9530

St.MOTHER THERESA ENGINEERING COLLEGE

COMPUTER SCIENCE ENGINEERING

NM-ID: 41963ba4d71577abdd94d90919773019

REG NO: 953023104124

DATE:22-09-2025

Completed the project named as

Phase 3

FRONT END TECHNOLOGY
Blog Site with Comment Section

SUBMITTED BY,

M.Sudalai Mani

7708352610

Thesis: Building a Blog Site with Core Features, Data Storage, and Testing

Introduction

A blog site serves as a platform for content creators to share ideas, information, and updates. With the advent of modern web development tools and frameworks, creating a robust and scalable blog platform has become more accessible. This thesis will explore the process of building a blog site from scratch, focusing on the core features of the site, data storage solutions, testing methodologies, and version control with GitHub. Additionally, the integration of a comment section, which allows readers to engage with content, will be a central component of this project.

---

1. Project Setup

1.1 Choosing the Tech Stack

The first step in creating a blog site is selecting the appropriate technologies for the project. The chosen tech stack for this blog site will include:

- **Frontend:** React (JavaScript framework) for a dynamic, interactive user interface.
- **Backend:** Node.js with Express for building the RESTful API that will handle requests and data.
- **Database:** MongoDB for document-based storage to store blog posts and comments.
- **Version Control:** GitHub for source code management and collaboration.

1.2 Setting Up the Development Environment

Before beginning, we'll need to set up a development environment:

- Install **Node.js** and **npm**.
- Set up **React** using **Create React App**.
- Initialize the **Express** backend with necessary dependencies like express, mongoose, cors, and dotenv.
- Create a GitHub repository to manage the project's version control.

---

2. Core Features Implementation

2.1 Blog Post Management

The primary feature of the site will be the creation, reading, updating, and deletion (CRUD operations) of blog posts.

- **Frontend:** React components will be created to display the list of blog posts and individual blog post details.
- **Backend:** An Express API will handle the routing for creating, reading, updating, and deleting blog posts. These operations will interact with the MongoDB database.

2.2 Comment Section

A blog is incomplete without engagement, so we'll implement a comment section that allows users to post, edit, and delete comments.

- **Frontend:** A comment component will allow users to post and view comments. Each comment will be associated with a specific blog post.

- **Backend:** The API will handle storing and retrieving comments. Each comment will reference the blog post ID, and comments can be added, deleted, or updated.

2.3 User Authentication

User authentication will be necessary for certain actions like posting and moderating comments. This will be handled via JWT (JSON Web Tokens).

- **Frontend:** A login and registration page will be implemented.

- **Backend:** Express middleware will be used to verify JWT tokens for protected routes (e.g., adding a comment or creating a blog post).

---

3. Data Storage (Local State/Database)

3.1 Local State Management

For dynamic UI features, React's state management will handle data locally within the component hierarchy.

- **Blog Post Data:** Posts fetched from the backend will be stored in the React component state.

- **Comments Data:** Comments for a blog post will be stored in a state variable specific to that post's details page.

3.2 Database Integration

For persistent data storage, MongoDB will be used.

- **Blog Posts:** Each post will be stored as a document in a MongoDB collection, containing fields such as title, content, author, and timestamps.

- **Comments:** Comments will be stored in a separate collection with a reference to the post they belong to.

Database interaction will be managed using **Mongoose**, an Object Data Modeling (ODM) library for MongoDB and Node.js.

---

4. Testing Core Features

4.1 Frontend Testing (React)

Testing React components and functionality will ensure that the user interface behaves as expected.

- **Jest** will be used as the testing framework, along with **React Testing Library** for simulating user interactions and verifying component behavior.

- Test cases will be written for:

  - Rendering blog posts and comments.

  - Handling user input and interactions.

  - Validating the responsiveness of the design.

4.2 Backend Testing (Node.js/Express)

The backend will be tested using **Mocha** and **Chai**, which are popular testing libraries for Node.js.

- Test cases will include:
  - Verifying API routes (POST, GET, DELETE, PUT) for creating, retrieving, updating, and deleting blog posts and comments.
  - Ensuring JWT authentication is correctly implemented.

---

5. Version Control (GitHub)

5.1 Initializing the Repository

Once the development environment is set up, the project will be initialized as a Git repository. The following steps will be undertaken:

- Create a .gitignore file to exclude node_modules and other unnecessary files from being committed.
- Initialize the repository with git init and make the first commit.

5.2 Branching and Collaboration

Version control with GitHub enables collaboration and efficient management of changes.

- **Branches** will be used for features like user authentication, comment section, etc.
-

Pull requests will be created to review and merge changes into the main branch.

- **GitHub Actions** can be integrated to automatically run tests on each pull request.

Conclusion

This project will result in a fully functional blog website with the ability to create, view, update, and delete blog posts, along with an interactive comment section. The project will use modern technologies like React for the frontend, Express for the backend, and MongoDB for data storage. Core features will be tested to ensure stability, and version control using GitHub will enable smooth collaboration and code management. Through this project, the blog platform will not only serve as a content hub but also a space for reader engagement.