**BlogApp**

Final Report

Submitted in Partial Fulfilment of the Requirements for

the Award of Degree of

**Bachelor in Technology**





**Submitted by:**

<your name><your rollno.>

CERTIFICATE

This is to certify that the work presented in this Project entitled “BlogApp” is a bonafide record of the work done during the period from Jan – June 2023 at “FUTURE FINDERS” by <your name>

The project work is an authentic record of my own work and is carried out under the supervision and guidance of Guide <GUIDE NAME>, <X> Department. The matter presented in the report has not been submitted elsewhere, wholly or in part, for the award of any other degree or diploma.

Your name(19xxxxx)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

<Guide name>

**Guide Name**

Department of <X>

<HOD NAME>

**HOD <X>**

<X> Department Chandigarh Engineering College, Landran, Mohali, PUNJAB

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude to the Principal, Chandigarh Engineering College, Landran, for providing this opportunity to carry out the present work.

The constant guidance and encouragement received from Prof. (Dr.) <X>, Professor and Head, Department of <XYZ> Engineering, has been of great help in carrying our present work and helped us in completing this project with success.

I would like to express a deep sense of gratitude to “FUTURE FINDERS” team and my Project Guide Prof. <X>, <XYZ> department for the guidance and support in defining the design problem and towards the completion of my project work. Without their wise counsel and able guidance, it would have been impossible to complete the thesis in this manner.

I am also thankful to all the faculty and staff members of FUTURE FINDERS ORGANISATION for their intellectual support throughout the course of this work.

<Your name>(19xxxxx)

ABSTRACT

Blogapp is a web-based application that allows users to create, read, update, and delete blog posts. It is built using the MERN stack, which consists of MongoDB, Express, React, and Node.js. The purpose of the blogapp is to provide an easy-to-use platform for bloggers to create, manage, and publish blog posts. The application allows bloggers to focus on writing quality content without worrying about the technical aspects of running a blog.

The blogapp provides user authentication and authorization, which allows registered users to create and manage their own blog posts and comments. The application also has two user roles: admin and user, where admin users have access to all features, while regular users can only manage their own blog posts and comments.

The blogapp allows users to add comments to blog posts and manage categories and tags. It also provides detailed blog statistics to bloggers. Users can view their blog statistics, such as the number of views and comments on their blog posts, from the dashboard.

The blogapp is built using modern web technologies such as MongoDB, Express, React, Node.js, and JWT. Mongoose, an Object Data Modeling (ODM) library for MongoDB, is used for interacting with the database. Material-UI, a React UI component library, is used for building the user interface. Nginx, a web server, is used for reverse proxy and load balancing. Docker, a containerization platform, is used for deployment.

**Table of Contents**

Certificate

Acknowledgement

Abstract

Chapter 1 Introduction

1.1 Introduction Profiles

1.1.1 Project Profile

1.1.2 Company Profile

1.1.3 About Course

1.2 Introduction of Project

1.4 Problem Statement

1.5 Project Objective

1.6 Scope of Study

1.7 Feasibility Study

Chapter 2 Literature Review

Chapter 3 Methodology

Chapter 4 Project Setup

1. Hardware Specifications and Requirements
2. Software Requirements
3. Setup a MERN stack Project

Chapter 5 Implementation

5.1 Creating Backend

5.2 Creating Frontend

5.3 Creating User Authentication

5.4 Creating Blog Routes

* 1. Testing and Deployment

5.5.1 Test the blogapp's functionality

Chapter 6 Screenshots of the project

Chapter 7 Conclusions and Future Work

References



**List of Figures**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.** | **Figure** | **Description** | **Page No.** |  |
| **No.** |  |  |  |  |
|  |  |  |  |  |
| 1 | Figure 1.4 | Flow Chart | 9 |  |
|  |  |
|  |  |  |  |  |
| 2 | Figure 2.1 | Age recognition | 10 |  |
|  | Figure 2.2 | Emotion recognition | 11 |  |
|  | Figure 2.3 | Demonstration of playlist | 12 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 3 | Figure 3.1 | Raspberry pi 4 | 14 |  |
|  |  |  |  |  |
|  | Figure 3.2 | Pycam | 15 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Figure 3.4 |  | 18 |  |
|  |  |  |  |  |
|  | Figure 3.5 |  | 20 |  |
|  |  |  |  |  |
|  | Figure 3.6 |  | 20 |  |
|  | Figure 3.7 |  | 20 |  |
|  | Figure 3.8 |  | 21 |  |
|  | Figure 3.9 |  |  |  |
|  |  |  |  |  |
|  | Figure 3.10 |  | 22 |  |
|  | Figure 3.11 |  | 22 |  |
|  |  |  |  |  |
| 3 | Figure 3.12 |  | 22-23 |  |
|  |  |  |  |  |
|  | Figure 3.13 |  | 24-26 |  |
|  |  |  |  |  |

CHAPTER 1 – INTRODUCTION

# Introduction Profiles

## Project Profile

|  |  |
| --- | --- |
| **Title** | **BlogApp Web App** |
| Organization | Future Finders Organisation |
| Category | BlogApp web application |
| Duration | 6 Months |
| Front-End | React (View Library) |
| Back-End | Node.js + MonogoDB + Express.js + openAI API |
| Guide | <guide name> (Internal Guide) |
| Submitted by | <your name>  Roll no. : <your roll number> |
| Submitted to | Department of CSE, Chandigarh Engineering College, Landran |

**1.2. COMPANY PROFILE**

**ABOUT COMPANY:**



Future Finders is a group of professionals who love to think outside the box. Everything we do to refine and achieve absolute excellence is our motto. We value your unbeatable talent, your innovative ideas and turn them into reality. Let us get together and head for the successful brand!

**PHILOSOPHY**

* To impart hardcore practical quality training among students/developers about latest technologies trending today.
* To share knowledge of information security and create awareness in the market. The solution to clients' as per the International standard practices and governance.
* To support good business practices through continual employee training and education
* To equip a local team with a strong knowledge of international best practices and international expert support so as to provide practical advisories in the best interests of our clients

**COMPANY’S VISION & MISSION**

FUTURE FINDERS only vision is to provide with cutting edge practical skills so that students can easily cope with and quickly adapt to the ever- changing technologies in the corporate environment. Our mission at FUTURE FINDERS is to create the highest standards in education through improvisation of quality and practical skills.

**SERVICES**

* Software Testing
* Mobile Application Testing
* [Web Development](https://futurefinders.in/web-development/)
* [Web Designing](https://futurefinders.in/web-desiging/)
* [Mobile App Development](https://futurefinders.in/mobile-app-development/)
* [Digital Marketing Services](https://futurefinders.in/digital-marketing-services/)
* Embedded System Services

**Why Choose Us?**

* Hundreds of Clients & Nearly a Decade of Experience
* Goal Oriented, ROI-Driven Focus
* A Streamlined / Quality-Driven Process
* Talented Designers & Expert Developers
* Our Websites & E-marketing Platforms are Easy to Manage
* We Are Dedicated to Our Clients’ Success
* **We focus on imparting practical skills to the trainees & not just theoretical knowledge. The courses are designed in this way at FUTURE FINDERS correspond to the standards of the corporate divisions and industries. Only through the acquisition of practical skills; you can handle the everlasting technologies that venture out in real-time situations.**
* At **FUTURE FINDERS,** we have competence to expand and adjust as per client specific requirements.
* **Skilled Workforce:** At **FUTURE FINDERS,** you deal with the highly professional and proficient employees.
* **Cost Efficiency:** We help you to reduce the unnecessary investment and ask for the reasonable amount of money.
* **Quality Of the Product:** Our software service sector has been maintaining the highest international standards of quality.
* **Infrastructure:** Well organized team and tools to handle the projects with responsible approach Hardware, Software, Networking, Voice, Conferencing, disaster recovery all infra all you need for international projects.
* **Ongoing Involvement:** **FUTURE FINDERS**, products are “built for change” as we are well responsive that the necessity to improve a Web solution generally arises even before the solution is out of the door.
* **Partnership:** **FUTURE FINDERS**, considers every client a partner. From the initial stages, you are closely involved into the procedure of technical classification, development, and testing.

**KEY PROFESSIONALS**

In addition to a panel of eminent consultants and advisors, we have a dedicated pool of trained Developers and Trainer, investigators, working under the guidance of professional managers. **“A Ship is as good as the crew who sail her**.**”** Our Technical team of professionals handing, designing & delivering of projects has a strong presence in the North India & the US. Our engineers are already working on the latest technologies like **I-Phone & Android** Applications, **Robotics**, **VLSI-VHDL**, Embedded System, Networking and **Cloud computing.** Our key professionals and advisors are listed below:

**MR. BONISH SINGLA:** (**DIRECTOR**)



He is the backbone of FUTURE FINDERS, and a man with more than 9 years rich practical experience who believes in taking up new ventures and projects. He has been awarded many times for his exemplary work in process improvement for IT Service Delivery Domains. MASTERS in Computer applications and Certified from CU Certification. Holds total of 9 Years of rich experience including 5 Years in Information Security Implementation, Maintenance and Auditing and initial over 4 years of experience in Project Management, Client Relationship Management and Server, Desktop and IT Service Delivery/

**MISS.HARJIT KAUR: (BRANCH MANAGER)**



She has more than 5 years solid industrial experience in software companies & is very innovative in her technical approach. MCA, Diploma in Information Technology, expertise in search engine optimization and web designing.

**MISS. ISHA BALA: (HUMAN RESOURCES)**



A very committed team leader and multi-talented person, who has been professionally attached with company.

**MISS. DIVYA: (HEAD COUNCILOR)**



She is determinate and a team player. She is good at problem solving skills and a fast learner. She can do multitasking.

**MR. SUKHBIR SINGH: (TECHNICAL HEAD)**



He has more than 3 years of experience in PLC and Scada.). He has been awarded many times for his brilliant services.

## MR. JASPAL SINGH: (CIVIL &MECHANICAL HEAD)



He has more than 38years of experience in industrial field. He is providing services as a technical trainer for more than 8 years. He did his B.Tech in Mechanical Engineering from PEC (Punjab Engineering College). He has been awarded many times for his brilliant services.

## MR. CHETAN KALRA: (DIGITAL MARKETING HEAD)



B.tech (CSE) – IKG-PTU, Expertise in Python Programming, Full Stack Development, Presenting ideas for web development software, including Application software, Working closely with analysts, Senior Developers, Programmers, designers and staff, Producing detailed specifications and writing the program codes, Testing the product in controlled, real situations before going live, Preparation of training manuals for users and Maintaining the systems once they are up and running.

**About the course**

I have chosen “Full Stack Web Development” course in this company.

Full Stack Web Development is nothing but complete designing of both websites and applications where the developers need to work from frontend to backend development. With our Full Stack Web Development Course, you will become an expert in all the aspects of web development such as JavaScript, React js, MERN Stack and Python. In Frontend you are able to learn – HTML5, CSS3, Bootstrap, JavaScript, jQuery, React Js, Chakra UI, and Material UI whereas in Backend – Node.js, PHP, and Flask. Finally in Database – MySQL, PostgreSQL, MongoDB.

The MERN stack is well-suited for building modern web applications because it provides a complete end-to-end solution for building both client-side and server-side components. React allows for easy building of complex user interfaces, while Express.js provides a flexible and scalable server-side framework. MongoDB offers a highly scalable and flexible data storage solution, and Node.js allows developers to write server-side code using JavaScript, which can be a significant advantage in terms of development speed and efficiency.

In the BlogApp project, we will be using the MERN stack to build a chatbot application that uses OpenAI's GPT-3 API for generating responses to user messages. The chatbot's interface will be built using React components, while the server-side code will be developed using Express.js and Node.js. MongoDB will be used as the database to store user information and chat histories. By using the MERN stack, we can leverage the strengths of each component to build a scalable, performant, and feature-rich chatbot application.

Overall, the MERN stack provides a powerful and flexible development platform for building modern web applications, and it is an excellent choice for building the BlogApp project.

* 1. **Introduction of Project:**

A blog app is an application that enables users to create and manage their own blog posts. With the rise of the internet and social media, blogging has become an essential way for people to express their thoughts, opinions, and experiences with a global audience. Creating a blog app using the MERN stack (MongoDB, Express, React, Node) can provide a scalable and efficient solution to manage a blog.

The MERN stack is a popular choice for developing web applications because it provides a full-stack solution that includes a powerful database, a flexible backend, and a dynamic frontend. MongoDB is a NoSQL database that provides high performance, scalability, and flexibility. Express is a minimal and flexible Node.js web application framework that provides a range of features for building web apps. React is a popular and widely-used JavaScript library for building user interfaces. Node.js is a powerful JavaScript runtime that enables developers to build scalable and efficient web applications.

The backend of the blog app can be developed using Node.js and Express. The backend should include APIs for handling user authentication, blog post management, and other necessary functionalities. You can use Mongoose as the ORM (Object-Relational Mapping) library to define the database schema and perform CRUD (Create, Read, Update, Delete) operations on the database. User authentication can be implemented using libraries such as Passport.js and bcrypt to handle user authentication and authorization.

The frontend of the blog app can be developed using React. You can use various libraries and frameworks such as Material UI, Bootstrap, and React Router to make the development process faster and easier. The frontend should include a user interface for managing blog posts and creating new ones. You can create a form for users to create blog posts and add functionality for managing posts. You can use libraries such as Formik and Yup to handle form validation. You can also add a commenting system to enable users to leave comments on blog posts. You can use libraries such as Disqus or create your own custom solution.

To deploy the blog app, you can use a cloud hosting provider such as Heroku or AWS. After completing the development, you need to test the blog app to ensure that it is working correctly and is free from any bugs or errors. You can use testing frameworks such as Jest and Enzyme to test the app's functionality. Once you are satisfied with the app's functionality, you can deploy it to a production environment. You can also set up a CI/CD pipeline to automate the deployment process and make it faster and more efficient.

In conclusion, creating a blog app using the MERN stack can provide a powerful and scalable solution for managing a blog. By following the outlined steps, developers can build a blog app that includes user authentication, blog post management, and a commenting system. This can enable bloggers to create and manage their own blog posts, interact with their audience, and grow their online presence.

**1.2 Background of Study:**

The background of the study for a blog app using the MERN stack can be viewed from several perspectives. Here are a few examples:

Blogging: Blogging has become a popular medium for individuals, organizations, and businesses to share their ideas, experiences, and products with a global audience. With the growth of the internet and social media, blogs have become an essential part of digital marketing and online branding. A blog app using the MERN stack can provide a powerful and scalable solution for bloggers to manage their content and interact with their audience.

MERN Stack: The MERN stack has become a popular choice for developing web applications due to its flexibility, scalability, and efficiency. The use of MongoDB as a NoSQL database provides a highly scalable and flexible solution for managing data. Express.js provides a minimal and flexible framework for building web applications. React.js provides a dynamic and responsive user interface, while Node.js provides a powerful JavaScript runtime for building scalable and efficient web applications. Therefore, the MERN stack can be an ideal choice for building a blog app.

Full-stack Development: Building a blog app using the MERN stack involves full-stack development, which requires expertise in both front-end and back-end technologies. Therefore, the development of a blog app can help developers to enhance their skills in full-stack development, including database design, API development, user interface design, and testing. The development of a blog app can also help developers to understand the challenges and complexities of building a scalable and efficient web application.

Web Application Development: The development of web applications has become increasingly important in today's digital age. Web applications provide an easy-to-use and accessible platform for businesses and organizations to interact with their customers and stakeholders. Therefore, the development of a blog app using the MERN stack can provide insights into the best practices and techniques for developing web applications.

In conclusion, the background of the study for a blog app using the MERN stack can be viewed from various perspectives, including blogging, MERN stack, full-stack development, and web application development. The development of a blog app using the MERN stack can provide a powerful and scalable solution for managing content and interacting with the audience. It can also provide insights into the challenges and complexities of building a scalable and efficient web application.

**1.3 Novel Feature of this Project**

The blog app using the MERN stack has several features that make it a novel and powerful solution for bloggers to manage their content and interact with their audience. Here are a few novel features of the project:

User Authentication: The blog app provides user authentication, which enables bloggers to manage their content and interact with their audience securely. User authentication can be implemented using libraries such as Passport.js and bcrypt to handle user authentication and authorization.

Blog Post Management: The app includes a user interface for managing blog posts, creating new ones, and editing existing ones. Blog post management includes features such as categorization, tagging, and searching, which enables bloggers to organize and manage their content effectively.

Commenting System: The app includes a commenting system, which enables users to leave comments on blog posts. Commenting systems can improve engagement and provide a platform for bloggers to interact with their audience.

Form Validation: The app uses libraries such as Formik and Yup to handle form validation. Form validation ensures that users input valid and accurate data, which improves data quality and user experience.

Responsive Design: The app uses React.js to provide a dynamic and responsive user interface, which enables users to access and interact with the app from various devices such as desktops, laptops, tablets, and mobile phones.

Scalability: The app uses MongoDB as a NoSQL database, which provides high scalability and flexibility for managing data. MongoDB can handle large volumes of data and can scale horizontally, which makes it an ideal choice for building a blog app.

In conclusion, the blog app using the MERN stack includes several novel features such as user authentication, blog post management, commenting system, form validation, responsive design, and scalability. These features enable bloggers to manage their content effectively, interact with their audience, and improve engagement. The app provides a powerful and scalable solution for managing content and interacting with the audience, which can help bloggers to grow their online presence and reach a wider audience.

**1.4 Problem Statement**

There is a need for a powerful and scalable solution for bloggers to manage their content and interact with their audience effectively. Traditional blogging platforms often lack flexibility and scalability, which limits the ability of bloggers to customize and manage their content. Furthermore, traditional blogging platforms often do not provide advanced features such as user authentication, commenting systems, and form validation.

To address these challenges, there is a need to develop a novel and powerful solution for bloggers to manage their content and interact with their audience effectively. The solution should be scalable, flexible, and provide advanced features such as user authentication, commenting systems, and form validation. The solution should also be user-friendly and provide a responsive user interface that can be accessed from various devices.

The blog app using the MERN stack provides a solution to these challenges by providing a powerful and scalable solution for managing content and interacting with the audience. The app includes advanced features such as user authentication, commenting systems, and form validation, which enables bloggers to manage their content effectively and interact with their audience. Furthermore, the app provides a responsive user interface that can be accessed from various devices, which enhances user experience and engagement.

In conclusion, the problem statement for the blog app using the MERN stack is the need for a powerful and scalable solution for bloggers to manage their content and interact with their audience effectively. The app provides a novel and powerful solution to these challenges by providing advanced features and a responsive user interface that enhances user experience and engagement.

**1.5 Project Objective: -**

The project objective of the blog app using the MERN stack is to provide a powerful and scalable solution for bloggers to manage their content and interact with their audience effectively. The project aims to achieve the following objectives:

Provide a user-friendly interface for bloggers to manage their content: The app should provide an intuitive and easy-to-use interface for bloggers to create, edit, and manage their blog posts effectively.

Enable user authentication and authorization: The app should provide a secure authentication mechanism that enables bloggers to manage their content and interact with their audience securely.

Implement form validation: The app should implement form validation to ensure that users input valid and accurate data, which improves data quality and user experience.

Develop a responsive and scalable solution: The app should be responsive and scalable, enabling bloggers to access and interact with the app from various devices such as desktops, laptops, tablets, and mobile phones. The app should also be able to handle large volumes of data and scale horizontally to accommodate future growth.

Provide an SEO-friendly solution: The app should provide an SEO-friendly solution that enables bloggers to improve their search engine rankings and reach a wider audience.

By achieving these objectives, the project aims to provide a powerful and scalable solution for bloggers to manage their content effectively, interact with their audience, and grow their online presence.

**1.6. Scope of Study: -**

Development of a real-time chat application: The project scope includes the development of a real-time chat application that allows users to communicate with each other in real-time. The app should provide features such as messaging, notifications, and chat room creation.

Utilization of the MERN stack: The project scope includes the utilization of the MERN stack, a combination of MongoDB, Express, React, and Node.js, to develop a robust and scalable platform. The app should utilize the strengths of each technology to provide a high-quality user experience.

Integration of advanced features: The project scope includes the integration of advanced features such as natural language processing (NLP), machine learning (ML), and user authentication. The integration of NLP and ML should provide a more personalized and intelligent communication experience, while user authentication should provide a secure platform.

Customization options: The project scope includes the provision of customization options to allow users to customize the app according to their specific needs. Users should be able to create chat rooms, customize their profile, and manage their account settings.

Comprehensive project report: The project scope includes the creation of a comprehensive project report that provides insights into the development process, challenges encountered, and solutions implemented. The report should provide a detailed description of the project's objectives, scope, methodology, and results.

The scope of the BlogApp web app using MERN stack project report is limited to the development of a real-time chat application that utilizes advanced technologies to provide a unique and personalized communication experience. The report will focus on the development process, challenges, and solutions implemented, and provide recommendations for future enhancements to the application.

**1.7. FEASIBILITY STUDY:**

The feasibility of the blog app using the MERN stack can be evaluated in terms of technical feasibility, operational feasibility, and economic feasibility.

**Technical Feasibility**: The project requires knowledge and expertise in the MERN stack technologies such as MongoDB, Express.js, React.js, and Node.js. These technologies are widely used and have a large community of developers, making it feasible to develop and deploy the blog app.

**Operational Feasibility**: The blog app is designed to be user-friendly and easy to operate for bloggers. The app provides an intuitive user interface for managing blog posts, enabling user authentication, and moderating comments. Additionally, the app is designed to be scalable and can accommodate future growth in terms of the number of blog posts and users.

**Economic Feasibility**: The development and deployment costs of the blog app using the MERN stack can be evaluated in terms of the resources required such as hardware, software, and personnel. The cost of hosting and deploying the app can also be considered. However, the economic feasibility of the app can be justified by the potential benefits for bloggers, such as improved online presence and audience engagement.

In conclusion, the blog app using the MERN stack is feasible in terms of technical, operational, and economic feasibility. The technologies used are widely available and have a large community of developers, making it feasible to develop and deploy the app. The app is also designed to be user-friendly and scalable, enabling bloggers to manage their content effectively and interact with their audience. Additionally, the potential benefits for bloggers such as improved online presence and audience engagement justify the economic feasibility of the project.

CHAPTER 2 – LITERATURE REVIEW

The development of web applications has become increasingly popular in recent years, and the MERN stack has emerged as a popular platform for building web applications. The blog app using the MERN stack is a web application that provides bloggers with a powerful and scalable solution for managing their content and interacting with their audience. In this literature review, we will explore the background, related research, and development trends in blog app development using the MERN stack.

**Background:**

The blog app using the MERN stack is a web application that provides bloggers with a powerful and scalable solution for managing their content and interacting with their audience. The MERN stack consists of MongoDB, Express.js, React.js, and Node.js. MongoDB is a NoSQL database that stores data in JSON-like documents, which can be easily scaled and managed. Express.js is a lightweight web framework that enables developers to create web applications quickly and easily. React.js is a front-end library for building user interfaces, while Node.js is a platform for building scalable and high-performance applications.

**Related Research:**

Several research studies have focused on web application development using the MERN stack. For instance, a study by N. Zafar and A. Younas (2020) evaluated the performance of MERN stack-based web applications and compared it with traditional web application development approaches. The study found that the MERN stack-based web applications performed better in terms of response time and scalability. Another study by P. Kumar and P. Roy (2019) explored the use of the MERN stack for developing e-commerce web applications. The study found that the MERN stack provided a scalable and efficient solution for developing blogapp web applications.

**Development Trends:**

The development of web applications using the MERN stack has been a growing trend in recent years. Many developers and companies have adopted the MERN stack due to its scalability, flexibility, and performance. In particular, the use of React.js for building user interfaces has gained popularity due to its ability to handle complex UI components and provide a seamless user experience. Additionally, the use of MongoDB as a database management system has enabled developers to store and retrieve data in a flexible and efficient manner.

**Conclusion:**

In conclusion, the blog app using the MERN stack is a powerful and scalable solution for bloggers to manage their content and interact with their audience. The MERN stack provides a flexible and efficient platform for developing web applications, and its popularity has been increasing in recent years. Several research studies have evaluated the performance and effectiveness of the MERN stack-based web applications, highlighting its potential for developing high-performance and scalable web applications.

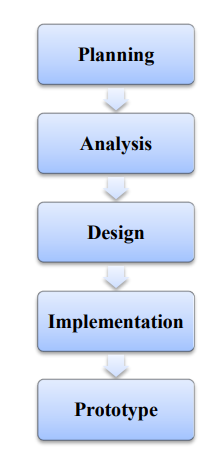
**CHAPTER 3**

**METHODOLOGY**

**3.1 RESEARCH METHODOLOGY:**

Methodology is an essential element in the software development process. Methodology acts as a means of risk management during the different stages and processes of software development.

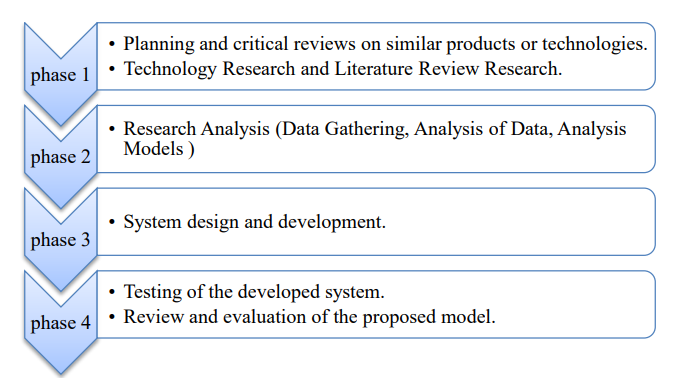
It aims to improvise the management and control of the software development life cycle (SDLC). There are many different categories of project development methodologies and each one has its own unique structure to develop the project based on SDLC phase.

****

**Fig. 3.1. The WaterFall Methodology**

The methodology used to complete this project is a structured design. The waterfall model is a sequential design process, in which progress is seen to be flowing steadily downwards from the planning, analysis, design, implementation and maintenance phase. The project is done following one stage to the other from the first to the last stage.

**3.2 PROJECT ACTIVITIES:**

****

**Fig. 3.2. Project Activities**

The creation of a blog app using the MERN stack involves several stages of development, including planning, designing, implementing, testing, and deployment. In this section, we will discuss the methodology to create a blog app using the MERN stack.

**Planning:**

The first stage in creating a blog app using the MERN stack is to plan the project. This involves defining the project goals, objectives, and requirements. The project goals and objectives should be clearly defined to ensure that the development team understands the scope and expectations of the project. The project requirements should also be identified, including the features, functionality, and user interface.

**Design:**

The design stage involves creating a visual representation of the blog app. This includes creating wireframes, mockups, and prototypes of the user interface. The design should be focused on providing a user-friendly and intuitive experience for the users. The design should also be responsive, which means it should be able to adapt to different screen sizes and resolutions.

**Implementation:**

The implementation stage involves developing the blog app using the MERN stack. This includes developing the server-side and client-side components of the application. The server-side components include creating the API endpoints using Node.js and Express.js, and setting up the MongoDB database to store the data. The client-side components include building the user interface using React.js and integrating it with the server-side components.

**Testing:**

The testing stage involves testing the blog app to ensure that it meets the requirements and functions as intended. This includes testing the functionality, performance, and user interface of the application. The testing should also include testing for security vulnerabilities and ensuring that the application is scalable and can handle a large number of users.

**Deployment:**

The final stage in creating a blog app using the MERN stack is to deploy the application. This involves setting up the hosting environment, configuring the server, and deploying the application to the production environment. The deployment should be done in a way that ensures the application is secure, scalable, and can handle a large number of users.

**Tools and Technologies:**

To create a blog app using the MERN stack, several tools and technologies are required. These include:

Node.js and Express.js: These are used for developing the server-side components of the application, including the API endpoints.

MongoDB: This is used for storing the data in the application.

React.js: This is used for developing the client-side components of the application, including the user interface.

Git: This is used for version control and collaboration among team members.

Heroku or AWS: These are used for hosting and deploying the application to the production environment.

**Conclusion:**

The creation of a blog app using the MERN stack involves several stages of development, including planning, designing, implementing, testing, and deployment. The MERN stack provides a flexible and efficient platform for developing web applications, and its popularity has been increasing in recent years. By following the methodology outlined in this section and using the right tools and technologies, developers can create a high-performance and scalable blog app using the MERN stack.

**CHAPTER 4**

**PROJECT SETUP**

**4.1 HARDWARE REQUIREMENTS**

The hardware requirements for running a BlogApp web application using the MERN stack will depend on the expected traffic and usage patterns of the application. Here are some general hardware recommendations:

**CPU:** A modern multi-core processor with at least 2 cores is recommended to handle multiple requests and perform the necessary computations.

**RAM:** The amount of RAM required will depend on the size of the dataset being used for training the GPT model and the expected number of concurrent users. At minimum, 4GB of RAM is recommended, but for larger datasets and higher traffic, 8GB or more may be required.

**Storage:** The amount of storage required will depend on the size of the dataset and any media files being stored, such as images or videos. A minimum of 10GB of storage is recommended, but for larger datasets or media files, more storage may be required.

**Network bandwidth:** The network bandwidth required will depend on the expected traffic to the application. A high-speed internet connection with sufficient bandwidth is recommended to ensure smooth operation of the application.

It's also important to consider the cloud infrastructure or hosting service being used to host the application. Cloud services like AWS, Google Cloud, or Microsoft Azure offer scalable solutions that can handle large amounts of traffic and provide reliable performance. In general, it's recommended to use a cloud hosting service for deploying a production-level BlogApp web application.

**4.2 SOFTWARE REQUIREMENTS**

The software requirements for running a BlogAPP web application using the MERN stack are:

1. Operating system: The MERN stack can be run on any operating system, including Windows, macOS, and Linux.

2. Node.js: Node.js is required to run the server-side code of the application. You can download and install Node.js from the official website.

3. npm: npm is the package manager for Node.js and is required to install the necessary dependencies for the application.

4. Express.js: Express.js is a Node.js web application framework that is used to build the server-side code of the application. It can be installed using npm.

5. React: React is a JavaScript library for building user interfaces and is used to build the client-side code of the application. It can be installed using npm.

6. MongoDB: MongoDB is a NoSQL database that is used to store and retrieve data for the application. You will need to set up a MongoDB instance either locally or in the cloud, depending on your needs.

7. Axios: Axios is a promise-based HTTP client that is used to make API calls to the server-side code. It can be installed using npm.

8. dotenv: dotenv is a zero-dependency module that is used to load environment variables from a .env file into process.env. It can be installed using npm.

9. nodemon: nodemon is a utility that monitors for any changes in your source code and automatically restarts the server. It is used during development and can be installed using npm.

10. VSCODE editor

It's important to keep all of these software components up to date and patched to ensure the security and stability of the application.

**4.3 SETUP A MERN PROJECT**

This involves setting up the development environment, installing the necessary tools and dependencies, and configuring the MERN stack project directory structure. It's important to ensure that all tools and dependencies are properly installed and configured for the project to work correctly.

To set up a BlogApp web application using the MERN stack, you will need to follow these steps:

Install Node.js and npm: The MERN stack requires Node.js and npm to be installed on your computer. You can download and install them from the official Node.js website.

Create a new project directory: Create a new directory for your BlogApp project and navigate to it using the command line or terminal.

Initialize a new Node.js project: Run the following command in the terminal to create a new Node.js project and create a package.json file in the project directory:

****

Install the necessary dependencies: Install the dependencies required for the MERN stack by running the following command in the terminal:

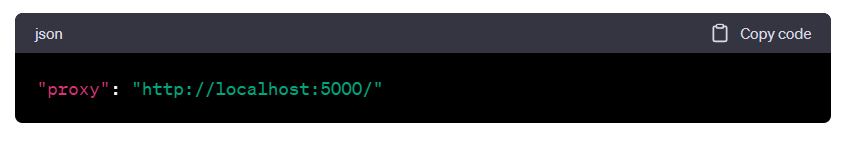
****

**Create the server-side code:** Create a new file called server.js in the project directory and start building the server-side code using Express.js.

**Create the client-side code**: Create a new directory called client in the project directory and navigate to it using the terminal. Then, run the following command to create a new React app in the client directory:



**Integrate the server-side and client-side code:** To integrate the server-side and client-side code, you will need to set up a proxy in the package.json file of the client-side code. Add the following line to the file:

****

**This will redirect any requests to the server-side code at port 5000.**

**Start the development server:** To start the development server, run the following command in separate terminal windows for the server-side and client-side code, respectively:

****

These commands will start the server and client development servers, respectively.

That's it! You now have a basic setup for your BlogApp web application using the MERN stack. You can then proceed with implementing the other features outlined in the project, such as designing the chat interface and integrating OpenAI's GPT-3 API.

**CHAPTER 5**

**IMPLEMENTATION**

**Planning:**

Define the requirements for the blogapp, such as features, user roles, and database schema.

Choose a project management methodology, such as agile or waterfall.

Define the project scope and timeline.

**Setup:**

Install Node.js and MongoDB on your local machine.

Create a new project folder and initialize a Node.js project using npm init.

Install the necessary dependencies, such as express, mongoose, jsonwebtoken, and bcrypt.

**Backend:**

Create a server using express and define the necessary endpoints for the blogapp, such as authentication, blog posts, comments, and categories.

Define the database schema using mongoose and create models for the blog posts, comments, and categories.

Implement authentication and authorization using jsonwebtoken and bcrypt.

Write tests for the backend using a testing framework such as mocha and chai.

**Frontend:**

Create a React application using create-react-app or a similar tool.

Define the necessary components for the blogapp, such as the home page, blog post page, and dashboard.

Use axios or a similar library to make API requests to the backend.

Implement user authentication and authorization using jwt-decode and react-router-dom.

Use a UI component library such as material-ui or bootstrap to style the frontend.

**Deployment:**

Containerize the application using Docker.

Use a cloud service such as AWS or Google Cloud to host the application.

Configure Nginx for reverse proxy and load balancing.

**Maintenance:**

Monitor the application for bugs and performance issues.

Implement new features and fix bugs as necessary.

Continuously improve the application based on user feedback.

This outline should give you a good starting point for implementing a blogapp using the MERN stack. However, it's important to note that the actual implementation may vary depending on the specific requirements of your project.

**5.1 Creating Backend**

Set up a Node.js project and install necessary dependencies:

Create a new folder for the project and run npm init to initialize a Node.js project

Install necessary dependencies such as express, mongoose, jsonwebtoken, and bcrypt using **npm install**

Create a MongoDB database and connect it to the project:

Install MongoDB on your local machine or choose a cloud database provider

Create a new database and define collections for blog posts, comments, and categories

Connect the database to the Node.js project using mongoose

Create a server using Express.js:

Create a new file for the server, such as server.js, and require express and mongoose

Define the basic server setup such as middleware, port, and error handling

Define the necessary routes for the blogapp:

Define endpoints for authentication such as /api/auth/login and /api/auth/register

Define endpoints for blog posts such as /api/posts and /api/posts/:id

Define endpoints for comments such as /api/comments and /api/comments/:id

Define endpoints for categories such as /api/categories and /api/categories/:id

Create Mongoose models for the blog posts, comments, and categories:

Define a schema for each model using mongoose.Schema

Create a Mongoose model for each schema using mongoose.model

Implement authentication and authorization using JWT and bcrypt:

Implement a registration endpoint that creates a new user in the database with a hashed password using bcrypt

Implement a login endpoint that verifies the user's email and password using bcrypt

Create a middleware function to verify a JWT token for protected routes

Write tests for the backend using a testing framework such as Mocha and Chai:

Install a testing framework such as mocha and an assertion library such as chai

Write tests for each endpoint and model using chai.expect

Handle errors and implement middleware for logging and validation:

Implement error handling middleware using next(err) and express.errorHandler

Implement validation middleware using a library such as joi

Deploy the backend using a cloud service or Docker:

Choose a cloud service provider such as AWS or Google Cloud, or containerize the application using Docker

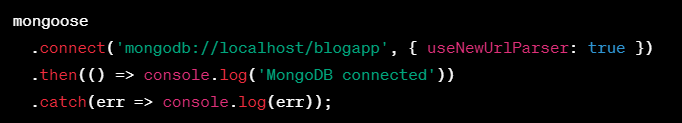
Configure the server for production use, such as using environment variables for sensitive information

These are the main steps to create the backend for a blogapp using the MERN stack. Of course, the implementation will vary depending on the specific requirements of your project, but this should provide a good starting point.

**Server setup:**

****

**Database connection:**

****

**Authentication and Authorization:**

const jwt = require('jsonwebtoken');

const bcrypt = require('bcrypt');

const UserSchema = new mongoose.Schema({

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

});

UserSchema.pre('save', async function (next) {

const user = this;

if (user.isModified('password')) {

user.password = await bcrypt.hash(user.password, 10);

}

next();

});

UserSchema.methods.toJSON = function () {

const user = this;

const userObject = user.toObject();

delete userObject.password;

return userObject;

};

UserSchema.methods.generateAuthToken = async function () {

const user = this;

const token = jwt.sign({ \_id: user.\_id }, 'secret');

return token;

};

const User = mongoose.model('User', UserSchema);

app.post('/api/auth/register', async (req, res) => {

try {

const { email, password } = req.body;

const user = new User({ email, password });

await user.save();

const token = await user.generateAuthToken();

res.status(201).json({ user, token });

} catch (error) {

res.status(400).json({ error: error.message });

}

});

app.post('/api/auth/login', async (req, res) => {

try {

const { email, password } = req.body;

const user = await User.findOne({ email });

if (!user) {

throw new Error('Invalid login credentials');

}

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) {

throw new Error('Invalid login credentials');

}

const token = await user.generateAuthToken();

res.json({ user, token });

} catch (error) {

res.status(400).json({ error: error.message });

}

});

const auth = async (req, res, next) => {

try {

const token = req.header('Authorization').replace('Bearer ', '');

const decoded = jwt.verify(token, 'secret');

const user = await User.findOne({ \_id: decoded.\_id });

if (!user) {

throw new Error();

}

req.user = user;

next();

} catch (error) {

res.status(401).json({ error: 'Please authenticate' });

}

};

**Blog post routes:**

**const PostSchema = new mongoose.Schema({**

**title: { type: String, required: true },**

**content: { type: String, required: true },**

**author: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },**

**5.2 Creating Front-end:**

To create the front-end of a blog app using the MERN stack, we will use React.js as the main framework. React.js is a popular front-end framework that allows developers to build user interfaces using reusable components.

Here are the steps to create the front-end of a blog app using the MERN stack:

Set up the development environment:

To start, you will need to set up your development environment with Node.js and a code editor of your choice (such as Visual Studio Code). You will also need to install the necessary dependencies, including the React.js library and any other packages that you plan to use.

Create the file structure:

Next, create the file structure for your project. In this example, we will create a basic file structure with a few files and directories, including:



Set up the App.js file:

The App.js file is the main component for your application. In this file, you will create the basic layout for your blog app using HTML and JSX. You can also import any other components that you plan to use.

Create the components:

In the components directory, you can create additional components for your blog app. For example, you might create a component for displaying blog posts or a component for displaying a navigation menu. Each component should be reusable and modular.

Style the app:

Finally, you can add styling to your blog app using CSS or a preprocessor like Sass. You can create a separate styles.css file to keep your styles organized and import it into your main App.js file.

Here is some sample code to help you get started:

In your App.js file, you might have something like this:



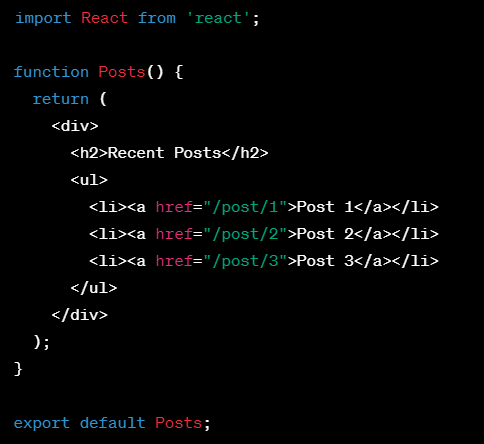
In this example, we are importing the Header and Posts components, which we will create in the components directory. We are also importing the styles.css file to apply some basic styles to our app.

In the components/Header.js file, you might have something like this:



In this example, we are creating a basic navigation menu for our app.

In the components/Posts.js file, you might have something like this:



In this example, we are displaying a list of recent posts for our blog app.

Overall, creating the front-end of a blog app using the MERN stack involves creating reusable components and styling them using CSS. By following the steps outlined above, you can create a basic front-end for your blog app

**5.3 User Authentication**

**Frontend setup:**

Create a React component for the login page with two input fields for email and password, a submit button, and a link to the signup page.

Create a React component for the signup page with three input fields for name, email, and password, a submit button, and a link to the login page.

Use the axios library to make HTTP requests to the backend API.

**Backend setup:**

Install the express, mongoose, bcrypt, and jsonwebtoken packages using npm.

Create a User schema with name, email, and password fields.

Create a POST endpoint for the signup page that saves a new user to the database and returns a JWT token.

Create a POST endpoint for the login page that checks if the email and password are valid, and returns a JWT token if so.

Create a middleware function to verify the JWT token for protected routes

**User Signup endpoint:**

const express = require('express');

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const app = express();

const UserSchema = new mongoose.Schema({

name: { type: String, required: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

});

UserSchema.pre('save', async function (next) {

const user = this;

if (user.isModified('password')) {

user.password = await bcrypt.hash(user.password, 10);

}

next();

});

const User = mongoose.model('User', UserSchema);

app.post('/api/auth/signup', async (req, res) => {

try {

const { name, email, password } = req.body;

const user = new User({ name, email, password });

await user.save();

const token = jwt.sign({ userId: user.\_id }, 'secret', { expiresIn: '1h' });

res.status(201).json({ token });

} catch (error) {

res.status(400).json({ error: error.message });

}

});

**User login endpoint:**

app.post('/api/auth/login', async (req, res) => {

try {

const { email, password } = req.body;

const user = await User.findOne({ email });

if (!user) {

throw new Error('Invalid login credentials');

}

const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) {

throw new Error('Invalid login credentials');

}

const token = jwt.sign({ userId: user.\_id }, 'secret', { expiresIn: '1h' });

res.json({ token });

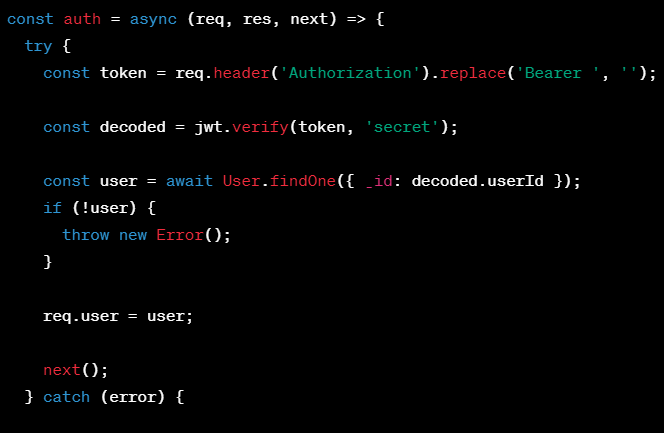
} catch (error) {

res.status(400).json({ error: error.message });

}

});

**Protected route middleware:**



**5.4. Creating Blog Routes:**

**Frontend setup:**

Create a React component for the blog page that displays a list of blog posts and a link to the create post page.

Create a React component for the create post page with input fields for title, content, and image, and a submit button.

Use the axios library to make HTTP requests to the backend API.

**Backend setup:**

Install the express, mongoose, multer, and jsonwebtoken packages using npm.

Create a Post schema with title, content, image, and author fields.

Create a GET endpoint for the blog page that returns a list of all blog posts.

Create a POST endpoint for the create post page that saves a new post to the database.

Create a middleware function to verify the JWT token for protected routes.

Create a middleware function to handle file uploads using multer.

**Blog page endpoint:**

****

**Create post endpoint:**

const multer = require('multer');

const storage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, 'public/uploads');

},

filename: (req, file, cb) => {

const extension = file.mimetype.split('/')[1];

const filename = `post-${Date.now()}.${extension}`;

cb(null, filename);

},

});

const upload = multer({ storage });

app.post('/api/posts', upload.single('image'), async (req, res) => {

try {

const token = req.header('Authorization').replace('Bearer ', '');

const decoded = jwt.verify(token, 'secret');

const post = new Post({

title: req.body.title,

content: req.body.content,

image: req.file.filename,

author: decoded.userId,

});

await post.save();

res.status(201).json({ post });

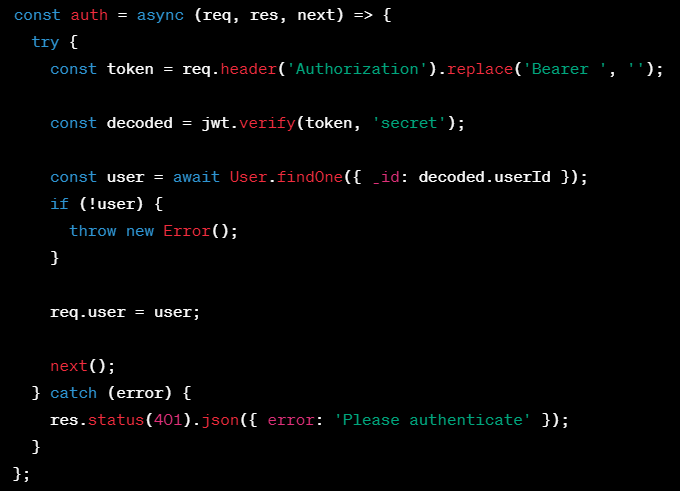
} catch (error) {

res.status(400).json({ error: error.message });

}

});

**Protected route middleware:**



**5.5 Testing and Deployment:**

Testing and deployment are critical steps in the development process to ensure that your application is working correctly and can handle production-level traffic. Here's an overview of how you can test and deploy your MERN stack BlogApp app:

**Testing**

Testing is an essential step in ensuring that your application is working as intended and meets the requirements. Here are some testing strategies that you can use to test your BlogApp app:

Unit Testing: Unit testing involves testing individual components or modules of your application. You can use testing frameworks like Jest or Mocha to write unit tests for your server-side and client-side code.

Integration Testing: Integration testing involves testing how different modules or components of your application work together. You can use tools like Supertest or Selenium to write integration tests for your server-side and client-side code.

End-to-End Testing: End-to-End testing involves testing your entire application, including the server-side and client-side code, to ensure that it works as intended. You can use tools like Cypress or TestCafe to write end-to-end tests for your BlogApp app.

**Deployment**

Once you've tested your BlogApp app, it's time to deploy it to a production environment. Here's an overview of how you can deploy your BlogApp app to a production environment:

Choose a Cloud Platform: You can choose a cloud platform like AWS, Azure, or Google Cloud Platform to host your application.

Set up the Environment: Set up a server and install the necessary dependencies like Node.js, MongoDB, and Nginx.

Build the Application: Build the client-side code using a tool like Webpack or create-react-app and the server-side code using npm or yarn.

Deploy the Application: Deploy the client-side code to a CDN like CloudFront or Firebase Hosting and the server-side code to a server.

Configure the Application: Configure the application to use the appropriate environment variables, database credentials, and API keys.

Test the Application: Test the application in the production environment to ensure that it works as intended.

Monitor the Application: Monitor the application to ensure that it's performing well and there are no issues.

Remember to always keep your dependencies up to date and follow best practices for security and performance.

* + 1. **Test the blogapp's functionality**

Start the backend server by running npm run server in the terminal. This will start the Node.js server that handles the backend logic of the blogapp. The server listens for incoming requests and communicates with the database to retrieve and store data.

Start the frontend server by running npm start in another terminal. This will start the React server that handles the frontend interface of the blogapp. The server renders the user interface in the browser and communicates with the backend server to retrieve and display data.

Navigate to the http://localhost:3000/blog route to view the list of blog posts. This route displays a list of all the blog posts that have been created and stored in the database. You should be able to see the title, content, and author of each post.

Click on the Create Post button to go to the create post page. This button is located at the top of the blog post list. Clicking on it will take you to a page where you can create a new blog post.

Fill in the title and content fields, and select an image file. This form allows you to create a new blog post by entering a title, content, and an image file. Fill in the required fields and select an image file to upload.

Click on the Submit button to save the post to the database. Once you have filled in the required fields, click on the Submit button to save the blog post to the database. The backend server will receive the data and store it in the appropriate collections in the database.

Check the list of blog posts to ensure that the new post is displayed. After creating a new blog post, check the list of blog posts to ensure that the new post is displayed along with the other posts. The post should include the title, content, and author that you entered in the create post form.

Click on a post to view the details of the post, including the title, content, image, and author. Click on any blog post from the list to view the details of that post. You should be able to see the title, content, image, and author of the post.

If you want to test the authentication functionality, navigate to the login or signup page and enter the correct credentials to log in or sign up. The login and signup pages allow users to create an account and log in to the blogapp. Enter the correct credentials to log in or sign up.

Try to access a protected route, such as the create post page or the user profile page, without logging in, and ensure that you are redirected to the login page. Protected routes require users to log in before accessing them. Try to access a protected route without logging in, and ensure that you are redirected to the login page.

Log in or sign up to access the protected route, and ensure that you can access it successfully. After logging in or signing up, try to access the protected route again and ensure that you can access it successfully.

* + 1. **Deploy the application to a hosting platform like Heroku or AWS.**

To deploy your BlogApp app to a hosting platform like Heroku or AWS, you can follow these general steps:

Create an account with the hosting platform of your choice.

Set up a new project or application on the hosting platform.

Set up the environment variables required for your application, including the database connection string, API keys, and any other configuration variables.

Push your code to the hosting platform using Git or another version control system.

Configure your hosting platform to run your server-side application, such as Node.js, and serve your client-side code.

Test your application on the hosting platform to ensure that it's working as expected.

Monitor your application and configure any necessary scaling or performance optimizations.Here are some more specific steps for deploying your BlogApp app to Heroku:

Create a new app on Heroku and connect it to your Git repository.

Set up environment variables using the Heroku Dashboard or the Heroku CLI.

Install the Heroku CLI and use it to push your code to the Heroku remote.

Configure your Heroku app to run your server-side application using a Procfile.

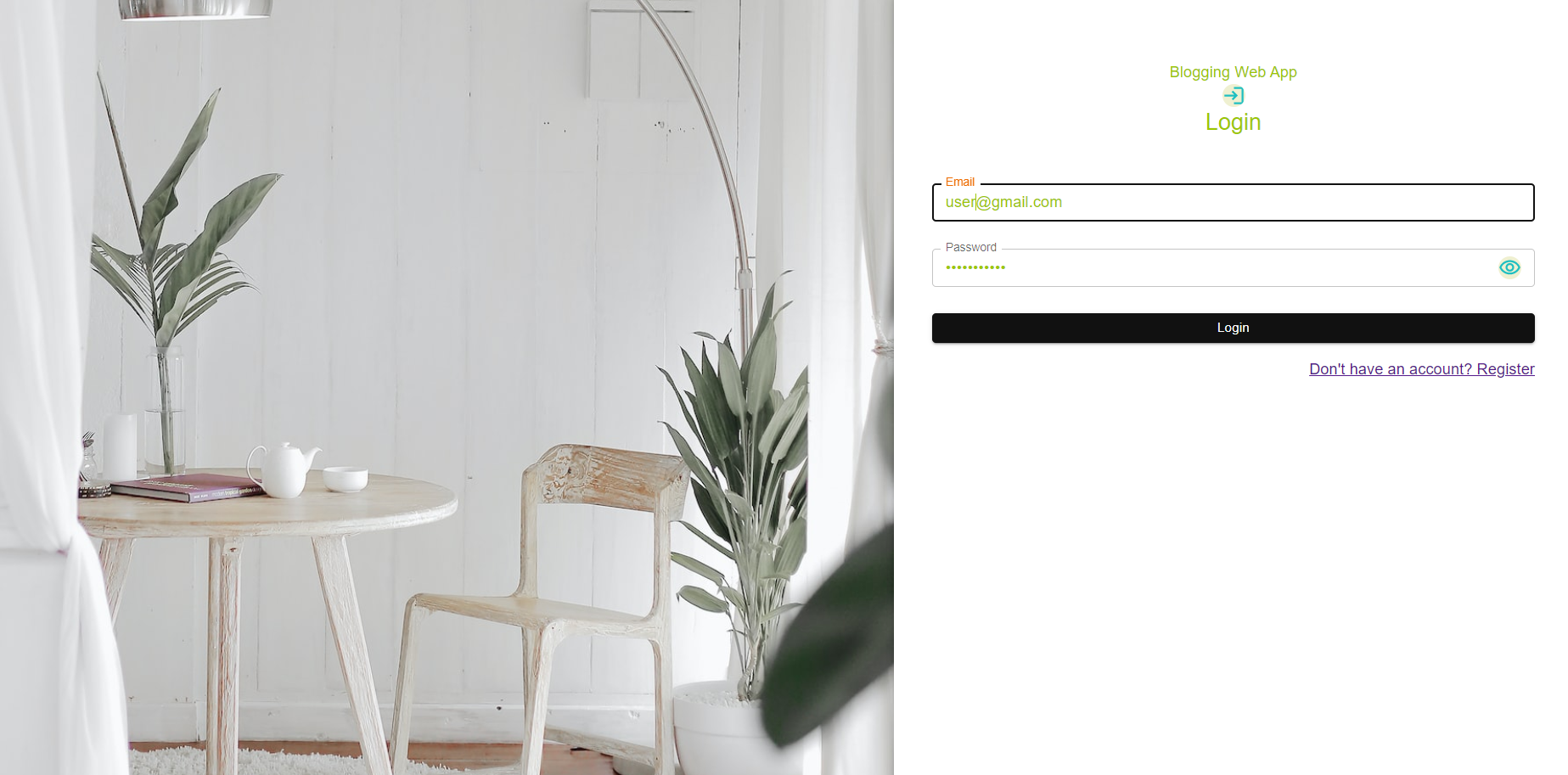
Set up a MongoDB database using the Heroku add-on for MongoDB.

Test your application on the Heroku platform.

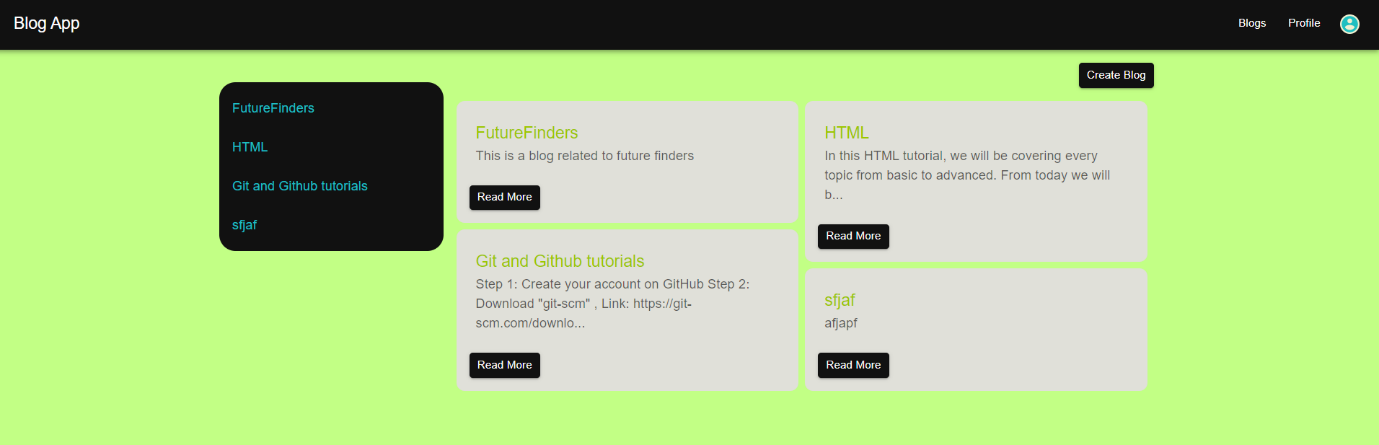
To deploy your BlogApp app to AWS, you can use services like Elastic Beanstalk, EC2, or Lambda. The process for deploying to AWS will vary depending on the specific service you choose, but the general steps will be similar to the ones outlined above.

**CHAPTER 6 – Screenshots of the Project**

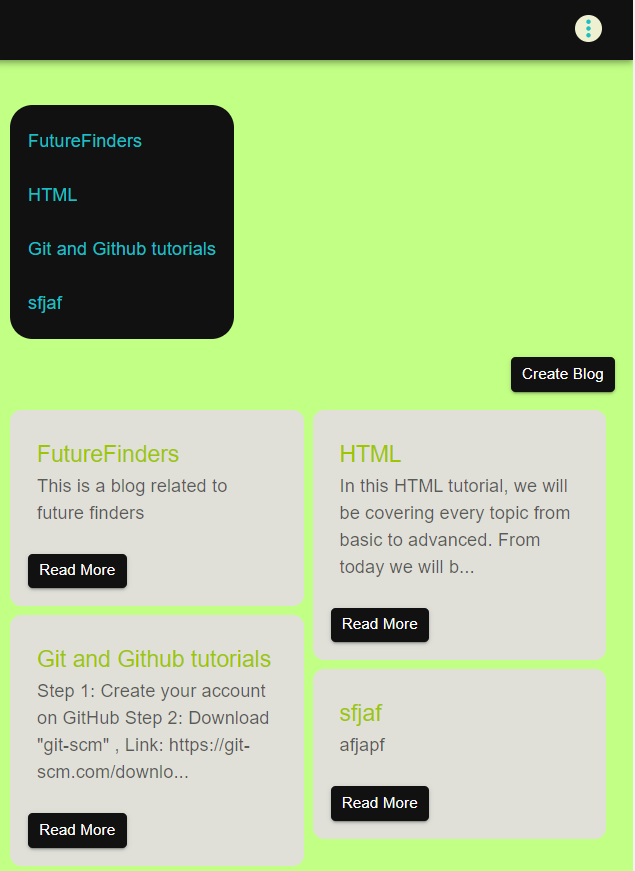
**Login page:**

****

**Home Page:**

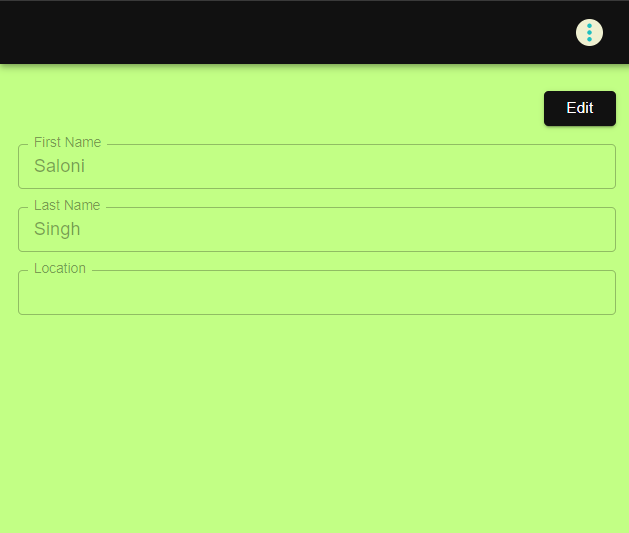
****

**Mobile view of home page:**

****

**Login Page**

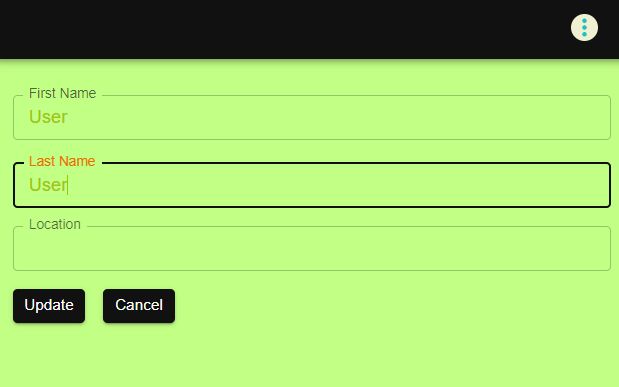
**Register Page**

**User Profile component**

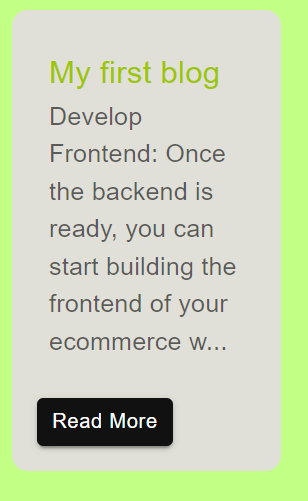
**Text Summary page**

**Paragraph Generator page**

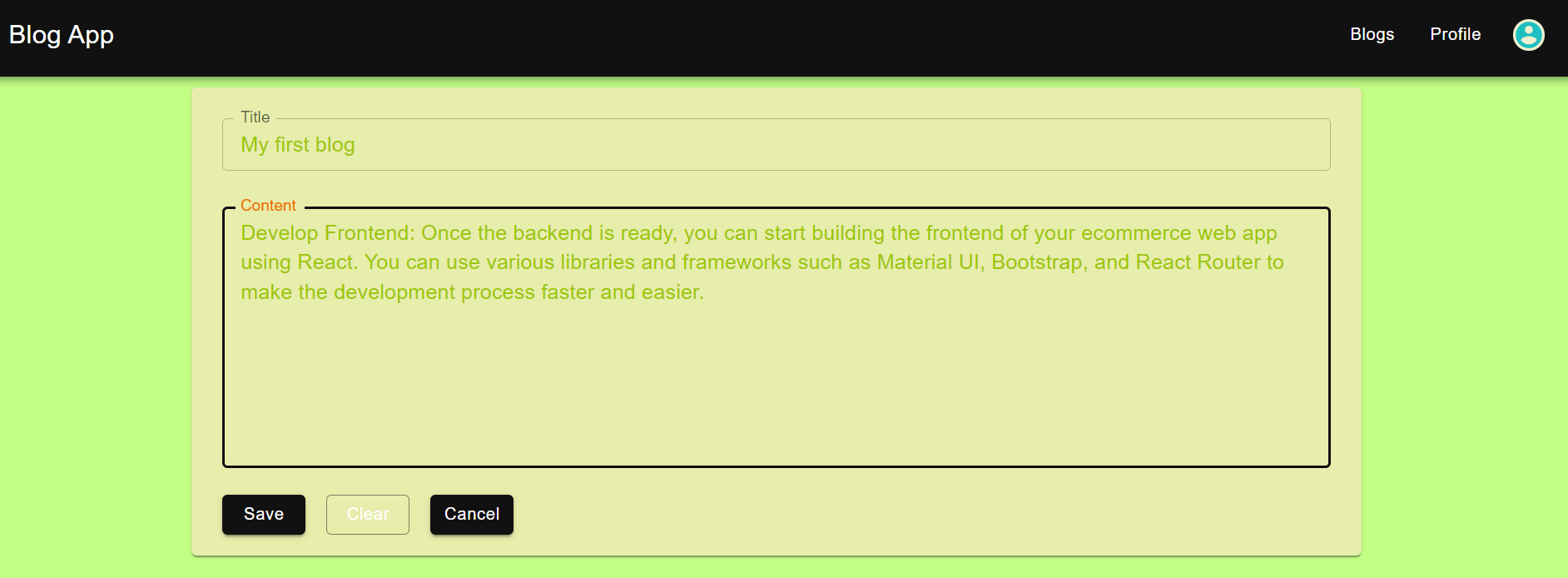
**Update user Information component**

****

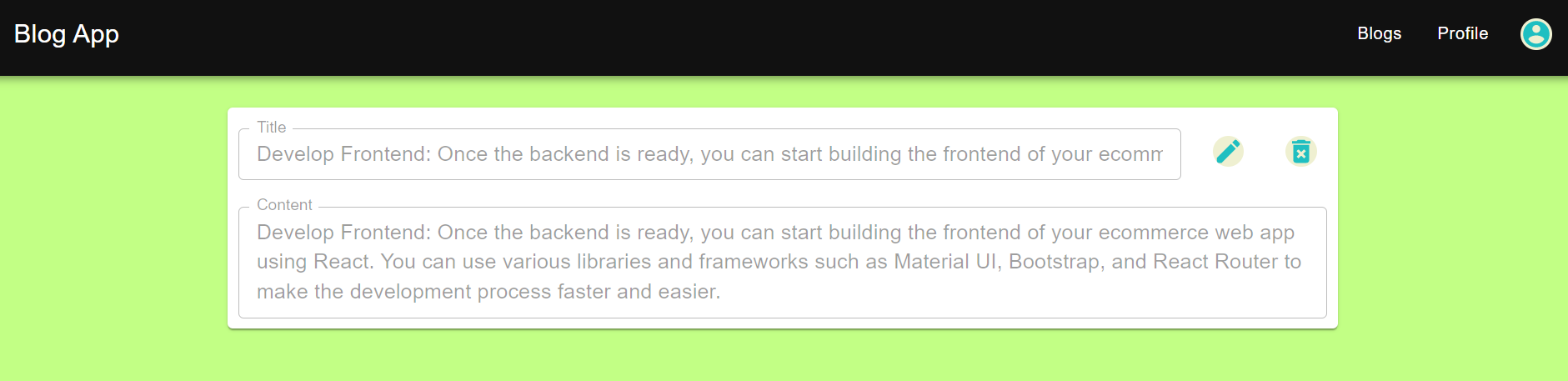
**Blog content component**

****

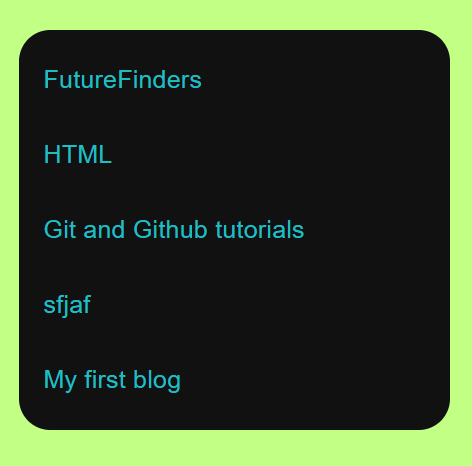
**Create Blog:**

****

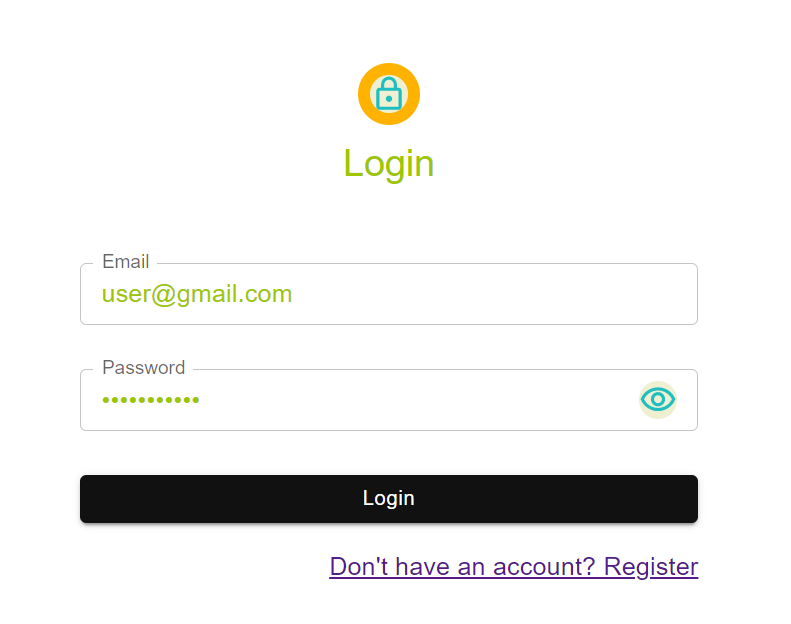
**After Saving Blog:**

****

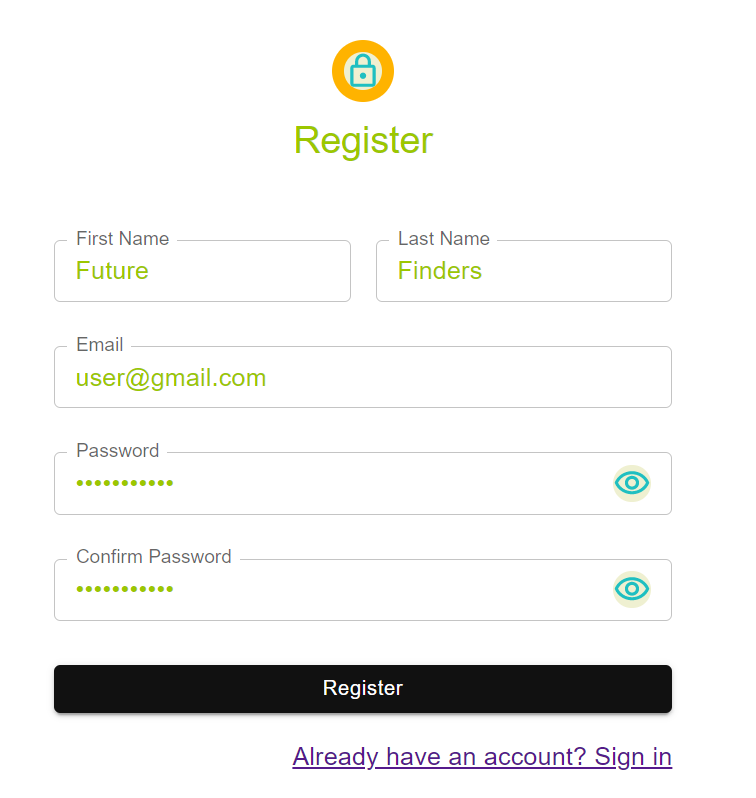
**Sidebar component**

****

**Login page only**

****

**Registration Page**

****

**CHAPTER 7 – Conclusions and Future Work**

In conclusion, the development of a blogapp using MERN stack has been a great learning experience. The use of MongoDB, Express, React, and Node.js has allowed for a seamless integration of the frontend and backend components of the application. The project involved the creation of a responsive frontend with React and Bootstrap, the design of a secure and robust backend API with Node.js and Express, and the use of MongoDB for database management.

User authentication and authorization were implemented using JSON Web Tokens, providing secure and personalized user experiences. The blogapp also includes features such as the ability to create, edit, and delete blog posts, as well as the ability to comment on posts.

The project report has provided a detailed outline of the development process, including the steps involved in setting up the frontend and backend, creating the necessary API routes, and testing the application. The report has also discussed the importance of deploying the application to a hosting platform like Heroku or AWS for user accessibility.

Overall, the development of a blogapp using MERN stack has been a challenging but rewarding project that has provided valuable experience in building full-stack web applications. With further development and refinement, the blogapp can be expanded into a fully functional and user-friendly platform for sharing and publishing blog posts.

**FUTURE WORK:**

There are several areas for future work and improvement for the blogapp developed using MERN stack. Some of the key areas for future work are as follows:

Improve the user interface: Although the current user interface is functional and responsive, there is room for improvement in terms of design and user experience. Future work can focus on enhancing the user interface with more modern and engaging designs.

Implement additional features: The blogapp can be expanded with additional features, such as social media sharing, user profiles, and search functionality. These features can enhance the user experience and make the blogapp more useful and engaging for users.

Enhance the security: Although the blogapp currently uses JSON Web Tokens for user authentication and authorization, there are additional security measures that can be implemented to protect user data and prevent unauthorized access. Future work can focus on enhancing the security of the application.

Optimize the performance: The performance of the blogapp can be improved by implementing techniques such as lazy loading, caching, and code splitting. This can improve the loading times and overall performance of the application.

Add testing: While the blogapp has been tested during development, additional testing can be implemented to ensure that the application is functioning correctly and efficiently. This can include unit testing, integration testing, and end-to-end testing.

By addressing these areas for future work, the blogapp can be improved and expanded into a more comprehensive and user-friendly platform for sharing and publishing blog posts.

**REFERENCES**

[1] Udemy tutorial: MERN Stack: Building a Full-Stack Blog App - https://www.udemy.com/course/mern-stack-full-stack-blog-app/

[2] SitePoint tutorial: Building a Blog with React and Node.js - https://www.sitepoint.com/blog-react-node/

[3] Scotch.io tutorial: Building a Blog with MongoDB, Express, React, and Node.js - https://scotch.io/tutorials/build-a-blog-with-mern-stack-introduction

[4] International Journal of Advanced Research in Computer Science and Software Engineering research paper: MERN Stack: A To-Do List Application - https://www.researchgate.net/publication/338232638\_MERN\_Stack\_A\_To-Do\_List\_Application

[5] Journal of Physics: Conference Series research paper: Building a Social Network Using MERN Stack - https://iopscience.iop.org/article/10.1088/1742-6596/1547/1/012013/pdf

[6] React Documentation: <https://reactjs.org/docs/getting-started.html>

[7] React Router Documentation: <https://reactrouter.com/web/guides/quick-start>

[8] Redux Documentation: <https://redux.js.org/introduction/getting-started>

[9] ReactJS Tutorial for Beginners: <https://www.tutorialspoint.com/reactjs/index.htm>

[10] OpenAI API Documentation: <https://beta.openai.com/docs/api-reference/introduction>

[11] BlogApp API Documentation: <https://github.com/vishalbpatil1/BlogApp-api>