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

a. ***moment:*** A JavaScript date library for parsing, validating, manipulating, and formatting dates.
b. ***Graphql:*** The JavaScript reference implementation for GraphQL, a query language for APIs created by Facebook.

c. ***graphql-request:*** Minimal GraphQL client supporting Node and browsers for scripts or simple apps

d. ***html-react-parser:*** HTML to React parser that works on both the server (Node.js) and the client (browser)

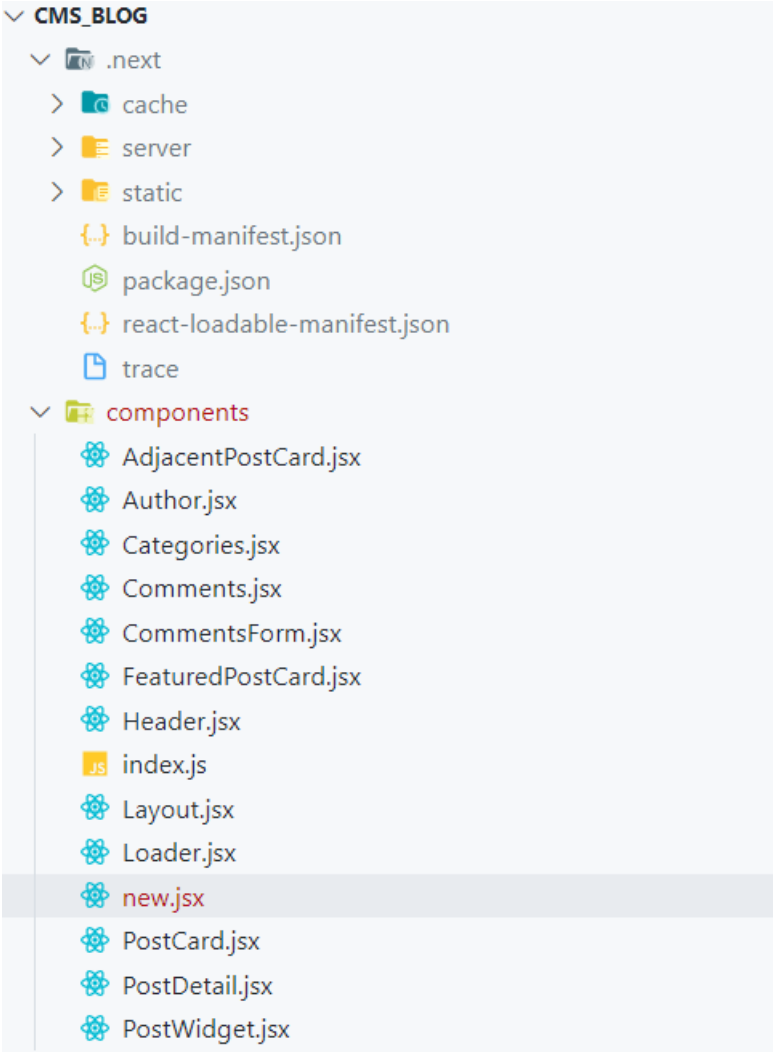
e. ***react-multi-carousel:*** Production-ready, lightweight fully customizable React carousel component that rocks supports multiple items and SSR(Server-side rendering).

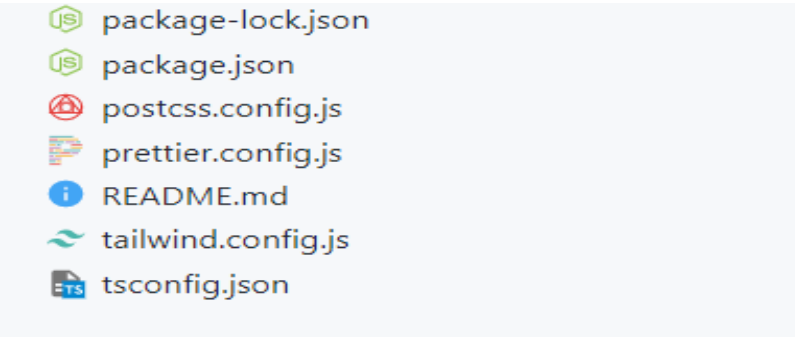
f. ***Sass:*** Sass is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets. SassScript is the scripting language itself.

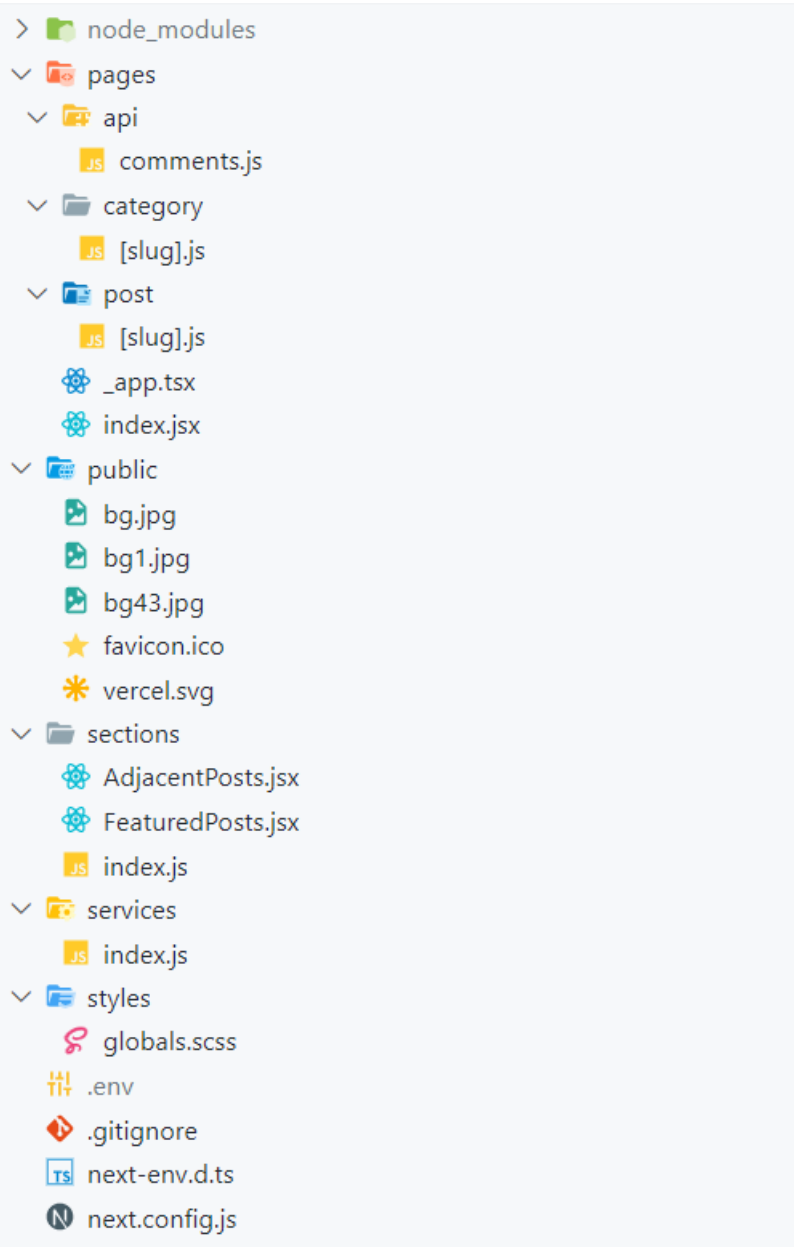
run the application: *navigate into the project directory and run the following command*

$ npm run dev

***Folder structure*:**




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


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

I. ***server:*** this directory contains all the sever side code files

II. ***components:*** this directory contains the code blocks that will render individual components of the user interface

III. ***pages:*** this directory contains the background code of individual pages in the application

   a. ***api:*** this folder contains the code files of api which will fetch the post comments from the graph CMS.

   b. ***category:*** this folder contains the code files which will fetch the information of each category in the blog.

   c. ***post:*** this folder contains the code files which will fetch the information of each post in the blog.

IV. ***Public:*** this folder contains the assets of the project like images, videos etc.

V. ***Sections:*** this folder contains the code files which will configure the layout for individual sections in the application

VI. ***Services:*** this is the service folder for the application, which will fetch the data from the graph CMS using graphQL queries

VII. ***Styles:*** this will contain the stylesheets for entire application

VIII. ***.env:*** A .env file is a simple text configuration file for controlling your applications environment constants. Between local, staging and production environments majority of your application will not change.

IX. ***.gitignore:*** gitignore tells git which files (or patterns) it should ignore. It's usually used to avoid committing transient files from your working directory that aren't useful to other collaborators, such as compilation products, temporary files IDEs create, etc.

X. ***Package.json:*** The package. json file is the heart of any Node project. It records important metadata about a project which is required before publishing to NPM, and also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the entry point to our package.

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:

New.jsx

```
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.
- **Categories.jsx :** displays all the categories in a card form
- **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
- **Layout.jsx:** render the header component and the children props
- **PostWidget.jsx :** displays the layout for adjacent posts and recent posts
- **PostDetail.jsx :** detailed view of the post
- **Author.jsx :** display author info inside detailed post component
- **CommentsForm.jsx :** contains the form elements to collect commentator details like name, email etc.
- **Comments.jsx :** displays the actual comments done by the users, but they were published once they were approved by the owners
- **FeaturedPosts.jsx :** displays the featured blogs at the top of home screen
- **AdjacentPostCard.jsx :** displays the relevant posts of each category
- **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

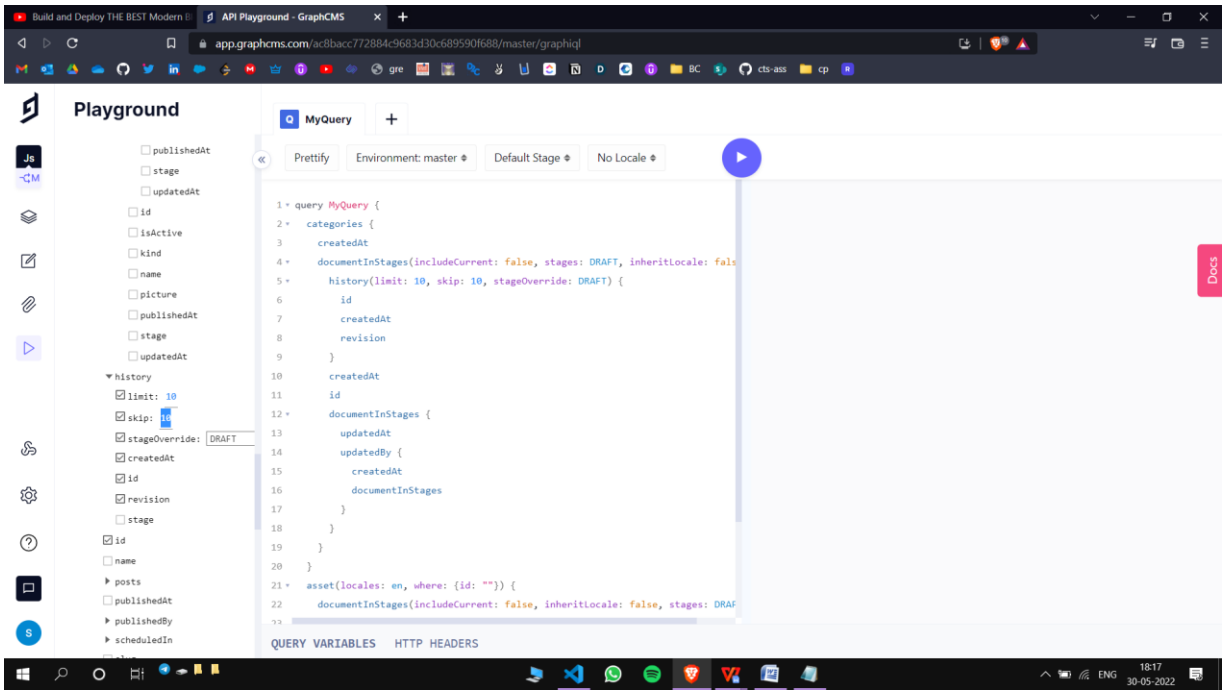
}

- 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:



3.5.1

Example of a graphql query:

gql`

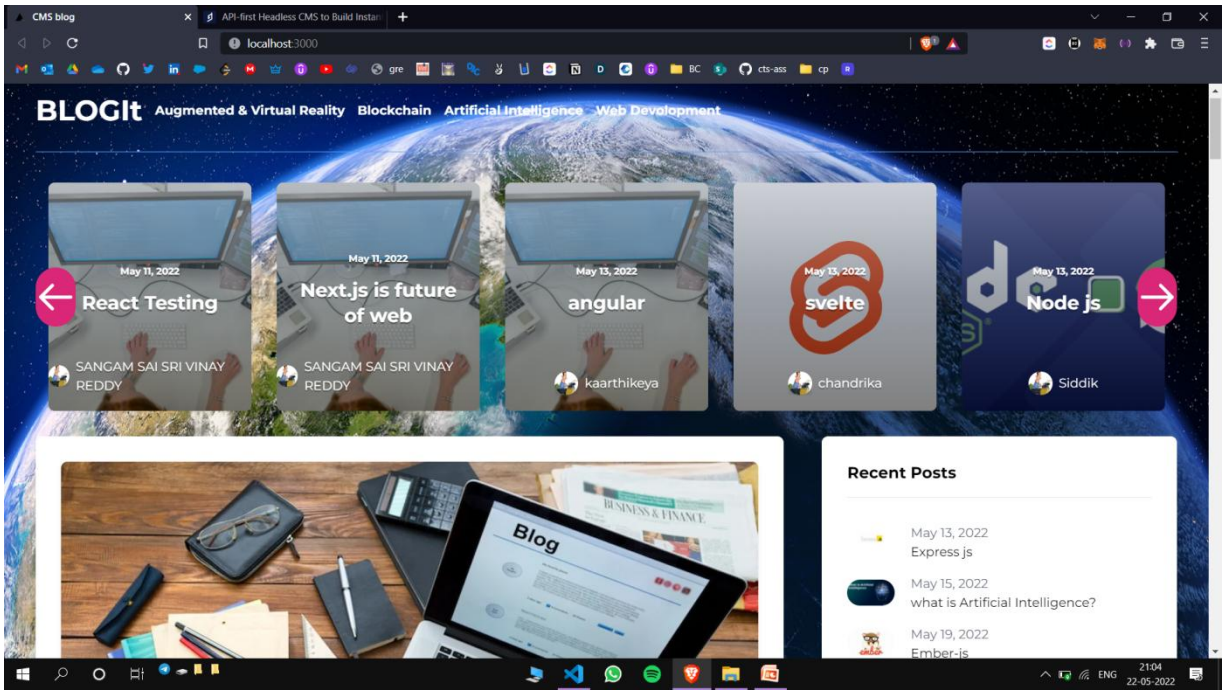
```
query MyQuery {
  categories {
    createdAt
    documentInStages(includeCurrent: false, stages: DRAFT, inheritLocale: false) {
      history(limit: 10, skip: 10, stageOverride: DRAFT) {
        id
        createdAt
        revision
      }
      createdAt
      id
      documentInStages {
        updatedAt
        updatedBy {
```

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

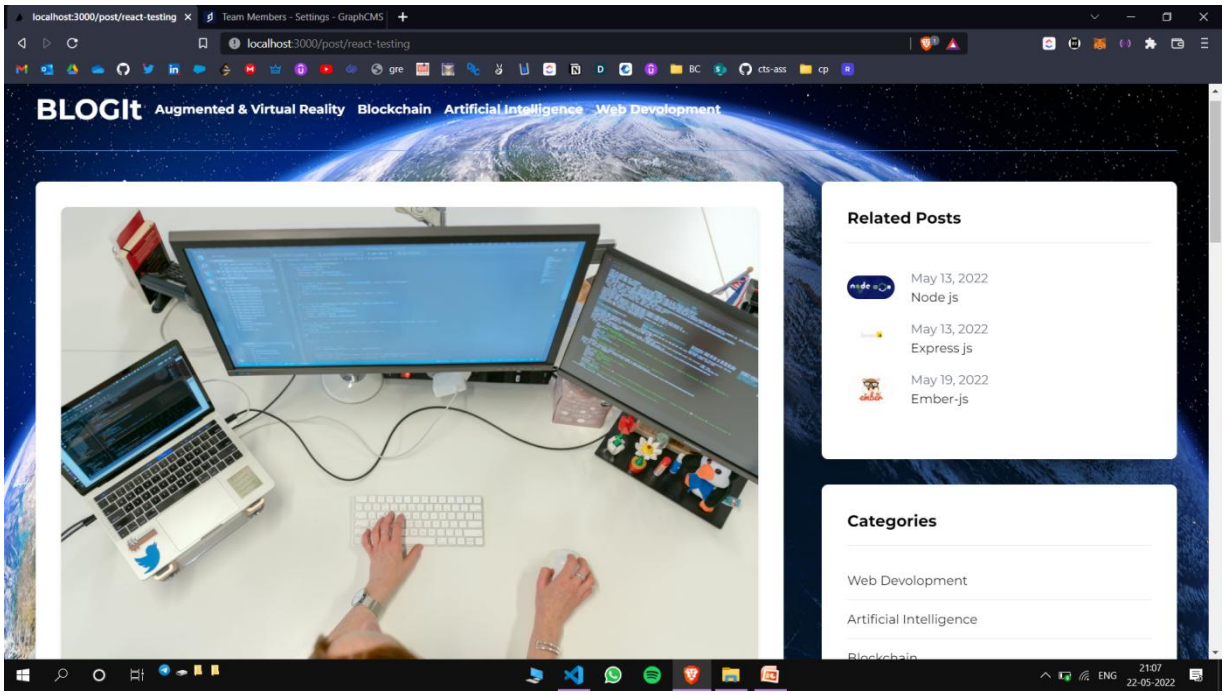
Exploring the SERVICES folder:

- **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’;

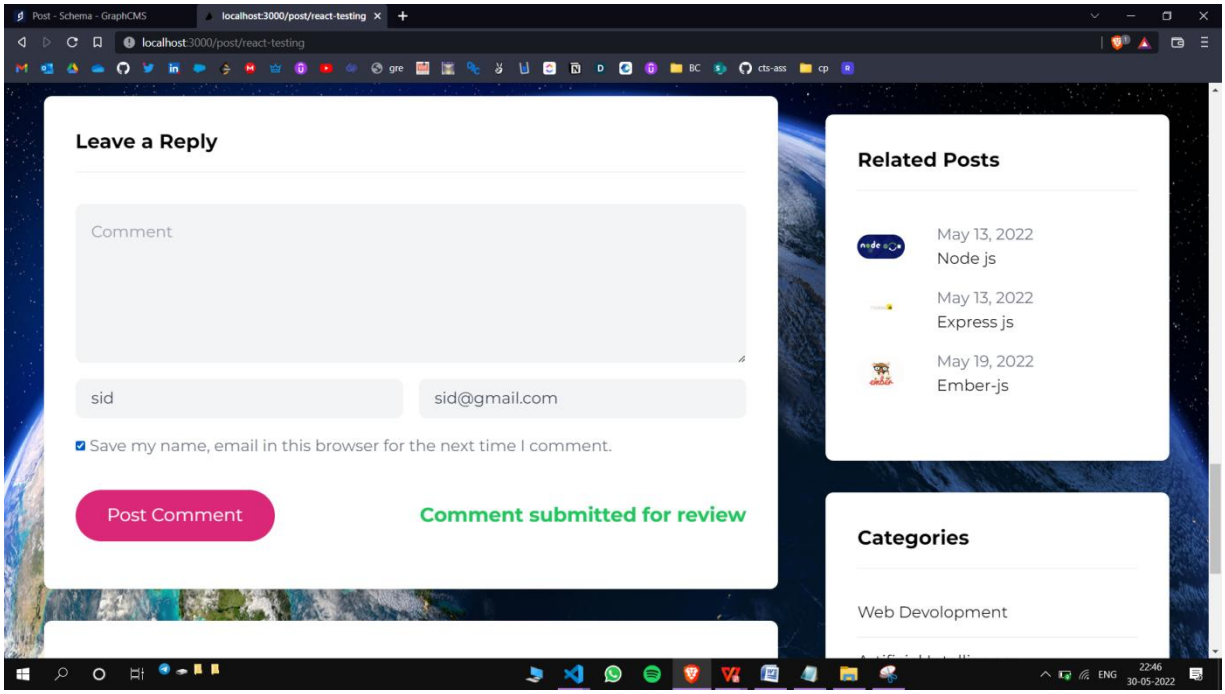
3.6 SCREENSHOTS



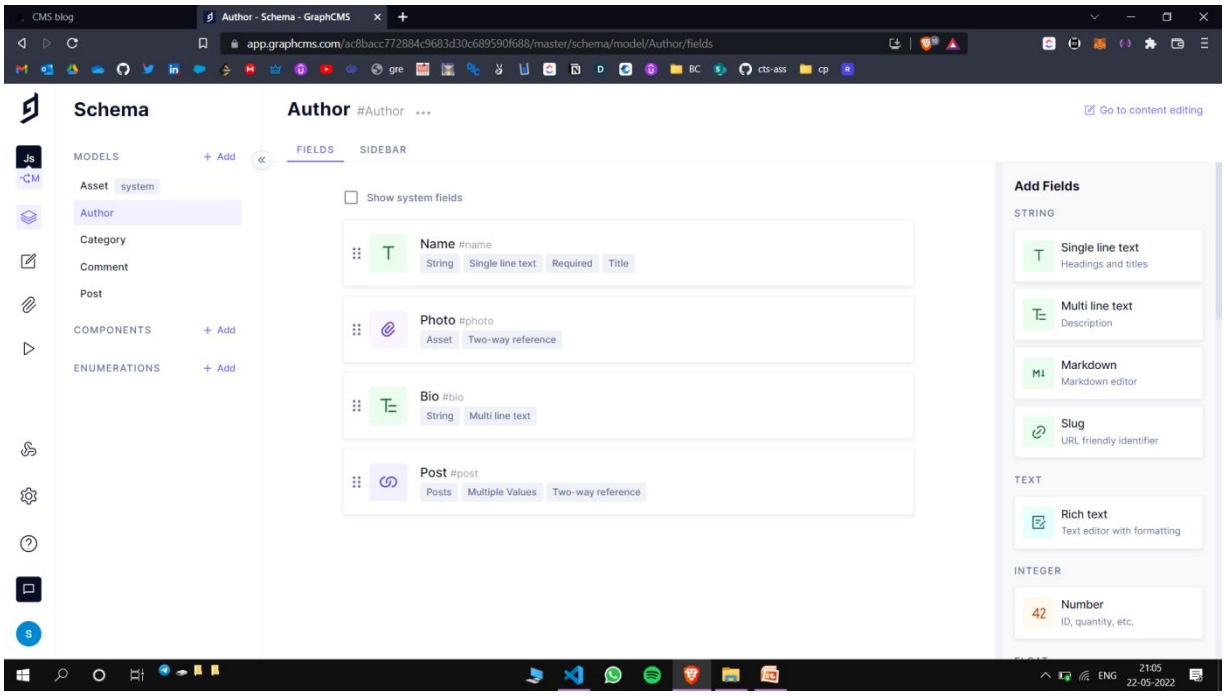
Home screen



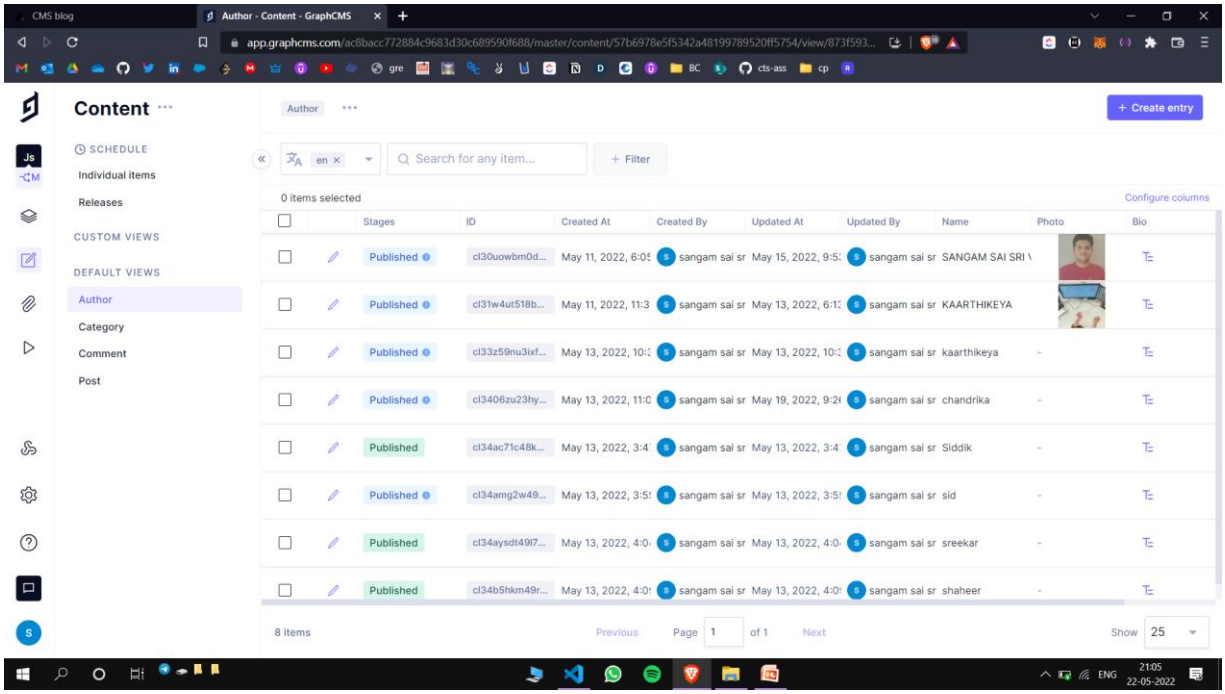
Detailed post view



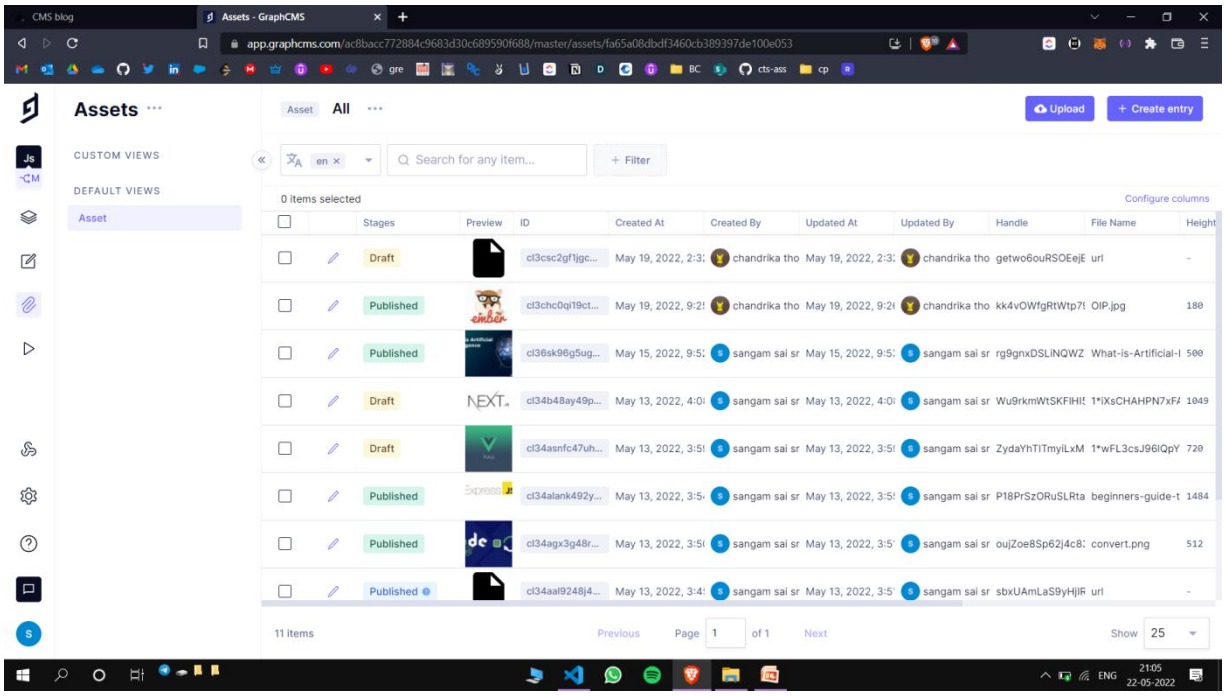
Comment submission



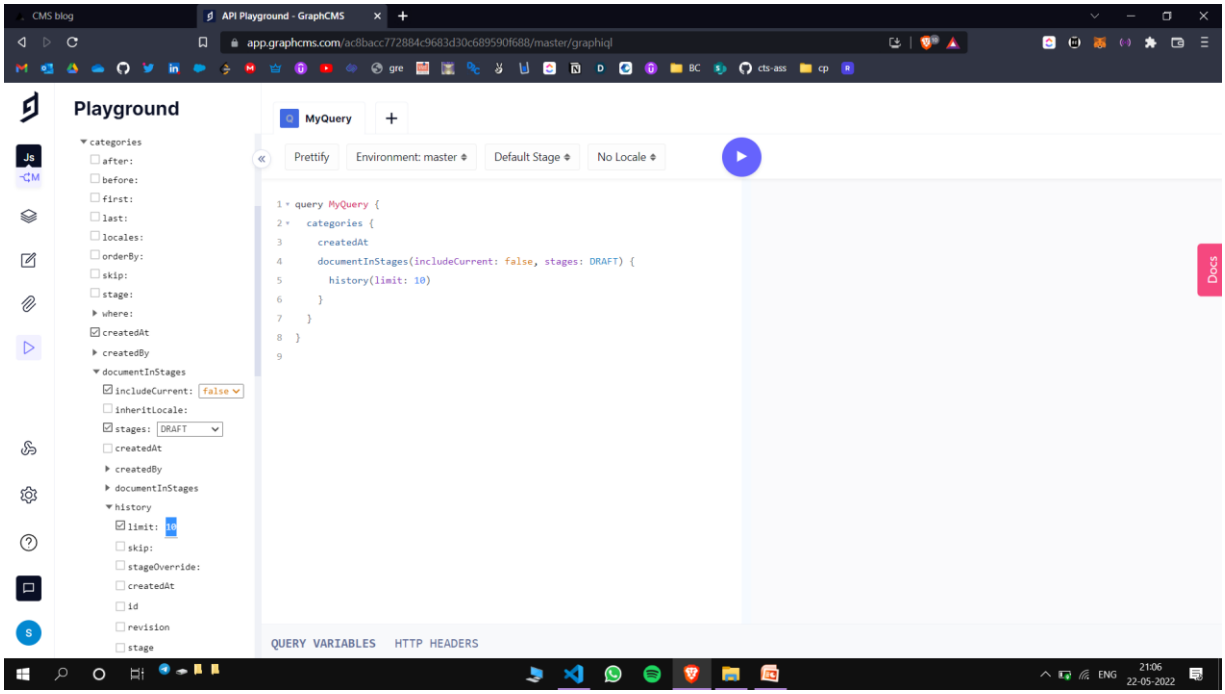
Author schema



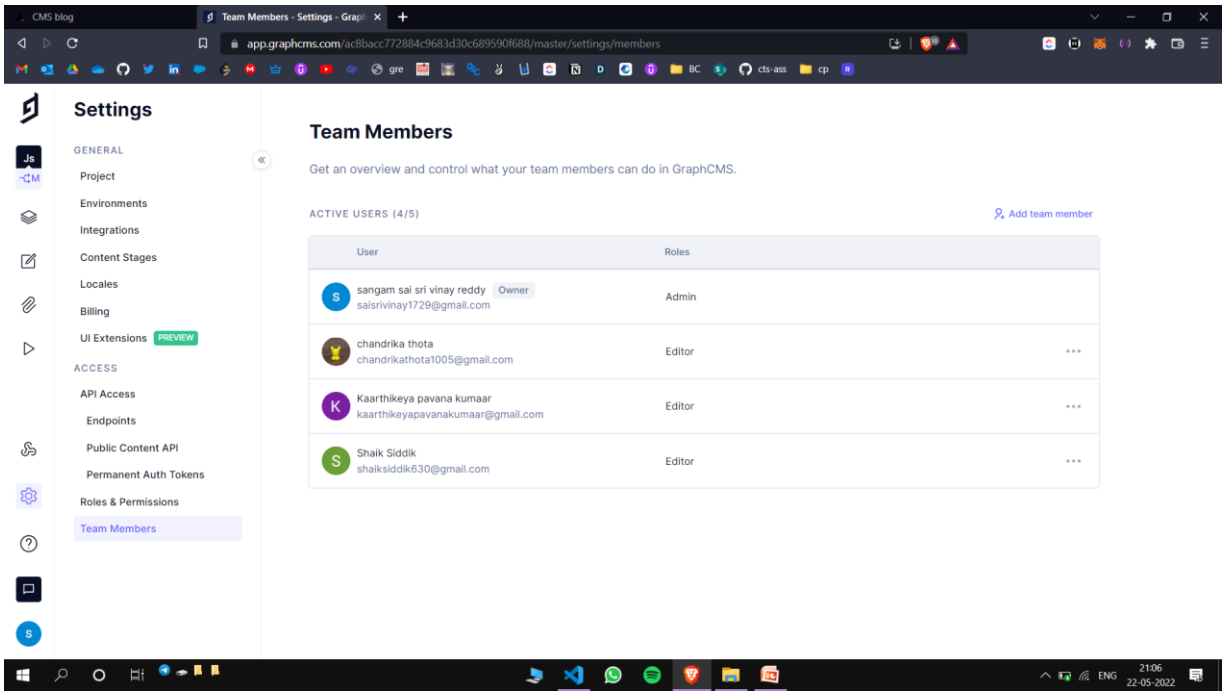
Document maintenance



Asset Maintenance



API playground



Admin panel

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