1. Goals for this Phase

The main objective of Phase 3 is to implement the Minimum Viable Product (MVP) of the blog site with a fully functional comment section. By this stage, the requirements and design are already finalized, and the focus now shifts towards development and implementation.

The MVP should allow users to perform essential tasks such as creating, viewing, and managing blog posts along with posting comments.

Another important goal is to ensure that the product is secure, stable, and scalable, even in its minimal form. Testing and version control will also play a critical role.

2. Project Setup

Initialize the project repository in GitHub and configure version control with main and development branches.

Set up the development environment using frameworks and tools. For frontend, React is chosen; for backend, Node.js with Express; for database, PostgreSQL.

Install dependencies such as database drivers, authentication libraries, and UI libraries.

Organize the folder structure into frontend, backend, and infrastructure directories for easier maintainability.

Add environment configuration files (.env) to store sensitive data such as database connection strings and API keys.

3. Core Features Implementation

Blog Posts: Enable users or administrators to create, edit, and delete blog posts. Blog content may be stored in Markdown or rich text format, with support for cover images, tags, and excerpts.

Comment Section: Allow users to comment under blog posts. Each comment should be linked to a specific user and post. Features include posting, replying, editing, and deleting comments.

Authentication: Implement user login, registration, and logout functionality. Use JWT for token-based authentication.

Admin Features: Provide moderation capabilities such as hiding or deleting inappropriate comments. Admins should also manage flagged comments.

Blog Listing and Detail Pages: Display all blog posts on the homepage with links to detailed pages where users can read the post and view comments.

4. Data Storage

Use PostgreSQL for relational data storage. Separate tables should be created for Users, Posts, Comments, and Tags.

Design relationships such as one user can have many comments, and one post can have many comments.

Store passwords securely by hashing them using bcrypt or Argon2.

Implement indexes on frequently queried fields like post IDs and timestamps to improve performance.

As an alternative, MongoDB may be used for flexible schema handling, especially for large volumes of comments.

5. Testing Core Features

Unit Testing: Test individual functions such as comment creation, slug generation, and validation functions.

Integration Testing: Test the interaction between different modules such as ensuring that comments posted are properly linked to their parent blog posts in the database.

End-to-End Testing: Simulate a real user flow where a user logs in, reads a blog post, posts a comment, and sees it appear in real-time.

User Interface Testing: Validate forms for creating posts and submitting comments. Ensure validation rules are applied correctly.

Accessibility Testing: Check if the website is usable by screen readers and follows accessibility guidelines.

6. Version Control (GitHub)

The GitHub repository should be initialized at the beginning of this phase.

Branching Strategy: Use 'main' for production-ready code, 'dev' for ongoing development, and feature branches for specific tasks like 'feature-comments' or 'feature-auth'.

Frequent commits should be made with descriptive messages to track progress.

Pull requests should be used for merging code into the main branch after peer review.

GitHub Actions can be set up for continuous integration, running tests automatically on each push.

7. Deployment & Infrastructure

The frontend can be deployed on Netlify or Vercel for fast static hosting.

The backend server can be hosted on Render, Heroku, or AWS.

Database: Use a managed PostgreSQL instance such as Supabase or Amazon RDS for secure and scalable storage.

File uploads such as images should be stored in AWS S3 or similar cloud storage with CDN integration for fast access.

Monitoring tools like Sentry can be used for error tracking, and UptimeRobot can monitor system uptime.

8. Acceptance Criteria

Users should be able to register, log in, and log out securely.

Users should be able to create and view blog posts on the homepage and detail pages.

Authenticated users should be able to post comments on blogs.

Comments should appear under the correct blog post in real time.

Admins should have the ability to delete or hide inappropriate comments.

The application should pass unit, integration, and end-to-end tests without critical bugs.

The codebase should be version-controlled and deployed to a live environment.