DISSERTATION PROJECT

ON

"BLOG WEB APPLICATION WITH ADMIN DASHBOARD"

IN THE PARTIAL FULFILLMENT OF REQUIREMENT OF

MASTER OF COMPUTER APPLICATION (MCA)

UNDER THE GUIDANCE OF

DR. SANGITA PUNDE

SUBMITTED BY

MR. AAYUSH VIPIN PARULKAR

ENROLLMENT NO.: MITU22MCAD0001
BATCH- 2022-24

SUBMITTED TO

MIT COLLEGE OF MANAGEMENT, PUNE

2022-2024

Certificate from Company

MIT College of Management, Pune

This is to certify that, Mr. Aayush Vipin Parulkar has submitted a Project Report on Blog Web Application MIT – ADT University, Pune for the partial fulfillment of Master in Computer Application (Data Science) (MCA) (2022-24 Batch)

We further certify that to the best of our knowledge and belief, the matter presented in this project has not been submitted to any Degree or Diploma course.

PRN No: MITU22MCAD0001

Dr. Sangita Phunde	Dr. Vijay Gondane	Prof. Dr. Sunita Karad
HOD, MCA MITCOM	**PG HEAD**	**Executive Director-**

External Examiner **Sign of Examiners**:

1. _____ _____

Internal Examiner

2. _____ _____

DECLARATION

I hereby declare that the project work entitled "Blog Web Application with Admin Dashboard" submitted to the MIT – ADT University, Pune, is a record of an original work done by me under the guidance of Dr. Sangita Phunde, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Master of Computer Application. The project work in this report has not been submitted to any other University or Institute for the award of any degree or diploma. This is my own and original work.

Date: Signature of the Candidate

CERTIFICATE OF THE GUIDE

This is to certify that, Mr. Aayush Vipin Parulkar of MCA Course (Data Science)

have/has successfully completed his Project Work Titled "Blog Web Application with Admin Dashboard", under my guidance during the Academic Year 2022-2024.

Date:

Prof: Dr. Sangita Phunde

Project Guide name & Signature

ACKNOWLEDGEMENT

I would like to convey my sincere gratitude to all those who have been instrumental in the development of the project.

I am thankful to the organization ***Interns Planet*** for giving me an opportunity to work with them. Sincere thanks are uttered towards project guide ***Mr. Gulshan Joshi*** of ***Interns Planet*** towards the motivation, technical support and inspiration provided to me.

I am greatly thankful to Honorable **Dr. Prof. Sunita Karad**, Executive Director of MITCOM for all her timely support.

I express my gratitude to the PG Head **Dr. Vijaya Gondane** & Head of MCA Department **Dr. Sangita Phunde** who helped me in my extreme solutions.

I am also thankful to ***Dr. Sangita Phunde,*** my internal project guide for his/her invaluable guidance, help and great support during the project work.

I am greatly thankful to the staff of MITCOM, Pune for helping me through the entire course.

Student Name & Signature:

Date:

Place: MITCOM, MIT-ADT University, Pune

CHAPTER 1:

INTRODUCTION:

1.1 COMPANY PROFILE

Company Name: Interns Planet

Founded: 2019

Industry: Information Technology, Sales & Services

Company Description: Interns planet is an ISO 9001:2015, IAF (International Accreditation Forum) & EGAC (Egyptians Accreditation Council) certified company with a registered office at Indore. Interns Planet believes in getting ready quality manpower for esteem organizations for their professional growth. Interns Planet is a specialist organization committed to providing training and internship to students, work experience and cultural exchange in the corporate business environment..

Key Services:

- Web Design & Development (JavaScript, HTML, CSS, React.js)
- Sales and Services
- Digital Marketing (SEO, Email Marketing)

Company Strengths:

- Experienced team of developers and designers
- Agile development methodologies
- Competitive pricing
- 24/7 support
- Focus on quality and customer satisfaction

1.2 ABSTRACT

In today's digital era, blog applications have become essential platforms for content creation, sharing, and engagement. This abstract introduces a comprehensive blog application equipped with an administrative dashboard, designed to empower users with robust content management capabilities. The blog application facilitates seamless content creation, publication, and interaction, catering to both content creators and readers.

This blog application offers a user-friendly interface where users can register, create personalized profiles, and access a diverse range of content. The administrative dashboard enhances the user experience by providing administrators with advanced tools for content moderation, user management, and analytics. Through the dashboard, administrators can oversee user activity, manage posts, monitor engagement metrics, and ensure the platform's integrity.

Key features of the blog application include user authentication, customizable profiles, rich text editing for content creation, comment management, social sharing integrations, and responsive design for optimal accessibility across devices. The administrative dashboard complements these features with intuitive controls, empowering administrators to maintain a vibrant and engaging blogging ecosystem.

By integrating modern web technologies such as React.js, Next.js, MongoDB, and Express.js within a comprehensive MERN stack architecture, this blog application achieves scalability, performance, and security. The use of Next.js ensures server-side rendering for enhanced SEO and performance, while MongoDB provides a flexible and scalable database solution.

The proposed blog application and its administrative dashboard aim to elevate the blogging experience for both creators and consumers. By combining intuitive user interfaces with robust backend functionalities, the application encourages content creation, fosters community engagement, and supports the seamless management of a dynamic blogging platform in the digital age..

1.3 EXISTING SYSTEM:

The existing landscape of blog applications encompasses a diverse array of platforms tailored to different user needs and preferences. Traditional blog platforms like WordPress, Blogger, and Medium have long dominated the space, offering robust content management systems with varying levels of customization and user interaction. These platforms provide users with tools to create, edit, and publish content, along with features for managing comments, analyzing traffic, and customizing layouts.

However, this traditional approach presents several drawbacks:

1. **Limited Accessibility:** Many blog platforms are hosted on centralized servers, which may lead to slower access or even regional restrictions in certain areas due to content delivery networks (CDNs) or hosting limitations
2. **Time-Consuming:** Older or less advanced devices may struggle to render complex blog layouts or interactive features, reducing accessibility for users relying on such devices.
3. **Writer Frustration:** Users in regions with limited internet connectivity or slower speeds may experience difficulties in accessing and interacting with blog content. This can impact user engagement and reach.

1.4 NEED FOR THE SYSTEM

The need for a blog application with an admin dashboard arises from several key requirements and objectives:

1. **Content Management and Publication:** Users, including administrators, require a platform to create, manage, and publish content effectively. An admin dashboard allows administrators to oversee content creation, moderation, and publishing workflows efficiently.
2. **User Engagement and Interaction:** A blog application facilitates user engagement through features like comments, likes, and sharing. An admin dashboard enables administrators to monitor and moderate user interactions to maintain a positive and engaging environment.
3. **Customization and Branding:** Businesses and individuals need to customize their blog's appearance, layout, and branding to reflect their unique identity. An admin dashboard provides tools for customizing themes, layouts, colors, and other visual elements.
4. **Analytics and Insights:** Administrators require access to analytics and insights to understand user behavior, traffic patterns, and content performance. An admin dashboard integrates analytics tools to provide actionable data for optimizing content strategies.
5. **Security and Access Control:** Blog applications need robust security measures to protect user data, prevent unauthorized access, and mitigate potential threats like spam or malicious content. An admin dashboard includes tools for managing user roles, permissions, and security settings.
6. **SEO and Content Optimization:** To improve visibility and reach, blogs must be optimized for search engines (SEO) and social sharing. An admin dashboard offers SEO tools and features to optimize content metadata, URLs, and other SEO elements.
7. **Scalability and Growth:** As the blog audience grows, the platform needs to scale efficiently to handle increased traffic and content volume. An admin dashboard assists in managing scalability, performance optimizations, and infrastructure scaling.

1.5 SCOPE OF SYSTEM

User Features:

The scope of a blog application with an admin dashboard encompasses a range of functionalities and features tailored to meet the needs of content creators, administrators, and users. The scope includes:

1. **User Management:**
    - User registration and authentication.
    - User profile management (e.g., editing profiles, preferences).

2. **Content Management:**
    - Content creation (e.g., writing articles, uploading media).
    - Content moderation (e.g., approving, editing, or deleting user-generated content).
    - Categorization and tagging of content for organization and navigation.

3. **Admin Dashboard:**
    - Centralized interface for administrators to manage the blog platform.
    - Tools for content moderation, user management, and analytics.
    - Access control and permissions management.

4. **Frontend Features:**
    - Responsive and user-friendly interface for all devices.
    - Rich text editor for content creation and formatting.
    - Commenting system with moderation capabilities.

5. **SEO and Analytics:**
    - Integration with SEO tools for optimizing content visibility.
    - Analytics dashboard to track user engagement, traffic, and performance metrics.

6. **Social Sharing and Engagement:**
    - Social media integration for sharing blog posts.
    - Features to encourage user engagement (e.g., likes, shares, follow buttons).

7. **Customization and Branding:**
    - Theme customization options (e.g., colors, fonts, layout).
    - Branding elements (e.g., logo, favicon) configurable from the admin dashboard.

8. **Security and Privacy:**
    - Robust security measures to protect user data and prevent unauthorized access.
    - Compliance with data protection regulations (e.g., GDPR, CCPA).

9. **Scalability and Performance:**
    - Architecture designed for scalability to handle increasing traffic and content volume.
    - Performance optimizations (e.g., caching, lazy loading) for improved user experience.

10. **Community Building:**
    - Features to foster community engagement (e.g., forums, events, user groups).
    - Collaboration tools for users to contribute collaboratively (e.g., guest posts, co-authoring).

11. **Integration and Extensibility:**
    - APIs and integrations with third-party services (e.g., email marketing, analytics).
    - Plugin architecture or extensibility for adding custom features and functionalities.

The scope of the system aims to provide a comprehensive platform that empowers content creators to publish and manage engaging content, while administrators have the tools to oversee and optimize the blog's performance, security, and user experience. The system's flexibility and scalability ensure it can adapt to evolving requirements and support the growth of the blog community effectively**.**

1.6 OPERATING ENVIRONMENT

Client Side Environment:

Hardware Requirement:

- **Processor** : Dual Core
- **Memory(RAM)** : 2GB and Above
- **Hard disk(Storage)** : 20GB and Above

Software Requirement:

- **Browser** : Google Chrome, Mozilla Firefox, Opera, Edge, Brave, Safari, Etc.

Server Side Environment

Hardware Requirement:

- **Processor** : Dual Core or i3
- **Memory(RAM)** : 8GB and Above
- **Hard disk(Storage)** : 20GB and Above

Software Requirement:

- **Operating System** : Windows 7 and Above
- **Database** : MySql 5.7 and above
- **Server** : WAMP or XAMPP server
- **Browser** : Google Chrome, Mozilla Firefox, Opera, Edge, Brave, Safari, Etc.

1.7 TECHNOLOGIES USED

1. **HTML (Hypertext Markup Language):**

   HTML is the cornerstone of web development, providing a standardized markup language for structuring and presenting content on web pages. It utilizes tags to define elements like headings, paragraphs, lists, links, images, and forms, thereby establishing the structure of a webpage. With HTML5, semantic elements such as <header>, <footer>, <nav>, and <article> are introduced, improving content structure and accessibility. HTML enables the creation of accessible, search engine-friendly, and responsive web pages that seamlessly render across various browsers and devices.

2. **Cascading style sheets are used.**

   The visual presentation and layout of HTML elements on web pages are governed by the fundamental styling language of CSS.

   It allows developers to specify styles for diverse aspects of a webpage, including colors, fonts, spacing, borders, background, and positioning.

   Advanced features like animations, transitions, gradients, shadows, and flexbox/grid layout can be used in the creation of visually appealing and dynamic user interfaces.

   CSS promotes modular design, code reusability, and maintainability, facilitating the development of consistent and aesthetically pleasing web experiences by Separating content from presentation.

3. **JavaScript:**

   JavaScript is a flexible programming language that provides interactivity, conduct, and functionality to web pages.

   It enables developers to create dynamic consumer interfaces, take care of person occasions, manipulate the DOM (document item model), make asynchronous requests (Ajax), and implement consumer-facet validation.

JavaScript has advanced with features like ES6 (ECMAScript 2015) and beyond, helping modern programming paradigms which includes arrow features, instructions, modules, guarantees, and async/await syntax, thereby enhancing code clarity and maintainability.

Frameworks and libraries like React.js, Vue.js, and AngularJS expedite complex internet application improvement via imparting reusable additives, kingdom management, and routing competencies.

4. **React.js:**

   React.js is a popular JavaScript library for building user interfaces, particularly single-page applications (SPAs). It allows you to create reusable UI components that efficiently render and update based on data changes. React's component-based architecture promotes code reusability, modularity, and easier state management..

5. **Bootstrap:**

   Bootstrap stands as a most excellent front-stop framework for crafting responsive, cell-first web sites and applications.

   advanced by means of Twitter, Bootstrap includes a wealthy set of CSS training, additives, and JavaScript plugins for fast prototyping and development.

   Its grid device, typography, paperwork, buttons, navigation, and modal components streamline the advent of regular and visually appealing UI layouts throughout devices and screen sizes.

   Bootstrap's extensibility and customization options, in conjunction with complete documentation and network help, make it a favored choice for developers of varying expertise levels.

6. **Redux:**

   Redux is a predictable state container for JavaScript apps, commonly used with React.js. It centralizes your application's state in a single store, making it easier to manage and update data across components. Redux facilitates a unidirectional data flow and provides tools for managing complex application states.

7. **MongoDB:**

   MongoDB is a popular NoSQL database that stores data in flexible, JSON-like documents. It is well-suited for handling diverse and unstructured data typical of web applications, such as user profiles, blog posts, and comments. MongoDB's scalability and flexibility make it ideal for managing large volumes of data in a modern web app.

8. **Ajax (Asynchronous JavaScript and XML):**

   Ajax revolutionizes internet improvement by using allowing asynchronous conversation between the patron-side and server-facet components of internet programs.

   It facilitates seamless statistics retrieval and submission with out necessitating a complete page reload, thereby improving user revel in and responsiveness

   normally utilized for functions like form submissions, actual-time facts updates, car-complete search fields, endless scrolling, and interactive maps, Ajax elevates the usability and performance of internet programs.

   With current internet APIs like Fetch API and XMLHttpRequest degree 2, Ajax maintains to play a pivotal function in building interactive and information-pushed internet studies.

9. **Firebase:**

   Firebase is a comprehensive platform for building web and mobile applications developed by Google. It offers a suite of backend services, including authentication (Firebase Authentication), real-time database (Firestore), cloud hosting, analytics, and cloud functions. Firebase simplifies backend development tasks and provides scalable infrastructure for your blog app.

10. **Tailwind CSS:**

    Tailwind CSS is a utility-first CSS framework that offers a set of predefined utility classes to style your app's components. Instead of writing custom CSS, developers can use Tailwind's classes directly in HTML templates to apply styles such as margins, paddings, colors, and responsive layouts. Tailwind streamlines the styling process and encourages consistency.

CHAPTER 2:
PROPOSED SYSTEM

2.1 FEASIBILITY STUDY

1. **Technical Feasibility:**
    - The technical feasibility of an online hotel booking website involves an assessment of available technology stacks and resources necessary for its development. This encompasses frontend technologies like HTML, CSS, JavaScript, Next.js and backend technologies such as Node.js.
    - The app uses the video and audio API of the Stream platform
    - The website's functionality must support user authentication, email verification, and password management to ensure secure access.
    - The SignIn and SignUp functionality is secured by the MongoDB Atlas.
2. **Financial Feasibility:**
    - Financial feasibility entails an evaluation of costs associated with website development, hosting, maintenance, and updates.
    - Initial investment costs for development and implementation, including software licenses, server hosting, and need estimation.
    - Revenue projections consider factors such as booking volume, commissions from the organizations and potential advertising revenue.
    - Conducting a comprehensive cost-benefit analysis aids in determining the return on investment (ROI) and overall project viability.
3. **Operational Feasibility:**
    - Ensure the app is easy to use and intuitive, requiring minimal training for users to schedule, join, and manage video conferences.

- Ensure compatibility with a variety of devices and operating systems, including desktops, laptops, tablets, and smartphones, to maximize accessibility.
- Enable seamless integration with existing collaboration tools, email platforms, and calendar systems to streamline scheduling and joining meetings.
- Risk analysis identifies potential operational challenges, such as server downtime, security breaches, or technical glitches, with mitigation strategies developed accordingly.

4. **Schedule Feasibility:**
    - Schedule feasibility evaluates whether the project can be completed within the specified timeframe.
    - A detailed project timeline, including milestones for development, testing, and deployment, is essential.
    - Identification of critical dependencies, such as third-party integrations and regulatory approvals, ensures timely project completion.
    - Maintaining schedule flexibility accommodates unforeseen delays or changes in project scope.
5. **Legal and Regulatory Feasibility:**
    - Legal and regulatory feasibility involves assessing compliance with laws, regulations, and industry standards governing online transactions, data privacy, and consumer protection.
    - Implementation of measures to ensure user data security and confidentiality, including encryption protocols and compliance with GDPR or CCPA regulations, is essential.
    - Drafting terms of service and privacy policies outlines user rights and responsibilities, liability, and dispute resolution mechanisms.

2.2 OBJECTIVE OF PROPOSED SYSTEM

The objective of the proposed system, a blog app with an admin dashboard, is to provide a comprehensive platform for content creation, management, and publication. The primary objectives include:

1. **Content Creation and Management:**
    - Enable users to create, edit, and manage blog posts with rich text formatting, images, and multimedia content.
    - Implement features for organizing content into categories, tags, or topics to facilitate navigation and content discovery.

2. **User Authentication and Authorization:**
    - Implement secure user authentication mechanisms to allow users to sign up, sign in, and manage their profiles.
    - Provide role-based access control to differentiate between regular users and administrators with elevated privileges.

3. **Admin Dashboard Functionality:**
    - Empower administrators with tools to manage user accounts, review and moderate user-generated content, and oversee site operations.
    - Offer analytics and reporting features to monitor site performance, user engagement, and content popularity.

4. **Interactive User Experience:**
    - Implement features like comments, likes, and shares to encourage user engagement and community interaction.
    - Enable users to customize their profiles, follow other users, and receive notifications about new content or interactions.

5. **Scalability and Extensibility:**
    - Design the system with scalability in mind to accommodate future growth in user base and content volume.

- Build the app using modular components and scalable technologies to facilitate future enhancements and feature additions.

6. **Performance and Reliability:**
    - Ensure the app is responsive, fast, and reliable by leveraging optimized frontend frameworks (e.g., React.js) and scalable backend services (e.g., MongoDB and Firebase).
    - Implement caching, asynchronous operations, and error handling mechanisms to enhance performance and user experience.

7. **SEO and Accessibility:**
    - Optimize the app for search engines (SEO) by implementing SEO-friendly URLs, metadata, and structured data.
    - Enhance accessibility by adhering to web accessibility standards (e.g., WCAG) to ensure usability for users with disabilities.

8. **Continuous Improvement and Maintenance:**
    - Establish processes for continuous improvement through user feedback, analytics, and iterative development cycles.
    - Implement robust monitoring, logging, and error reporting to facilitate maintenance, troubleshooting, and bug fixes.

The overarching objective is to create a feature-rich blog app with an admin dashboard that empowers users to create and share content, fosters community engagement, and provides administrators with the tools to manage and optimize the platform effectively. The system aims to deliver a seamless, enjoyable experience for both content creators and consumers while ensuring scalability, performance, and maintainability over time.

CHAPTER 3:
ANALYSIS AND DESIGN

1.1 SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Introduction

1.1 Purpose

- The purpose of this Software Requirements Specification (SRS) document is to outline the requirements for the development of a Blog Application. This system aims to provide a comprehensive platform for users to search, upload, and write the blog seamlessly. The SRS defines the functional and non-functional requirements of the system to guide the development process effectively.

1.2 Scope

- The scope of this blog app with an admin dashboard encompasses a comprehensive set of features designed to facilitate content creation, user engagement, and platform management. Key components of the app's scope include user registration and authentication, allowing users to sign up, create profiles, and interact with content. The system supports content management functionalities such as creating, editing, and organizing blog posts, incorporating rich media elements and categorization options. For administrators, the scope includes robust dashboard capabilities to oversee user accounts, moderate content, and analyze platform performance through built-in analytics. The app's scope also emphasizes scalability and extensibility to accommodate future enhancements and growing user bases.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **API:** Application Programming Interface
- **CRUD:** Create, Read, Update, Delete

OVERALL DESCRIPTION

2.1 Product Perspective

- From a product perspective, this blog app with an admin dashboard serves as an integrated platform that combines user-facing features with robust administrative functionalities. The product caters to two primary user groups: general users and administrators. For general users, the app provides an intuitive interface for reading, creating, and interacting with blog content. Users can register, log in, and personalize their profiles, enhancing engagement and interactivity. Administrators benefit from a dedicated dashboard to oversee user activity, manage content, and monitor platform performance. This product perspective emphasizes a cohesive user experience, ensuring seamless interaction between users and administrators while maintaining efficient content management and platform control.

2.2 Product Functions

The key functions of the Online Hotel Booking System include:

- User Registration and Authentication
- User profile management and personalization
- Blog post creation, editing, and deletion
- Commenting and interaction on blog posts
- Search and filtering capabilities for blog content
- Administrator dashboard for user management and content moderation
- Analytics and reporting features for platform performance
- Booking Management for Administrators
- Dynamic Content Management for Administrators

2.3 User Classes and Characteristics

The system will cater to two primary user classes:

- **Users:** Individuals who will be able to create the blog and will be able to post it.
- **Administrators**: Admins can edit the content and also check the analytics.

2.4 Operating Environment

- The system will operate on web browsers such as Google Chrome, Mozilla Firefox, and Safari. It will be compatible with Windows, MacOS, and Linux operating systems. The server-side environment will consist of Apache or Nginx web servers, PHP or Node.js runtime environments, and MySQL or PostgreSQL databases.

SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

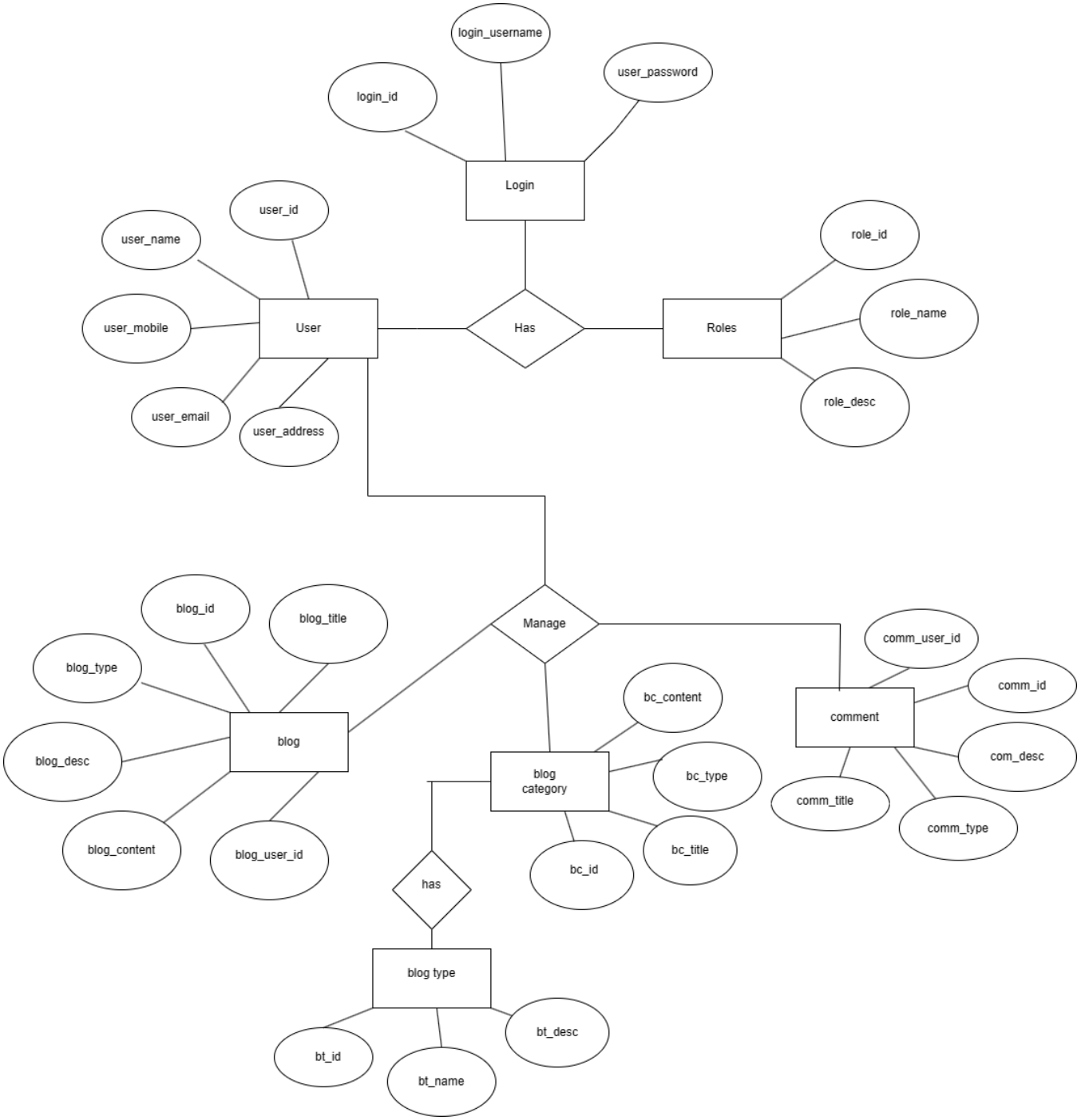
3.1.1 User Interface

- The user interface shall be intuitive, responsive, and accessible across various devices including desktops, laptops, tablets, and smartphones. It shall adhere to modern design principles for optimal user experience.

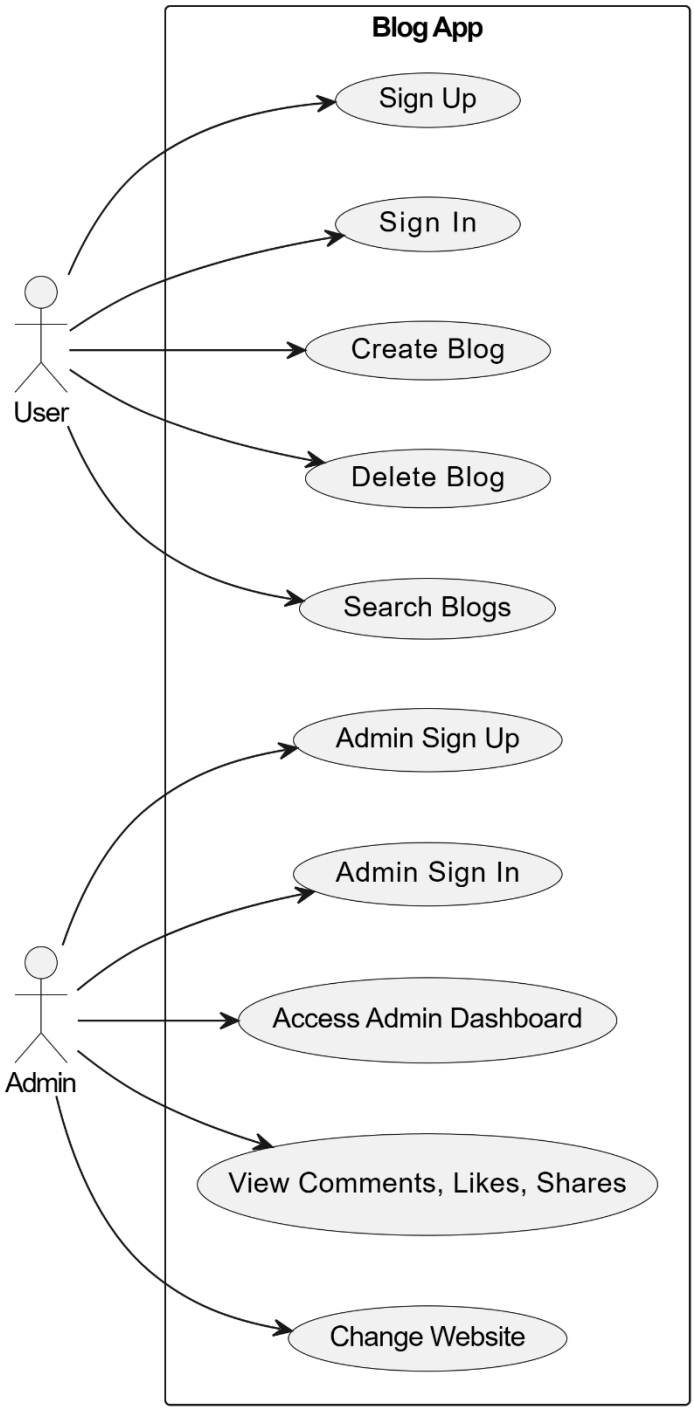
3.1.2 API Interfaces

- The system shall integrate with external APIs for functionalities such as payment processing and mapping services.

1.2 ERD DIAGRAM

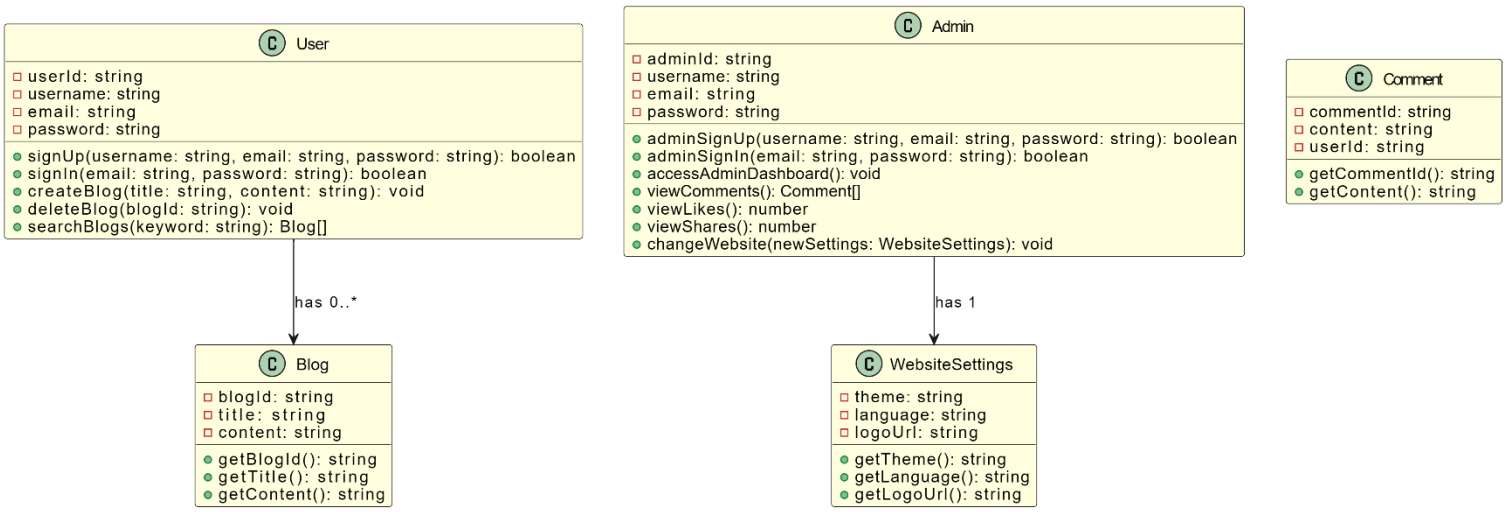


1.3 USE CASE

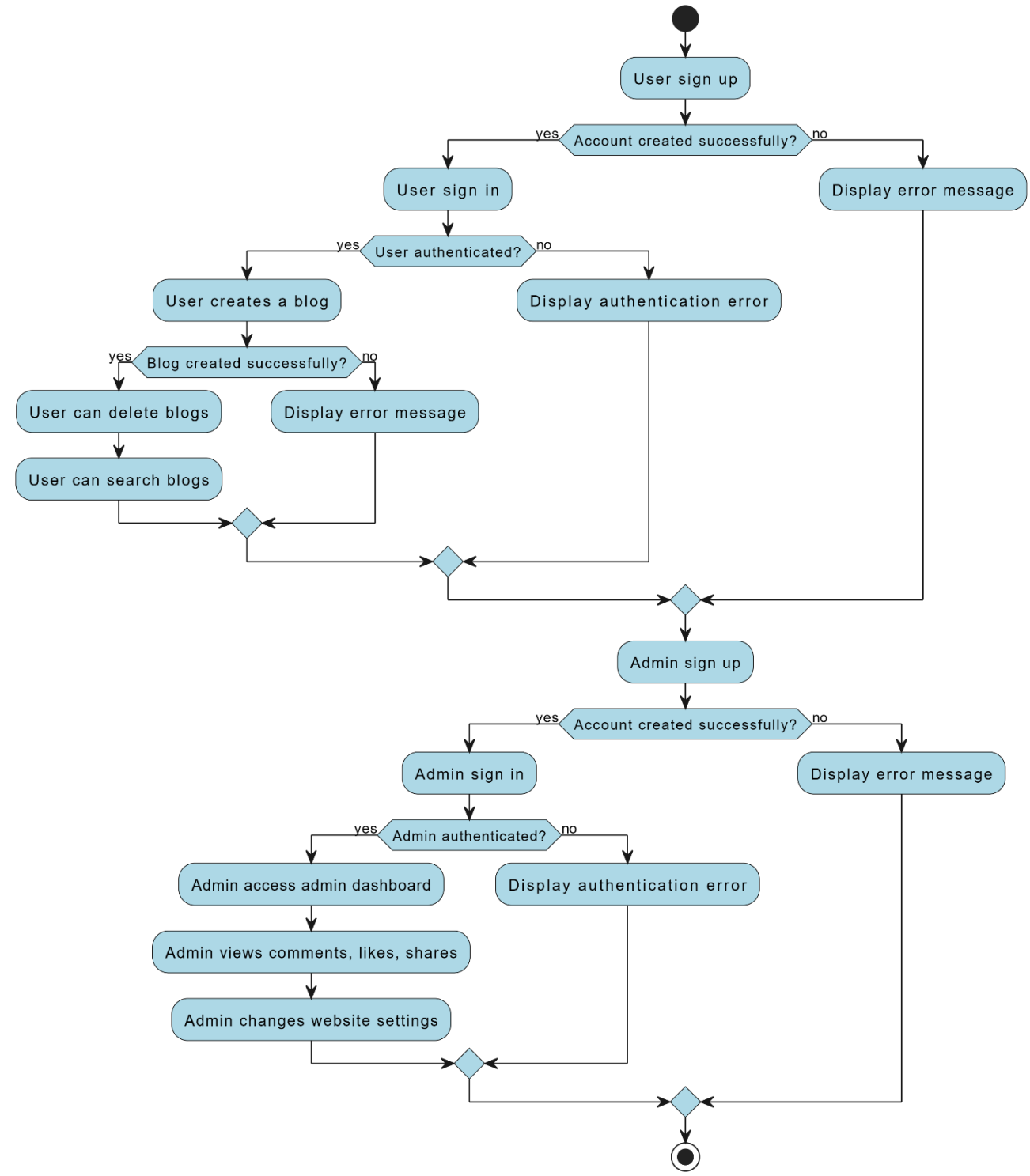

Blog App
Sign Up
Sign In
Create Blog
Delete Blog
Search Blogs
User
Admin Sign Up
Admin Sign In
Access Admin Dashboard
View Comments, Likes, Shares
Change Website
Admin

1.4 CLASS DIAGRAM

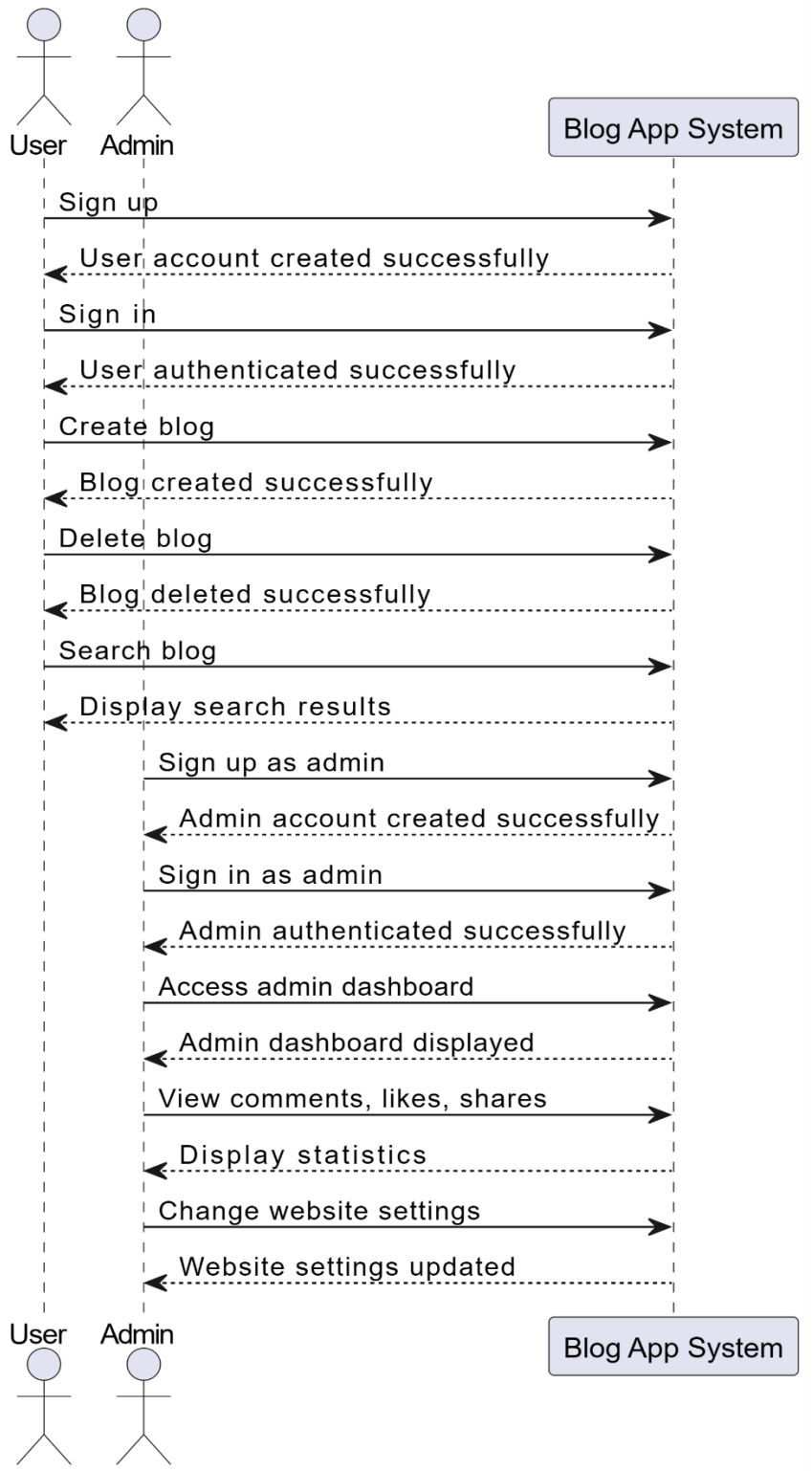
2.



2.1 ACTIVITY

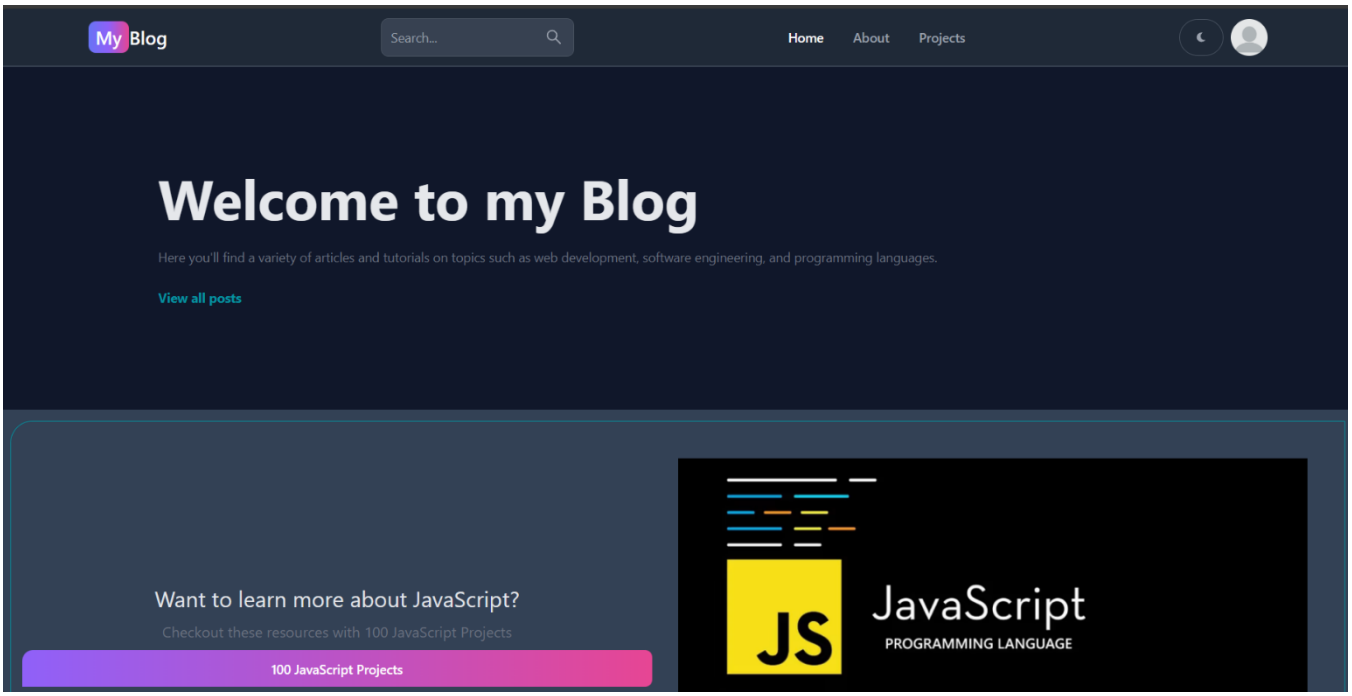

User sign up
Account created successfully?
yes
no
User sign in
Display error message
User authenticated?
yes
no
User creates a blog
Display authentication error
Blog created successfully?
yes
no
User can delete blogs
Display error message
User can search blogs
Admin sign up
Account created successfully?
yes
no
Admin sign in
Display error message
Admin authenticated?
yes
no
Admin access admin dashboard
Display authentication error
Admin views comments, likes, shares
Admin changes website settings

2.2 SEQUENCE DIAGRAM

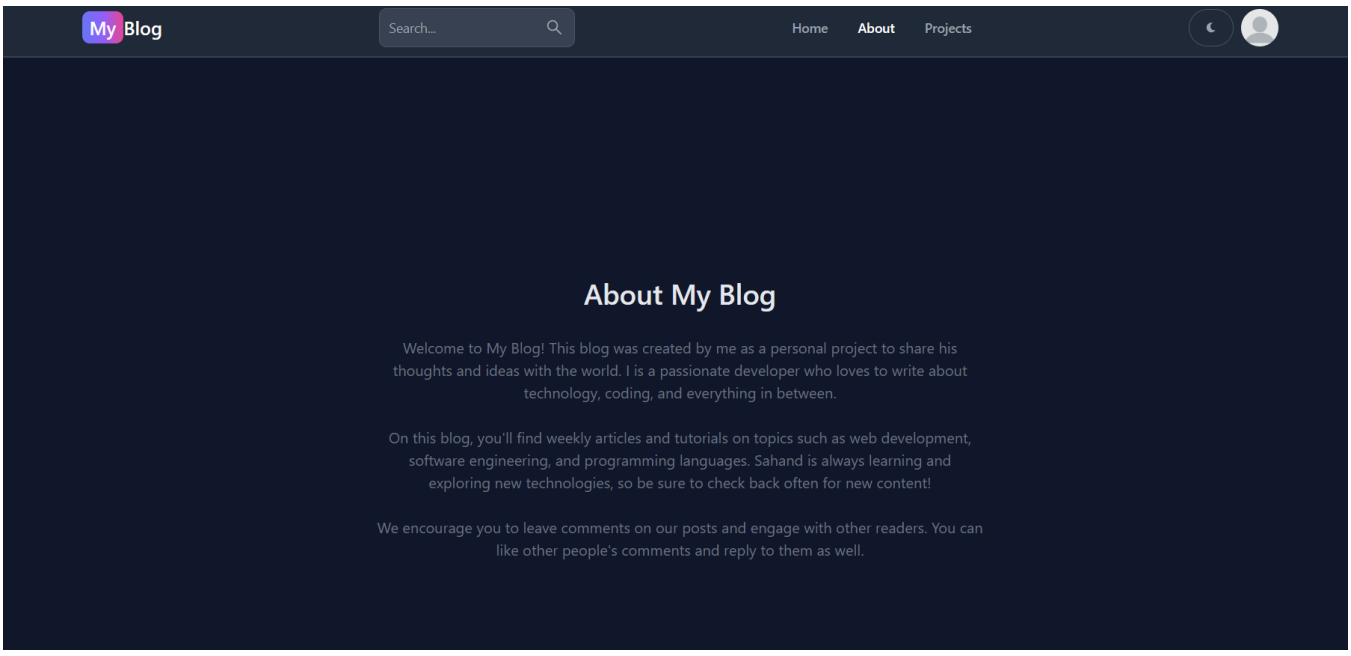


CHAPTER 4:-

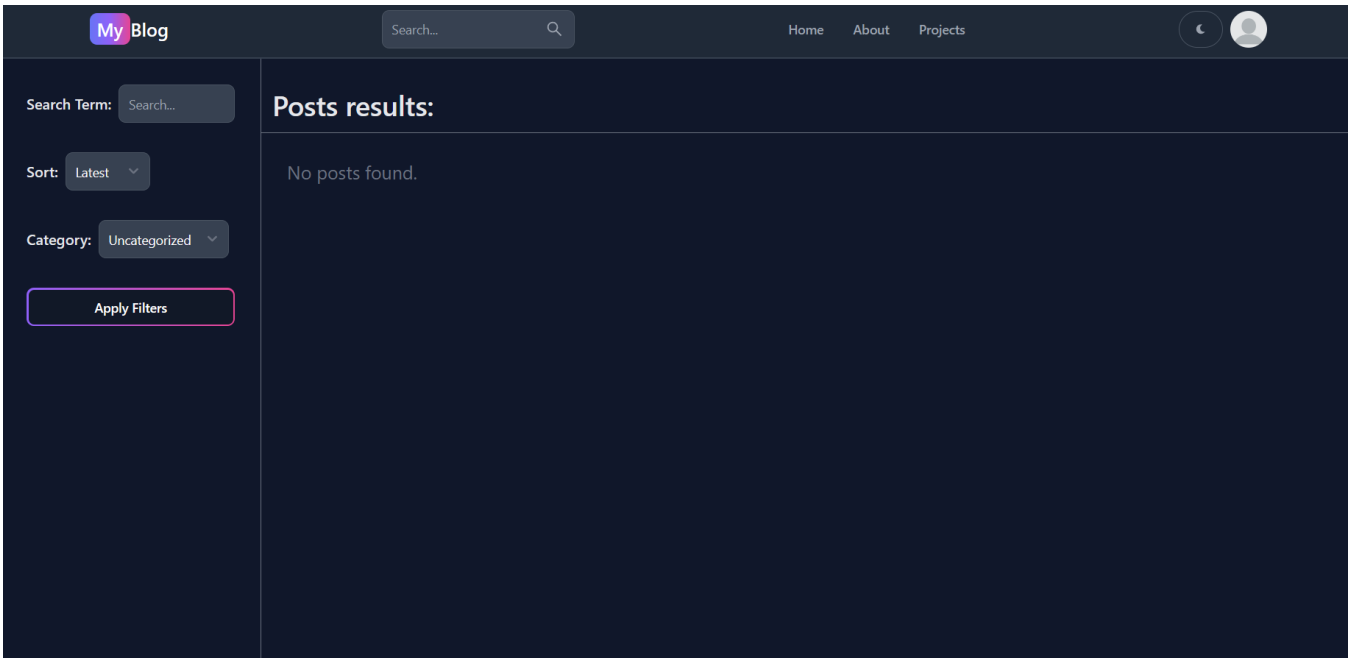
HOMEPAGE SCREENSHOTS



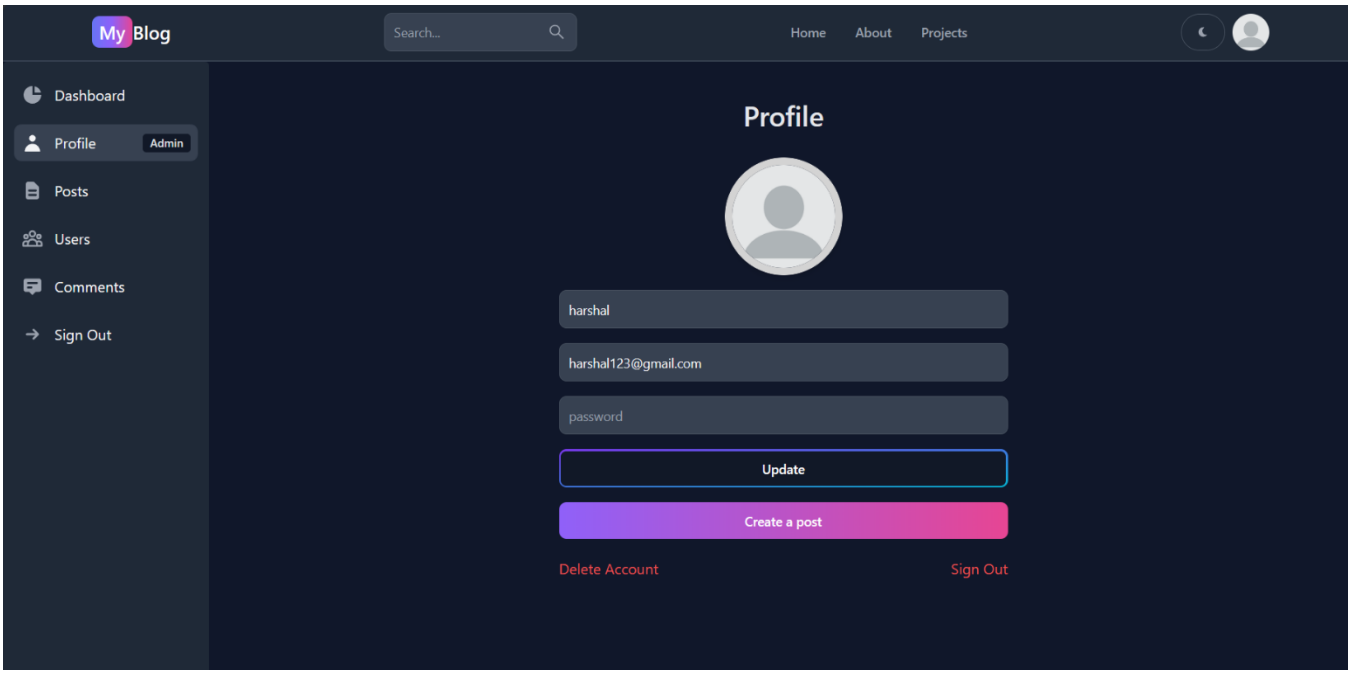
ABOUT SECTION



SEARCH SECTION

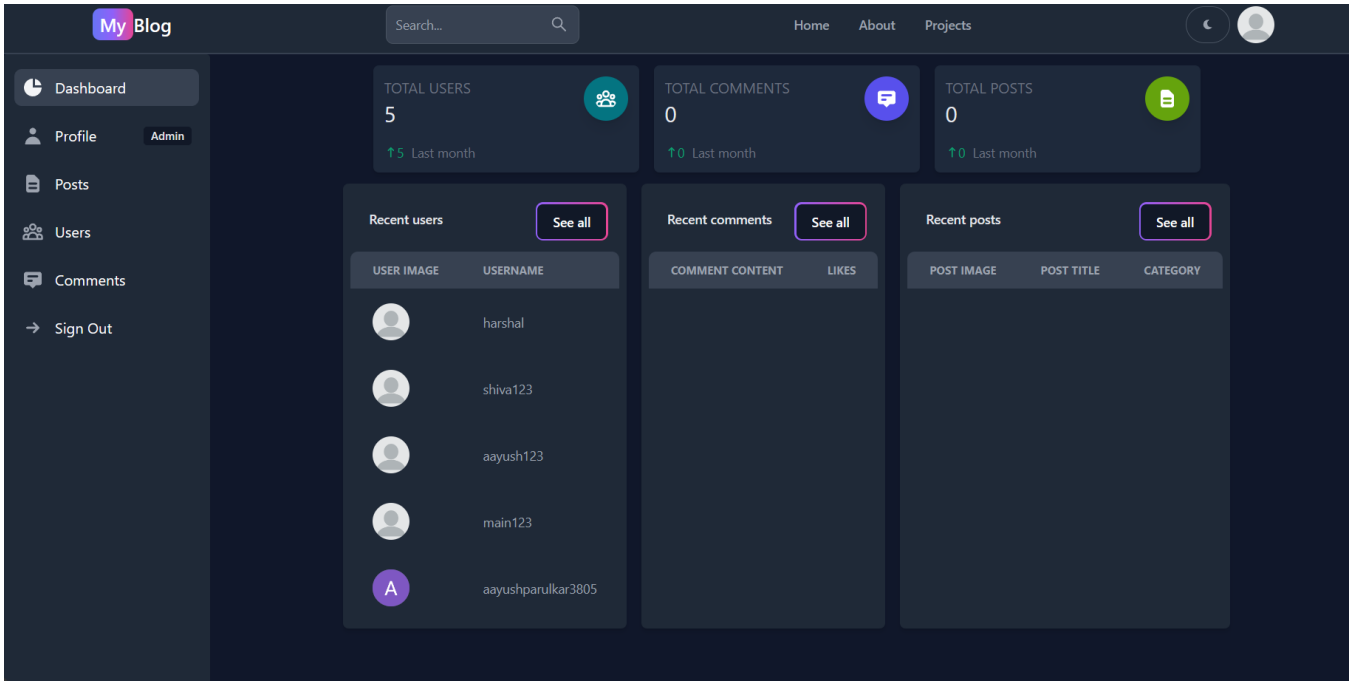


PROFILE SECTION

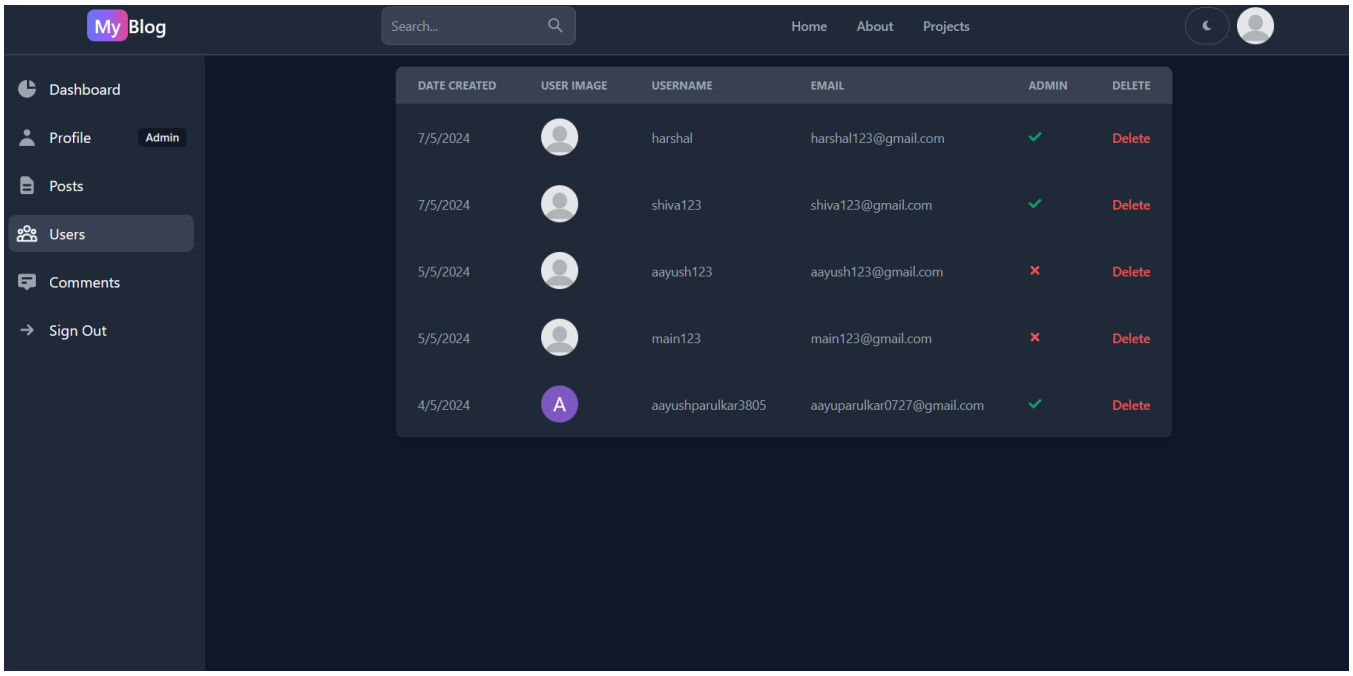


CHAPTER 5:ADMIN PANEL

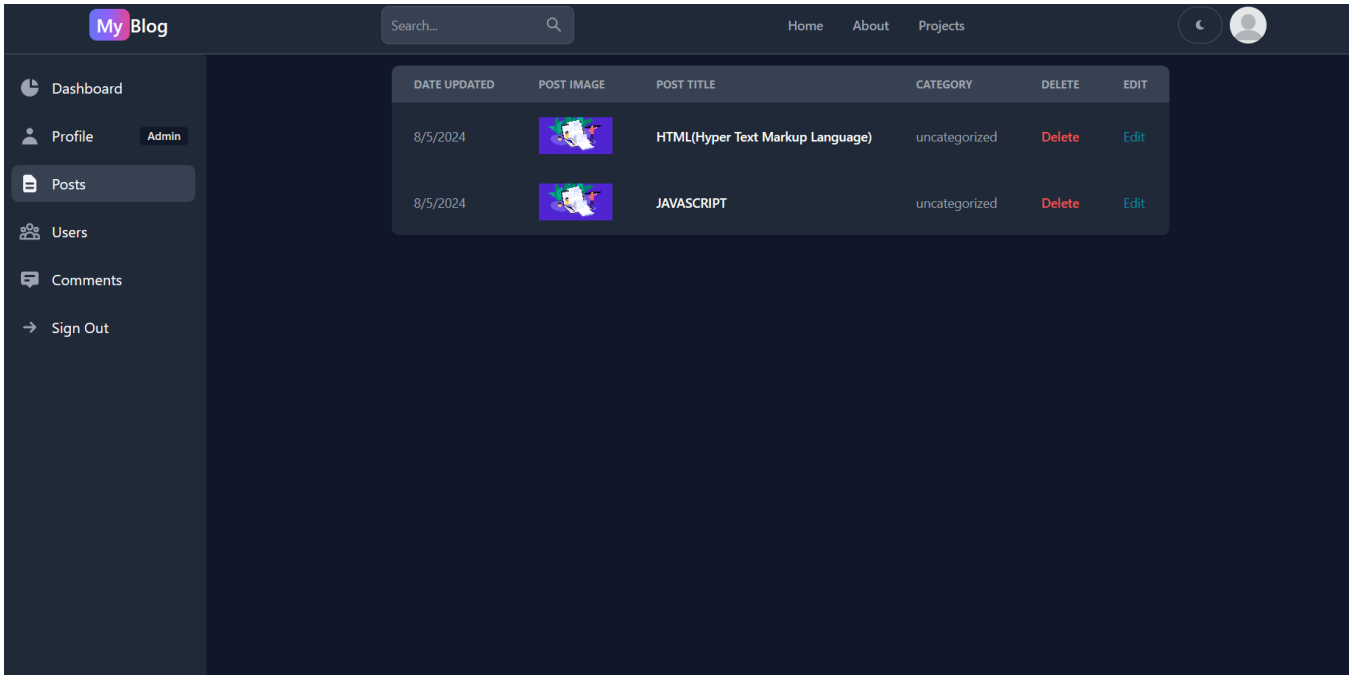
DASHBOARD



USERS SECTION



POSTS SECTION



CHAPTER 6:

CODE SNIPPETS

APP.JS

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import SignIn from "./pages/SignIn";
import Dashboard from "./pages/Dashboard";
import Projects from "./pages/Projects";
import SignUp from "./pages/SignUp";
import Header from "./components/Header";
import Footer from "./components/Footer";
import PrivateRoute from "./components/PrivateRoute";
import OnlyAdminPrivateRoute from "./components/OnlyAdminPrivateRoute";
import CreatePost from "./pages/CreatePost";
import UpdatePost from "./pages/UpdatePost";
import PostPage from "./pages/PostPage";
import ScrollToTop from "./components/ScrollToTop";
import Search from "./pages/Search";

export default function App() {
  return (
    <BrowserRouter>
      <ScrollToTop />
      <Header />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/sign-in" element={<SignIn />} />
        <Route path="/sign-up" element={<SignUp />} />
        <Route path="/search" element={<Search />} />
        <Route element={<PrivateRoute />}>
          <Route path="/dashboard" element={<Dashboard />} />
        </Route>
        <Route element={<OnlyAdminPrivateRoute />}>
          <Route path="/create-post" element={<CreatePost />} />
          <Route path="/update-post/:postId" element={<UpdatePost />} />
        </Route>

        <Route path="/projects" element={<Projects />} />
        <Route path="/post/:postSlug" element={<PostPage />} />
      </Routes>
      <Footer />
    </BrowserRouter>
  );
}
```

FIREBASE.JS

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// const firebaseConfig = {
//   apiKey: "AIzaSyCsjTUpvceJqOBmPU9jREQGCffeHjnAHeI",
//   authDomain: "mernstackblog-794a3.firebaseapp.com",
//   projectId: "mernstackblog-794a3",
//   storageBucket: "mernstackblog-794a3.appspot.com",
//   messagingSenderId: "449865902794",
//   appId: "1:449865902794:web:72322278678cf2f305e233",
// };

const firebaseConfig = {
  apiKey: "AIzaSyCsjTUpvceJqOBmPU9jREQGCffeHjnAHeI",
  authDomain: "mernstackblog-794a3.firebaseapp.com",
  projectId: "mernstackblog-794a3",
  storageBucket: "mernstackblog-794a3.appspot.com",
  messagingSenderId: "449865902794",
  appId: "1:449865902794:web:0447dcd778edf69905e233",
};

// Initialize Firebase
export const app = initializeApp(firebaseConfig);
```

CHAPTER 7:

TESTING

User Features:

User Registration and Authentication:

1. **Test Case:** Verify User Registration

Test Steps:

- Navigate to the registration page.
- Fill in valid user details including email and password.
- Submit the registration form.

Expected Result: User should receive a verification email and be redirected to the login page.

2. **Test Case:** Verify User Authentication

Test Steps:

- Navigate to the login page.
- Enter valid credentials.
- Click on the login button.

Expected Result: User should be successfully logged in and redirected to the dashboard.

3. **Test Case:** Verify that Search functionality is working

Test Steps:

- Navigate to the Search Page.
- Try applying filter to search.
- Try different prompts.

Expected Result: It should display different Blogs based on search.

4. **Test Case:** Update Profile

Test Steps:

- Navigate to the Profile Page.
- Enter valid details
- Try changing credentials.

Expected Result: The users should be able to Login using the updated credentials.

Admin Features:

Dashboard:

5. **Test Case:** Verify Dashboard Overview

Test Steps:

- Log in to the admin dashboard.
- Check the dashboard for checking the analytics like comments ,likes and no of posts .

Expected Result: Dashboard should display relevant statistics and analytics accurately.

6. **Test Case:** Verify User Account Management

Test Steps:

- Access the user management section.
- Modify user details or ban a user.

Expected Result: Changes to user accounts should be reflected in the system.

7. **Test Case:** Check the Posts.

Test Steps:

- Navigate to the post section in dashboard.
- Respond to or delete user queries and reviews.

Expected Result: System should allow admins to manage the posts.

Content Management:

8. **Test Case:** Create Post Section

Test Steps:

- Navigate to the Create Post section.
- Try adding title and create post according to need along with the image.

Expected Result: The admin should be able to create the post according to need.

9. **Test Case:** Mode check

- Try switching between dark and light mode

Expected Result: Changes should be reflected on the user and admin interface immediately.

CHAPTER 8:
LIMITATION

1. **Limited Scalability:** Depending on the architecture and hosting infrastructure, the blog app may face limitations in handling high traffic and large amounts of data, affecting its scalability.

2. **Feature Set:** The app's feature set may be limited compared to more comprehensive content management systems (CMS), with fewer customization options and functionalities.

3. **Security Concerns:** Without robust security measures, the blog app may be vulnerable to cyber threats such as data breaches, unauthorized access, or injection attacks.

4. **Dependency on Third-Party Services:** The app might rely on third-party services for specific functionalities (e.g., user authentication, database hosting), which could introduce dependencies and potential points of failure.

5. **Admin Dashboard Complexity:** The admin dashboard's capabilities may be basic, lacking advanced analytics, reporting tools, or user management features found in enterprise-level solutions.

6. **Maintenance and Updates:** Regular maintenance and updates are essential for optimal performance and security. Limited resources or expertise could impact the app's stability and longevity.

7. **Customization Limitations:** Users and admins may have limited control over customization options, including theme changes, layout modifications, or content styling.

8. **Integration Challenges:** Integrating additional features or third-party services into the existing app architecture may be complex and require significant development effort.

9. **User Engagement Tools:** The app may lack advanced user engagement tools like automated email campaigns, social media integration, or personalized content recommendations.

10. **Support and Documentation:** Limited support resources or documentation may hinder user and admin understanding of the app's features and troubleshooting processes.

CHAPTER 9:
PROPOSED ENHANCEMENT

1. **Limited Scalability:** Depending on the architecture and hosting infrastructure, the blog app may face limitations in handling high traffic and large amounts of data, affecting its scalability**.**

2. **Feature Set:** The app's feature set may be limited compared to more comprehensive content management systems (CMS), with fewer customization options and functionalities.

3. **Security Concerns:** Without robust security measures, the blog app may be vulnerable to cyber threats such as data breaches, unauthorized access, or injection attacks.

4. **Dependency on Third-Party Services:** The app might rely on third-party services for specific functionalities (e.g., user authentication, database hosting), which could introduce dependencies and potential points of failure.

5. **Admin Dashboard Complexity:** The admin dashboard's capabilities may be basic, lacking advanced analytics, reporting tools, or user management features found in enterprise-level solutions.

6. **Maintenance and Updates:** Regular maintenance and updates are essential for optimal performance and security. Limited resources or expertise could impact the app's stability and longevity.

7. **Customization Limitations:** Users and admins may have limited control over customization options, including theme changes, layout modifications, or content styling.

8. **Integration Challenges:** Integrating additional features or third-party services into the existing app architecture may be complex and require significant development effort.

9. **User Engagement Tools:** The app may lack advanced user engagement tools like automated email campaigns, social media integration, or personalized content recommendations.

10. **Support and Documentation:** Limited support resources or documentation may hinder user and admin understanding of the app's features and troubleshooting processes.

CHAPTER 10:
CONCLUSION

In conclusion, the blog app with an integrated admin dashboard presents a comprehensive solution for content management and user interaction. This platform enables users to create and manage blog posts efficiently while providing administrators with tools to oversee and enhance the overall blogging experience. The app's user-centric features, including account management, blog creation, deletion, and search functionality, cater to the needs of both content creators and consumers. Additionally, the admin dashboard offers vital insights into user engagement metrics such as comments, likes, and shares, empowering administrators to make data-driven decisions.

Despite its strengths, the blog app may encounter limitations related to scalability and performance under high user loads. To address these challenges, proposed enhancements could focus on optimizing database performance, implementing caching mechanisms, and adopting cloud-based solutions for improved scalability. Moreover, enhancing security measures to safeguard user data and preventing unauthorized access to the admin dashboard will be crucial for maintaining user trust and compliance with data protection regulations.

Looking ahead, proposed enhancements for the blog app include advanced user management features, such as user roles and permissions, to enable different levels of access within the platform. Additionally, integration with third-party services for SEO optimization, social media sharing, and email marketing can expand the app's reach and impact. By prioritizing continuous improvement and innovation, the blog app aims to evolve into a leading platform for content creators and bloggers, fostering meaningful engagement and connections within the online community.

CHAPTER 11:

BIBLOGRAPHY

1. Tailwind CSS: https://tailwindcss.com/docs

2. HTML: MDN Web Docs: HTML: https://developer.mozilla.org/en-US/docs/Web/HTML

3. CSS: MDN Web Docs: CSS: https://developer.mozilla.org/en-US/docs/Web/CSS

4. JavaScript (JS): MDN Web Docs: JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript

5. React.js: React Reference Overview – React

6. Firebase: https://firebase.google.com/docs

7. Node.js : Index | Node.js v22.1.0 Documentation (nodejs.org)

PLAGARISM REPORT