Institute of Engineering & Technology

Final Project Report: Blogging Web Application Using MEAN Stack

Submitted by:

Shubham Sharma (181500696)

Vivek Kumar Sigh (181500818)

Submitted to:

Faculty Name: Mr. Akash Kumar Choudhary

Declaration

I the undersigned solemnly declare that the project report BLOGGING WEB APPLICATION USING MEAN STACK is based on my own work carried out during the course of our study under the supervision of Mr. Akash Kumar Choudhary. I assert the statements made and conclusions drawn are an outcome of my work. I further certify that I. The work contained in the report is original and has been done by me under the general supervision of my supervisor. II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad. III. We have followed the guidelines provided by the university in writing the report. IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Signature of Candidate: Vivek, Shubham

Name of Candidate: Vivek Kumar & Shubham Sharma

Roll. No. :181500818, 181500696

Course: B. Tech. CSE

Year: 2020-21

Semester: VI

Acknowledgement

This project is an acknowledgement to the intensity drive and technical competence of many persons who have contributed to it. I express my heartiest gratitude and deepest thanks to, Project Instructor Mr. Akash Kumar Choudhary for his proper guidance, suggestions and helping me in completing the project and I am very grateful towards the support he showed throughout the whole project. I am highly grateful to my parents who have been the source of money and encouragement during the course of my work. I would also like to thank all those who directly or indirectly supported or helped me in completing my project in time. I would like to express my gratitude towards my parents and members of my college for their kind cooperation and encouragement which helped me in completion of this project. All of them have willingly helped me out with their abilities.

Table of Contents

1. Introduction to the Project

1.1. Reason for selecting the topic

We chose the topic because we wanted to build something really dynamic system for posting blogs for the users. The main aim of developing this platform is to create a team blog, allowing multiple bloggers to contribute to a single blog. We select which team members have administrative authority and which are authors only. We can also choose to make our blog private and restrict who can view it.

1.2. Objectives of the project

Publishing content to a blog is frequently seen as a decent method to connect with people with similar interest. This blog is a kind of site that is included sections either made by the understudies or different individuals. A blog is an extraordinary path for an individual to figure out how to make their own site. There are many options available for blog like WordPress, web blog etc. In this we can create only individual blogs on which we can post or share anything that we want which is publicly available. There is no way to create team blogs for any organization, company or institute. This will be done using the mean stack which basically is an open-source JavaScript software for building web sites and web application. ***MEAN*** is an acronym for MongoDB, ExpressJS, AngularJS and Node. The main objective of the web Blogging System is to manage the small print of Blogs, Topic ,Idea, Content, Entries. It manages all the knowledge regarding Blogs, Views, Entries, Blogs. The project is completely engineered at body finish and so solely the administrator is secure the access. the aim of the project is to create AN application to cut back the manual work for managing the Blogs, Topic, Views, Idea. It tracks all the small print regarding the thought, Content, Entries.

1.3. Feasibility Study

After doing the project blogging system, study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible – given unlimited resources and infinite time.

A) Economical Feasibility

B) Technical Feasibility

C) Operational Feasibility

1.4. Future Scope

This web application will be utilizing this publishing content to a blog stage educator can undoubtedly advance significant notification identified with the association to their understudies. Understudies can post their recommendations and input for the association. In the event that somebody can't communicate her/his view before individuals those likewise share the things through this contributing to a blog framework.

It can be summarized that the future scope of the project circles around maintaining information regarding:

1) Implement the backup mechanism for taking backup.

2) Automatic detection of errors like grammar.

3) Automatic notification to followers after posting blog

1.5. Modules in Blogging Web Application

- **Blogs Management Module:** Will be used for managing the Blogs details.
- **Views Module:** Will be used for managing the details of views.
- **Content Module:** Will be used for managing the details of content.
- **Login Module:** Will be used for managing the login details.
- **Users Module:** Will be used for managing the users of the system.

1.6. Input Data and Validation of Project on Online Blogging System

- All the fields such as Blogs, Topic, Views are validated and does not take invalid values.
- Each form in the application will not accept the blank value fields.
- Validations for user input will be done.
- Checking of the Coding standards to be maintained during coding.
- Testing the module with all the possible test data.

2. About Online Blogging Web Application

The application is reduced as much as possible to avoid errors while entering the data. It also provides error messages while entering the invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it provides it is user-friendly. Online Blogging System, as described above, can lead to error free, reliable, secure and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on record keeping. The purpose of Online Blogging Web Application is to manage all the blogs, comments and all other basic information about the blog. Making User able to see the blogs posted by other users. The user can edit their own blogs as per their requirements. The aim is to make adding, updating and removing the posts as efficient as possible as to improve the resource management of the blogs data.

3. Methodology

This web application will be designed using HTML, CSS, JavaScript, AngularJS, mongo dB, NodeJS, and Express. The frontend part of the project will be done by HTML, CSS, JavaScript and the backend part will be done by NodeJS. All the data will be stored in mongoDB.

Mean Stack refers to a collection of JavaScript technologies used to develop web applications. Therefore, from the client to the server and from server to database, everything is based on JavaScript. ***MEAN*** is a full-stack development toolkit used to develop a fast and robust web applications. MEAN is a user-friendly stack which is the ideal solution for building dynamic websites and applications. This free and open-source stack offers a quick and organized method for creating rapid prototypes for web-based applications.

Components of MEAN Stack

MEAN is comprised of four different technologies:

- **M**ongoDB express is a schema less NoSQL database system
- **E**xpress JS is a framework used to build web applications in Node
- **A**ngularJS is a JavaScript framework developed by Google
- **N**ode.js is a server-side JavaScript execution environment

The online blogging system has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The Application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus, by this all it proves it is user-friendly. Online blogging system. It can Assist the user to concentrate on their other activities rather to concentrate on the record keeping.

HTML: HTML, which stands for Hypertext Mark-Up Language, is the language for describing structured documents as well as the language used to create web pages in the Internet.

CSS**:** Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

JavaScript: JavaScript is a scripting language and it is used on both client-side and server-side for making web pages interactive. It is a text-based language. Open and cross-platform.

MongoDB: MongoDB is a NoSQL document-oriented database; it is used for storing large amounts of data.

This platform provides an environment for posting or sharing knowledge throughout the organization. This gives detailed information about functional and non-functional requirements of users or students i.e. whatever they want to share then they can easily write on their blog and it can be viewed by the member in the organization. The purpose meets the goal of sharing the information in formal or in secure way.

4. Introduction to Technologies of MEAN Stack

- **M: -** M stands for MongoDB. Working with MongoDB NoSQL database is much easier than working with any relational database. There are no tables in MongoDB. All the data is stored in JSON format i.e. key-value pairs. In JSON, we define a unique key with a value associated with it. These key-value pairs are stored in a document, which in turn is stored in a collection. A collection in MongoDB can have any number of documents and such documents can have any number of key-value pairs. As I mentioned earlier, data in the MongoDB database is stored in BSON. BSON is nothing but extended JSON. It supports more data types than JSON. We store anything like, string, integer, boolean, double, binary data, object, array, javascript code and many more.

  These documents are grouped inside a collection. A collection can be equivalent to a table in a relational SQL database. A collection always exists in a database and there is no pre-defined structure of a collection. In SQL, the database contains tables and in MongoDB, the database contains collections.

- **E: -** E stands for Express.js. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy. Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love. Many popular frameworks are based on Express.

- **A: -** A stands for Angular. Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

  The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks of the Angular framework are Angular components that are organized into NgModules. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

- **N: -** N stands for Node.js. As an asynchronous event-driven JavaScript runtime,

  Node.js is designed to build scalable network applications.

5. Hardware and Software Requirements

1) For VS Code Editor: -

Hardware

• 1.6 GHz or faster processor

• 1 GB of RAM

Platforms

• OS X Yosemite (10.10+)

• Windows 7 (with .NET Framework 4.5.2), 8.0, 8.1 and 10 (32-bit and 64-bit)

• Linux (Debian): Ubuntu Desktop 16.04, Debian 9

• Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 8, Fedora 24 2)

For MongoDB: - Recommended Platforms

• Amazon Linux 2 • Debian 9 and 10

• RHEL / CentOS 6, 7, and 8 • SLES 12 and 15

• Ubuntu LTS 16.04, 18.04, and 20.04

• Windows Server 2016 and 2019

3) Technologies Used HTML, CSS, JAVASCRIPT, NodeJS, AngularJS, ExpressJS, MongoDB, TypeScript

4) Browser Opera, Chrome, Firefox

6. Test Technologies

For testing the web applications we will use “**burp suite”** to find and flaws in the website like XXS(cross site scripting), broken authentication, Security Misconfiguration and etc. With these testing we will make sure that the web application that we created is safe for the users and any errors will be fixed in real time. Apart from software we will make sure to make some test cases to check the website’s feasibility individually. On the other hand, we will make sure to check OWASP to stay update with new bugs and injections.

7. What contribution would the project make and where?

We think that the project will be best suited for someone who likes to get exposer or want to expand their domain, personally or business wise. Because it will help them to send their target audience a message about themselves. This will help them grow and connect to more people. The field it will be effective the most would be for e-commerce for expanding business.

8. Problem Statement

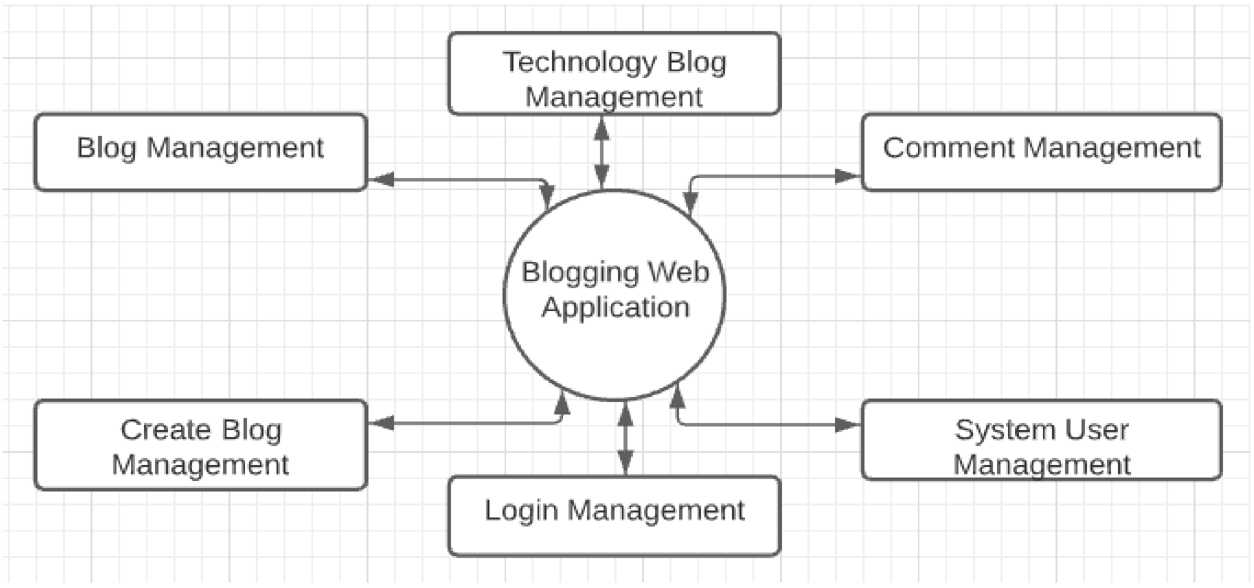
The problems we are going to solve through this project is the common problem that every new blogger face. Through this we are trying to give exposer to the user and convey a message to the target audience that the user need. It will be a place where people can share their knowledge to the world.

9. Scope for Extension into a Major Project

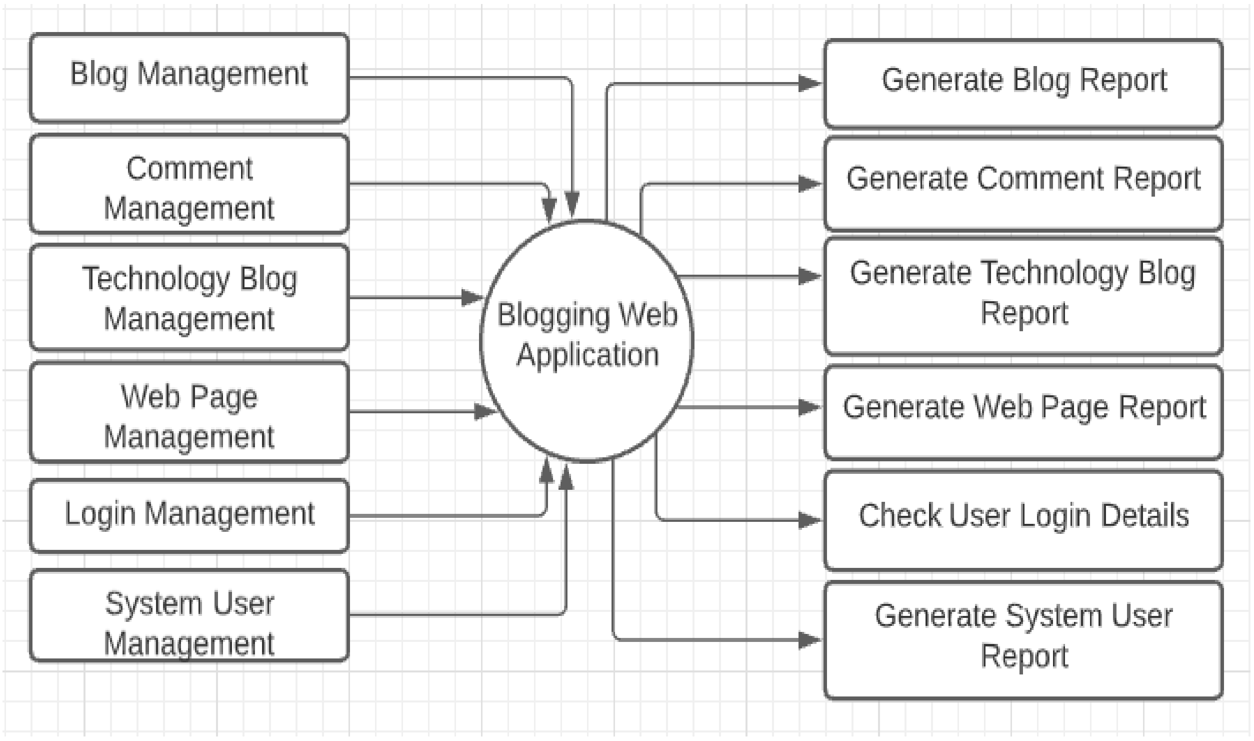
This project can further be extended to major project by adding some of the features which will make it easy for the user to find the most relevant content according to his or her interests by providing a certain set of different categories to the user. Furthermore the people who are following a person will get notified when that person will post some new content. If a user finds something interesting the he can save that post to favorites and at the same time they can leave a like for that post to which the author will get notified if somebody likes their post. Then the user will also be able to share the content to other users.

10. Software Designing

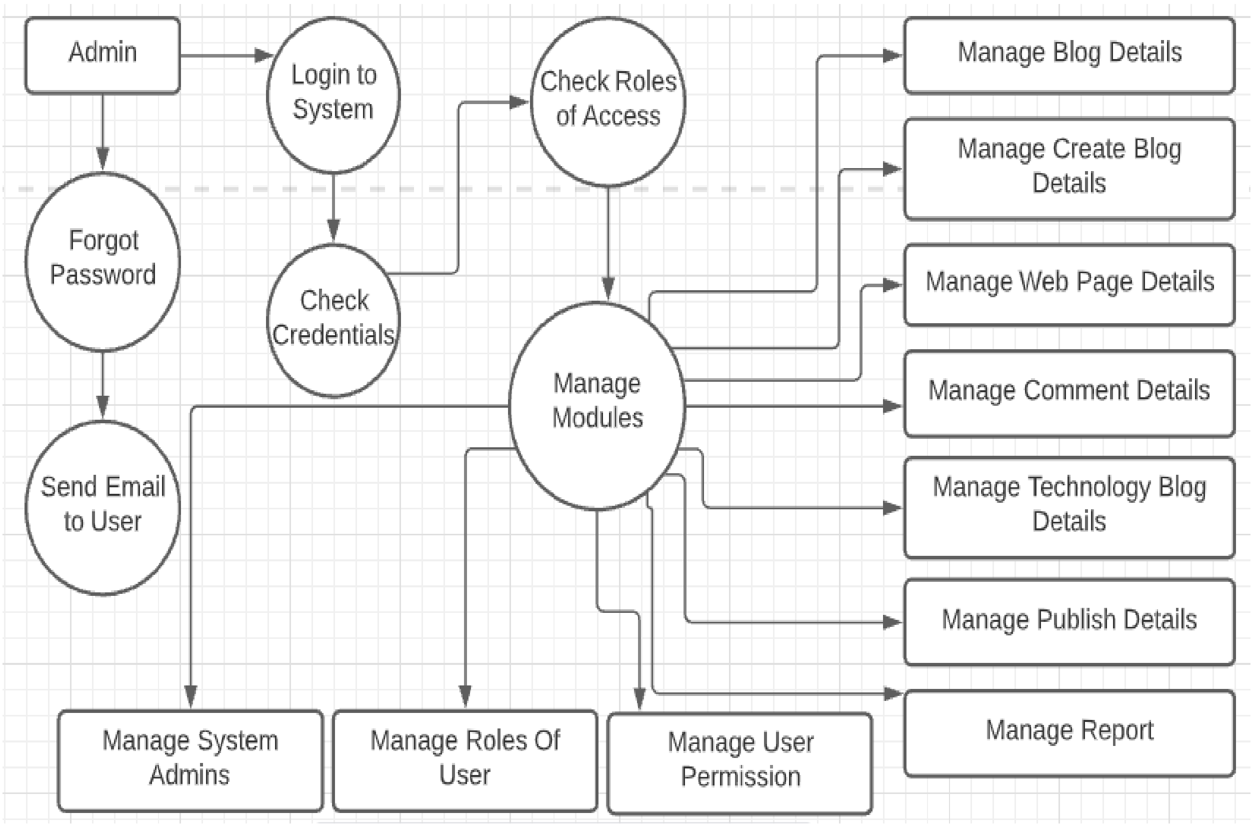
10.1. Zero Level DFD for Blogging Web Application



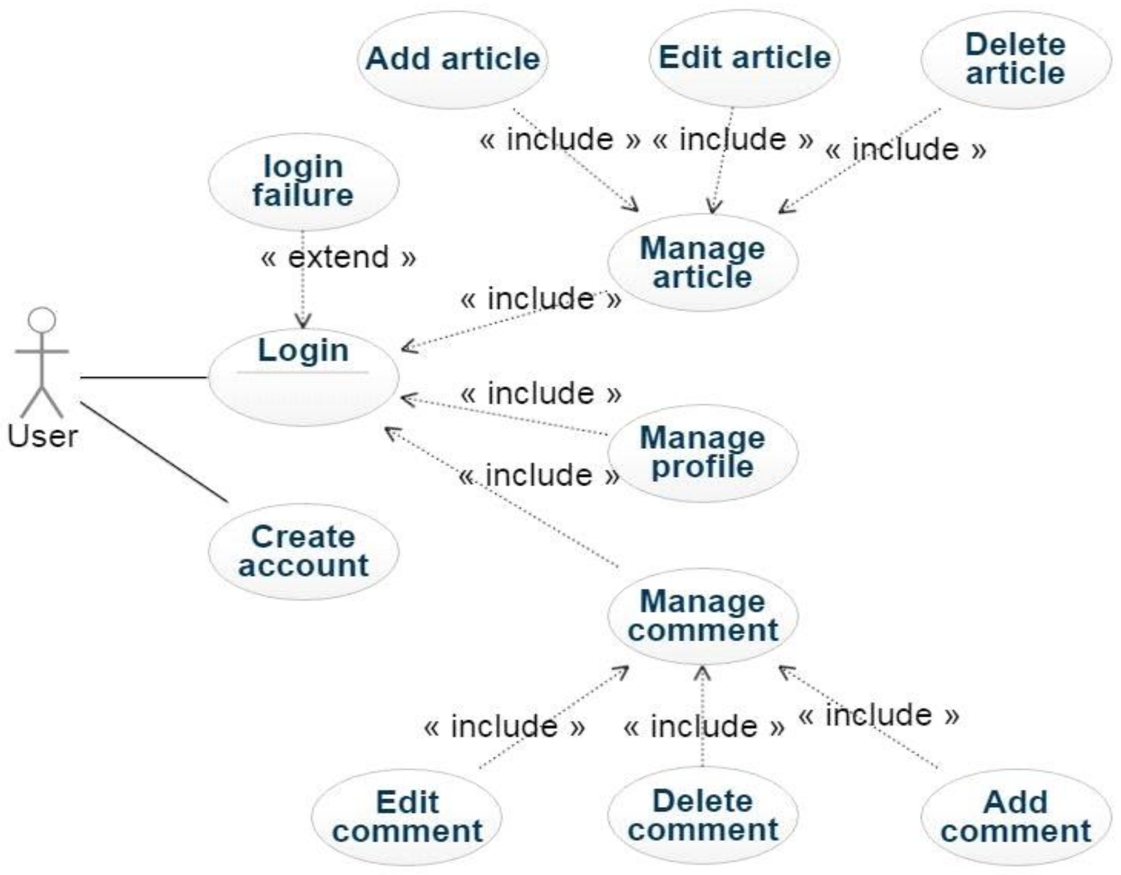
10.2. First Level DFD for Blogging Web Application



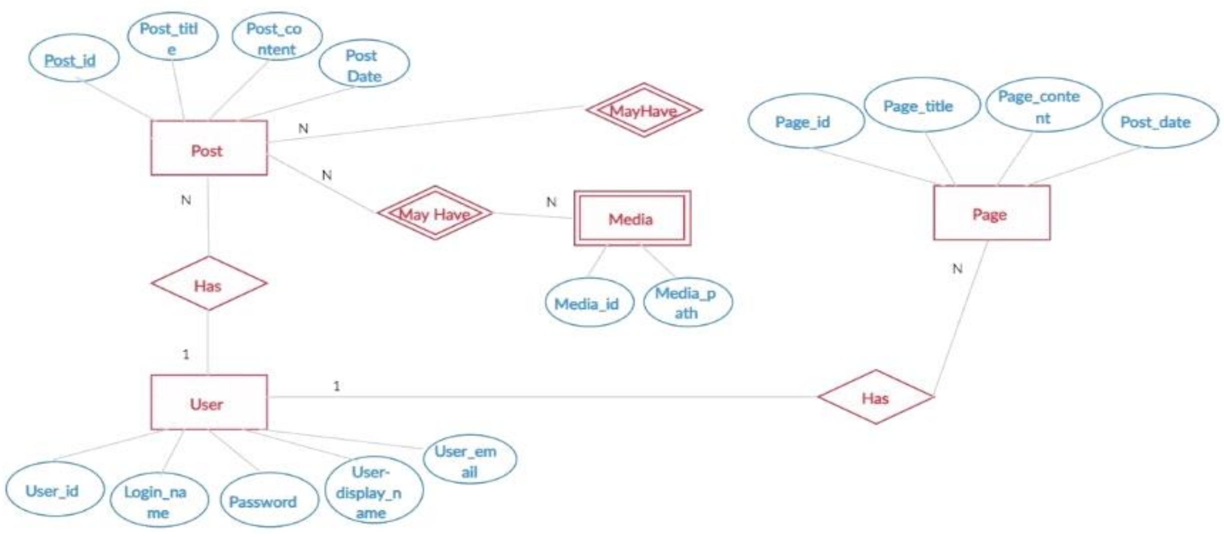
10.3. Second Level DFD for Blogging Web Application



10.4. Usecase Diagram for Blogging Web Application

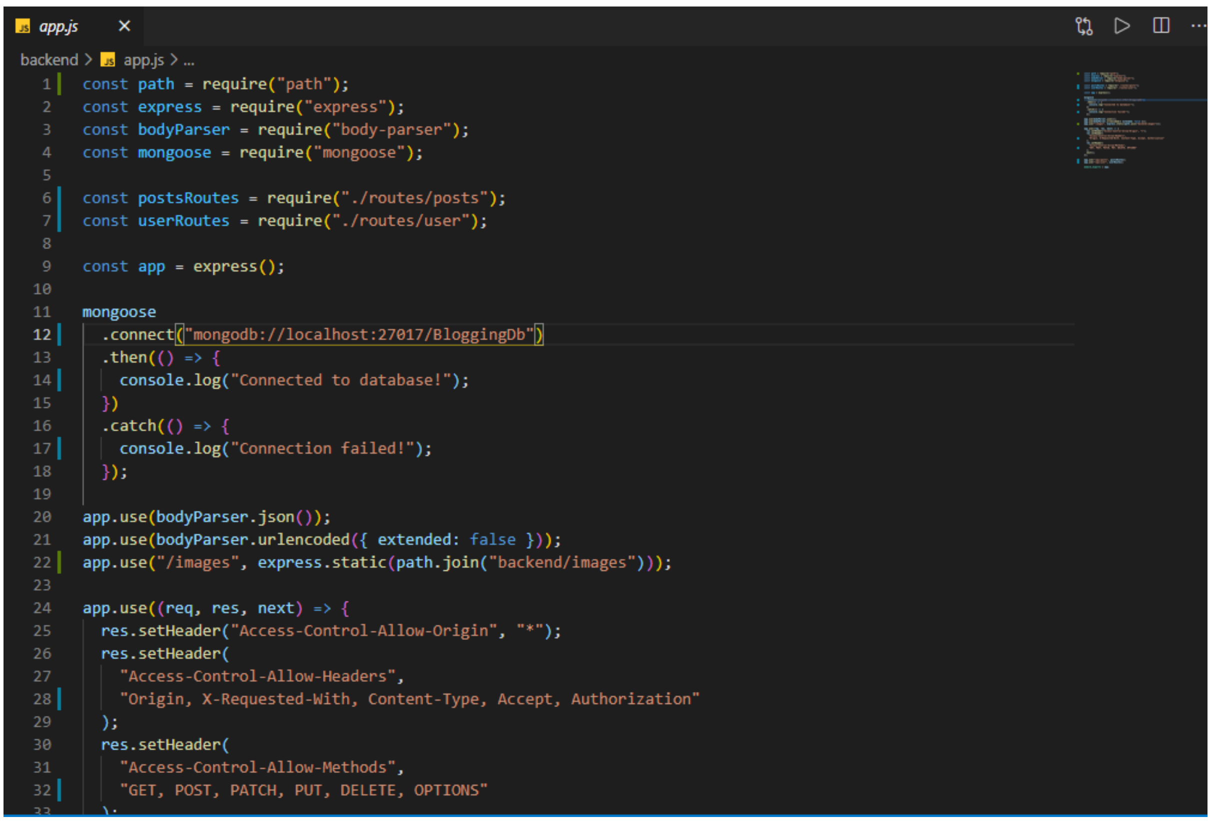


10.5. ER Diagram for Blogging Web Application

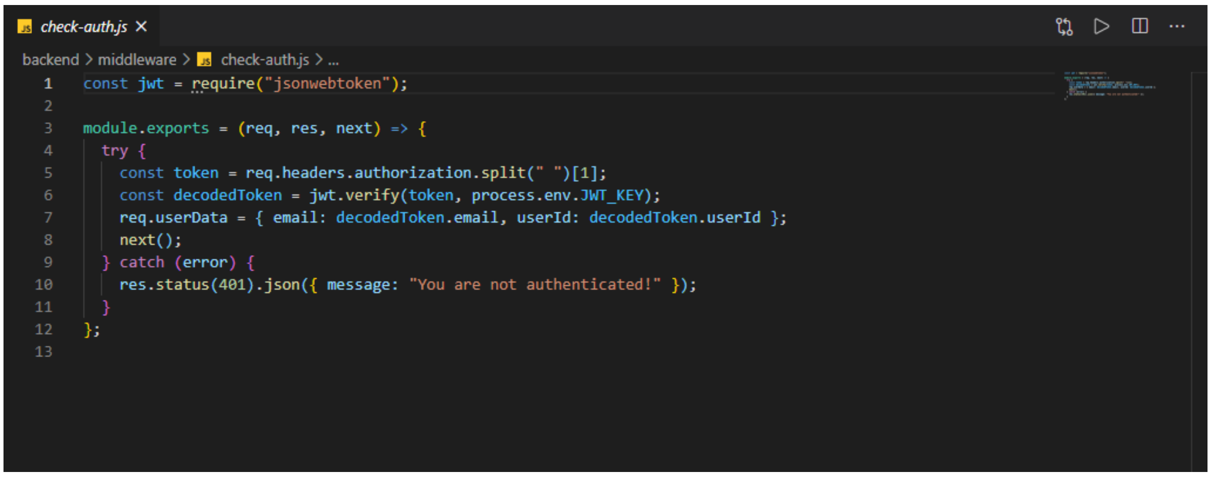


11. Screenshots for Online Blogging Web Application

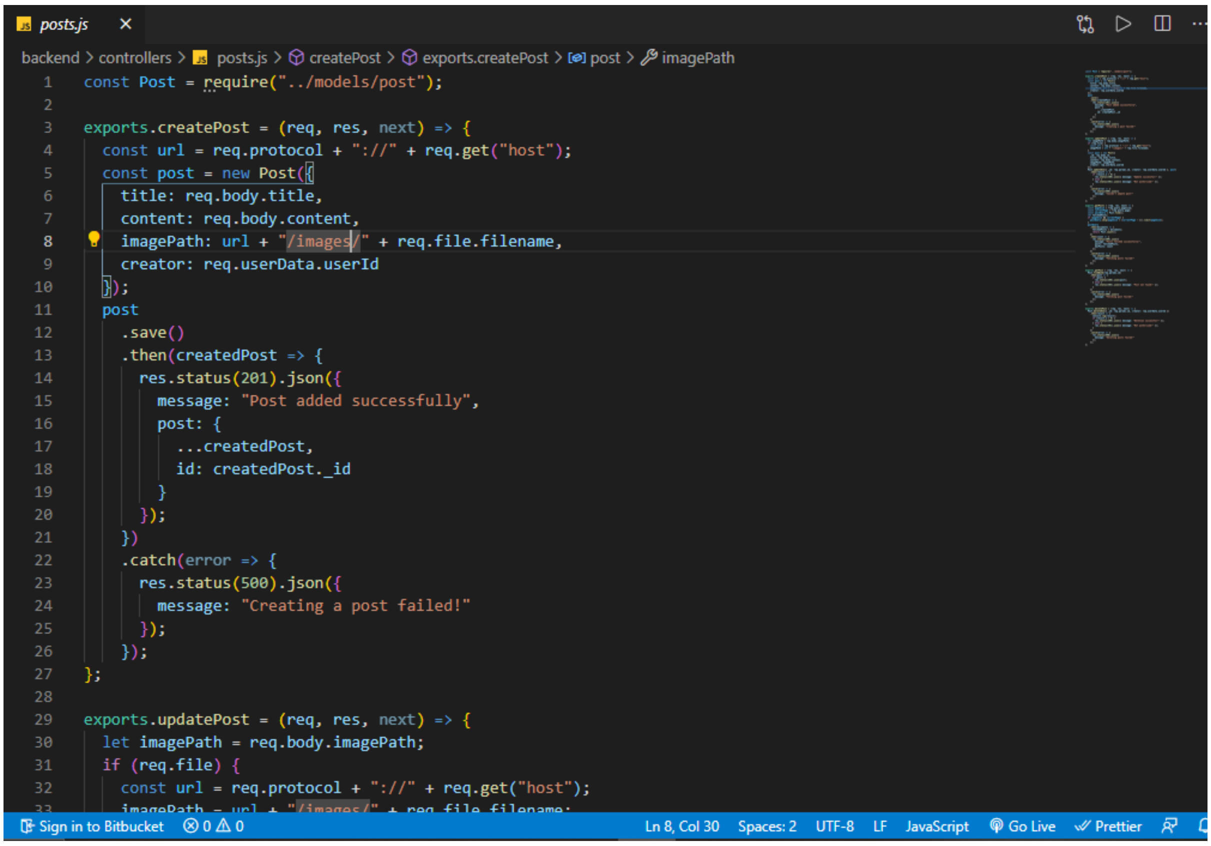
11.1. Implementation Screenshots



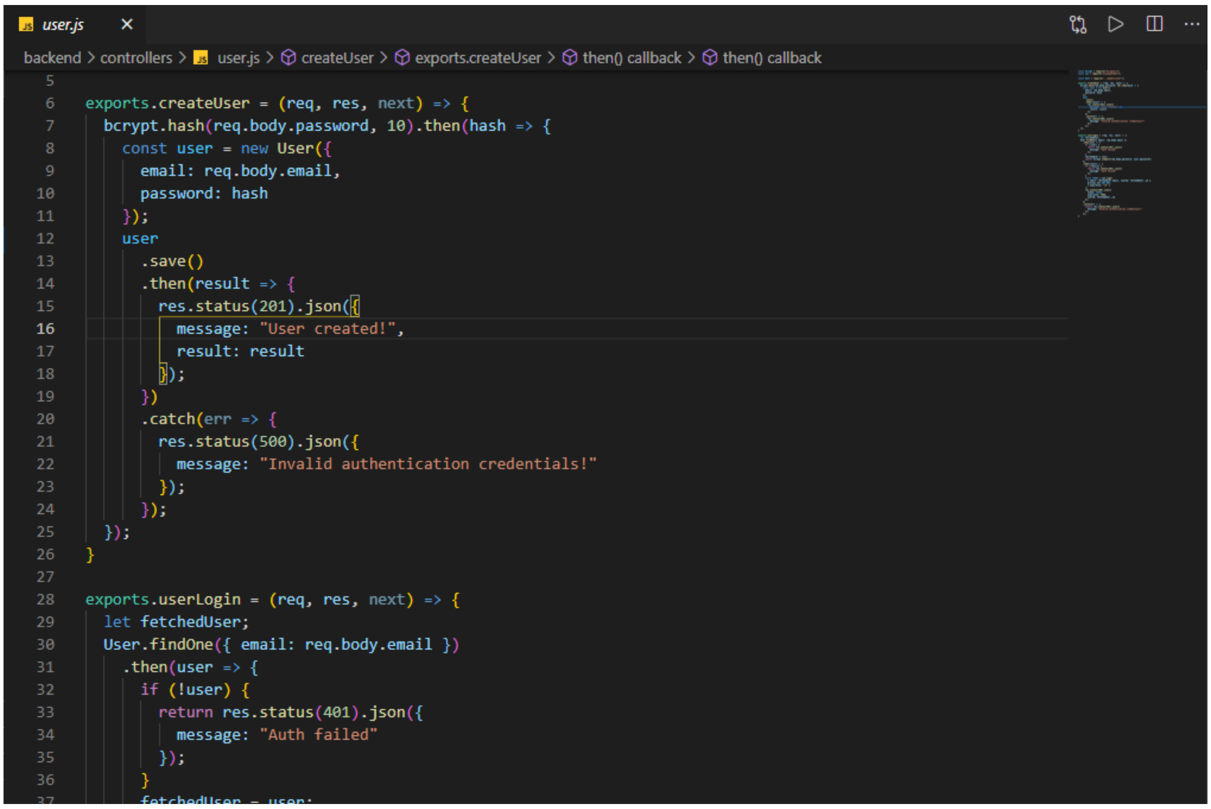
```
const path = require("path");
const express = require("express");
const bodyParser = require("body-parser");
const mongoose = require("mongoose");

const postsRoutes = require("./routes/posts");
const userRoutes = require("./routes/user");

const app = express();

mongoose
  .connect("mongodb://localhost:27017/BloggingDb")
  .then(() => {
    console.log("Connected to database!");
  })
  .catch(() => {
    console.log("Connection failed!");
  });

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use("/images", express.static(path.join("backend/images")));

app.use((req, res, next) => {
  res.setHeader("Access-Control-Allow-Origin", "*");
  res.setHeader(
    "Access-Control-Allow-Headers",
    "Origin, X-Requested-With, Content-Type, Accept, Authorization"
  );
  res.setHeader(
    "Access-Control-Allow-Methods",
    "GET, POST, PATCH, PUT, DELETE, OPTIONS"
```



```javascript
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {
  try {
    const token = req.headers.authorization.split(" ")[1];
    const decodedToken = jwt.verify(token, process.env.JWT_KEY);
    req.userData = { email: decodedToken.email, userId: decodedToken.userId };
    next();
  } catch (error) {
    res.status(401).json({ message: "You are not authenticated!" });
  }
};
```

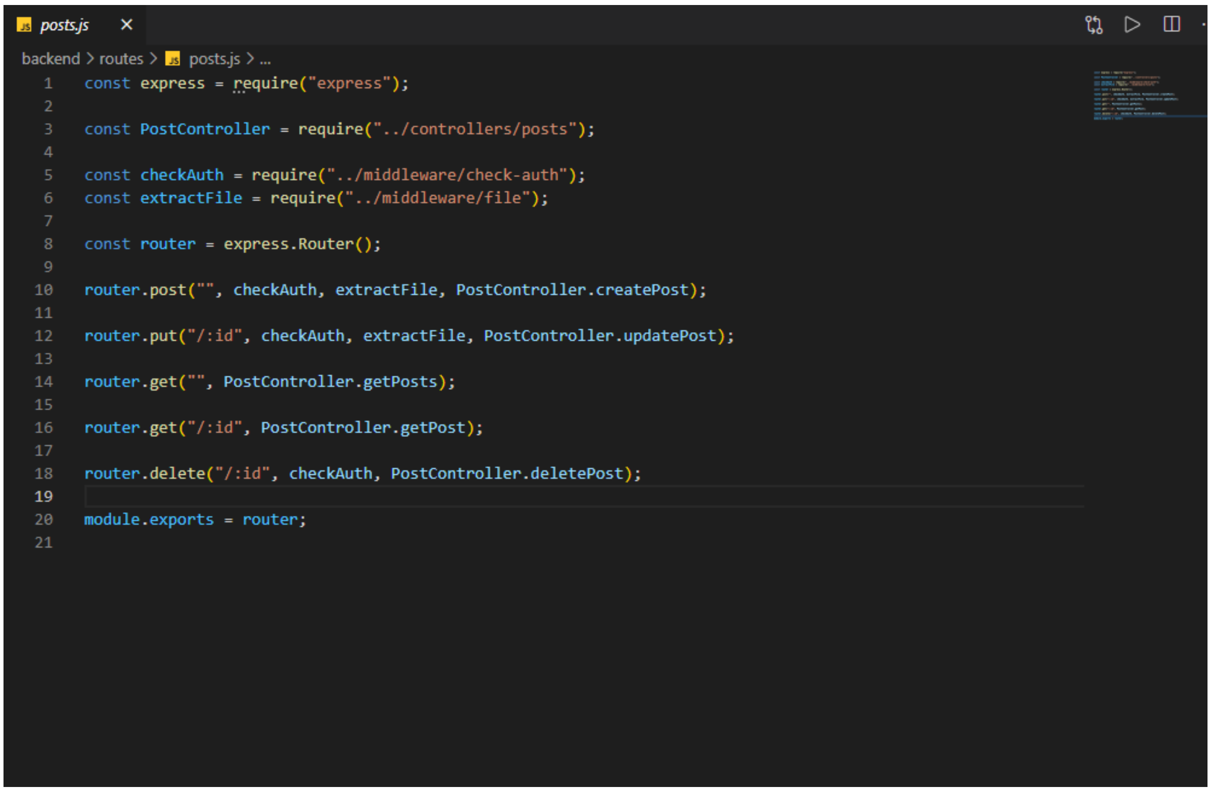

posts.js
backend > controllers > posts.js > createPost > exports.createPost > post > imagePath
Sign in to Bitbucket
0 0
Ln 8, Col 30
Spaces: 2
UTF-8
LF
JavaScript
Go Live
Prettier

```javascript
const Post = require("../models/post");

exports.createPost = (req, res, next) => {
  const url = req.protocol + "://" + req.get("host");
  const post = new Post({
    title: req.body.title,
    content: req.body.content,
    imagePath: url + "/images/" + req.file.filename,
    creator: req.userData.userId
  });
  post
    .save()
    .then(createdPost => {
      res.status(201).json({
        message: "Post added successfully",
        post: {
          ...createdPost,
          id: createdPost._id
        }
      });
    })
    .catch(error => {
      res.status(500).json({
        message: "Creating a post failed!"
      });
    });
};

exports.updatePost = (req, res, next) => {
  let imagePath = req.body.imagePath;
  if (req.file) {
    const url = req.protocol + "://" + req.get("host");
    imagePath = url + "/images/" + req.file.filename;
```

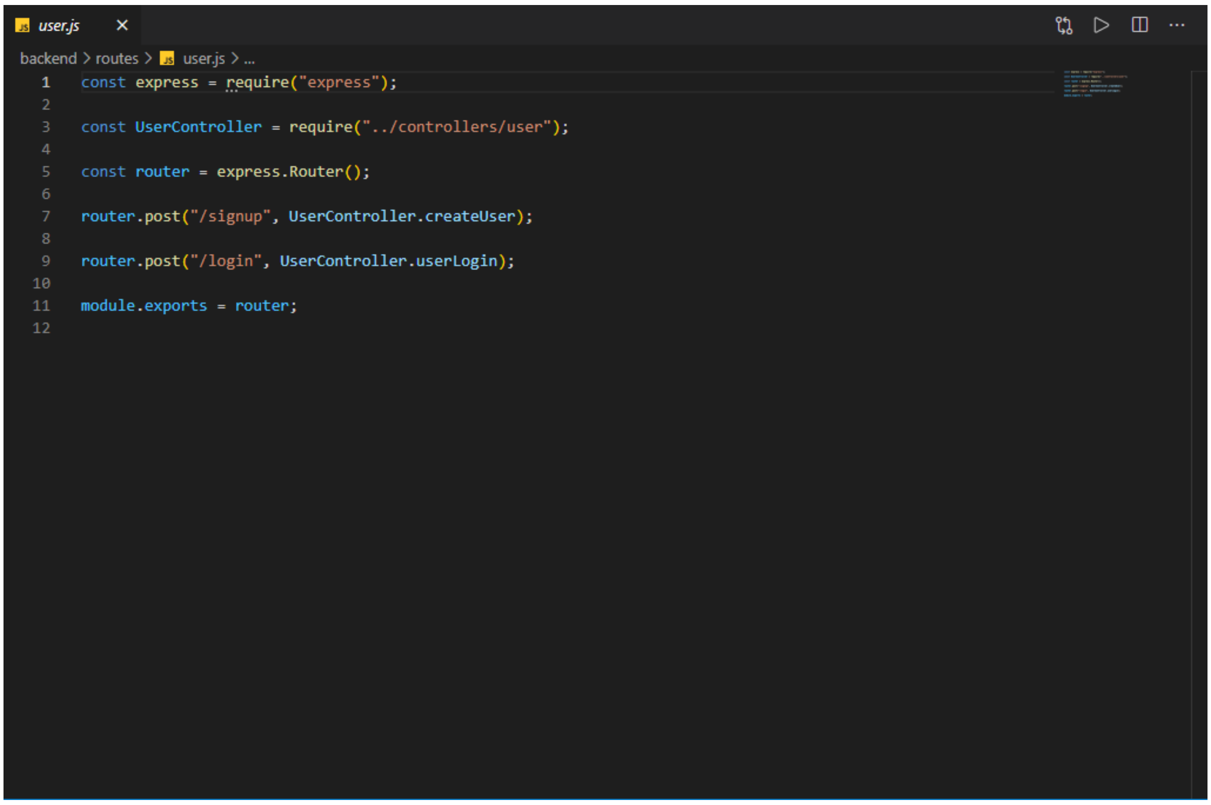


user.js
backend > controllers > user.js > createUser > exports.createUser > then() callback > then() callback

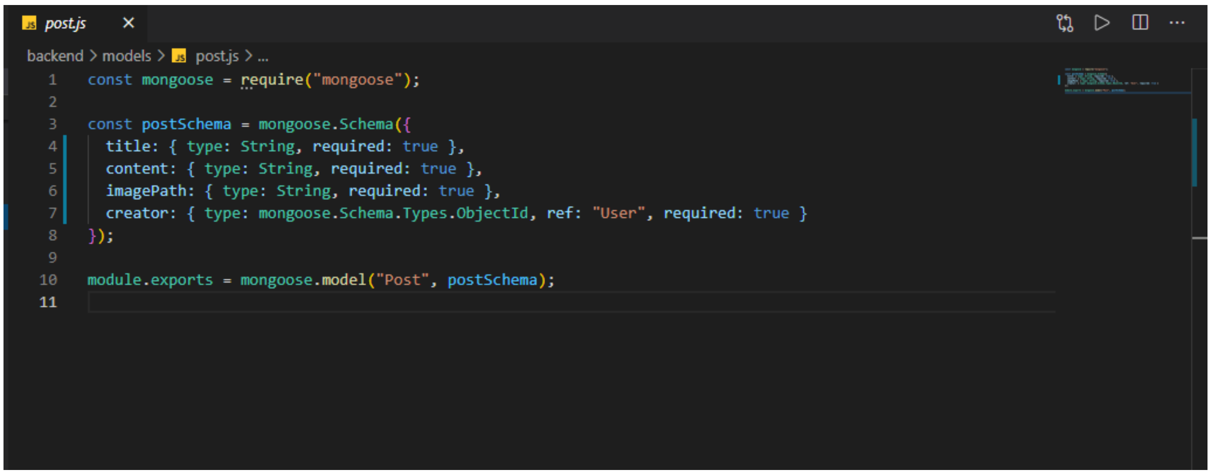
```javascript

exports.createUser = (req, res, next) => {
  bcrypt.hash(req.body.password, 10).then(hash => {
    const user = new User({
      email: req.body.email,
      password: hash
    });
    user
      .save()
      .then(result => {
        res.status(201).json({
          message: "User created!",
          result: result
        });
      })
      .catch(err => {
        res.status(500).json({
          message: "Invalid authentication credentials!"
        });
      });
  });
}

exports.userLogin = (req, res, next) => {
  let fetchedUser;
  User.findOne({ email: req.body.email })
    .then(user => {
      if (!user) {
        return res.status(401).json({
          message: "Auth failed"
        });
      }
```



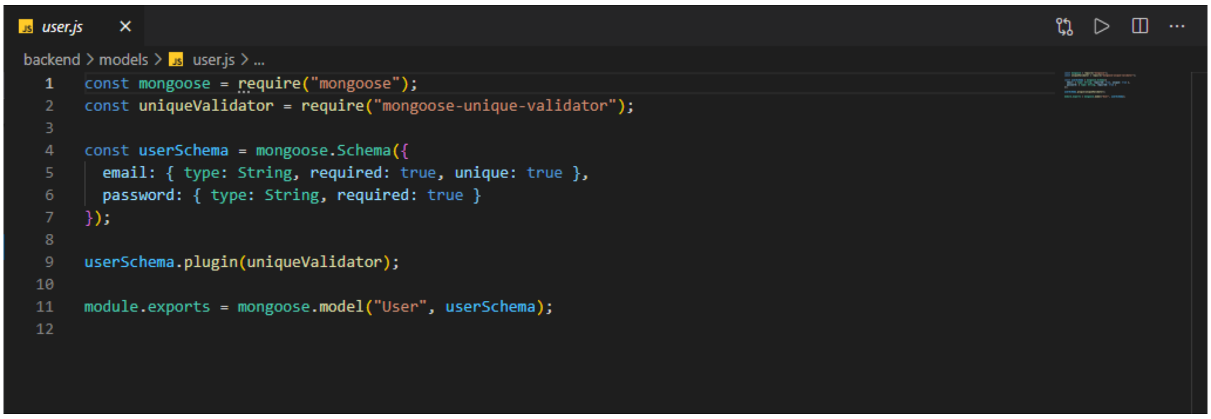
```javascript
const express = require("express");

const PostController = require("../controllers/posts");

const checkAuth = require("../middleware/check-auth");
const extractFile = require("../middleware/file");

const router = express.Router();

router.post("", checkAuth, extractFile, PostController.createPost);

router.put("/:id", checkAuth, extractFile, PostController.updatePost);

router.get("", PostController.getPosts);

router.get("/:id", PostController.getPost);

router.delete("/:id", checkAuth, PostController.deletePost);

module.exports = router;

```



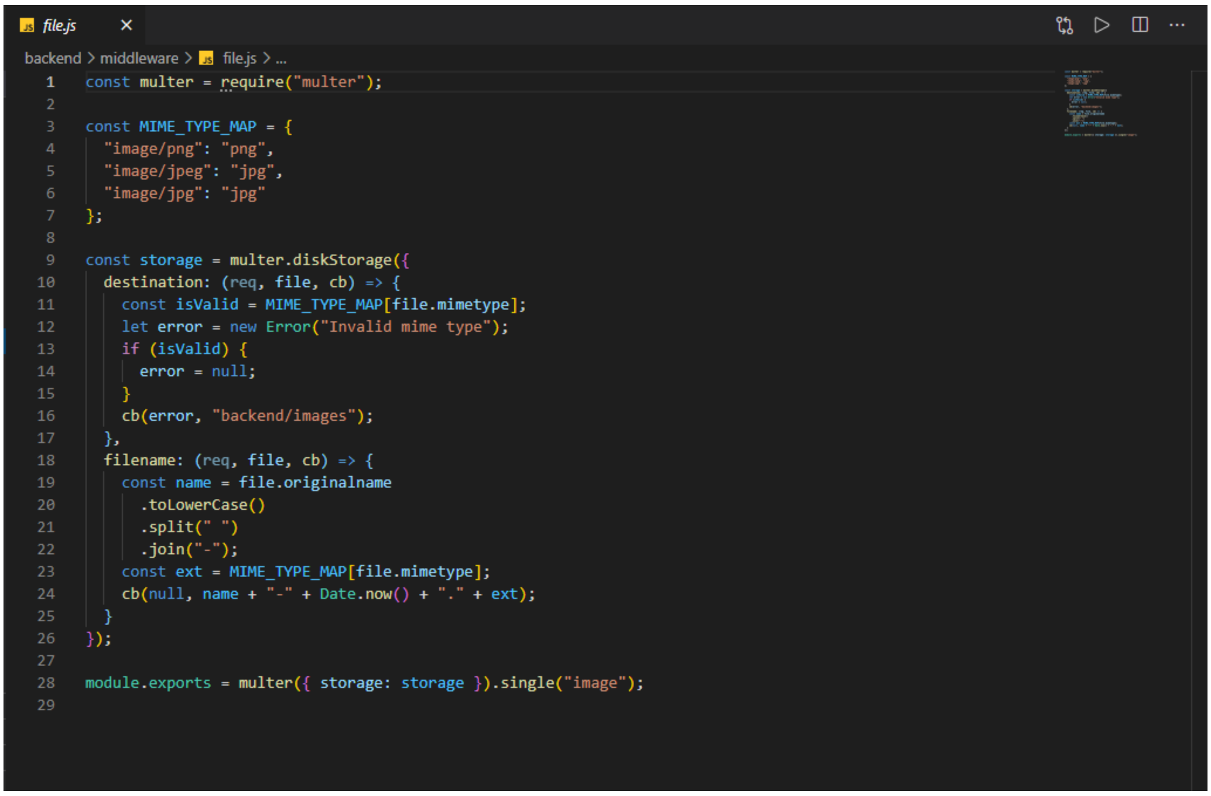
```javascript
const express = require("express");

const UserController = require("../controllers/user");

const router = express.Router();

router.post("/signup", UserController.createUser);

router.post("/login", UserController.userLogin);

module.exports = router;
```



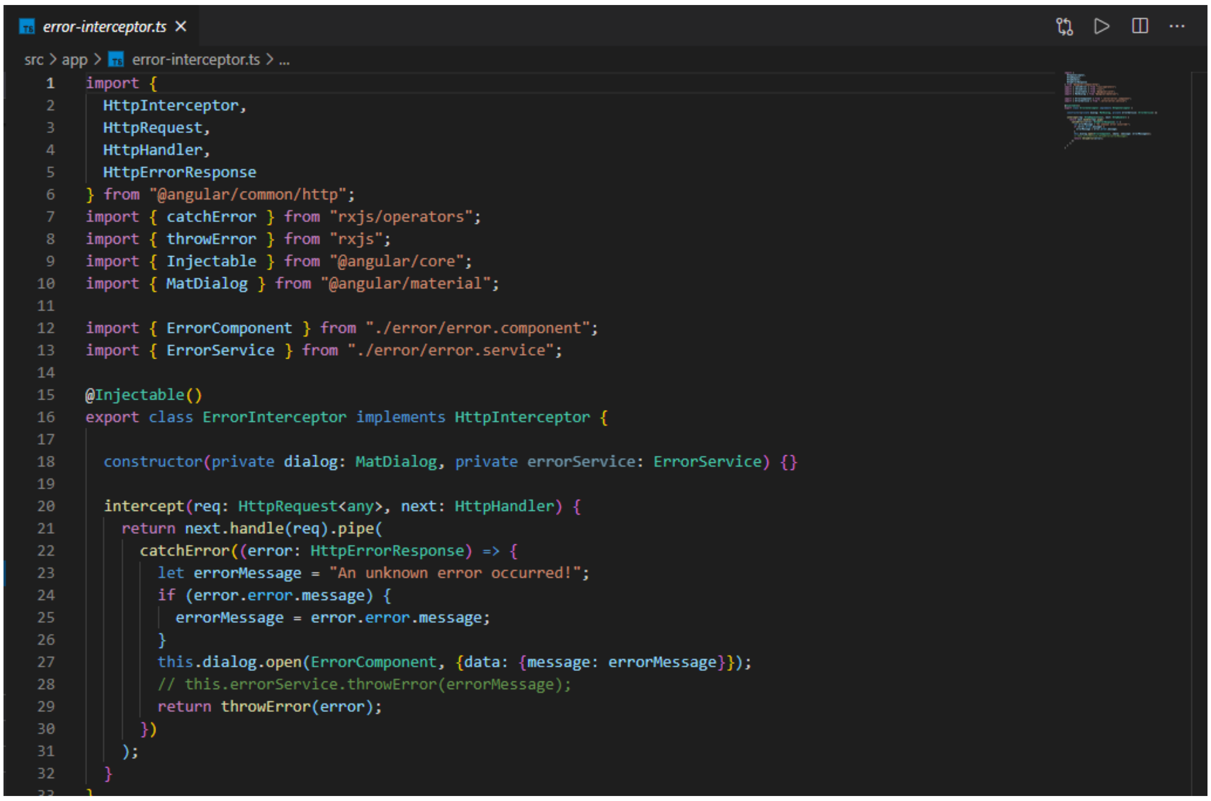
```javascript
const mongoose = require("mongoose");

const postSchema = mongoose.Schema({
  title: { type: String, required: true },
  content: { type: String, required: true },
  imagePath: { type: String, required: true },
  creator: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true }
});

module.exports = mongoose.model("Post", postSchema);
```



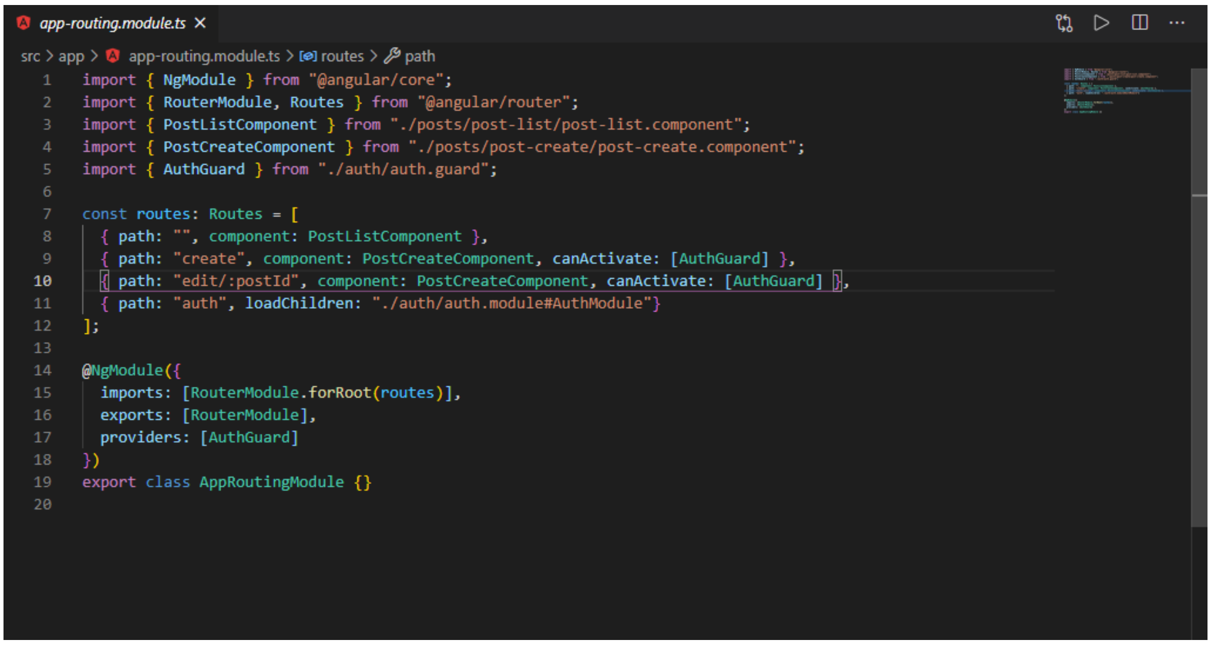
```javascript
const mongoose = require("mongoose");
const uniqueValidator = require("mongoose-unique-validator");

const userSchema = mongoose.Schema({
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

userSchema.plugin(uniqueValidator);

module.exports = mongoose.model("User", userSchema);
```



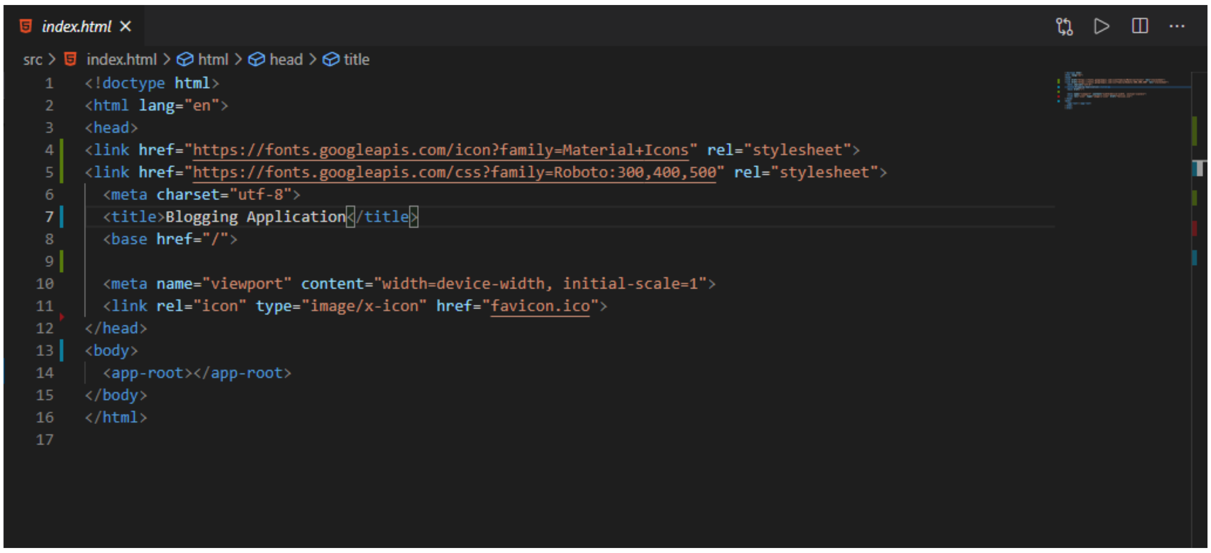
```js
const multer = require("multer");

const MIME_TYPE_MAP = {
  "image/png": "png",
  "image/jpeg": "jpg",
  "image/jpg": "jpg"
};

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    const isValid = MIME_TYPE_MAP[file.mimetype];
    let error = new Error("Invalid mime type");
    if (isValid) {
      error = null;
    }
    cb(error, "backend/images");
  },
  filename: (req, file, cb) => {
    const name = file.originalname
      .toLowerCase()
      .split(" ")
      .join("-");
    const ext = MIME_TYPE_MAP[file.mimetype];
    cb(null, name + "-" + Date.now() + "." + ext);
  }
});

module.exports = multer({ storage: storage }).single("image");
```



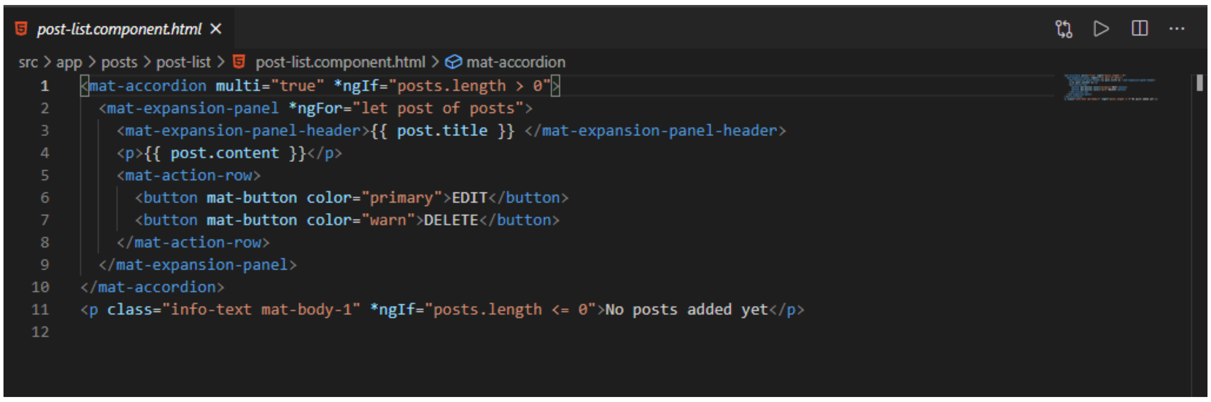
```ts
import {
  HttpInterceptor,
  HttpRequest,
  HttpHandler,
  HttpErrorResponse
} from "@angular/common/http";
import { catchError } from "rxjs/operators";
import { throwError } from "rxjs";
import { Injectable } from "@angular/core";
import { MatDialog } from "@angular/material";

import { ErrorComponent } from "./error/error.component";
import { ErrorService } from "./error/error.service";

@Injectable()
export class ErrorInterceptor implements HttpInterceptor {

  constructor(private dialog: MatDialog, private errorService: ErrorService) {}

  intercept(req: HttpRequest<any>, next: HttpHandler) {
    return next.handle(req).pipe(
      catchError((error: HttpErrorResponse) => {
        let errorMessage = "An unknown error occurred!";
        if (error.error.message) {
          errorMessage = error.error.message;
        }
        this.dialog.open(ErrorComponent, {data: {message: errorMessage}});
        // this.errorService.throwError(errorMessage);
        return throwError(error);
      })
    );
  }
```



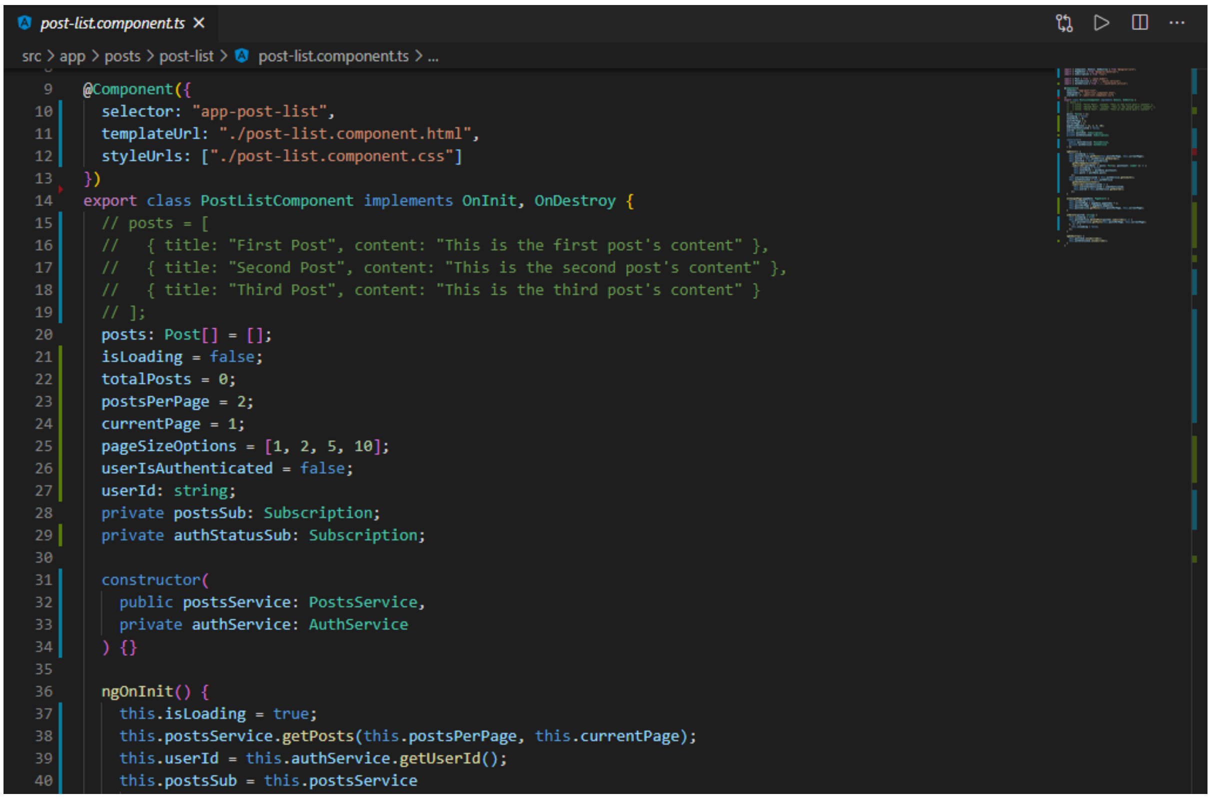
```typescript
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
import { PostListComponent } from "./posts/post-list/post-list.component";
import { PostCreateComponent } from "./posts/post-create/post-create.component";
import { AuthGuard } from "./auth/auth.guard";

const routes: Routes = [
  { path: "", component: PostListComponent },
  { path: "create", component: PostCreateComponent, canActivate: [AuthGuard] },
  { path: "edit/:postId", component: PostCreateComponent, canActivate: [AuthGuard] },
  { path: "auth", loadChildren: "./auth/auth.module#AuthModule"}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
  providers: [AuthGuard]
})
export class AppRoutingModule {}
```



```html
<!doctype html>
<html lang="en">
<head>
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500" rel="stylesheet">
  <meta charset="utf-8">
  <title>Blogging Application</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

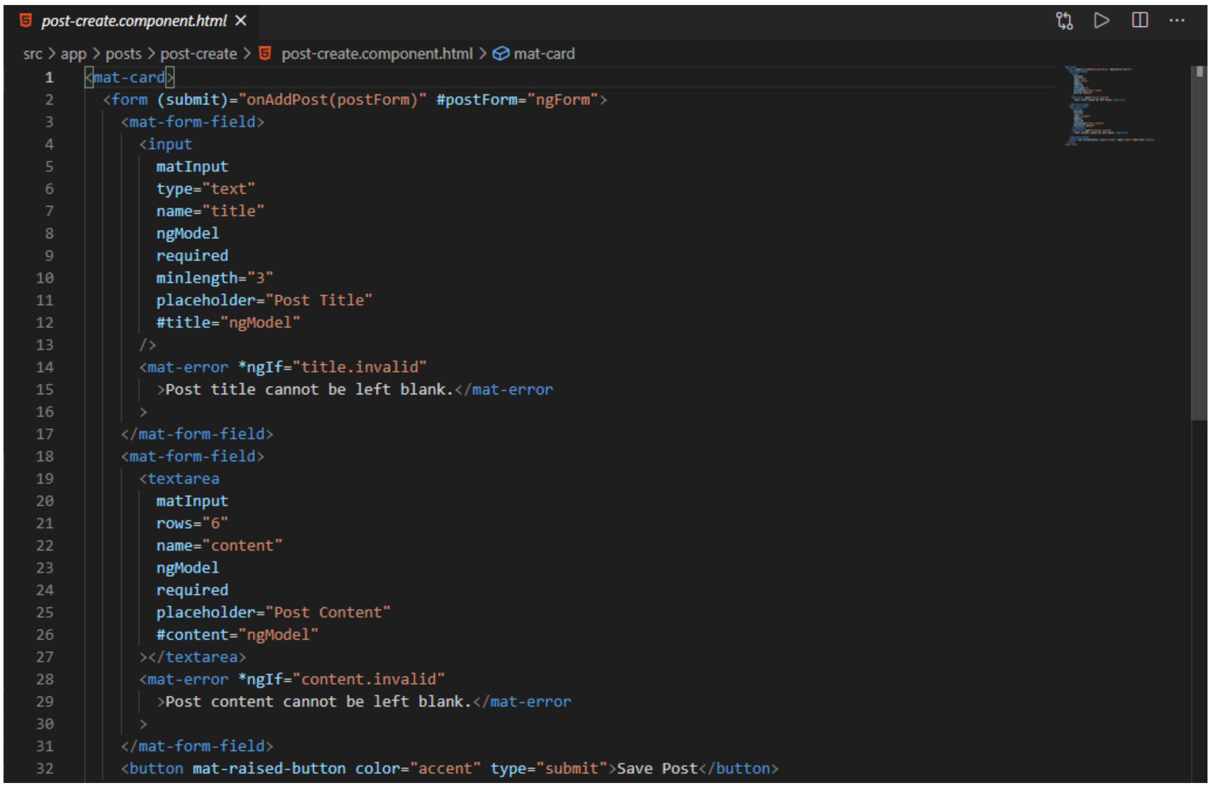

post-list.component.html
src > app > posts > post-list > post-list.component.html > mat-accordion

```html
<mat-accordion multi="true" *ngIf="posts.length > 0">
  <mat-expansion-panel *ngFor="let post of posts">
    <mat-expansion-panel-header>{{ post.title }} </mat-expansion-panel-header>
    <p>{{ post.content }}</p>
    <mat-action-row>
      <button mat-button color="primary">EDIT</button>
      <button mat-button color="warn">DELETE</button>
    </mat-action-row>
  </mat-expansion-panel>
</mat-accordion>
<p class="info-text mat-body-1" *ngIf="posts.length <= 0">No posts added yet</p>
```


post-list.component.ts

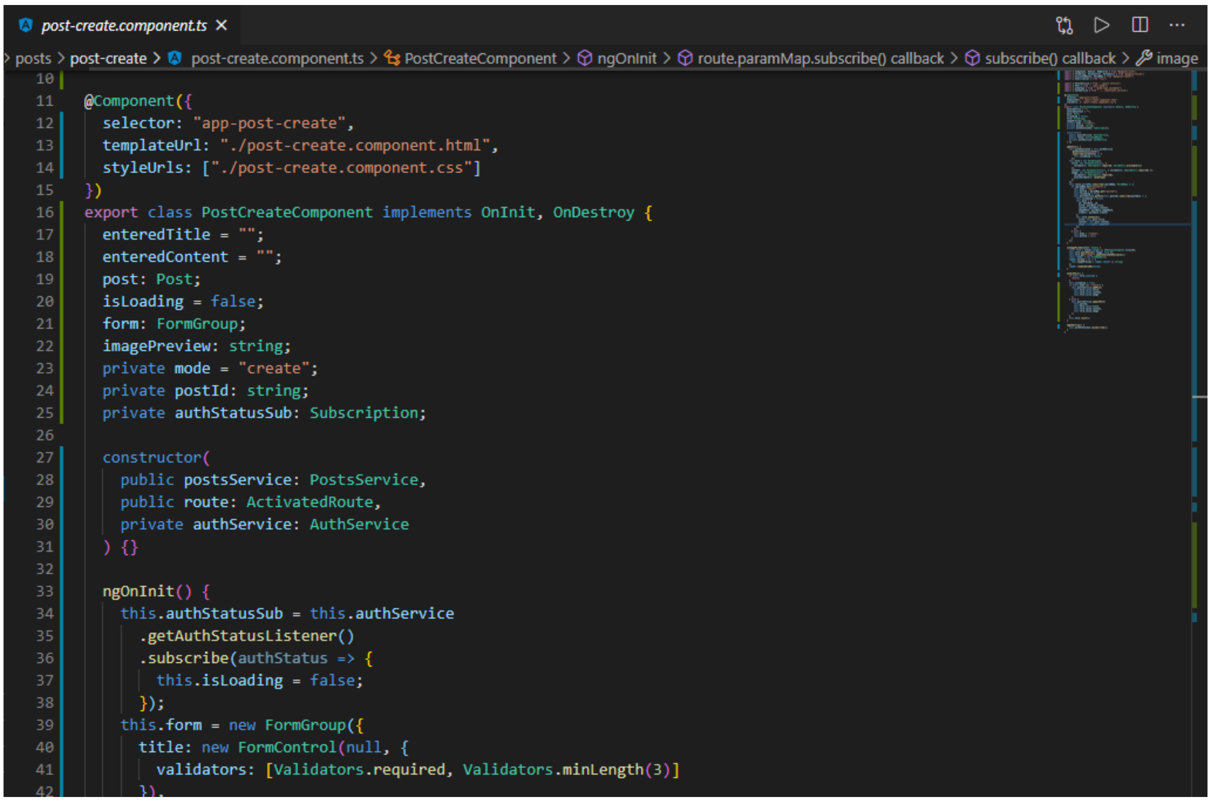
src > app > posts > post-list > post-list.component.ts > ...

```
@Component({
  selector: "app-post-list",
  templateUrl: "./post-list.component.html",
  styleUrls: ["./post-list.component.css"]
})
export class PostListComponent implements OnInit, OnDestroy {
  // posts = [
  //   { title: "First Post", content: "This is the first post's content" },
  //   { title: "Second Post", content: "This is the second post's content" },
  //   { title: "Third Post", content: "This is the third post's content" }
  // ];
  posts: Post[] = [];
  isLoading = false;
  totalPosts = 0;
  postsPerPage = 2;
  currentPage = 1;
  pageSizeOptions = [1, 2, 5, 10];
  userIsAuthenticated = false;
  userId: string;
  private postsSub: Subscription;
  private authStatusSub: Subscription;

  constructor(
    public postsService: PostsService,
    private authService: AuthService
  ) {}

  ngOnInit() {
    this.isLoading = true;
    this.postsService.getPosts(this.postsPerPage, this.currentPage);
    this.userId = this.authService.getUserId();
    this.postsSub = this.postsService
```

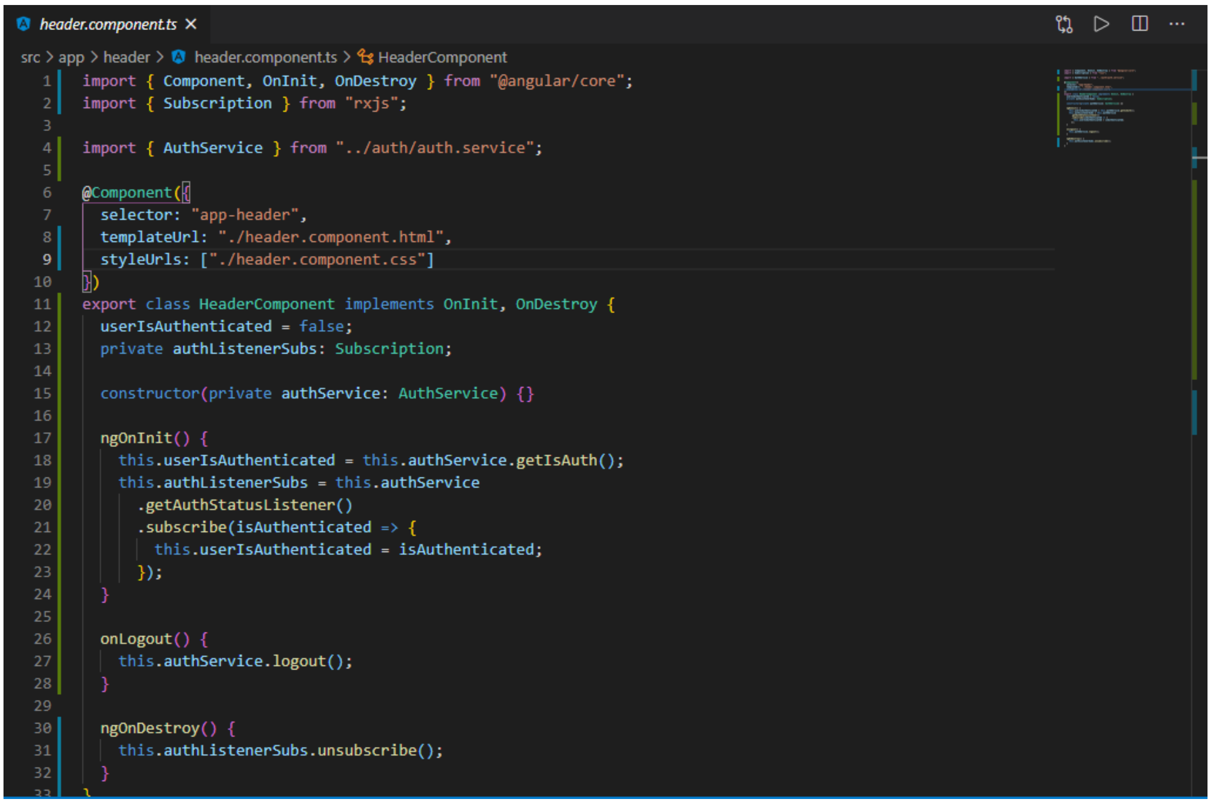

post-create.component.html

src > app > posts > post-create > post-create.component.html > mat-card

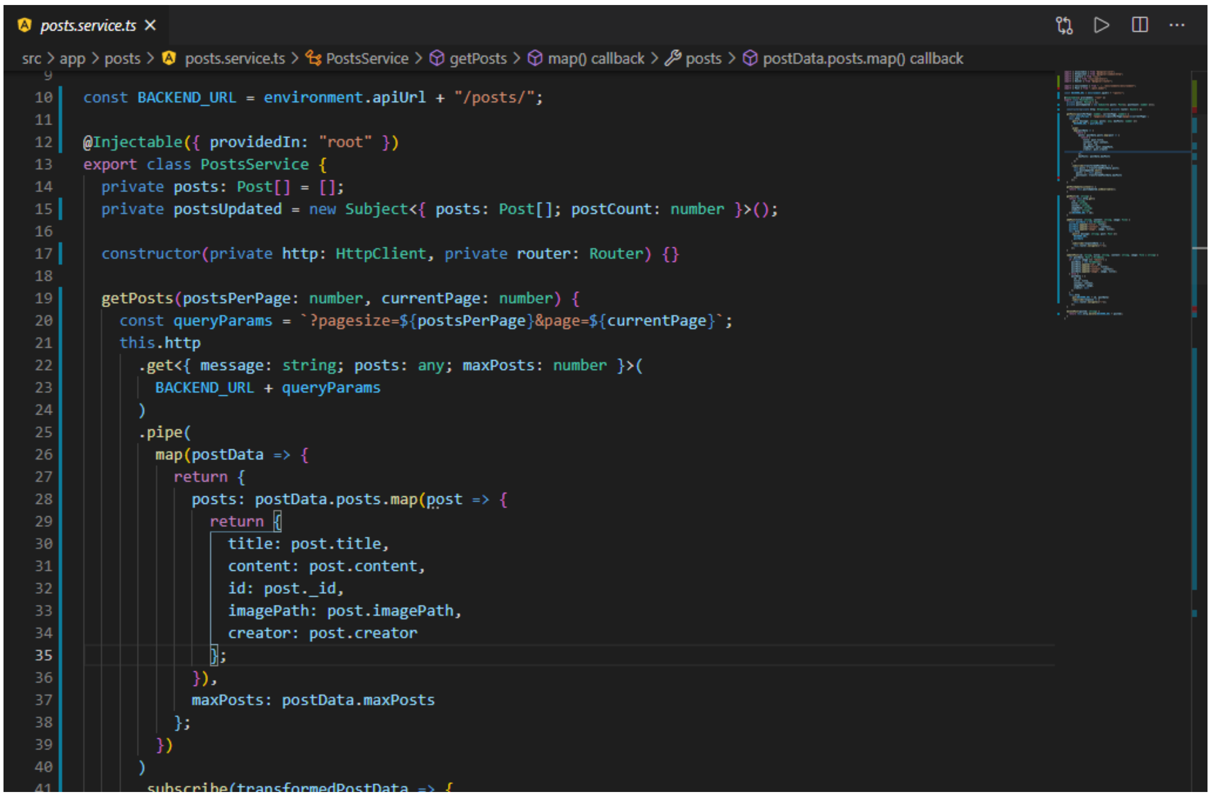
```
<mat-card>
  <form (submit)="onAddPost(postForm)" #postForm="ngForm">
    <mat-form-field>
      <input
        matInput
        type="text"
        name="title"
        ngModel
        required
        minlength="3"
        placeholder="Post Title"
        #title="ngModel"
      />
      <mat-error *ngIf="title.invalid"
        >Post title cannot be left blank.</mat-error
      >
    </mat-form-field>
    <mat-form-field>
      <textarea
        matInput
        rows="6"
        name="content"
        ngModel
        required
        placeholder="Post Content"
        #content="ngModel"
      ></textarea>
      <mat-error *ngIf="content.invalid"
        >Post content cannot be left blank.</mat-error
      >
    </mat-form-field>
    <button mat-raised-button color="accent" type="submit">Save Post</button>
```


post-create.component.ts
› posts › post-create › post-create.component.ts › PostCreateComponent › ngOnInit › route.paramMap.subscribe() callback › subscribe() callback › image

```typescript

@Component({
  selector: "app-post-create",
  templateUrl: "./post-create.component.html",
  styleUrls: ["./post-create.component.css"]
})
export class PostCreateComponent implements OnInit, OnDestroy {
  enteredTitle = "";
  enteredContent = "";
  post: Post;
  isLoading = false;
  form: FormGroup;
  imagePreview: string;
  private mode = "create";
  private postId: string;
  private authStatusSub: Subscription;

  constructor(
    public postsService: PostsService,
    public route: ActivatedRoute,
    private authService: AuthService
  ) {}

  ngOnInit() {
    this.authStatusSub = this.authService
      .getAuthStatusListener()
      .subscribe(authStatus => {
        this.isLoading = false;
      });
    this.form = new FormGroup({
      title: new FormControl(null, {
        validators: [Validators.required, Validators.minLength(3)]
      }),
```

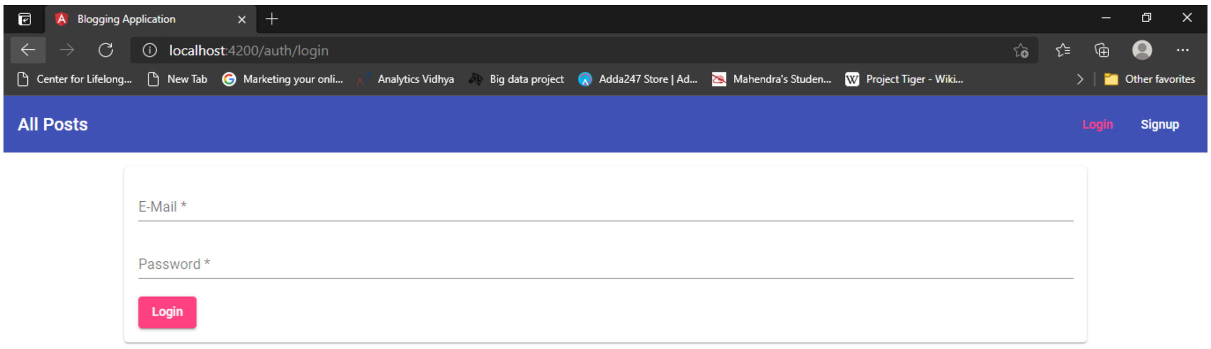


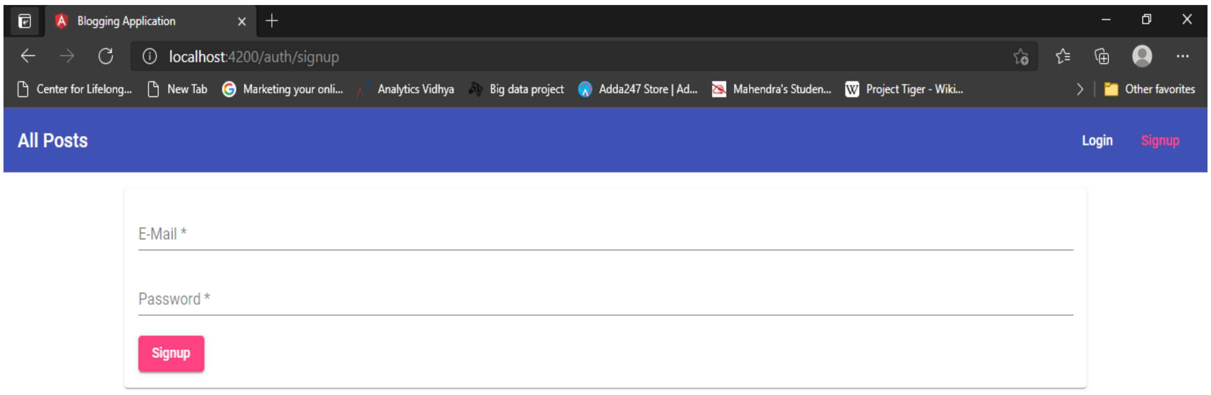
```typescript
import { Component, OnInit, OnDestroy } from "@angular/core";
import { Subscription } from "rxjs";

import { AuthService } from "../auth/auth.service";

@Component({
  selector: "app-header",
  templateUrl: "./header.component.html",
  styleUrls: ["./header.component.css"]
})
export class HeaderComponent implements OnInit, OnDestroy {
  userIsAuthenticated = false;
  private authListenerSubs: Subscription;

  constructor(private authService: AuthService) {}

  ngOnInit() {
    this.userIsAuthenticated = this.authService.getIsAuth();
    this.authListenerSubs = this.authService
      .getAuthStatusListener()
      .subscribe(isAuthenticated => {
        this.userIsAuthenticated = isAuthenticated;
      });
  }

  onLogout() {
    this.authService.logout();
  }

  ngOnDestroy() {
    this.authListenerSubs.unsubscribe();
  }
}
```

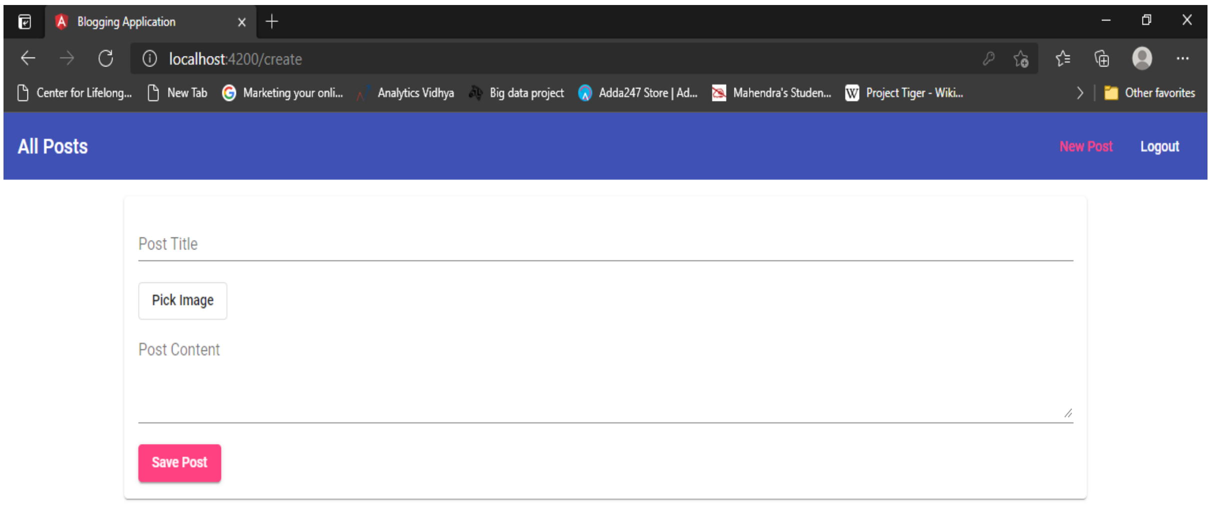

posts.service.ts
src > app > posts > posts.service.ts > PostsService > getPosts > map() callback > posts > postData.posts.map() callback

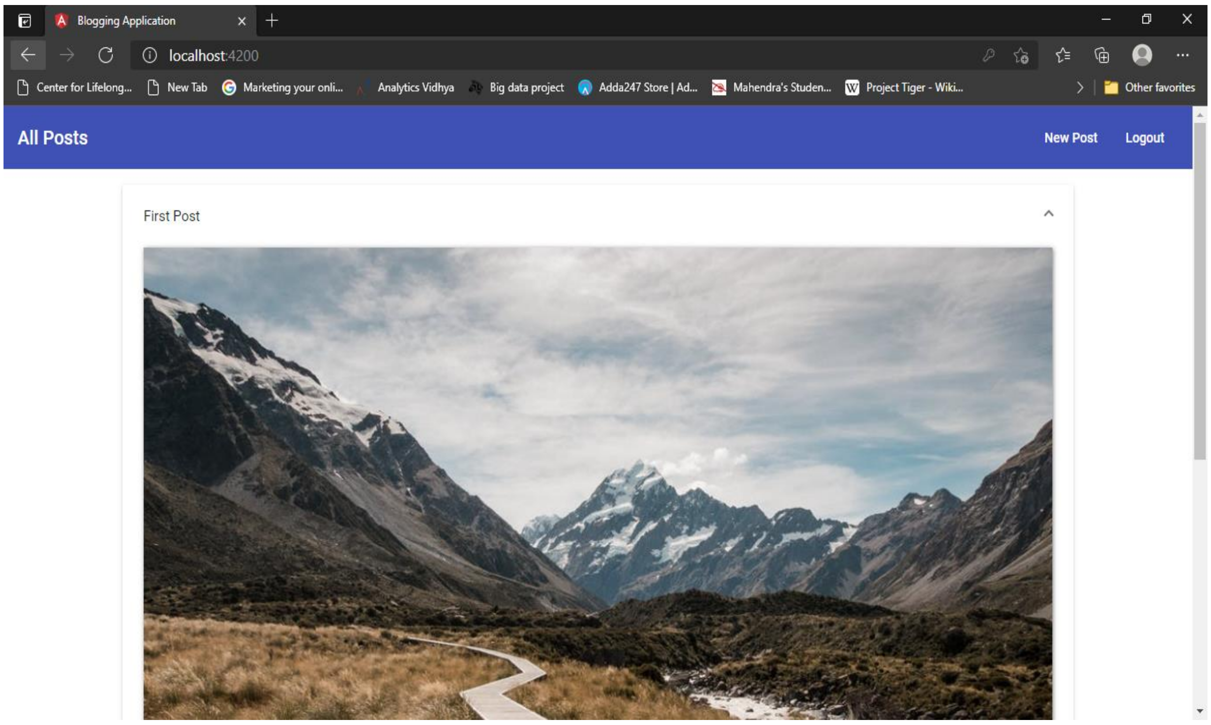
```
const BACKEND_URL = environment.apiUrl + "/posts/";

@Injectable({ providedIn: "root" })
export class PostsService {
  private posts: Post[] = [];
  private postsUpdated = new Subject<{ posts: Post[]; postCount: number }>();

  constructor(private http: HttpClient, private router: Router) {}

  getPosts(postsPerPage: number, currentPage: number) {
    const queryParams = `?pagesize=${postsPerPage}&page=${currentPage}`;
    this.http
      .get<{ message: string; posts: any; maxPosts: number }>(
        BACKEND_URL + queryParams
      )
      .pipe(
        map(postData => {
          return {
            posts: postData.posts.map(post => {
              return {
                title: post.title,
                content: post.content,
                id: post._id,
                imagePath: post.imagePath,
                creator: post.creator
              };
            }),
            maxPosts: postData.maxPosts
          };
        })
      )
      .subscribe(transformedPostData => {
```

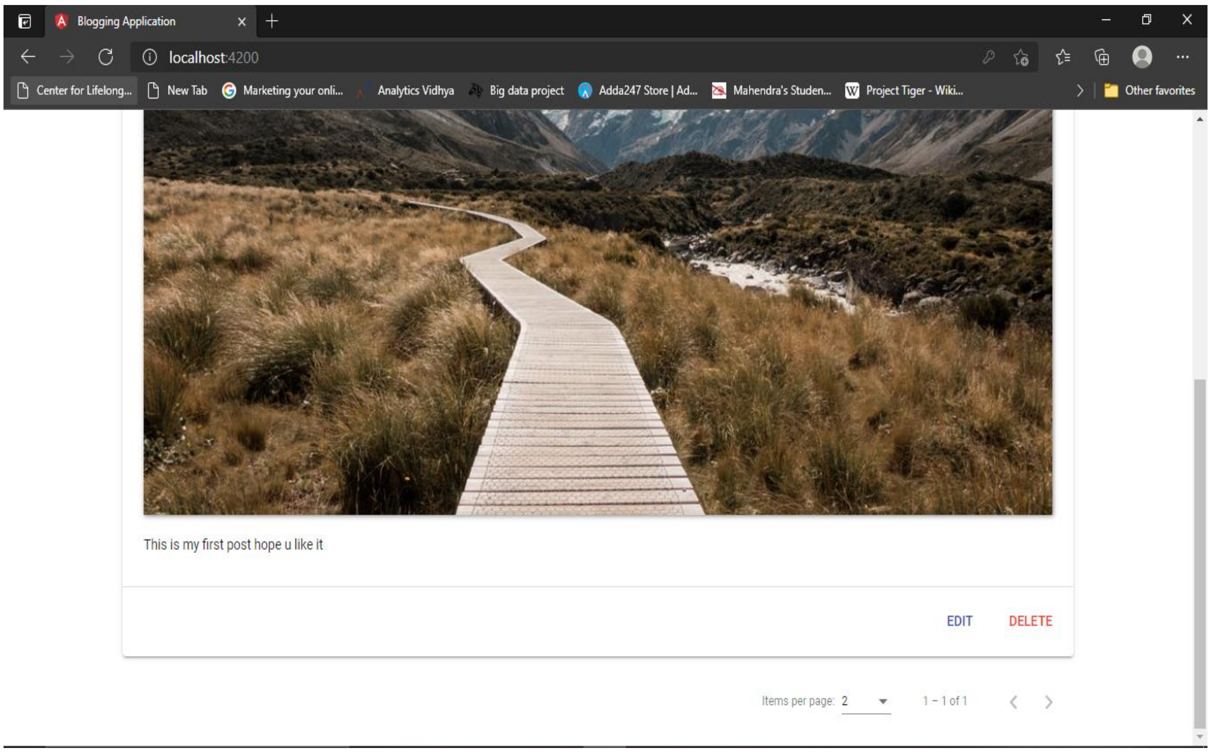
11.2. UI




Blogging Application
localhost:4200/auth/signup
Center for Lifelong... New Tab Marketing your onli... Analytics Vidhya Big data project Adda247 Store | Ad... Mahendra's Studen... Project Tiger - Wiki... Other favorites
All Posts
Login
Signup
E-Mail *
Password *
Signup


Blogging Application
localhost:4200/create
Center for Lifelong... New Tab Marketing your onli... Analytics Vidhya Big data project Adda247 Store | Ad... Mahendra's Studen... Project Tiger - Wiki... Other favorites
All Posts
New Post
Logout
Post Title
Pick Image
Post Content
Save Post





This is my first post hope u like it

11.3 List of restrictions available in the Blogging online system:

• Not designed for blog export to Excel. Based on some criticisms.

• it is not possible to create offline reports for blogs, technical blogs, and new categories.

11.4 Existing system:

In the existing system, the test can only be done manually. In the proposed system, we must use this application to computerize the test.

• Insufficient data security.

• More labor.

• It takes a long time.

• Engaged in a lot of educational work.

• You need to calculate manually.

• Senior officials have no direct role.

11.5 Proposed system of online blogging system

The proposed system allows you to overcome all the limitations of the existing system. The system provides sufficient safety and reduces manual work.

• Data Security.

• Make sure the information is correct.

• Minimize manual data entry.

• The shortest time for different processes.

•Improve efficiency.

11.6 Features of blogging application

Features of the mission Online Blogging System:

• Product and Component based

• Creating & Changing Issues at ease

• Query Issue List to any depth

• User Accounts to manipulate the get admission to and preserve security

• Simple Status & Resolutions

- Multi-stage Priorities & Severities.
- Targets & Milestones for directing the programmers
- Attachments & Additional Comments for extra information
- Robust database back-end
- Various stage of stories to be had with quite a few clear out out criterias
- Accuracy in work.
- Easy & speedy retrieval of information.

12. Conclusion

This project is only a major goal is to venture to satisfy the need to manage their project work. With the help of ***MEAN*** stack that we have used to make it possible for the user. This will prove to be a powerful web application satisfying all the requirements of the users. The objective of software is to show that the framework is capable of making the user satisfying their needs, fast and reliable.

The point of this paper is to examine the need Advanced publishing content to a blog which is a sort of site that permits parcel of passages either made by the individuals from the association. A blog is an incredible path for an individual to figure out how to make their own blog. Blogger appreciates the distinction of being one of the most mainstream contributing to a blog stages particularly for the association. This high-level publishing content to a blog has made the website well known on the grounds that it is free and offers anybody the capacity to deal with their own blog without the weight of details in facilitating the equivalent.

Few concluding points

- A description of the background and context of the project and its relation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We define the problem on which we are working in the project
- We Describe the requirement specification of the system and the actions that can be done on these things
- We understand the problem domain and produce a model of the system which describes operations that can be performed on the system.
- We included features and operation in detail, including screen layout.
- We designed user interface and security issue to system

13. References

• Online Course References: o MEAN Stack Course (Udemy) (https://www.udemy.com/course/angular-2-and-nodejs-the-practical-guide/)

• NPM Documentation: o https://www.npmjs.com/

• GeeksForGeeks https://www.geeksforgeeks.org/introduction-to-mean-stack/

• Faculty Guidelines: o Mr. Akash Kumar Choudhary

- **Getting MEAN with Mongo, Express, Anguar and Node** by Simon Holmes
- **Pro MEAN Stack Development** by Elad Elrom