

# PROJECT TITLE – BLOGSPHERE

## Execution Steps

### Phase 1: Environment setup

Set up Node.js, npm, MongoDB, Git, Visual Studio Code, and Postman on all development machines. Create the project root folder blogsphere/ with backend/ and frontend/ subfolders and initialize Git. files. Verify that MongoDB server runs locally or that a cloud instance (e.g., Atlas) is reachable.

### Phase 2: Backend development

Initialize the backend Node.js project, install Express, Mongoose, bcrypt, JWT, dotenv, and related libraries. Design and implement Mongoose schemas for Users, Blogs, and Categories Models. Develop authentication middleware using bcrypt for hashing and JWT for token-based login and route protection. Implement REST endpoints for user, blog, and category CRUD, and smoke-test them with basic requests.

### Phase 3: Frontend development

Create the React app, add Bootstrap and Axios, and configure routing for main pages (Home, Login, My Blogs). Build reusable components for login/register, blog list, blog form, category menu. Integrate Axios with the backend APIs, attaching JWT tokens from local storage in Authorization headers. Apply responsive Bootstrap styling and basic client-side validation for form inputs.

### Phase 4: Integration and end-to-end testing

Run backend and frontend together and verify full flows: register, login, create blog, view, edit, delete, and filter. Check that unauthorized actions are correctly blocked. Fix mismatches in API contracts, error messages, and field names between frontend and backend. Validate that category selections and “My Blogs” views correctly reflect data in MongoDB.

### Phase 5: Performance and security checks

Confirm that passwords are stored only as bcrypt hashes and secrets come from environment variables. Test JWT expiration, invalid token handling, and access control on all protected routes. Use repeated or concurrent API calls to measure response times and observe behaviour under moderate load.

### Phase 6: Documentation and deployment preparation

Document all REST endpoints with sample requests/responses using your API testing tool. Write concise README files for backend and frontend describing setup, configuration, and run/build commands. Prepare production configuration files, including .env templates with database URIs and JWT secrets. Generate a production React build and verify that it serves correctly in a local test environment.

### Phase 7: Final review and deployment

Perform a final code review focusing on style, duplication, security, and maintainability. Execute regression tests on the near-production setup to ensure all core features still work. Deploy the backend API to a production host and the React build to a static hosting platform. Configure basic logging/monitoring and conduct post-deployment smoke tests with real users or sample data.