Software Requirement Specification (SRS)

for MERN Blog Website

1. Introduction

1.1 Purpose

The purpose of this document is to outline the software requirements for the development of a MERN (MongoDB, Express.js, React.js, Node.js) based blog website. The blog website aims to provide a platform for users to create, edit, and manage blog posts.

1.2 Scope

The MERN blog website will include features such as user authentication, blog post creation and editing, commenting, and categorization of blog posts. The website will also have a responsive design to ensure a seamless user experience across various devices.

2. System Overview

2.1 System Description

The MERN blog website will be a web application that allows users to register, create, and manage blog posts. The system will use MongoDB as the database, Express.js for server-side development, React.js for the front-end, and Node.js for server-side scripting.

2.2 System Features

1. User Registration and Authentication

- Users can register with a valid email address and password.
- Secure authentication mechanism to protect user accounts.

2. Blog Post Management

- Registered users can create, edit, and delete their blog posts.
- Users can categorize their blog posts and add tags for better organization.

3. Commenting System

- Users can leave comments on blog posts.
- Moderation features for managing and deleting comments.

4. Responsive Design

- The website should be accessible and user-friendly on various devices (desktop, tablet, and mobile).

5. Search Functionality

- Users can search for blog posts based on keywords, tags, or categories.

6. Admin Panel

- An admin panel to manage users, blog posts, and comments.
- Admins can edit or remove any user, post, or comment.

3. Functional Requirements

3.1 User Registration and Authentication

1. The system shall allow users to register with a unique email address.
2. The system shall encrypt and securely store user passwords.
3. Registered users shall be able to log in with their credentials.

4. Unauthorized users shall not have access to create, edit, or delete blog posts.

3.2 Blog Post Management

1. Users shall be able to create a new blog post with a title, content, and optional tags.
2. Users shall be able to edit or delete their existing blog posts.
3. The system shall support the categorization of blog posts.
4. Users shall be able to add tags to their blog posts for better organization.

3.3 Commenting System

1. Users shall be able to leave comments on blog posts.
2. The system shall display comments below each blog post.
3. Admins shall have the ability to moderate and delete comments.

3.4 Responsive Design

1. The website shall be responsive and compatible with various devices and screen sizes.

3.5 Search Functionality

1. Users shall be able to search for blog posts based on keywords, tags, or categories.

3.6 Admin Panel

1. The admin panel shall be accessible only to authorized administrators.
2. Admins shall be able to manage users, including editing or removing user accounts.
3. Admins shall have the ability to manage blog posts and comments.

4. Non-Functional Requirements

4.1 Performance

1. The website shall load within 3 seconds on average internet speed.
2. The system shall support up to 1000 simultaneous users without significant performance degradation.

4.2 Security

1. User passwords shall be securely stored using encryption.
2. The system shall implement protection against common web vulnerabilities (e.g., Cross-Site Scripting, Cross-Site Request Forgery).

4.3 User Interface

1. The user interface shall be intuitive and user-friendly.
2. The design shall follow best practices for web accessibility.

4.4 Scalability

1. The system architecture shall be designed to accommodate future scalability requirements.

5. Constraints

1. The project budget is limited to a specified amount.
2. The development timeline is fixed and must be adhered to.

6. Assumptions and Dependencies

1. Users have a modern web browser with JavaScript enabled.
2. Internet access is required to use the application and store/retrieve data from the server.

7. Glossary

- MERN: MongoDB, Express.js, React.js, Node.js

- CRUD: Create, Read, Update, Delete

This Software Requirement Specification is intended to provide a detailed description of the MERN blog website's functionality and features. It serves as a foundation for the development team to understand and implement the requirements effectively.

MERN Blog Website Modules

1. Frontend (React)

- Home Page: Display a list of blog posts.
- Post Detail Page: Display the full content of a blog post.
- Create/Edit Post Page: Form to create or edit a blog post.
- Navigation Bar: Navigate between different sections of the website.
- Authentication: Allow users to sign up, log in, and log out.

2. Backend (Node.js and Express)

- Express Server: Set up a server to handle HTTP requests.
- Routes: Define routes for handling CRUD operations on blog posts.
- Database Models: Define MongoDB schemas for blog posts and users.
- Middleware: Implement middleware for tasks like authentication.
- Authentication: Implement user authentication using JWT (JSON Web Tokens).
- File Upload: If you want to allow users to upload images for their blog posts.

3. Database (MongoDB)

- Blog Post Collection: Store information about each blog post (title, content, author, date, etc.).
- User Collection: Store user information (username, email, hashed password, etc.).

4. Authentication (JWT)

-User Registration: Allow users to register with a unique username and email.
- User Login: Enable users to log in with their credentials.
- Token Generation: Generate and validate JWT tokens for authenticated users.

- Password Hashing: Hash and salt user passwords for secure storage.

5. State Management (Redux)

- Actions: Define actions for fetching, creating, updating, and deleting blog posts.
- Reducers: Manage the state changes based on different actions.
- Store: Create a Redux store to hold the application state.

6. API Integration

- Axios or Fetch: Use Axios or Fetch to make HTTP requests from the frontend to the backend.

7. User Interface (UI) Design

- CSS Framework: Use a CSS framework (e.g., Bootstrap, Material-UI) for a responsive and visually appealing design.
- Responsive Design: Ensure the website looks good on various screen sizes.

8. Testing

- Unit Testing: Write unit tests for backend routes, functions, and frontend components.
- Integration Testing: Test the interactions between different parts of the application.

9. Deployment

- Heroku, AWS, or Similar: Deploy the backend and frontend to a hosting platform.

10. Additional Features

- Comments: Allow users to comment on blog posts.

- Tags and Categories: Categorize and tag blog posts for better organization.
- Search Functionality: Implement a search feature for finding specific blog posts.
- Pagination: Display a limited number of blog posts per page with pagination.

The hardware and software requirements for a MERN (MongoDB, Express.js, React, Node.js) blog website development environment can vary based on the scale and complexity of your project. Below are general recommendations:

Hardware Requirements:

Development Machine:
- Processor: Multi-core processor (e.g., Intel Core i5 or AMD Ryzen 5).
- RAM: 8 GB or higher.
- Storage: SSD for faster read/write speeds.
- Network: High-speed internet connection for efficient package installations and updates.

Production Server (for deployment):
- Processor: Multi-core processor with sufficient power for anticipated traffic.
- RAM:16 GB or higher.
- Storage: SSD with ample storage capacity for the database and static assets.

- Network: Reliable and high-speed internet connection.
- Security: SSL certificate for secure data transmission (HTTPS).

Software Requirements:

Development Environment:

1. Operating System:
   - macOS, Linux, or Windows (although Linux is often preferred for server deployments).

2. Text Editor/IDE:
   - Visual Studio Code, Atom, Sublime Text, or any preferred code editor.
   - IDEs like WebStorm or Visual Studio if preferred.

3. Node.js and npm:
   - Install the latest stable version of Node.js and npm (Node Package Manager).

4. Version Control:
   - Git for version control.
   - GitHub, GitLab, or Bitbucket for remote repositories.

Backend (Node.js and Express):

1. Node.js Packages:
   - Express.js: Web application framework for Node.js.
   - mongoose: MongoDB object modeling for Node.js.
   - jsonwebtoken: For JWT-based authentication.

Frontend (React):

1. Node.js and npm:

- Required for installing and managing frontend dependencies.

2. Create React App:
   - Tool to bootstrap a React application quickly.

Database (MongoDB):

1. MongoDB:
   - Install and set up MongoDB locally or use a cloud-based MongoDB service like MongoDB Atlas.

State Management (Redux):

1. Redux:
   - Install Redux for state management.

Other Tools:

1. Postman:
   - For API testing.

2. Robo 3T or MongoDB Compass:
   - GUI tools for interacting with MongoDB.

3. Browser Developer Tools:
   - Chrome DevTools, Firefox Developer Edition, etc.

Introduction

In the era of rapid technological advancements, the exchange of knowledge and insights in the field of technology has never been more crucial. As the digital realm expands, individuals seek platforms to share their experiences, discoveries, and expertise with a global audience. It is in this context that we introduce our two-month mini project – the creation of a Tech Blog Website using the MERN (MongoDB, Express.js, React, and Node.js) stack.

Literature Review

This literature review highlights key trends and practices in web development and technology relevant to creating a Tech Blog Website with MERN technologies.

1. Web Application Development Trends: Web apps have shifted from traditional server-rendered to dynamic single-page applications (SPAs). The MERN stack (MongoDB, Express.js, React, and Node.js) is popular for its flexibility and efficiency.

3. MongoDB and NoSQL Databases: MongoDB, part of MERN, excels in flexible schema design and scalability, handling unstructured data efficiently.

4. Express.js and Node.js for Backend Development: Express.js and Node.js have revolutionized backend development with their non-blocking, event-driven nature. Node.js is highly performant, while Express.js simplifies RESTful API creation.

5. React for Front-End Development: React, by Facebook, dominates front-end development, emphasizing component-based architecture and virtual DOM for interactive, maintainable UIs.

6. User Authentication and Authorization: Security is paramount in web apps. Robust user authentication and authorization, including token-based and OAuth strategies, safeguard user data.

7. Community-Driven Content Sharing: Platforms like Stack Overflow and GitHub showcase collaborative knowledge sharing in the tech industry, fostering community engagement and insights exchange.

This condensed literature review provides insights into the relevant research and industry developments related to the MERN stack and web application development.

Project/Research Objective

The primary objective of this project is to design, develop, and deploy a fully functional Tech Blog Website using the MERN (MongoDB, Express.js, React, and Node.js) stack within a two-month timeframe. The specific goals and research objectives of this project are as follows:

1. Platform Creation: Develop a user-friendly and visually appealing web platform that serves as a dedicated space for technology enthusiasts to share their knowledge, insights, and experiences in the tech sector.

2. Content Management: Enable registered users to create, edit, and publish tech-related blogs effortlessly. Implement features for users to manage their blog posts effectively, including editing and deletion.

3. Dynamic Content Presentation: Design the website to display recently published blogs prominently on the homepage, ensuring that visitors have easy access to the most current and relevant content.

4. User Authentication and Authorization: Implement robust user authentication mechanisms to safeguard user data and ensure secure access to the platform. Define user roles and authorization levels to manage user privileges effectively.

5. Comment System: Incorporate a comment system that allows users to engage in discussions and provide feedback on blog posts, fostering interaction and community engagement.

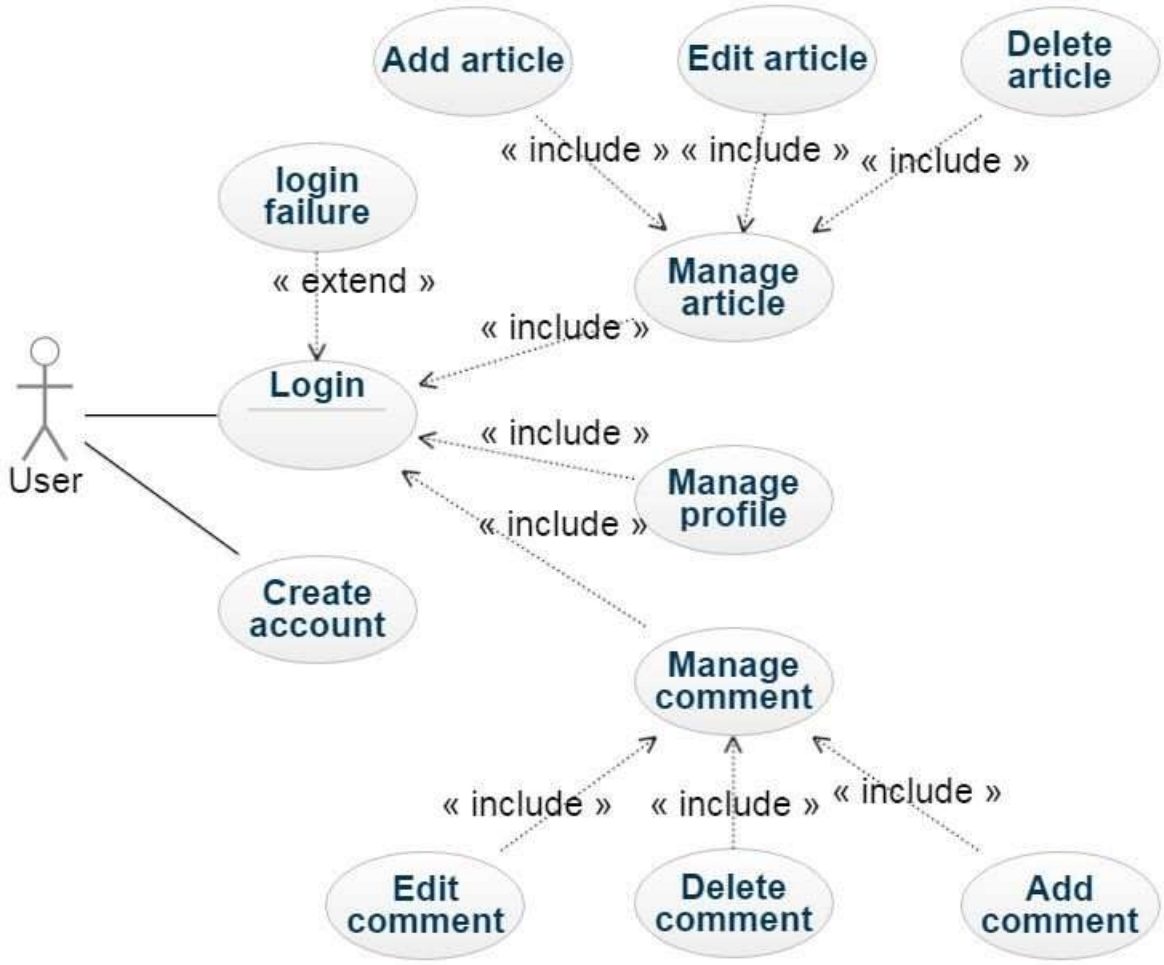
6. Efficient Backend Operations: Develop the back-end of the application using Node.js and Express.js, ensuring efficient handling of API requests, data storage, and management of user interactions.

7. Database Design and Optimization: Design a MongoDB database schema that efficiently stores blog posts and user data. Optimize database performance to handle large volumes of content and users.

8. **Responsive Front-End Design:** Utilize React to create a responsive and visually appealing front-end interface, ensuring an optimal user experience across various devices and screen sizes.

9. Challenges and Problem Solving: Document and address technical challenges encountered during development, including asynchronous operations, database performance optimization, and UI responsiveness.

Use case Diagram:


Add article
Edit article
Delete article
« include »
« include »
« include »
login failure
« extend »
Manage article
« include »
Login
User
« include »
Manage profile
« include »
Create account
Manage comment
« include »
« include »
« include »
Edit comment
Delete comment
Add comment

Project/Research Outcome

The two-month project to develop a MERN stack-based Tech Blog Website yielded the following key outcomes:

1. **Website Deployment:** Successful deployment of a fully functional Tech Blog Website, meeting the primary project objective.
2. **User-Friendly Interface:** Creation of an intuitive and responsive user interface for enhanced cross-device user experiences.
3. **Content Management:** Implementation of efficient tools enabling registered users to create, edit, and publish tech-related blog posts.
4. **Dynamic Content Presentation:** Prominent display of recent blog posts on the homepage, ensuring easy access to the latest and most relevant content.
5. **User Authentication and Authorization:** Establishment of robust user authentication and authorization mechanisms to secure user data and control access.
6. **Comment System:** Integration of a comment system fostering user engagement and discussions on blog posts.
7. **Efficient Backend Operations:** Effective utilization of Node.js and Express.js to manage API requests, ensuring efficient backend processes.
8. **Database Optimization:** Implementation of MongoDB database optimization strategies for improved query performance and scalability.

These outcomes represent the core achievements of the project, showcasing the successful development of a feature-rich and user-friendly web application.

Proposed Time Duration

The project's proposed time duration was two months, reflecting the constraints of a mini-project. This timeframe was chosen to challenge the team to efficiently plan, develop, and deploy a feature-rich web application while adhering to best practices and maintaining high standards of quality.

The two-month duration allowed for a comprehensive exploration of the MERN stack's capabilities in web development, including database design, front-end and back-end development, user authentication, and performance optimization. It also facilitated the demonstration of effective project management, teamwork, and agile methodologies in a constrained time frame.

Overall, the proposed two-month duration was successfully met, resulting in the creation of the Tech Blog Website and showcasing the team's ability to deliver a valuable web application within limited project constraints.

REFERENCES

This document contains provisions which, through reference in this text, constitute provisions of the present document.

1) Google Search Engine for various searching

2) Learncodeonline.in

3) Techarge.in

4) w3schools.com