COLLEGE CODE : 9623

COLLEGE NAME : Amrita College Of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID : BDEE936B6F75EC18C6640B3F81D816C5

ROLL NO : 962323104077

DATE : 25-09-2025

Completed the project named as Phase_03_ TECHNOLOGY

PROJECT NAME : Blog Site with Comment Section

SUBMITTED BY,

NAME : Prenesh Kumar R

MOBILE NO : 96293 10500

Project Setup

Blog Site with Comment Section- MVP Implementation

The project setup phase is the initial and most crucial stage of development where all the essential tools, technologies, and environment are prepared to enable smooth implementation. In this project, the setup was carried out by configuring **Visual Studio Code** as the development IDE, **GitHub** for version control, and **XAMPP** as the local server environment to run PHP scripts and manage the MySQL database. During this phase, the database schema was designed with three key tables: **Users**, **BlogPosts**, and **Comments**, establishing the relationships required for the MVP. The frontend structure was organized using HTML, CSS, and JavaScript, while backend scripts were connected to the database for handling authentication, blog management, and comment processing.

Key Activities in this Phase

- Installation and configuration of IDE, server, and database.
- Designing and creating the initial database schema.
- Setting up the folder structure for frontend and backend.

Establishing database connectivity with backend scripts.

- Initial testing on localhost to validate environment readiness.

This structured setup ensured a reliable and consistent foundation for further development and allowed smooth transition into the implementation phase.

Core Features Implementation

1. User Authentication

- Implemented registration, login, and logout functionality.
- Passwords stored securely in the database using hashing.
- Only logged-in users can create posts or add comments.

2. Blog Management

- Authors can create blog posts with a title, content, and timestamp.
- Posts are stored in the database and displayed on the homepage.
- Each blog post includes author name and date of creation.

3. Comment Section

- Logged-in users can add comments under blog posts.
- Comments are stored in the database with references to user and post.
- Comments are displayed in chronological order for better readability.

4. Frontend Interface

- Developed using HTML, CSS, and JavaScript.
- Includes pages for login, registration, homepage, and blog detail view.
- Provides simple navigation and a clean user experience.

5. Backend & Database Integration

- Backend scripts (PHP/Node.js) handle logic and database communication.
- MySQL database used to manage users, posts, and comments.
- Proper relationships maintained between tables for data integrity.

Data Storage

1. Local State Management

- Temporary data such as form inputs (blog drafts, comment text) is stored in the local state before submission.
- Client-side state ensures smooth interaction (e.g., typing a comment without reloading the page).
- JavaScript or React state variables are used to handle UI updates instantly.

2. Database Storage

- Persistent data such as **users, blog posts, and comments** is stored in a relational database (MySQL).
- Tables are linked using relationships (e.g., one-to-many: one blog post can have many comments).
- Data is retrieved via API calls and displayed dynamically on the frontend.

3. Synchronization Between State and Database

- On submitting a post or comment, local state is cleared after saving to the database.
- Ensures that frontend and backend data remain consistent.
- Validation checks prevent duplicate or invalid entries before storage.

Testing Core Features

1. View Blog Posts:

- **Objective:** Ensure all published blogs are displayed correctly.
- **Test Steps:**
    1. Open the blog site.
    2. Check if all blog posts are listed.
    3. Verify that each blog shows title, content, and author.
- **Expected Result:** All blog posts are visible and properly formatted.

2. Post Comments:

- **Objective:** Ensure users can add comments under a blog.
- **Test Steps:**
    1. Select a blog post.
    2. Enter a comment in the comment box.
    3. Submit the comment.
- **Expected Result:** Comment appears under the blog and is stored in the database.

3. Basic Navigation:

- **Objective:** Ensure users can move between pages or sections.
- **Test Steps:**
    1. Use the menu or links to navigate to different blogs or sections.
- **Expected Result:** Navigation works smoothly without errors.

Version Control

Effective version control practices are crucial for managing the project source code and supporting collaborative development.

Git:

Git will be used to **track changes in the codebase**, allowing developers to manage different versions of the project efficiently. This facilitates:

- Code review
- Branching for feature development
- Easy rollback if issues arise

GitHub Repository:

Hosting the project on GitHub provides a **centralized platform for collaboration**. It enables:

- Issue tracking
- Pull requests
- Continuous integration workflows
  These features improve code quality and enhance team coordination.

Best Practices:

- Commit code frequently with **clear and descriptive messages**
- Use **branching strategies** (e.g., feature branches, main/master branches) to organize work and reduce conflicts
- Maintain a structured workflow to ensure smooth development, easier debugging, and better documentation of the project's evolution