Charter Document

Blog Application

Purpose

To build a secure and interactive blog application that allows users to create, read, update, and delete blog posts, with features such as email/password-based **user authentication,** post categorization, and comment functionality. Users can mark blog posts as **favorites** to easily access preferred content. The platform supports image uploads. It also includes **role-based access control** for managing content and users effectively, along with a clean and responsive user interface.

Objective

To provide a secure, user-friendly, and feature-rich blogging platform, offering:

- Authenticated CRUD operations on blog posts.
- Categorized and searchable content.
- Favorite posts management for quick access
- Clean and responsive user interface
- Commenting and post interaction support
- Role-based content management for authors

Opportunity

Opportunity-Based

There is a growing demand for personalized content-sharing platforms as users increasingly seek control over their own content and digital identity.

Reference: Medium & Statista Reports on User-Generated Content Trends

These reports highlight the increasing shift toward independent blogging platforms as alternatives to mainstream social media. Users are looking for spaces where they can express ideas freely, manage their content securely, and engage meaningfully with their audience — creating an ideal opportunity for a modern, secure blog application.

Business Requirement:

- **User Authentication**
  Users must register and log in using secure email/password credentials to create and manage blog content, ensuring authenticated access.

- **Blog Post Management (CRUD)**
  Authenticated users can **create, read, update, and delete** blog posts. Each post supports rich text or markdown formatting, and optional image uploads.

- **Categorization**
  Posts can be organized under categories and tags, enabling better content discovery and navigation for readers.

- **Favorites Feature**
  Users can mark blog posts as **favorites** to easily revisit content they like, which is saved in a personalized list on their profile.

- **Commenting System**
  Authenticated users can comment on blog posts, allowing community engagement and discussion under each article.

- **Role-Based Access Control**
  Different roles such as **Author** and **Reader** determine the level of access and control a user has over blog content and moderation.

- **Search Functionality**
  Users can search for blog posts by title, category or author.

- **Responsive UI/UX**
  The platform must have a clean and responsive interface, optimized for desktop, to ensure seamless reading and writing experiences

Technical Requirement:

- **User Authentication**
  - UI for secure email/password login and registration
  - API routes for user sign-up, login, logout, and session management
  - Passwords securely hashed and stored in the backend database
- **Blog Post CRUD Operations**
  - API endpoints for creating, reading, updating, and deleting blog posts
  - Support for markdown formatting and optional image/file uploads
- **Favorites Management**
  - UI to allow users to mark and unmark blog posts as favorites
  - Backend endpoints to add, retrieve, and manage favorite posts linked to user profiles
- **Commenting System**
  - API routes to create, retrieve comments on blog posts
- **Search**
  - Backend logic for searching posts by title, tags, category.
  - Client-side filters and sorting mechanisms for user-friendly navigation
- **Data Storage**
  - Use of a secure and scalable database (e.g., MongoDB) to store user profiles, posts, comments, and favorites

Technological Requirement:

- **Frontend**
  - React.js – For building a responsive, component-based UI
  - Tailwind CSS – For fast and customizable styling

- **Backend**
  - Node.js + Express.js – RESTful APIs for handling all server-side logic
  - JWT– integration for user management and secure auth flows

- **Database**
  - MongoDB– Primary NoSQL database for storing blog data, users, comments, etc.

- **Dev & Testing Tools**
  - Visual Studio Code – Code editor for development
  - GIT & GitHub – Version control and collaboration
  - Postman – For testing and debugging API endpoints
  - MongoDB– GUI for managing and visualizing MongoDB data

Stakeholder:

Stakeholder	Name	Count
Developers	Pooja Jagtap Divya Gatkal	2
DB Designers	Pooja Jagtap	1
Testers	Sanika Kundekar Shravani Sakore Sayali Deshmukh	3
Cloud Service Provider	AWS Cloud	1
Hosting Provider	Render	1
Security Team	Divya Gatkal	2
Investor	TBD	0

Resources Needed:

- **Documentation**
  - React Js
  - Tailwind Css
  - Express Js
  - MongoDB

- **Cloud Account**
  - AWS

- **Hosting**
  - Render
  - Vercel

- **Human Resource**

Role	**Count**	**Name**
UI/UX Designer	1	Divya Gatkal
Frontend Developer	2	UI Development 1. Divya Gatkal API Integration 1. Pooja Jagtap
Backend Developer	1	Pooja Jagtap
DB Designer	1	Pooja Jagtap
Project Management	2	Divya Gatkal Pooja Jagtap
Documentation	2	*Creator* 1. Pooja Jagtap *Reviewer* 1. Divya Gatkal
Tester	3	SanikaKundekar Shravani Sakore Sayali Deshmukh

PESTEL Analysis:

- **Political:**
  - No direct political constraints expected.
  - Content moderation policies may require compliance with local regulations.
- **Economic:**
  - Utilizes a low-cost tech stack (MERN) enabling affordable development and deployment.
  - Suitable for startups and individual developers with budget constraints.
- **Social:**
  - Growing user demand for personalized content management and easy sharing features.
  - Social sharing and favorites features encourage user engagement and community building.
- **Technological:**
  - Modern web technologies (React, Node.js) ensure responsive and scalable application.
  - Secure authentication with JWT Auth supports user trust and data privacy.
- **Environmental:**
  - Cloud-based hosting minimizes local infrastructure and energy consumption.
  - Serverless and managed services reduce resource wastage and improve efficiency.

Risk Analysis:

Risk	**Description**	**Mitigation**
Software Compatibility	Frequent updates to frameworks/libraries (React, Express, JWT etc.) may cause version conflicts or break existing features.	Perform regular dependency updates, use version control (e.g., package-lock.json), and conduct periodic compatibility tests.
Performance Bottlenecks	Increased number of users or blog entries could slow down app performance, especially if data isn't efficiently handled.	Optimize database queries, implement pagination and caching where necessary, and monitor performance with analytics tools.
Over Budget	Costs for hosting (e.g., Render), cloud storage (e.g., AWS S3), and email/notification services may exceed budget as the app scales.	Plan with a clear budget, utilize free-tier services where possible, and switch to cost-effective providers (e.g., SES over SendGrid).
Security Risks	Unauthorized access or vulnerabilities could compromise user data or app integrity.	Use authentication, validate user input, sanitize data, and conduct regular security audits. Use HTTPS and secure headers.
Data Loss	Blog posts or user data might be lost due to accidental deletion, app crashes, or server failure.	Enable automatic backups for database and storage (AWS S3), implement undo/delete confirmation, and consider a soft-delete mechanism.

Timeline / Milestone:

Phase	Milestone	Tasks	Timeline
1	Requirement Analysis & Planning	- Gather project requirements - Define & construct functional flow of the application - Identify third-party services and dependencies - Document all gathered requirements	Week 1
2	Database & Model Design	- - Design database schema for blogs, users, and favorites - Define relationships (users ↔ blogs, blogs ↔ favorites)	Week 2
3	Backend Development	- - Set up Node.js + Express.js project structure - Develop RESTful API endpoints for blog CRUD and favorites	Week 3-4
4	Frontend Development	- Create UI wireframes for blog listing, single post view, add/edit/delete post - Build components using React and Tailwind CSS - Implement favorite button and filters	Week 5-6
5	API Integration	- Connect frontend with backend for user authentication. - Implement error handling & validation for form inputs and API responses	Week 7
6	Testing & Debugging	- - Conduct unit and integration testing - Test favorite functionality and blog flows - Fix bugs and optimize performance	Week 7
7	Deployment & Final Review	- Host the project on Render - Set up environment variables and storage (e.g., AWS S3 for blog images) - Final testing and deployment	Week 8