articles and provide a means for users to navigate to and through each other’s domains. This could be done by allowing a user to register their own domain and simply install the blog application into it (WordPress), thus achieving a degree of separation from the rest of the internet. Alternatively, an organization can register a domain and then provide subdomains to their users who create their blogs in their space (akin to MySpace or Youtube). Either way, this also opens the possibility for higher level blog customizations. One of the aspects of blogs that make them popular is the high degree of control that an owner can have in managing the look, feel, and navigation of their own space. WordPress is an excellent example of this with their large body of themes and templates. Themes are used to give the whole domain a consistent appearance and are great for setting a blog apart from others and making it recognizable at a glance. Templates are used for individual pages and organize the content on that page to meet the needs for that specific article. One common theme that many people easily recognize because it is prevalent in many applications is darkmode. As its name suggests, this theme changes the color scheme of its application to contain many darker colors (usually black) in the background and makes the writing appear white. This has the effect of improving readability with lower ambient light and reduces eye strain.

So, to summarize the key points for making an effective blogging application:

1. Individual articles should be distinct from each other.
2. Articles should be capable of displaying a variety of content including text, and images.
3. Navigation within a blog should be provided and optionally navigation to other blogs may be provided depending on the scope of the application.
4. The blog should be customizable, either through themes and templates or with finer tuned settings options that allow greater control over individual elements.
5. **DEVELOPMENT**

When blog applications are operating at their full potential, they can see traffic from thousands of individuals per day. To make a scalable accessible platform, it is necessary to divide the workload between a web application and a server. Adhering to this model will allow the application to be purpose built with specialized technologies for all the goals it needs to meet. The web application can manage user interaction and experience while the backend will handle data management and retrieval. MERN is a typical tech stack for this sort of design and is well suited for our purposes, it consists of:

* MongoDB (database)
* Express (backend)
* React (frontend)
* Node.js (backend)

This set of technologies utilizes JavaScript on both sides of the framework which simplifies development and maintenance. React is a JavaScript library for designing user interfaces. Node.js is a JavaScript runtime environment that can be used for the server-side. Express is a Node.js framework for handling server-side logic. MongoDB is a NoSQL document database.

**Frontend**

Please refer to Appendix A for context as you read through the description of the user interface. The entirety of the front end of this project was written using React. In order to meet the goals of the project in a clear and navigable manner, it was decided that several routes would be needed, and they would be managed by a BrowserRouter component. Individual routes were provided for the home page, the timeline page, the add posts page, and the settings page. In addition, the edit post page is set as a child of the timeline page, and its route is determined using URL parameters that pass the id number of the post. The home page is simply a splash page for the domain, and serves as a redirect location in case another portion of the code fails for some reason.

The Add Posts page is where the author can create new posts (articles), giving them a title and the desired written and image content. The exact format of the text is preserved using a <textarea> component during creation and the style setting: whitespace: pre-wrap at rendering time. This allows the author to have full control over the formatting of their written content. After a post is created, it is added to the full list of posts for that user and displayed on the timeline page.

Posts are a separate component that contain the formatting for their content. They are rendered on the timeline page in order and contain several functions of their own. Each post has a button for deleting and editing. If the delete button is pressed a confirmation window pops up and requests confirmation before removing the post from the storage. The edit button redirects the author to the edit post page where the existing content is autoloaded into the fields. The interface is very similar to the add posts page, although there is an additional cancel button, which simply ignores any pending changes and returns the author to the timeline page.

The settings page is where the author can control some aspects of their blog’s appearance. Some colors for the background and navigation bar can be manipulated to change the look and feel of the blog as a whole. This is done using high level variables declared in the .css files and are used throughout the styling to make a consistent color scheme. On the React side the useEffect() hook allows these settings to update the .css variables in real time. Besides color schemes, several options for font styling are available.

The navigation bar is provided through the use of a <NavLink> component and is available on every page in the blog. It allows easy access to all of the high-level domain paths such as the home, timeline, add posts, and settings pages.

**Backend**

The server side of the application consists of express, node.js, and MongoDB. Node.js and Express together have all the tools for managing server requests from the frontend. Meanwhile MongoDB is responsible for storing and serving data requested from the server. There are a few key files that make the server function.

Server.mjs is a node application that starts the server and sets it listening on the designated port. It also instantiates express and ties URL endpoints to express routers. This forms the basis for all the routing we will need. Additional .mjs files are used to manage different routes. In this project, post (objects) have their own router that will control the flow of requests from the React app to the database. Endpoints that are declared include:

* GET (“/”): Retrieves all posts.
* GET (“/:id”): Retrieves a post by id.
* POST (“/”): Creates a new post item
* PATCH (“:id”): Updates an existing post
* DELETE (“/:id”): Deletes a post by id

Each of these will transform a JSON request they receive into a query and await a response. Upon receiving it, they will transform the response back into a JSON document and send it back out to the React app. JSON formatted documents form the basis for all communications between the client and server in this project.

In our case we used MongoDB Atlas as our database which is cloud based. So in order to access it, we have one more critical file. Conn.mjs is responsible for establishing a secure connection to the database using the MongoClient class. Communication between the server and database is hidden and managed by the MongoClient class, greatly simplifying the network communications.

1. **CONCLUSIONS**

The development of this blogging website successfully demonstrates the core functionalities required of a modern blog platform. By leveraging the MERN stack—MongoDB, Express, React, and Node.js—this project delivers a scalable and user-friendly application that enables users to create, manage, and personalize their blog content with ease. The system allows for seamless creation, editing, and deletion of posts and provides users with the ability to customize the appearance of their blogs through color schemes and fonts. The separation of concerns between the frontend and backend ensures maintainability and scalability, while the use of RESTful APIs and JSON formatting streamlines data communication between components.

The project meets its primary objective of providing a simple yet effective blogging platform that gives users control over their own content. It lays a strong foundation for further enhancements, including advanced customization options, authentication mechanisms, and richer interactive features.

1. **FUTURE WORK**

Additions to this project fall into three major categories: function, customization, and interaction. Additionally, functionality could include login capabilities with authorization and authentication. Access to sections of the blog could be contingent on this. Customization could come in many forms, truthfully there is no end to the amount of additional customization options. From expanding the theming choices (colors, fonts, textures) to offering unique page layouts to post formatting. Lastly, user experience is a high concern. More interactive layouts and components can improve the experience and maintain freshness. Interactable expanding and shrinking cards for each blog post could be an interesting addition in this category.

1. **REFERENCES**

Gunn D. (2024, March 13). The history of Blogging: From 1997 Until Now (With Pictures*)*. *Themeisle*. <https://themeisle.com/blog/history-of-blogging/#gref>

Jordana A. (2024, January 14). How to use WordPress: A complete guide to build a WordPress website. *Hostinger*. <https://www.hostinger.com/tutorials/how-to-use-wordpress>

**Appendix A: User Interface Screenshots**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Timeline Route**

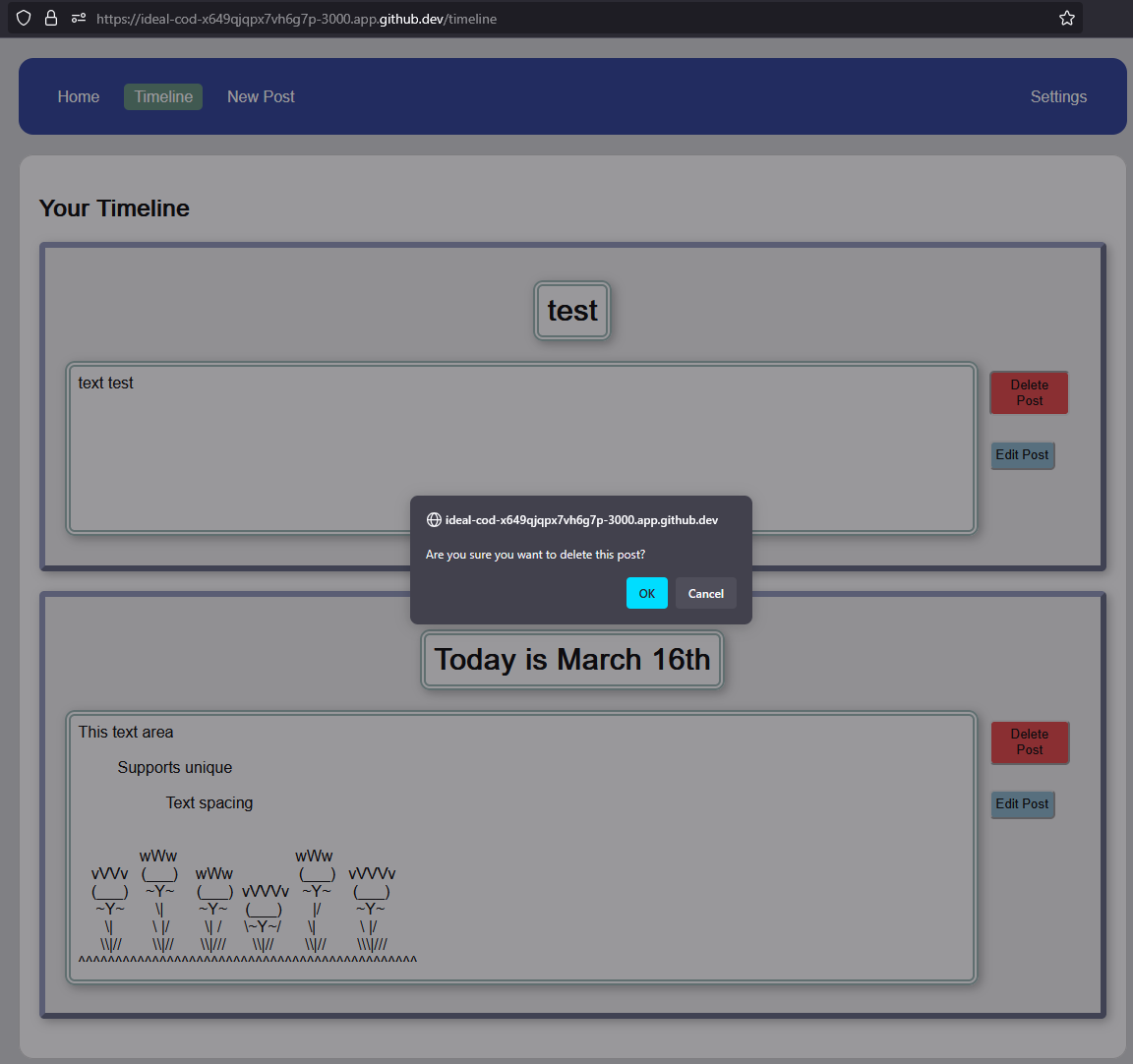
**A screenshot of a computer

AI-generated content may be incorrect. New Post Route**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Timeline After New Post**

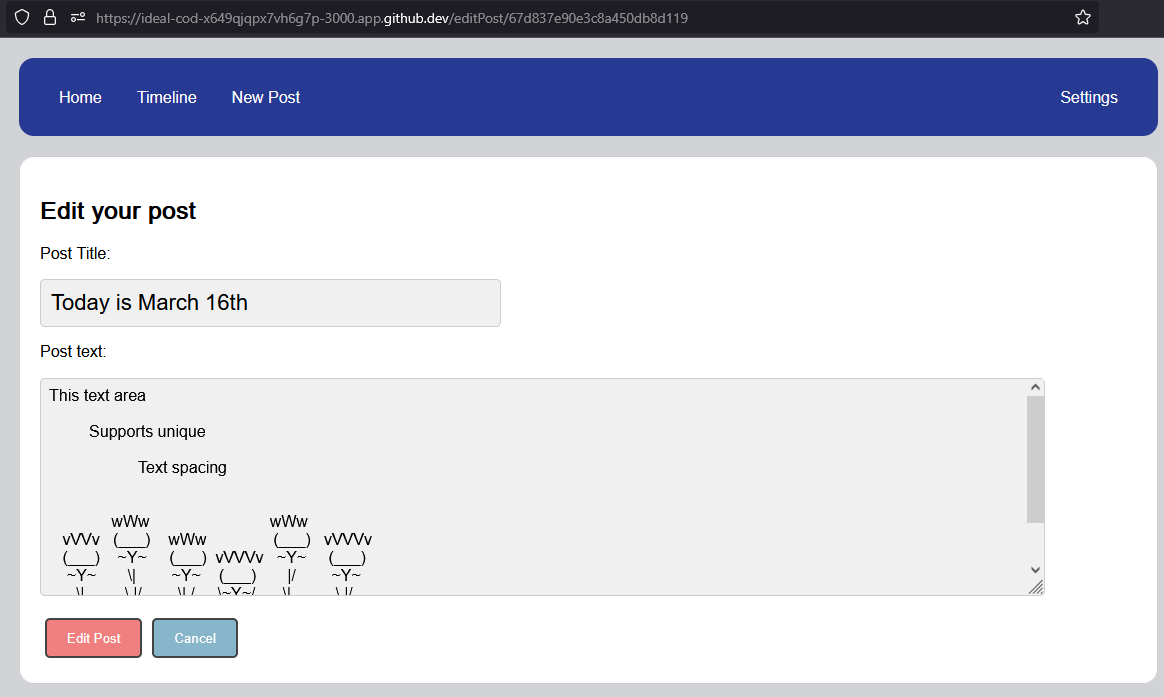
****

**Timeline Post Functionality**

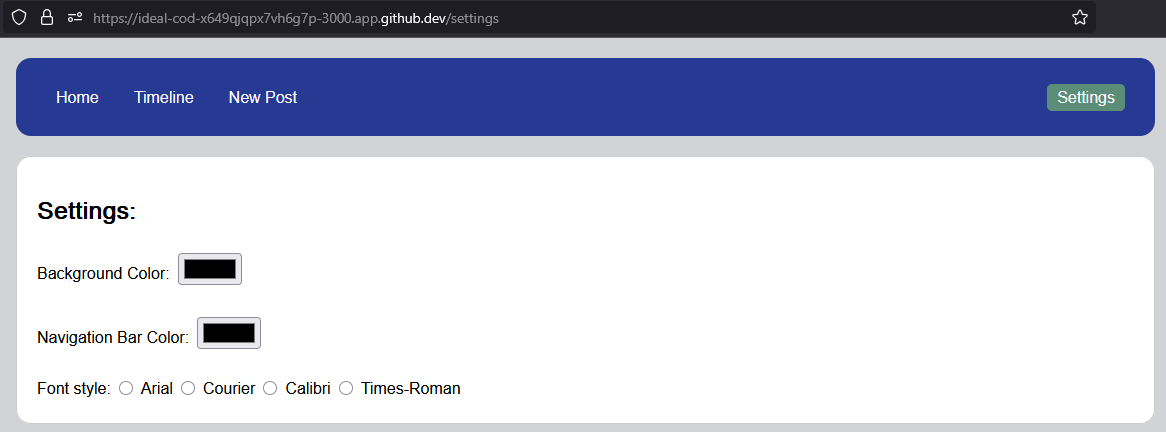
**A screenshot of a computer

AI-generated content may be incorrect.**

**Timeline After Deleted Post**

****

**Edit Post Functionality**

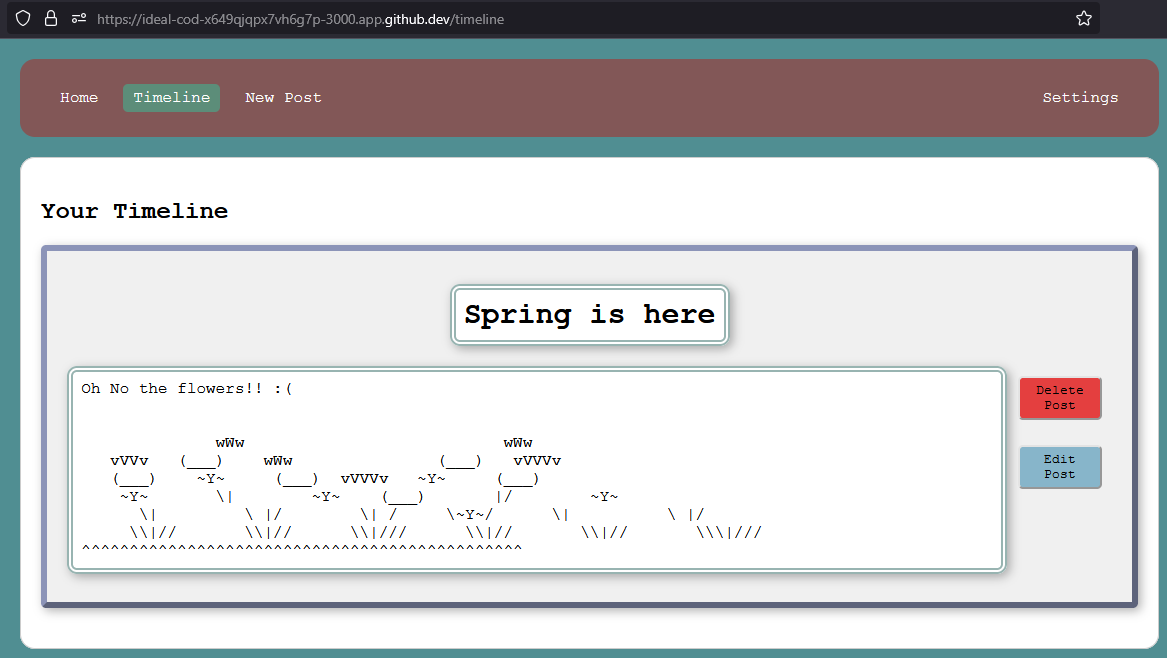
****

**Settings Route**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Updated Color and Font Settings**

****

**Timeline Final View**