THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: Information Technology

SPECIALTY: Data Science & Artificial Intelligence

PROJECT TITLE:

A Website Supporting Vietnamese Text Editing

Instructor: **PhD. PHAM CONG THANG**
Student: **BUI DUY HOANG**
Student ID: **102210314**
Class**: 21TCLC_KHL2**

Da Nang, 06/2025

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: Information Technology

SPECIALTY: Data Science & Artificial Intelligence

PROJECT TITLE:

A Website Supporting Vietnamese Text Editing

Instructor: **PhD. PHAM CONG THANG**
Student: **BUI DUY HOANG**
Student ID: **102210314**
Class**: 21TCLC_KHL2**

Da Nang, 06/2025

SUMMARY

Topic title: A Website Supporting Vietnamese Text Editing

Student name: Bui Duy Hoang

Student ID: 102210314 Class: 21TCLC_KHDL2

The project **"A Website Supporting Vietnamese Text Editing"** aims to provide a useful tool that enables users to easily and accurately compose content in Vietnamese. The website features intelligent sentence suggestions, text summarization, speech-to-text conversion, and an integrated smart assistant (chatbox). These functionalities help users save time when writing and minimize grammatical errors.

In addition, the tool is particularly beneficial in educational, administrative, and media environments, where high standards of Vietnamese writing are required. By integrating natural language processing (NLP) technology, the website goes beyond being a simple text editor—it serves as a Vietnamese language assistant that enhances writing quality and boosts user productivity.

DA NANG UNIVERSITY
UNIVERSITY OF SICIENCE AND TECHNOLOGY
FALCUTY OF INFORMATION TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: Bui Duy Hoang Student ID: 102210314

Class: 21TCLC_KHDL2 Faculty: Information Technology

Major: Data Science and Artificial Intelligence

1. *Topic title: A Website Supporting Vietnamese Text Editing*
2. *Project topic : ☐ has signed an intellectual property agreement for final result*
3. *Initial figure and data: None*

Content of the explanations and calculations:

Introduction –

Chapter 1: TOPIC OVERVIEW

Chapter 2: THEORETICAL BACKGROUND

Chapter 3: SYSTEM ANALYSIS AND DESIGN

Chapter 4: SYSTEM IMPLEMENTATION AND RESULTS

Conclusion –

4. *Drawings, charts (specify the types and sizes of drawings): None*
5. *Name of instructor: PhD. Pham Cong Thang*
6. *Date of assignment: ……../……./2025*
7. *Date of completion: ……../……./2025*

Da Nang , date month 06 year 2025

Head of Division……. **Instructor**

PREFACE

First and foremost, I would like to express my sincere gratitude and deepest appreciation to Dr. Pham Cong Thang for his dedicated guidance and support throughout the entire process of completing my graduation project. His invaluable mentorship has enabled me to acquire essential knowledge and successfully complete the project on schedule.

I would also like to extend my heartfelt thanks to all the lecturers in the Faculty of Information Technology, as well as all the teachers at the University of Science and Technology – The University of Danang, who have taught and supported me over the past four years. Their teachings have provided me with a strong foundation of knowledge for my academic journey.

I am also thankful to my classmates from class 21TCLC_KHDL2 for their encouragement and support during my time at the university.

Last but not least, I would like to express my boundless gratitude to my parents and family. Their unwavering support, constant motivation, and emotional encouragement have been a vital source of strength, helping me overcome challenges and complete this project.

Once again, I sincerely thank you all!

Sincerely,

Bui Duy Hoang

ASSURRANCE

I, Bui Duy Hoang, hereby declare that this graduation project titled "**A Website Supporting Vietnamese Text Editing**" is entirely my own work, conducted under the guidance of my supervisor, PhD. Pham Cong Thang. All contributions from external sources, including literature, code, and datasets, have been appropriately cited and referenced following academic standards.

I confirm that:

- The research and analyses presented in this report were performed by me.
- No part of this work has been plagiarized or submitted elsewhere to fulfill another academic requirement.

If wrong, I would like to take full responsibility and bear all the discipline of the faculty as well as the university.

Performed Student,

Bui Duy Hoang

TABLE OF CONTENT

LIST OF TABLES

LIST OF PICUTURES

LIST OF ACRONYMS

ACRONYM:

Word	Abbreviation of
API	Application Programming Interface
RESTful	Representational State Transfer
HTTP	Hypertext Transfer Protocol
DB	Database
NLP	Mesage Queuing Telemetry Transport
EC2	Amazon Elastic Compute Cloud
SSE	Sum of Squared Error

INTRODUCTION

In today's rapidly advancing digital age, the demand for fast, accurate, and efficient text editing has become increasingly important—especially for the Vietnamese language, which features rich and complex grammatical structures. However, most existing text editing tools primarily focus on English, while Vietnamese has yet to receive adequate attention. Therefore, the project titled **"A Website Supporting Vietnamese Text Editing"** was developed with the goal of creating a user-friendly online platform to assist users during the writing process through features such as sentence suggestions, text summarization, speech-to-text conversion, and a smart assistant (chatbox). These features aim to improve writing quality and save time.

The objective of the project is to develop a web-based system with a simple and intuitive interface, integrating features that support Vietnamese text composition using natural language processing (NLP) techniques. In addition, the system is designed to ensure high accuracy in sentence processing, offer quick responses, and maintain stable operation across common web browsers. The target users include students, teachers, office workers, and anyone in need of writing or editing Vietnamese documents online. The research scope primarily focuses on general-purpose text processing and does not delve into legal, technical, or highly specialized documents.

Throughout the development process, the applied research methods include surveying user needs, analyzing and designing the system based on function-oriented modeling, selecting appropriate technologies for implementation, and integrating Vietnamese NLP libraries. The system is also tested to evaluate its performance and effectiveness. The thesis is structured into several chapters, including an introduction to the project and related technologies; detailed system analysis and design; implementation using web technologies; testing and evaluation; and finally, conclusions along with future development directions.

Chapter 1: TOPIC OVERVIEW

1.1 . Overview

During both study and work, text editing is an essential skill. However, not everyone has the ability to express ideas fluently and coherently, or knows how to present content in a clear and grammatically correct manner. This is especially true for Vietnamese writing, where composing texts that are concise, accurate, and stylistically appropriate remains a challenge—particularly for students and general users.

Currently, there are several text editing tools on the market, such as Google Docs, Microsoft Word, and Vietnamese input method editors. However, these tools primarily focus on spell-checking and text input, and do not truly assist users in sentence construction, idea expression, or intelligent summarization.

In contrast, English-language tools like Grammarly and QuillBot have proven to be highly effective in assisting with writing, editing, and summarizing content. However, these tools lack full support for the Vietnamese language. This indicates a clear gap in the availability of user-friendly, Vietnamese-specific writing tools equipped with practical support features.

In response to this need, the project was initiated to develop a website that supports users in writing Vietnamese text more effectively. The website will integrate features such as sentence suggestions, text summarization, speech-to-text conversion, and a smart assistant (chatbox). It will be built as a web-based platform for easy access and use from anywhere with an internet connection.

This is a highly practical and applicable project that meets real-world demands and helps users save time while improving the quality of their Vietnamese writing.

1.2 Objectives and Significance of the Project

The main objective of this project is to develop a website that supports users in composing Vietnamese texts more easily, quickly, and effectively. The website is designed with a user-friendly interface and integrates key features such as sentence suggestion, text summarization, speech-to-text conversion, and a smart assistant

(chatbox). These functionalities aim to enhance users' writing capabilities and save time during the composition process.

Through this system, users no longer need to spend excessive time thinking about how to express ideas or re-reading the entire text to condense the content. The platform can assist in generating clear and coherent passages, thereby improving the overall quality of written texts.

In terms of significance, the project offers high practical value and meets the real needs of a wide range of users, including students, office workers, and anyone who frequently works with Vietnamese texts. Furthermore, the project contributes to promoting the application of natural language processing (NLP) technologies in real-world contexts in Vietnam, paving the way for future research and the development of intelligent applications specifically designed for the Vietnamese language.

1.3 Implementation Steps

The development of the project *"Website Supporting Vietnamese Text Editing"* was carried out through the following main steps:

- **Requirement Analysis and Feature Definition:**Conduct a survey to understand users' practical needs and define essential features, including sentence suggestion, text summarization, speech-to-text conversion, and an integrated virtual assistant (chatbox)
- **Technology Research and Selection:**Explore, compare, and select suitable technologies for system development, including programming languages, frontend/backend frameworks, natural language processing (NLP) libraries, and data storage solutions.
- **Data Collection and Preprocessing:**Gather Vietnamese text datasets from various sources, then clean and preprocess the data to serve as input for training language models.
- **Fine-tuning Language Models:**Select pre-trained Vietnamese language models such as PhoBERT, BARTpho, or GPT-Vi, and fine-tune them on the prepared datasets to optimize their performance for tasks like summarization, sentence suggestion, and natural Vietnamese language understanding.
- **System and UI Design:**Design the overall system architecture and develop a user-friendly interface. Define the interactions between system components such as frontend, backend, and AI models.

- **Feature Development and Integration:** Implement system functionalities and connect the user interface with the backend and language processing APIs. Integrate the fine-tuned AI models into the processing pipeline.
- **Testing and Evaluation:** Test system features under real-world conditions, assess accuracy, performance, and user experience. Record any errors and make improvements as needed.
- **Website Deployment and Report Completion:**Deploy the system to a live environment for user access and feedback. Finalize the graduation thesis report and prepare presentation materials for the defense session.

1.4 Expected Outcomes

1.4.1 Theoretical Contributions

The project also offers practical experience and technical understanding in developing a full-stack web application, particularly in the interaction between the client, server, and database layers, as follows:

- Backend: Gained knowledge of how web applications communicate via APIs, specifically RESTful APIs *[1]*, and successfully built a set of APIs using the FastAPI framework.
- Frontend: Learned and applied the development of a cross-platform user interface using Vue.js *[2],* and implemented effective communication between the frontend and backend through API calls.
- Database: Gained experience in designing and building a relational database using MySQL *[3]* to manage and store application data efficiently.
- System Deployment: Applied Docker *[4]* technology for packaging and deploying the application. The backend and database were deployed on a Virtual Private Server (VPS) *[5]* to ensure stable system performance and scalability.

1.4.2 Application Contributions

The project focuses on researching and developing core functionalities that support Vietnamese text composition and improve writing quality. The system aims to deliver a smooth and accurate user experience through intelligent tools. It deeply integrates Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies to optimize user assistance. Additionally, the application provides an

intuitive and user-friendly interface, fast processing speed, and system stability to minimize errors during usage.

- Administrator Functions
    - User Management:
        - View all users
        - Add new users
        - Update user information
        - Delete users
    - Document Management
        - View all documents
        - Delete documents

- User Functions
    - Account Management
        - Register a new account
        - Log in to the system
    - Text Editing and Composition
        - Create, open, and save documents
        - Use basic text formatting tools (font, font size, color, bold, italic, underline, lists, alignment, etc.)

- Smart Writing Assistance
    - Smart sentence suggestion: Provides context-aware sentence completions or structure recommendations
    - Text summarization: Automatically condenses long paragraphs into concise summaries
    - Speech-to-text conversion: Records user voice and converts it into text in real-time
    - AI-powered virtual assistant (Chatbox): Ask questions, request semantic explanations, find synonyms/antonyms, or get help with phrasing
- Additional Utilities
    - Restore document edit history (version control)
    - Export documents to common formats (e.g., .txt, .docx, .pdf)
    - Convert text to speech (Text-to-Speech)

Chapter 2: THEORETICAL AND TECHNOLOGICAL BASIS

2.1. FastAPI (Python)

FastAPI *[6]* is a modern, high-performance Python web framework designed to build APIs quickly and easily. Built on top of **Starlette** (for ASGI performance) and **Pydantic** (for data validation and type hinting), FastAPI enables developers to create powerful, standards-compliant APIs with minimal effort. It stands out for its exceptional development speed, automatic API documentation, and excellent developer experience.

Key Features and Highlights of FastAPI:

- **Outstanding Performance:** Thanks to Starlette and asynchronous programming (async/await), FastAPI achieves performance comparable to Node.js and Go, making it ideal for high-load, low-latency applications.
- **Automatic API Documentation:** Automatically generates interactive API documentation using OpenAPI (formerly Swagger) and ReDoc directly from your code, saving time and keeping documentation always up-to-date.
- **Strong Data Validation:** With Pydantic and Python's type hints, FastAPI provides automatic, clear, and robust input/output data validation, reducing errors caused by invalid data.
- **Rapid Development:** Reduces code writing time by 50% to 80% compared to traditional frameworks, thanks to intuitive syntax, built-in automation, and rich IDE support.
- **Flexible Dependency Injection (DI):** Offers a simple yet powerful DI system for managing dependencies, sharing logic, and integrating external components with ease.
- **Based on Open Standards:** Fully compatible with OpenAPI and JSON Schema, ensuring wide interoperability and integration capabilities.
- **Asynchronous Programming Support:** Designed with async/await from the ground up, enabling the natural development of high-performance I/O-bound applications.

- **Integrated Security:** Provides easy-to-use tools to implement common security mechanisms such as OAuth2 (including JWT tokens), HTTP Basic Auth, API keys, and more.
- **WebSocket and GraphQL Support:** In addition to RESTful APIs, FastAPI (via Starlette) supports real-time two-way communication using WebSocket *[7]* and can be easily integrated with GraphQL libraries like Strawberry or Ariadne.
- **Seamless Integration with the Python Ecosystem:** Works well with a wide range of Python libraries, from ORMs (like SQLAlchemy, Tortoise ORM) to machine learning and data processing tools.
- **Easy Testing:** Provides utilities that make writing unit and integration tests for APIs simple and effective.
- **Microservices & Containerization Friendly:** With its lightweight footprint, fast startup, and high performance, FastAPI is ideal for building microservices and packaging into Docker containers for deployment on Kubernetes *[8]*.
- **ASGI Compatible:** As an ASGI (Asynchronous Server Gateway Interface) framework, FastAPI can run on high-performance ASGI servers like Uvicorn, Daphne, or Hypercorn.

Figure 2.1 is the logo of the FastAPI framework.

Figure 2.1. Python Framework FastAPI

2.2. Overview of Python

Python is a versatile programming language that is easy to learn and use. With its simple syntax and minimal requirement for prior programming knowledge, Python is an ideal language for beginners. However, it is also widely used in complex projects and various fields such as data science, artificial intelligence, web development, and more.

Notable features of Python include:

- **Simple syntax**: Python uses straightforward and easy-to-understand syntax, allowing programmers to write clean and readable code.
- **Rich library ecosystem**: With over 137,000 libraries and modules, Python provides developers with a vast set of tools to build complex and efficient applications.
- **Versatility**: Python can be used to develop desktop, web, mobile, game, and other types of applications.
- **Object-oriented support**: Python is an object-oriented language, enabling programmers to create classes and objects to enhance code reusability and organization.
- **Interactive interface**: Python offers an interactive shell that allows programmers to test code snippets and interact with Python easily.
- **Cross-platform compatibility**: Python supports multiple platforms such as Windows, Linux, and macOS.
- **Web development support**: Python provides various web development frameworks like Django, Flask, and Pyramid, which help developers build web applications easily and quickly.

Thanks to these outstanding features, Python has become one of the most popular and widely used programming languages in the programming community. It is extensively applied across different fields, from desktop application development to artificial intelligence and data science.

Figure 2.2 provides an overview of the Python programming language.

Figure 2.2. Overview of Python

2.3. Overview of Vue

Vue.js is a progressive and flexible JavaScript framework, created by Evan You and the community, designed for building user interfaces (UI) and Single Page Applications (SPAs). Vue.js stands out for its ease of adoption, high performance, and the ability to integrate easily into existing projects or build complex applications from scratch. It uses JavaScript (increasingly with TypeScript) as its primary language, providing developers with an intuitive and efficient way to build dynamic and high-quality web applications.

Key features of Vue.js include:

- Progressive Framework Architecture: Vue.js is designed to be adopted incrementally. Its core focuses only on the View layer, making it easy to integrate into existing projects or libraries. You can use Vue to control just a part of a page or to build a full-fledged SPA.
- High Performance with Virtual DOM: Vue.js utilizes a Virtual DOM *[9]* to optimize updates to the actual DOM, minimizing costly operations and ensuring smooth and responsive performance.
- Reactivity System: Vue automatically tracks data dependencies and efficiently updates the DOM when the data changes, resulting in clean and manageable code.
- Component-Based Architecture: User interfaces are built from small, independent, and reusable components. This encourages better code organization and enhances maintainability.
- Intuitive HTML Templates: Vue uses HTML-based template syntax, allowing developers to easily bind the DOM to Vue instance data. Powerful and easy-to-learn directives like v-if, v-for, v-bind, and v-on simplify development.
- Hot Module Replacement (HMR): With modern build tools like Vite or Vue CLI, Vue.js supports HMR, enabling developers to see code changes instantly without reloading the entire page—significantly speeding up the development process.
- Rich Ecosystem:
    - Vue Router: Official library for client-side routing.
    - Pinia (or Vuex): Powerful state management solutions.
    - Vue Devtools: Extremely useful browser debugging tools.

    - Vite/Vue CLI: Command-line tools for project scaffolding and management.
- Excellent Documentation & Large Community: Vue.js is known for its clear, detailed, and frequently updated documentation. It also has a vibrant and enthusiastic global developer community.
- Easy Integration: Vue can easily integrate with various backends via APIs (using fetch or axios) and connect to services like Firebase, Supabase, etc.
- Lightweight Core: The Vue.js core library is very lightweight, contributing to faster initial load times.

Noteworthy Considerations (Rather than strict "Disadvantages"):

- High Flexibility: While beneficial, Vue's flexibility can lead to inconsistent project structures across teams without a common convention.
- Job Market (Region-dependent): Although popular, Vue job opportunities may be fewer than React or Angular in some regions—though this is rapidly changing.
- JavaScript/TypeScript Requirement: For developers from other platforms, becoming familiar with JavaScript and its ecosystem may take some time.

Despite that, with its ease of adoption, good performance, excellent documentation, and a strong ecosystem, Vue.js has become one of the leading JavaScript frameworks and is widely loved in the web development community.

Figure 2.3 is a diagram illustrating the lifecycle of a component in Vue.js.

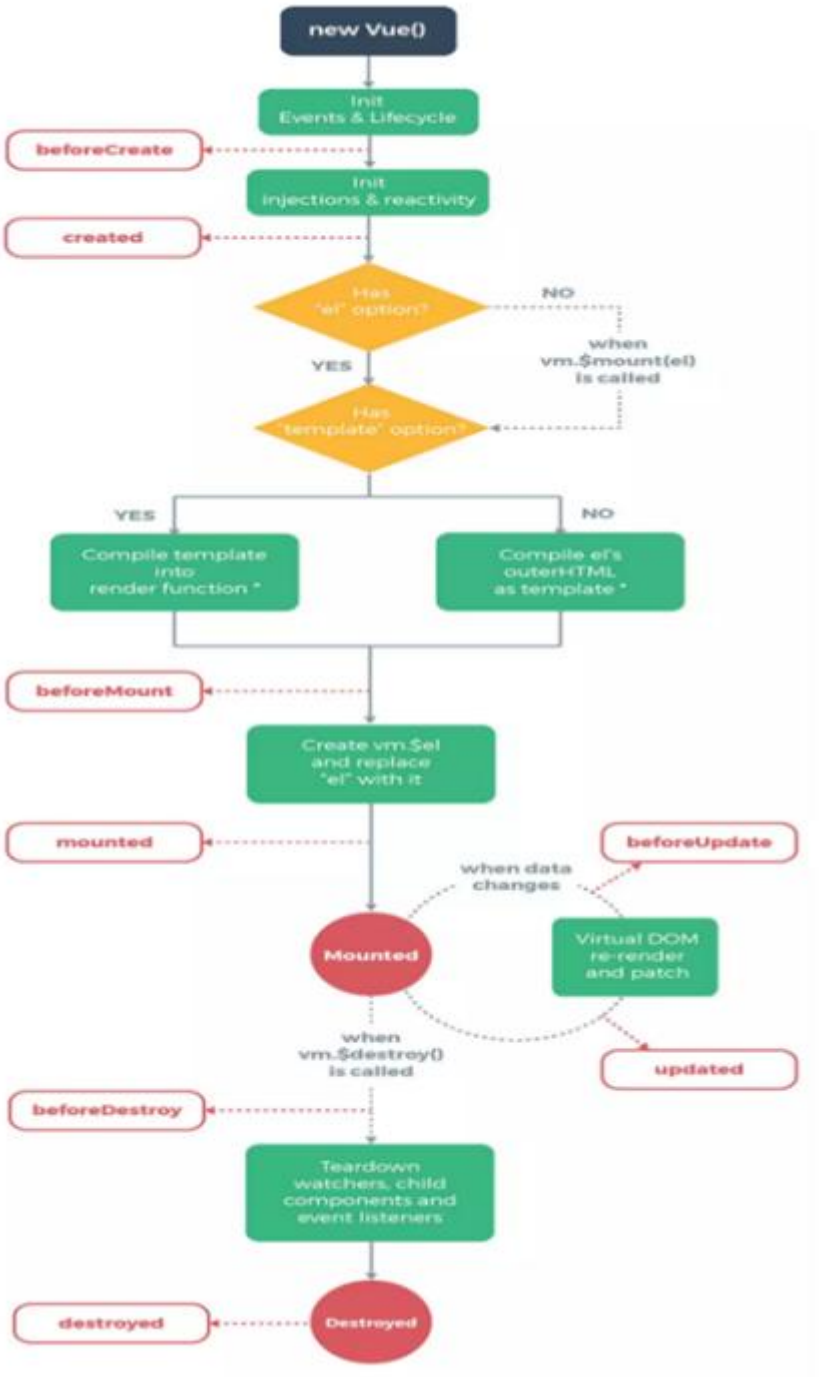

Figure 2.3. Lifecycle Diagram of a Component in Vue.js

State Management with Pinia (Similar to GetX):

Pinia *[10]* is the official state management library recommended for Vue.js applications (especially Vue 3). It is designed to be intuitive, fully compatible with

TypeScript, and extremely lightweight. Pinia provides a simple yet powerful API for managing the global state of an application.

Key features of Pinia include:

- Intuitive and Type-Safe State Management: Pinia offers a clear approach to defining "stores" (state containers). It has excellent TypeScript support, enabling auto-completion and early error detection.
- No Mutations (Compared to Classic Vuex): Unlike traditional Vuex, Pinia simplifies state updates by allowing direct modification of state within actions or through patch, resulting in more concise code.
- Asynchronous Actions: Easily perform asynchronous operations (e.g., API calls) inside actions, making it a good fit for modern applications.
- Getters for Computed Data: Similar to computed properties in components, getters allow you to derive or compute values from the state.
- Great Devtools Integration: Seamlessly works with Vue Devtools, allowing easy tracking of state, actions, and getters.
- Modular Structure and Extensibility: Supports splitting the state into separate modules (stores), which helps manage large applications more efficiently.
- Extremely Lightweight: Pinia has a very small footprint and doesn't significantly increase your app's bundle size.
- Server-Side Rendering *[11]* (SSR) Support: Pinia is fully compatible with SSR environments, making it suitable for both client and server-rendered apps.

With these features and benefits, Pinia is quickly becoming the go-to choice for state management in the Vue.js ecosystem, making application development simpler and more efficient.

Figure 2.4 shows the logo of Vue.js.

Figure 2.4. Overview of Vue.js

2.4 Overview of MySQL (SQL) Database

SQL (Structured Query Language) is a programming language designed for managing and accessing relational databases. A relational database is a method of organizing data in which information is stored in tables, where each table represents a specific type of object or event.

Key Characteristics of SQL Relational Databases:

- Structured Organization: Relational databases have a clear structure, with predefined tables consisting of columns and rows.
- Data Integrity: These databases ensure data integrity by enforcing constraints to maintain accuracy and consistency.
- Interactivity: SQL allows users to interact with the database easily through query commands and operations on tables.
- Flexibility: Users can easily add, remove, or modify tables and columns to adapt to changing requirements.
- High Performance: Relational databases are optimized for handling large datasets and high-traffic environments.

However, SQL relational databases also have certain limitations:

- Cannot Store Unstructured Data: They are limited to storing structured data and are not well-suited for unstructured data such as text documents or multimedia.

- Scalability Challenges: Scaling SQL databases can be difficult as the number of users or data volume increases.
- Less Ideal for Read-Heavy Applications: SQL databases are typically more suitable for write-intensive operations. For read-heavy workloads, NoSQL systems may be more appropriate.

In summary, SQL-based relational databases are widely used for their structured nature, data integrity, interactivity, and flexibility. However, their limitations must be carefully considered depending on the application's specific requirements, especially when dealing with unstructured data, scalability, or read-intensive needs.

Figure 2.5 shows the logo of MySQL Database.

Figure 2.5. MySQL Database

2.5 T5 Model – Text Summarization

T5 (Text-to-Text Transfer Transformer) *[12]* is an advanced machine learning model developed by Google AI, distinguished by its Transformer architecture and a unique approach: treating all Natural Language Processing (NLP) tasks as "text-to-text" problems. Instead of building separate models for each task, T5 is trained to take an input text and produce an output text, regardless of the task. This is achieved by adding specific prefixes to the input text to specify the desired task, such as "summarize:" for summarization.

In text summarization, T5 has demonstrated exceptional effectiveness by generating high-quality abstractive summaries—not only extracting key sentences but also rephrasing and rewriting the content concisely and coherently.

Key Features of T5 in Text Summarization:

- Text-to-Text Framework:
  - *Benefit for summarization:* Simplifies the workflow. Just provide the original text with the prefix "summarize:", and the model will understand and generate the summary. This makes integration and usage straightforward.
- Powerful Transformer Architecture:
  - Uses the full encoder-decoder Transformer *[13]* architecture, enabling deep contextual understanding of the input text and producing coherent and cohesive output.
- Pre-trained on Massive Dataset (Colossal Clean Crawled Corpus *[14]* - C4):
  - Pre-training on a massive and diverse dataset gives T5 a broad foundational knowledge of language, grammar, and common informational structures—essential for generating accurate and natural summaries.
- Strong Abstractive Summarization Capability:
  - T5 doesn't just copy from the original text. It can comprehend key points, reinterpret them, and produce entirely new sentences that concisely convey the core information, similar to human summarization.
- Flexibility and Fine-Tuning Support:
  - Although powerful out of the box, T5 can be fine-tuned on specific summarization datasets (e.g., news articles, scientific documents, legal texts) to improve performance in specialized domains or writing styles.
- Multiple Model Sizes
  - T5 is available in various sizes (e.g., T5-Small, T5-Base, T5-Large, T5-3B, T5-11B), allowing users to choose a model that fits their summarization quality needs and available computational resources.
- Community and Ecosystem Support:
  - Pre-trained and fine-tuned T5 models for summarization are widely available via libraries like Hugging Face Transformers, making deployment and experimentation easier.

How T5 Performs Text Summarization:

1. Prepare Input: The text to be summarized is prefixed with a task identifier. For summarization,it'susually"summarize:".
   Example: summarize: [Long document text to summarize...]
2. Processed by the Model: This prefixed text is fed into the T5 model.
3. Generate Output: T5 generates a new text segment which is the summary of the input.

Advantages of Using T5 for Summarization:

- High-Quality Summaries: Produces summaries that are accurate, capture the main ideas, and retain important information.
- Coherent and Natural Flow: The generated summaries are typically fluent, easy to read, and have a natural writing style.
- Deep Contextual Understanding: Can grasp complex relationships and nuanced meanings in the text.
- Time and Effort Saving: Automates the summarization process, allowing users to quickly grasp the essence of long documents.

Considerations When Using T5:

- Computational Resource Requirements: Larger T5 models require significant hardware (GPU/TPU) for training, fine-tuning, and even inference at acceptable speeds.
- Hallucination Phenomenon: Like other large language models, T5 may sometimes generate plausible-sounding but inaccurate or hallucinated information not found in the original text.
- Input/Output Length Limitations: Transformer models have token length limits. For very long texts, additional processing techniques (e.g., chunking, summarizing sections) may be necessary.
- Need for High-Quality Fine-Tuning Data (if needed): To achieve optimal performance for highly specific domains, fine-tuning with domain-specific high-quality datasets is important.

The T5 model, with its innovative text-to-text approach and the power of Transformer architecture, has established itself as a highly powerful and effective tool for text summarization. Its ability to produce high-quality, abstractive summaries makes it a

top choice for applications requiring automated and intelligent distillation of large volumes of textual information.

Figure 2.6 illustrates the text summarization task of the T5 model.

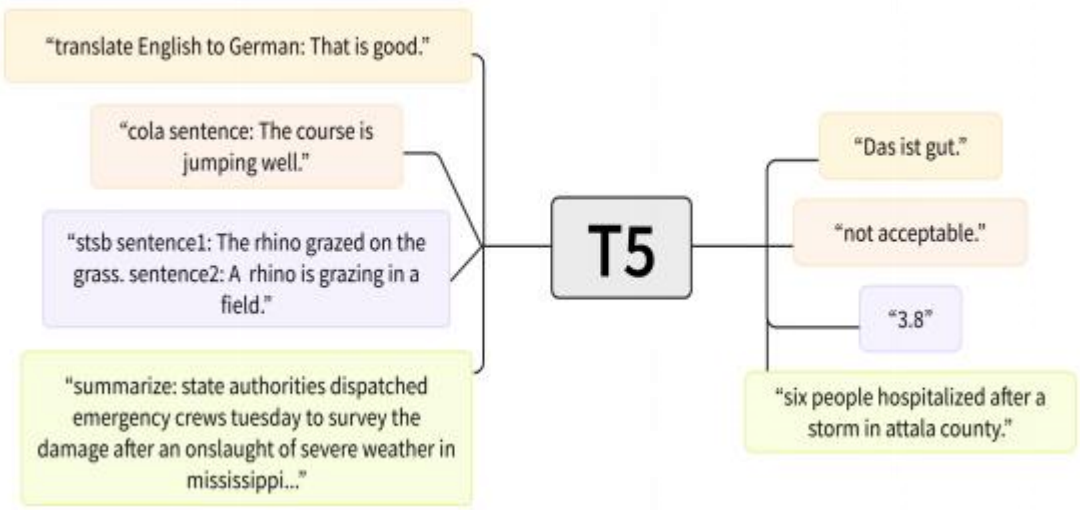

Figure 2.6. Overview of the T5 Model [15]

2.6. GPT-2 Model – Sentence Suggestion

2.6.1 Introduction

This chapter presents the core theoretical foundations used to build and deploy the text suggestion model. Key concepts include the Transformer architecture, the GPT-2 *[16]* language model, and the transfer learning approach through fine-tuning techniques. A solid understanding of these fundamentals is essential to comprehend the operating mechanism, advantages, and implementation methods of the proposed solution in this report.

2.6.2. Transformer Architecture and the Self-Attention Mechanism

Before the advent of Transformers, sequential models such as Recurrent Neural Networks (RNNs) *[17]* and Long Short-Term Memory (LSTM) *[18]* were the dominant approaches for sequence processing. However, these architectures struggled with capturing long-range dependencies in text and were difficult to parallelize during computation.

In 2017, Vaswani et al. *[19]* introduced the Transformer architecture in the groundbreaking paper *"Attention Is All You Need"*, which revolutionized the field of Natural Language Processing (NLP).

The key innovation of the Transformer is the complete removal of recurrence, replaced by the Self-Attention mechanism. This mechanism allows each word in a sentence to compute an "attention" score with every other word in the same sentence. As a result, the model can identify semantic and syntactic relationships between words regardless of their distance. Specifically:

- **Input**: Each word is represented by a vector.
- **Mechanism**: From each input vector, the model generates three vectors: Query, Key, and Value.
- **Computation**: The attention score between one word and others is calculated by taking the dot product of the Query vector of that word with the Key vectors of the other words. These scores are then normalized using the Softmax function to create a weight distribution, which is multiplied with the corresponding Value vectors to generate the output vector.
- **Result**: The output vector of each word is a weighted sum of all the words in the sequence, where the weights reflect their level of relevance.

Thanks to its parallel processing capability and effective context understanding, the Transformer has become the foundation of most modern large language models.

2.6.3. GPT-2 Language Model (Generative Pre-trained Transformer 2)

GPT-2, developed by OpenAI, is a large language model based on the Transformer architecture. Its name reflects its key characteristics:

- **Generative**: The main goal of the model is to generate coherent and natural-sounding text.
- **Pre-trained**: The model is pre-trained on a massive text corpus (40GB of Internet text), allowing it to learn deep knowledge about language, grammar, and common facts.
- **Transformer**: Its underlying architecture is based on the Transformer model.

A notable feature of GPT-2 is that it uses only the decoder part of the original Transformer architecture. This "decoder-only" design is well-suited for text generation

tasks because it operates in an auto-regressive manner — predicting the next word based on all previously generated words.

To achieve this, GPT-2 employs a variant of the self-attention mechanism called Masked Self-Attention. During computation, a "mask" is applied to prevent the model from "seeing" future tokens in the sequence, ensuring that the prediction at position t only depends on tokens from positions 1 to $t-1$.

Figure 2.7 describes the decoder-only architecture of the GPT-2 model, which is suitable for the sentence generation task.

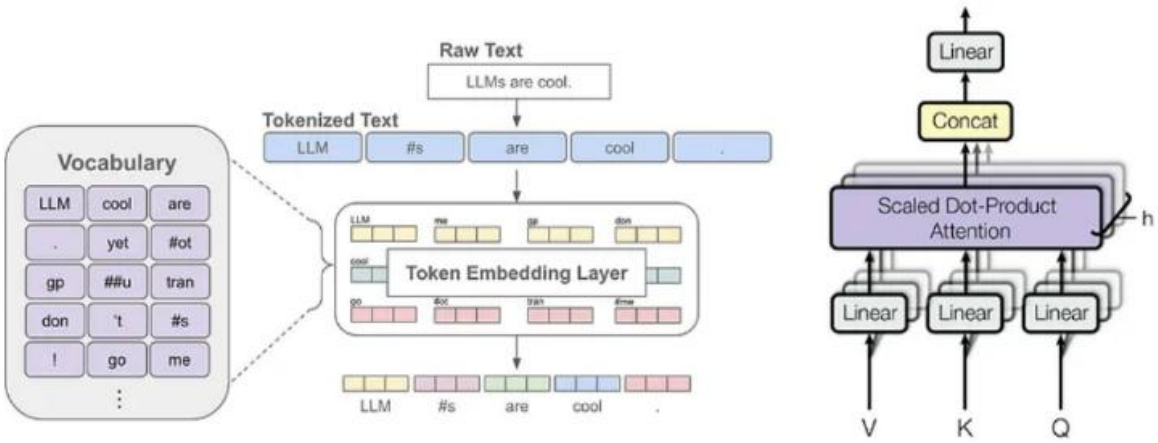

Figure 2.7. Decoder-Only Model Architecture [20].

2.6.4. Transfer Learning and Fine-Tuning

a. Transfer Learning

Transfer learning is a method in machine learning where a model developed for one task is reused as the starting point for a model on a second task. Instead of training a model from scratch—which requires a large amount of data and computational resources—we leverage the "knowledge" learned from a source task *[21]*.

In Natural Language Processing (NLP), this process typically involves two stages:

- **Pre-training:** A language model is trained on a massive unlabeled dataset (e.g., the entire Wikipedia, books, and websites). The goal of this phase is for the model to learn general language representations, including grammar, semantics, and background knowledge. GPT-2 is a product of this stage.

- **Fine-tuning:** The pre-trained model is then further trained on a smaller, domain-specific dataset tailored to a particular task (e.g., sentiment classification, question answering, or in this report's case, text suggestion for a specific domain).

b. Fine-Tuning Process for the Text Suggestion Task

Fine-tuning is the process of "specializing" a general language model. In this project, the GPT-2 fine-tuning procedure is conducted as follows:

- **Initialization:** Load the weights of the pre-trained GPT-2 model. These weights contain general knowledge of language learned during pre-training.
- **Data preparation:** Construct a specialized text dataset (e.g., financial reports, legal documents, work emails). This data reflects the writing style, terminology, and structure that the model is expected to learn.
- **Continued training:** Retrain the model on this domain-specific dataset with a low learning rate. This process does not change the model architecture but slightly updates the weights to adapt to the new data.
- **Result:** After fine-tuning, the GPT-2 model retains its ability to generate coherent text but now favors producing sentences and structures aligned with the specialized domain it has learned. This is the core mechanism that enables the generation of high-quality, context-appropriate text suggestions for the user *[22]*.

2.7 Model – Speech-to-Text Conversion

I will build a model similar to Whisper. Whisper by OpenAI is a representative Automatic Speech Recognition (ASR) model that utilizes the full Encoder-Decoder architecture of the Transformer. Whisper is designed to solve the problem of converting audio signals into corresponding text sequences *[23]*.

2.7.1. Kiến trúc Encoder-Decoder của Whisper

The Whisper model consists of two main components that work sequentially to process audio signals:

a.Encoder:

The Encoder's task is to take in raw audio signals and transform them into a sequence of hidden feature representations. This process includes:

- **Audio Preprocessing:** The input audio signal (e.g., .wav or .mp3 file) is divided into small 30-second segments. It is then converted into a log-Mel spectrogram—a 2D representation that reflects the intensity of different frequencies over time. This is the standard input format for modern ASR models.
- **Encoding:** The log-Mel spectrogram is passed through a series of Transformer encoder layers. Similar to the original Transformer architecture, Whisper's Encoder uses self-attention blocks to identify relationships across different parts of the audio segment, allowing it to learn important phonetic and contextual features. The Encoder's output is a sequence of feature vectors containing the encoded information from the audio signal.

b.Decoder:

The Decoder's role is to take the encoded feature representations from the Encoder and generate the corresponding text transcript.

- **Cross-Attention Mechanism:** This is a key difference from decoder-only architectures like GPT-2. In addition to using masked self-attention to process the previously generated text tokens, Whisper's Decoder includes a cross-attention layer. This layer allows the Decoder to "attend" to the Encoder's outputs. At each step of text generation, the Decoder decides which parts of the audio signal are most relevant for predicting the next text token.
- **Autoregressive Generation:** Similar to GPT-2, the Decoder operates in an autoregressive manner. It starts with a special token (e.g., <|startoftranscript|>) and sequentially predicts the next token based on both the audio representations from the Encoder and the previously generated text tokens. This process continues until the model generates the end-of-sequence token (e.g., <|endoftext|>).

Figure 2.8 provides a complete illustration of the encoder-decoder architecture in the Transformer model.

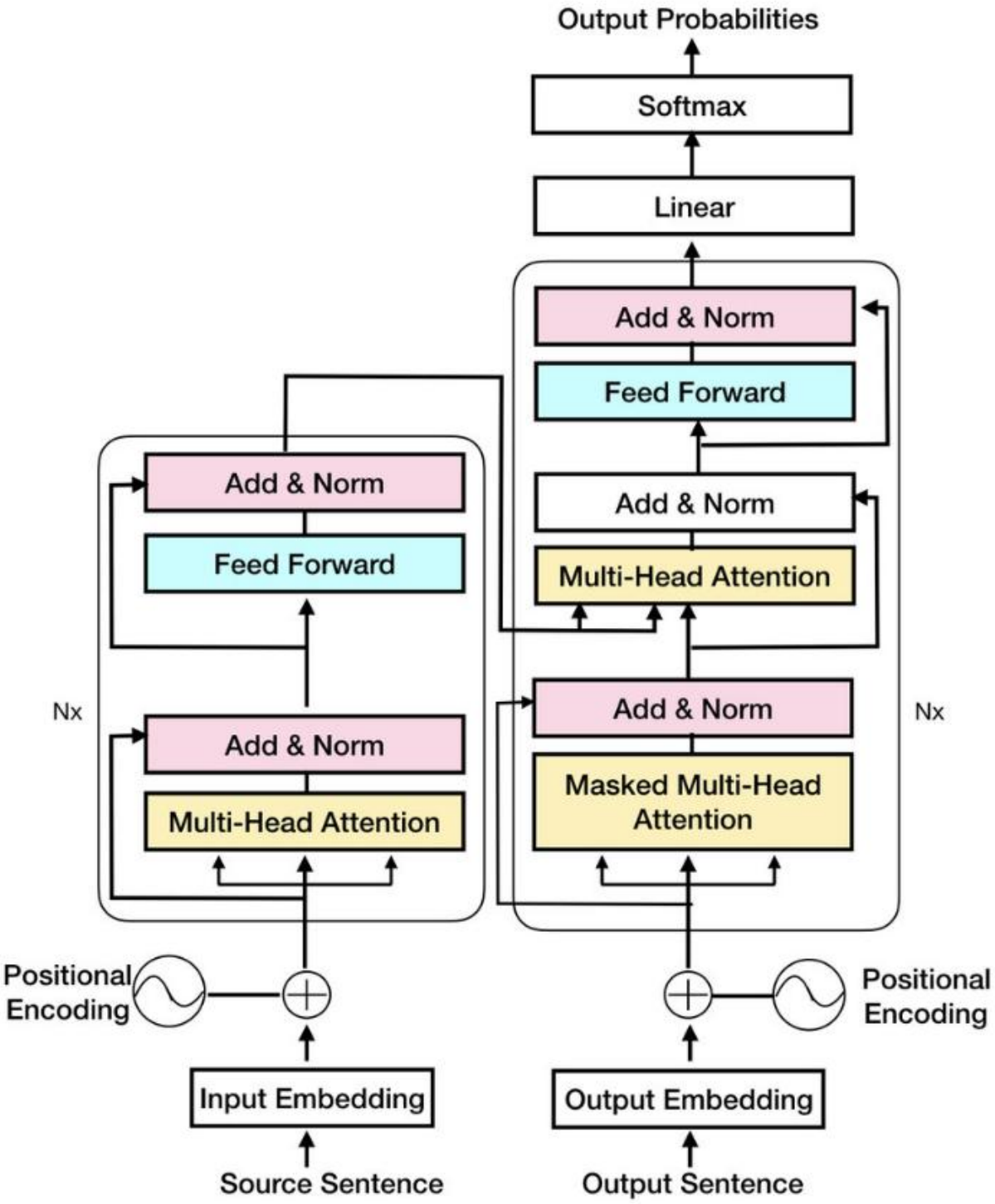

Figure 2.8. The Encoder-Decoder Transformer Architecture

2.7.2. Key Features of Whisper

Whisper brings a breakthrough to the field of ASR thanks to the following characteristics:

- **Trained on massive and diverse data:** Unlike earlier models that were trained on carefully labeled, clean datasets, Whisper is trained on 680,000 hours of audio collected from the internet. This data includes multiple languages, accents, background noises, and topics. Such diversity gives Whisper exceptional robustness, allowing it to perform well in real-world conditions rather than ideal studio settings.
- **Multilingual and multitask support:** Thanks to its diverse training data, Whisper is not only a transcription model but also supports multiple tasks within a single model:
    - **Language Identification:** Detects the spoken language in an audio clip.
    - **Multilingual Transcription:** Converts spoken words from nearly 100 different languages into text.
    - **Speech Translation:** Performs direct speech-to-English translation from any source language.
- **Open-source model:** OpenAI has released the source code and pretrained weights of Whisper, allowing researchers and developers to freely use, deploy, and customize the model. This openness fosters innovation and supports applications requiring high data privacy, as all processing can be done locally without transmitting data to third parties *[24]*.

Chapter 3: SYSTEM ANALYSIS AND DESIGN

3.1. Use Case Diagram

Figure 3.1 is the use case diagram of the entire text editor system.

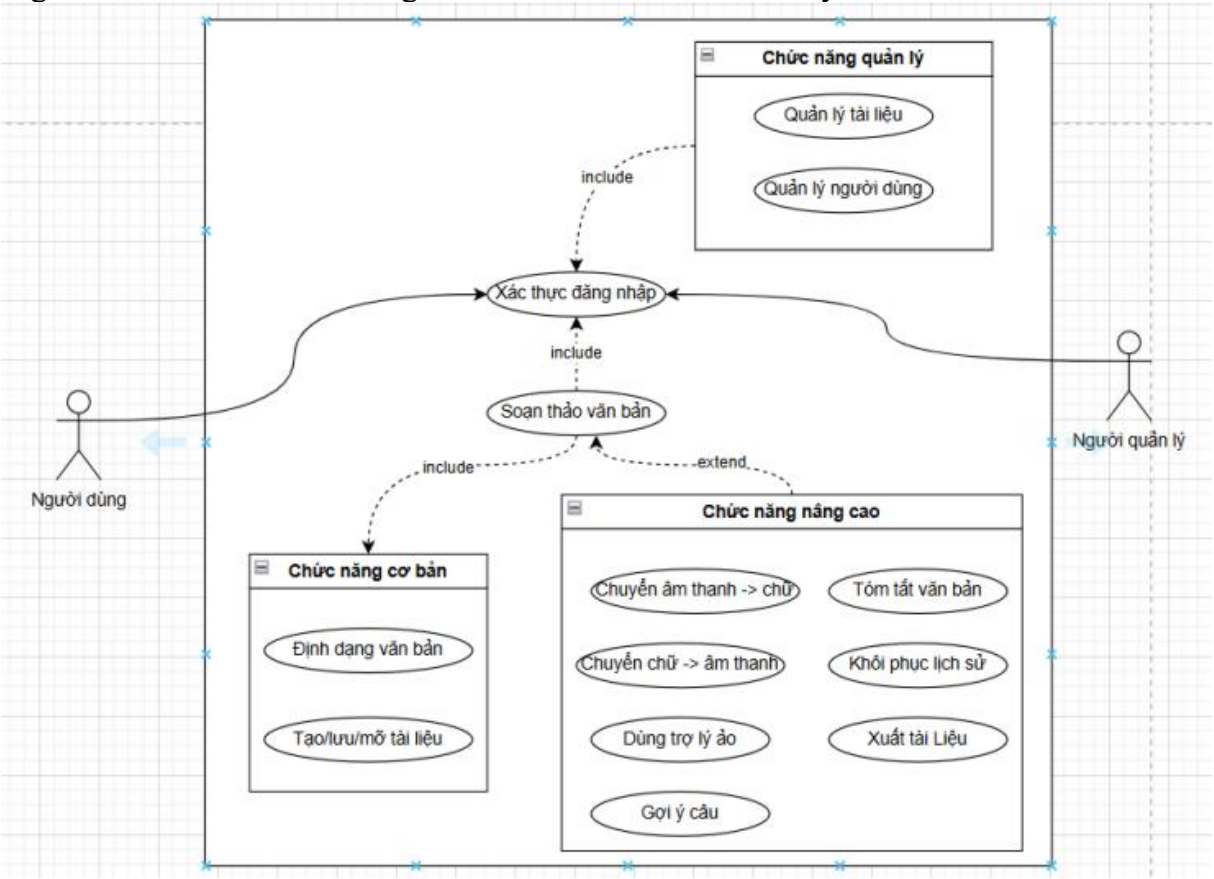

Figure 3.1. General Use Case Diagram of the System

3.2. Use Case Diagram Specification

3.2.1. Login Function

Table 3.1 provides a detailed description of how the login function operates within the system.

Table 3.1. Login Function Specification

Name Use - Case	**Login**
Actor	User, Administrator

Description	Allows users to log into the application and assigns roles based on user types.
Trigger Condition	The user accesses the login screen and clicks the “Sign In” button.
Steps to Perform	1. The user enters login information: Username and Password. 2. Clicks the “Sign In” button. 3. The system verifies the credentials. 4. Navigates to the homepage screen.
Result	Navigates to the homepage screen.
Error Cases	1. The user enters incorrect login information. 2. The device is not connected to the internet.

3.2.2. Registration Function

Table 3.2 provides a detailed description of how the Register function operates within the system.

Table 3.2. Registration Function Specification

Use-Case Name	**Register an account**
Actor	User, Administrator
Description	Allows users to register for using the application.
Trigger Condition	The user has accessed the registration screen.

Steps to Perform	1. Enter the required information. 2. Click the “Sign Up” button. 3. The system validates the account and processes the registration.
Result	Displays the message “Sign Up Successful, please Login to continue” and returns to the login screen.
Error Cases	1. Invalid input format in one or more fields. 2. Username already exists. 3. Device is not connected to the internet.

3.2.3. Function to Create a New Document

Table 3.3 provides a detailed description of how the create new document function operates within the system.

Table 3.3. Function Specification: Create New Document

Use-Case Name	**Create New Document**
Actor	User, Administrator
Description	Allows users to create a blank document for editing.
Trigger Condition	The user has logged in and selected the "Create New" function.
Steps to Perform	1. Click the "Create New" button. 2.The system opens a blank document interface.

Result	A new document is displayed, ready for editing.
Error Case	The device is not connected to the internet.

3.2.4. Text-to-Speech Function

Table 3.4 provides a detailed description of how the read text function operates within the system.

Table 3.4. Function Specification: Read Text

Use-Case Name	***Read Text***
Actor	*User, Administrator*
Description	*Allows users to listen to the text they have composed.*
Trigger Condition	*The user has composed a document.*
Steps to Perform	*1. Click the "Read Text" button.* *2. The system reads aloud the composed text.*
Result	*The system reads the composed text aloud.*
Error Cases	*The device is not connected to the internet.* *No composed text is available.*

3.2.5. Save Document Function

Table 3.5 provides a detailed description of how the save document function operates within the system.

Table 3.5. Function Specification: Save Document

Use-Case Name	**Save Document**
Actor	User, Administrator
Description	Allows users to save the contents of a document.
Trigger Condition	The user has edited the content and presses Ctrl + S.
Steps to Perform	1. Press Ctrl + S. 2. The system stores the data.
Result	The system saves the document
Error Case	The device is not connected to the internet.

3.2.6. Summarize Text Function

Table 3.6 provides a detailed description of how the summarize text function operates within the system.

Table 3.6. Function Specification: Summarize Text

Use-Case Name	**Summarize Text**

Actor	User, Administrator
Description	Automatically extracts the main content from a long text.
Trigger Condition	The document has content and the user selects “Summarize”.
Steps to Perform	1. Click the “Summarize” button. 2. The system processes and displays the result.
Result	The summarized content is displayed.
Error Case	Slow internet connection or server error.

3.2.7. Ask Questions via Chatbox Function

Table 3.7 provides a detailed description of how the ask questions via chatbox function operates within the system.

Table 3.7. Function Specification: Ask Questions via Chatbox

Use-Case Name	**Ask Questions via Chatbox**
Actor	User, Administrator
Description	The chatbox will respond to questions related to the composed text.
Trigger Condition	The user opens the chatbox interface and enters a question.

Steps to Perform	1. Enter the question. 2. Click “Send”. 3. The system responds.
Result	The answer is displayed in the chat window.
Error Case	The chatbot does not respond due to network disconnection or server error.

3.2.8. User Management Function

Table 3.8 provides a detailed description of how the user management function operates within the system.

Table 3.8. Function Specification: User Management

Use-Case Name	**User Management**
Actor	User, Administrator
Description	Allows the administrator to view all users and perform add, edit, or delete operations.
Trigger Condition	User logs in with admin role.
Steps to Perform	1. Log in. 2. Perform tasks to view, add, edit, or delete users.

Result	Users are viewed, added, edited, or deleted successfully.
Error Case	The device is not connected to the internet.

3.2.9. Document Management Function

Table 3.9 provides a detailed description of how the document management function operates within the system.

Table 3.9. Function Specification: Document Management

Use-Case Name	**Document Management**
Actor	Administrator
Description	Allows the administrator to view all documents and delete them if necessary.
Trigger Condition	User logs in with admin role.
Steps to Perform	1. Log in. 2. Perform tasks to view or delete documents.
Result	Documents are viewed or deleted successfully.

Error Case	The device is not connected to the internet.

3.3. Use Case Diagram Specification

3.3.1 Sequence Diagram for Login Function

Figure 3.2 shows the login sequence diagram, illustrating the interaction between the user, application, and server during the login process.

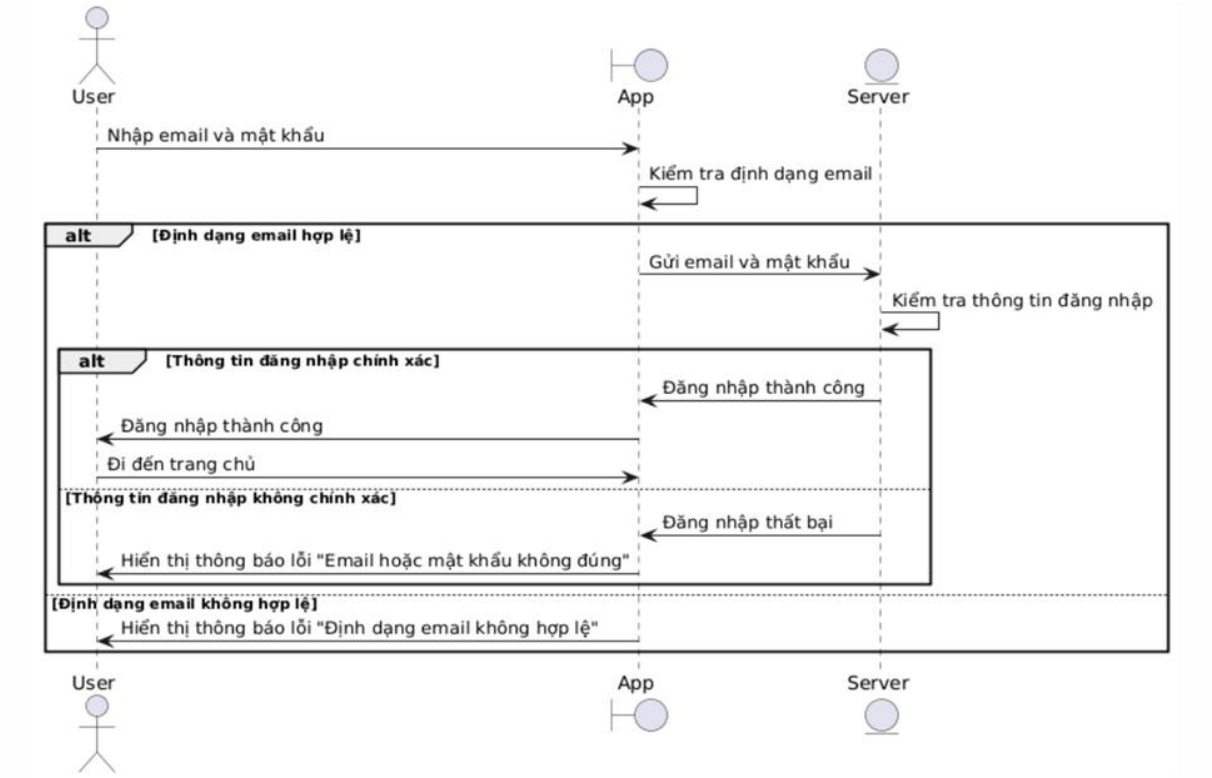

Figure 3.2. Sequence Diagram for Login Function

3.3.2 Sequence Diagram for Registration Function

Figure 3.3 shows the register sequence diagram, illustrating the interaction between the user, application, and server during the register process.

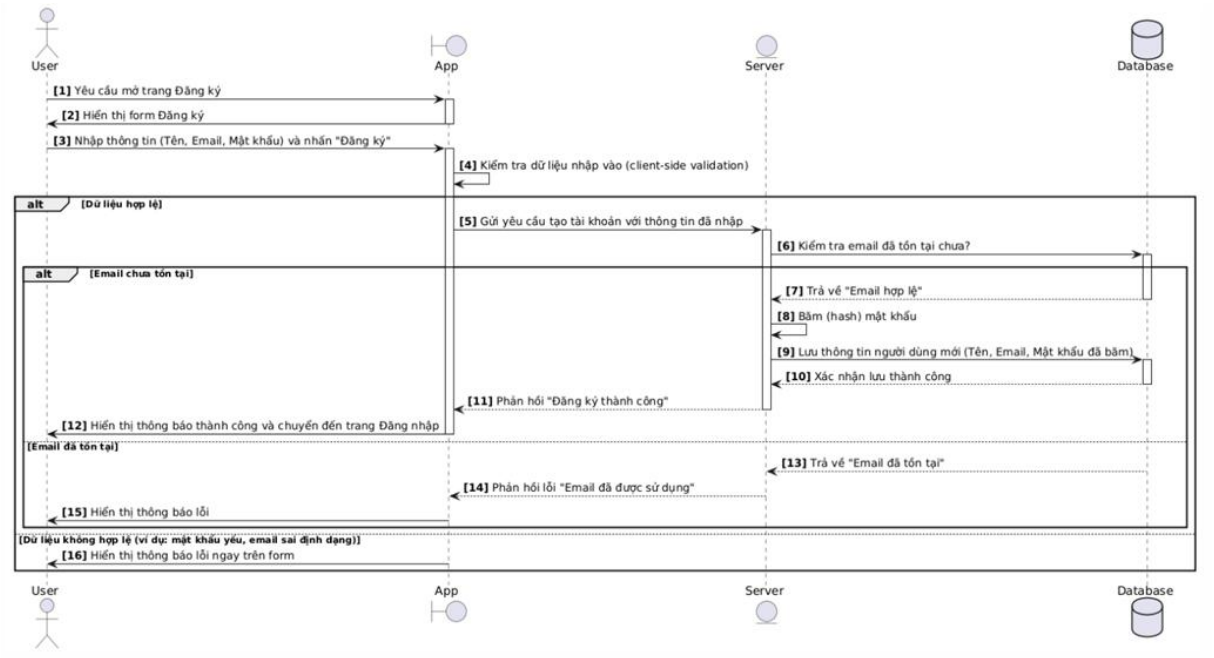

Figure 3.3. Sequence Diagram for Registration Function

3.3.3 Sequence Diagram for Text-to-Speech Function

Figure 3.4 shows the text to speech diagram, illustrating the interaction between the user, application, and server during the text to speech process.

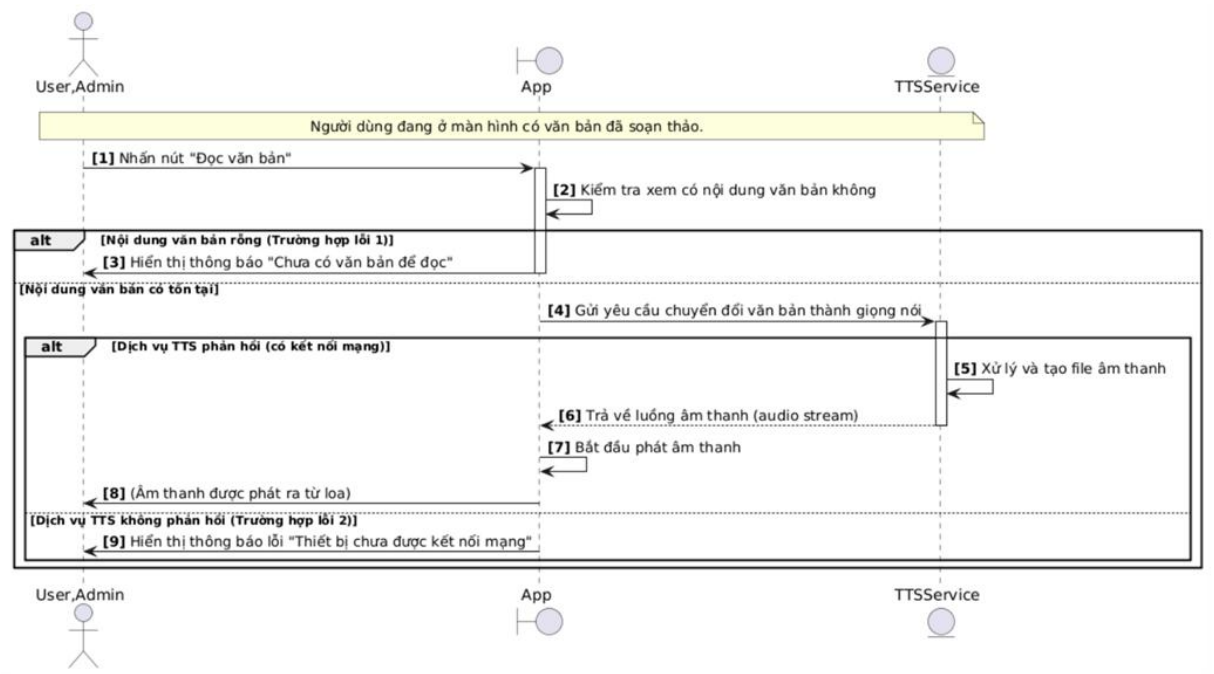

Figure 3.4. Sequence Diagram for Text-to-Speech Function

3.3.4 Sequence Diagram for Save Document Function

Figure 3.5 shows the save document diagram, illustrating the interaction between the user, application, and server during the save document process.

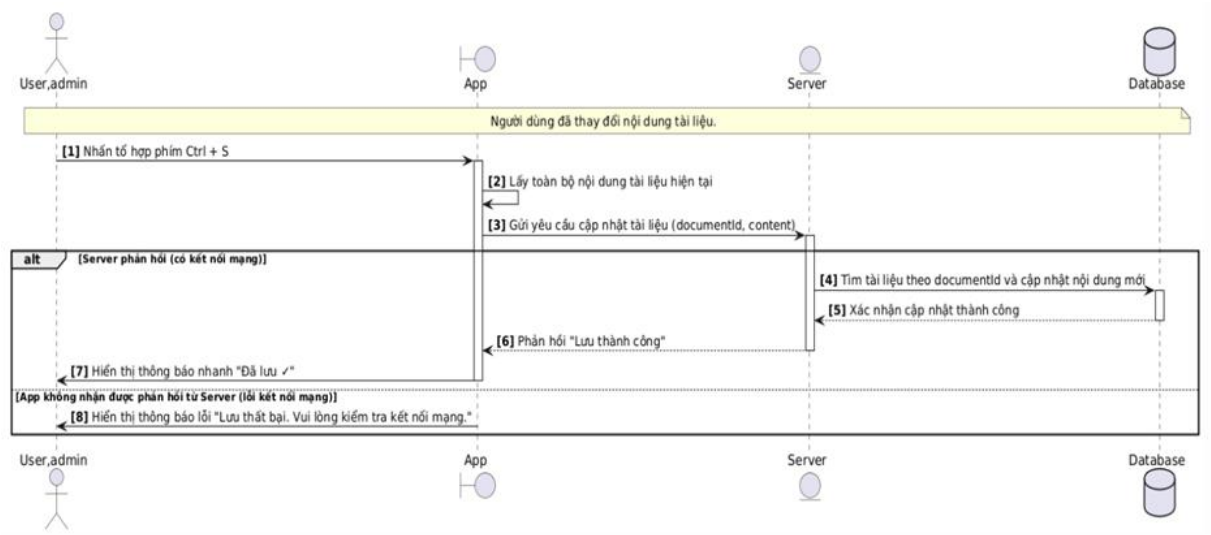

Figure 3.5. Sequence Diagram for Save Document Function

3.3.5 Sequence Diagram for Text Summarization Function

Figure 3.6 shows the text summarization diagram, illustrating the interaction between the user, application, and server during the text summarization process.

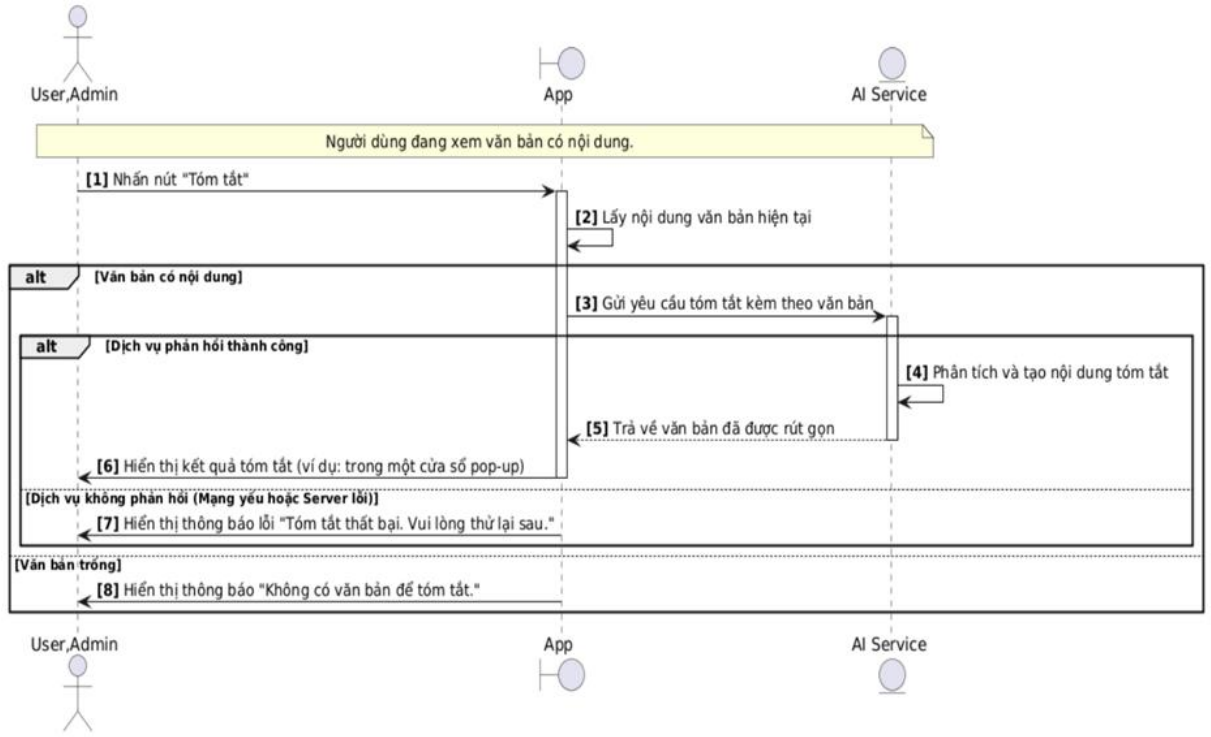

Figure 3.6. Sequence Diagram for Text Summarization Function

3.3.6 Sequence Diagram for Virtual Assistant Function

Figure 3.7 shows the Virtual assistant sequence diagram, illustrating the interaction between the user, application, and server during the virtual assistant process.

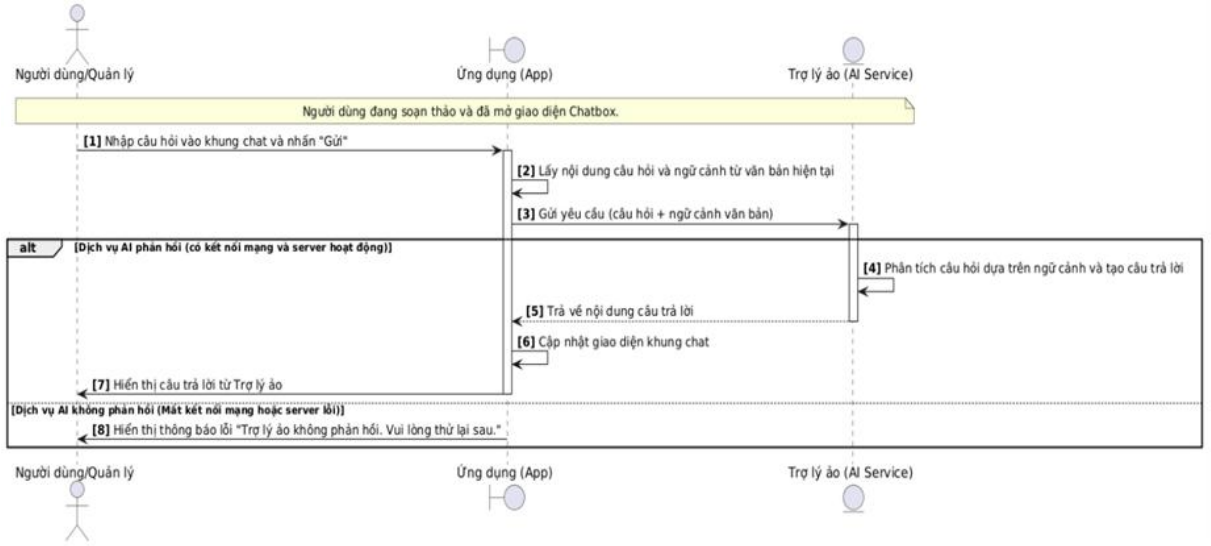

Figure 3.7. Sequence Diagram for Virtual Assistant Function

3.3.7 Sequence Diagram for User Management Function

Figure 3.8 shows the user management sequence diagram, illustrating the interaction between the user, application, and server during the user management process.

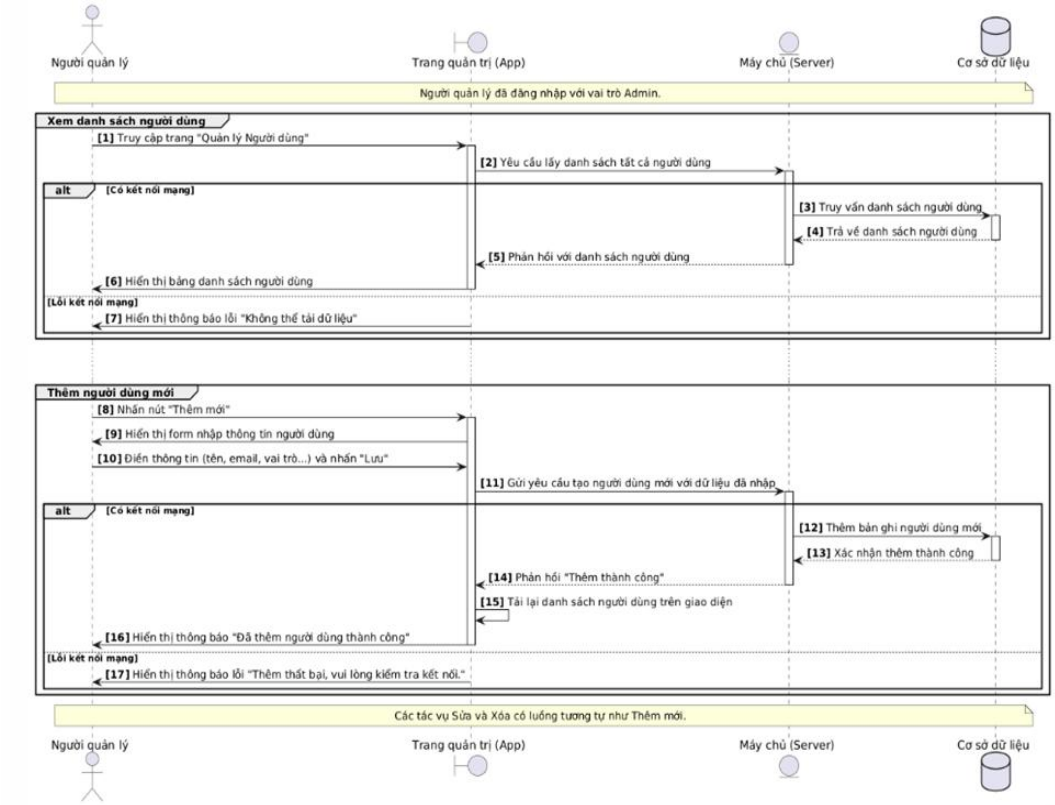

Figure 3.8. Sequence Diagram for User Management Function

3.3.8 Sequence Diagram for Document Management Function

Figure 3.9 shows the document management sequence diagram, illustrating the interaction between the user, application, and server during the document management process.

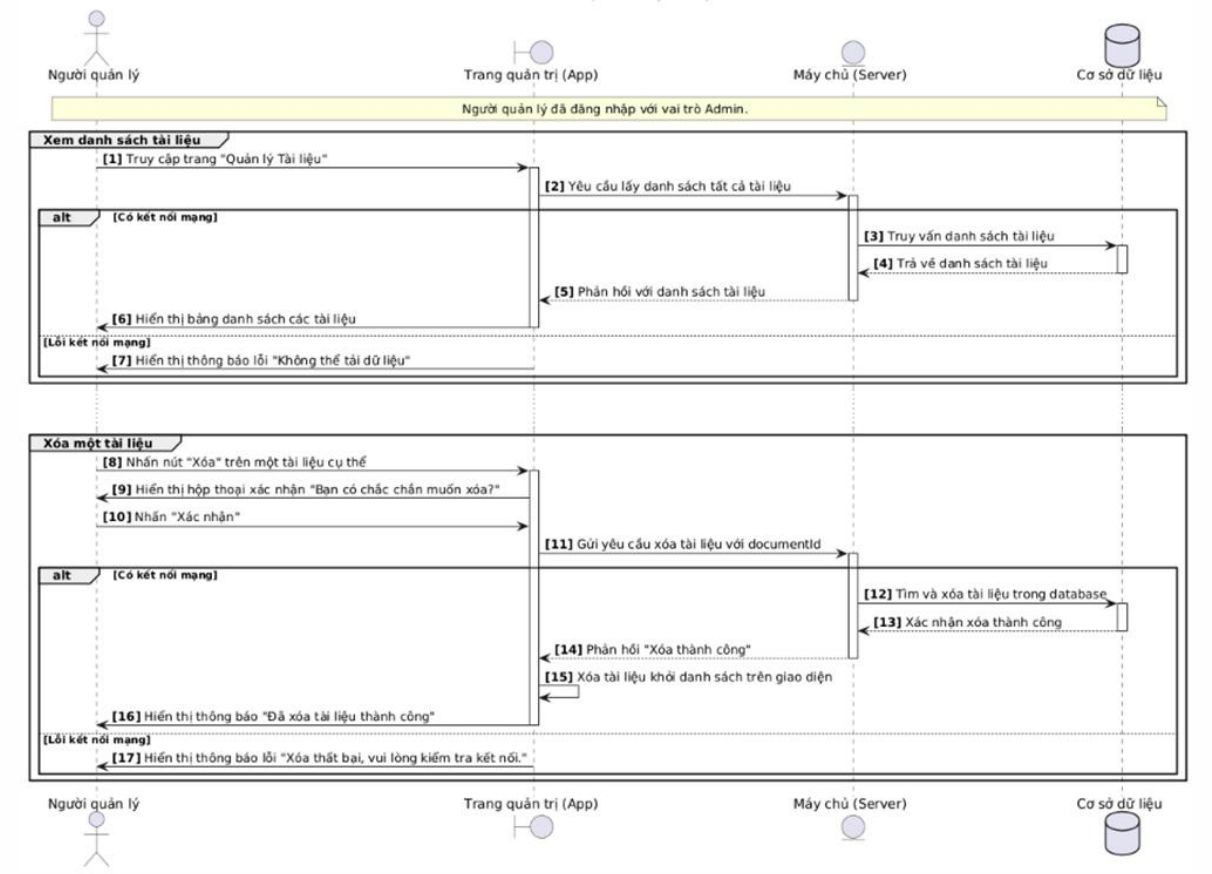

Figure 3.9. Sequence Diagram for Document Management Function

3.4. System Architecture

Figure 3.10 describes in detail the system architecture and the interaction between the user and the system components

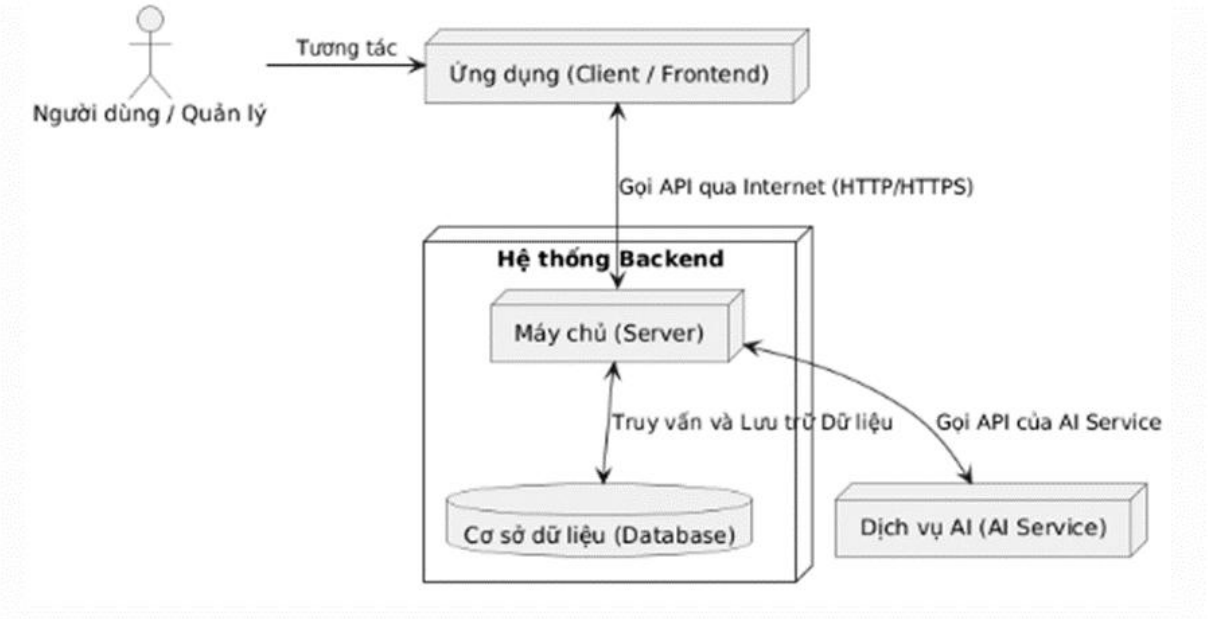

Figure 3.10. System Architecture

3.5. Database

3.5.1 Overview

The system's database is built on SQL – a relational database management system that ensures efficient data management for users, documents, and editing history. The core entities of the system include:

- **User**: Stores user information, including:
  - Basic info: id, name, email, password_hash
  - role, status, and created_at timestamp
  - **Relationships**:
    - A user can create multiple documents
    - A user can perform multiple edits (document history)
- **Document**: Stores information about the documents created or edited by users:
  - Attributes: id, title, file_path, created_by, created_at, updated_at
  - Each document is linked to its creator and can have multiple edit history records.
- **DocumentHistory**: Logs the history of document edits:
  - Includes: id, document_id, editor_id, file_path_snapshot, edited_at
  - Allows version tracking, identifying editors and timestamps.

These relationships are designed to be scalable, supporting advanced features in the future such as:

- Role-based access control (admin/user)
- Restoring previous versions
- Real-time edit tracking
- Integration with online collaboration tools

The database is optimized for integrity, security, and scalability to support the operation of an intelligent online document editing platform.

3.5.2 Table "users"

Table 3.10 describes the structure of the user table in the database.

Table 3.10. Users

Users		
Column name	Data Type	Description
Id	Int	Primary key
Email	String	Unique email adress
Name	String	User's full name
Password_hash	Text	Encrypted password
Role	String	User role(e.g, admin/user)
Created_at	Timestamp	Account creation time
Status	String	Account Status

3.5.3 Table "documents"

Table 3.11 describes the structure of the documents table in the database.

Table 3.11. Documents

documents		
Column name	Data Type	Description

Id	Int	Primary key
file	String	Document title
file_path	String	File path in storage system
created_by	int	User ID who created the document
created_at	timestamp	Document creation time
updated_at	timestamp	Last update time

3.5.4 "document_history"

Table 3.12 describes the structure of the history to edit document table in the database.

Table 3.12. Document edit history

document_history		
Column name	Data Type	Description
Id	Int	Primary key
document_id	int	Related document ID
editor_id	int	User ID who made the edit
file_path_snapshot	string	Path to the snapshot of the document at edit time
updated_at	timestamp	Time when edit was made

3.6. Artificial Intelligence Applications

3.6.1 Model Text Generation

3.6.1.1 Overview

This section presents the experimental process of building a text suggestion model by fine-tuning the GPT-2 language model. It covers model selection, data preparation and analysis, environment setup, training parameters, and concludes with evaluation results and performance analysis.

3.6.1.2 Model Selection

To perform text suggestion/completion tasks, the base model selected is GPT-2 (Generative Pre-trained Transformer 2). The rationale includes:

- **Architecture Suitability:** GPT-2 is a decoder-only, autoregressive model tailored for text generation. It aligns perfectly with the next-word prediction task.
- **Pre-trained Knowledge:** GPT-2 is pre-trained on a massive corpus, providing it with strong linguistic knowledge in grammar, semantics, and general facts. This enhances fine-tuning efficiency with a smaller dataset.
- **Accessibility:** GPT-2 is open-source and available in various sizes. The small version (124M parameters) is used for this project to balance performance and computational feasibility on common hardware.

3.6.1.3 Dataset

The effectiveness of fine-tuning depends heavily on the quality and relevance of training data.

- Data Source: Texts from online journals such as Dantri, CafeF, etc.
- Collection and Formatting: Documents are organized into folders by topic to enable the model to learn context and paragraph structure.
- Size: 3615 files

Preprocessing:

- Tokenizer: Default GPT-2 tokenizer is used for compatibility.
- Padding: Since GPT-2 lacks a native pad_token, the end-of-sequence token (eos_token) is used for padding to handle varying input lengths.

Data Statistics:

- Size: ~13.2 MB
- Word Count (estimated): ~850,000
- Token Count (after encoding): ~1,200,000

Training Configuration:

- Optimizer (AdamW): An adaptive optimizer that decouples weight decay from gradient updates to improve generalization.
- Learning Rate Scheduler (linear with warmup): Starts with a warmup phase to stabilize early training, followed by a linear decay for better convergence.
- Training Loop:
    - Forward Pass: Compute predicted output and loss.
    - Backward Pass: Backpropagate loss to calculate gradients.
    - Gradient Clipping: Caps gradients at 1.0 to prevent exploding gradients.
    - Parameter Update: Gradients are used by AdamW to update model weights.

3.6.1.4 Training Environment and Hyperparameters

Environment:

- Libraries: PyTorch, Transformers, Datasets (Hugging Face), Accelerate
- Hardware: Google Colab with NVIDIA T4 GPU, 16GB VRAM

Table 3.13 presents the evaluation training hyper-parameters of model GPT2 for text generation task.

Table 3.13. Training Hyperparameters

Tham số	**Value**
Model_name : GPT2	Base model used for fine-tuning
Output_dir:/kaggle/working/gpt2-finetuned	Directory to store the fine-tuned model
Per_device_train_batch_size = 4	Number of samples processed per update

learning_rate = 5e-5	Learning rate for parameter updates
Max_length = 128	Maximum input sequence length in tokens
Save_Steps = 500	Frequency of saving model checkpoints
Logging_steps=100	Frequency of logging training loss
Num-train-epochs = 30	Number of full passes over the training dataset

The model is trained using Hugging Face's Trainer class, which automates training loops, computes loss, and updates weights using the AdamW optimizer.

3.6.1.5 Experimental Results

a. Qualitative Evaluation

Comparison between GPT-2 base and fine-tuned versions on the same prompts:

Table 3.14 describes the results of the GPT-2 model, which are used to evaluate the model's performance.

Table 3.14. Text Genearation Comparison

Prompt	GPT2-Base Output	Fine-tuned GPT2 Output

"Hợp đồng lao động là"	"Hợp đồng lao động là một khái niệm rất quan trọng. Nó có thể được tìm thấy trong nhiều lĩnh vực khác nhau của cuộc sống, từ kinh doanh đến chính phủ."	"Hợp đồng lao động là sự thoả thuận giữa người lao động và người sử dụng lao động về việc làm có trả công, tiền lương, điều kiện lao động, quyền và nghĩa vụ của mỗi bên."
"Người lao động có quyền"	"Người lao động có quyền được trả lương cho công việc của mình. Đó là một trong những quyền cơ bản nhất của con người."	"Người lao động có quyền: a) làm việc; tự do lựa chọn việc làm, nơi làm việc, nghề nghiệp; b) hưởng lương phù hợp với trình độ, kỹ năng nghề; c) được bảo hộ lao động."

"Thời giờ làm việc không quá"	"Thời giờ làm việc không quá 8 giờ trong một ngày và 40 giờ trong một tuần. Đây là một quy định tốt để đảm bảo sức khỏe."	"Thời giờ làm việc không quá 08 giờ trong 01 ngày và không quá 48 giờ trong 01 tuần. Người sử dụng lao động có quyền quy định thời giờ làm việc theo tuần hoặc ngày."

Observation:

- Base GPT-2: Produces grammatically correct but general sentences.
- Fine-tuned GPT-2: Outputs domain-specific, legally accurate, and contextually appropriate sentences reflecting the training data.

b. Quantitative Evaluation

- **Training Loss:** Decreased steadily from ~3.5 to ~0.21, indicating successful learning and convergence.
- **Perplexity:** Lower perplexity signifies better prediction confidence and model performance.

3.6.1.6 Overall Evaluation and Future Directions

a. General Evaluation:

Fine-tuning GPT-2 proved highly effective for text suggestion in a specialized domain. The fine-tuned model captured legal-specific terminology and writing style, producing coherent and relevant outputs.

b. Limitations:

- Data Dependence: Model quality is constrained by the quality and coverage of the training data.
- Hallucination: The model may still generate plausible but factually incorrect content.

- Resource Intensive: Fine-tuning requires significant compute resources, especially GPUs.

c. Future Work:

- Data Expansion: Enhance training quality by increasing data size and variety.
- Model Scaling: Explore larger GPT-2 variants (medium/large) for better performance.
- Hyperparameter Optimization: Tune training parameters (e.g., learning rate, batch size) for improved results.
- Application Development: Deploy the fine-tuned model in real-world applications such as text editor plugins or chatbots for end-user support.

3.6.2 Convert speech to text

3.6.2.1.Overview

Whisper is an advanced deep learning model developed by OpenAI, specialized in Automatic Speech Recognition (ASR) and speech translation. This model is built on the Transformer architecture in a sequence-to-sequence format, enabling it to process and generate text sequences from audio input sequences.

One of Whisper's key strengths is its multitask and multilingual capabilities. It was trained on a massive dataset of 680,000 hours of labeled audio, covering a wide range of languages and recording conditions. This allows the model to achieve high accuracy and exceptional generalization.

The overall working process of the model can be summarized in five main steps:

- Input: An audio segment.
- Preprocessing: Normalize the audio by trimming or padding it into 30-second segments.
- Feature Extraction: Convert the audio into log-Mel spectrogram features.
- Encoding: Feed the audio features into the encoder to produce a semantic representation.
- Decoding: Pass the encoded representation to the decoder to generate the predicted text (either transcription or translation).

The roadmap of the speech-to-text process is detailed step-by-step in Figure 3.11.

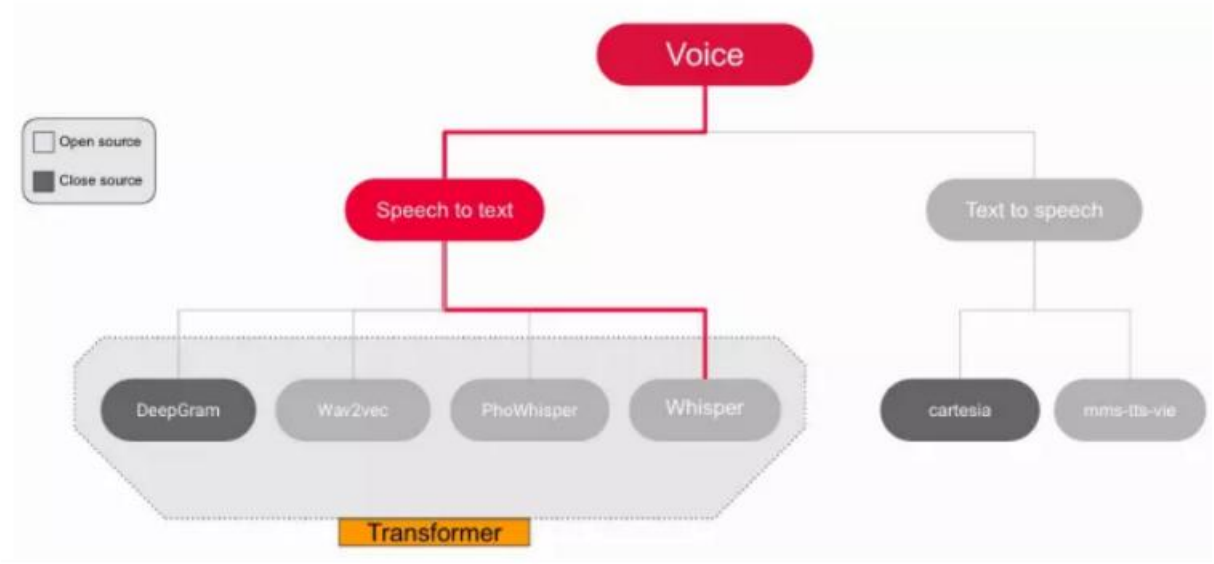

Figure 3.11: Roadmap

3.6.2.2.Detailed Architecture

The model consists of two main components: an Encoder and a Decoder.

The detailed architecture of the speech-to-text model is illustrated in Figure 3.12.

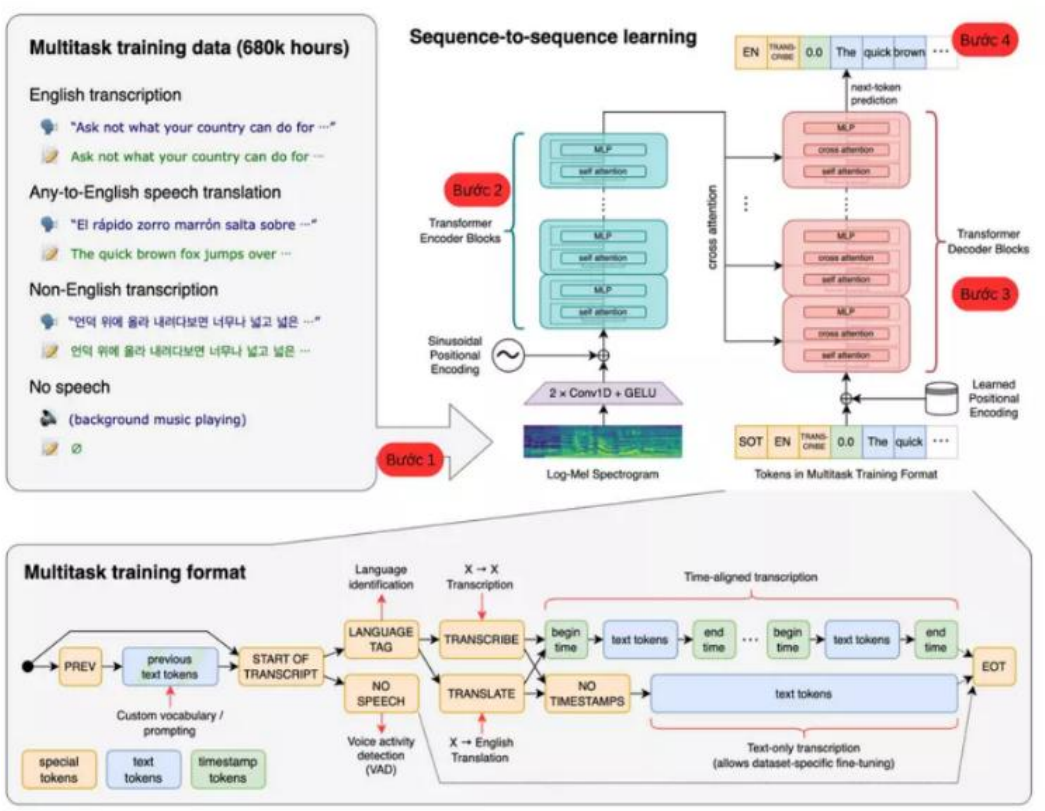

Figure 3.12. Model Architecture

- **Step 1:** Audio Processing:
    - **Format normalization:** The audio data is converted to mono format with a sampling rate of 16,000 Hz.
    - **Segmentation:** The audio is divided into fixed-length segments of 30 seconds. Segments shorter than 30 seconds are padded with silence to meet the required length.
    - **Feature extraction:** The Fast Fourier Transform (FFT) is applied to convert the audio signal from the time domain to the frequency domain. The result is then transformed into a log-Mel spectrogram, a vector-based representation that captures the key features of the audio. This spectrogram serves as the input to the encoder.

Figure 3.13 illustrates the process of input audio preprocessing in the speech-to-text model.

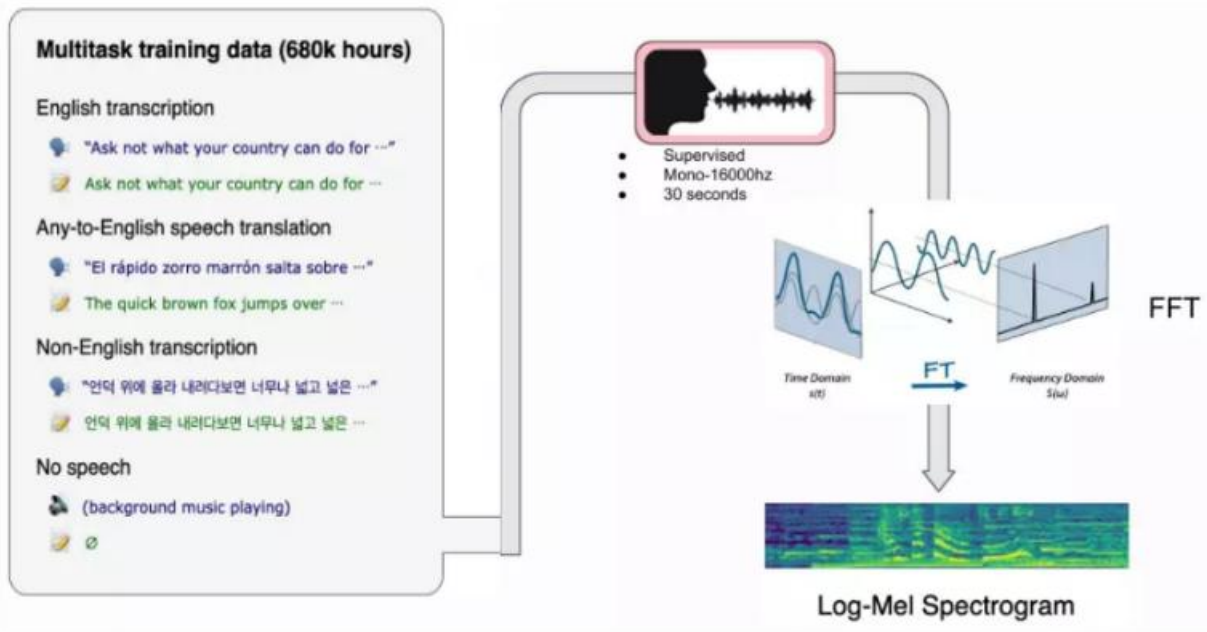

Figure 3.13: Input Audio Processing

Step 2: Encoder:The first component of the model is the Encoder, which is responsible for processing the audio and extracting features that represent the spoken content within the audio segments. It consists of the following parts:

- 2 x Conv1D layers with GELU activation function: Used to extract features from the input log-Mel spectrogram.

- Positional Embedding: Whisper uses sinusoidal positional embeddings to encode both the absolute position of each token and the relative positions between tokens.

Figure 3.14 provides a detailed illustration of the Sinusoidal Positional Embedding process in the speech-to-text model.

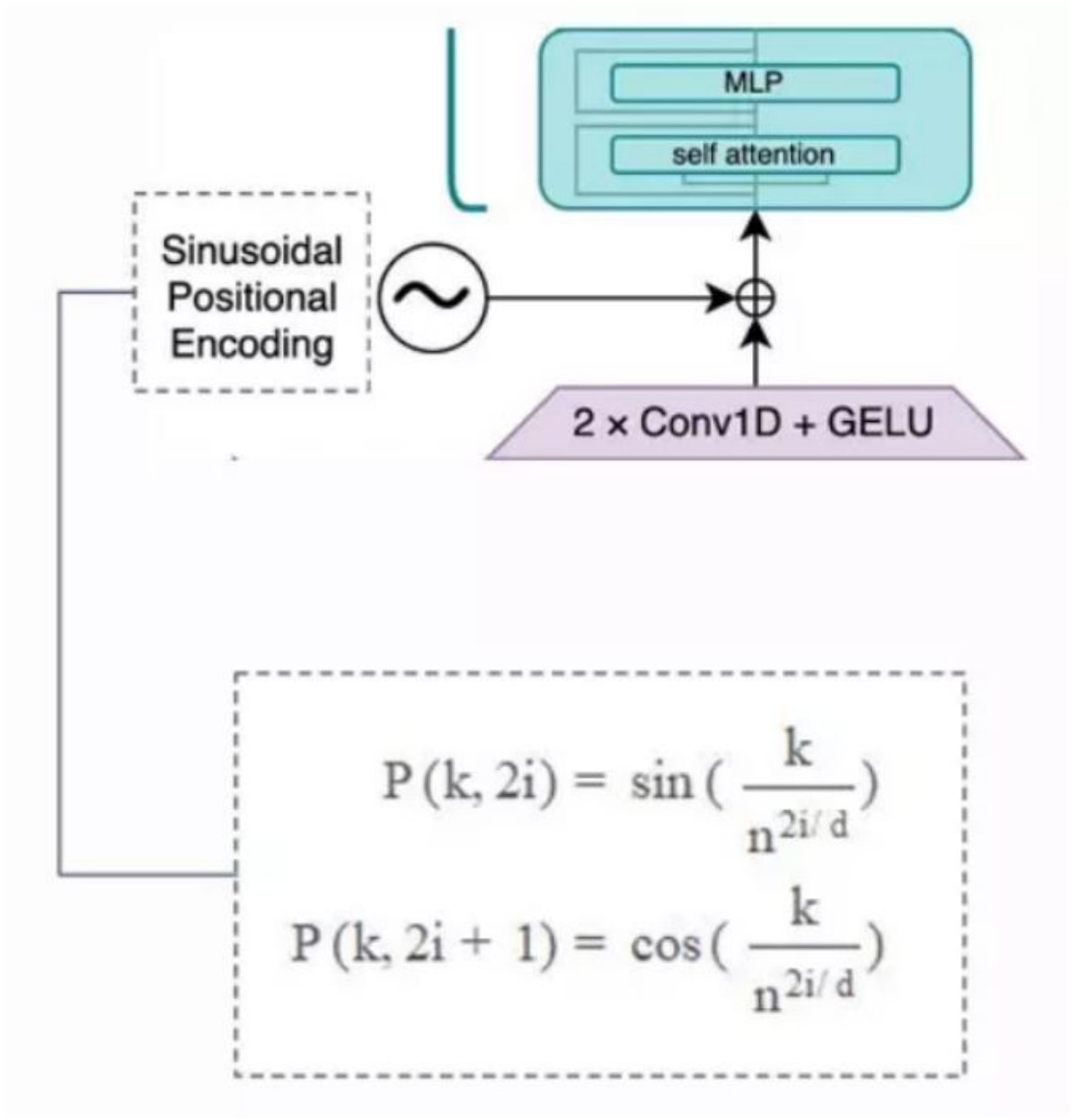

Figure 3.14: Sinusoidal Positional Embedding

- Encoder block:

Figure 3.15 illustrates the detailed architecture of the Transformer encoder used for the speech-to-text task.

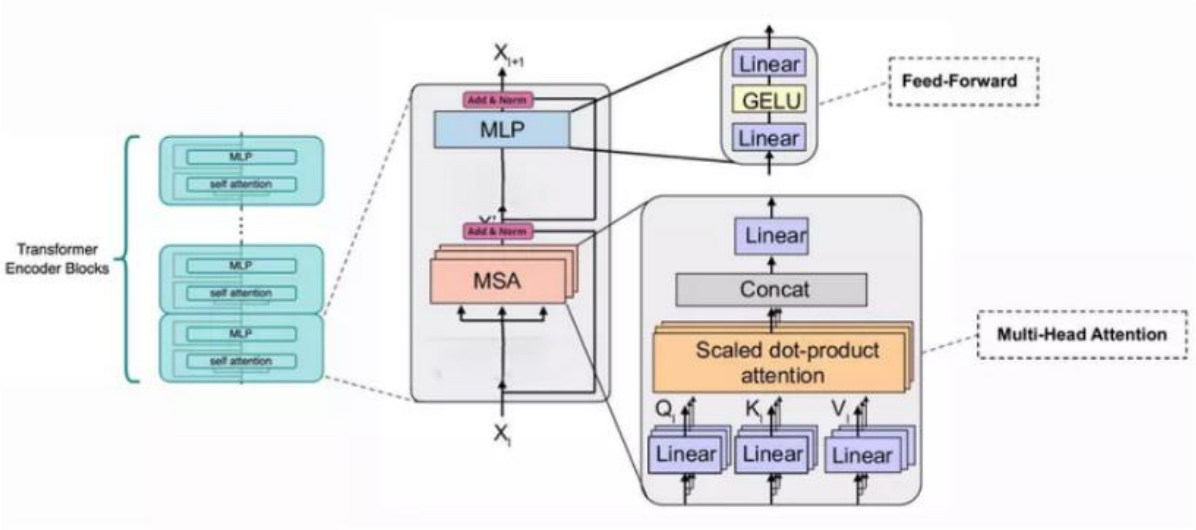

Figure 3.15. Transformer Encoder Blocks

- A standard Encoder block consists of a multi-headed self-attention layer and a feed-forward layer, both followed by layer normalization and the GELU activation function.
- Multi-Head Attention helps mitigate the limitations of global attention by capturing relationships between words from multiple perspectives. It works by splitting each of the Q, K, and V matrices into h smaller parts (heads). Then, Scaled Dot-Product Attention is applied to each part, and the results are concatenated and combined.
- Scaled Dot-Product Attention focuses on the more important parts of the input sequence, effectively modeling semantic relationships and improving the model's ability to learn long-range dependencies.
- Add & Norm Layer simply normalizes the output of the multi-head attention layer, improving convergence and model stability.
- Residual Connection minimizes the vanishing gradient problem and ensures that contextual representations of input tokens retain their original information.

- Feed-Forward Network (FFN): This block has a structure similar to other Transformer models, consisting of two linear layers with a non-linear GELU activation function in between.

Step 3: Decoder:

- The decoding part of the model is the Decoder, which is responsible for generating the text output (either transcription or translation). Whisper uses a fairly standard decoder architecture, similar to those found in many Transformer-based decoders. It includes the following components:
- Multitask training format:
  This is a simple format used to define all tasks and relevant information as a sequence of input tokens fed into the Decoder. These tokens include a language tag (e.g., *en* or *vi*) and a task specification indicating whether the output should be a transcription or a translation. This clever design enables the model to handle a wide range of NLP tasks with flexibility.

- Prev (Previous tokens): These are the input tokens for the model, including the previous tokens that have already been processed.
- Prev text tokens: These are the previously decoded text tokens.
- Start of transcript: The model identifies initial information such as the language of the audio and then decides whether to proceed with transcription or translation.
- Language tag: Used to instruct the model to process the audio in the corresponding language.
- Transcribe: Converts the audio signal into text in the same language.
- Translate: Translates the audio signal from one language into English.
- Voice Activity Detection: This block determines whether speech is present in the audio signal. If no voice is detected, the model skips transcription for that segment and jumps directly to the End of Transcript (EOT).
- Time-aligned transcription: This process produces a transcription with timestamps. The model inserts special tokens indicating the start time and end time of each transcribed segment.

- Text-only transcription: The model generates only text tokens without timestamps, meaning only the content is provided without information on start/end times for each phrase.
- EOT (End of Transcript): Marks the end of the transcription or translation process. The model has completed its task and there are no further tokens or data to process.

Positional Embedding:

Figure 3.16 shows the token sequence used in the multitask training format, where each token is combined with learned positional encoding to provide input for the model.

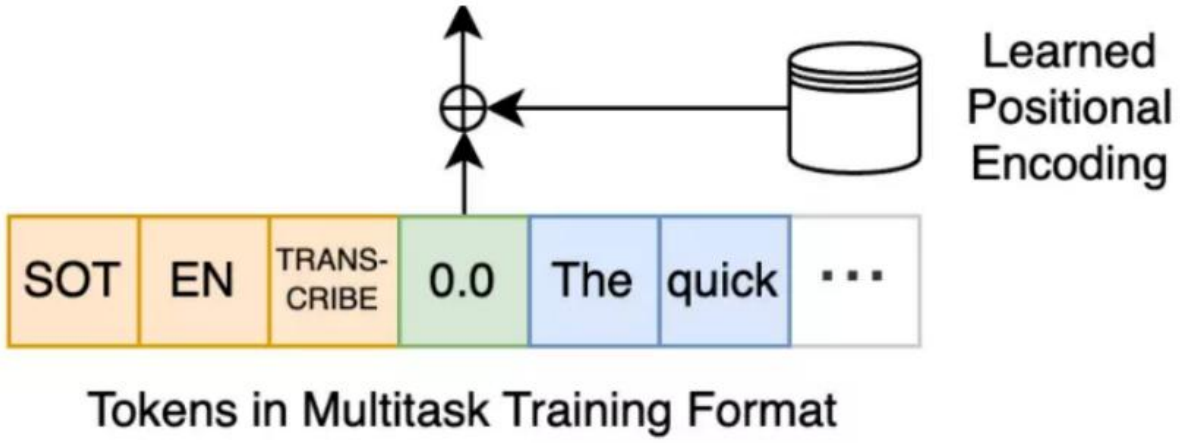

Figure 3.16: Positional Embedding

- Model uses learned positional embeddings for the decoder block. Each token is represented by a learnable vector during training, similar to how token embeddings are learned.
- Decoder block: The standard decoder block consists of Masked Multi-Head Attention, a Feed-Forward layer, and Cross-Attention layers, all followed by Layer Normalization and the GELU activation function.

Figure 3.17 illustrates the architecture of the decoder in the Transformer model used for the speech-to-text task.

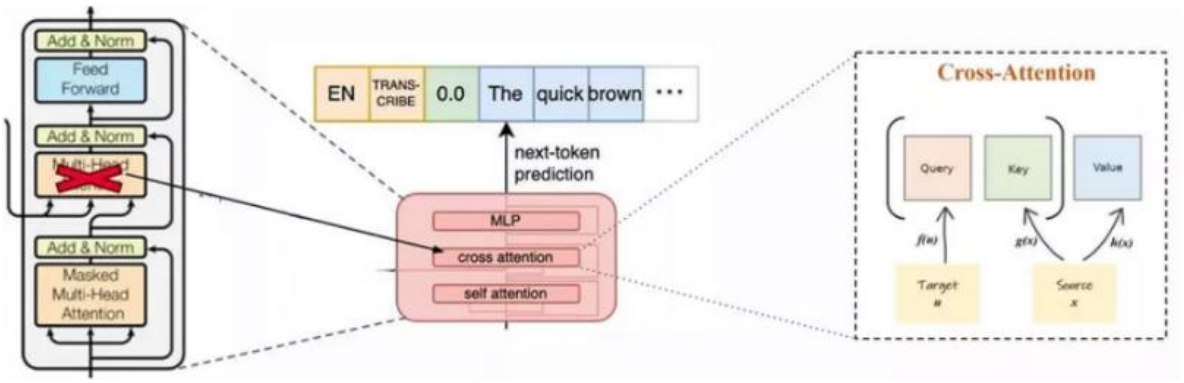

Figure 3.17: Decoder block

Step 4: Generate Output Tokens:Add the predicted token to the sequence of predicted tokens.

- Example:
  - Start with the token sequence:
    |<EN>|,|<TRANSCRIBE>|,|<0.0>||<EN>|, |<TRANSCRIBE>|, |<0.0>||<EN>|,|<TRANSCRIBE>|,|<0.0>|

  - If the next predicted token is |"The"||"The"||"The"|, the updated token sequence becomes:

    |<EN>|,|<TRANSCRIBE>|,|<0.0>|,|"The"||<EN>|, |<TRANSCRIBE>|, |<0.0>|,|"The"||<EN>|,|<TRANSCRIBE>|,|<0.0>|,|"The"|

Table 3.14 describes the probability distribution of words in the speech-to-text model.

Token	Probability distribution
“nhanh”	0.7
“chậm”	0.1
“trai”	0.05
“gái”	0.15

Table 3.14: Probability Distribution Table

  - Sample a token from the distribution:The token |<“quick”>|, which has the highest probability of 0.7, is selected as the next token.

- Repeat steps 1–4 until either the entire audio has been transcribed or the end-of-sequence token |<“eos”>| is encountered. The current token sequence is |<EN>|,|<TRANSCRIBE>|,|<0.0>|,|“The”|,|“quick”||<EN>|, |<TRANSCRIBE>|, |<0.0>|, |“The”|, |“quick”||<EN>|,|<TRANSCRIBE>|,|<0.0>|,|“The”|,|“quick”|, and the process continues until a complete sentence is generated. An example of a completed sequence could be: |<EN>|,|<TRANSCRIBE>|,|<0.0>|,|“The”|,|“quick”|,|“fox”|,|<“eos”>|| <EN>|, |<TRANSCRIBE>|, |<0.0>|, |“The”|, |“quick”|, |“fox”|, |<“eos”>||<EN>|,|<TRANSCRIBE>|,|<0.0>|,|“The”|,|“quick”|,|“fox”|,| <“eos”>|

3.6.2.3. Multitask Training Mechanism and Parameters

Model uses a very smart input format for the decoder, which allows it to perform multiple tasks simply by changing the special tokens at the beginning of the sequence.

- PREV: Contains the tokens of the previous context (if any).
- START OF TRANSCRIPT: Indicates the beginning of the transcription/translation process.
- Language tag: A token specifying the language of the audio (e.g., <en>, <vi>).
- Task tag: Specifies whether the task is transcription (same language) or translation into English.
- NO SPEECH: A token indicating that the audio segment contains no speech.
- Timestamp tokens: Allow the model to perform time-aligned transcription.
- End of Transcript: Marks the end of the text generation process.

Thanks to this input format, Whisper can perform multiple tasks with a single model, including:

- Language Identification
- Monolingual Transcription
- Any-to-English Translation
- Voice Activity Detection
- Training data: 680,000 hours of labeled audio
- Input audio format: Mono, 16,000 Hz
- Processing segment length: 30 seconds
- Input features: Log-Mel Spectrogram
- Architecture: Transformer Encoder-Decoder
- Positional Embedding (Encoder): Sinusoidal

- Positional Embedding (Decoder): Learned

3.6.2.4. Inference Process

The text generation process in Model is autoregressive.

- **Start:** The input sequence for the decoder is initialized with special tokens, e.g., <SOT>,<en>,<TRANSCRIBE>,<0.0><SOT>, <en>, <TRANSCRIBE>, <0.0><SOT>,<en>,<TRANSCRIBE>,<0.0>.
- **First token prediction:** The model predicts a probability distribution over possible text tokens. The token with the highest probability (or sampled from the distribution) is selected.
  Example: "The".
- **Iteration:** The predicted token is appended to the decoder's input sequence: <SOT>,<en>,<TRANSCRIBE>,<0.0>,"The"<SOT>, <en>, <TRANSCRIBE>, <0.0>, "The"<SOT>,<en>,<TRANSCRIBE>,<0.0>,"The".
  The process repeats, predicting one new token at a time based on both the audio representation and the previously generated tokens.
- **Termination:** The process stops when the model predicts the <EOT> (End of Transcript) token or reaches the maximum sequence length.

3.6.2.5. Evaluation of Results

- **Outstanding performance:** Thanks to its massive dataset and Transformer architecture, Whisper achieves high accuracy in both clean studio conditions and noisy environments.
- **Flexibility:** Its multitask and multilingual capabilities within a single model represent a breakthrough, simplifying complex speech processing workflows.
- **Generalization ability:** The model performs well across various accents, dialects, and topics without requiring fine-tuning for specific datasets.
- **Long-context handling:** Processing 30-second segments enables the model to capture longer contextual information compared to traditional methods.
- **High computational resource requirements:** Large Transformer-based models like Whisper demand powerful hardware (e.g., GPUs) for both training and inference.
- **Latency:** The autoregressive nature of the model may introduce latency, making it less suitable for applications that require strict real-time processing.

- **"Hallucination":** In some cases, the model may generate text that is not present in the original audio, especially when the input signal is unclear or repetitive.

3.6.2.6. Conclusion

Model is a powerful, comprehensive, and groundbreaking model for speech recognition and translation.By combining a modern Transformer architecture with a large-scale training dataset and an intelligent multitask mechanism, Model has set a new standard for performance and flexibility in the field of automatic speech processing.This report has outlined the model's architecture and operational mechanisms in detail, highlighting the complexity and sophistication behind its impressive performance.

Chapter 4: IMPLEMENTATION AND EXPERIMENTATION

4.1. Deployment Environment

4.1.1. Source Code Storage and Management Environment

To manage the source code and development workflow of the project, we adopted a modern set of tools and methodologies, with Git serving as the core foundation. Git is a Distributed Version Control System (DVCS) that enables each developer to maintain a complete and independent copy of the entire project, including its full history of changes. This architecture promotes parallel development and operational flexibility.

To facilitate collaboration and centralized storage, we utilized GitHub, a cloud-based platform built on top of Git. GitHub not only serves as a remote repository but also provides a collaborative workspace where community-driven features such as discussions, issue tracking, and code review are integrated, significantly enhancing teamwork efficiency.

Our workflow is structurally designed to ensure code quality. All development activities—whether implementing new features or fixing bugs—are performed on independent branches created from the main branch (master). Once work on a feature branch is complete, a developer opens a Pull Request (PR) on GitHub to propose merging their changes. This PR acts not only as a merge request but also as the entry point for our automated quality assurance pipeline.

Immediately, GitHub Actions, our integrated Continuous Integration/Continuous Deployment (CI/CD) system, is triggered. It automatically executes a series of tasks such as building the project, running test suites, and performing code analysis. A Pull Request is only reviewed and approved by other team members after all automated checks pass successfully, ensuring the overall integrity and quality of the codebase.

Figure 4.1 illustrates the GitHub management environment within the text editor system.

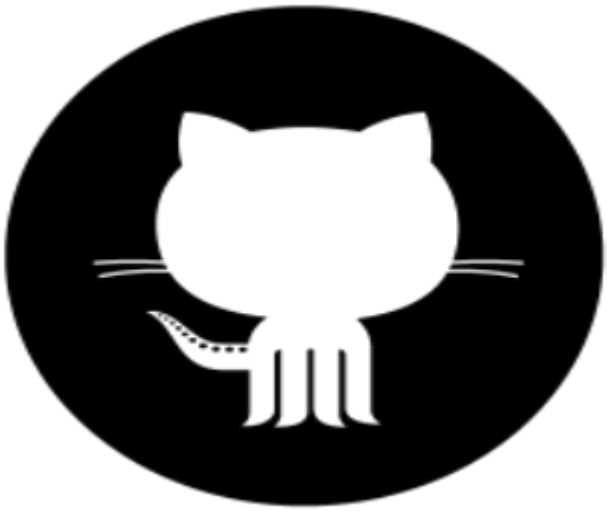

Figure 4.1: Github

4.1.2. Project Development Environment

IDE: Visual Studio Code

- Visual Studio Code (commonly referred to as VS Code) is a lightweight yet powerful source code editor developed by Microsoft. It supports development across various platforms including Windows, macOS, and Linux, and is well-suited even for computers with moderate specifications due to its optimized performance.
- VS Code provides a streamlined coding experience with features such as syntax highlighting, intelligent code completion, debugging, and built-in Git integration. These capabilities significantly accelerate development and improve code quality.
- Additionally, the extensibility of VS Code is a major strength. A vast library of extensions is available to support a wide range of programming languages and frameworks. Notable extensions used in this project include:
    - **ESLint** – for real-time JavaScript/TypeScript code linting and quality enforcement
    - **GitLens** – enhances Git capabilities and history exploration
    - **ReactJS**, **FastAPI**, **Vue.js** – for framework-specific development support
- These tools collectively contribute to a flexible, developer-friendly environment that enhances productivity during all stages of project development.

Figure 4.2 illustrates the code execution environment within the text editor system.

Figure 4.2: Visual studio code

API Testing Tool: Postman

- Postman is a widely used platform for designing, testing, and documenting APIs, particularly RESTful services. It offers an intuitive graphical user interface that enables developers to construct HTTP requests, inspect responses, and automate test scenarios without the need to write code from scratch.
- Postman supports a wide range of HTTP methods (GET, POST, PUT, DELETE, etc.) and allows for the inclusion of parameters, headers, body content, and authentication tokens. This flexibility makes it an essential tool for validating API functionality, verifying data exchange formats (e.g., JSON, XML), and ensuring endpoint correctness throughout the development lifecycle.
- In this project, Postman was employed during the integration and testing phases to:
    - Interactively test RESTful API endpoints.
    - Simulate real-world API calls and inspect server responses.
    - Perform regression tests for critical API functions.
    - Ensure proper communication between the front-end and back-end modules.

- Its ease of use, combined with extensive support for automation and scripting, makes Postman an indispensable tool for modern API development and testing workflows

Figure 4.3 illustrates the API testing environment within the text editor system.

Figure 4.3: Postman

Database: MySql WorkBench

Figure 4.4 illustrates the database management environment within the text editor system.

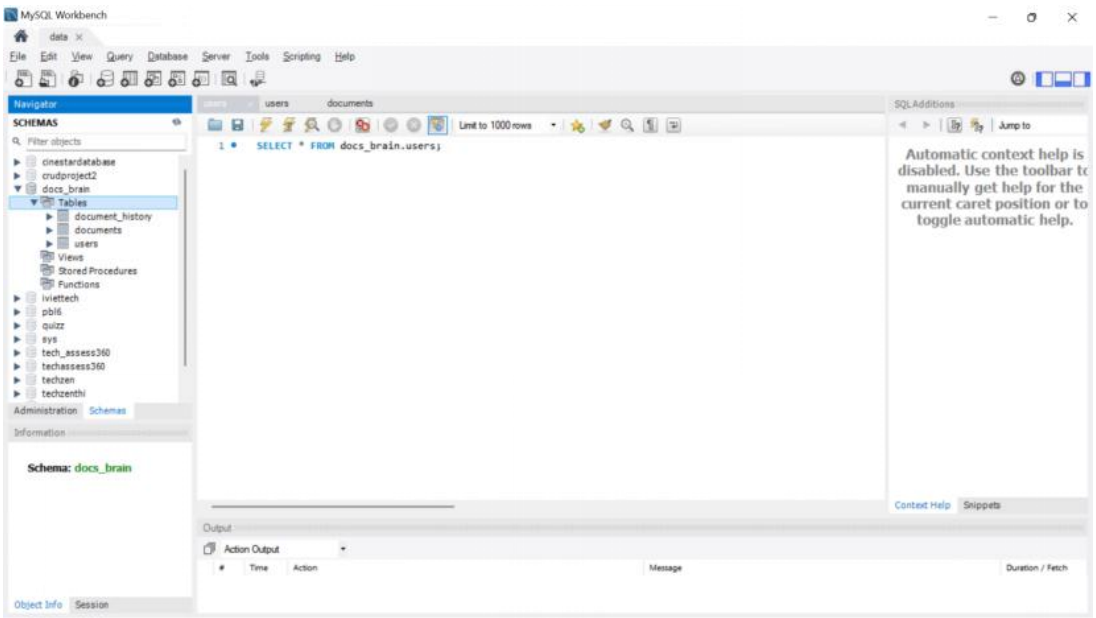

Figure 4.4: Mysql WorkBench

4.2. Achieved Results

4.2.1. Main Interface of the Application

Figure 4.5 illustrates the login interface in the text editor system.

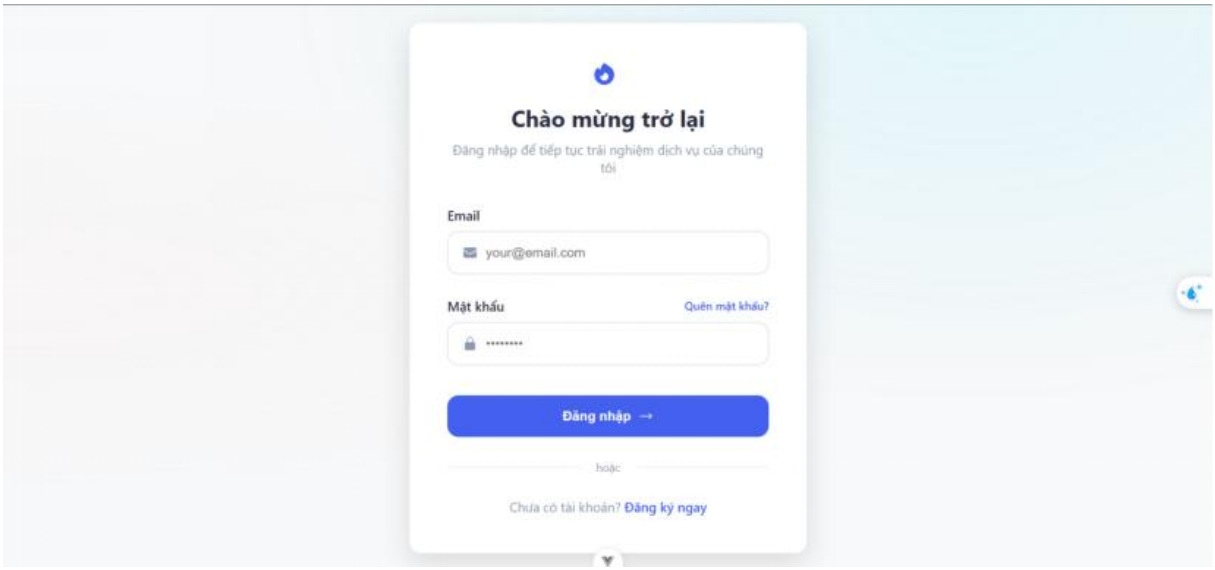

Figure 4.5. Login Interface

Figure 4.6 illustrates the Register interface in the text editor system.

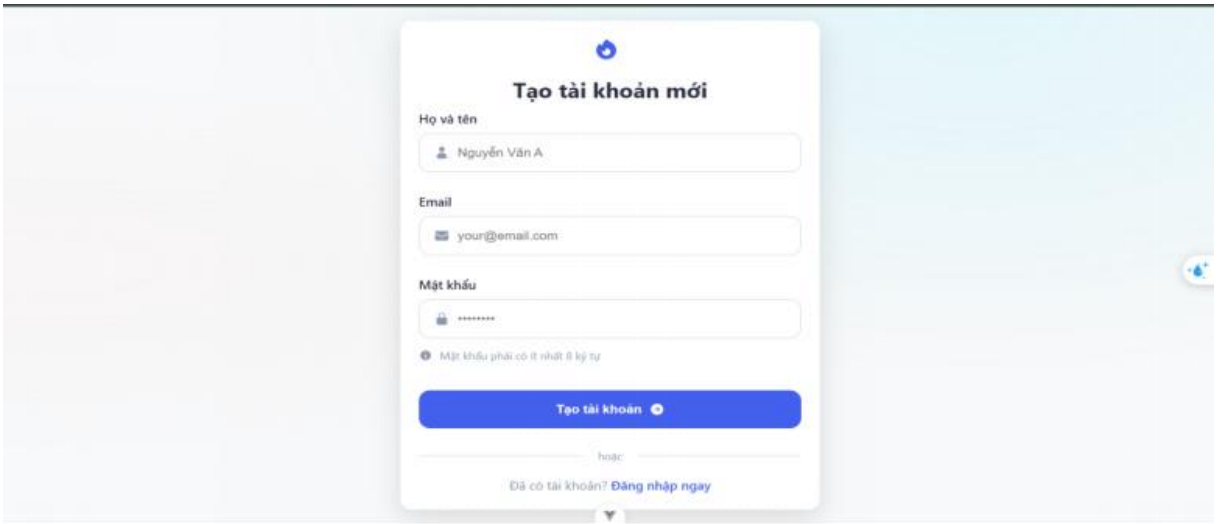

Figure 4.6. Register Interface

Figure 4.7 illustrates the Home page interface in the text editor system.

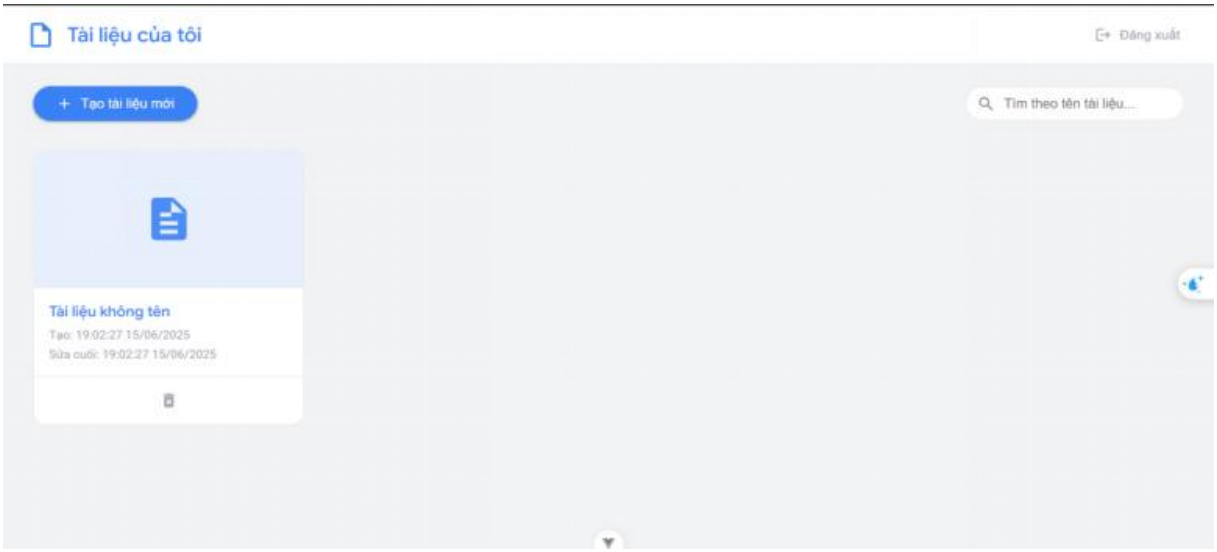

Figure 4.7: Home Page Interface

Figure 4.8 illustrates the editor interface in the text editor system.

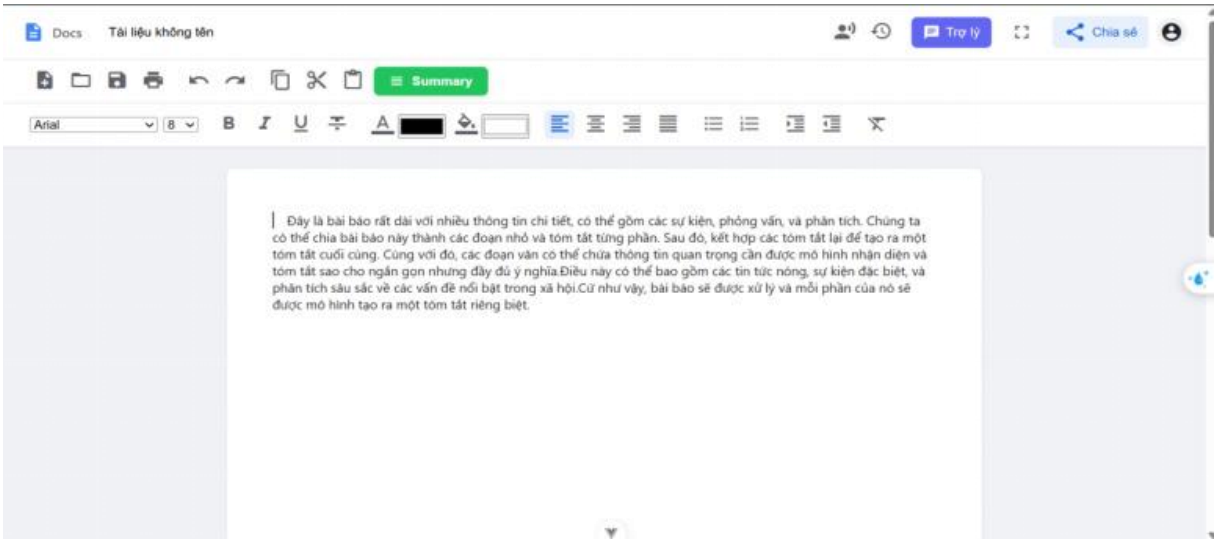

Figure 4.8:Editor Interface

Figure 4.9 illustrates the text summarization interface in the text editor system.

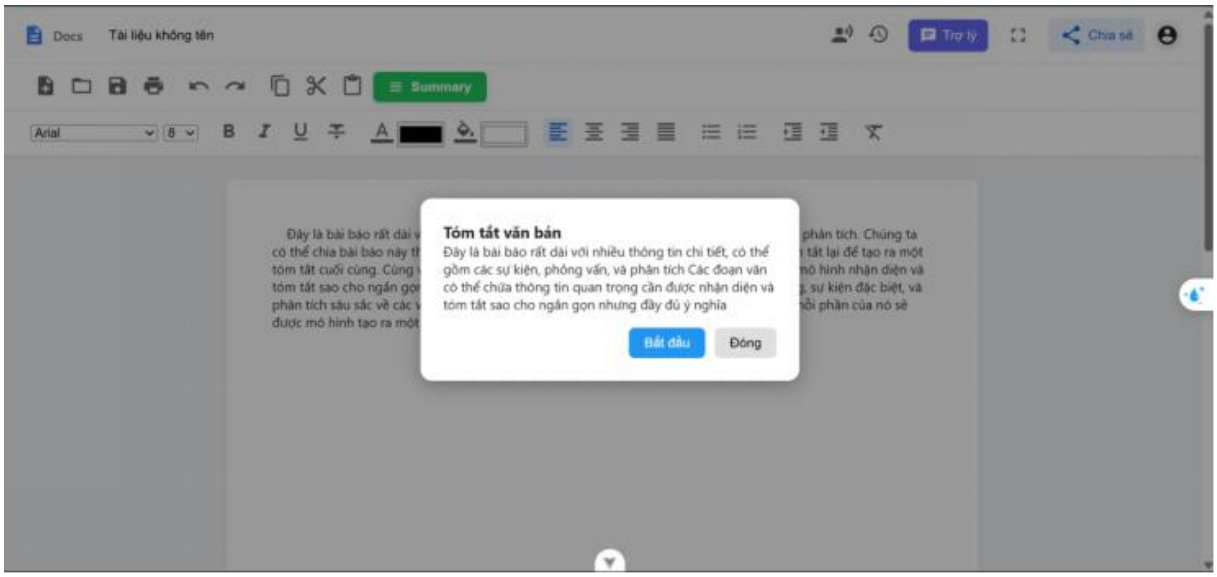

Figure 4.9. Text Summarization Interface

Figure 4.10 illustrates the sentence suggestion interface in the text editor system.

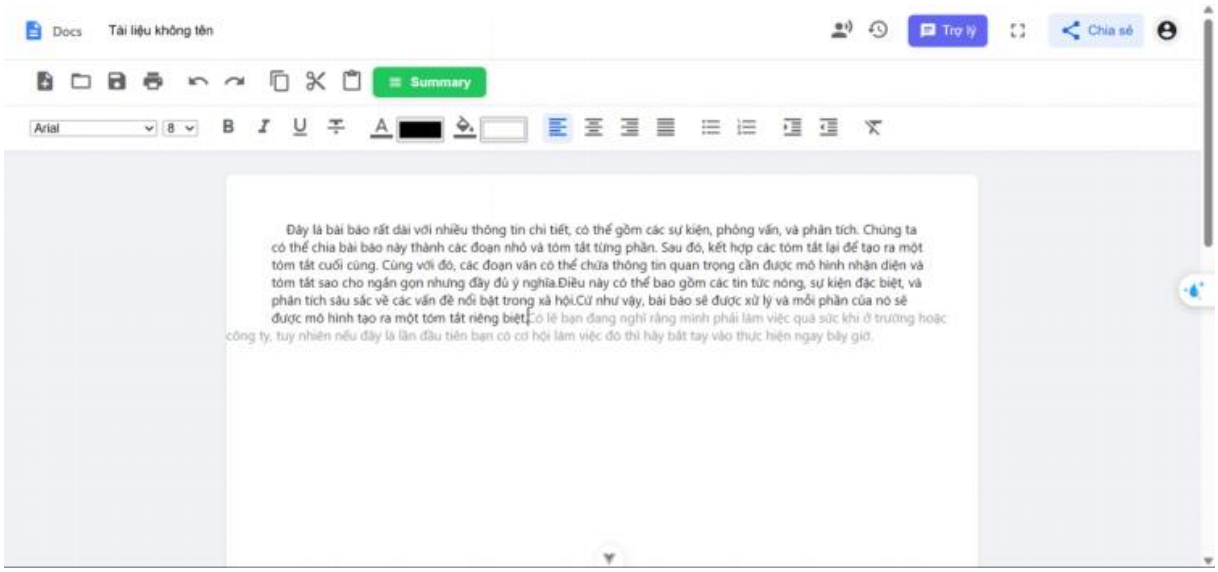

Figure 4.10. Sentence Suggestion Interface

Figure 4.11 illustrates the document assistant interface in the text editor system.

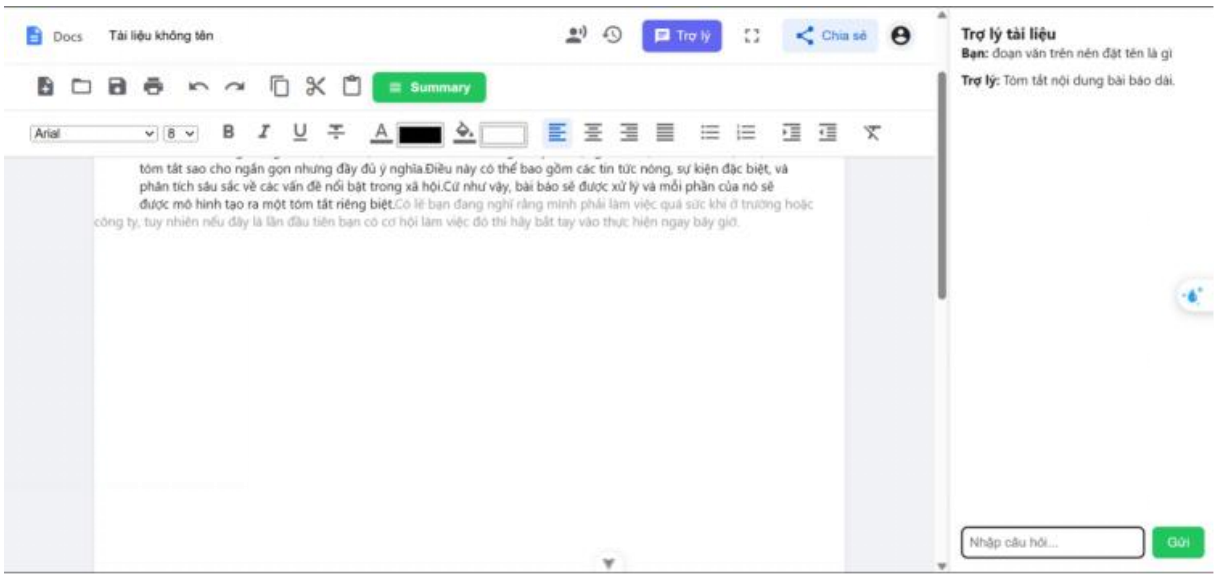

Figure 4.11. Document Assistant Interface

4.2.2. Model Training Results

4.2.2.1Model text summarization result

Figure 4.12 illustrates the results of the T5 model in the text summarization task.

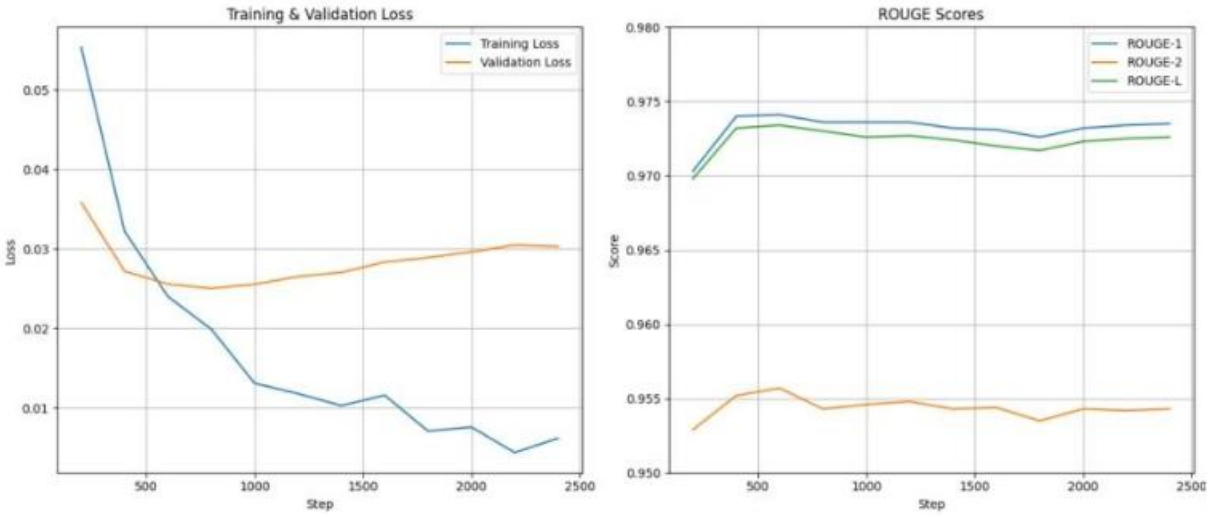

Figure 4.12. T5 Summarization Model Training Results

Table 4.1 describes the results of training loss, validation loss, ROUGE-1, ROUGE-2, and ROUGE-L scores.

Table 4.1. T5 Summarization Model Training Results

Parameters	Value
Train Loss	0.0062
Val Loss	0.0302
Rouge1	0.9735
Rouge2	0.9543
RougeL	0.9726

- The training chart clearly illustrates the model's learning process, with a consistent decrease in training loss, indicating a continuous improvement in the model's learning performance over each step. However, the validation loss, after an initial decline, starts to slightly increase after around 1000 steps. This reflects signs that the model begins memorizing the training data rather than generalizing – a common phenomenon known as overfitting. Meanwhile, the ROUGE scores reach high values early on and remain stable throughout the training process. This indicates that the model can generate text with good quality, preserving content and structure similar to the original text. Therefore, it can be concluded that the model has learned a suitable representation for the task. However, early stopping should be considered to prevent a decline in generalization capability.

4.2.2.2 Model text generation

Table 4.2 describes the results of the text generation model with evaluation metrics including ROUGE-1, ROUGE-2, and ROUGE-L.

Table 4.2. Result model text generation

Parameter	Value
Rouge 1	46.63
Rouge 2	32.17

Rouge L	38.2

- ROUGE-1 (46.63%) indicates that the model is capable of preserving nearly half of the important words from the reference text — this reflects a good level of lexical coverage.
- ROUGE-2 (32.17%) is a fairly impressive result, showing that the model not only retains vocabulary but also captures contextual word pairs, which is crucial in tasks like text summarization or text generation.
- ROUGE-L (38.20%) suggests that the model can generate sequential word structures that match the reference text, indicating coherence and structural consistency in the generated sentences.

4.2.2.3 Model speech to text

Table 4.3 describes the results of the Spech to text model with evaluation metrics including

Table 4.3. Result model speech to text

Parameter	Value
SLER	32.6
WER	38.2

- Errors in word segmentation or spacing have a greater impact on WER, but less so on SLER.
- SLER is a more appropriate metric for evaluating ASR performance in Vietnamese or other languages with unclear word boundaries.
- The model demonstrates decent syllable recognition with an accuracy of approximately 67.4%.
- However, further improvements are needed in context handling and word segmentation to reduce WER.
- It is recommended to prioritize SLER when evaluating ASR systems for Vietnamese, and consider applying post-processing techniques to enhance WER for real-world applications.

CONCLUSION

1. Achieved Results

During the process of studying the theory and implementing the application, the project “Building a Website to Support Vietnamese Text Editing” has achieved the following notable results:

- **Theoretically**, the team applied learned knowledge to:
  - Design the system from the backend to the user interface.
  - Analyze and build the database to store documents and user data.
  - Develop APIs using FastAPI.
  - Deploy the system using Docker for service packaging, and VPS as the infrastructure platform, ensuring the system runs stably and is easy to maintain.
- **Functionally**, the system has completed the main features such as:
  - Account management: user registration and login.
  - Professional text editing and formatting.
  - Intelligent context-aware sentence suggestions.
  - Quick summarization of long text content
  - Speech-to-text conversion
  - Integration of a virtual assistant (chatbot) to support explanations, synonyms/antonyms, and effective expressions.
  - Document version history management and exporting files in popular formats such as .txt, .docx, and .pdf.
  - User and document management.
- **Interface-wise**, the application is designed to be modern, intuitive, user-friendly, and ready for future commercial expansion.

2. Limitations

Alongside the achieved results, the system still has some limitations such as:

- Some advanced features, like multi-user collaboration on a single document or automatic style editing, are still in the research phase.
- Certain AI modules, such as speech-to-text, do not perform well in noisy environments or with regional accents.

3.Development Directions

In the near future, the system will continue to be improved in the following directions:

- Develop a dedicated website administration interface for admins to monitor and manage users, documents, and activity history.
- Add real-time collaboration features, allowing multiple users to edit a document simultaneously.
- Deploy mobile applications on Android and iOS to increase convenience.
- Integrate an internal social network where users can share documents, receive feedback, and engage in academic exchanges.
- Enhance AI capabilities, especially features such as plagiarism checking, writing style improvement suggestions, and personalized recommendations based on writing purposes.

REFERENCES

[1] Nguyen, T. T., & Le, H. T. (2020). RESTful API Design. In API Design Patterns (pp. 19–45). Springer. https://books.google.com.vn/books?id=DYC3DwAAQBAJ

[2] Willebrands, M. (2020). Learning Vue.js 2.0. Packt Publishing. https://books.google.com.vn/books?id=nszcDgAAQBAJ

[3] Murach, J., & Syverson, B. (2018). Murach's MySQL (3rd ed.). Mike Murach & Associates. https://books.google.com.vn/books?id=hS8FEQAAQBAJ

[4] Docker Inc. (n.d.). Docker overview. Docker Documentation. https://docs.docker.com/get-started/docker-overview/

[5] User.com.vn. (n.d.). VPS là gì? Hiểu về máy chủ ảo VPS và ứng dụng. https://user.com.vn/vps-la-gi-hieu-ve-may-chu-ao-vps-va-ung-dung/

[6] Tiangolo, S. (n.d.). FastAPI. https://fastapi.tiangolo.com/

[7] GeeksforGeeks. (n.d.). What is WebSocket and how it is different from the HTTP?. https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-different-from-the-http/

[8] Kubernetes. (n.d.). Kubernetes overview. Kubernetes Documentation. https://kubernetes.io/docs/concepts/overview/

[9] Viblo. (2019, May 8). Hiểu sâu về Virtual DOM trong ReactJS. https://viblo.asia/p/hieu-sao-ve-virtual-dom-trong-reactjs-bWrZngDblxw

[10] Pinia. (n.d.). Introduction. Pinia Documentation. https://pinia.vuejs.org/introduction.html

[11] 200Lab. (n.d.). SSR là gì? Tìm hiểu về Server-Side Rendering. https://200lab.io/blog/ssr-la-gi

[12] Khan, B., Usman, M. (Senior Member, IEEE), Khan, I., Khan, J., Hussain, D., & Gu, Y. H. (2025). Next-generation text summarization: A T5-LSTM FusionNet hybrid approach for psychological data. Proceedings of the IEEE International Conference on Advanced Computing and Communication Systems (ICACCS). (Note: Conference details and exact pages to be confirmed based on full publication data).

[13] NVIDIA. (n.d.). What is a transformer model?. NVIDIA Blogs. https://blogs.nvidia.com/blog/what-is-a-transformer-model/

[14] Papers with Code. (n.d.). C4 dataset. https://paperswithcode.com/dataset/c4

[15] H. Nguyen, L. Phan, J. Anibal, A. Peltekian and H. Tran, VieSum: How robustare transformer-based models on Vietnamese summarization? preprint, arXiv:2110.04257[cs.CL] (2021)

[16] Lagler, K., Schindelegger, M., Böhm, J., Krásná, H., & Nilsson, T. (2013). GPT2: Empirical slant delay model for radio space geodetic techniques. Geophysical Research Letters, 40(6), 1069–1073. https://doi.org/10.1002/grl.50288

[17] GeeksforGeeks. (n.d.). Introduction to recurrent neural network. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[18] GeeksforGeeks. (n.d.). Deep learning: Introduction to long short-term memory. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[19] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017). (Note: Exact pages and DOI to be confirmed based on full publication data).

[20] Das, A., Kong, W., Sen, R., & Zhou, Y. (2025). A decoder-only foundation model for time-series forecasting. Proceedings of the International Conference on Machine Learning (ICML). (Note: Exact pages and DOI to be confirmed based on full publication data).

[21] Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1037– 1042

[22] Long Phan , Hieu Tran , Hieu Nguyen , Trieu H. Trinh ,VietAI Research.2023.ViT5: Pretrained Text-to-Text Transformer for Vietnamese Language Generation.

[23] Viblo. (2023, November 9). ASR paper reading: Tìm hiểu cách sử dụng model Whisper speech-to-text cho các đoạn audio dài. https://viblo.asia/p/asr-paper-reading-tim-hieu-cach-su-dung-model-whisper-speech-to-text-cho-cac-doan-audio-dai-m2vJPo1K4eK.

[24] Viet-Bac Le and Laurent Besacier.2009.Automatic Speech Recognition for Under-Resourced Languages: Application to Vietnamese Language.