COLLEGE CODE: 8203

COLLEGE: A.V.C COLLEGE OF ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: 355D90D88AE7822824F49869175ED7E6

ROLL NO:23IT101

DATE:15/09/2025

Completed the project named as Phase 2

TECHNOLOGY PROJECT NAME: Blogging Platform

SUBMITTED BY,

NAME: SONASRI V

MOBILE NO: 7845869297

Tech Stack Selection:

Frontend:

- **React.js** – For building a responsive, dynamic, and interactive user interface.
- **Axios** – For making API requests from the frontend to the backend.

Backend:

- **Node.js** – Provides a fast, event-driven runtime environment for server-side operations.
- **Express.js** – A lightweight web framework to manage routes, middleware, and REST API logic.

Database:

- **MongoDB** – NoSQL database to store flexible JSON-like documents.
- **Mongoose** – ODM (Object Data Modeling) library for defining schemas, relationships, and validations.

Authentication & Security:

- **JWT (JSON Web Token)** – For secure user authentication and authorization.
- **bcrypt.js** – For securely hashing passwords before storing them.

Deployment & Hosting:

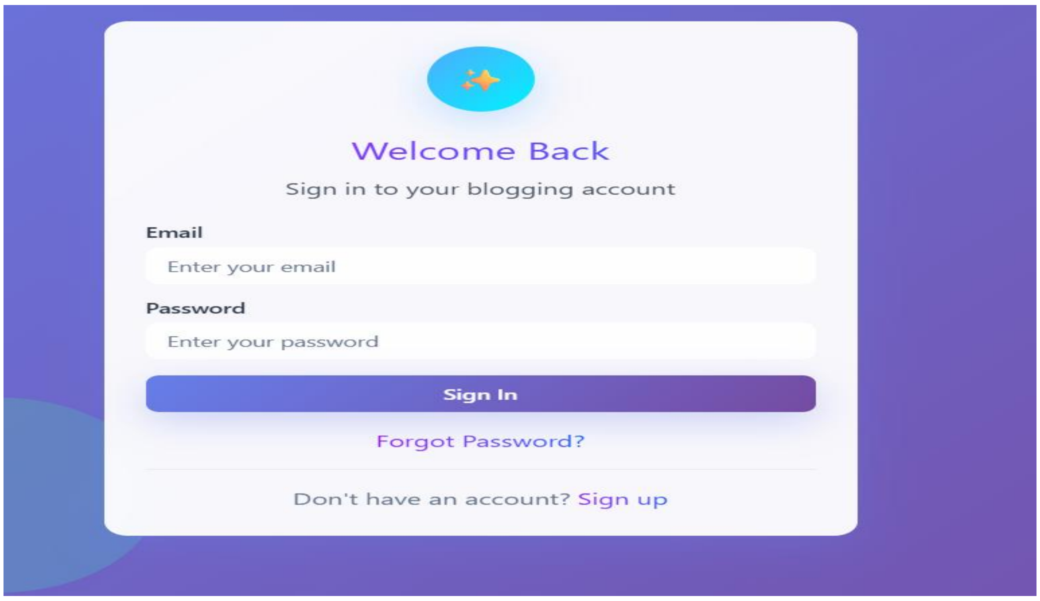
- **Frontend:** Vercel / Netlify
- **Backend:** Render / AWS EC2
- **Database:** MongoDB Atlas (cloud-hosted solution for scalability and reliability)

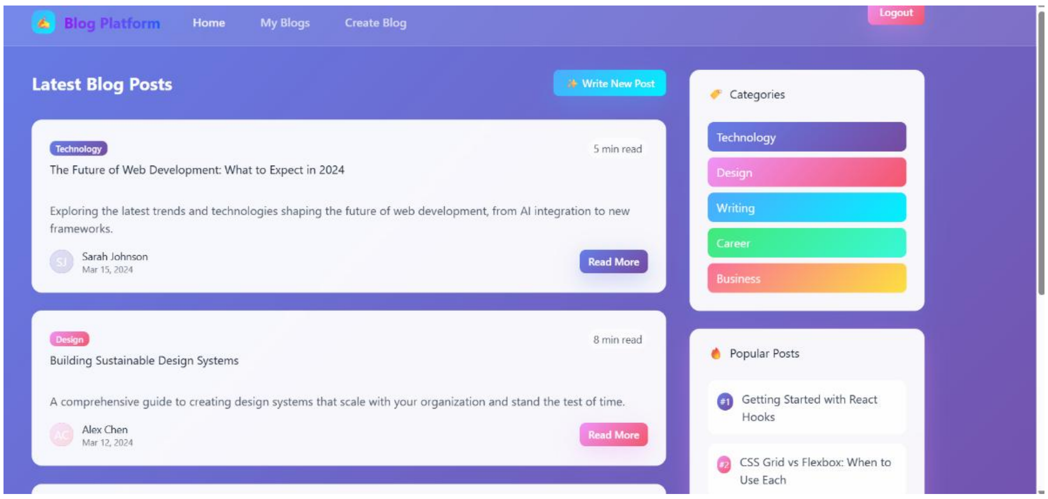
UI Structure & API Schema Design:

UI Structure

The Blogging Platform will have a simple, clean, and user-friendly interface, divided into the following main sections:

1. **Authentication Pages**
   - **Register Page:** Fields for username, email, and password.
   - **Login Page:** Fields for email and password, with JWT-based session handling.
2. **Dashboard Page**
   - Displays the list of the user's blogs.
   - Button to **Create Blog**.
   - Options to edit or delete existing blogs.
3. **Create/Edit Blog Page**
   - Form with **Title** and **Content** (Markdown editor).
   - Buttons for **Save** and **Cancel**.
4. **Blog View Page**
   - Displays the full blog content with Markdown formatting.
   - Section for **Comments** (list and add new).
5. **Profile Page**
   - Displays user details (username, email, join date).
   - Option to update profile or delete account.


Welcome Back
Sign in to your blogging account
Email
Enter your email
Password
Enter your password
Sign In
Forgot Password?
Don't have an account? Sign up


Blog Platform Home My Blogs Create Blog Logout

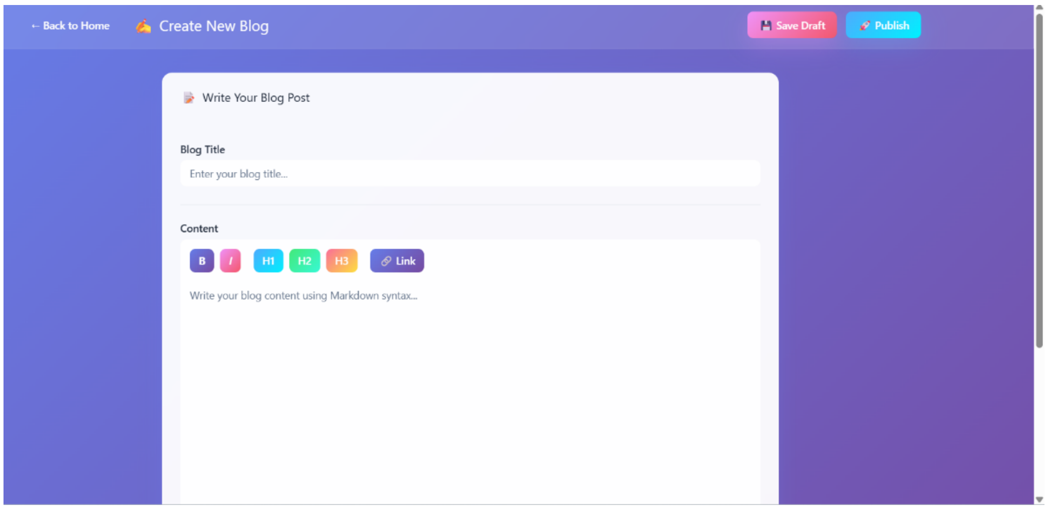
Latest Blog Posts
Write New Post

Technology
5 min read
The Future of Web Development: What to Expect in 2024
Exploring the latest trends and technologies shaping the future of web development, from AI integration to new frameworks.
Sarah Johnson
Mar 15, 2024
Read More

Design
8 min read
Building Sustainable Design Systems
A comprehensive guide to creating design systems that scale with your organization and stand the test of time.
Alex Chen
Mar 12, 2024
Read More

Categories
Technology
Design
Writing
Career
Business

Popular Posts
#1 Getting Started with React Hooks
#2 CSS Grid vs Flexbox: When to Use Each


← Back to Home
Create New Blog
Save Draft
Publish

Write Your Blog Post
Blog Title
Enter your blog title...
Content
B I H1 H2 H3 Link
Write your blog content using Markdown syntax...

API Schema Design:

User Schema

JSON

```
{
  "username": "string",
  "email": "string",
  "password": "string (hashed)",
  "createdAt": "Date"
}
```

Blog Schema

JSON

```
{
  "title": "string",
  "content": "string (Markdown supported)",
  "author": "ObjectId(User)",
  "comments": [
    {
      "user": "ObjectId(User)",
      "text": "string",
      "createdAt": "Date"
  }
  ],
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

Comment Schema

JSON

```
{
  "blogId": "ObjectId(Blog)",
  "user": "ObjectId(User)",
  "text": "string",
  "createdAt": "Date"
}
```

Data Handling Approach-Blogging platform:

The Blogging Platform follows a secure and structured approach for managing data. Since data integrity, scalability, and performance are critical, a **NoSQL database (MongoDB)** with **Mongoose ORM** is used to define and enforce schema rules.

Data Flow

1. **Frontend Input:**
   - User provides data (e.g., blog title, content, comments) via UI forms.
   - Input validation is performed on the client side (e.g., empty fields, incorrect formats).
2. **API Layer:**
   - Data is sent to the backend through REST APIs (POST, GET, PUT, DELETE).
   - Express.js middleware handles request parsing and authentication.
3. **Backend Processing:**
   - Mongoose validates incoming data against defined schemas.
   - Business logic (such as Markdown conversion, slug generation, timestamps) is applied.
4. **Database Storage:**
   - Validated data is stored in **MongoDB collections**:
     - **Users Collection** → User details, hashed passwords, roles.
     - **Blogs Collection** → Blog posts with title, content, author, timestamps.
     - **Comments Collection** → Linked to blogs via blogId reference.
5. **Data Retrieval:**
   - Queries are optimized with indexing on fields like authorId, createdAt.
   - Aggregation pipelines support filtering, sorting, and pagination of blogs/comments.

Security in Data Handling

- **Authentication:** JWT tokens secure API requests.

- **Authorization:** Middleware ensures only blog authors can edit/delete their posts.
- **Password Handling:** Stored with bcrypt.js hashing, never in plain text.
- **Validation:** Both client-side (UI) and server-side (Mongoose schema rules).

Benefits of This Approach

- **Scalable:** Supports increasing number of blogs and users.
- **Secure:** Protects against unauthorized access and data leaks.
- **Efficient:** Fast CRUD operations with MongoDB's flexible schema design.
- **Consistent:** Enforced schema ensures clean and predictable data handling.

Components of a Blogging Platform:

1. Authentication Component

- Handles **user signup, login, logout**.
- Uses **JWT tokens** for secure session management.
- Ensures that only authenticated users can create or edit blogs.

2. User Component

- Stores and manages **user profiles** (name, email, bio, etc.).
- Tracks user roles (Admin, Author, Reader).
- Provides endpoints to update or fetch user details.

3. Blog Component

- Core of the platform – manages **CRUD operations**:
    - Create a blog post.
    - Read (fetch) blog posts.

    - Update blog posts (only by author).
    - Delete blog posts.
- Supports **Markdown editor** for formatting content.
- Stores metadata like title, content, author, timestamps.

4. Comment Component

- Allows readers to **add, edit, delete comments**.
- Comments are linked to specific blog posts.
- Supports nested/threaded comments for discussions.

5. Database Component

- Uses **MongoDB + Mongoose** for storing all data.
- Collections:
    - **Users** → user accounts, hashed passwords.
    - **Blogs** → blog content, author info, timestamps.
    - **Comments** → comment text, user, and blog reference.
- Ensures **data validation** and **relationships**.

6. UI / Frontend Component

- Built with **React.js** for dynamic user interaction.
- Key screens:
    - Home page (list of blogs).
    - Blog detail page.
    - Create/Edit blog form.
    - Login/Signup page.
    - User profile/dashboard.

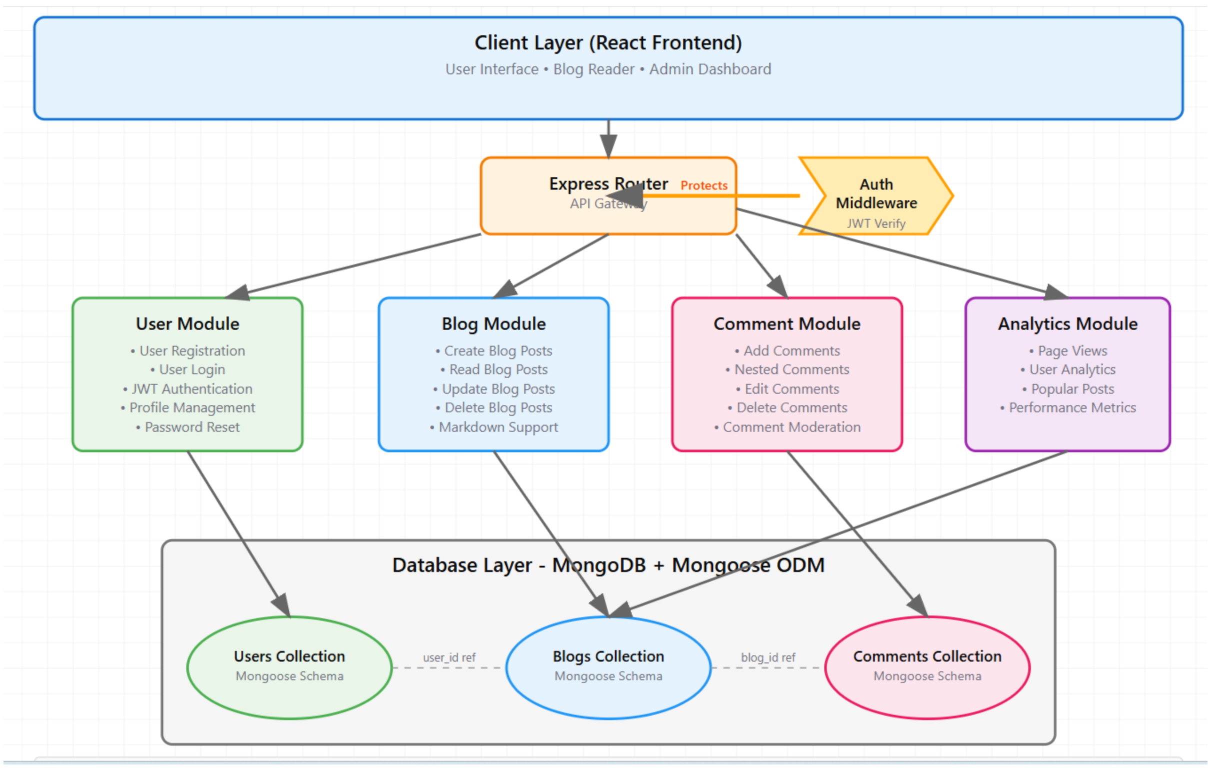
7. API / Backend Component

- RESTful APIs built with **Node.js + Express.js**.
- Endpoints for Authentication, Blogs, Comments, and Users.
- Middleware for validation, error handling, and security.

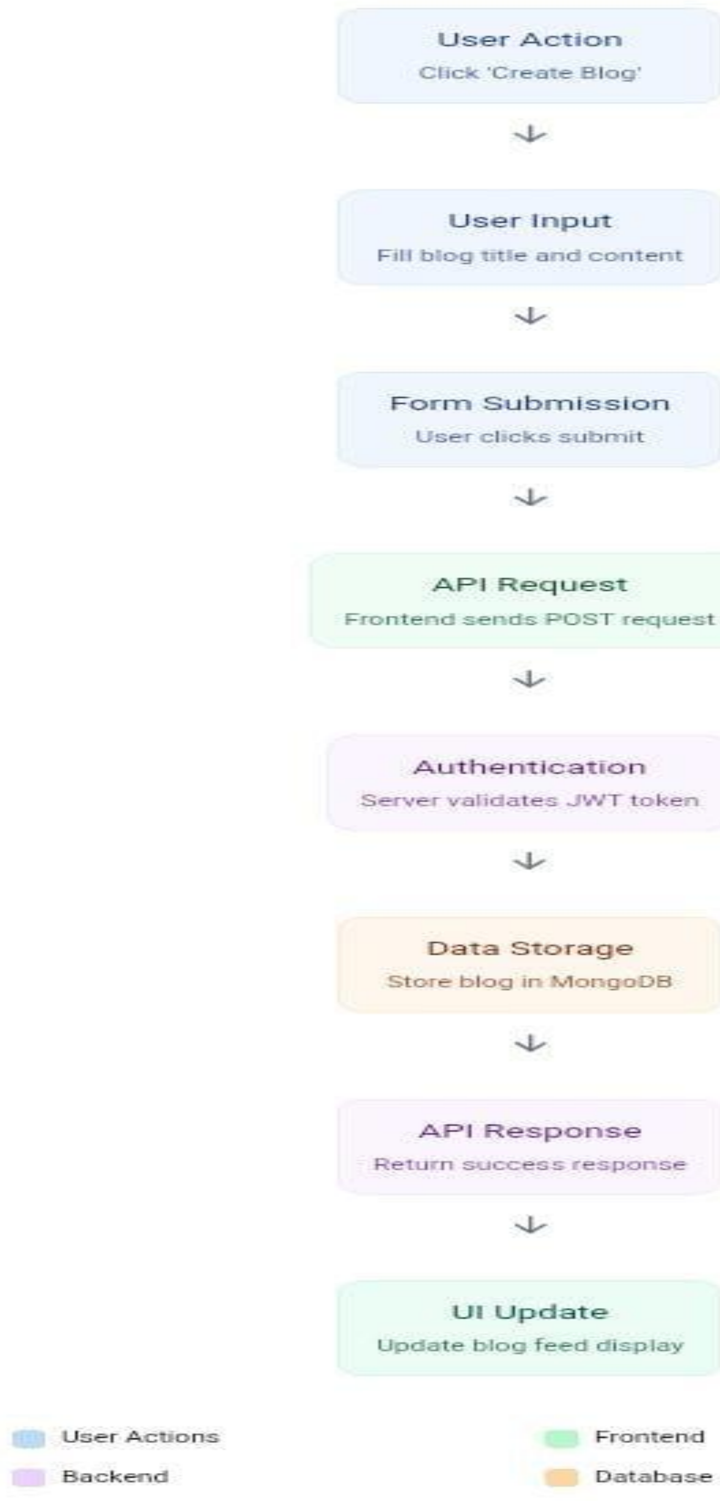
8. Security Component

- **JWT Authentication** for secure access.
- **bcrypt.js** for password hashing.
- Middleware to prevent unauthorized CRUD operations.

Module Diagram:



Basic Flow Diagram:


User Action
Click 'Create Blog'
User Input
Fill blog title and content
Form Submission
User clicks submit
API Request
Frontend sends POST request
Authentication
Server validates JWT token
Data Storage
Store blog in MongoDB
API Response
Return success response
UI Update
Update blog feed display
User Actions
Frontend
Backend
Database