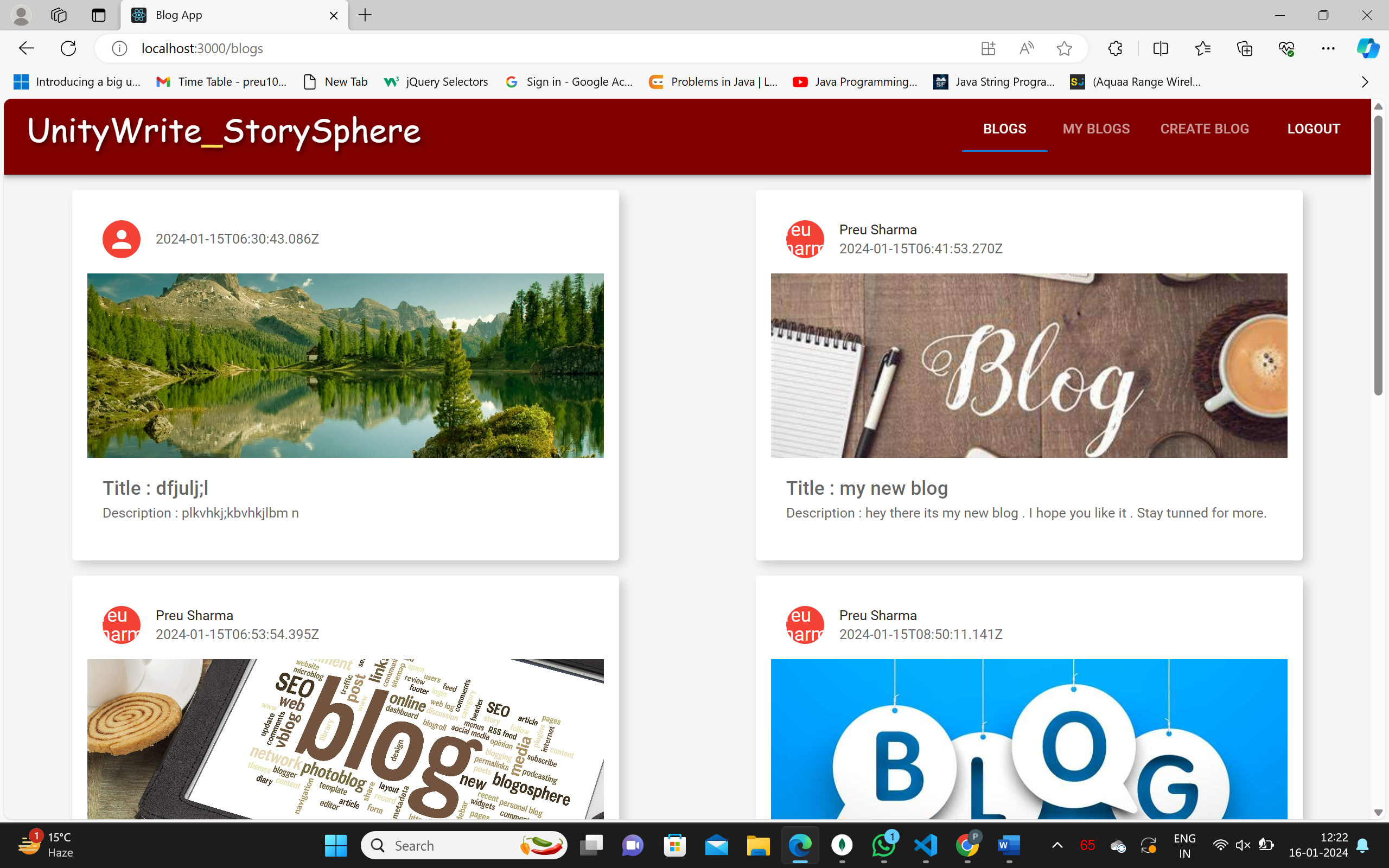
**CMS PROJECT ON BLOGGING WEBSITE**

**UNITYWRITE\_STORYSPHERE**

**AN INTERACTIVE BLOGGING WEBSITE WHERE USERS CAN CREATE, READ, UPDATE, DELETE BLOG POSTS**



**MADE BY-**

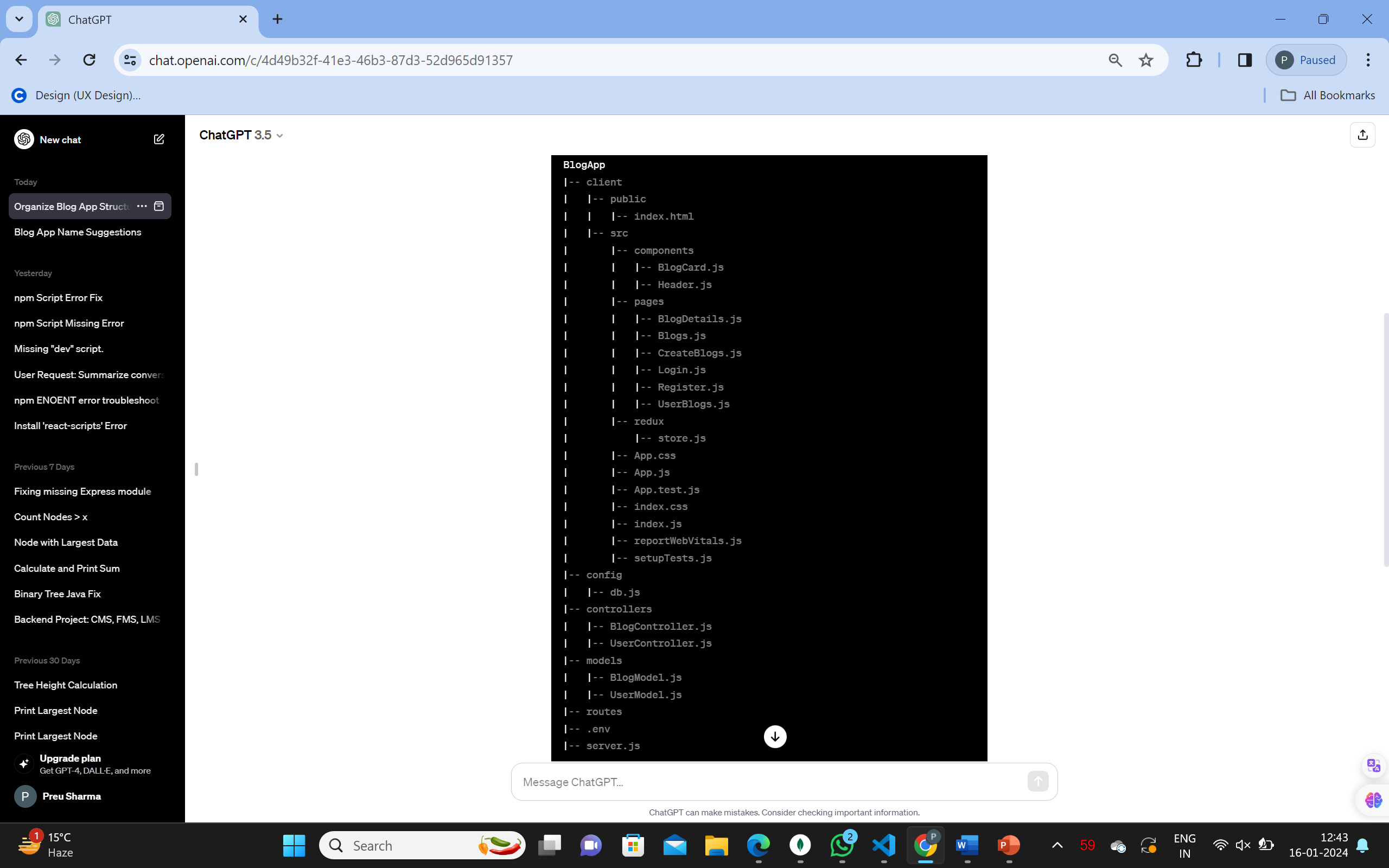
**NAME- PREU SHARMA**

**ROLL NO- 2110991061**

**GROUP-2**

***File Structure***

***Mern Stack Blog APP***



***FRONT-END***

***Sure, let's break down the key aspects of our React project:***

1. ***Folder Structure:***

* ***src/components: Contains React components, such as BlogCard.js and Header.js.***
* ***src/pages: Holds React components representing different pages like Blogs.js, UserBlogs.js, etc.***
* ***src/redux: Manages global state using Redux. Defines the authentication state in store.js.***
* ***src: The root folder contains the main application files.***

***2. Component: BlogCard.js***

* ***Displays a card for each blog with title, description, image, username, and time.***
* ***Provides options for editing and deleting if the user is the creator.***

***3. Component: Header.js***

* ***The app's header with the title "UnityWrite\_StorySphere" and navigation tabs.***
* ***Navigation tabs include "Blogs," "My Blogs," and "Create Blog."***

***4.Pages:***

* ***Blogs.js: Displays all blogs.***
* ***UserBlogs.js: Shows blogs created by the logged-in user.***
* ***CreateBlog.js: Form for creating a new blog.***
* ***BlogDetails.js: Displays details of a specific blog for editing.***

***5.Pages: Authentication***

* ***Login.js: Provides a form for users to log in.***
* ***Register.js: Allows users to register.***

***6.Redux:***

* ***store.js: Manages global state using Redux. Contains actions for login and logout.***

***7.App Structure (app.js):***

* ***Imports the necessary components and pages.***
* ***Configures the app's routes using react-router-dom.***
* ***Uses Redux for global state management.***
* ***Renders the Header component at the top.***
* ***Displays different pages based on the route.***

***8.Styling:***

* ***Minimal styling has been done using MUI (Material-UI) components.***
* ***Flexbox is used for layout in certain components.***
* ***CSS-in-JS approach with MUI's sx prop for styling.***

***9.Testing:***

* ***A basic test (app.test.js) is present, which currently checks if "learn react" text is present.***

***10.React Entry Point (index.js):***

* ***Renders the main App component inside a BrowserRouter.***
* ***Utilizes Provider to connect the Redux store with the React app.***

***11.Web Vitals:***

* ***A setup for reporting web vitals (performance metrics) is included in reportWebVitals.js.***

***12.Additional Files:***

* ***index.css: Contains global CSS styles (resetting margins and paddings).***
* ***setupTests.js: Configures testing libraries.***
* ***In summary, our project is a simple blog application with user authentication, allowing users to view, create, and edit blogs. Material-UI components are used for styling, and global state is managed using Redux.***

***BACK-END***

***Here's a simplified overview of our blog project's backend code:***

***1. Controllers***

***a) blogController.js***

***getAllBlogsController: Fetches all blogs, populates the user data, and sends the list as a response.***

***createBlogController: Creates a new blog, validates input, associates it with an existing user, and sends the new blog as a response.***

***updateBlogController: Updates an existing blog using its ID and sends the updated blog as a response.***

***getBlogByIdController: Retrieves a single blog by its ID and sends it as a response.***

***deleteBlogController: Deletes a blog by its ID, removes the blog from the associated user, and sends a success message.***

***userBlogControlller: Retrieves blogs associated with a specific user ID and sends them as a response.***

***b) userController.js***

***registerController: Registers a new user, validates input, checks for existing users, hashes the password, and sends the new user as a response.***

***getAllUsers: Fetches all users and sends the list as a response.***

***loginController: Handles user login, validates input, checks if the email is registered, verifies the password, and sends the user details upon successful login.***

***2. Models***

***a) blogModel.js***

***Defines the schema for a blog, including title, description, image, and a reference to a user.***

***b) userModel.js***

***Defines the schema for a user, including username, email, password, and an array of associated blogs.***

***3. Routes***

***a) blogRoutes.js***

***Defines routes for handling blog-related operations using the blog controller functions.***

***b) userRoutes.js***

***Defines routes for user-related operations using the user controller functions.***

***4. Config***

***a) db.js***

***Connects to the MongoDB database using the provided URL.***

***5. Environment Variables (.env file)***

***Contains the MongoDB connection URL and a development mode indicator.***

***6. Package.json***

***Lists project details, dependencies, and scripts for starting the server and client concurrently.***

***7. Server.js***

***Middleware Setup: Configures Express middleware like CORS, JSON parsing, and Morgan for logging.***

***Route Setup: Associates routes from userRoutes and blogRoutes with their respective prefixes.***

***Database Connection: Connects to the MongoDB database.***

***Error Handling: Includes handlers for unhandled promise rejections and uncaught exceptions.***

***Server Listening: Starts the server on the specified port.***