Technical Requirements Document for Notes App

**1. Overview**

The Notes App is a web application designed to provide users with the ability to create, read, update, and delete notes securely. This document outlines the technical requirements for building and deploying the application.

**2. Architecture**

* Frontend: React
* Backend: Node.js with Express
* Database: MongoDB (MongoDB Atlas)
* Authentication: JSON Web Tokens (JWT)
* Deployment: Vercel (Frontend) and Render (Backend)

**3. Frontend Requirements**

Framework and Libraries

* React: The main framework for building the user interface.
* React Router: For handling routing within the application.
* Axios or Fetch API: For making HTTP requests to the backend.
* Tailwind CSS: For styling the application.
* Redux or Context API: For state management (optional, based on complexity).

Components

* User Interface Components: Header, Footer, NoteCard, NoteList, NoteForm, etc.
* Pages: Home, Login, Register, Profile, NoteDetails, etc.

Functionality

* User Authentication:
  + Login and registration forms.
  + Token management (storing JWT securely).
* Note Management:
  + Forms for creating and updating notes.
  + Displaying lists of notes.
  + Search functionality.
* Responsive Design: Ensure the app is usable on mobile, tablet, and desktop devices.

##### **Deployment**

* **Vercel:** Configure and deploy the frontend application.

**4. Backend Requirements**

Environment and Libraries

* Node.js: JavaScript runtime for server-side development.
* Express: Web framework for building the API.
* Mongoose: ODM for MongoDB.
* dotenv: For managing environment variables.
* jsonwebtoken: For generating and verifying JWTs.
* bcrypt: For hashing passwords.

API Endpoints

User Management:

* POST /create-account: Register a new user.
* POST /login: Authenticate a user and return a JWT.
* GET /get-user: Retrieve authenticated user information.

Note Management:

* POST /add-note: Create a new note.
* GET /get-all-notes: Retrieve all notes for the authenticated user.
* PUT /edit-note/:noteId: Update an existing note.
* PUT /update-note-pinned/:noteId: Update the pinned status of a note.
* DELETE /delete-note/:noteId: Delete a note.
* GET /search-notes: Search for notes by query.

Middleware

* Authentication Middleware: Verify JWT and protect routes.
* Error Handling Middleware: Standardise error responses.

Database Schema

* User Schema:
* fullName: String
* email: String (unique)
* password: String (hashed)

Note Schema:

* title: String
* content: String
* tags: [String]
* isPinned: Boolean (default: false)
* userId: ObjectId (reference to User)

Deployment

* Render: Configure and deploy the backend application.

**5. Security Requirements**

Authentication and Authorization

* JWT: Use JSON Web Tokens for securing API endpoints.
* Password Hashing: Use bcrypt to hash passwords before storing them in the database.

Data Validation

* Validation Libraries: Use libraries like Joi or express-validator to validate request data.

CORS

* CORS Middleware: Configure CORS to allow requests from the frontend domain.

Environment Variables

* dotenv: Manage sensitive data using environment variables.

**6. Performance Requirements**

API Performance

* Response Time: Ensure API response times are under 200ms for most operations.
* Scalability: Design the backend to handle increasing numbers of users and notes efficiently.

Frontend Performance

* Lazy Loading: Implement lazy loading for large components or routes.
* Optimised Assets: Minify CSS and JavaScript, optimise images.

**7. Deployment and CI/CD**

Version Control

* GitHub: Use GitHub for version control and collaboration.

Continuous Integration/Continuous Deployment (CI/CD)

* Vercel: Configure CI/CD for automatic deployment of the frontend.
* Render: Configure CI/CD for automatic deployment of the backend.

Monitoring and Logging

* Logging: Implement logging for backend errors and significant events.
* Monitoring: Use monitoring tools to track the performance and health of the deployed applications.

**8. Testing Requirements**

Unit Testing

* Frontend: Use Jest and React Testing Library for component testing.
* Backend: Use Jest and Supertest for API endpoint testing.

Integration Testing

* Full-Stack Testing: Ensure end-to-end functionality works as expected.

Security Testing

* Vulnerability Scanning: Regularly scan for security vulnerabilities.
* Penetration Testing: Perform penetration testing to identify and fix security issues.

## Summary

This document outlines the technical requirements for building and deploying the Notes App. It covers the frontend and backend frameworks, libraries, API endpoints, security measures, performance benchmarks, deployment strategies, and testing protocols. Adhering to these requirements will ensure the successful implementation and deployment of a secure, scalable, and user-friendly Notes App.