VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

WEB APPLICATION DEVELOPMENT PROJECT

By

Phạm Vũ Quang – ITCSIU21096

Mã Phùng Nghĩa – ITCSIU21206

Nguyễn Bình Phương Huy – ITCSIU21189

CONTENT MANAGEMENT SYSTEM

Instructors: Assoc. Prof. Nguyen Van Sinh & MSc. Nguyen Trung Nghia

Table of Contents

A. INTRODUCTION

I. OBJECTIVE

With the rapid advancement of web applications and software in the IT sector, numerous websites have been created by individuals for a variety of purposes. Most of these websites utilize a range of effective web development techniques. In response to this trend, our web development team —CMS Team — has undertaken a project as part of the Web Application Development Course at International University, Vietnam National University. This project aims to implement a Content Management System while also enhancing it by incorporating various web development technologies. In addition, our product supports Markdown, a lightweight markup language that enables users to format plain text without employing HTML tags or a formal text editor. Markdown is widely used for blogging, documentation, instant messaging, and online forums.

II. PRODUCT’S INFORMATION

In today's digital landscape, the ability to efficiently manage content is critical for organizations of all sizes. A Content Management System (CMS) streamlines the process of creating, storing, and publishing digital content, making it essential for businesses, educators, and content creators. This project aims to develop a robust and user-friendly CMS that caters to the diverse needs of its users, from simple blogs to complex enterprise applications.

The primary objective of this Markdown Project is to design and implement a flexible Content Management System that allows users to create, edit, and publish content effortlessly.

III. DEVELOPMENT PROCESS

For ease of maintainability and scalability, our team has decided to adopt the agile methodology for our project. Agile’s iterative development allows us to improve and adapt faster to the changing requirements during the development time. Furthermore, applying this approach gives us a chance to enhance the collaboration and communication between members, which is extremely helpful throughout the development process.

Our development process follows the process as image below:

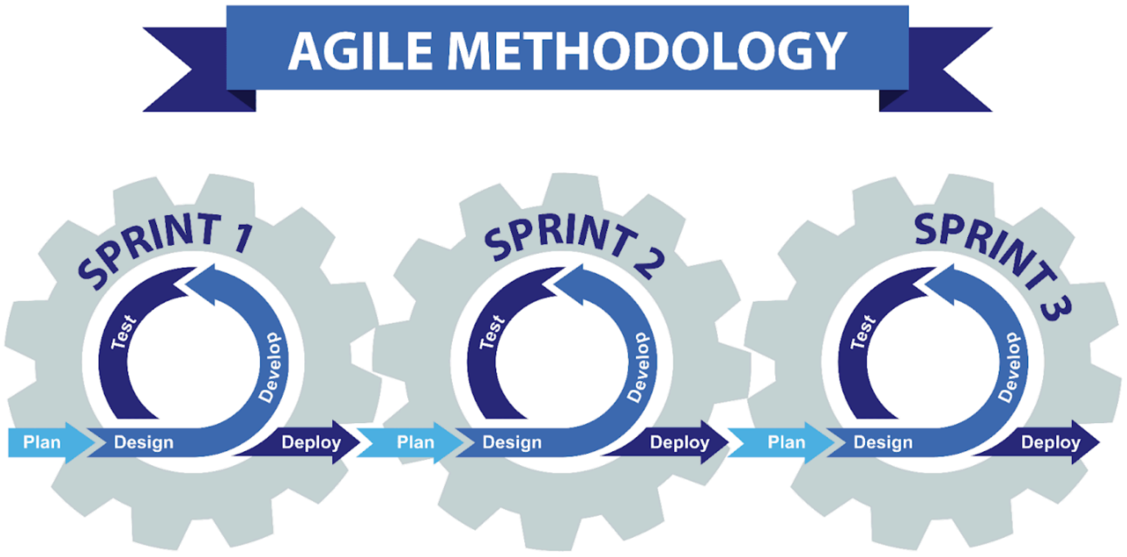


Figure 1. Agile Development Process

IV. WEB TECHNIQUES AND PROGRAMMING LANGUAGES

Due to the fact that this is the website, the product is demonstrated using some Web Application Design and Programming Language under the model of MVC. Techniques, programming languages and tools are used within our system:

a) Techniques:

1. **Monorepo Architecture**: A centralized repository designed to manage both frontend and backend code.
2. **Server – Side Rendering (SSR):** Employed in the frontend to enhance page load speeds and boost SEO.
3. **Dockerized Development Environment**: Utilizes Docker and configuration files to create consistent development setups.
4. **Agile Development Practices**: Focuses on iterative and incremental development through methods like sprint planning, daily stand-ups, and retrospectives.

5. **Continuous Integration and Deployment (CI/CD):** Automates the processes of building, testing, and deploying changes to the code.
6. **Role – Based Access Control (RBAC)**: Controls user permissions and access to features based on their roles.
7. **SEO Technique**: Implements distinctive, SEO-friendly slugs for content.

b) Language and tools

From the above listed techniques, this project are built from these following programming language and tools :

1. **TypeScript**: Utilized for both frontend and backend development to provide type safety and enhanced developer experience.

2. **Next.js**: A React-based framework used for building the frontend application, with server-side rendering (SSR) to improve performance and SEO.

3. **Jotai**: A state management library employed to manage application state efficiently.

4. **PostgreSQL (via Supabase)**: Used as the relational database to manage structured data, including user accounts and other application data.

5. **Prisma**: An ORM (Object-Relational Mapping) tool that simplifies database interactions, offering a type-safe query experience with PostgreSQL.

6. **Markdown**: Applied for content creation to format and render rich-text material seamlessly.

Programming Tools: Our project was conducted in Visual Studio Code, a well-known and lightweight code editor that offers strong capabilities for debugging, extensions, and an integrated terminal. This CMS allows for editing and debugging of both the frontend (Next.js) and backend (Go) codebases. Its compatibility with DevPod and Docker enhances the development workflow.

B. REQUIREMENT ANALYSIS AND DESIGN

This section provides a concise overview of the requirements analysis and design process. This iteration has been developed to establish a roadmap for the future execution of the project. Using this requirements specification as our foundation, we will implement each function in alignment with the specified conditions, as well as the functional and non-functional requirements outlined by the clients. Throughout the implementation phase, we will continually revise and update this document to ensure a clear understanding of the project's progress.

I. REQUIREMENT ANALYSIS

a) USE CASES

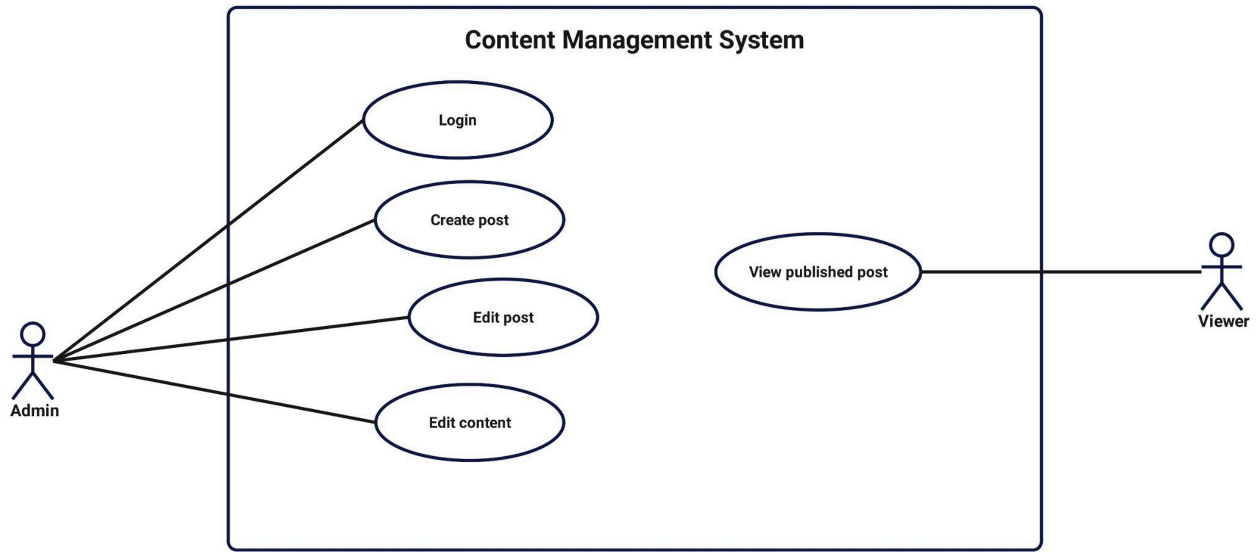


Figure 2. Use Case Diagram - Content Management System

b) FUNCTIONAL REQUIREMENTS

The content management system (CMS) is architected as a monorepo, integrating a frontend developed with Nuxt.js and a backend constructed using the Go programming language. This document delineates the essential functionalities that the CMS must encompass to effectively address the requirements of its diverse user base, which includes administrators, developers, and content creators.

1. **User Roles and Permissions**

i. Role – Based Access Control:

- The content management system needs to accommodate various user roles such as Admin, Editor, Viewer, and Developer.
- Access to specific features or data must be adjustable based on the role to either limit or grant permissions.

ii. Authentication and Authorization

- Users must log in using secure credentials.
- The system should support single sign-on (SSO) and multi-factor authentication (MFA).

iii. Content Management Features

a. Content Creation and Editing

- Users should have the capability to create, modify, and remove content through a rich text editor that offers support for Markdown.
- Drafts must be auto-saving to prevent loss of unsaved changes.

b. Media Management

- Users should have the capability to upload, categorize, and handle media files including images, videos, and documents.
- Media files need to be stored in a secure manner and should include metadata (such as file name, size, and upload date).

c. Content Organization

- Content needs to be organized using a versatile system of categories and tags.
- A slug field is necessary to create URLs that are optimized for SEO.

iv. Frontend Features

a. ***Dynamic Content Rendering***:

- The frontend should display content dynamically according to user roles and permissions.
- Server - side rendering (SSR) needs to be integrated to enhance page loading speeds and boost SEO.

b. ***Responsive Design:*** The frontend should be completely responsive and usable on devices with different screen dimensions.

c. ***Search Functionality***: Users should have the capability to find content by utilizing keywords, tags, or categories.

v. Backend Features

a. Content API:

- The backend needs to offer RESTful APIs for content management and retrieval.
- APIs should enable pagination, filtering, and sorting to enhance performance and user experience.

b. Database Management

- The backend needs to handle content information, user information, and media metadata using a relational database.
- Migration scripts should be included for updates to the database.

c. ***Tag and Category Management***: The system should enable the ability to perform CRUD (Create, Read, Update, Delete) functions for tags and categories.

vi. Developer Features:

a. ***Monorepo Support*****:** The CMS ought to adopt a monorepo architecture to enable developers to handle both frontend and backend codebases cohesively.

b. ***Environment Setup***: Developers should have the capability to configure their environment utilizing the supplied DevPod settings and Docker images.

c. ***Extensibility:*** The system should allow for the incorporation of new features via a plugin or module framework.

vii. Workflow and Automation

a. ***Scheduled Publishing*****:** Users should have the ability to plan content for publication at a later time.

b. ***Automated Backups*****:** The content and configuration data should be backed up periodically by the CMS.

c. ***Notifications*****:** Users should receive notifications for workflow activities like content approval, publishing, or errors.

viii. Reporting and Analytics

a. ***Content Insights*****:** The system should deliver insights regarding the performance of content, including metrics like views, likes, and shares.

b. ***User Activity Tracking*****:** Administrators need to monitor user actions for the sake of audits.

c. ***Error Logging***: Mistakes in the system must be recorded and communicated to administrators.

ix. Integration Features

a. ***Third – Party Services***: The CMS needs to work seamlessly with external tools like Google Analytics, email services, and cloud storage solutions.

b. ***Import / Export Functionality***: Users should have the capability to import and export data in common formats (such as JSON, XML, CSV).

The functional requirements specified in this document outline the essential capabilities of the CMS. These features guarantee that the system fulfills user needs, facilitates efficient content management, and offers a strong platform for development and deployment. Complying with these requirements will lead to a scalable, secure, and user – friendly CMS.

c) NON – FUNCTIONAL REQUIREMENTS

This document details the non-functional specifications for the Content Management System (CMS). The CMS prioritizes performance, scalability, development simplicity, and secure deployment to effectively facilitate content management activities.

i. Performance Requirements

a. ***Response Time:*** The system is designed to ensure that user actions, such as retrieving or saving content, are executed within a timeframe of two seconds under normal operational conditions.

b. ***Scalability:*** The backend, developed in Go, needs to effectively manage high-concurrency situations, accommodating a minimum of 10,000 concurrent users.

c. ***Availability:*** To ensure minimal disruption, it is essential for the system to maintain an uptime of 99.9%.

ii. Usability Requirements

a. ***Developer – Friendly Setup***: Developers ought to configure the environment with DevPod, utilizing Docker to ensure uniformity and standardization.

b. ***Ease of Development***: The deployment procedure should replicate the development environment to reduce the chances of encountering deployment-related problems.

c. ***Cross – Browser Compatibility***: To ensure a seamless user experience, the frontend must operate flawlessly across all major web browsers, including Chrome, Firefox, Safari, and Edge.

iii. Maintainability and Extensibility

a. ***Monorepo Architecture***: The monorepo design should promote teamwork and efficient updates between the frontend and backend components.

b. ***Modular Design***: Both the frontend (using Nuxt.js) and the backend (utilizing Go) parts should implement a modular structure, allowing for standalone updates or the integration of new features.

c. ***Documentation***: Detailed setup and usage instructions for the /web and /server directories need to be supplied to facilitate swift onboarding and problem-solving for developers.

iv. Reliability Requirements

a. ***Fault Tolerance***: The backend server should effectively manage errors, including network disruptions or database connectivity issues, without fail.

b. ***Data Integrity***: The system needs to guarantee consistent and dependable management of user data, safeguarding it against loss or corruption during processes.

v. Security Requirements

a. ***Secure Communication***: To ensure the security and confidentiality of data, all communication between the frontend and backend should be conducted using HTTPS protocols for data encryption.

b. ***Access Control***: Access to different sections of the CMS should be restricted through role-based access control to avoid unauthorized entry.

c. ***Script Permissions***: Running scripts like start.sh should necessitate explicit permission to avoid unauthorized usage.

vi. Compatibility Requirements

a. ***Frontend – Backend Integration***: The frontend built with Nuxt.js and the backend developed in Go need to work together smoothly, guaranteeing consistent data transfer and system integration.

b. ***Development Tools Integration***: The system should seamlessly work with widely used development tools and IDEs to create standardized environments.

vii. Portability Requirements

a. ***Containerized Environment***: The development and production environments should utilize Docker to guarantee uniform behavior across different systems.

b. ***Cross – Platform Support***: The CMS should operate smoothly on Windows, macOS, and Linux platforms without needing significant configuration efforts.

viii. Supportability Requirements

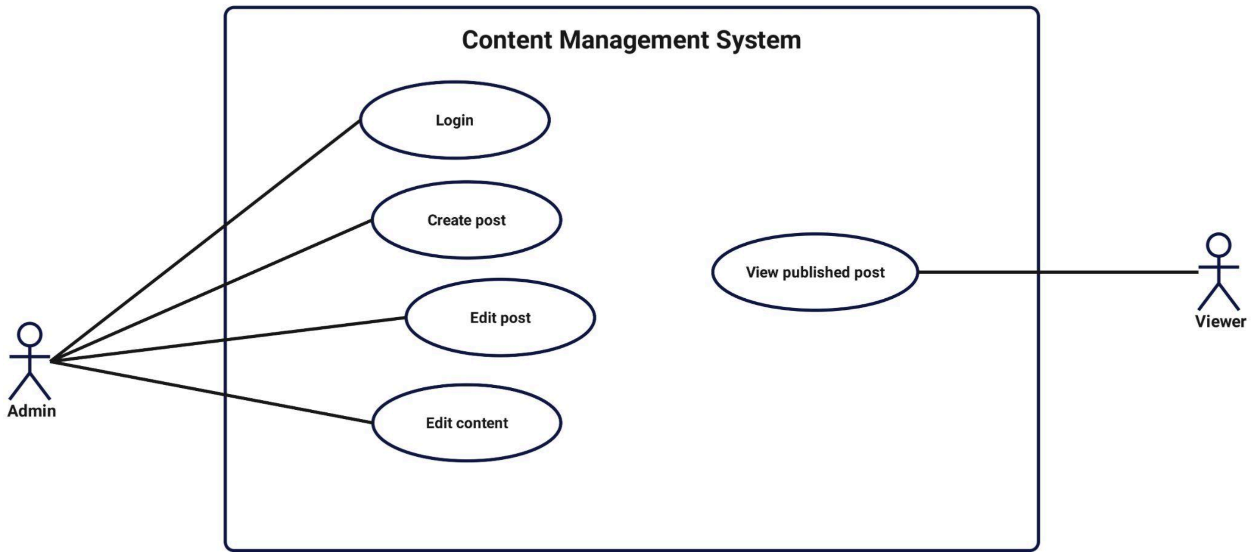
a. ***Logging and Monitoring***: Thorough logging systems ought to be implemented for both frontend and backend elements to assist in diagnosing problems and troubleshooting.

b. ***Version Control***: The monorepo should be compatible with version control systems (such as Git) to facilitate collaborative development and keep track of changes.

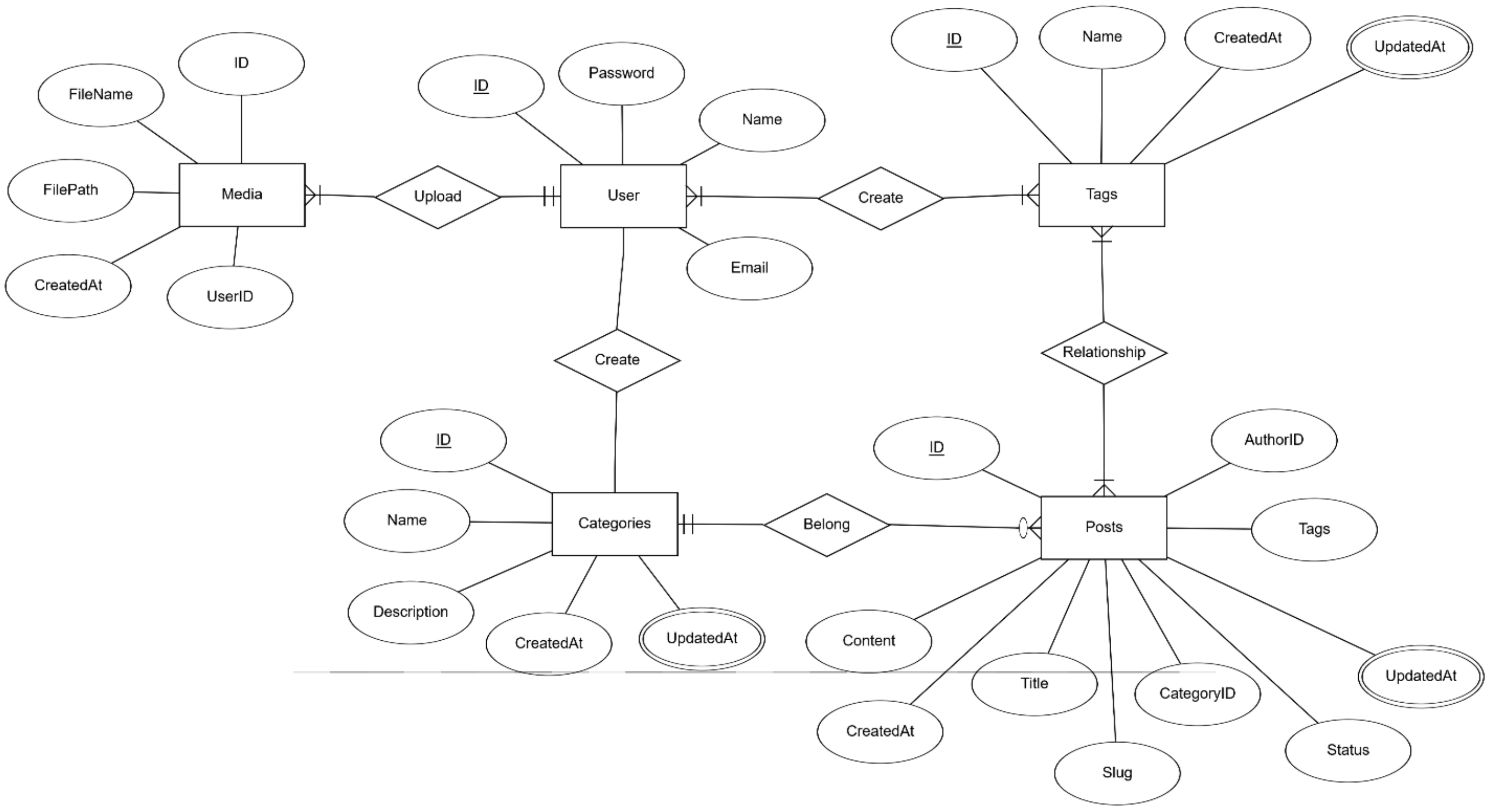
The non-functional requirements outlined above are designed to guarantee that the CMS provides exceptional performance, reliability, security, and user-friendliness for both developers and end-users. These criteria align with the architectural objectives of building a robust, scalable, and developer-centric system that meets contemporary standards in content management.

d) DIAGRAMS IMPLEMENTATION

Use Case Diagram



Entity – Relationship Diagram (ERD)



Sequence Diagram:

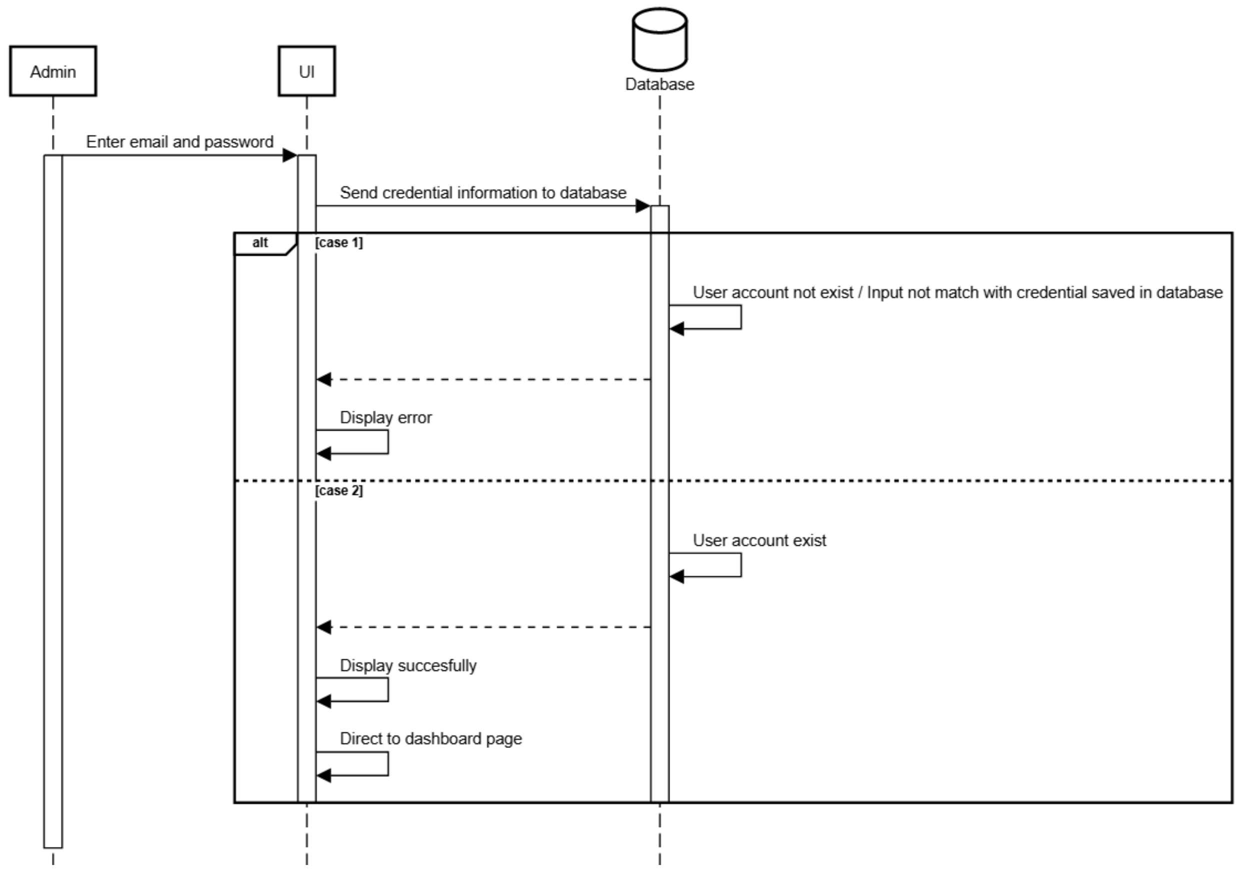


Figure 4. Sequence Diagram - Use Case 1: Login

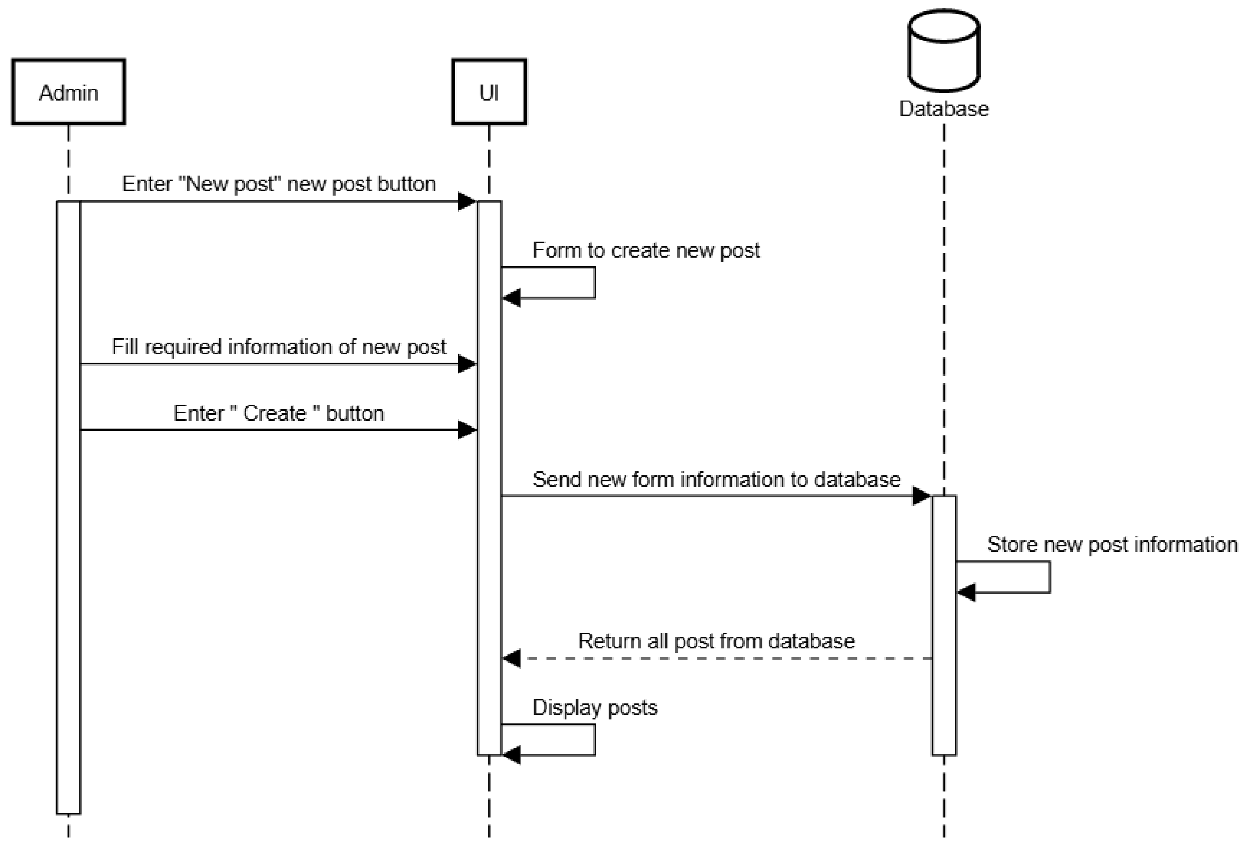

Admin
UI
Database
Enter "New post" new post button
Form to create new post
Fill required information of new post
Enter " Create " button
Send new form information to database
Store new post information
Return all post from database
Display posts

Figure 5. Sequence Diagram - Use Case 2: Create post

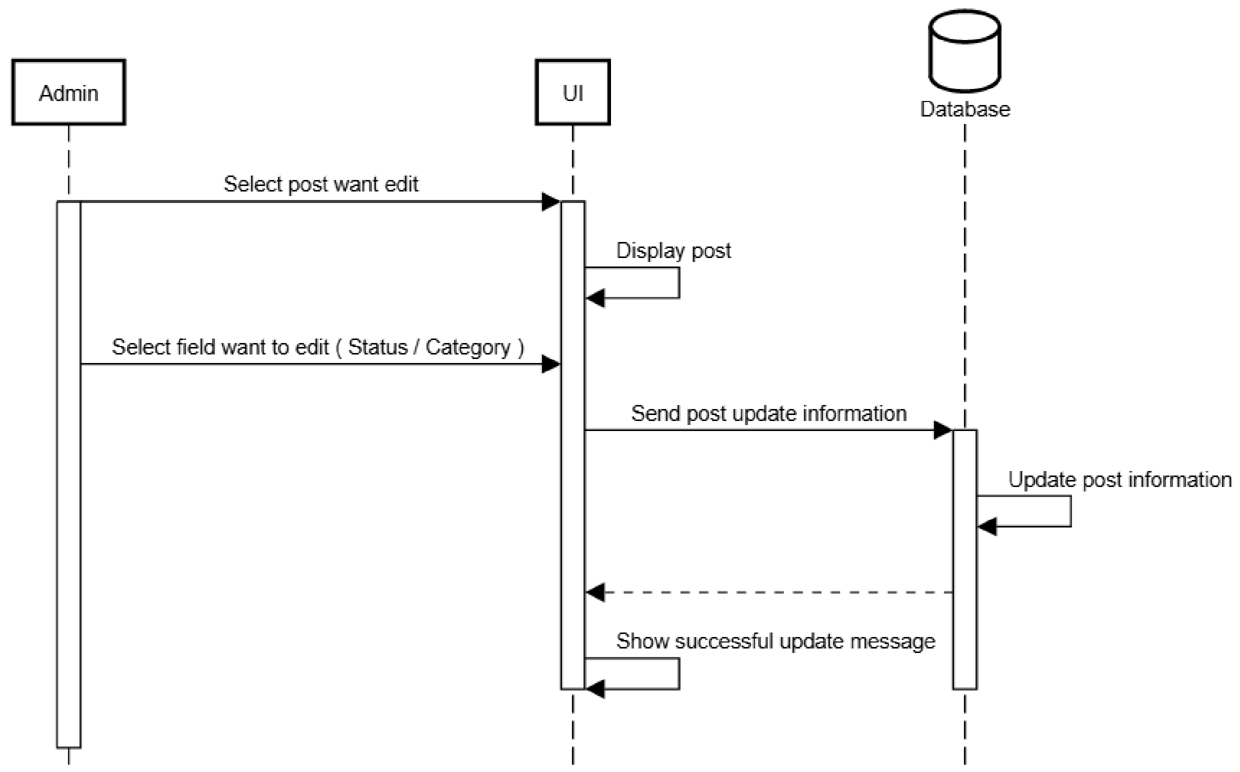

Admin
UI
Database
Select post want edit
Display post
Select field want to edit (Status / Category)
Send post update information
Update post information
Show successful update message

Figure 6. Sequence Diagram - Use Case 3: Edit post

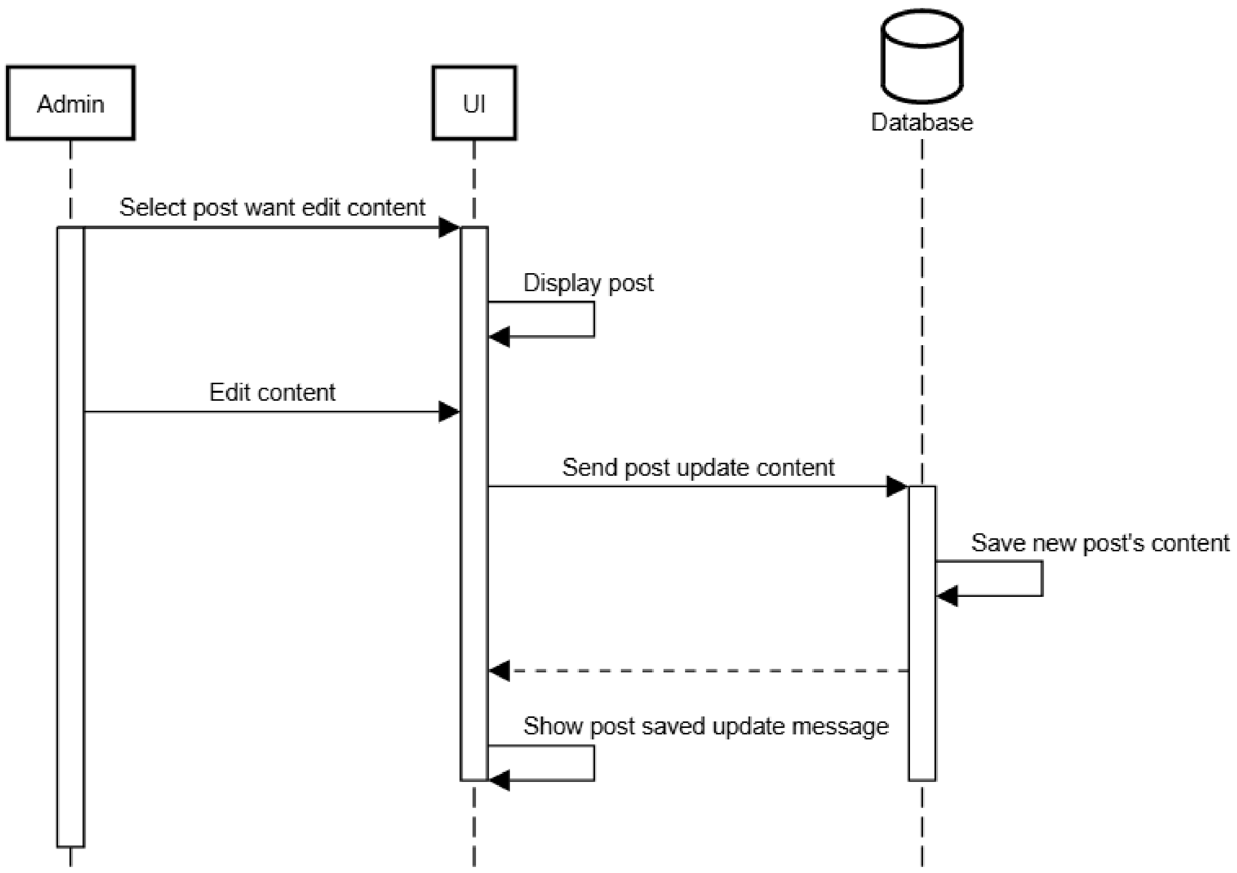


Figure 7. Sequence Diagram - Use Case 4: Edit content

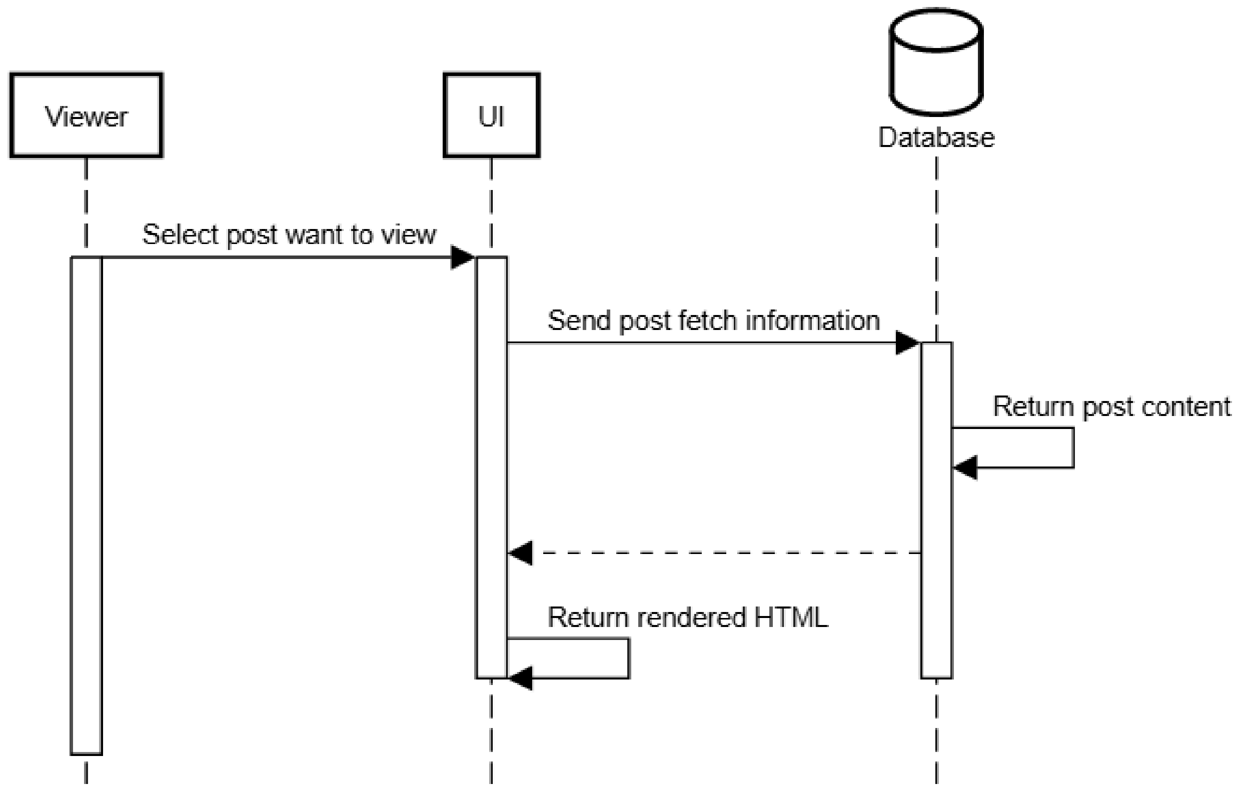
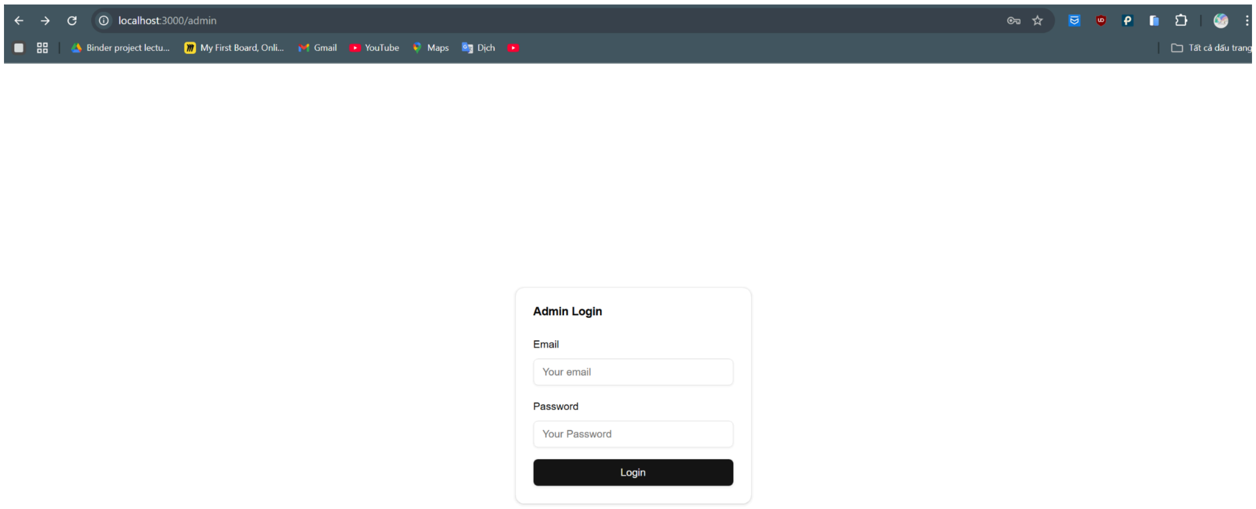
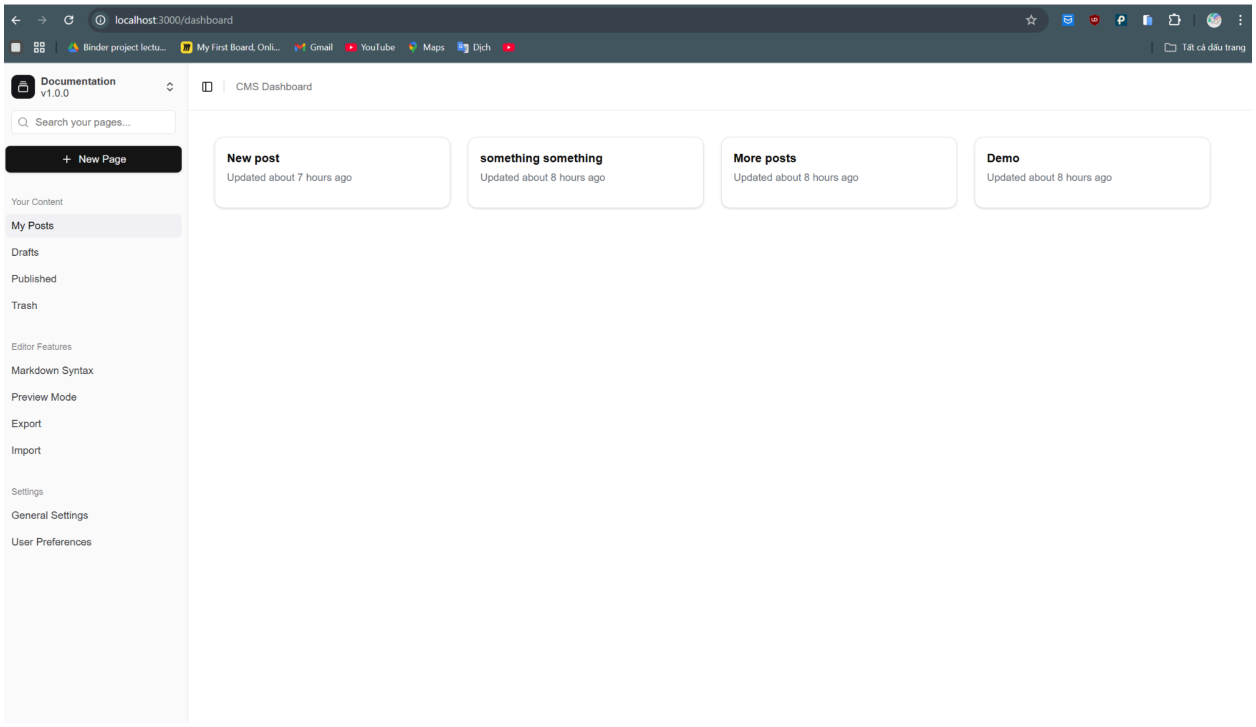


Figure 10. Sequence Diagram - Use Case 7: View published post

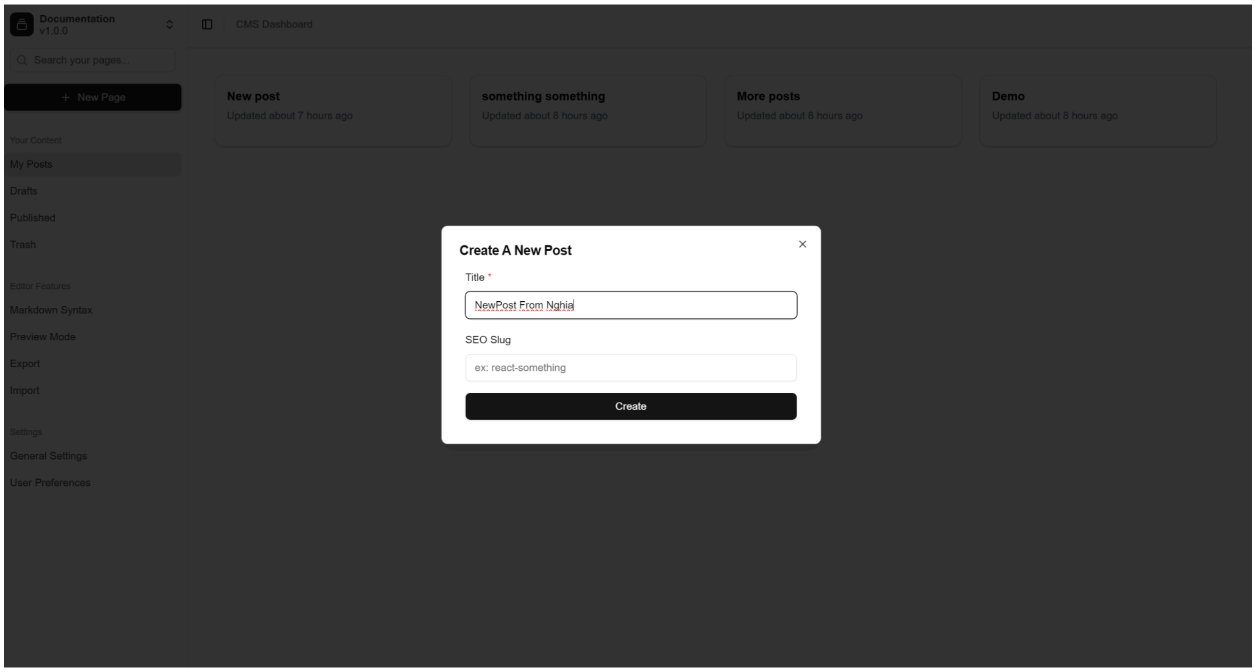
C. IMPLEMENTATION



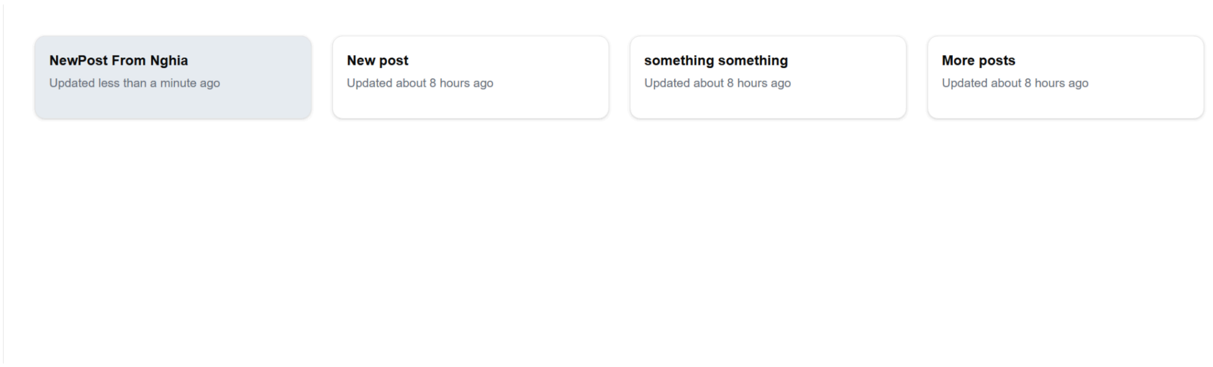
Admin login page



Dashboard



Create new post



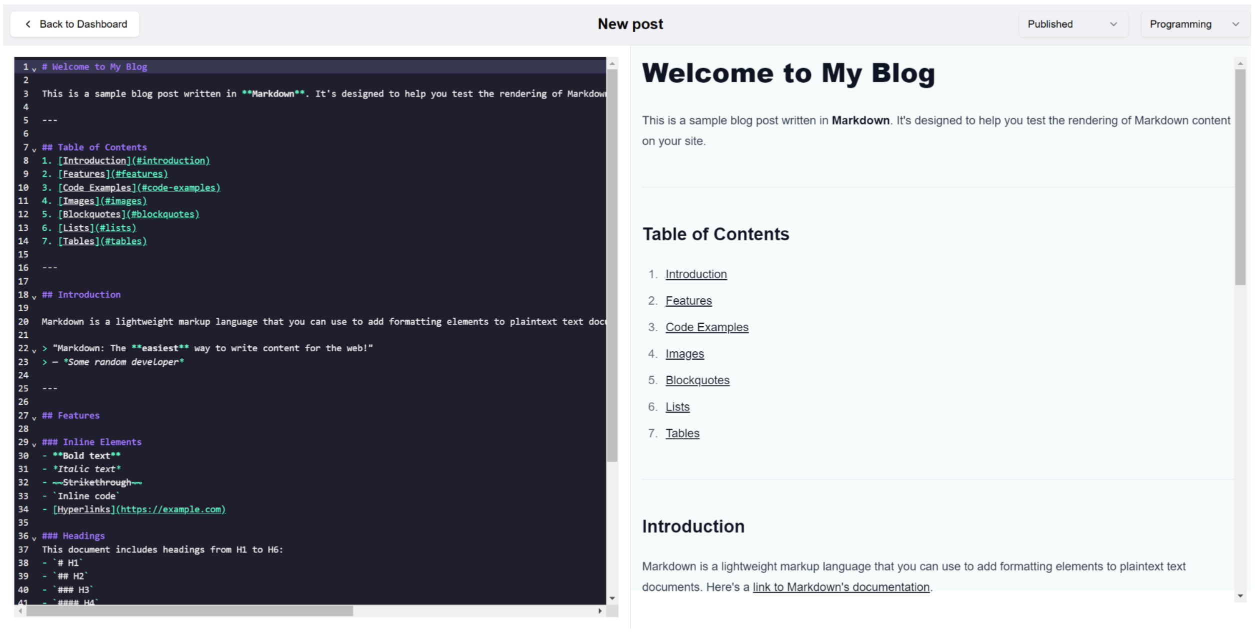
NewPost From Nghia
Updated less than a minute ago

New post
Updated about 8 hours ago

something something
Updated about 8 hours ago

More posts
Updated about 8 hours ago

Post is shown after created



```
# Welcome to My Blog

This is a sample blog post written in **Markdown**. It's designed to help you test the rendering of Markdow

---

## Table of Contents
1. [Introduction](#introduction)
2. [Features](#features)
3. [Code Examples](#code-examples)
4. [Images](#images)
5. [Blockquotes](#blockquotes)
6. [Lists](#lists)
7. [Tables](#tables)

---

## Introduction

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text doc

> "Markdown: The **easiest** way to write content for the web!"
> — *Some random developer*

---

## Features

### Inline Elements
- **Bold text**
- *Italic text*
- ~~Strikethrough~~
- `Inline code`
- [Hyperlinks](https://example.com)

### Headings
This document includes headings from H1 to H6:
- `# H1`
- `## H2`
- `### H3`
- `#### H4`
```

Welcome to My Blog

This is a sample blog post written in **Markdown**. It's designed to help you test the rendering of Markdown content on your site.

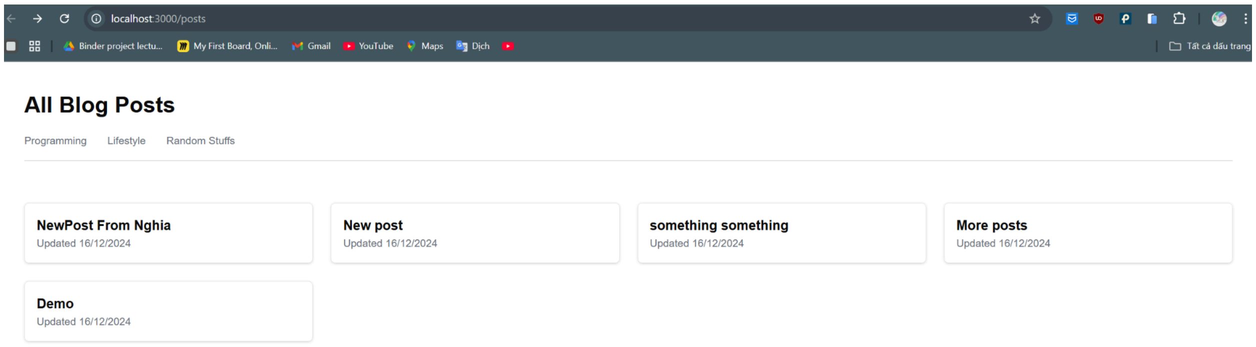
Table of Contents

1. Introduction
2. Features
3. Code Examples
4. Images
5. Blockquotes
6. Lists
7. Tables

Introduction

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Here's a link to Markdown's documentation.

Edit content of post



All Blog Posts

Programming Lifestyle Random Stuffs

NewPost From Nghia
Updated 16/12/2024

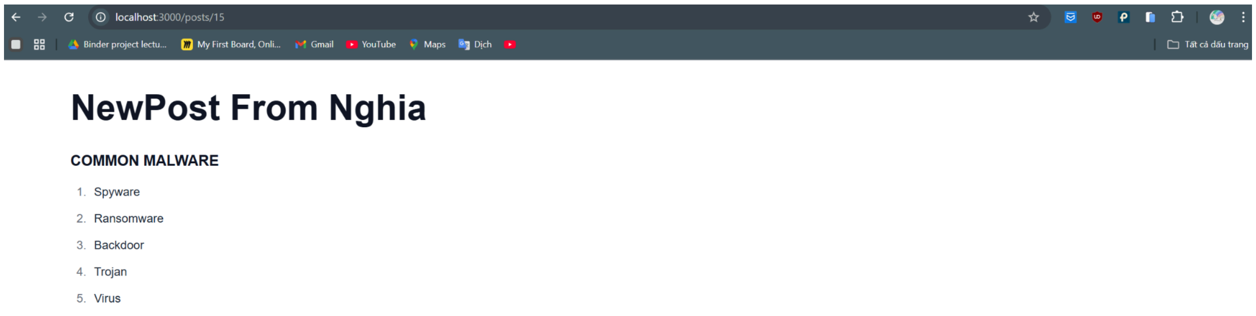
New post
Updated 16/12/2024

something something
Updated 16/12/2024

More posts
Updated 16/12/2024

Demo
Updated 16/12/2024

Page loaded when normal viewer access



NewPost From Nghia

COMMON MALWARE

1. Spyware
2. Ransomware
3. Backdoor
4. Trojan
5. Virus

The content from " New post " when user click

ACCOUNT TO TEST

Email : admin@admin.com

Password : 123456

D. REFERENCE

1. Document360. (2022, October 28). *Agile Documentation: Methodology & Best Practices*. Agile Documentation: Methodology & Best Practices. Retrieved October 10, 2024, from https://document360.com/blog/agile-documentation/

2. Hege, S., & Refsnes, J. E. (1998, January 01). *Learn to Code With the world's largest web developer site*. W3Schools Online Web Tutorials. Retrieved October 10, 2024, from https://www.w3schools.com/

3. Vercel, Inc. (2015, October 25). *Introduction to Next.js*. Introduction | Next.js. Retrieved November 02, 2024, from https://nextjs.org/docs

4. Walke, J. (2013, May 29). *React | The library for web and native user interfaces*. React. Retrieved November 02, 2024, from https://react.dev/

5. Wathan, A., Reinink, J., Hemphill, D., & Schoger, S. (2019, May 13). *Tailwind CSS*. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. Retrieved October 27, 2024, from https://tailwindcss.com/