# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
|  | **TITLE** | **PAGE** |
|  | **ABSTRACT** | **ii** |
|  | **INTRODUCTION**    **EXISTING SYSTEM**    **PROPOSED SYSTEM**    **METHODOLOGY**    **IMPLEMENTATION**    **CONCLUSION** | **ii**  **ii**  **iii**  **iii**  **ix**  **xiv** |

**ABSTRACT**

Blog Application - Build a blog application end to end with modules such as login, sign up, new post addition deletion updation, search etc. In our new website we have introduced many new features for the blogger to enjoy his/her way of creating the blogs and we have also improved the security of our blogging website.

**INTRODUCTION**

Blog application is a software that allows users to create and publish blogs on the internet, typically using a web browser. Blog applications allow users to easily create new blogs, manage their blogs, and publish content.

Blog application can be used for personal or business purposes. A personal blog application may be used to share personal thoughts and experiences with the world, while a business blog application may be used to share company news, product announcements, and more.

Blog applications allow users to quickly and easily create new blogs, manage their blogs, and publish content. Because blog applications are web-based, they are easy to use and can be accessed from anywhere in the world.

**EXISTING SYSTEM**

Currently, in many blogging websites there is no login page. Also, there are more flaws and security issues are present. And so, the website present today are not more flexible to the user to change the options. Present days, all the blogging websites use a heavy theme which affects the performance of that website.

**PROPOSED SYSTEM**

In my website, the user can able to access more options and it is flexible to the user to change or insert any information to his/her blog. Also, I have addressed may security features in my blogging website. I have also included a login page for the exiting users login and a sign up page for the new users login. You can also sign in to this blog using your gmail id. Here, I kept my website as a light-weighted theme which greatly improves the performance of my blogging website.

**METHODOLOGY**

I have implemented my project using the React App and node js. I have divided each segment into each various files and I linked it with javascript and css. If you click on the options, it will redirect you to a new section on that website, where you can browse and add or remove or edit your blogs. Also, all your activities are can be noted and can be synced to the google account or to the cloud database. I have also created a database which can be connected to the mongo db for the registration and retrieval of users data.

**MODULE DESCRIPTION**

**Some of the Features of the website :**

**User Functions :**

* **Register or Login with Email**
* **Register or Login with Google**
* **Register or Login Facebook**
* **Update Password**
* **Forgot Password**
* **Reset Password**
* **Update Password**
* **Save Blog**
* **Get Any User Profile**

**Blogs Functions :**

* **Create Blog**
* **Update Blog**
* **Delete Blog**
* **Comment on Blog**
* **Get Blog Details**

**Technology Used :**

**Frontend :**

**React, React-redux, React-hooks**

**Backend :**

**Node, Express**

**Database :**

**Mongodb (mongodb Atlas - for deployment)**

**Deployment :**

**Heroku**

**To send reset-mails :**

**Nodemailer**

**How to use this project :**

**Step 1 : Download the zip file**

**Step 2 : UnZip the folder and open the folder in VS Code or any editor**

**Step 3 : Open a terminal and split it in to two (in vs code)**

**One terminal is for client and one for server respectively.**

**Step 4 : Inside both the terminal run the following commands :**

**npm install**

**Step 5 : Go inside the package.json of client folder and chnage proxy to localhost**

**"proxy": "http://localhost:5000"**

**Note : The above step is necessary to run the server locally.**

**Step 6 : Add .env file inside the server folder , and add these environment variable in it :**

**CONNECTION\_URL = ''**

**CLOUDINARY\_NAME=""**

**CLOUDINARY\_API\_KEY=""**

**CLOUDINARY\_API\_SECRET=""**

**SMPT\_SERVICE='gmail'**

**SMPT\_HOST='smtp.gmail.com'**

**SMPT\_PORT=465**

**SMPT\_MAIL=""**

**SMPT\_PASSWORD=''**

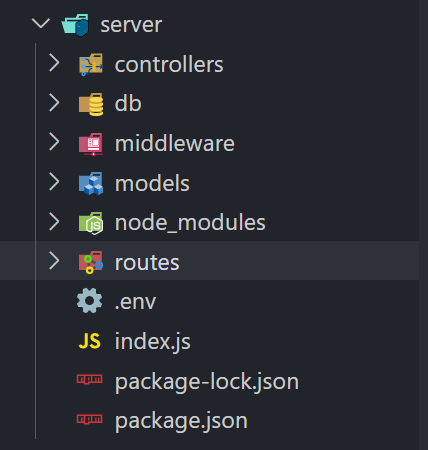
**BASE\_URL='http://localhost:3000'**

**Step 7 : Add Mongoose Url to connect to database**

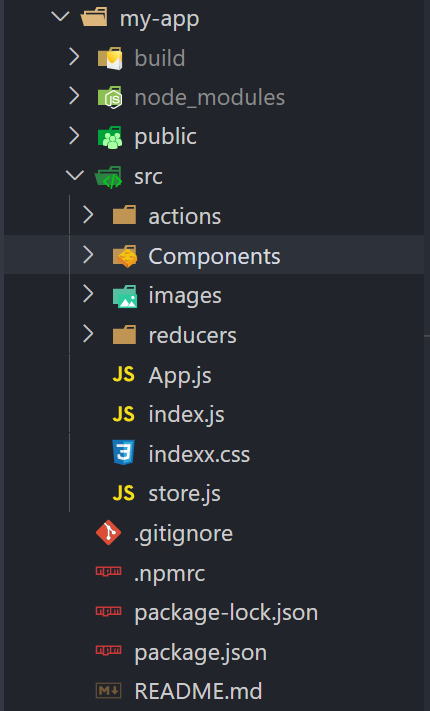
* **For Database, hosted locally : mongodb://localhost:27017/blogdb blogdb - is the name of the database**
* **For Database, hosted on cloud : Get the url, and add it to .env file**

**Step 8 : Folder Structure :**

**Server Folder Structure :**

**[](https://user-images.githubusercontent.com/47392217/192132530-7d91d352-e2f4-4271-a6c0-3b8b53eb705a.png)**

**Client Folder Structure :**

**[](https://user-images.githubusercontent.com/47392217/192132563-6aa2481c-7062-429b-bbb2-79310f73e60e.png)**

**Step 9 : Run the following commands to execute the codes :**

* **Inside client folder terminal : npm start**
* **Inside server folder terminal : npm run dev**

**Hyper Text:** HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

**Markup language:** A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

**Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

* CSS stands for Cascading Style Sheet.
* CSS is used to design HTML tags.
* CSS is a widely used language on the web.
* HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity. Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

JavaScript is used to create interactive websites. It is mainly used for:

* Client-side validation,
* Dynamic drop-down menus,
* Displaying date and time,
* Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
* Displaying clocks etc.

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

Node.js is similar in design to, and influenced by, systems like Ruby's [Event Machine](https://github.com/eventmachine/eventmachine) and Python's [Twisted](https://twistedmatrix.com/trac/). Node.js takes the event model a bit further. It presents an [event loop](https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/) as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through callbacks at the beginning of a script, and at the end a server is started through a blocking call like EventMachine::run(). In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

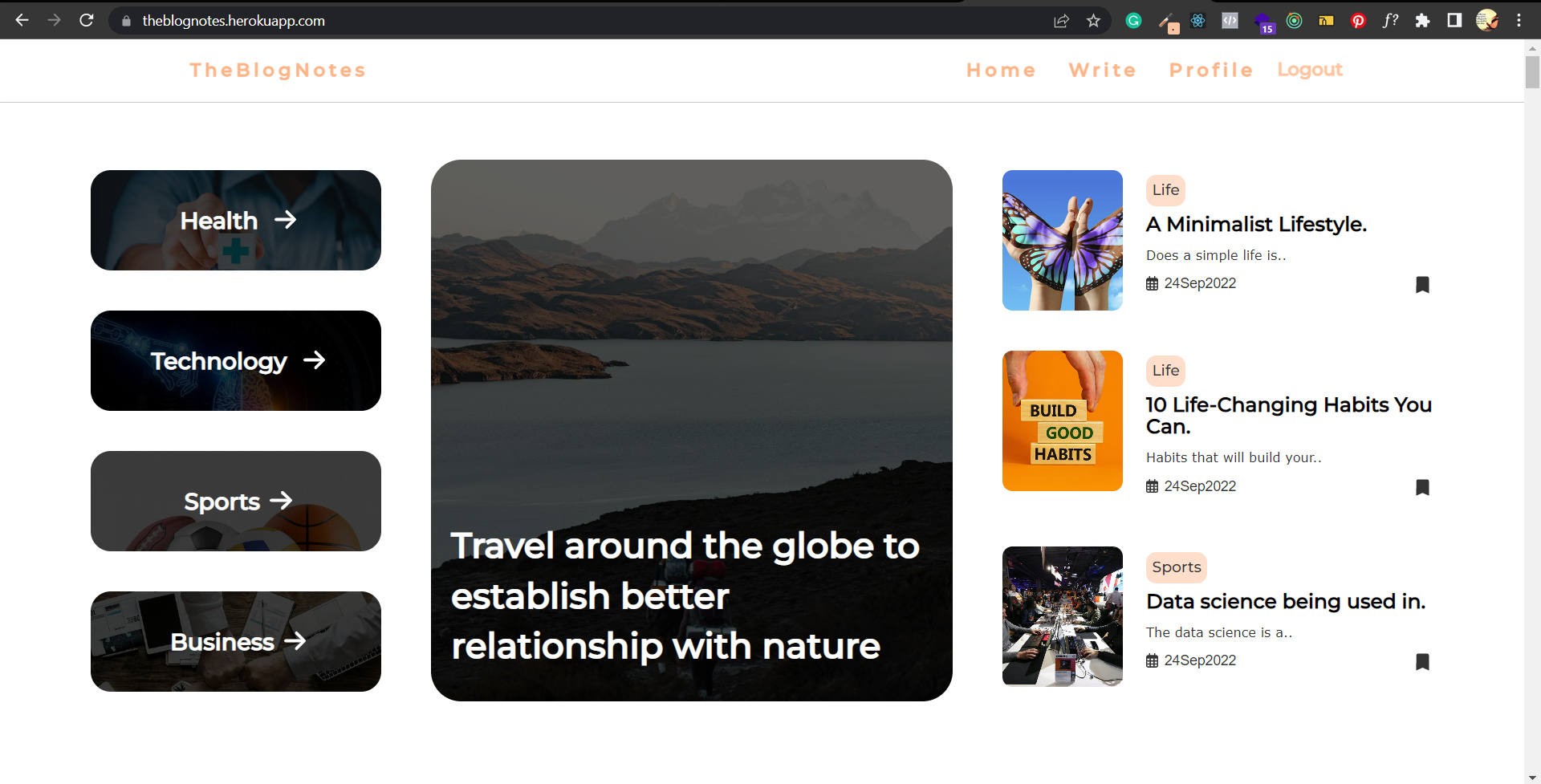
HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.

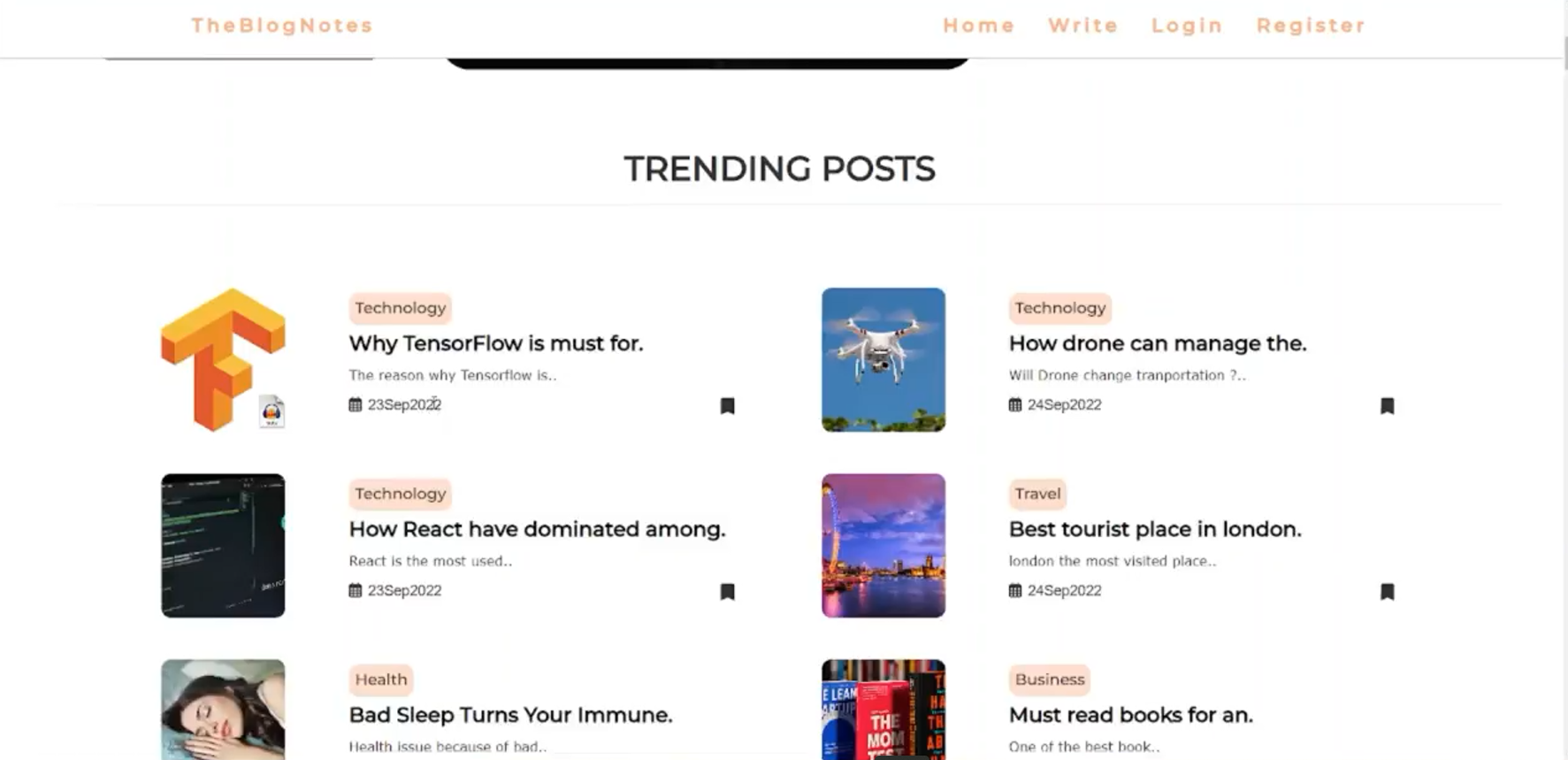
Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our [child\_process.fork()](https://nodejs.org/api/child_process.html#child_process_child_process_fork_modulepath_args_options) API, and are designed to be easy to communicate with. Built upon that same interface is the [cluster](https://nodejs.org/api/cluster.html) module, which allows you to share sockets between processes to enable load balancing over your cores.

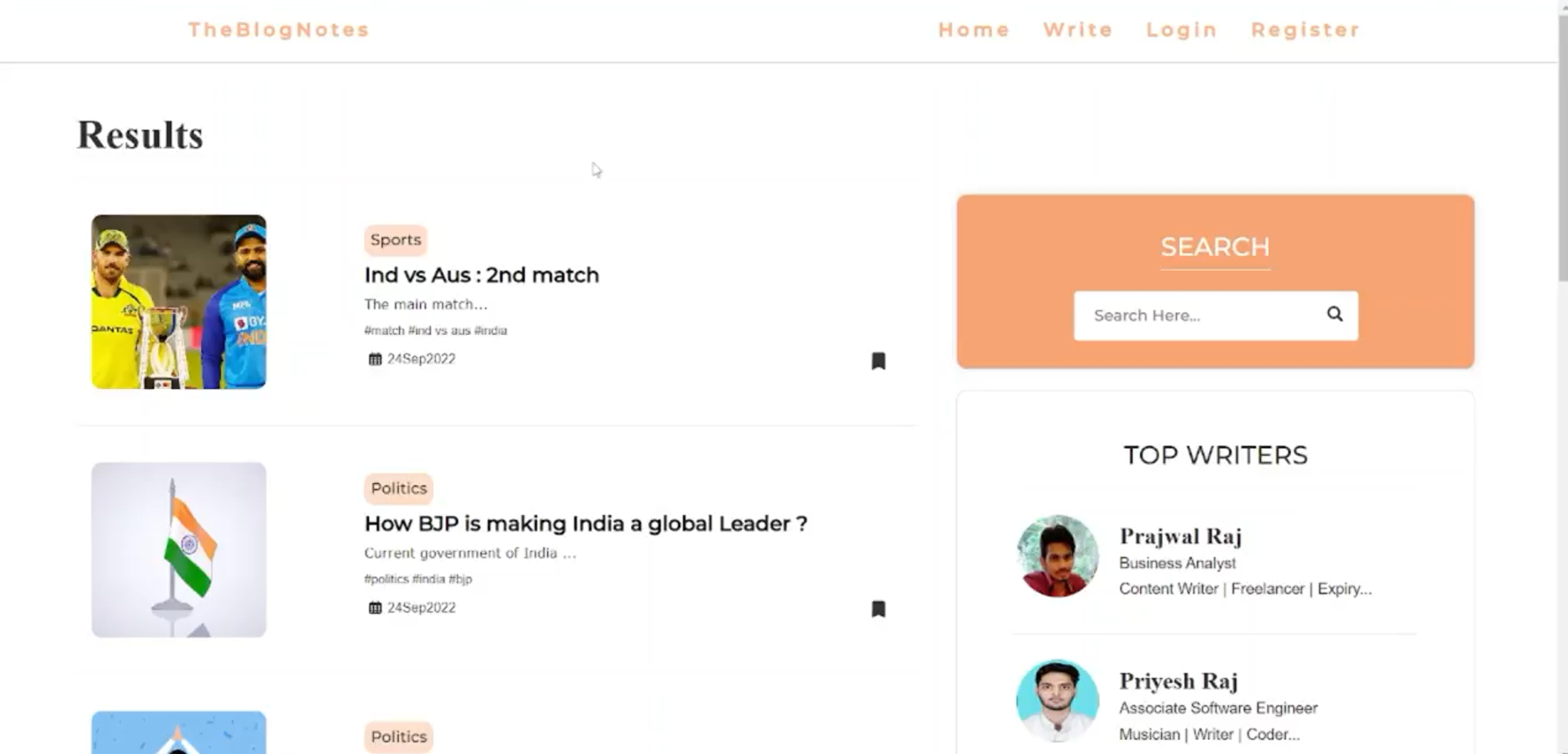
[**ReactJS**](https://reactjs.org/) is one of the most popular [**JavaScript**](https://www.hostinger.com/tutorials/what-is-javascript) libraries for mobile and web application development. Created by Facebook, React contains a collection of reusable JavaScript code snippets used for user interface (UI) building called components.

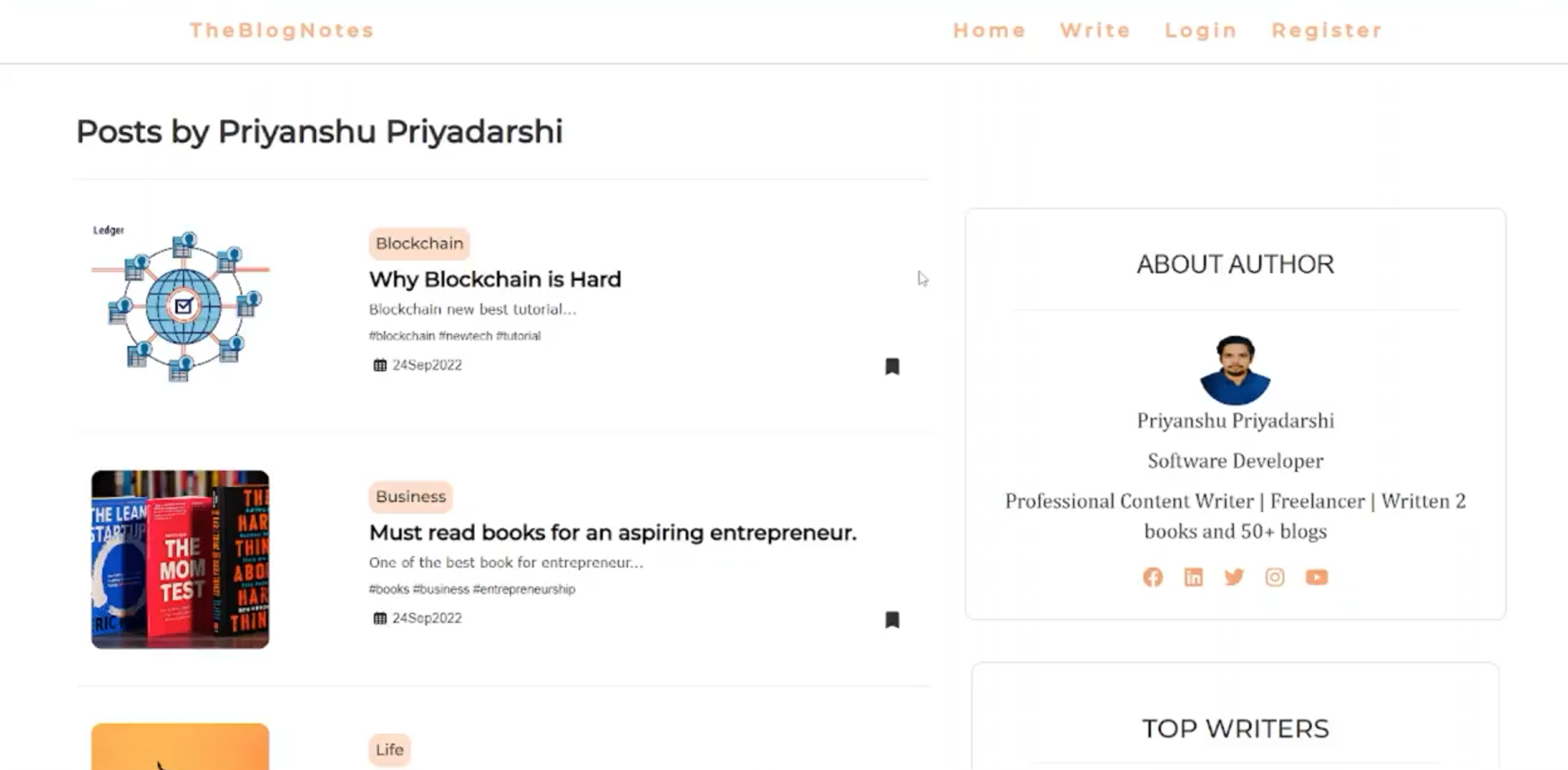
It’s important to note that ReactJS is not a JavaScript framework. That’s because it’s only responsible for rendering the components of an application’s view layer. React is an alternative to frameworks like [**Angular**](https://www.hostinger.com/tutorials/what-is-angular) and [**Vue**](https://vuejs.org/), which all allow to create complex functions.

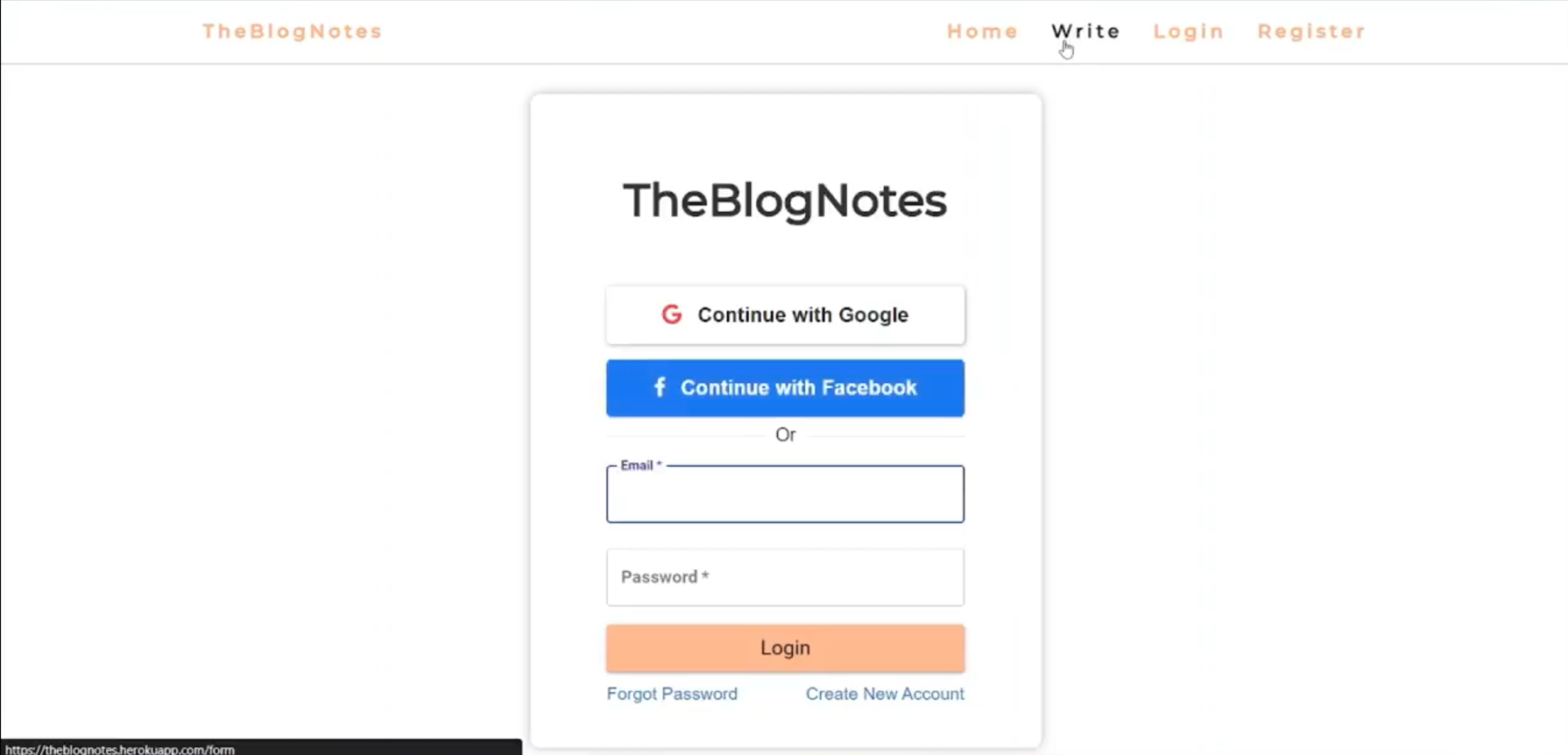
**IMPLEMENTATION**

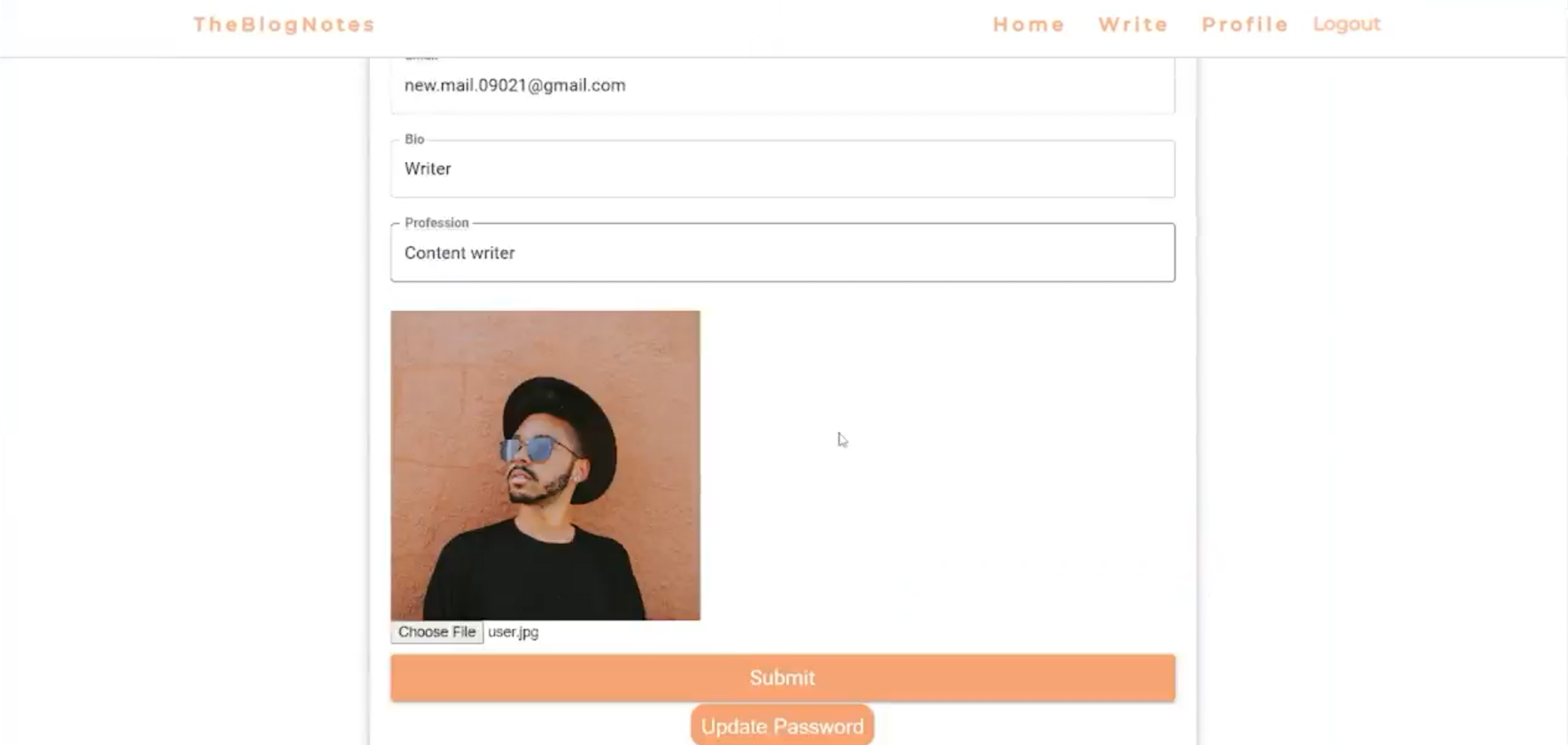


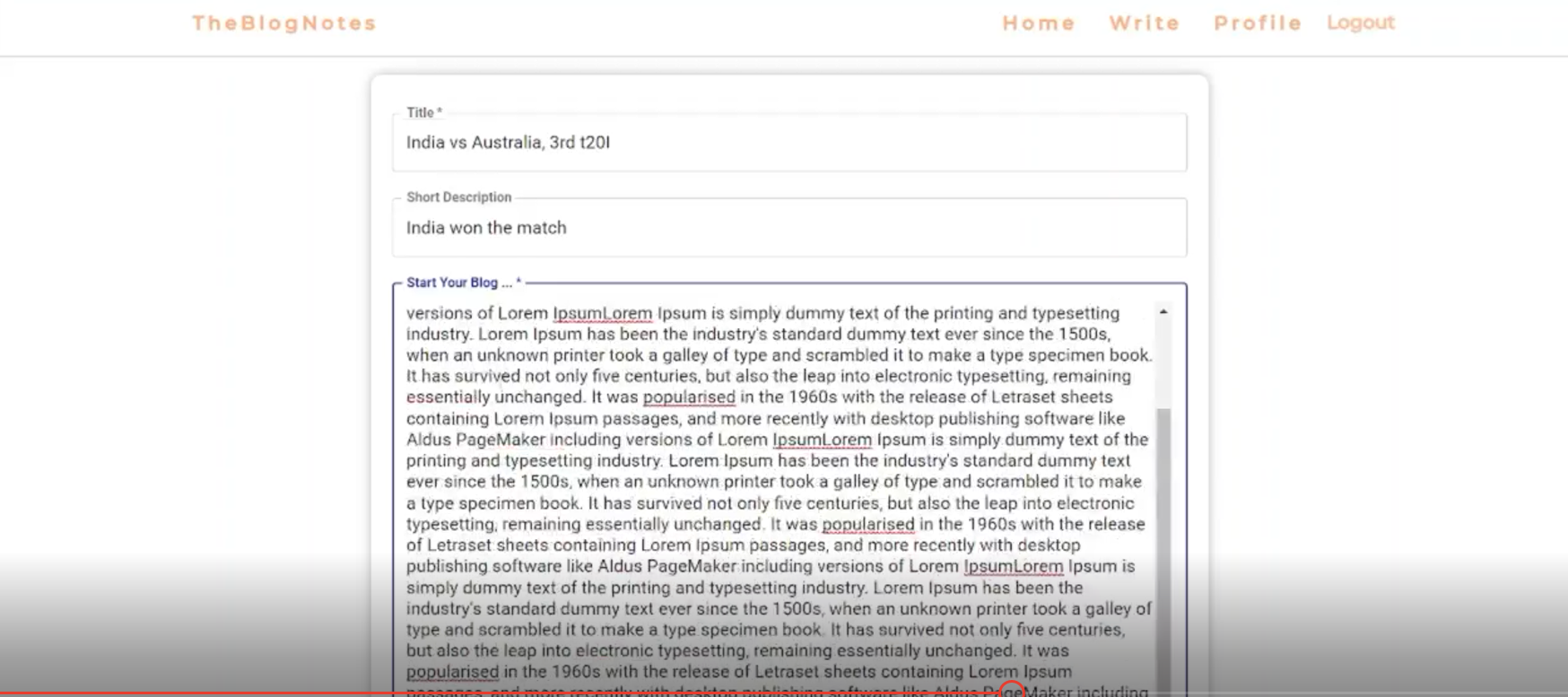




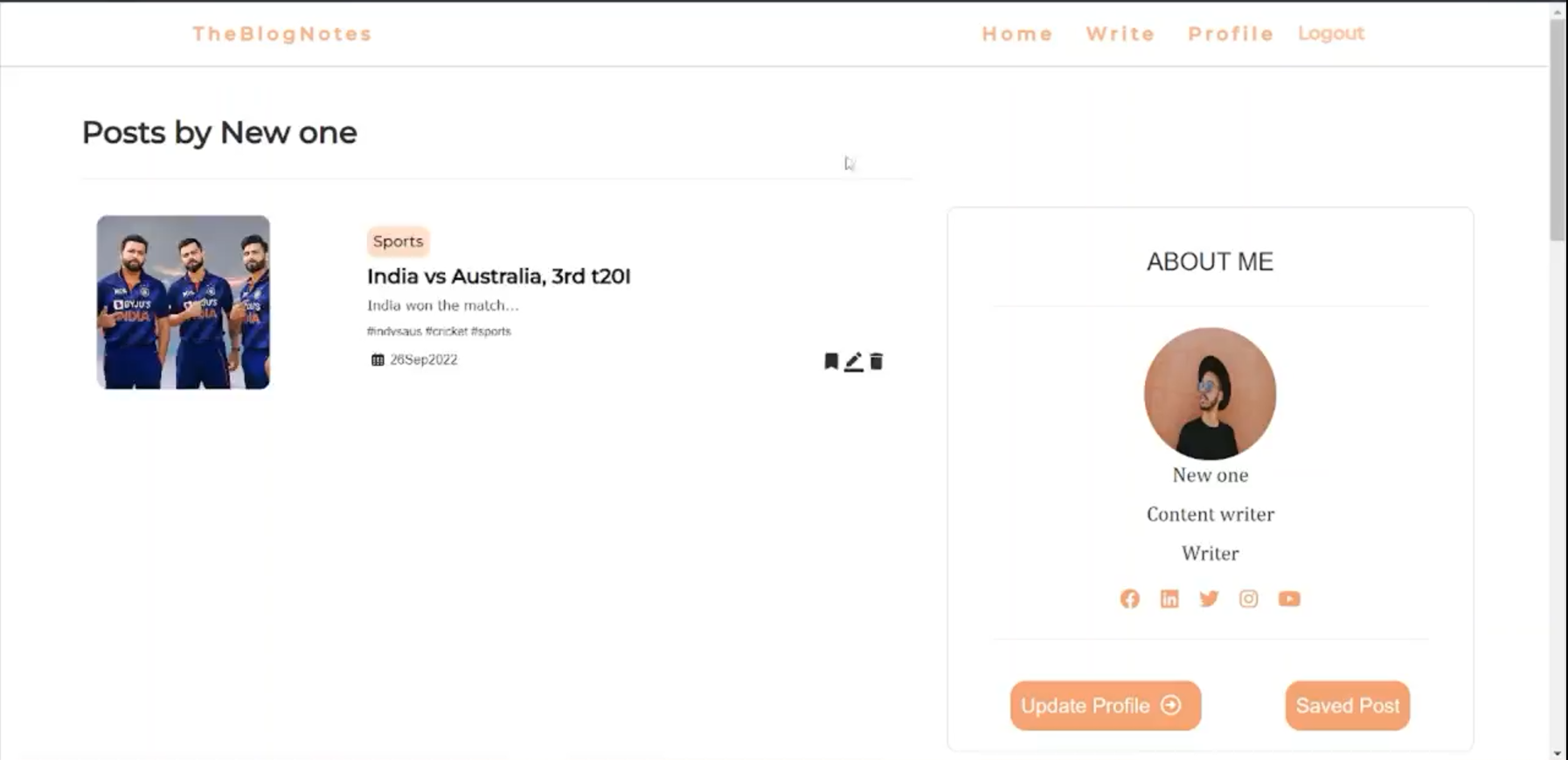


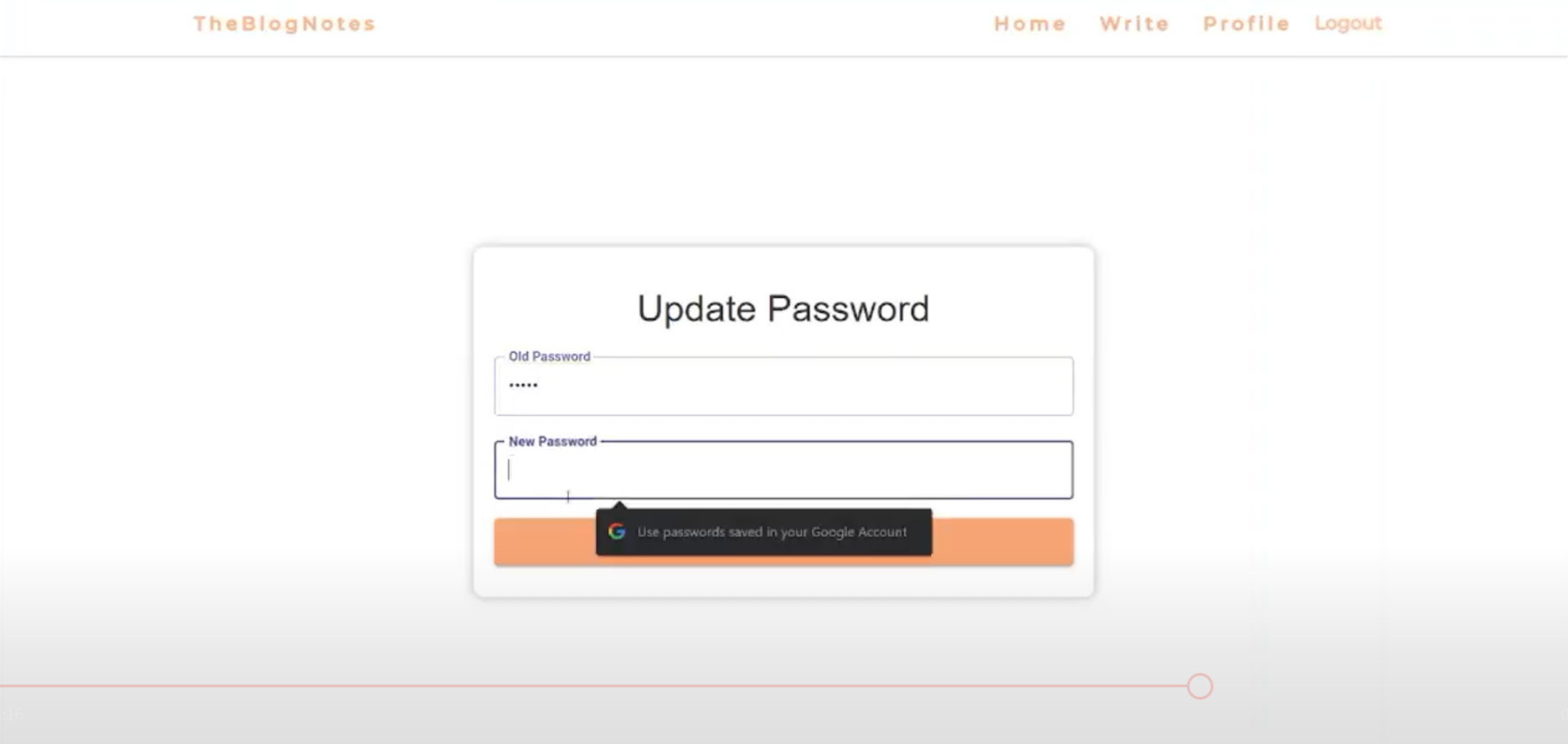


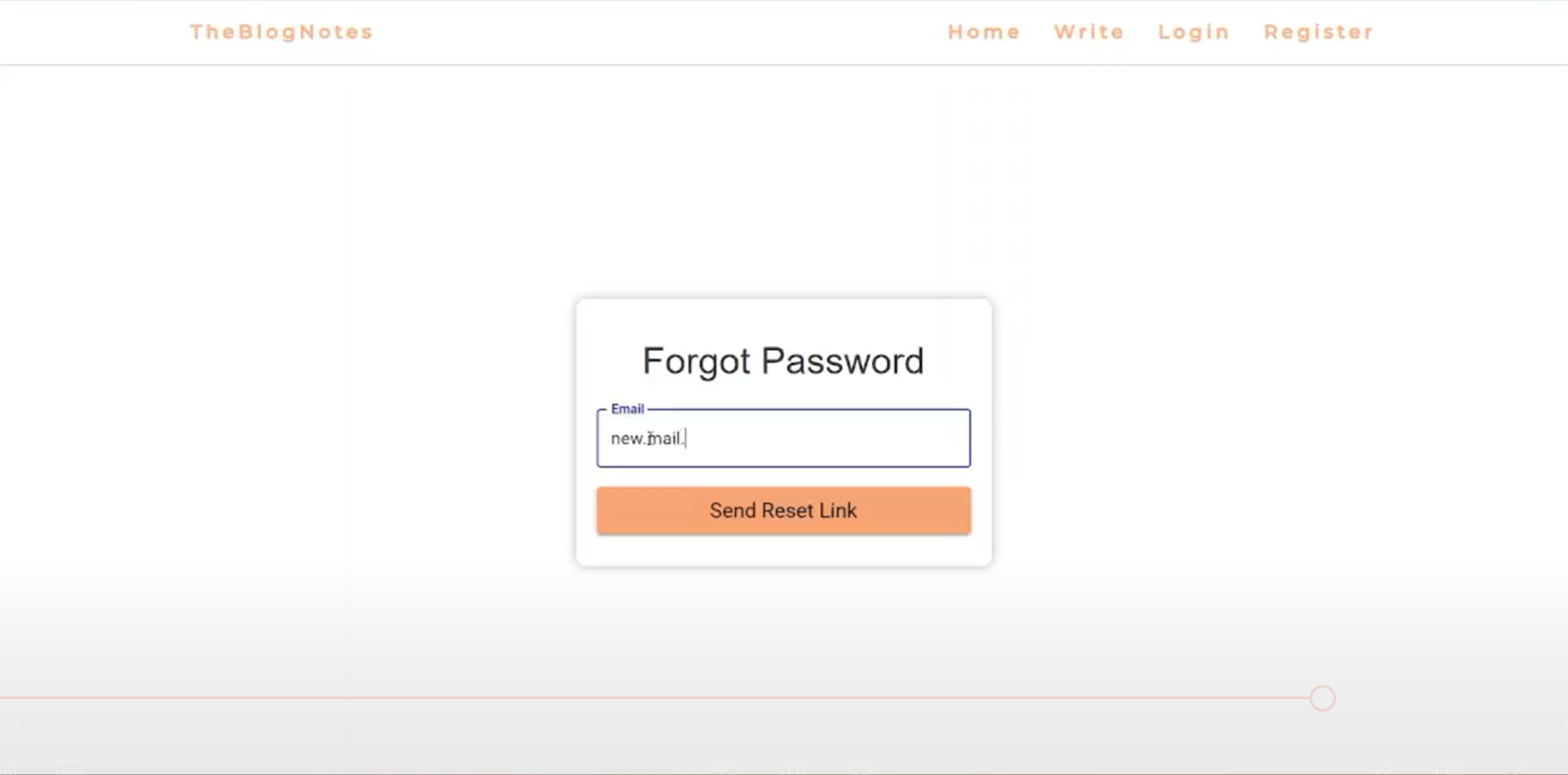
****

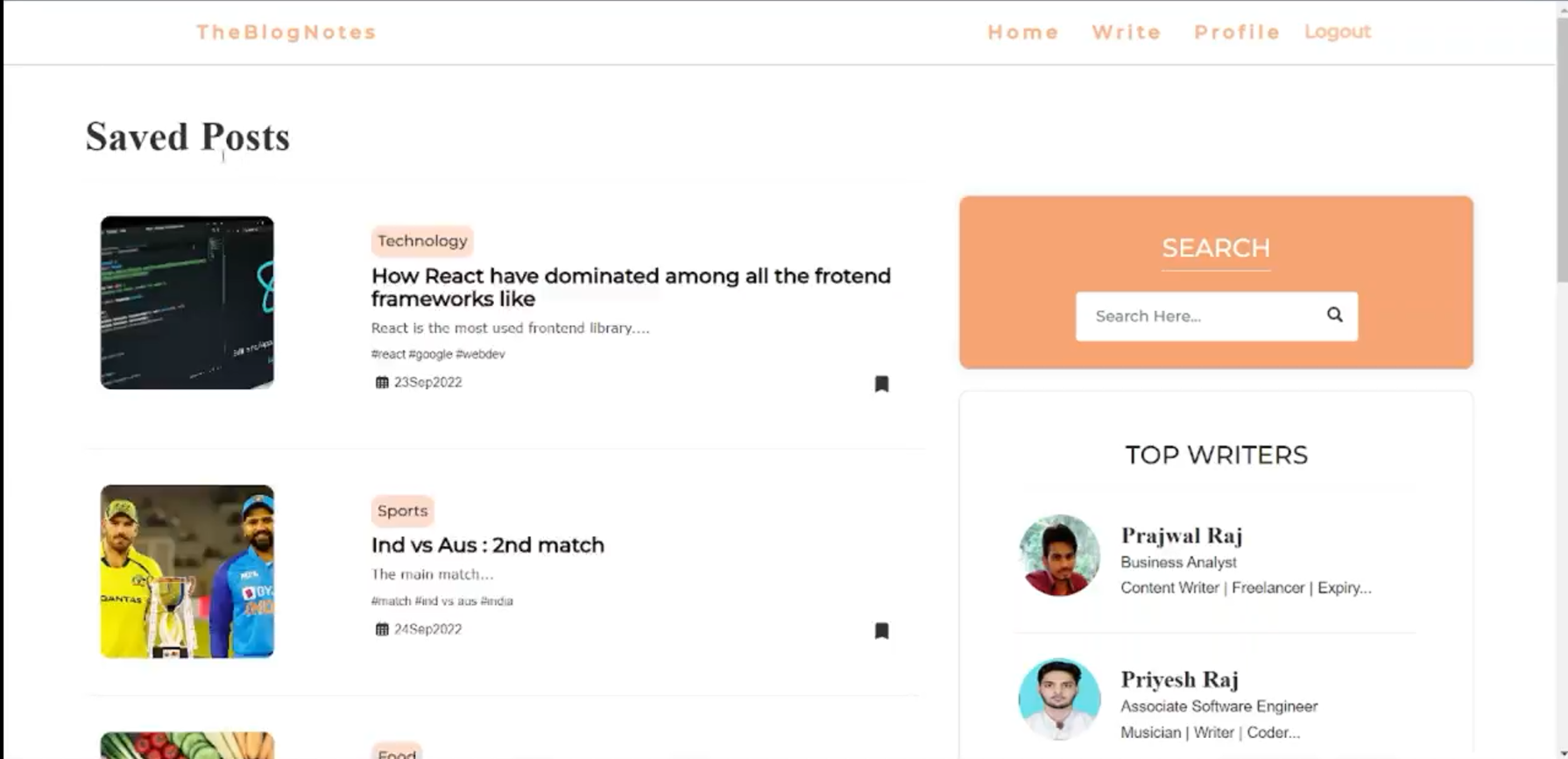
****

****

****

****

****

****

**CONCLUSION**

Thus, we have created a Blog Application with end to end modules such as login, sign up, new post addition deletion updation, search etc. we have implemented my project using the React App and node js. I have divided each segment into each various files and I linked it with javascript and css. If you click on the options, it will redirect you to a new section on that website, where you can browse and add or remove or edit your blogs. I have also improved more security features like if the hacker fetches the data from the database, the hacker will get only the Unicode encrypted data. Also, all your activities are can be noted and can be synced to the google account or to the cloud database. I have also created a database which can be connected to the mongo db for the registration and retrieval of users data.