Content Management System

Team Members :

- Gandhar Anirudha Khandagale (G01394026)
- Siddhesh Santosh Sawant (G01422071)

Project Description:

The Content Management System (CMS) is a web application that allows users to create, manage, and publish digital content, such as articles, blog posts, and multimedia. This project aims to develop a CMS using Node.js, Express, EJS, and MongoDB, providing a flexible and efficient platform for content creation and management. The CMS developed using Node.js, Express, and MongoDB offers a comprehensive solution for content creation and management, with a focus on user experience, performance, and scalability.

Technologies Used:

- **Frontend**: CSS,JavaScript
- **Backend**: NodeJS, ExpressJS
- **Database**: MongoDB

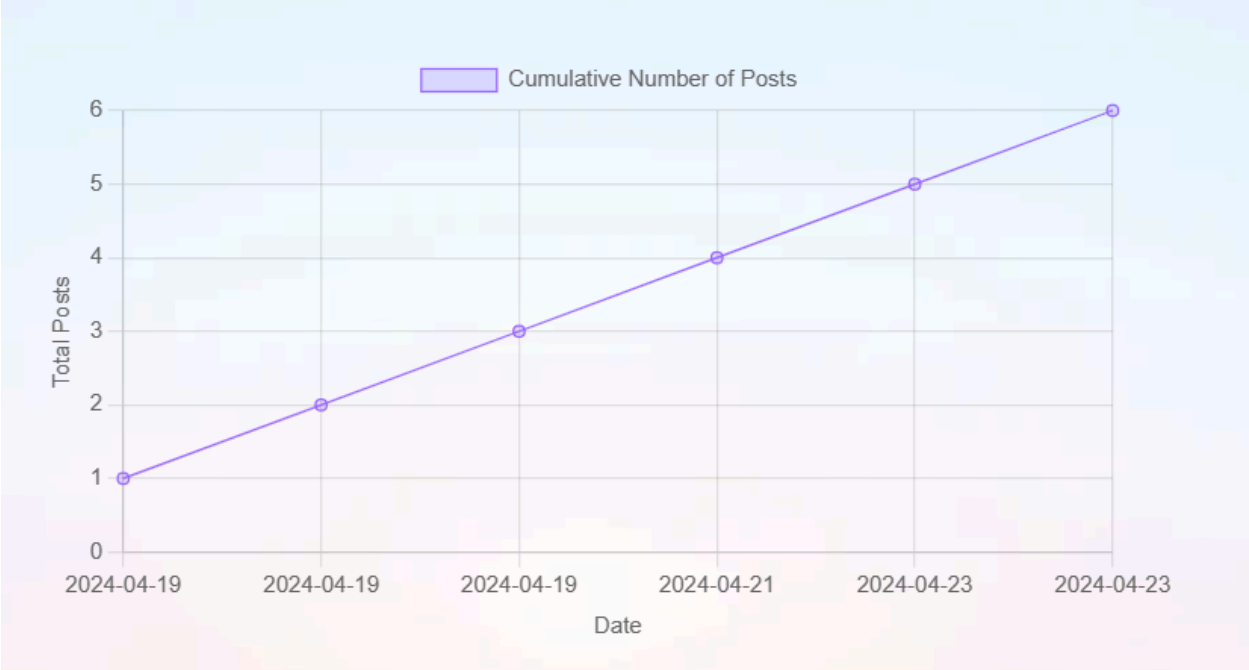
Collections:

- **Users:**
  _id, username, password
- **Posts :**
  _id, title, body, createdAt, updatedAt
- **Sessions:**
  _id, session, expires
- **Comments:**
  _id, postId, userId, text, createdAt, updatedAt
- **Favorites:**
  _id, postID, userID, createdAt
- **Tags:**
  _id, postId,name

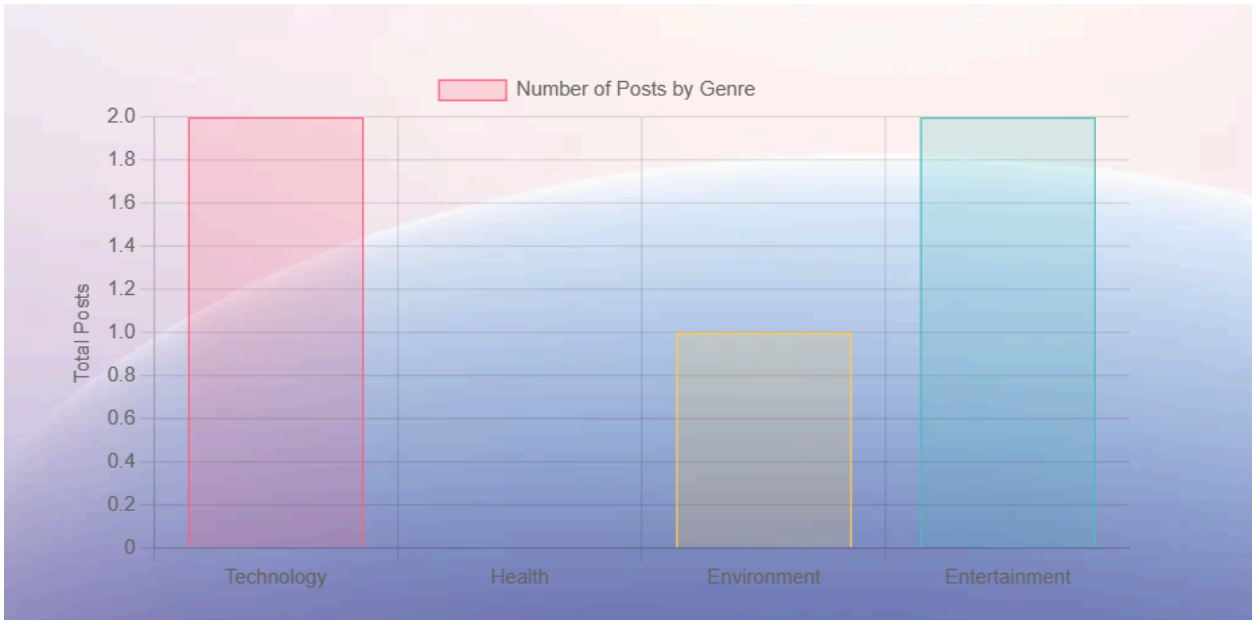
Operations:

Perform basic CRUD (Create, Read, Update, and Delete) operations.

Visualizations:



Visualization showing Date wise cumulative Number of Posts added with the help of Line chart.



Visualization showing Number of total posts by Genre with the help of Bar Chart.

Functionalities:

- Responsive Design: The application is built with a responsive design, meaning it adjusts to different screen sizes, providing a user-friendly experience on both desktops and mobile devices.
- CRUD Operations: The application supports Create, Read, Update, and Delete operations for blog posts. Users can add new posts, view them on the blog, edit existing posts, or delete them.
- Database Integration: It uses MongoDB for data storage, which includes storing the details of blog posts.
- User Authentication: The application includes functionality for user authentication. Users can sign up, log in, and log out.
- Title Generator Using Machine Learning: When adding content to a blog post, the application includes an innovative title generator that utilizes a machine learning method utilizing NLP and Hugging Face. This feature automatically generates a relatable and engaging title based on the content provided, enhancing the user experience and making the blog posts more appealing.
- Middleware for Session Handling: It uses middleware to manage sessions, ensuring that users remain logged in as they navigate through different parts of the blog.
- Environment Variables: It uses an .env file to securely manage environment variables like MongoDB URI and JWT secrets.

- Layout and Design: The front-end uses EJS templates for rendering pages. It also includes a link to the Figma design files used for the blog's layout.
- Development Tools: For local development, the project utilizes Nodemon to automatically restart the server when files change.
- Security Features: The application handles security through JSON Web Tokens (JWT) and secure handling of database credentials.

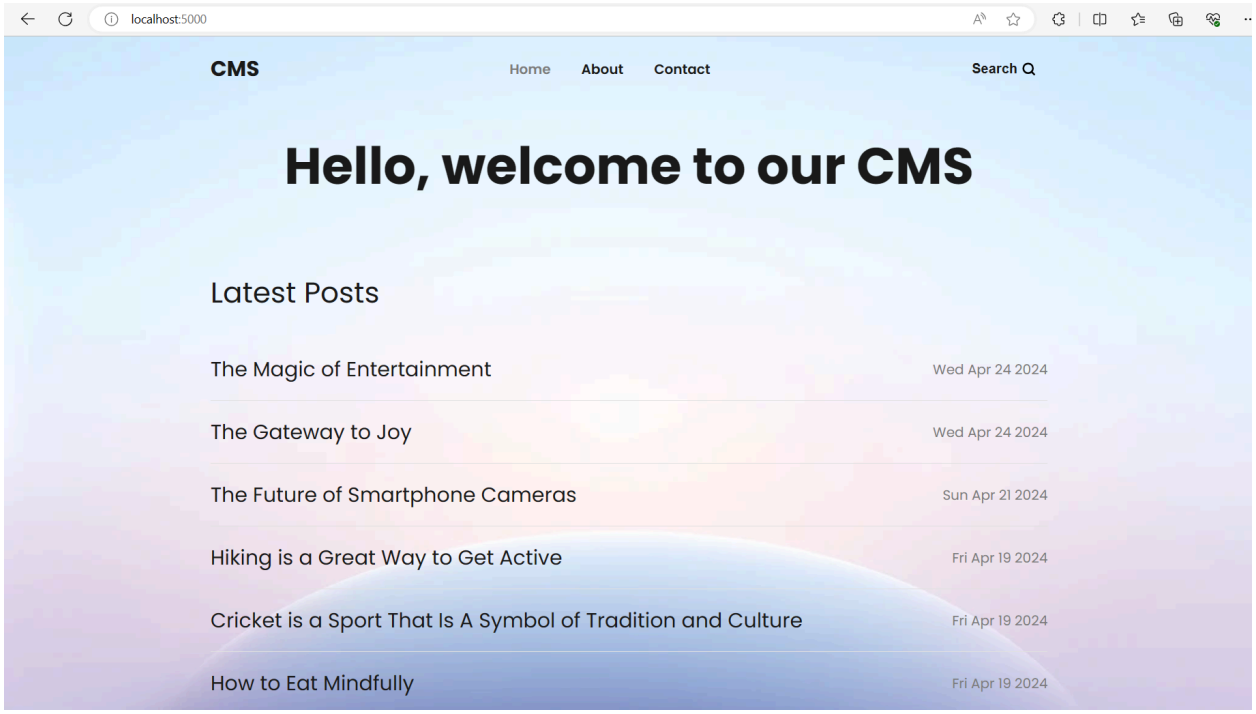
Complex Query:

Pagination of Posts: It fetches a paginated list of all blog posts, sorted by their creation date in descending order. The pagination is controlled by a page query parameter, defaulting to the first page with 10 posts per page.

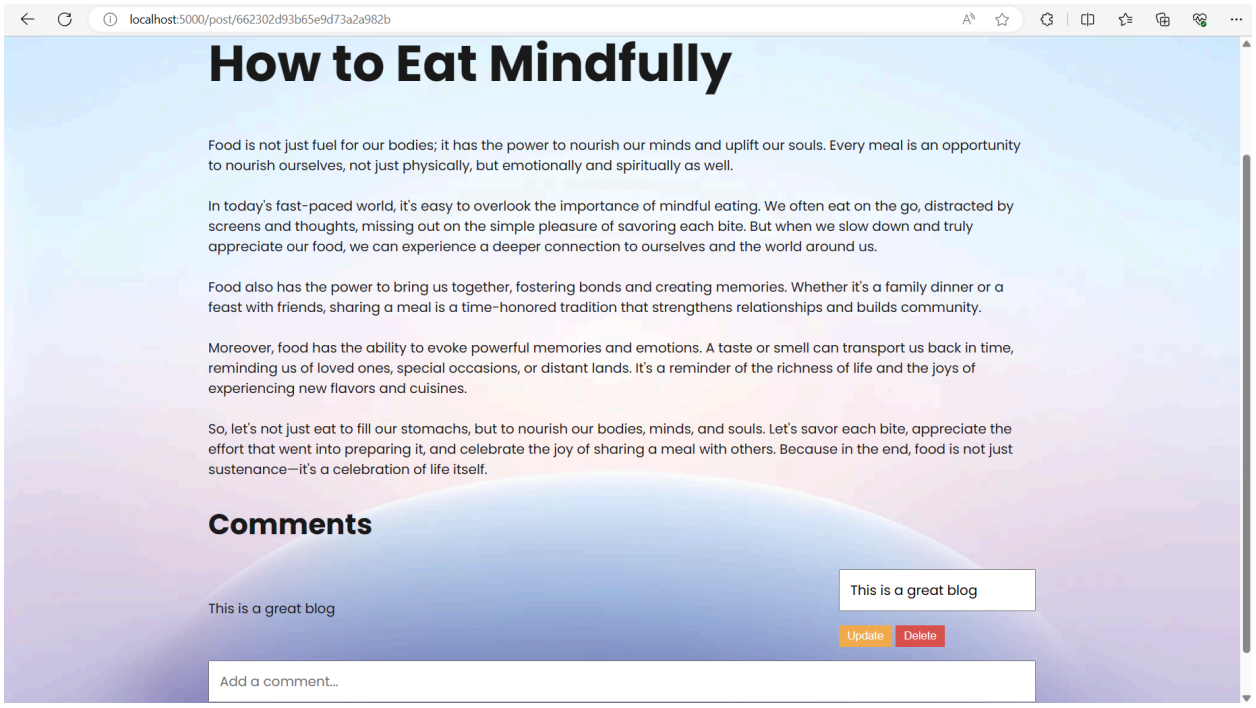
Top Commented Posts: It constructs an aggregation pipeline to identify the top three most commented posts. This involves:
Joining the posts collection with the comments collection to map comments to their respective posts.
Calculating the number of comments for each post.
Sorting these posts by the count of comments in descending order.
Limiting the output to only the top three posts.

Screenshots of the UI:

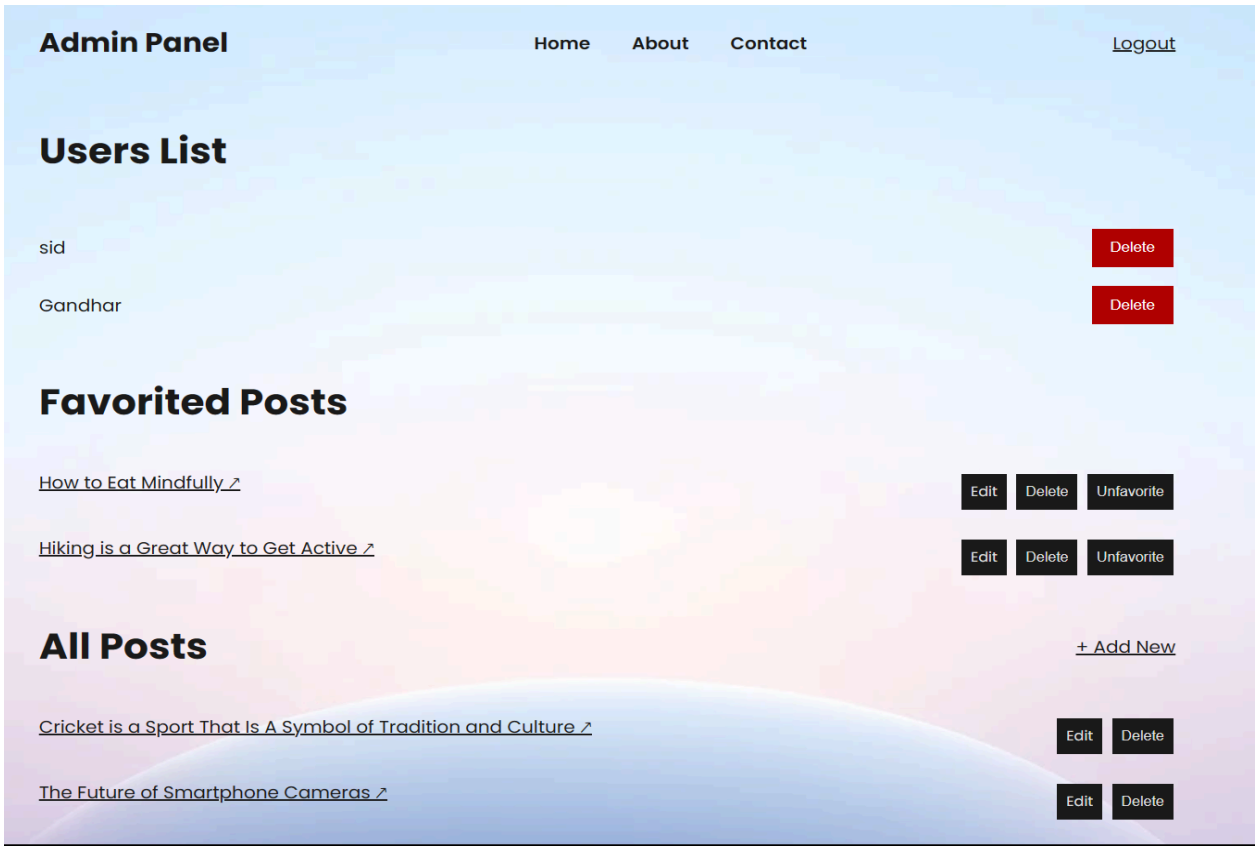
Home Page:



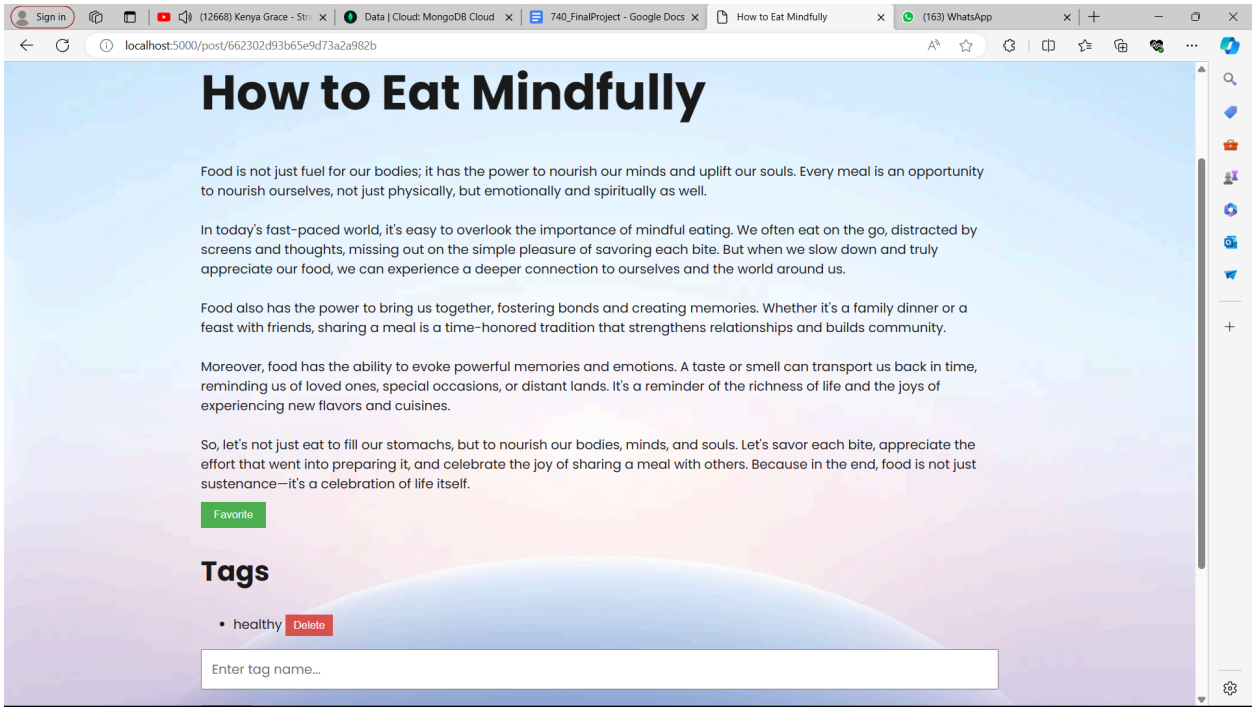
Comment & Post:



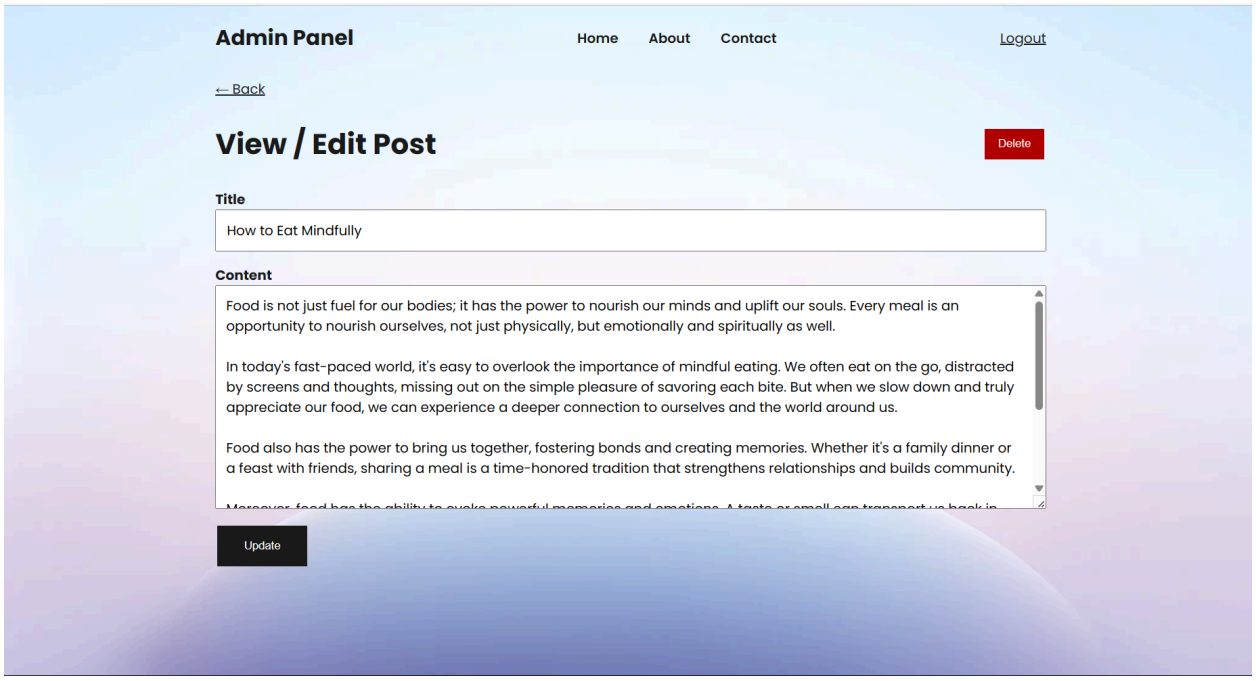
List of Users, Posts and Favorite Posts:



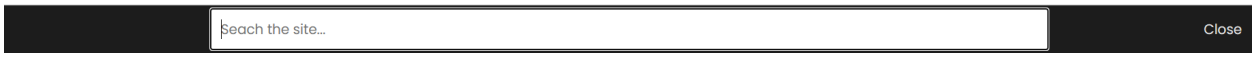
Post, Favorites and Tags:



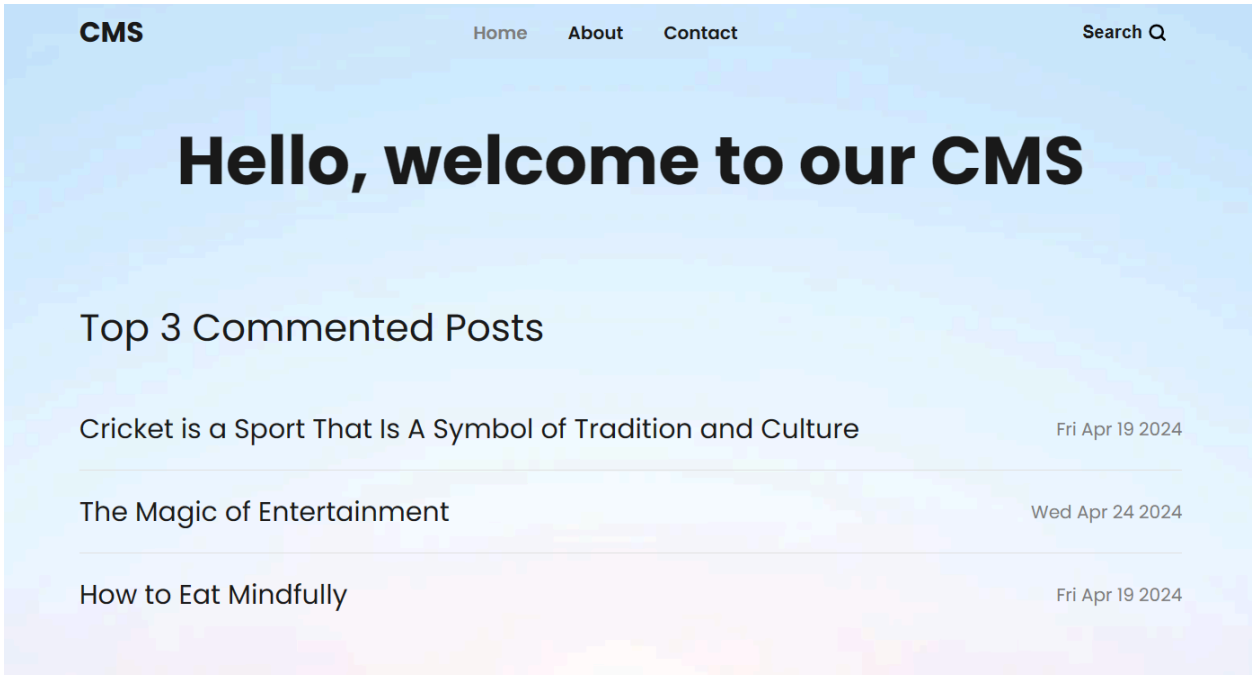
View / Edit Post:



Search Bar:

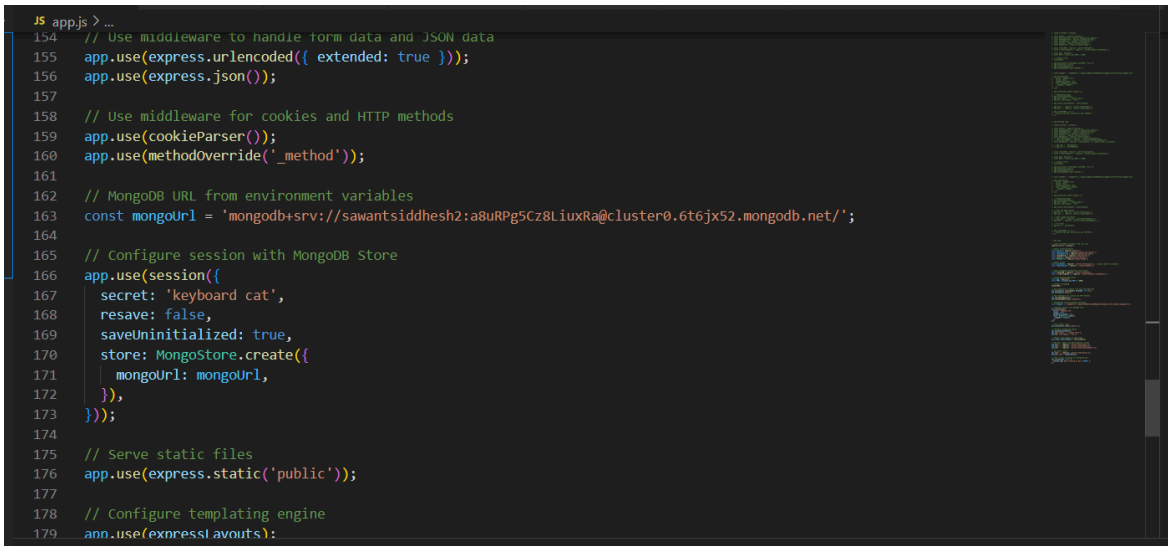


Top 3 Commented Post:

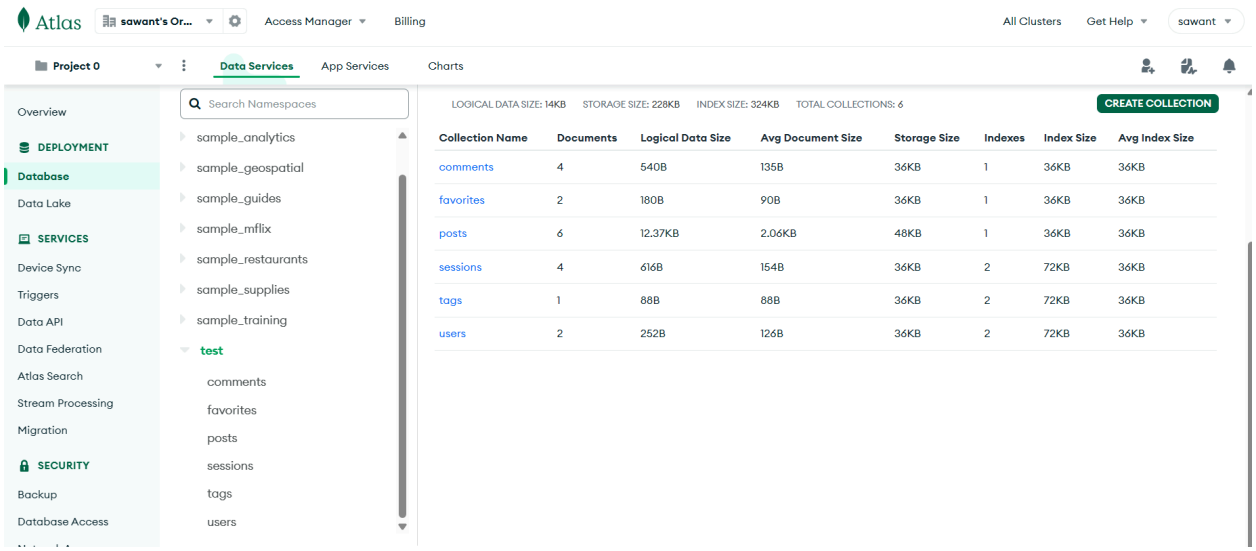


Data Base

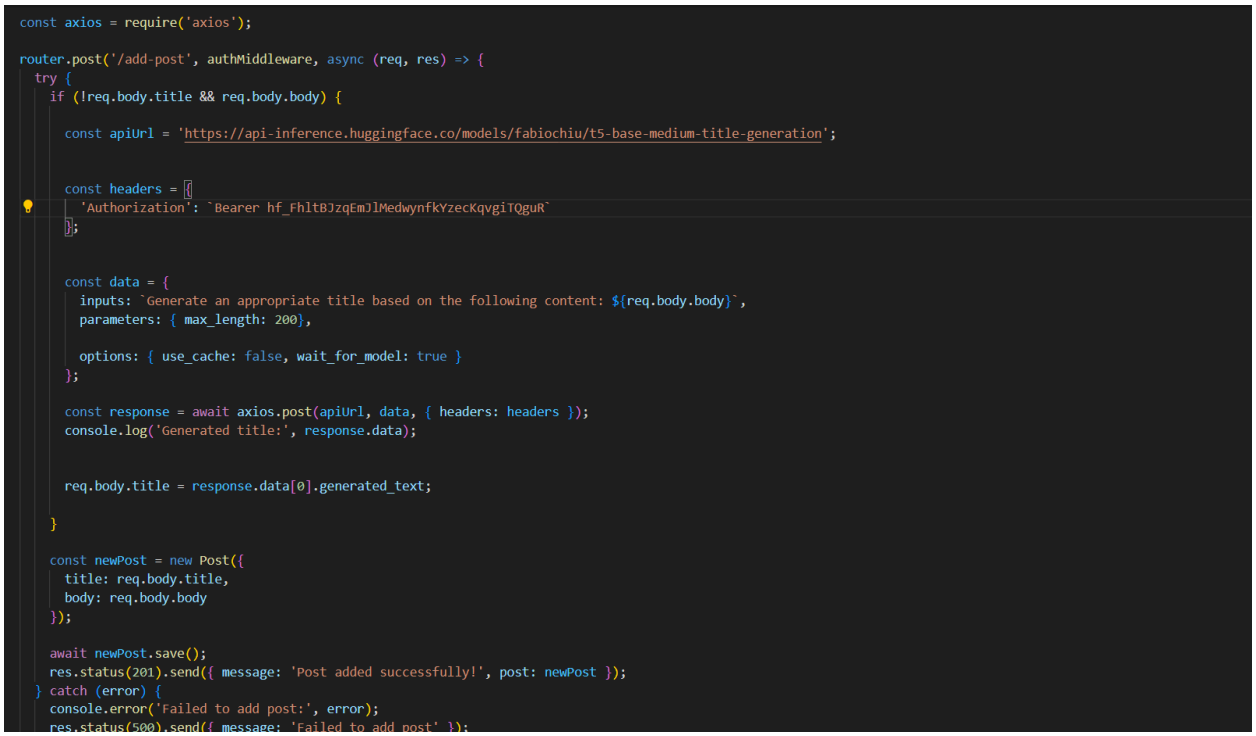
MongoDB:



```
JS app.js > ...
154  // Use middleware to handle form data and JSON data
155  app.use(express.urlencoded({ extended: true }));
156  app.use(express.json());
157
158  // Use middleware for cookies and HTTP methods
159  app.use(cookieParser());
160  app.use(methodOverride('_method'));
161
162  // MongoDB URL from environment variables
163  const mongoUrl = 'mongodb+srv://sawantsiddhesh2:a8uRPg5Cz8LiuxRa@cluster0.6t6jx52.mongodb.net/';
164
165  // Configure session with MongoDB Store
166  app.use(session({
167    secret: 'keyboard cat',
168    resave: false,
169    saveUninitialized: true,
170    store: MongoStore.create({
171      mongoUrl: mongoUrl,
172    }),
173  }));
174
175  // Serve static files
176  app.use(express.static('public'));
177
178  // Configure templating engine
179  app.use(expressLayouts);
```



Machine learning:



```
const axios = require('axios');

router.post('/add-post', authMiddleware, async (req, res) => {
  try {
    if (!req.body.title && req.body.body) {

      const apiUrl = 'https://api-inference.huggingface.co/models/fabiochiu/t5-base-medium-title-generation';


      const headers = {
        'Authorization': `Bearer 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`
      };


      const data = {
        inputs: `Generate an appropriate title based on the following content: ${req.body.body}`,
        parameters: { max_length: 200},

        options: { use_cache: false, wait_for_model: true }
      };

      const response = await axios.post(apiUrl, data, { headers: headers });
      console.log('Generated title:', response.data);


      req.body.title = response.data[0].generated_text;

    }

    const newPost = new Post({
      title: req.body.title,
      body: req.body.body
    });

    await newPost.save();
    res.status(201).send({ message: 'Post added successfully!', post: newPost });
  } catch (error) {
    console.error('Failed to add post:', error);
    res.status(500).send({ message: 'Failed to add post' });
```

- For the machine learning method, we utilized Natural Language Processing (NLP) techniques in conjunction with the Hugging Face (https://huggingface.co/)library.
- This allowed us to implement a title generator feature that automatically generates engaging titles for blog posts.
- When a user adds a new post, the NLP model analyzes the content and generates a suitable title based on the post's content.
- The Hugging Face library provides access to pre-trained models and facilitates the implementation of advanced NLP tasks, enhancing the accuracy and effectiveness of the title generation process.

Team member accomplishments:

- We were able to create a web application with the use of a NoSQL database i.e MongoDB and additional technologies like ExpressJS, ReactJS, and NodeJS.
- Gandhar contributed to the backend development using NodeJS and ExpressJS, integrating MongoDB for the database, and implementing a machine learning method for the title generator utilizing NLP and Hugging Face. Siddhesh focused on frontend development, working with HTML, CSS, JavaScript, and creating visualizations.