COLLEGE CODE:8203

COLLEGE NAME:A.V.C College of Engineering

DEPARTMENT: Computer Science And Engineering

STUDENT NM-ID:2774A1DBA16809C8C764F7BFA755494C

ROLL NO:8203231041091

DATE:22-09-2025

Completed the project named as Phase 3TECHNOLOGY

PROJECT NAME :Blogging Platform using Node.js, Express, MongoDB, and JWT.

SUBMITTED BY,

NAME: G.Santhosh

MOBILE NO:6369502254

Phase 3 — MVP Implementation :

Overview:

Following the successful completion of Phase 1 (Requirements Gathering) and Phase 2 (Design) Phase 3 marks the most critical stage of development: implementing the MVP (Minimum Viable Product). The goal of this phase is to transform the conceptual design and user requirements into fully working prototype of the blogging platform. The MVP will offer essential functionalities such as user authentication, blog creation, blog viewing, and commenting—all supported by persistent data storage and version control.

The following key tasks are outlined as part of this phase:

- Project Setup
- Core Features Implementation
- Data Storage Integration
- Testing of Core Features
- Version Control via GitHub

Each component is discussed in detail below.

1. Project Setup:

Setting up the foundation of the project ensures that all team members work in a standardized, collaborative, and consistent environment. The following steps are included in the initial setup:

1.1 Folder Structure and File Organization

Creating a scalable and modular folder structure is critical. A typical MERN (MongoDB, Express.js, React, Node.js) structure might include:

- /client → Frontend (React)
- /server → Backend (Node.js/Express)
- /models → MongoDB schemas
- /routes → Express API routes
- /controllers → Business logic
- /middleware → Auth and validation logic
- /config → Database connection and environment variables

1.2 Dependency Installation

Essential tools and packages will be installed on both frontend and backend:

Backend (Node.js):

- express
- mongoose
- bcryptjs (for hashing passwords)
- jsonwebtoken (for JWT)

- cors
- dotenv

Frontend (React):

- axios (for HTTP requests)
- react-router-dom (for page navigation)
- formik/yup (for form handling and validation)
- context API or Redux for state management (if needed)

1.3 Development Environment

Set up includes:

- Using environment variables via .env files
- Running both frontend and backend using tools like concurrently
- Enabling live reloading via nodemon for backend

2. Core Features Implementation:

This step is the heart of Phase 3 and involves coding all the primary features defined in the MVP. This includes both backend APIs and frontend components.

2.1 Authentication (JWT-Based)

Implement secure user registration and login using JSON Web Tokens (JWT). Key steps:

- Register: Hash passwords using bcrypt, save user data in MongoDB
- Login: Validate user credentials, return signed JWT
- Middleware: Create auth middleware to protect routes (e.g., blog creation)

Sample Routes:

- POST /api/auth/register
- POST /api/auth/login

2.2 Blog CRUD Operations

Authenticated users can create, read, update, and delete blog posts. Features:

- Create a blog: Requires title and content; associate blog with user ID
- Read: List all blogs or a specific blog
- Update/Delete: Only the author can edit or delete their own blogs

Sample Routes:

- POST /api/blogs
- GET /api/blogs

- PUT /api/blogs/:id
- DELETE /api/blogs/:id

2.3 Commenting System

Users can comment on blogs. Comments are stored with blog ID references. Sample route:

- POST /api/blogs/:id/comments

2.4 Frontend Pages

React components built to match the wireframes:

- Login and Register Pages
- Dashboard/Home Page (list of blogs)
- Blog View Page
- Blog Editor (create/edit)
- Comment Section

3. Data Storage (Local State / Database):

Data handling is critical for managing content and users securely and reliably. The following architecture will be implemented:

3.1 Database: MongoDB

MongoDB will serve as the primary database for storing:

- Users (name, email, hashed password)
- Blogs (title, content, author ID, timestamp)
- Comments (content, user ID, blog ID)

Database Connection:

- MongoDB Atlas or local MongoDB setup
- Connection managed via Mongoose (ODM)

3.2 Local State (Frontend)

Local component state or Context API will be used to manage:

- User authentication state (token in localStorage)
- Blog and comment data for rendering
- UI state (form inputs, loading indicators)

Optional: Redux may be introduced later for more complex state handling.

4. Testing Core Features:

Testing ensures that the core functionality works as expected and is free from major bugs.

4.1 Backend Testing

Tools: Postman or Insomnia

- Test each API route (register, login, blogs CRUD, comments)
- Test JWT token validation and access restrictions
- Handle edge cases: invalid data, missing fields, unauthorized access

4.2 Frontend Testing

- Functional testing for all user flows:
  - Register → Login → Create Blog → Edit → Delete
  - View Blog → Add Comment
- Ensure validation errors are shown when required fields are empty
- Check that protected routes are inaccessible when not logged in

Optional: Jest or React Testing Library can be used for automated testing.

5. Version Control (GitHub):

Efficient version control is essential for collaborative development and code safety.

5.1 Repository Setup

- Create a centralized GitHub repository
- Add proper README with setup instructions
- Use .gitignore to exclude node_modules, .env files, etc.

5.2 Branching Strategy

Recommended strategy:

- main: stable, deployable version
- dev: ongoing development
- feature/[feature-name]: for individual features

5.3 Commit Conventions

Use semantic commit messages:

- feat: new feature
- fix: bug fix
- refactor: code changes without altering functionality
- docs: documentation updates

5.4 Collaboration

- Pull Requests (PRs) for merging code

- Code reviews before merging into main or dev branches

6. Conclusion:

Phase 3 is where the project transitions from design into execution. Through structured setup, implementation of core features, integration with MongoDB, and thorough testing, this phase will produce a working MVP of the blogging platform. By the end of Week 8, the project should include:

- Secure login and registration system
- Full blog CRUD functionality
- Comments on blog posts
- GitHub repository with clean commit history
- Tested and validated REST APIs

This phase ensures that the platform is functional, testable, and ready for deployment and feedback in the next stages.