LOVELY PROFESSIONAL UNIVERSITY

Article Management System

Software Requirements System

Name – Van Raj Thakur

Reg no – 12111596

Roll no – RK21AKA22

Submitted To Mrs. Divya Thakur

Assistant Professor
School of Computer Science and Engineering

Table of Contents

Table of Contents

1. Introduction

1.1 Purpose

This document serves as a comprehensive guide to outline the software requirements for developing a blog website using NodeJS, MongoDB, Postman, and EJS. It aims to clarify the project's goals, functionalities, and technical specifications.

1.2 Scope

The scope encompasses the creation of a user-friendly web application that facilitates the management of articles through CRUD operations (Create, Read, Update, Delete). Users can interact with the system to publish new articles, browse existing ones, modify content, and remove articles as needed.

Definitions

CRUD Operations: Create, Read, Update, Delete operations used for managing data in a database system.

NodeJS: A JavaScript runtime environment that allows server-side scripting for web applications.

MongoDB: A NoSQL database system used for storing and retrieving structured data.

Postman: An API development tool used for testing and debugging API endpoints.

EJS (Embedded JavaScript): A templating language that allows embedding JavaScript code within HTML templates for dynamic content generation.

Acronyms and Abbreviations

HTTP/HTTPS: Hypertext Transfer Protocol/Secure, protocols for communication between clients and servers over the web.

DFD: Data Flow Diagrams, visual representations of data flow within a system.

CRUD: Create, Read, Update, Delete operations for data management.

API: Application Programming Interface, a set of rules and protocols for building and interacting with software applications.

XSS: Cross-Site Scripting, a type of security vulnerability in web applications.

SQL: Structured Query Language, a programming language used for managing and manipulating relational databases.

2. General Purpose

2.1 Product Perspective

This section describes how the blog website fits into the broader context of software systems, emphasizing its standalone nature and its interaction with the MongoDB database for data storage and retrieval.

2.2 Product Functions

Detailed functionalities are outlined, covering the ability to create new articles, read existing ones, update article content, delete articles, and manage user roles through authentication and authorization mechanisms.

2.3 User Characteristics

User personas are defined, including authors who create and manage articles and readers who browse and consume content. Understanding user roles is crucial for designing an intuitive and effective user experience.

2.4 General Constraints

Constraints such as compatibility with modern web browsers, data integrity requirements, and security measures are specified to ensure the application meets industry standards and user expectations.

2.5 Assumptions and Dependencies

Assumptions about user knowledge and dependencies on MongoDB for data management are stated, providing context for development and deployment considerations.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Detailed descriptions of user interfaces, including the homepage, article creation page, article details page, login page, and admin dashboard, are provided to guide the frontend development process.

3.1.2 Hardware Interfaces

Since the application is web-based, no specific hardware interfaces are required, focusing instead on software and communication interfaces.

3.1.3 Software Interfaces

Dependencies on NodeJS for server-side scripting, MongoDB for database operations, and Postman for API testing and development are specified to ensure seamless integration and functionality.

3.1.4 Communication Interface

The use of HTTP/HTTPS protocols for client-server communication is mentioned, highlighting the need for secure and reliable data exchange.

3.2 Functional Requirements

Detailed functional requirements outline user management capabilities, article management functionalities, search functionality, security measures, and error handling mechanisms to ensure a robust and user-friendly application.

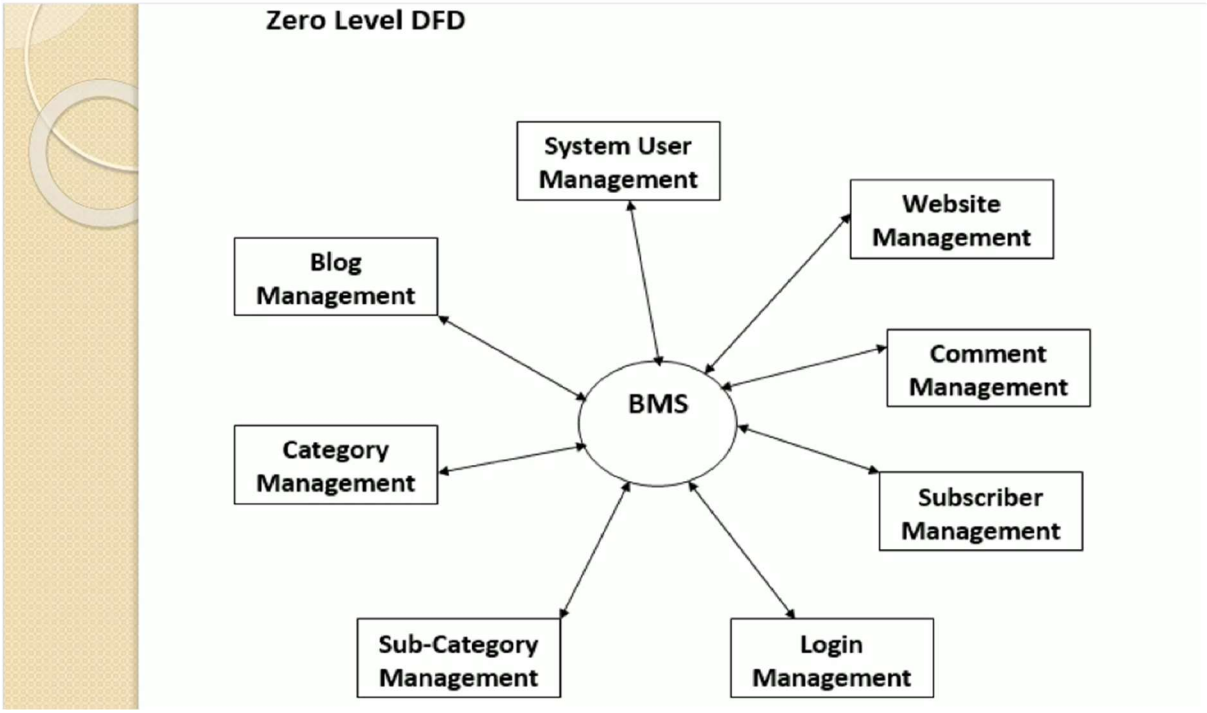
3.3 Non-Functional Requirements

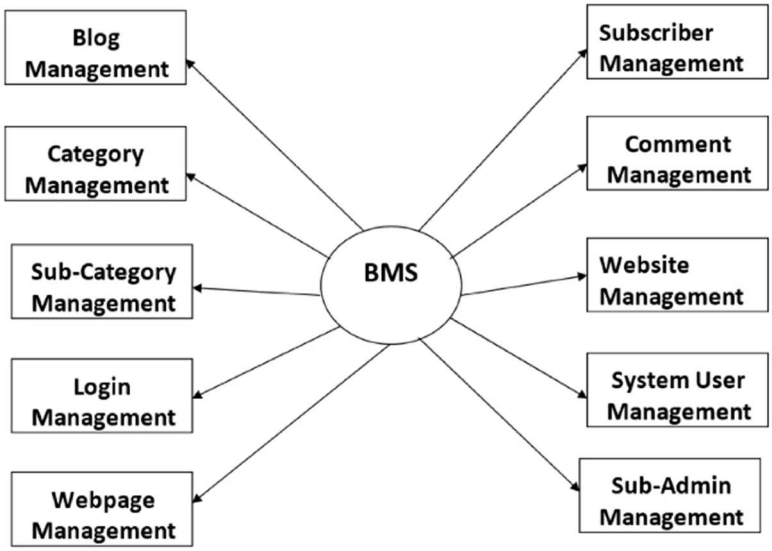
Non-functional requirements focus on performance, security, usability, and reliability aspects, setting expectations for system responsiveness, data protection, user experience, and system stability.

4. Analysis Models

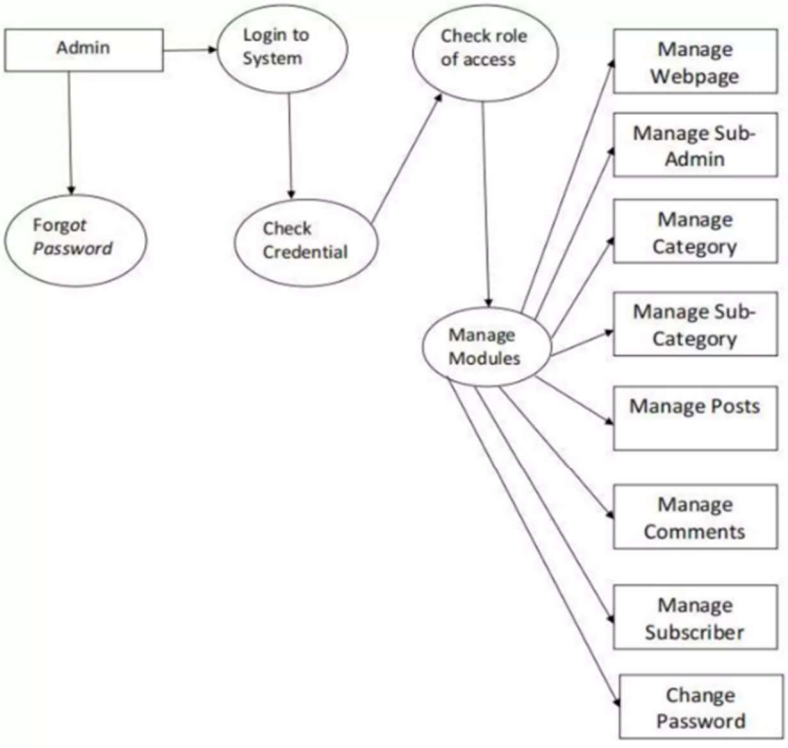
4.1 Data Flow Diagrams (DFD)

Data flow diagrams illustrate the flow of data within the system, including inputs, processes, and outputs, providing a visual representation of how information moves through the application.





Second Level DFD



5. GitHub Link

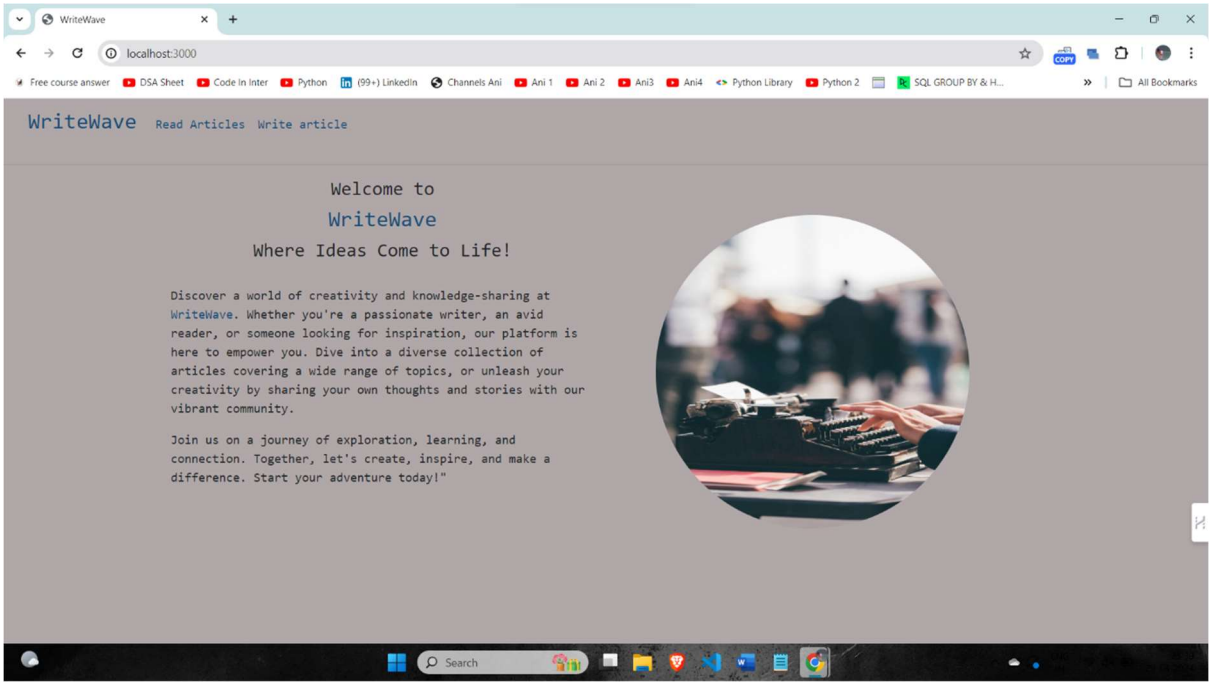
A link to the GitHub repository for the project is provided for version control, collaboration, and code management purposes.

https://github.com/VRThakur20/NodeJS_Project

6. Screenshots

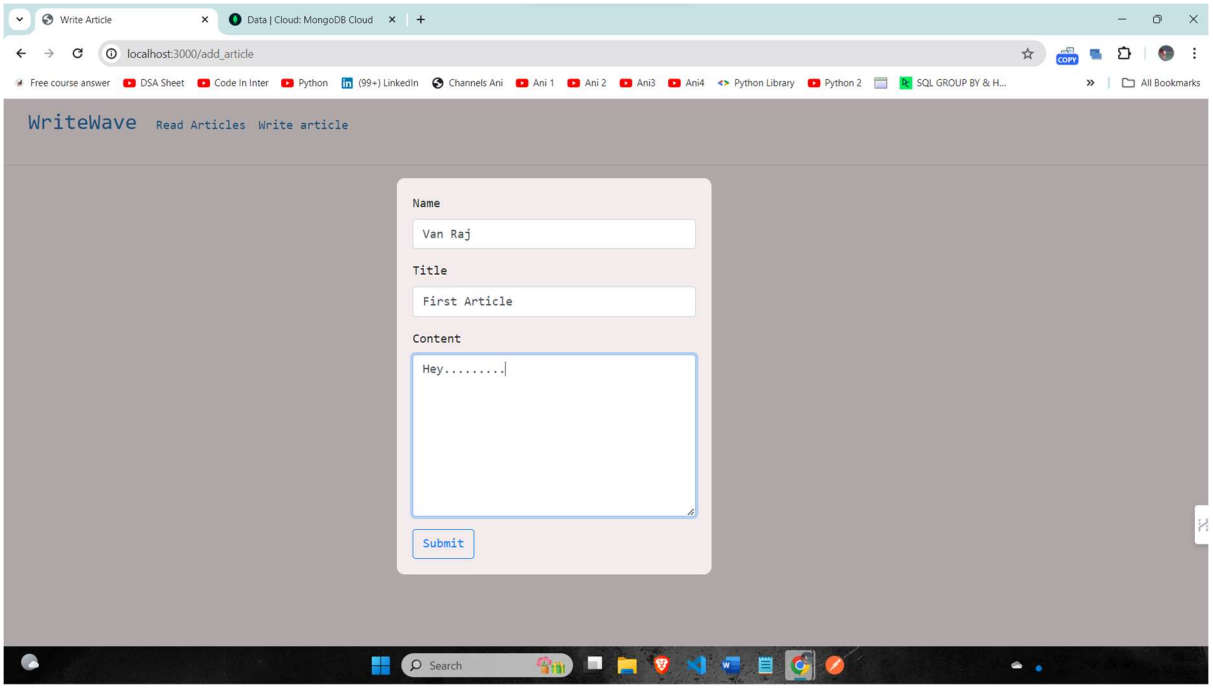
6.1 Homepage

[Include a screenshot of the homepage displaying articles.]



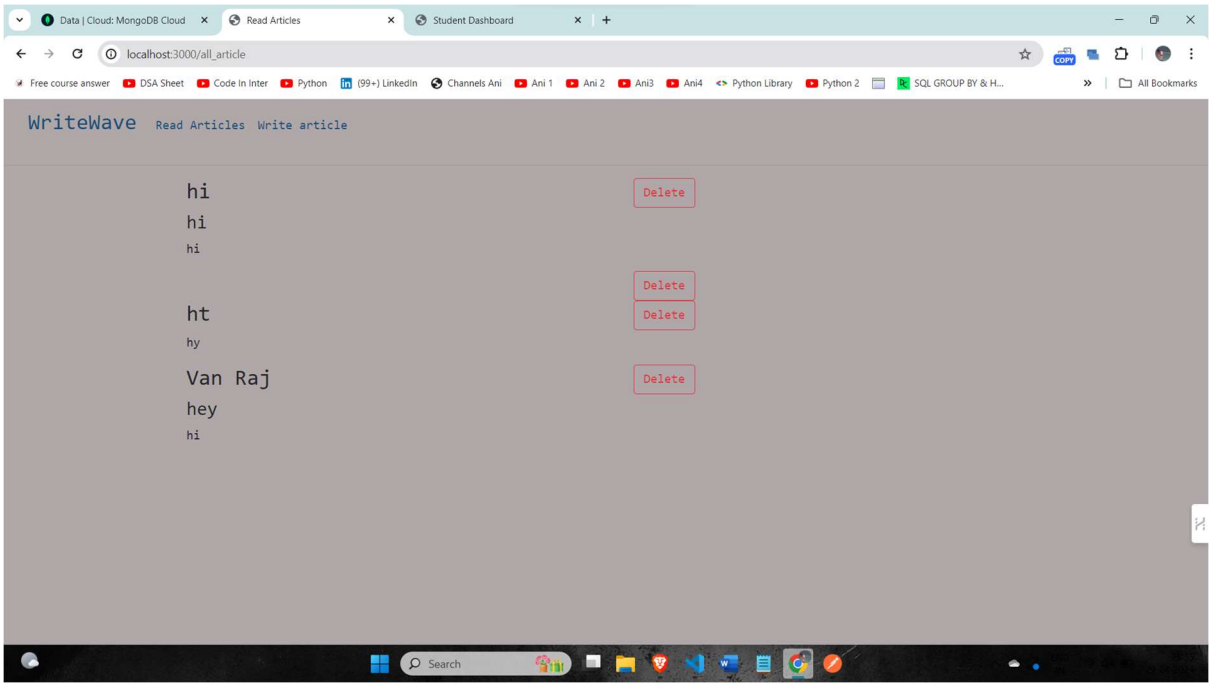
6.2 Write Articles

[Include a screenshot of the article creation page.]



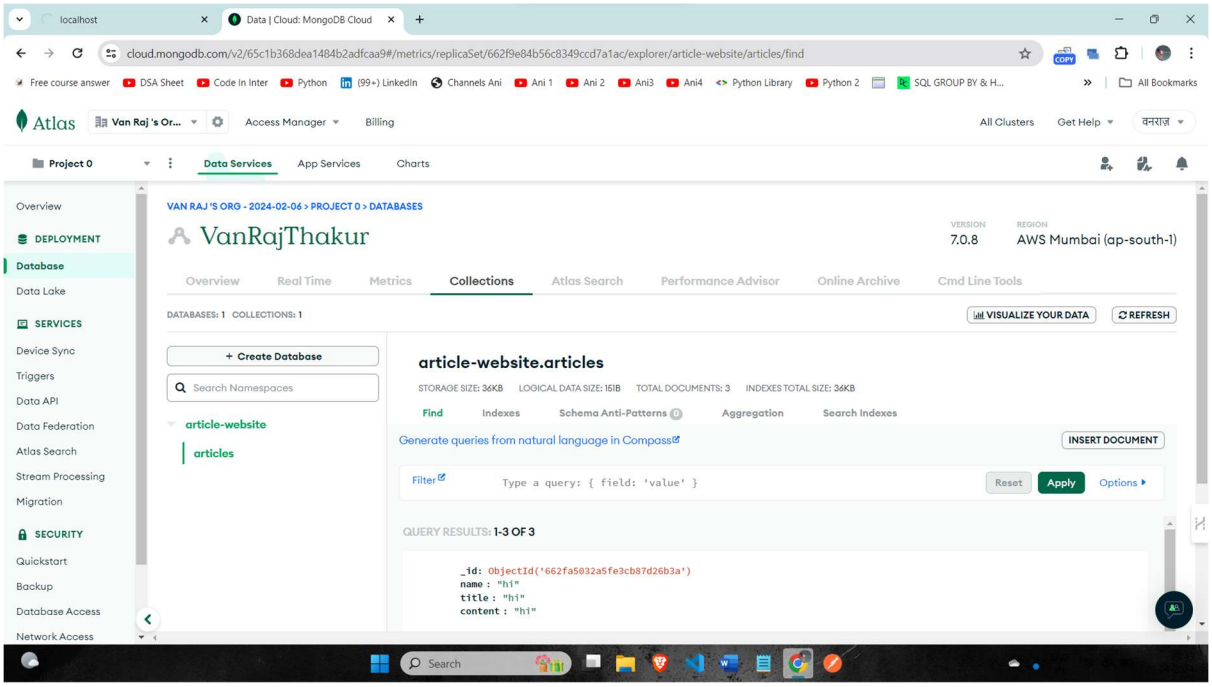
6.3 Read Articles

[Include a screenshot of the article details page.]



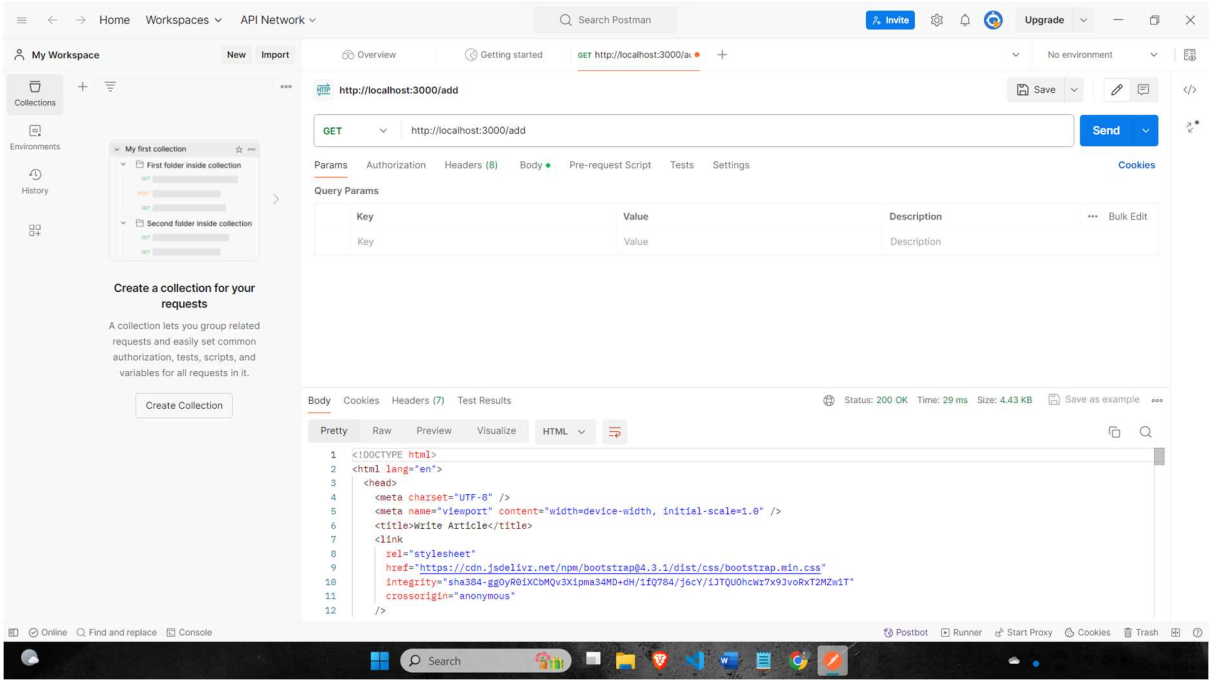
6.4 MongoDB Database

[Include a screenshot of the MongoDB database structure.]



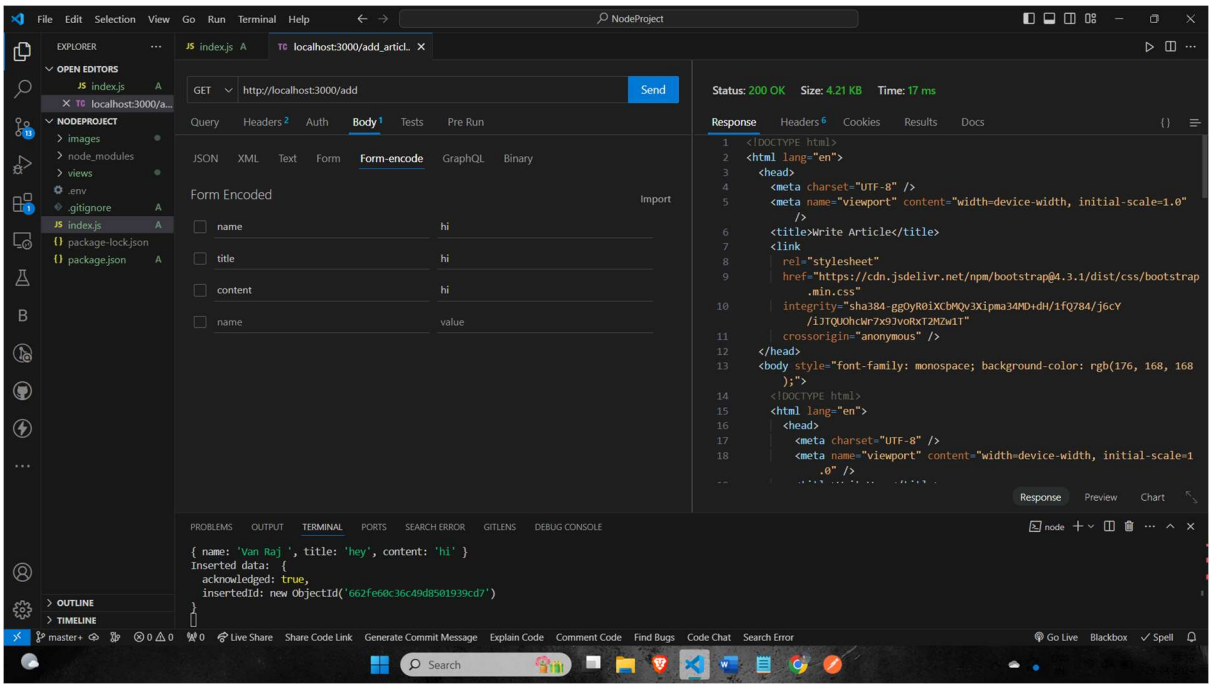
6.5 Postman Server

[Include a screenshot of the Postman server testing the API endpoints.]



6.6 Thunder Client

[Include a screenshot of Thunder Client testing API endpoints.]



7. Appendices

7.1 Architectural Diagram

Include an architectural diagram that illustrates the overall structure of the blog website application, showcasing components such as the frontend interface, backend server, database management, and external API interactions.

7.2 Workflow Diagram

Provide a workflow diagram depicting the step-by-step process of how users interact with the system, from registration and article creation to browsing and managing content, highlighting key functionalities and user pathways.

7.3 Technical Specifications

Include detailed technical specifications for the development environment, such as:

Programming languages used (JavaScript, HTML, CSS)

Frameworks and libraries (NodeJS, Express, EJS)

Database specifications (MongoDB configuration, data models)

API endpoints and routes

Security measures (authentication mechanisms, data encryption)

Deployment considerations (hosting platform, server setup)

7.4 Test Cases

List out test cases for various functionalities of the application, including:

User authentication and authorization testing

CRUD operations testing for articles

Search functionality testing

Error handling and validation testing

Performance testing (load testing, response times)

7.5 User Documentation

Provide user documentation or a user manual that explains how to use the blog website application, including:

Registration and login instructions

Article creation and management guidelines

Search and navigation tips

Troubleshooting common issues

Contact information for support or feedback

7.6 Code Repository

Include a link to the code repository (e.g., GitHub repository) where the source code for the blog website application is hosted, allowing stakeholders and developers to access and review the codebase.