FULL STACK BLOGGING WEB APPLICATION

A Mini project report submitted
in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

S.SAI SRI VINAY REDDY	-	**19BQ1A05J7**
T.CHANDRIKA	-	**19BQ1A05M9**
S.KAARTHIKEYA	-	**19BQ1A05L5**
Sk. SIDDIK	-	**19BQ1A05L0**
T.SATYA AKHIL	-	**19BQ1A05P8**

Under the Esteemed Guidance of

Dr. G. Sanjay Gandhi

M.Tech, Ph.D.

Professor, Dept of CSE

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY,
Nambur, Guntur

May 2022

BONAFIDE CERTIFICATE

Certified that this mini project report **"FULL STACK BLOGGING WEB APPLICATION"** is the bonafide work of **SANGAM SAI SRI VINAY REDDY, THOTA CHANDRIKA, SOMAROUTHU KAARTHIKEYA PAVANA KUMAAR, SHAIK SIDDIK, TORLIKONDA SATYA AKHIL** who carried out the project work under my guidance and supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
______________________	______________________
Dr. G. SANJAY GANDHI, Ph.D.	**Dr. V. RAMA CHANDRAN, Ph.D.**
GUIDE	**HEAD OF THE DEPARTMENT**

Submitted for Semester Mini-Project viva voce Examination held on ______________

______________________	______________________
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those people who made this project report work easier with the words of encouragement, motivation, discipline and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this seminar report.

First and foremost, I express my deep gratitude to **Sri Vasireddy Vidya Sagar,** Chairman, Vasireddy Venkatadri institute of technology for providing necessary facilities through out the program

I express my sincere gratitude to **Dr. Y. Mallikarjuna Reddy,** principal, Vasireddy Venkatadri institute of technology for his constant support and cooperation throughout the B.Tech program

I express my sincere gratitude to **Dr. V. Rama Chandran,** Professor, HOD, Computer science Engineering**,** Vasireddy Venkatadri institute of technology for his constant support and cooperation throughout the B.Tech program

I express my sincere gratitude to **Dr. G. Sanjay Gandhi,Ph.D**, Vasireddy Venkatadri institute of technology for his constant support and cooperation throughout the B.Tech program

I would like to take this opportunity to express my thanks to **teaching and non teaching** staff in Department of Computer Science & Engineering, VVIT for their invaluable help and support.

S. SAI SRI VINAY REDDY – 19BQ1A05J7

T.CHANDRIKA – 19BQ1A05M9

S. KAARTHIKEYA – 19BQ1A05L5

Sk. SIDDIK – 19BQ1A05L0

T. SATYA AKHIL – 19BQ1A05P8

FULL STACK BLOGGING APPLICATION

ABSTRACT

Blogging refers to writing, photography and other media that is self-published in online. It has started an opportunity for individuals to write diary-style entries. It enables you to reach the billions of people that use the internet. Looking into the technical elements, there are some popular blogging websites like Blogger, Tumblr etc. They are using older tech stack, and the UI looks boring. Every application has functional and non-functional requirements. Some non-functional requirements are design, performance, responsiveness and speed of the application.With the help of modern (MERN) stack, we can improve the UI design by using various frameworks (like MaterialUI) which contains reusable components, that makes the development easier and make the application to look attractive and responsive. With modern back-end technologies the process of fetching/ storing data became simple, efficient and reliable, hence the performance and speed is improved. The code redundancy is reduced, makes the web application simple and light in weight.The special features (functional requirements) of this web application is improved design patterns(like MVC), using like/ dislike mechanism, so that any user posting irrelevant data will be downgraded by others and his posts will not appear on the top, this is not present in most of the present websites.

TABLE OF CONTENTS

Ch.no	TITLE	PAGE NO
	ABSTRACT	i
	Table of contents	ii
1.	Introduction	1-4
	1.1 General	1
	1.2 Types of blogs	1
	1.3 Scenario of existing web applications	4
2.	Concepts & Methods	5-6
	2.1 Problem Description	5
	2.2 Proposed Solution	5
	2.3 System Requirements	6
3.	Implementation	7-34
	3.1 Tools used	7
	3.2 Softwares used	8
	3.3 Languages used	8-9
	3.4 Coding environment setup	9-11
	3.5 Pseudo code/ Algorithms	12-30
	3.6 Screenshots	31-34
4.	Summary (or) Conclusion	35-36
	4.1 Conclusion	35
	4.2 Future enhancemennts	35
	BIBILOGRAPHY	36

CHAPTER 1

INTRODUCTION

1.1 GENERAL

What is a blog?

A blog is a discussion or informational website published on the World Wide Web consisting of informal diary-style text entries (posts). Posts are typically displayed in reverse chronological order, so that the most recent post appears first, at the top of the web page.

The purpose of a blog is to provide content on your website that answers your prospective customers' questions and helps them learn about your product or service. It expands your brand's visibility by giving Google and other search engines content to index and serve up in search results.

1.2 TYPES OF BLOGS

There are 10 different types of blogs:

- Personal blogs
- Business/corporate blogs
- Personal brand/professional blogs
- Fashion blogs
- Lifestyle blogs
- Travel blogs
- Food blogs
- Affiliate/review blogs
- Multimedia blogs
- News blogs

Personal blog:

When you feel like writing, you just get to it without caring about reaching big audiences or selling something. The chances are, you do have a personal blog at the time of reading this.

Business/ corporate blog:

Some companies use blogs to make announcements about product launches, the projects they're working on, upcoming releases, contests, etc. Blogs help businesses increase their site traffic and, hence, improve their conversion rates through content promotion.

Personal brand/ professional blog:

These types of blogs are a combination of a business blog and a personal blog. This blog is usually the project of a single person that eventually takes the path of a business.

Fashion blogs:

This kind of blog is usually managed by one person who is passionate or an expert on the topic.

Lifestyle blogs:

Like fashion blogs, lifestyle blogs include a larger variety of topics from productivity, to wellness, workouts, nutrition, and other aspects of living a better life. Along with fashion, lifestyle is also popular because people need constant advice on cool, efficient, and practical tips for a high-0quality life.

Travel blogs:

Travel blogs seem to get a lot of attention lately because people do a lot of research before deciding on a trip or vacation. The work of travel bloggers is hard (even though it might seem fun and easy at a first sight) because they put hours into research, finding diverse destinations at the best prices, making itineraries, getting familiar with a country's culture, and sharing all the information with their readers.

Food blogs:

Diversifying our meals, cooking healthy food, and buying the ingredients are part of our daily routines, whether we like it or not. Food bloggers found an opportunity in this continuous need that we have and turned it into a business that proved to be an efficient way to make money with your hobby.

Affiliate/ review blogs:

These types of blogs focus on evaluating products or services on a specific market. The owners of these blogs take various products and review them, with pros and cons, pricing, and overall value. They also make recommendations through their expertise in the field.

Multimedia blogs:

After visual content gained popularity in the readers' circles. Vlogs, podcasts, and visual storytelling are the thing nowadays, with many people migrating to this type of content for entertaining, news, education, and information.

News blogs:

These types of blogs are the ones that keep you updated with what's new in an industry that you follow. They mostly focus on presenting the latest happenings, new releases, plans, and ideas that were or will be implemented in a specific area of interest. As opposed to the other types of blogs, news blogs do not usually share opinion posts or person-oriented content

1.3 Scenario of existing web applications

These are not very serious issues with most of the users, but with improving day to day technological trends, user experience is given more priority over functionality of the application.

Users are giving much importance to the visual experience of the application. They are expecting more about styling. With the existing applications, these aspects seems to be slightly outdated, so the main focus is done on these aspects

POPULAR BLOGGING WEBSITES:

- Medium.com
- Blogger.com
- Tumblr.com
- Wix.com

CHAPTER 2

CONCEPTS AND METHODS

2.1 PROBLEM STATEMENT

To develop a blogging web app that serves the purpose of bloggers/ content creators by giving more priority to non functional requirements of the application such as styling, visual experience, positioning of components and performance improvements with the use of modern web technologies and frameworks.

2.2 PROPOSED SOLUTION

- A sleek and attractive website is designed to display the posts, filter them by categories, added a dedicated slot to featured posts, recent posts, comments etc..
- All the data is maintained securely in the GraphCMS portal and is handled by admins and editors of the blog
- The comments posted by users are verified and published by a team of authorized members to prevent unrelated comments and to avoid spamming.
- Entire application is maintained as a Single Page Application(SPA), which is easy to maintain, easy to navigate between the sections and faster in performance.

Advantages:

- Content is separated from the logic.
- Maintenance is easy and secured.
- There is less chance of data redundancy and inconsistency. Hence, maintaining the information is easy

2.3 SYSTEM REQUIREMENTS

2.3.1 ***Hardware Requirements:***

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

• SYSTEM	:	WINDOWS 10
• HARD DISK	:	12 GB
• MONITOR	:	15” LCD
• INPUT DEVICE	:	KEYBOARD, MOUSE
• RAM	:	8GB

2.3.2 ***Software Requirements:***

The software requirements document is the software specification of the system. It should include both a definition and specification of requirements. It is a set of what the system should do rather than how it should do it. It is useful in estimating cost, planning team activities, performing tasks and tracking the team’s progress throughout the development activity.

• OPERATING SYSTEM	:	Windows/ Mac/ Linux
• PLATFORM	:	IA-32, X86-64
• SOFTWARES USED	:	Node JS, Next JS
• LANGUAGES USED	:	React JS, Next JS, Tailwind CSS, GraphQL
• TOOLS USED	:	VS code, Git and Github, BRAVE browser.

CHAPTER 3

IMPLEMENTATION

3.1 TOOLS USED

Visual Studio Code:

The visual studio code is a powerful development environment that lets you edit, debug, and publish apps. Although it is known as IDE for C, C++, C# etc.. it can be easily used for Javascript development environment.

Some of the best VSCode extensions for javascript developers:

- JavaScript(ES6) code snippets
- ESLint
- Prettier
- REST client
- Live Share
- Live Server

Git:

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance.

Github:

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

Browser:

In conventional programming languages, the souce code is passed through a program called compiler, which translates it into byte code and that the machine understands and code can execute. In contrast, JavaScript has no compilation step, instead an interpreter in the browser reads over the javascript code, interprets each line and executes it.

3.2 SOFTWARES USED

NodeJS SDK:

Node.js is a cross-platform JavaScript runtime environment for servers and applications. It is built on a single-threaded, non-blocking event loop, the Google Chrome V8 JavaScript engine, and a low-level I/O API.

3.3 LANGUAGES USED

ReactJS:

React JS is a JavaScript library used in web development to build interactive elements on websites. React is a JavaScript library that specializes in helping developers build user interfaces, or UIs. In terms of websites and web applications, UIs are the collection of on-screen menus, search bars, buttons, and anything else someone interacts with to USE a website or app.

In 2011, Facebook engineer Jordan Walke created React JS specifically to improve UI development.

In addition to providing reusable React library code (saving development time and cutting down on the chance for coding errors), React comes with two key features that add to its appeal for JavaScript developers:

- JSX : JSX stands for JavaScript XML. JSX allows us to write HTML in React. JSX makes it easier to write and add HTML in React.
- Virtual DOM : The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM.

NextJS:

Next. js is a React framework that gives you building blocks to create web applications. By framework, we mean Next. js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application.

Tailwind CSS:

Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override.

Graph QL:

GraphQL is designed to make APIs fast, flexible, and developer-friendly. It can even be deployed within an integrated development environment (IDE) known as GraphiQL. As an alternative to REST, GraphQL lets developers construct requests that pull data from multiple data sources in a single API call.

3.4 CODING ENVIRONMENT SETUP

Create a NextJS app

To create a Next.js app, open your terminal, cd into the directory you'd like to create the app in, and run the following command:

- npx create-next-app nextjs-blog --use-npm --example https://github.com/vercel/next-learn/tree/master/basics/learn-starter

You now have a new directory called nextjs-blog. Let's cd into it:

- cd nextjs-blog
- npm run dev

Now we will see the app is up and running on localhost:3000 by default and it can be opened in the browser

Install all the dependencies

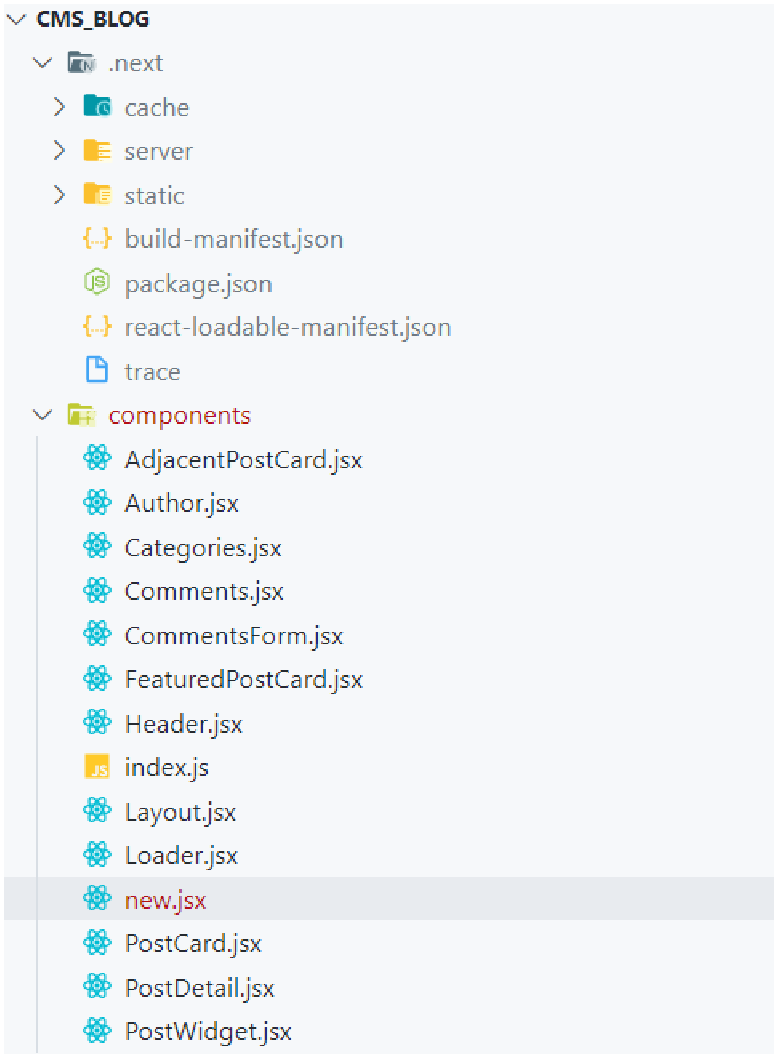
- npm install graphql graphql-request html-react-parser moment react-multi-carousel sass

moment: for date and time

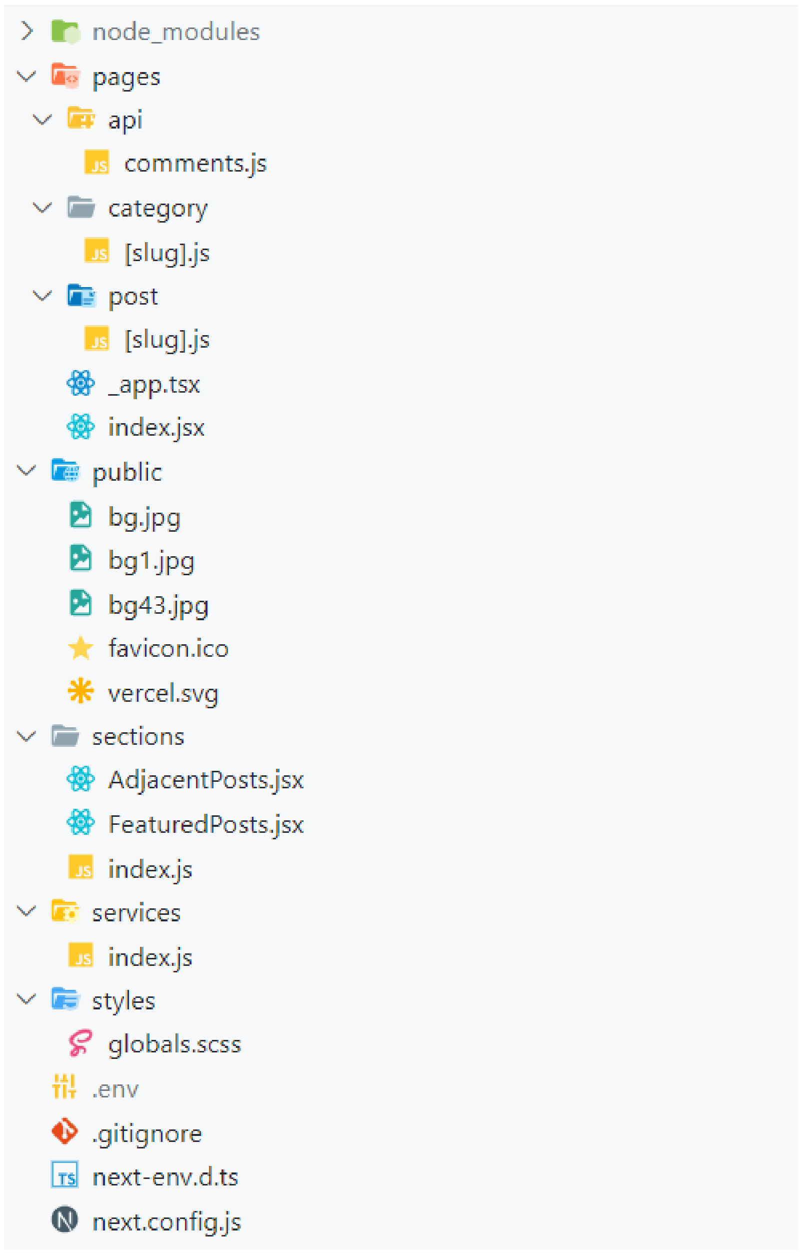
run the application:

- npm run dev

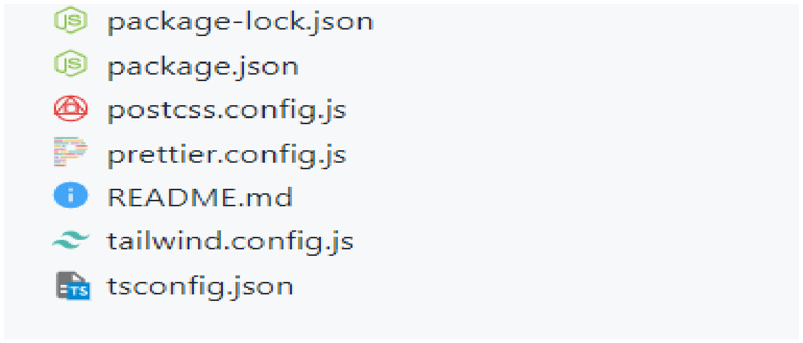
***Folder structure*:**



```
CMS_BLOG
  .next
    cache
    server
    static
    build-manifest.json
    package.json
    react-loadable-manifest.json
    trace
  components
    AdjacentPostCard.jsx
    Author.jsx
    Categories.jsx
    Comments.jsx
    CommentsForm.jsx
    FeaturedPostCard.jsx
    Header.jsx
    index.js
    Layout.jsx
    Loader.jsx
    new.jsx
    PostCard.jsx
    PostDetail.jsx
    PostWidget.jsx
```



```
> node_modules
v pages
  v api
      comments.js
  v category
      [slug].js
  v post
      [slug].js
    _app.tsx
    index.jsx
v public
    bg.jpg
    bg1.jpg
    bg43.jpg
    favicon.ico
    vercel.svg
v sections
    AdjacentPosts.jsx
    FeaturedPosts.jsx
    index.js
v services
    index.js
v styles
    globals.scss
  .env
  .gitignore
  next-env.d.ts
  next.config.js
```



```
  package-lock.json
  package.json
  postcss.config.js
  prettier.config.js
  README.md
  tailwind.config.js
  tsconfig.json
```

3.5 PSEUDO CODES, FUNCTIONS AND STYLING

Tailwind styles used: tailwind styling is used as internal styles

• mx	: horizontal margin
• px	: padding horizontal
• mb	: margin bottom
• container	: sets max width of an element
• grid	: By default, Tailwind includes grid-template-column utilities for creating basic grids with up to 12 equal width columns
• lg:	: specifies how the styles should appear on the large screen
• bg	: background of an element
• border – b	: indicates black border(short hand notation of colours)
• w-full	: short hand operator for width of a component
• inline-block	: used to control the flow of text and elements.
• float-left/right	: The float class defines the flow of content for controlling the wrapping of content around an element.
• cursor-pointer	: In this class, the cursor is a pointer and indicates a link
• hidden	: to set an element to display: none and remove it from the page layout
• overflow	: utilities for controlling how an element handles content that is too large for the container.
• Transition	: Utilities for controlling which CSS properties transition.
• Duration	: Utilities for controlling the duration of CSS transitions.
• Hover	: Style elements on hover, focus, and active using the hover, focus, and active modifiers
• Flex	: Utilities for controlling how flex items both grow and shrink.
• Align-items	: Utilities for controlling how flex and grid items are positioned along a container's cross axis.
• Justify	: Utilities for controlling how flex and grid items are positioned along a container's main axis.
• Rounded	: Utilities for controlling the border radius of an element.

Keywords and Tags in nextjs/ reactjs:

- className : class is a keyword in javascript and JSX is an extension of javascript. That's the principal reason why React uses className instead of class
    - eg:

```
import "./App.css";
function App() {
return <h1 className="heading1">This is an example code</h1>;
}
export default App;
```

- Link : Client-side transitions between routes can be enabled via the Link component exported by next/link.
    - Eg:

```
import Link from 'next/link';
function Home() {
  return (
    <ul>
      <li>
        <Link
          href={{
            pathname: '/about',
            query: { name: 'test' },
          }}
        >
          <a>About us</a>
        </Link>
      </li>
      <li>
        <Link
          href={{
            pathname: '/blog/[slug]',
            query: { slug: 'my-post' },
          }}
        >
          <a>Blog Post</a>
        </Link>
      </li>
    </ul>
  )
}
export default Home
```

- moment : MomentJS is a JavaScript library which helps is parsing, validating, manipulating and displaying date/time in JavaScript in a very easy way.
    - eg: .format('MM DD YYYY')
- html-react parser : The parser converts an HTML string to one or more React elements.

Key functions used:

- map() : map() creates a new array from calling a function for every array element. map() calls a function once for each element in an array. map() does not execute the function for empty elements. map() does not change the original array.

```
Eg: const numbers = [65, 44, 12, 4];
const newArr = numbers.map(myFunction)
function myFunction(num) {
  return num * 10;
}
```

- {props} : It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.
- Children props : children are a special prop that is used to pass the data from the parent component to the children component but this data must be enclosed within the parent's opening and closing tag.
- useEffect() : useEffect(callback, dependencies) is the hook that manages the side-effects in functional components. callback argument is a function to put the side-effect logic. dependencies is a list of dependencies of your side-effect: being props or state values.
- useState() : The React useState Hook allows us to track state in a function component. State generally refers to data or properties that need to be tracking in an application.
- useRef() : The useRef Hook allows you to persist values between renders. It can be used to store a mutable value that does not cause a re-render when updated. It can be used to access a DOM element directly.

- Async - await : The word “async” before a function means one simple thing: a function always returns a promise. Other values are wrapped in a resolved promise automatically.

  The keyword await makes JavaScript wait until that promise settles and returns its result.

```
Eg: async function f() {
        let promise = new Promise((resolve, reject) => {
        setTimeout(() => resolve("done!"), 1000)
        });
        let result = await promise; // wait until the promise resolves (*)
        alert(result); // "done!"
        }
        f();
```

- then : The then() method in JavaScript has been defined in the Promise API and is used to deal with asynchronous tasks such as an API call. Previously, callback functions were used instead of this function which made the code difficult to maintain.
- getStaticProps() : You should use getStaticProps if: The data required to render the page is available at build time ahead of a user's request. The data comes from a headless CMS. The page must be pre-rendered (for SEO) and be very fast — getStaticProps generates HTML and JSON files, both of which can be cached by a CDN for performance.

Writing a basic react function component:

<u>New.jsx</u>

```
import React from ‘react’
const New = () => {
  return (
    <div>new</div>
  )
}
export default New;
```

Exploring the styles folder:

- **globals.scss:** it is the global style sheet for entire application, all the components inherit the styles form this file

Exploring the COMPONENTS folder:

- **PostCard.jsx :** contains the card view of post, contains slug, excerpt, author name and description, button.

Code:

```
const PostCard = ({post}) => {
  //console.log(post);
  return (
    <div className='bg-white shadow-lg rounded-lg p-0 lg:p-8 pb-12 mb-8'>
      <div className='relative overflow-hidden shadow-md pb-80 mb-6'>
             <img src='http://cdn2.hubspot.net/hubfs/145335/blogging-for-business-heres-everything-you-need-to-know.jpg'
        alt={post.title}
   className='object-top absolute h-80 w-full object-cover shadow-lg rounded-t-lg lg:rounded-lg' />
  </div>
 <h1 className="transition duration-100 text-center mb-8 cursor-pointer hover:text-pink-600 text-3xl font-semibold">
      <Link href={`/post/${post.slug}`}>{post.title}</Link>
    </h1>
    <div className='bloc lg:flex text-center items-center justify-center mb-8 w-full'>
      <div className='flex items-center justify-center mb-4 lg:mb-0 w-full lg:w-auto mr-8'>
                <img src='https://www.myenglishteacher.eu/blog/wp-content/uploads/2021/05/other-word-for-writer-or-author.jpeg' alt={post.author.name} height="80px" width="80px" className='align-middle rounded-full' />
           <p className='inline align-middle text-gray-700 ml-2 text-lg'>{post.author.name}</p>
      </div>
      <div className='font-medium text-gray-700'>
        <span>{moment(post.createdAt).format('MMM DD YYYY')}</span>
      </div>

    </div>
```

```
      <p className='text-center text-lg text-gray-700 font-normal px-4 lg:px-20 mb-8'>{post.excerpt}</p>
    <div className='text-center'><Link href={`/post/${post.slug}`}>
        <span className='transition duration-500 transform hover:-translate-y-1 inline-block bg-pink-600 text-lg font-medium rounded-full text-white px-8 py-3 cursor-pointer'>
        Continue Reading
      </span>
      </Link></div>
      </div>
  )
}
```

- **Categories.jsx :** displays all the categories in a card form

  Code:

```
const Categories = () => {
  const [categories, setCategories] = useState([]);
  useEffect(() => {
    getCategories().then((newCategories) => {
      setCategories(newCategories);
    });
  }, []);
  return (
   <div className="bg-white shadow-lg rounded-lg p-8 pb-12 mb-8">
    <h3 className="text-xl mb-8 font-semibold border-b pb-4">Categories</h3>
    {categories.map((category, index) => (
        <Link key={index} href={`/category/${category.slug}`}>
           <span className={`cursor-pointer block ${(index === categories.length - 1) ? 'border-b-0' : 'border-b'} pb-3 mb-3`}>{category.name}</span>
        </Link>
      ))}
    </div>
  )
}
```

- **Components > index.js :** acts as default start point for the entire components folder, all the other components of the folder are exported to this file.
  - Eg: export { default as PostCard } from './PostCard';

- **Header.jsx :** contains the navbar component

  Code:

```
const Header = () => {
    const [categories, setCategories] = useState([]);
      useEffect(() => {
        getCategories().then((newCategories) => {
          setCategories(newCategories);
        });
      }, []);
    return (
        <div className="container mx-auto px-10 mb-8">
        <div className="border-b w-full inline-block border-blue-400 py-8">
            <div className='md:float-left block'>
                <Link href="/">
                    <span className='cursor-pointer font-bold text-4xl text-white'>
                        BLOGIt
                    </span>
                </Link>
                <div className='hidden md:float-left md:contents'>
                    {categories.map((category)=>(
                        <Link key={category.slug} href={`/category/${category.slug}`}>
                              <span className='md:float-right mt-2 align-middle text-white ml-4
font-semibold cursor-pointer'>
                                {category.name}
                            </span>
                        </Link>

                    ))}
                </div>
            </div>
        </div>
    </div>
  )
}
```

- **Layout.jsx:** render the header component and the children props

  Code:

```
const Layout = ({children}) => {
  return (
    <>
    <Header/>
    {children}
    </>
  )
}
export default Layout
```

- **PostWidget.jsx :** displays the layout for adjacent posts and recent posts

  Code:

```
const PostWidget = ({ categories, slug }) => {
  const [relatedPosts, setRelatedPosts] = useState([]);
  useEffect(() => {
    if (slug) {
      getSimilarPosts(categories, slug).then((result) => {
        setRelatedPosts(result);
      });
    }
    else {
      getRecentPosts().then((result) => {
        setRelatedPosts(result);
      });
    }
  }, [slug]);
  return (
    <div className="bg-white shadow-lg rounded-lg p-8 pb-12 mb-8">
     <h3 className="text-xl mb-8 font-semibold border-b pb-4">{slug ? 'Related Posts'
: 'Recent Posts'}</h3>
    {relatedPosts.map((post, index) => (
        <div key={index} className="flex items-center w-full mb-4">
          <div className="w-16 flex-none">
            <img

              alt={post.title}
              height="60px"
              width="60px"

              className="align-middle rounded-full"
              src={post.featuredimage.url}
            />
          </div>
     <div className="flex-grow ml-4">
                                        <p         className="text-gray-500         font-
xs">{moment(post.createdAt).format('MMM DD, YYYY')}</p>
                              <Link     href={`/post/${post.slug}`}     className="text-md"
key={index}>{post.title}</Link>
     </div>
    </div>
   ))}
  </div>
 )
}

export default PostWidget
```

- **PostDetail.jsx :** detailed view of the post

  Code:

```
<div className='bg-white shadow-lg rounded-lg lg:p-8 pb-12 mb-8'>
    <div className='relative overflow-hidden shadow-md mb-6'>
      <img src={post.featuredimage.url} alt={post.title} className='object-top h-
full w-full rounded-t-lg'/>

    </div>
    <div className='px-4 lg:px-0'>
      <div className='flex items-center mb-8 w-full'>
      <div className='bloc lg:flex text-center items-center  mb-8 w-full'>
   <div className='flex items-center justify-center mb-4 lg:mb-0 w-full lg:w-auto
mr-8'>
    <img src='https://www.myenglishteacher.eu/blog/wp-
content/uploads/2021/05/other-word-for-writer-or-author.jpeg' alt={post.author.name}
height="80px" width="80px" className='align-middle rounded-full' />
   <p className='inline align-middle text-gray-700 ml-2 text-
lg'>{post.author.name}</p>
   </div>
   <div className='font-medium text-gray-700'>
    <span>{moment(post.createdAt).format('MMM DD YYYY')}</span>
   </div>
      </div>

</div>
<h1 className="mb-8 text-3xl font-semibold">{post.title}</h1>
```

```
{post.content.raw.children.map((typeObj, index) => {
            const children = typeObj.children.map((item, itemindex) =>
getContentFragment(itemindex, item.text, item));
            return getContentFragment(index, children, typeObj, typeObj.type);
          })}
        </div>
        </div>
  )
}

export default PostDetail
```

- **Author.jsx :** display author info inside detailed post component

  Code:

```
const Author = ({author}) => {
  //console.log(author);
  return (
    <div className='text-center mt-20 mb-8 p-12 relative rounded-lg bg-black bg-
opacity-20'>
      <div className='absolute left-0 right-0 -top-14'>
      <Image
      unoptimized
      alt={author.name}
      height="100px"
      width="100px"
      className='alin-middle rounded-full'
      src='https://www.storey-lines.com/wp-content/uploads/2013/04/Writer-V-Author-
Whats-The-Difference.jpg'
      />
      </div>
  <h3 className='text-white my-4 text-xl font-bold'>{author.name}</h3>
  <p className='text-white text-lg'>{author.bio}</p>
    </div>
  )
}

export default Author
```

- **CommentsForm.jsx :** contains the form elements to collect commentator details like name, email etc.

  Code:

```
<div className="bg-white shadow-lg rounded-lg p-8 pb-12 mb-8">
      <h3 className="text-xl mb-8 font-semibold border-b pb-4">Leave a Reply</h3>
      <div className="grid grid-cols-1 gap-4 mb-4">
        <textarea value={formData.comment} onChange={onInputChange}
className="p-4 outline-none w-full rounded-lg h-40 focus:ring-2 focus:ring-gray-200
bg-gray-100 text-gray-700" name="comment" placeholder="Comment" />
      </div>
      <div className="grid grid-cols-1 lg:grid-cols-2 gap-4 mb-4">
        <input type="text" value={formData.name} onChange={onInputChange}
className="py-2 px-4 outline-none w-full rounded-lg focus:ring-2 focus:ring-gray-
200 bg-gray-100 text-gray-700" placeholder="Name" name="name" />
        <input type="email" value={formData.email} onChange={onInputChange}
className="py-2 px-4 outline-none w-full rounded-lg focus:ring-2 focus:ring-gray-
200 bg-gray-100 text-gray-700" placeholder="Email" name="email" />
      </div>
      <div className="grid grid-cols-1 gap-4 mb-4">
        <div>
          <input checked={formData.storeData} onChange={onInputChange}
type="checkbox" id="storeData" name="storeData" value="true" />
          <label className="text-gray-500 cursor-pointer" htmlFor="storeData"> Save
my name, email in this browser for the next time I comment.</label>
        </div>
      </div>
      {error && <p className="text-xs text-red-500">All fields are mandatory</p>}
      <div className="mt-8">
        <button type="button" onClick={handlePostSubmission} className="transition
duration-500 ease hover:bg-indigo-900 inline-block bg-pink-600 text-lg font-medium
rounded-full text-white px-8 py-3 cursor-pointer">Post Comment</button>
        {showSuccessMessage && <span className="text-xl float-right font-semibold
mt-3 text-green-500">Comment submitted for review</span>}
      </div>
    </div>
  );
};

export default CommentsForm;
```

- **Comments.jsx :** displays the actual comments done by the users, but they were published once they were approved by the owners

  Code:

```
const Comments = ({ slug }) => {
  const [comments, setComments] = useState([]);

  useEffect(() => {
    getComments(slug).then((result) => {
      setComments(result);
    });
  }, []);

  return (
    <>
      {comments.length > 0 && (
        <div className="bg-white shadow-lg rounded-lg p-8 pb-12 mb-8">
          <h3 className="text-xl mb-8 font-semibold border-b pb-4">
            {comments.length}
            {' '}
            Comments
          </h3>
            {comments.map((comment, index) => (
              <div key={index} className="border-b border-gray-100 mb-4 pb-4">
                <p className="mb-4">
                  <span className="font-semibold">{comment.name}</span>
                  {' '}
                  on
                  {' '}
                  {moment(comment.createdAt).format('MMM DD, YYYY')}
                </p>
                <p className="whitespace-pre-line text-gray-600 w-
full">{parse(comment.comment)}</p>
              </div>
            ))}
        </div>
      )}
    </>
  );
};
```

```
export default Comments;
```

- **FeaturedPosts.jsx :** displays the featured blogs at the top of home screen

  Code:

```
const FeaturedPostCard = ({ post }) => (
  <div className="relative h-72">
    <div className="absolute rounded-lg bg-center bg-no-repeat bg-cover shadow-md
inline-block w-full h-72" style={{ backgroundImage: `url('${post.featuredimage.url}')`
}} />
    <div className="absolute rounded-lg bg-center bg-gradient-to-b opacity-50 from-
gray-400 via-gray-700 to-black w-full h-72" />
    <div className="flex flex-col rounded-lg p-4 items-center justify-center absolute
w-full h-full">
      <p className="text-white mb-4 text-shadow font-semibold text-
xs">{moment(post.createdAt).format('MMM DD, YYYY')}</p>
      <p className="text-white mb-4 text-shadow font-semibold text-2xl text-
center">{post.title}</p>
      <div className="flex items-center absolute bottom-5 w-full justify-center">
        <Image
          unoptimized
          alt={post.author.name}
          height="30px"
          width="30px"
          className="align-middle drop-shadow-lg rounded-full"
          src='https://www.storey-lines.com/wp-content/uploads/2013/04/Writer-V-
Author-Whats-The-Difference.jpg'
        />
        <p className="inline align-middle text-white text-shadow ml-2 font-
medium">{post.author.name}</p>
      </div>
    </div>
    <Link href={`/post/${post.slug}`}><span className="cursor-pointer absolute w-
full h-full" /></Link>
  </div>
);

export default FeaturedPostCard;
```

- **AdjacentPostCard.jsx :** displays the relevant posts of each category

  Code:

```
const AdjacentPostCard = ({ post, position }) => (
  <>
    <div className="absolute rounded-lg bg-center bg-no-repeat bg-cover shadow-md
inline-block w-full h-72" style={{ backgroundImage: `url('${post.featuredImage.url}')`
}} />
    <div className="absolute rounded-lg bg-center bg-gradient-to-b opacity-50 from-
gray-400 via-gray-700 to-black w-full h-72" />
    <div className="flex flex-col rounded-lg p-4 items-center justify-center absolute
w-full h-full">
      <p className="text-white text-shadow font-semibold text-
xs">{moment(post.createdAt).format('MMM DD, YYYY')}</p>
      <p className="text-white text-shadow font-semibold text-2xl text-
center">{post.title}</p>
    </div>
    <Link href={`/post/${post.slug}`}><span className="z-10 cursor-pointer absolute
w-full h-full" /></Link>
    {position === 'LEFT' && (
      <div className="absolute arrow-btn bottom-5 text-center py-3 cursor-pointer bg-
pink-600 left-4 rounded-full">
        <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-white w-
full" fill="none" viewBox="0 0 24 24" stroke="currentColor">
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M10
19l-7-7m0 0l7-7m-7 7h18" />
        </svg>
      </div>
    )}
    {position === 'RIGHT' && (
      <div className="absolute arrow-btn bottom-5 text-center py-3 cursor-pointer bg-
pink-600 right-4 rounded-full">
        <svg xmlns="http://www.w3.org/2000/svg" className="h-6 w-6 text-white w-
full" fill="none" viewBox="0 0 24 24" stroke="currentColor">
          <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M14
5l7 7m0 0l-7 7m7-7H3" />
        </svg>
      </div>
    )}
  </>
);
```

export default AdjacentPostCard;

- **Loader.jsx :** renders the "loading…" graphics which is actually an SVG file

Creating the schemas in graph CMS:

- Author{

  Name – single line text, required, title field

  Photo – Asset, two-way reference

  Bio – multi line text

  Post – multiple values, two way reference

  }

- Category{

  Name – single line text, required, unique, title

  Slug – slug, required, unique

  Posts – multiple values, tow way reference

  }

- Comment{

  Name – single line text, required

  Email – single line text, required

  Comment – multi line text, required

  Post – two way reference

  }

- Post{

  Title - single line text, required, title

  Slug – slug, required, unique

  Excerpt – multi line text, required

  Content – Richtext, embed, required

  Featured image - Asset, two-way reference, required

Featured post – Boolean, required

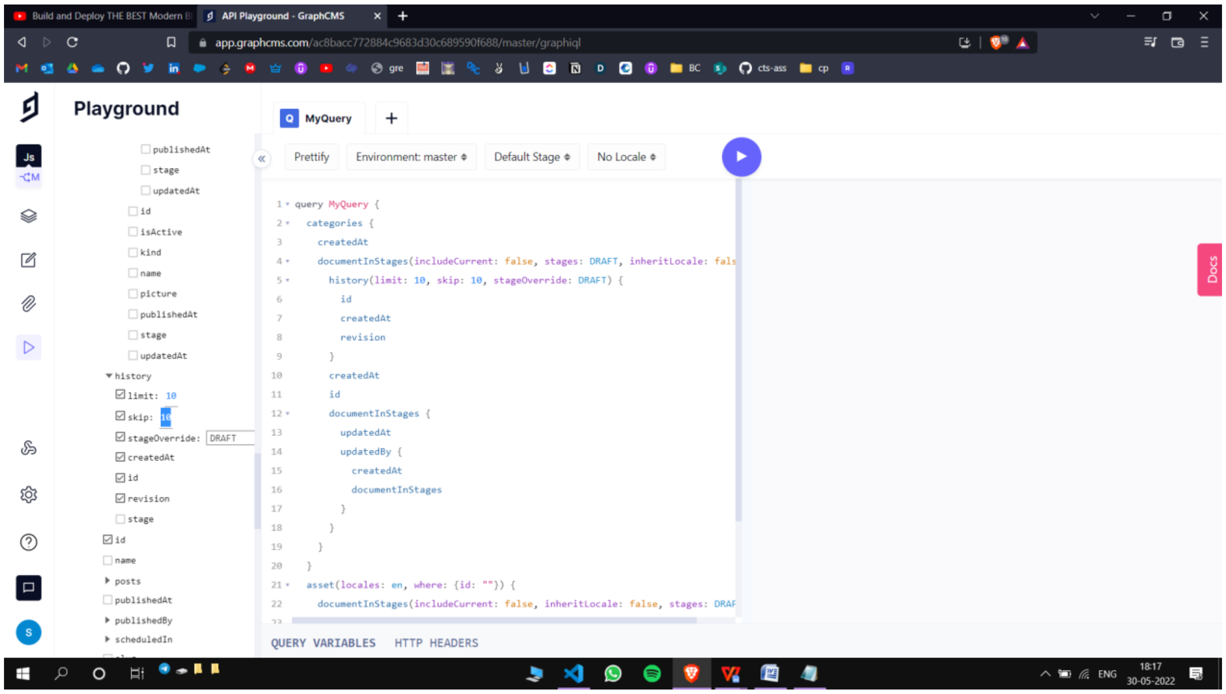
Categories – multiple values, two way reference

Comments - multiple values, two way reference

Author - two way reference

}

Generating the API calls from the playground:



```
query MyQuery {
  categories {
    createdAt
    documentInStages(includeCurrent: false, stages: DRAFT, inheritLocale: fals
      history(limit: 10, skip: 10, stageOverride: DRAFT) {
        id
        createdAt
        revision
      }
      createdAt
      id
      documentInStages {
        updatedAt
        updatedBy {
          createdAt
          documentInStages
        }
      }
    }
  }
  asset(locales: en, where: {id: ""}) {
    documentInStages(includeCurrent: false, inheritLocale: false, stages: DRAF
```

3.5.1

Example of a graphql query:

gql`

query MyQuery {

  categories {

    createdAt

    documentInStages(includeCurrent: false, stages: DRAFT, inheritLocale: false) {

      history(limit: 10, skip: 10, stageOverride: DRAFT) {

        id

```
      createdAt
      revision
     }
     createdAt
     id
     documentInStages {
      updatedAt
      updatedBy {
       createdAt
       documentInStages
      }
     }
    }
  }
  asset(locales: en, where: {id: ""}) {
   documentInStages(includeCurrent: false, inheritLocale: false, stages: DRAFT) {
    id
   }
  }
}`
```

Exploring the SERVICES folder:

- **index.js:** in this file we will write the graph ql queries to fetch the data from graph cms
  import {request, gql} from ‘graphql-request’;

pages>post>[slug].js:

replaces the routing task of react

no need to create routes

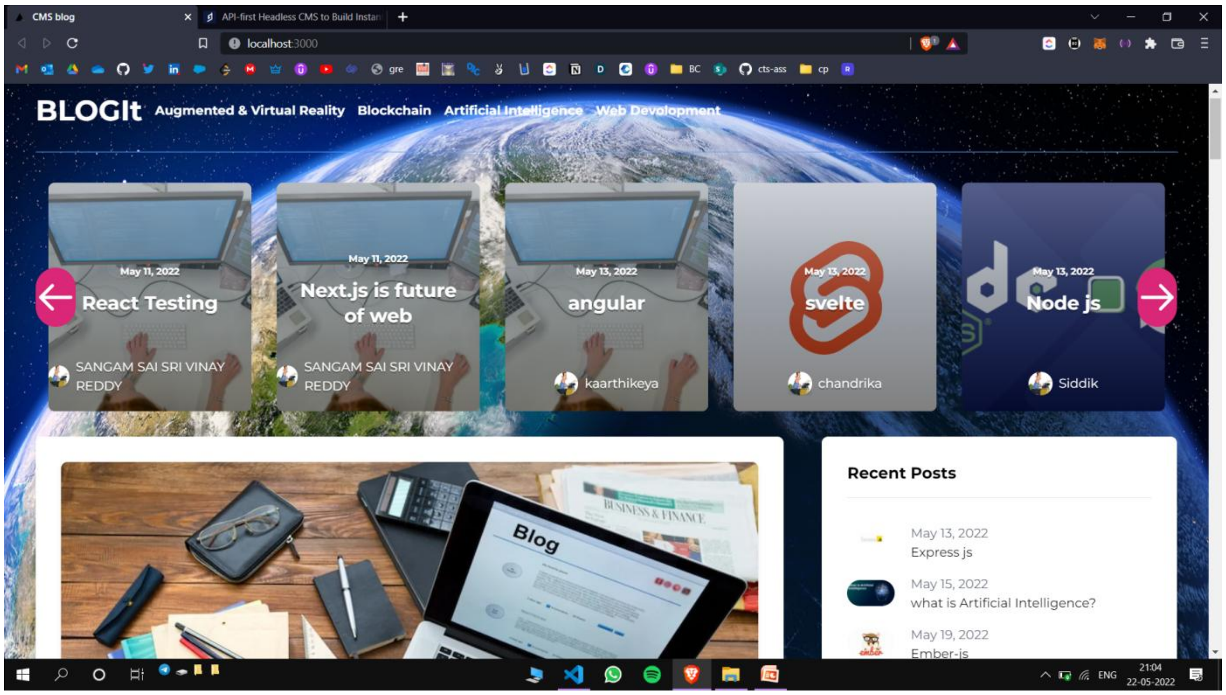
directly creates endpoints with the help of arrow functions

Code:

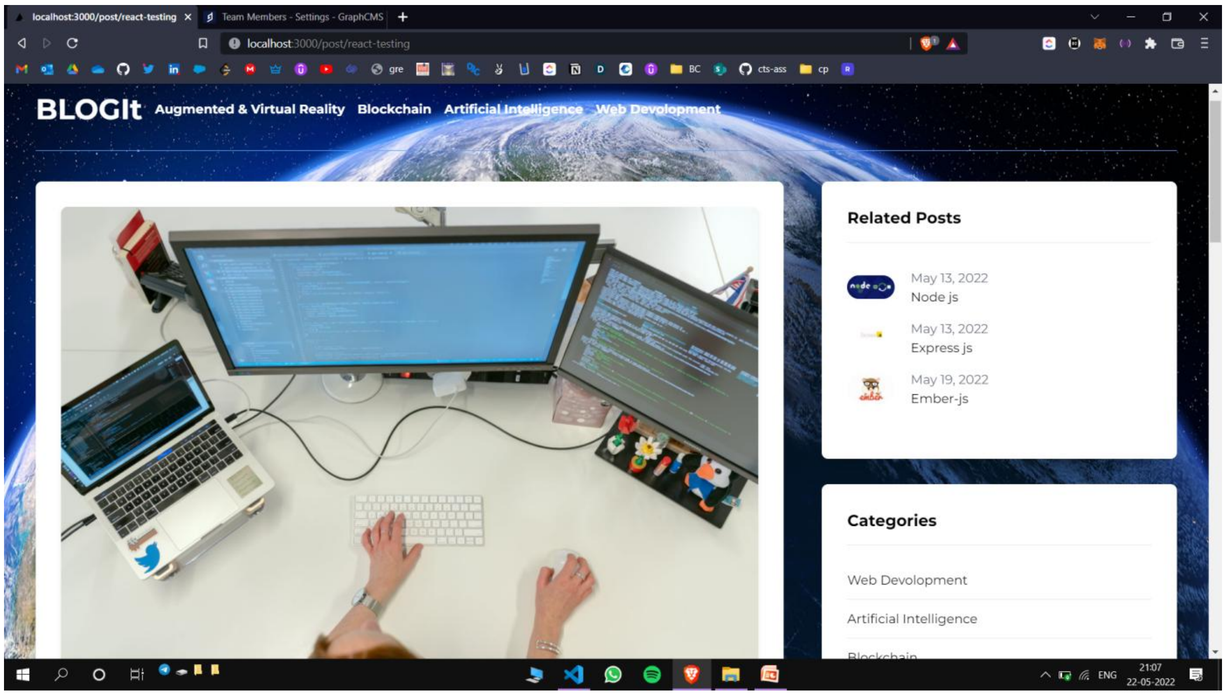
```
const PostDetails = ({post}) => {
  const router = useRouter();

  if (router.isFallback) {
    return <Loader />;
  }
    //console.log(post);
  return (
    <div className="container mx-auto px-10 mb-8">
        <div className="grid grid-cols-1 lg:grid-cols-12 gap-12">
            <div className="col-span-1 lg:col-span-8">
                <PostDetail post={post}/>
                <Author author={post.author}/>
                <CommentsForm slug={post.slug}/>
                <Comments slug={post.slug}/>
            </div>
            <div className="col-span-1 lg:col-span-4">
                <div className='relative lg:sticky top-8'>
                    <PostWidget slug={post.slug}
categories={post.categories.map((category)=>category.slug)} />
                    <Categories />
                </div>
            </div>
        </div>
        </div>
  )
}

export default PostDetails

export async function getStaticProps({params}) {
    const data= await getPostDetails(params.slug);
    return {
      props: {
        post:data
      }
    }
  }
```

```
export async function getStaticPaths() {
  const posts = await getPosts();
  return {
    paths: posts.map(({ node: { slug } }) => ({ params: { slug } })),
    fallback: true,
  };
}
```

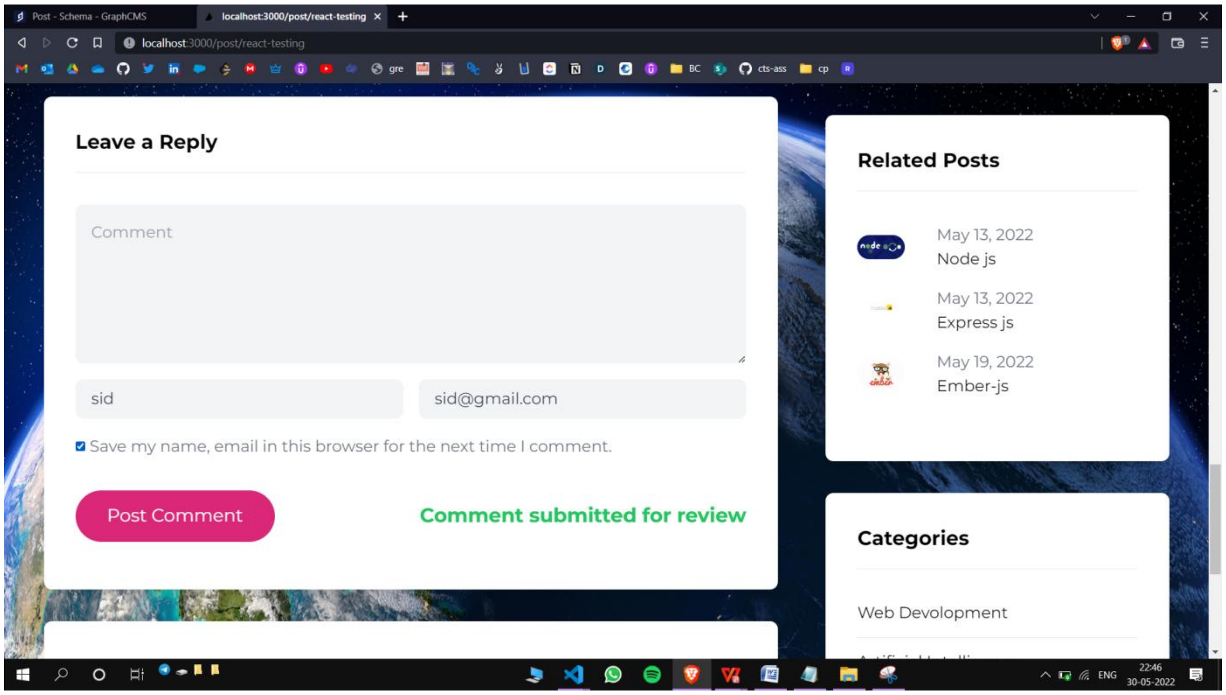
3.6 SCREENSHOTS



3.6.1



3.6.2



Leave a Reply

Comment

sid

sid@gmail.com

Save my name, email in this browser for the next time I comment.

Post Comment

Comment submitted for review

Related Posts

May 13, 2022
Node js

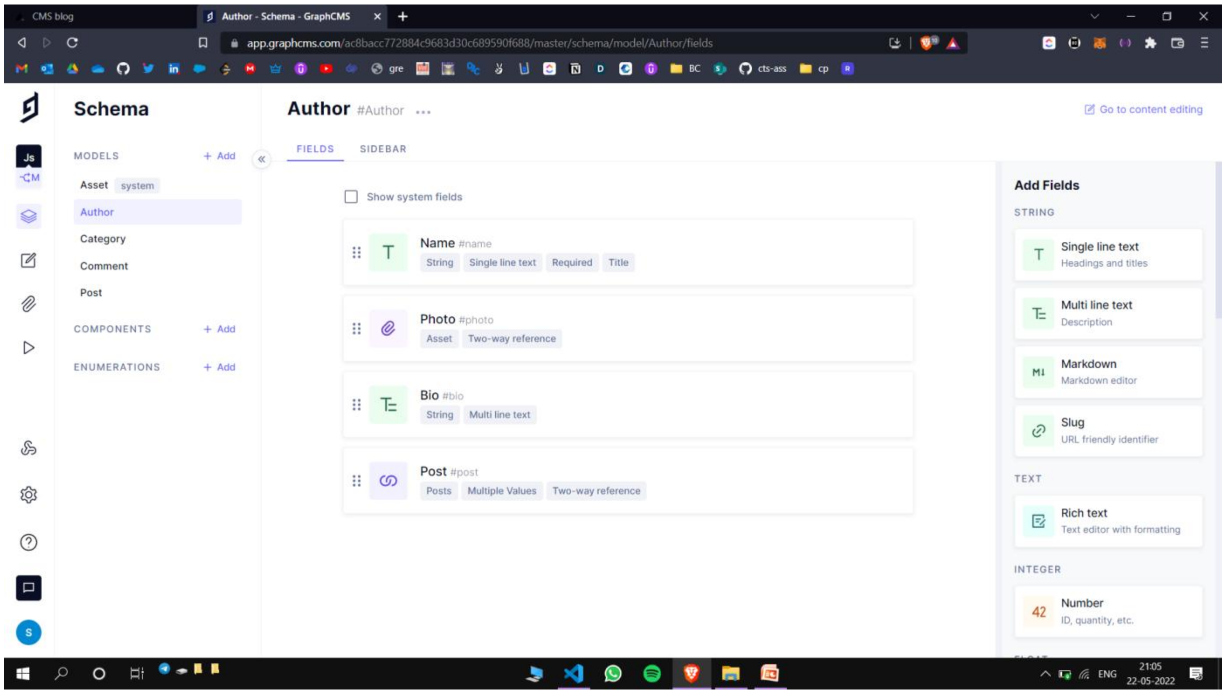
May 13, 2022
Express js

May 19, 2022
Ember-js

Categories

Web Devolopment

3.6.3



Schema

MODELS + Add

Asset system
Author
Category
Comment
Post

COMPONENTS + Add

ENUMERATIONS + Add

Author #Author

FIELDS SIDEBAR

Show system fields

Name #name — String, Single line text, Required, Title

Photo #photo — Asset, Two-way reference

Bio #bio — String, Multi line text

Post #post — Posts, Multiple Values, Two-way reference

Add Fields

STRING

Single line text — Headings and titles

Multi line text — Description

Markdown — Markdown editor

Slug — URL friendly identifier

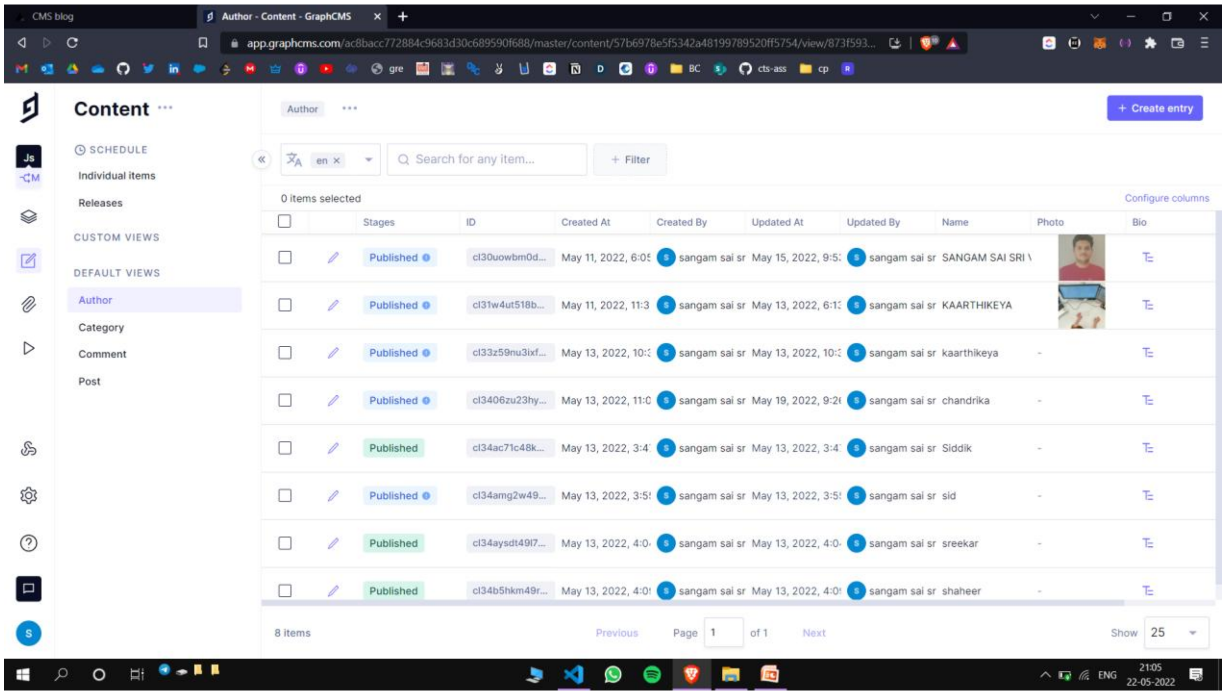
TEXT

Rich text — Text editor with formatting

INTEGER

Number — ID, quantity, etc.

3.6.4



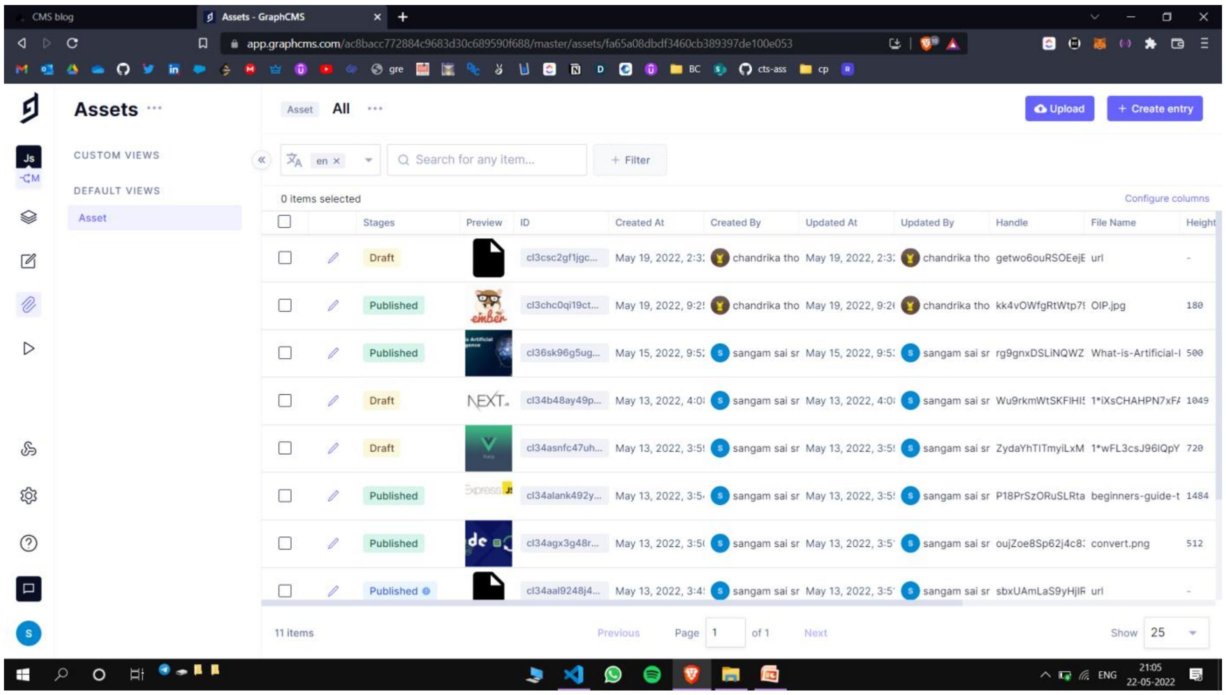
Content

0 items selected

Stages	ID	Created At	Created By	Updated At	Updated By	Name	Photo	Bio
Published	cl30uowbm0d...	May 11, 2022, 6:0	sangam sai sr	May 15, 2022, 9:5	sangam sai sr	SANGAM SAI SRI \		
Published	cl31w4ut518b...	May 11, 2022, 11:3	sangam sai sr	May 13, 2022, 6:1	sangam sai sr	KAARTHIKEYA		
Published	cl33z59nu3ixf...	May 13, 2022, 10:	sangam sai sr	May 13, 2022, 10:	sangam sai sr	kaarthikeya	-	
Published	cl3406zu23hy...	May 13, 2022, 11:0	sangam sai sr	May 19, 2022, 9:2	sangam sai sr	chandrika	-	
Published	cl34ac71c48k...	May 13, 2022, 3:4	sangam sai sr	May 13, 2022, 3:4	sangam sai sr	Siddik	-	
Published	cl34amg2w49...	May 13, 2022, 3:5	sangam sai sr	May 13, 2022, 3:5	sangam sai sr	sid	-	
Published	cl34aysdt49l7...	May 13, 2022, 4:0	sangam sai sr	May 13, 2022, 4:0	sangam sai sr	sreekar	-	
Published	cl34b5hkm49r...	May 13, 2022, 4:0	sangam sai sr	May 13, 2022, 4:0	sangam sai sr	shaheer	-	

8 items Previous Page 1 of 1 Next Show 25

3.6.5



Assets

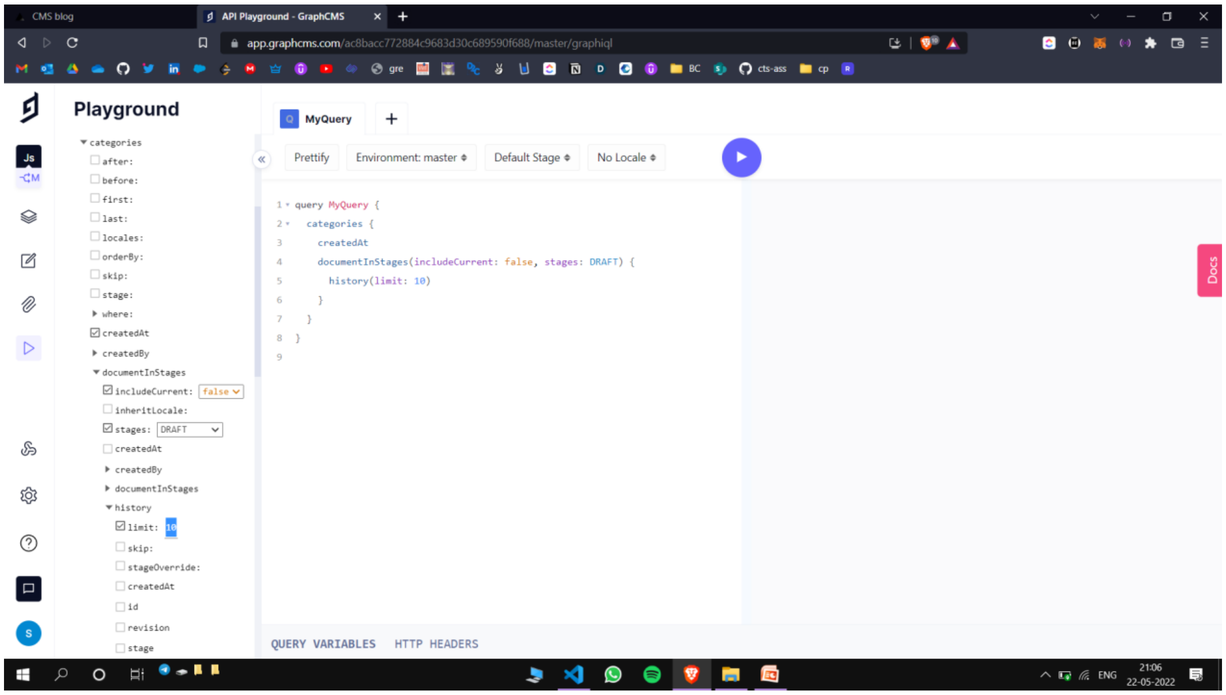
Asset All

0 items selected

Stages	Preview	ID	Created At	Created By	Updated At	Updated By	Handle	File Name	Height
Draft		cl3csc2gf1jgc...	May 19, 2022, 2:3	chandrika tho	May 19, 2022, 2:3	chandrika tho	getwo6ouRSOEejE	url	-
Published		cl3chc0qi19ct...	May 19, 2022, 9:2	chandrika tho	May 19, 2022, 9:2	chandrika tho	kk4vOWfgRtWtp7	OIP.jpg	180
Published		cl36sk96g5ug...	May 15, 2022, 9:5	sangam sai sr	May 15, 2022, 9:5	sangam sai sr	rg9gnxDSLiNQWZ	What-is-Artificial-I	500
Draft		cl34b48ay49p...	May 13, 2022, 4:0	sangam sai sr	May 13, 2022, 4:0	sangam sai sr	Wu9rkmWtSKFIHI	1*iXsCHAHPN7xF	1049
Draft		cl34asnfc47uh...	May 13, 2022, 3:5	sangam sai sr	May 13, 2022, 3:5	sangam sai sr	ZydaYhTITmyiLxM	1*wFL3csJ96lQpY	720
Published		cl34alank492y...	May 13, 2022, 3:5	sangam sai sr	May 13, 2022, 3:5	sangam sai sr	P18PrSzORuSLRta	beginners-guide-t	1484
Published		cl34agx3g48r...	May 13, 2022, 3:5	sangam sai sr	May 13, 2022, 3:5	sangam sai sr	oujZoe8Sp62j4c8	convert.png	512
Published		cl34aal9248j4...	May 13, 2022, 3:4	sangam sai sr	May 13, 2022, 3:5	sangam sai sr	sbxUAmLaS9yHjIR	url	-

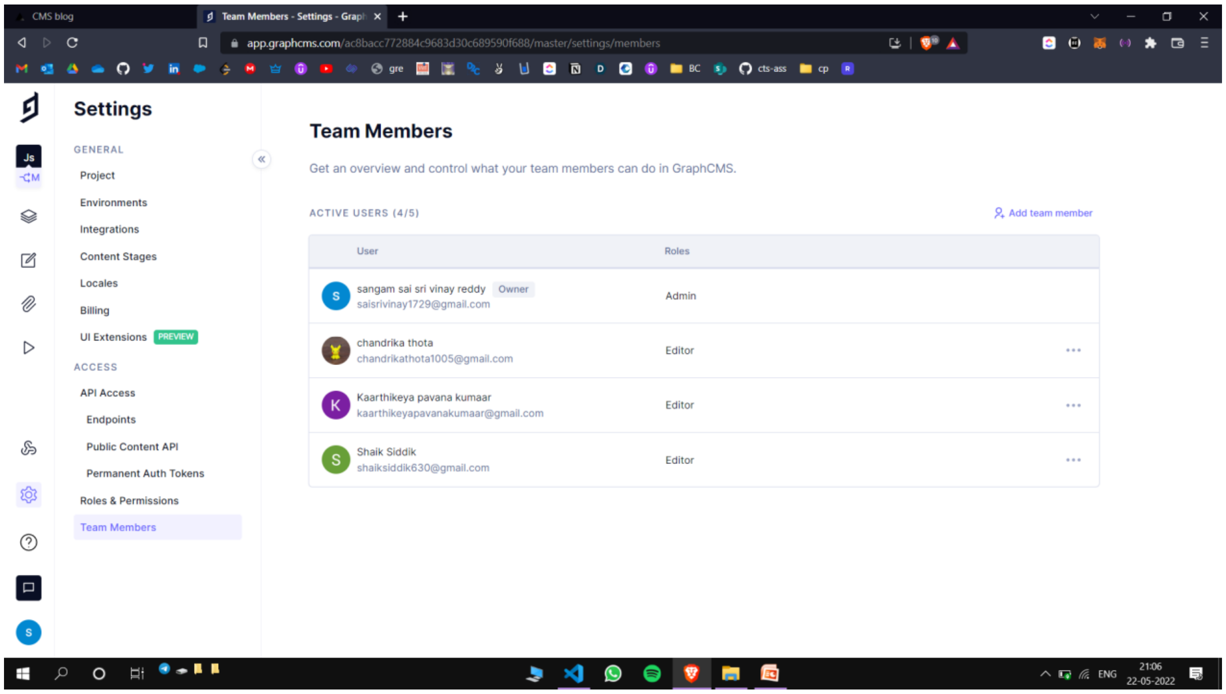
11 items Previous Page 1 of 1 Next Show 25

3.6.6



```
query MyQuery {
  categories {
    createdAt
    documentInStages(includeCurrent: false, stages: DRAFT) {
      history(limit: 10)
    }
  }
}
```

3.6.7



Team Members

Get an overview and control what your team members can do in GraphCMS.

ACTIVE USERS (4/5)

User	Roles
sangam sai sri vinay reddy Owner saisrivinay1729@gmail.com	Admin
chandrika thota chandrikathota1005@gmail.com	Editor
Kaarthikeya pavana kumaar kaarthikeyapavanakumaar@gmail.com	Editor
Shaik Siddik shaiksiddik630@gmail.com	Editor

3.6.8

CHAPTER 4

4.1 CONCLUSION

Hence, by taking the non functional requirements such as visual experience, performance etc.. into consideration, an attractive website is designed for the purpose of blogging.

4.2 FUTURE ENHANCEMENTS

- Improve the UI further by adding new features such as no. of likes, shares etc..
- Adding more categories other than professional ones, like productivity tips, developer life stories, corporate life and work styles etc..

BIBILOGRAPHY

[1] https://reactjs.org/

[2] https://nextjs.org/

[3] https://tailwindcss.com/

[4] https://graphql.org/

[5] https://graphcms.com/

[6] https://code.visualstudio.com/

[7] https://nodejs.org/en/

[8] https://www.youtube.com/c/CleverProgrammer

[9] https://www.youtube.com/c/Freecodecamp.org