

Project Title:
Multilingual Learning Web Application

Supervisor:
Muhammad Zubair Khan
zubair2021@namal.edu.pk

Contact Information:
Solae Kim (VC1)
solae.kim82@bcmail.cuny.edu

Multilingual Learning Web Application

0. Introduction

0.1 Preface

This project originated from my personal experience with language learning. I was born and raised in Korea, and later continued my studies in France and the United States. Throughout this journey, I had to master two foreign languages to the level of understanding university-level coursework, which allowed me to deeply empathize with the struggles of language learners.

Today, with smartphone apps, it's easy to look up unfamiliar words. However, when I first started my studies in France, I often relied on paper or electronic dictionaries, searching for words one by one. Beyond grammar and vocabulary, I also had to study idiomatic expressions, synonyms, antonyms, and more. Looking up each word and writing it down to memorize it was not only time-consuming but also laborious. Moreover, the accuracy of online translation tools at the time was questionable, and reading just a few pages of a book could take half a day.

When I moved to the United States to study English again, I faced another challenge. While both French and English use the same alphabet, their pronunciation systems are completely different, and sometimes words that look similar have entirely different meanings. Using both

languages simultaneously became confusing. As I focused more on English, my French skills naturally began to decline. Each time this happened, I tried to write and memorize both languages in my notes. But that process was exhausting and cumbersome, eventually leading me to give up. During those times, I often wished there had been a book that could help me learn both languages at the same time.

Remarkably, today, I study languages in a way that is vastly different from when I first started my studies abroad. Instead of referencing paper or electronic dictionaries to find individual words, I can now simply point to a word in an image and instantly discover its meaning. Additionally, tools like ChatGPT allow me to seamlessly translate entire pages of text into Korean, French, and English, while also capturing the nuances of the original content. In just seconds, I can extract words, idiomatic expressions, and example sentences from lengthy passages, making the language learning process far more efficient. I believe this approach not only benefits those learning multiple languages simultaneously, but also individuals preparing for language proficiency exams such as TOEIC or IELTS.

With this background, I began exploring ways to leverage advanced technologies to make language learning more efficient and accessible. This project represents the first step toward realizing that vision, focusing not only on language translation, but also on providing tools that help learners maximize their language skills while minimizing the time and effort involved. Through this endeavor, I hope to offer tangible support to language learners worldwide, helping them to discover new opportunities and broaden their horizons.

0.2 Abstract

This project is grounded in my personal experience with foreign language acquisition, aiming to develop a web application that facilitates the efficient learning of multiple languages simultaneously. Traditionally, language learners have relied on paper or electronic dictionaries to look up individual words, a process that is both time-consuming and inefficient. However, the advent of AI-powered tools, such as ChatGPT, has ushered in a paradigm shift in language learning methodologies. These tools offer rapid and accurate translations, along with real-time suggestions for words, idiomatic expressions, and example sentences, thus enabling a more intuitive and efficient learning experience. As a result, learners can significantly reduce the time spent searching for and verifying vocabulary, improving overall learning efficiency. This project seeks to leverage these technological advancements to address the inherent inefficiencies of traditional language learning approaches. By integrating AI-driven solutions, the application aims to create an environment where learners can study multiple languages simultaneously, minimizing the time and effort required while maximizing learning outcomes.

1. Technology Stack

1.1 Development Tools

- 1) Version Control Platform: **GitHub** (<https://github.com/srsk-sci/MultiLingo>)
- 2) IDE: **Visual Studio Code**

1.2 Core Technology

The backend system is implemented using a Python-based web stack, selected for its robustness, flexibility, and ecosystem maturity.

- 1) **Programming Language:** The backend logic is written in **Python**, a high-level programming language known for its clear syntax and strong community support. Python is widely used in web development, data science, and AI applications, making it a fitting choice for extensible language-related services.
- 2) **Backend Framework:** The application is built on **Django**, a high-level Python web framework that promotes rapid development and pragmatic design. Django provides a built-in ORM, secure authentication, modular architecture, and scalable design patterns, making it particularly well-suited for data-centric web applications requiring clean integration with frontend services.

1.3 Frontend Stack (React-Based)

The frontend of the application is built using a modern React-based stack designed for component-oriented architecture and rapid UI development.

- 1) **React** serves as the core JavaScript library for building user interfaces. Its component-based structure enables modularity, reusability, and efficient rendering, which are crucial for maintaining complex frontend systems.
- 2) **React Router DOM** is integrated to support client-side routing in a single-page application (SPA) context. It allows the application to dynamically switch components based on the URL path, facilitating smooth navigation without full page reloads.
- 3) **Tailwind CSS**, in combination with the **Tailwind Vite Plugin**, provides a utility-first approach to styling. Developers can compose responsive and maintainable UIs using predefined utility classes. The Vite plugin enhances the developer experience by enabling instant style updates during development through real-time hot module reloading.

- 4) **Lucide React** is employed for scalable and customizable SVG icon components. It offers a lightweight and consistent visual language, which can be seamlessly embedded in the React environment to enhance the interface's aesthetic coherence.
- 5) **Framer Motion** is used to introduce declarative animations and transitions. This library enables visually rich, interactive interfaces by allowing developers to easily define and manage UI animations in a React-friendly manner.

1.4 Bundling and Development Server

- 1) The build process and development server are powered by **Vite**, a modern frontend tooling framework optimized for speed and developer productivity. Vite offers fast hot module replacement (HMR) and significantly reduced build times compared to traditional bundlers, making it ideal for iterative development in React environments.

1.5 AI Integration Services & Language Data

The system integrates several lightweight, no-authentication-required APIs to support language-related features such as translation, definitions, and semantic search.

- 1) **MyMemory API** utilizes artificial intelligence (AI) to support language translation in real time. It is particularly well-suited for lightweight, educational, or prototype applications where ease of use and rapid integration are prioritized. Although it does not achieve the same level of accuracy or contextual understanding as commercial-grade APIs, it offers reliable performance for non-critical use cases.
- 2) **Free Dictionary API** supplies English word definitions through a RESTful JSON interface. Its straightforward structure enables seamless parsing and integration on both frontend and backend components.
- 3) **Datamuse API** allows the querying of semantically related terms—including synonyms, antonyms, and phonetically similar words—making it especially useful for building language tools based on lexical relationships.

All three APIs share key advantages: they are free, require no authentication, and adhere to RESTful standards. These characteristics make them ideal for use in Django- and React-based architectures focused on education or rapid prototyping, where minimal configuration and quick iteration are critical.

2. Environment Setup and Execution Guide

To ensure proper execution and compatibility of the application, the following software versions and system configurations are recommended. These requirements reflect the minimum environment specifications for successful deployment and development of the web application.

2.1 Backend Setup

2.1.1 Creating and Activating a Virtual Environment

To manage Python dependencies in isolation, a virtual environment is created for the project. The command differs slightly depending on the operating system:

Operating System	Command(s)
Windows	<code>python -m venv venv</code> <code>venv\Scripts\activate</code>
macOS / Linux	<code>python3 -m venv venv</code> <code>source venv/bin/activate</code>

2.1.2 Installing Required Packages and Running the Server

Once the virtual environment is activated, the necessary dependencies specified in the `requirements.txt` file can be installed using the following command:

```
pip install -r requirements.txt
python manage.py runserver
```

Alternatively, in case of installation errors or when only specific dependencies are required, each package listed in the `requirements.txt` file can be installed individually using the `pip` command.

For example: `pip install nltk` or `pip install django-cors-headers`.

2.2 Frontend Setup (React + Vite)

The frontend is developed using the Vite build tool and is structured around React

components styled with Tailwind CSS. The development server enables fast iteration through hot module replacement and minimal build overhead.

2.2.1 Installing Dependencies

Navigate to the frontend project directory and install the required packages using npm:

```
cd frontend  
npm install
```

2.2.2 Running the Development Server

After installation, the development server can be started with the following command:

```
npm run dev
```

3. Methodology

3.1 Project Motivation & Background

This project was inspired by my experience of using AI tools to study multiple languages simultaneously. Through this process, I recognized the potential of AI in streamlining language learning by providing instant translations, contextual word meanings, grammar explanations, and usage examples. Below is a glimpse of how I utilize ChatGPT to retrieve the learning information I need.



America to Fort Bragg,
what's behind Trump's
name changes?



Kayla Epstein
National digital reporter

16 February 2025
President Donald Trump has always understood
the power of branding. As a celebrity
businessman he affixed his name to the facades

Translate this article into Korean and French. Tell me the main grammar, words, idioms, and expressions, respectively.

Here is the translation of the article into **Korean** and **French**, along with grammar explanations, vocabulary, idioms, and expressions.

Korean Translation

제목:
미국만(灣)에서 포트 브래그까지, 트럼프의 이름 변경 뒤에는 무엇이 있는가?

본문:
도널드 트럼프 전 대통령은 항상 브랜딩(상표화)의 힘을 이해해왔다.
유명한 사업가로서, 그는 자신의 이름을 마천루의 외벽에 붙이고, 자신의 이름을 호텔부터 와인에 이르기까지 다양한 제품에 라이선스했다.

French Translation

Titre:
Du golfe du Mexique à Fort Bragg, quelle est la raison des changements de nom de Trump ?

Texte:
Le président Donald Trump a toujours compris le pouvoir du branding.
En tant qu'homme d'affaires célèbre, il a apposé son nom sur les façades de ses gratte-ciels et a concédé sous licence son nom à une gamme de produits, allant des hôtels aux vins.

Figure 1. Example of language learning with ChatGPT that inspired the project

7

Vocabulary (단어 & Vocabulaire)

English	Korean	French
Gulf of America	미국만(灣)	Golfe du Mexique
Fort Bragg	포트 브래그	Fort Bragg
Name change	이름 변경	Changement de nom
President	대통령	Président
Branding	브랜딩 (상표화)	Branding
Celebrity	유명인	Célébrité
Businessman	사업가	Homme d'affaires
Affix	붙이다	Apposer
Facade	외벽	Façade
Skyscraper	마천루	Gratte-ciel
License	라이선스하다	Concéder sous licence
Array	다양한	Gamme
Product	제품	Produit
Hotel	호텔	Hôtel
Wine	와인	Vin

Idioms & Expressions (속어 및 관용 표현)

1. "The power of branding"
 - Korean: 브랜딩(상표화)의 힘
 - French: Le pouvoir du branding
 - Branding의 중요성을 강조하는 표현.
2. "Affixed his name"
 - Korean: 자신의 이름을 붙이다
 - French: Apposer son nom
 - 이름을 어디에 부착하거나 새기는 것을 의미.
3. "Licensed his name"
 - Korean: 자신의 이름을 라이선스하다
 - French: Concéder sous licence son nom
 - 이름을 상업적으로 사용하게 허가하는 것.
4. "An array of products"
 - Korean: 다양한 제품
 - French: Une gamme de produits
 - 여러 종류의 제품을 뜻하는 표현.

Figure 2. Extracting vocabulary, idioms, and expressions using ChatGPT during language study

Create a comparison table of English and French tenses and add Korean translations.

Comparison Table of English and French Tenses with Korean Translations

영어와 프랑스어 시제 비교표 (한국어 번역 포함)

English Tense	French Tense	Explanation (설명)	Example Sentence (예문)
Simple Present (현재형)	Présent de l'indicatif (직설법 현재)	Describes general facts, habits, or current states. (일반적인 사실, 습관, 현재 상태를 나타냄)	English: She works every day. French: Elle travaille tous les jours. Korean: 그녀는 매일 일한다.
Present Continuous (현재 진행형)	Présent progressif (or just "Présent") (현재 진행형 또는 직설법 현재)	Expresses actions happening right now. (지금 이 순간 일어나고 있는 행동을 나타냄)	English: She is working now. French: Elle est en train de travailler maintenant. Korean: 그녀는 지금 일하는 중이다.
Simple Past (과거형)	Passé composé (복합 과거)	Describes completed actions in the past. (과거에 완료된 행동을 나타냄)	English: She worked yesterday. French: Elle a travaillé hier. Korean: 그녀는 어제 일했다.
Past Continuous (과거 진행형)	Imparfait (반과거)	Describes past actions in progress. (과거에 진행 중이었던 행동을 나타냄)	English: She was working when I called. French: Elle travaillait quand j'ai appelé. Korean: 내가 전화했을 때 그녀는 일하고 있었다.
Present Perfect (현재 완료형)	Passé composé (복합 과거)	Describes actions with present relevance. (과거에 일어났지만 현재와 관련이 있는 행동을 나타냄)	English: She has worked here for 5 years. French: Elle a

Message ChatGPT



Figure 3. Tense comparison table generated using ChatGPT

3.2 User-Level System Flowchart

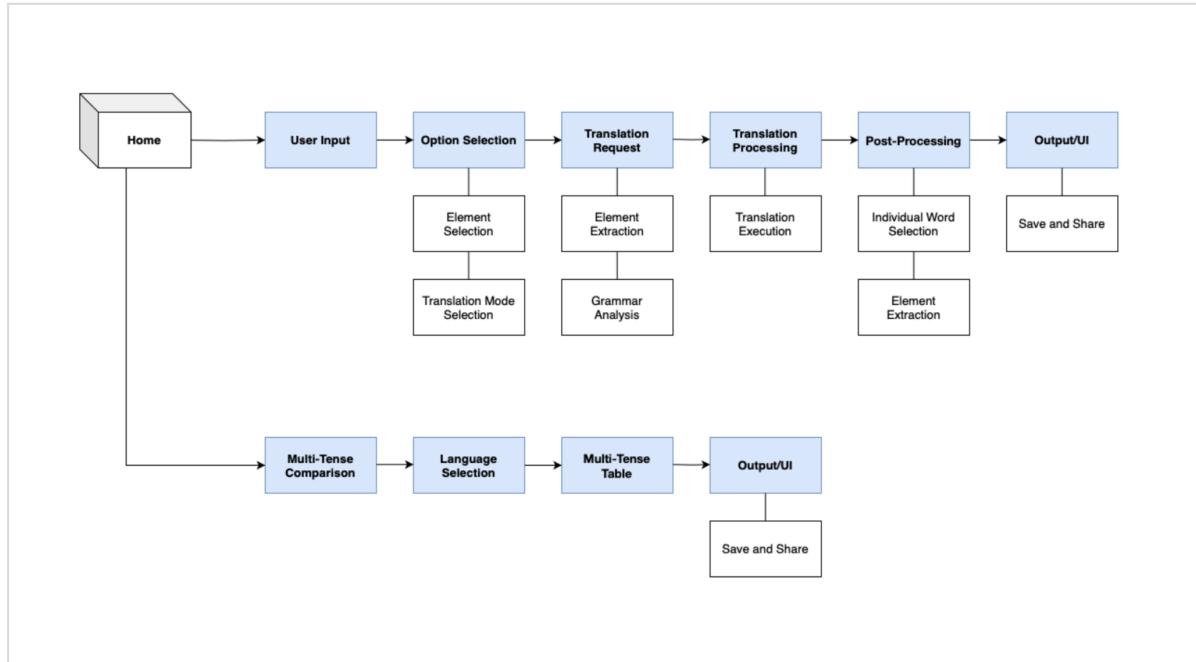


Figure 4. Early-stage flowchart of the multilingual translation web application system

This diagram presents the fundamental processes of a multilingual translation web application, which is structured into two primary workflows. The first process is the **general translation workflow**, which begins when the user inputs text and selects translation parameters. In this phase, users can specify translation elements, including words, idioms, as well as choose the translation mode—either simultaneous or separate translation. The system subsequently conducts text analysis, extracts relevant linguistic elements, and performs grammatical analysis before executing the translation. Following this, a post-processing phase enables users to refine the translation by selecting specific words or adjusting synonyms and antonyms. The final output is then presented through the user interface (UI), where users have the option to save or share the translated content.

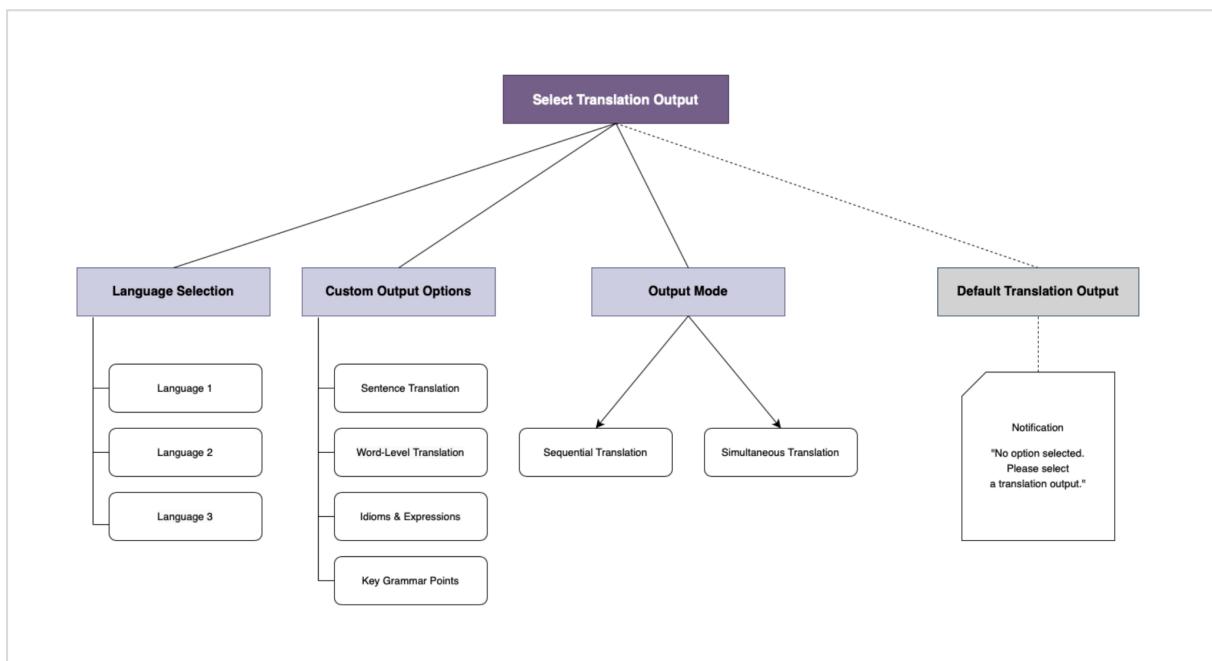


Figure 5. Initial UI design of available user options in the translation application

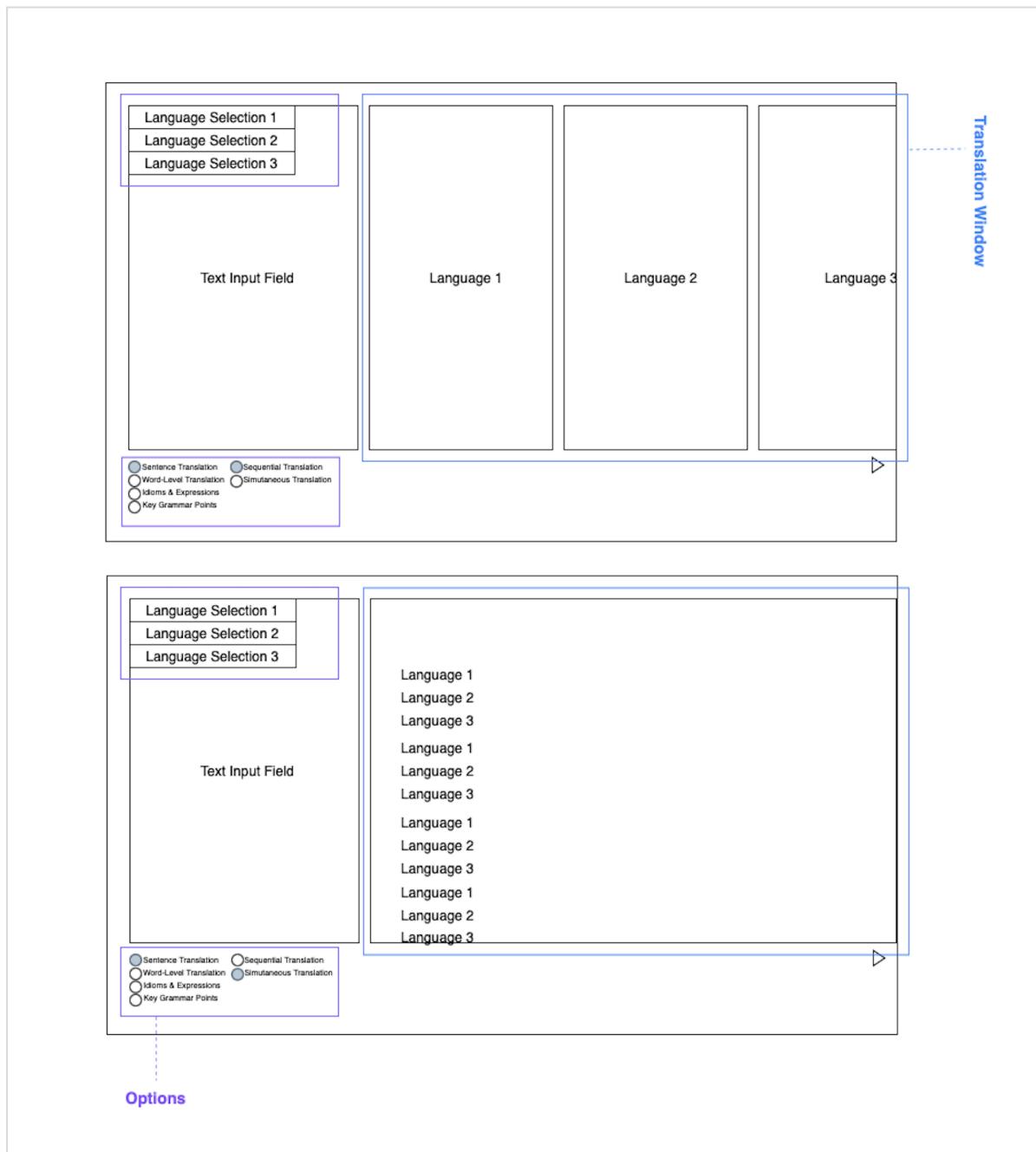


Figure 6. Early-stage UI design comparison of sequential and simultaneous translation windows

The second process focuses on **multilingual tense comparison**, a feature designed to facilitate cross-linguistic tense analysis. Users initiate this function by selecting the languages they wish to compare. The system then generates a comparative tense table, allowing for a systematic examination of tense structures across multiple languages. The results are displayed within the UI, with options for users to save or share the comparison data.

This application integrates both general translation and multilingual tense comparison functionalities, addressing diverse needs in language translation and linguistic analysis.

While the general translation workflow primarily focuses on lexical and syntactic processing, the multilingual tense comparison feature provides a structured framework for understanding temporal variations across languages. By combining these elements, the system serves as a comprehensive tool for both translation and cross-linguistic grammatical studies.

3.3 End-to-End Translation Workflow

This system flow is organized into six distinct stages, illustrating the complete process from user interaction to the handling of responses from an external API. The frontend (React) and backend (Django) have clearly defined responsibilities. The external API, which supports translation and language processing, is integrated as an auxiliary service within the backend.

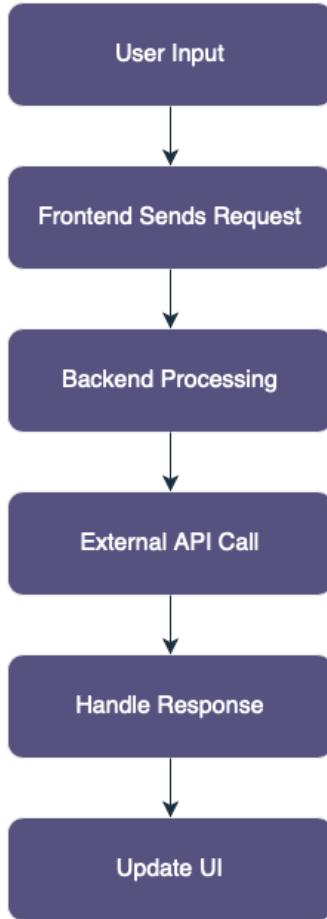


Figure 7. Translation request workflow

Step 1: User Input

The user starts by entering a sentence into a React-based interface and selecting the desired target language (e.g., French, Spanish). This marks the beginning of the translation request.

Step 2: Frontend Request

The frontend constructs a JSON object containing the input text, target language code, and the translation mode (either sequential or batch), and sends a POST request to the `/api/translate/` endpoint.

Step 3: Backend Processing

The Django backend receives the request, performs parameter validation, and prepares a query for the external API. This logic resides within the view function, which includes input validation and branching logic based on the selected translation mode.

Step 4: External API Call

The backend sends a GET request to the MyMemory Translation API, including the input query and the language pair. The API responds with the translated sentence and related metadata.

Step 5: Response Handling

The backend extracts the translated text from the API response, formats it into a structured JSON object, and returns it to the frontend.

Step 6: UI Update

Upon receiving the response, the React frontend updates its state and dynamically displays the translated result, along with additional linguistic data such as definitions and synonyms for selected words.

This architecture follows a modular and layered design, promoting scalability, maintainability, and ease of testing. All external API interactions are handled exclusively by the backend, while the frontend focuses solely on presenting the structured results.

3.4 Django Directory Structure

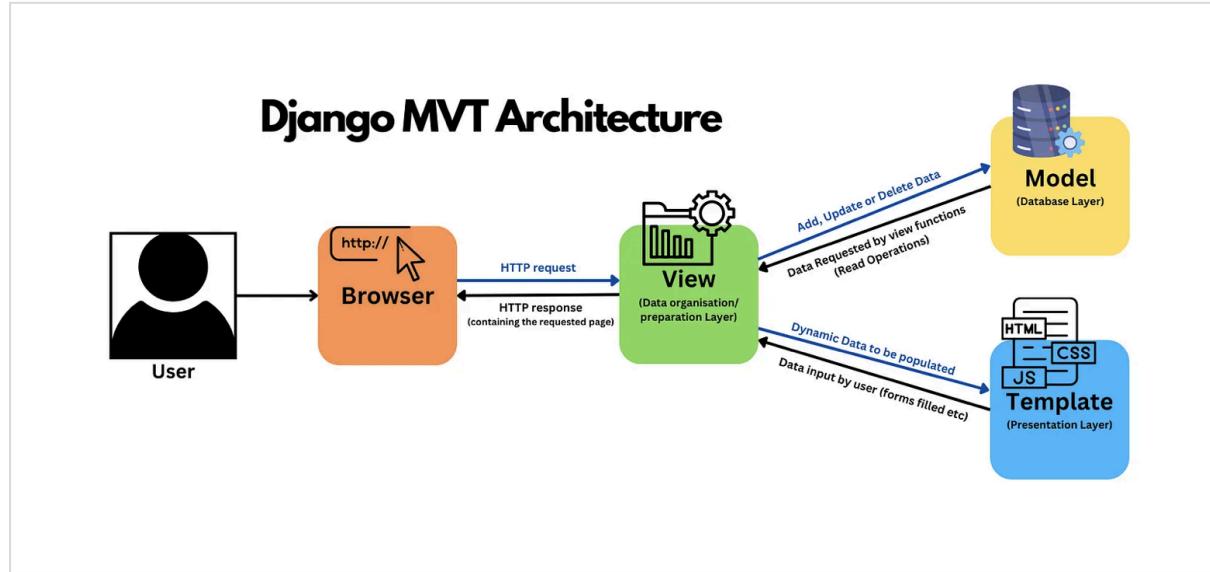


Figure 8. Django MVT architecture (Image source: Adapted from CodeMaple, n.d.)

The adoption of Django's project structure in my project aims to maximize modularity, scalability, and maintainability. Django follows the MVT (Model-View-Template) architecture, and effectively managing this architecture requires the separation of functionalities into distinct directories and files. The structured approach outlined below enhances code readability, enforces a clear separation of concerns, and optimizes development efficiency:

1. Project Root/	- Project root directory
a. config/	<ul style="list-style-type: none"> i. init.py ii. settings.py iii. urls.py iv. asgi.py v. wsgi.py
	<ul style="list-style-type: none"> - Core Django configuration and global URL routing - Marks directory as a Python package - Project-wide settings (DB, Middleware, Installed Apps, etc.) - Global URL routing for the entire project - Asynchronous server gateway interface (Used for asynchronous server, Needed for real-time functionality) - Web server gateway interface (Connects Django with web server, Handles Django's default web pages)
2. Backend/	
a. translator/	<ul style="list-style-type: none"> i. init.py ii. views.py iii. urls.py iv. models.py v. serializers.py vi. admin.py vii. apps.py
	<ul style="list-style-type: none"> - App directory (handles translation-related functionality) - Indicates this is a Python package - Handles translation logic and returns API responses - URL patterns specific to this app - Database schema/models - Converts models to JSON (DRF) - Admin panel settings for this app - App configuration

viii.	migrations/	- Tracks database schema changes
b.	tenses/	- Another app directory (handles tenses-related logic)
3.	Frontend/	
a.	static/	- Static files (CSS, JS, images, etc.)
b.	templates/	- HTML templates rendered by Django views

Figure 9. Example of Django project structure

For instance, in this structure, *views.py* is responsible for handling client requests, *models.py* manages database interactions, and *urls.py* defines the API endpoints utilized within the application, ensuring a clear separation of concerns. Moreover, *settings.py* enables centralized configuration management, allowing seamless adaptation to different deployment environments, while *migrations/* facilitates the secure tracking and application of database schema modifications. Adhering to this structured approach ensures that the project remains maintainable as it scales, preserving code consistency when integrating new functionalities or expanding existing ones. Furthermore, the use of *static/* and *templates/* directories enables the separation of static assets and HTML templates, thereby establishing a distinct boundary between frontend and backend responsibilities. Consequently, this structure is not merely a means of organizing files but a fundamental architectural strategy that leverages Django's strengths to enhance collaboration, maintainability, and scalability.

3.5 Code Structure and Responsibilities

One of the core components of the frontend architecture is a file named [Translation.jsx](#), which serves as the primary interface through which users input text and access the application's translation functionality. Due to the extensive amount of code contained within this component, a dedicated architectural diagram was created alongside a detailed line-by-line explanation in the documentation. This supplementary material facilitates a clearer understanding of the component's structure and logic at a glance. Additionally, a flowchart was developed to illustrate the overall translation process, providing a comprehensive overview of the application's operation—from user input to final output.

Lines 1-23:

The *useState* and *useEffect* hooks from the React library were used to manage component state and perform specific actions during page initialization or updates. To enhance the visual

presentation and improve clarity in information delivery, various icons—such as arrows, books, and sparkles—were imported from the *lucide-react* library. Animation effects were added using *motion* and *AnimatePresence* from the *framer-motion* library to create a more dynamic UI. For seamless navigation within the application without triggering full-page reloads, the *Link* component from *react-router-dom* was utilized.

Lines 24-88:

This section defines multiple states using the *useState* hook to store and update user input and other values dynamically.

Lines 91-135:

Language codes for translation are defined here. Stop words are also filtered out to exclude them from word-level analysis (e.g., synonyms and antonyms).

Lines 137-190:

A function is implemented to send input text to the backend for word analysis by extracting key terms. If an error occurs, mock analysis is used as a fallback to handle exceptions gracefully.

Lines 191-230:

Another API function is defined to handle single-language translation. It receives an input text and a selected language, sends the request to the backend, and displays the response on the frontend. Error handling is also included.

Lines 233-242:

A progress indicator is added here to visually show translation progress to users.

Lines 243-557:

A function is defined to handle multilingual translation in both sequential and simultaneous modes. The function splits the input text into individual lines, checks for special cases like “U.S.A.”, and selects the appropriate API to call based on the selected mode. Results are displayed upon receiving the backend response.

Lines 559-643:

A function is created to compare translations across different tenses and languages. The backend receives the input text and language list, and returns translated results accordingly.

Lines 560-718:

Logic is implemented to handle both single and multiple language selections, as well as tab-based UI (e.g., tense vs. language). Animations are applied to enhance user interaction.

Lines 719-893:

The app's title, name, and tab structure (e.g., About section) are defined with responsive design for different device types.

Lines 894-982:

Language selection logic is implemented for both single and multi-language cases. After selection, it determines whether to perform sequential or simultaneous translation.

Lines 983-1037:

A text area is created to receive user input.

Lines 1038-1104:

This section includes the UI for single-language selection and a button that triggers the corresponding action.

Lines 1105-1238:

Code is implemented to support multi-language selection and the associated action button.

Lines 1239-1297:

Displays the result of single-language translation.

Lines 1298-1436:

Shows word analysis results (e.g., synonyms, antonyms) for a single language.

Lines 1437-1538:

A progress bar is shown here during multilingual translation.

Lines 1522-1640:

Displays the multilingual translation results based on the selected mode (sequential or simultaneous).

Lines 1642-1827:

Shows word analysis results across all selected languages, including synonyms, antonyms, and other linguistic data.

Lines 1829-2009:

Accepts input text and displays a comparison table of selected languages across three specified tenses. Upon clicking a button, all comparison results are displayed.

Lines 2010-2153:

Displays the About section of the application, which provides a static overview of the app and the technologies used. This section does not perform any dynamic actions.

At the end of the file, the component is exported so it can be used in the main app and in

other components for routing and various integration needs.

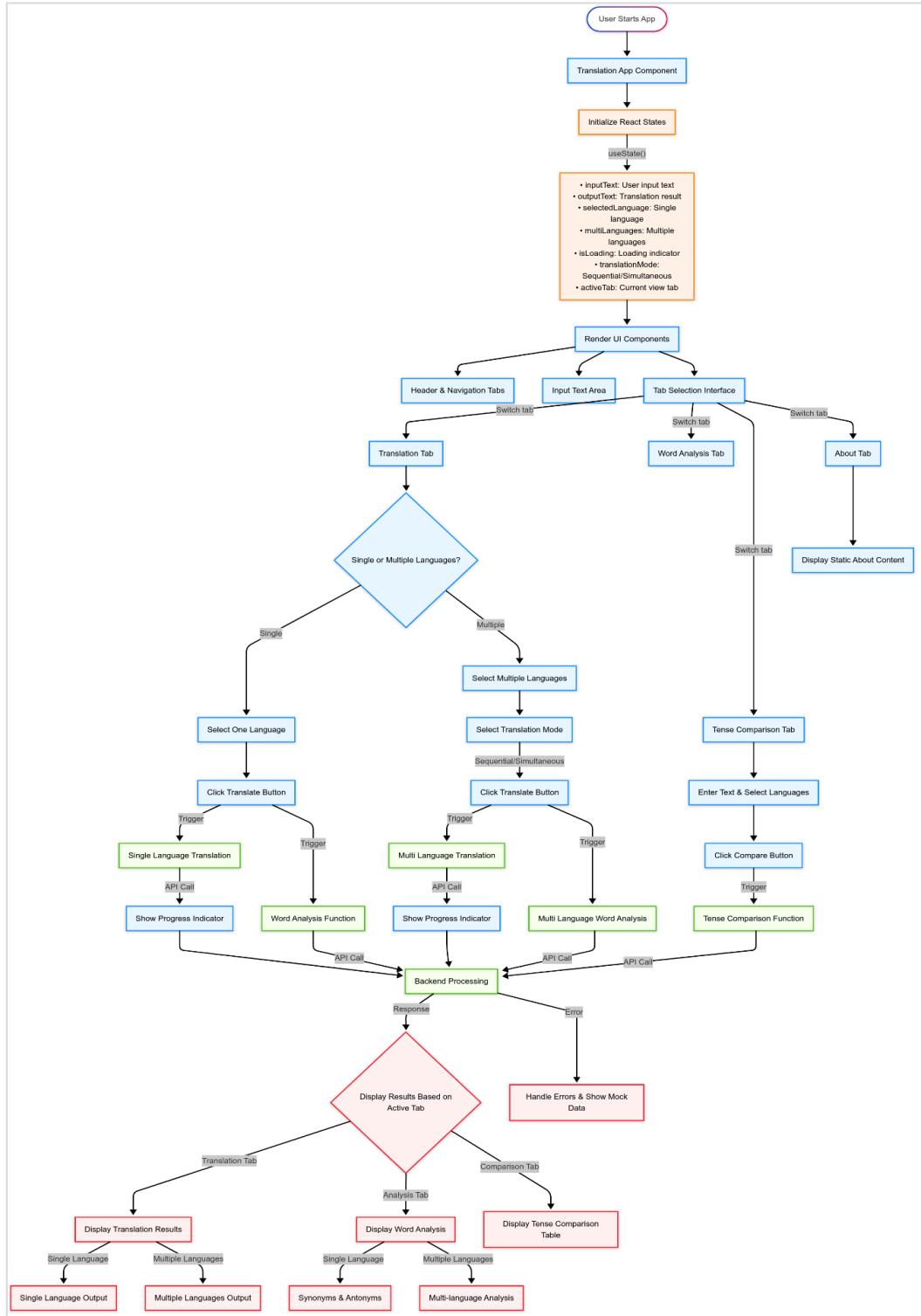


Figure 10. Architectural design of the *Translation.jsx* module in the project frontend

3.6 UI Demonstration



Figure 11. Project implemented on a local server

A screenshot of the same web application, now showing a section titled "Key Features" with six highlighted components arranged in a grid. Each component includes an icon and a brief description:

- AI-enhanced translations** for real-time high-quality translations.
- Side-by-Side Language Comparison** Compare sentence structures, tenses, and grammar across multiple languages.
- Idioms & Context Extraction** Get idioms, synonyms, and example usages directly from text.
- Save & Share Outputs**
- Mobile Responsive UI**
- Built with Django + React**

The top navigation bar and "Launch App" button are also visible.

Figure 12. Landing page presenting feature highlights and an entry point to the application

How It Works

Step 1

Input Text

Enter the text you want to translate or analyze.

Step 2

AI Processing

Our AI-integrated models analyze text to provide translations, grammar tips, and more.

Step 3

Review & Learn

Review the output, save it for later, or share with others.

What Users Say

"This app helped me understand Spanish grammar like never before! I use it every day."

- Sara K.

"Combining these translation & word analyzer in one interface is genius. I saved so much time."

- James M.

"The examples and idioms are so helpful, especially when writing essays in French."

- Michell J.

Figure 13. Landing page designed to resemble a commercial app

Frequently Asked Questions

Is this app free to use?

Yes, it's completely free for educational and personal use. You can explore all features without cost.

Which languages are supported?

The app supports over 50 languages including English, French, Korean, Spanish, Chinese, and more.

Can I save the outputs?

Yes! You can save or copy the generated translations and examples with one click.

Ready to Learn Smarter?

Dive into the world of intelligent language learning. Launch the app now and experience the future of multilingual education.

Launch App

© 2025 MultiLingo. All rights reserved.

Figure 14. Final section of the landing page featuring FAQs and app launch prompt

President Donald Trump said Thursday that he would replace Ed Martin as his nominee for U.S. Attorney for the District of Columbia after the longtime right-wing activist faced opposition from a key Republican member of the Senate Judiciary Committee.

"He is a terrific person. He wasn't getting the support," Trump said of Martin. "I'm very disappointed in that. ... I'm one person. I can only lift that little phone so many times in a day."

Select Target Language

French

Translate

Translation Result ✓

Le président Donald Trump a déclaré jeudi qu'il remplacerait Ed Martin en tant que candidat au poste de procureur américain du district de Columbia après que le militant de droite de longue date ait été confronté à l'opposition d'un membre républicain clé du Comité judiciaire du Sénat. « C'est une personne formidable. Il ne recevait pas de soutien », a déclaré Trump à propos de Martin. « Je suis très déçu de cela. ... je suis une personne. Je ne peux soulever ce petit téléphone que tant de fois par jour. »

Copy

Word Analysis 5 words

President →

Definition
An act in the past which may be used as an example to help decide the outcome of similar instances in the future

Figure 15. Single-language translation interface enabling users to enter source text, choose a target language, and send a translation request via the Translate button.

Hello, I'm from South Korea. I live in New York City. I'm a student at Brooklyn College. I'm a computer science major.

Select Target Languages (Up to 3)

French ✓ Spanish ✓ German ✓ Italian Portuguese Russian Chinese Japanese Korean Urdu

French

Spanish

German

Translate

Simultaneous Translation Results ✓

1 French 2 Spanish 3 German

Figure 16. Multi-translation interface. Users can select one or more target languages for translation and choose the translation mode (simultaneous or sequential) from the top right corner.

Simultaneous Translation Results

1 French
Bonjour, je viens de Corée du Sud. J'habite à New York. Je suis étudiante au Brooklyn College. Je suis une majeure en informatique.

2 Spanish
Hola, soy de Corea del Sur. Vivo en la ciudad de Nueva York. Soy estudiante en Brooklyn College. Soy estudiante de informática.

3 German
Hallo, ich komme aus Südkorea. Ich lebe in New York City. Ich bin Student am Brooklyn College. Ich bin Hauptfach Informatik.

Figure 17. Simultaneous translation results displaying side-by-side multilingual output for comparative analysis.

Sequential Translation Results

1 French
Hello, I'm from South Korea.
Bonjour, je viens de Corée du Sud.

I live in New York City.
J'habite à New York.

I'm a student at Brooklyn College.
Je suis étudiante au Brooklyn College.

I'm a computer science major.
Je suis une majeure en informatique.

2 Spanish
Hello, I'm from South Korea.
Hola, soy de Corea del Sur.

I live in New York City.
Vivo en la ciudad de Nueva York.

I'm a student at Brooklyn College.
Soy estudiante en Brooklyn College.

I'm a computer science major.
Soy estudiante de informática.

3 German
Hello, I'm from South Korea.
Hallo, ich komme aus Südkorea.

I live in New York City.
Ich lebe in New York City.

I'm a student at Brooklyn College.
Ich bin Student am Brooklyn College.

I'm a computer science major.
Ich bin Hauptfach Informatik.

Figure 18. Sequential translation results displaying line-by-line translations of input text across multiple target languages.

Analysis Results

President →

- Definition**: An act in the past which may be used as an example to help decide the outcome of similar instances in the future.
- Synonyms**: chair, chairman, chairperson, chairwoman, chief executive
- Idioms**: • The vast majority of presidents have been male.
- Antonyms**: No antonyms found
- Expressions**: • vice president
• president elect

Donald →

- Synonyms**: ronald, macdonald, donalda, tang, downer
- Idioms**: No idioms found
- Antonyms**: No antonyms found
- Expressions**: • john donald budge
• donald trump
• donald duck

Trump →

- Definition**: The suit, in a game of cards, that outranks all others.
- Synonyms**: best, scoop, outdo, ruff, outflank
- Idioms**: • He played an even higher trump.
• In this election, it would seem issues of national security trumped economic issues.
• He knew the hand was lost when his ace was trumped.
• Diamonds were declared trump(s).
- Antonyms**: No antonyms found
- Expressions**: • donald trump

said →

- Definition**: Mentioned earlier; aforesaid.

Figure 19. Word-level analysis of extracted terms from user input, with lexical and idiomatic information.

The screenshot shows the 'Multilingual Tense Comparison' section of the MultiLingo application. At the top, there is a search bar with the word 'analyze' and a blue 'Analyze' button. Below the search bar is a table comparing verb conjugations across four columns: English, French, and Korean. The table includes rows for Present, Past, Future, and Conditional tenses.

TENSE	ENGLISH	FRENCH	KOREAN
Present	I analyze	J'analyse	나는 분석한다
Past	I analyzed	J'ai analysé	나는 분석했다
Future	I will analyze	J'analyserai	나는 분석 할 것이다
Conditional	I would analyze	J'analyserais	나는 분석할 것이다 (가정법)

Figure 20. Tense comparison interface for cross-linguistic verb conjugation analysis.

The screenshot shows the 'About This Application' page. It begins with a brief introduction stating that the application helps language learners by providing simultaneous translation in multiple languages. It then lists key features: Single & Multi-language Translation, Word Analysis, Tense Comparison, and Simultaneous & Sequential Translation. Below this is a 'Technology Stack' section, which details the components used: Frontend (React.js with TailwindCSS), Backend (Django (Python)), AI Services (MyMemory API), and Dictionary (Free Dictionary API, Datamuse API).

Figure 21. Overview page presenting the app's purpose, core features, and technology stack

3.7 Translation Mode Architecture and Rationale

To accommodate a range of user needs and usage contexts, MultiLingo's translation API is structured around three distinct modes: Simultaneous, Sequential, and Tense Comparison. These modes are designed not only for technical efficiency but also to support pedagogical goals, highlighting MultiLingo's dual function as both a translation engine and an educational tool.

- **Simultaneous Mode:** In Simultaneous Mode, all translation requests are executed in parallel. This parallel processing significantly reduces latency, providing a fast and

responsive experience. It is particularly suitable for short texts or for users who prioritize speed. By minimizing wait time, this mode contributes to a smoother and more immediate interaction.

- **Sequential Mode:** Sequential Mode processes translation requests one by one, preserving the original sentence order. This approach is ideal for longer texts such as paragraphs or essays. It reduces the need for users to scroll back and forth, allowing them to read and comprehend the translated content in a linear, coherent flow. This improves cognitive processing and aligns with how readers naturally engage with extended texts.
- **Tense Comparison Mode:** Tense Comparison Mode is a learning-oriented feature that allows users to input a single verb and view its conjugated forms across multiple tenses in parallel. Traditional translation tools typically provide a single output per input sentence, often obscuring the underlying grammatical structure, especially for language learners. This mode addresses that limitation by making tense variation explicit.

By visually presenting how a verb changes depending on tense, the mode helps users build an intuitive understanding of conjugation patterns and temporal structure. It also supports contrastive learning, allowing learners to directly compare forms such as present, past, future, conditional, and subjunctive. This comparative approach deepens grammatical awareness and reinforces pattern recognition, which are essential for internalizing tense usage.

Furthermore, by isolating the verb from full sentence context, this mode encourages focused learning on core grammar without distractions. It enables learners to explore verb usage systematically and provides a valuable resource for self-paced study or classroom support.

4. Implementation Challenges and Solutions

During the development of the multilingual learning application, I encountered several technical challenges related to server configuration, environment compatibility, and the integration of external APIs. This section summarizes the major issues I faced and how I addressed them throughout the development process.

1) Limited Experience with Django Server Configuration

At the beginning of the project, I struggled to properly configure and launch the

Django development server, as I had limited prior experience with Django's architecture. It took time to understand how views, URL routing, and models interact within the framework. I also encountered routing errors and server crashes during early testing. To overcome these difficulties, I relied heavily on hands-on experimentation, instructional videos that provided step-by-step guidance, and occasional support from my supervisor.

2) Python Version Compatibility Issues

I also faced issues related to Python version compatibility. Some of the packages I used required Python 3.8 or higher, but my development environment initially defaulted to an older version, which caused package installation failures and runtime errors. To resolve this, I created a virtual environment with the correct Python version and used a version management tool (pyenv) to ensure consistency across devices and environments.

3) Limited Experience with AI Translation Integration

Integrating AI-based translation services, such as the MyMemory Translation API, proved to be particularly challenging. Due to my lack of prior experience with natural language processing (NLP) and handling responses from external APIs, I encountered difficulties in constructing proper HTTP requests, parsing nested JSON data, and managing error responses effectively.

Furthermore, since all development and testing were conducted entirely in a local environment without relying on paid services, it was difficult to fully simulate real-world conditions such as network latency or asynchronous communication typically encountered in production. In the absence of remote hosting platforms or automated deployment tools, I had to repeatedly configure and test the API integration manually to ensure stable performance.

These issues were gradually resolved by repeatedly revising the code structure, refactoring backend logic, and referring to supervisor guidance, online resources, and relevant tutorials found on YouTube. As a result, I was able to successfully implement a translation feature prototype that operated reliably within the local development environment.

Overall, the demanding schedule of my final semester left me with limited time to deeply engage with traditional academic literature. Instead, I relied heavily on practical advice from my supervisor, as well as more accessible learning resources such as focused online searches and instructional YouTube videos. These proved invaluable in helping me resolve

technical challenges and develop practical implementation skills efficiently.

Given the time constraints and the steep learning curve associated with API design and backend architecture, I also referred to GPT-generated example code to better understand how to structure HTTP requests and scaffold basic functions. Nevertheless, the overall workflow, core data-processing logic, and idiom-and-expression filtering mechanisms were entirely designed, implemented, and refined by myself through hands-on experimentation and iterative problem-solving.

5. Outstanding Challenges and Opportunities for Enhancement

1) Inaccuracy in Sequential Translation

The initial implementation of sequential translation caused segmentation errors, especially around punctuation marks such as quotation marks.

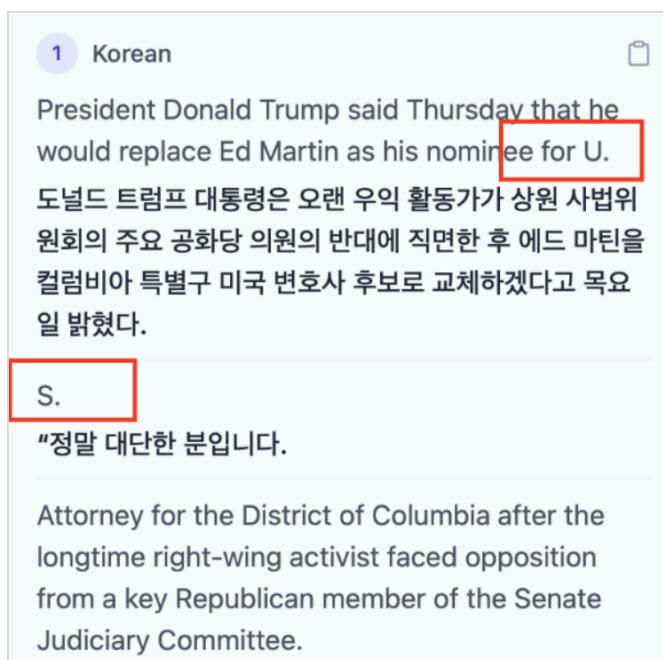


Figure 22. Misalignment previously caused by incorrect sentence splitting in sequential translation.

For example, as shown in the image above, phrases like “in the U.S.” were previously split into “in the U.” and “S.”, treating them as separate sentences. This disrupted syntactic and semantic alignment between the source text and its translations, reducing accuracy and coherence.

To resolve this issue, sentence-splitting logic was revised to recognize and preserve common abbreviations and punctuation patterns. As a result, phrases such as “in the U.S.” are now processed as complete units, significantly improving the alignment and overall translation quality.

2) Separation of Idioms and Expressions from Word-Level Vocabulary Extraction

The current system extracts idiomatic expressions and multi-word phrases from individual vocabulary items parsed from the input text. However, this implementation does not align with the original design intent. Idioms and expressions should be extracted directly and independently from the raw input text, rather than being derived from word-level analysis results.

Separating idiom extraction into an independent process would allow for faster and more accurate identification of expressions that are often difficult to infer from individual word meanings and are not easily found through standard dictionary lookups. Since idiomatic expressions typically cannot be interpreted literally, direct extraction from the full input text enables more precise context-aware language processing. Accordingly, the system architecture will be revised in future updates to isolate idiom extraction as a distinct processing stage.

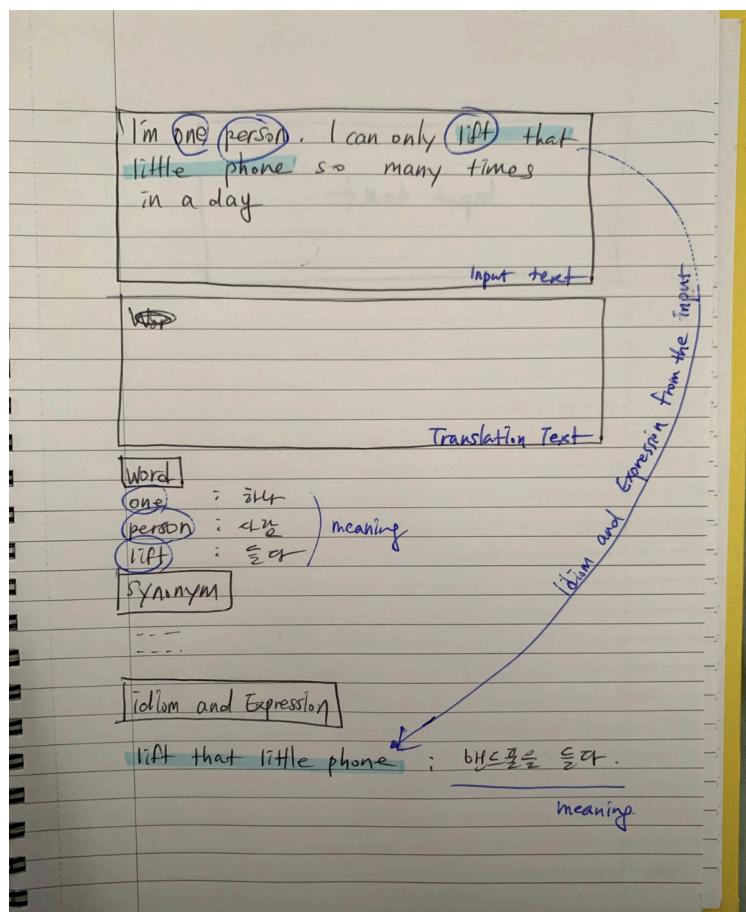


Figure 23. Concept sketch illustrating a planned update for idiom and expression extraction.

3) Long Sentence Handling Errors

The screenshot shows a 'Sequential Translation Results' interface with three panels: French, Spanish, and German. The French and Spanish panels show full sentences, while the German panel shows a truncated version of the same sentence.

French	Spanish	German
The highly choreographed and secretive process of electing a pope technically came to an end when Pope Leo XIV was introduced on Thursday. Le processus hautement chorégraphié et secret d'élection d'un pape s'est techniquement terminé lorsque le pape Léon XIV a été présenté jeudi.	The highly choreographed and secretive process of electing a pope technically came to an end when Pope Leo XIV was introduced on Thursday. El proceso altamente coreografiado y secreto de elegir a un Papa técnicamente llegó a su fin cuando el Papa León XIV fue presentado el jueves.	The highly choreographed and secretive process of electing a pope technically came to an end when Pope Leo XIV was introduced on Thursday. am Donnerstag.

Figure 24, Example of truncation in long sentence handling during sequential translation

4) Need for Expanding Tense Coverage in the Comparison Feature

The current tense comparison feature includes only four basic tenses—present, past, future, and conditional—which, while useful, do not fully represent the diversity of tense structures found in real-world language use. Languages such as French, Korean, and English utilize a wider array of forms, including the perfect, progressive, past perfect, future perfect, and subjunctive, with subtle variations across languages.

Without these additional tenses, learners may develop an incomplete understanding of verb conjugation systems and overlook critical grammatical distinctions essential for fluency. These nuanced forms are vital for accurately expressing time, aspect, and mood in different linguistic contexts.

As this functionality requires more advanced linguistic processing and technical implementation, the expansion was deferred in order to prioritize the development of the core translation engine within the given timeframe. Future updates will aim to incorporate a more sophisticated tense identification and comparison system tailored to learner proficiency and contextual nuance.

5) Folder Structure Issue Caused by Redundant Directory Naming

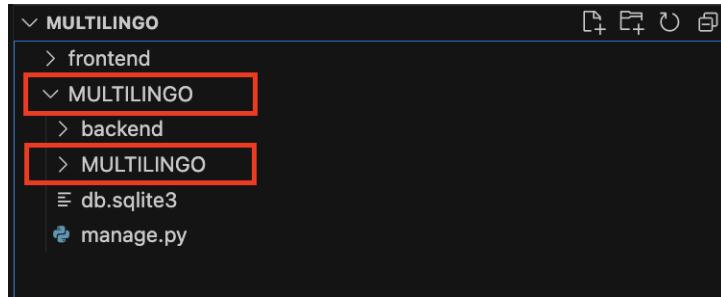


Figure 25. Project folder structure with nested directories sharing the same name

There is a structural issue in the current project folder where directories named *MULTILINGO* are nested with the same name. This occurred unintentionally during the integration process of the translation tool and the tense comparison app, which were initially developed as separate modules. While this duplication did not cause significant inconvenience during solo development, it may lead to confusion in future scenarios involving code sharing or team collaboration, as the folder structure lacks clarity.

This issue was recognized near the end of the project timeline. However, the folder structure has not yet been revised, as renaming directories at this stage could result in path-related errors or incorrect module references.

To address this, future plans include renaming directories based on their functionality—for example, using names like *multilingo_core* or *translation_app*—or restructuring the project to follow Django’s app-based architecture for clearer separation of concerns. If the folder restructuring is expected to impact the entire codebase, the plan is to postpone changes until the system is stabilized and then perform a gradual refactor in a separate branch.

In conclusion, while the current structure is maintainable in the short term, it is a critical area for improvement when considering collaboration and long-term maintainability.

6. Practical Applications

6.1 Use Cases

This application can be fundamentally utilized in three distinct ways: single-language translation, multi-language translation, and multi-tense table generation. Each function is designed to optimize linguistic accessibility and facilitate advanced multilingual text processing.

- 1) The single-language translation feature enables users to input text and select a target language for translation. For instance, if a user enters "Climate change is a pressing global issue." and selects French, the system generates "*Le changement climatique est un problème mondial urgent.*". Furthermore, users may access additional linguistic insights such as word-level translations (e.g., pressing → *pressant*, global → *mondial*) and grammatical analysis, highlighting syntactic structures and tense usage.

- 2) The multi-language translation function allows users to translate text into up to three different languages simultaneously, fostering comparative linguistic analysis. For example, if a user inputs "Technological advancements are reshaping the global economy." and selects French, Spanish, and German, the system produces:

French: "Les avancées technologiques transforment l'économie mondiale."

Spanish: "Los avances tecnológicos están transformando la economía global."

German: "Technologische Fortschritte verändern die Weltwirtschaft."

Users can further customize the translation process by choosing between two distinct modes: Simultaneous Translation, where each sentence is instantly translated into multiple languages, and Sequential Translation, where the entire text is translated into one language before being processed into the next. This structural flexibility accommodates diverse linguistic and academic needs.

- 3) The Multi-Tense Table Generation feature facilitates the comparative analysis of verb conjugations across multiple languages. When a user requests a tense comparison table for English, French, and Korean, the system generates a structured table that displays verb conjugations across various tenses. For example:

Tense	English	French	Korean
Present	<i>I analyze</i>	<i>J'analyse</i>	나는 분석한다
Past	<i>I analyzed</i>	<i>J'ai analysé</i>	나는 분석했다
Future	<i>I will analyze</i>	<i>J'analyserai</i>	나는 분석 할 것이다
Conditional	<i>I would analyze</i>	<i>J'analyserais</i>	나는 분석 할 것이다 (가정법)

This feature is especially valuable for linguists, researchers, and language learners aiming to analyze cross-linguistic grammatical structures. Additional enhancements, such as irregular verb detection and contextual usage insights, further enhance the system's analytical capabilities.

By integrating these functionalities, this application serves not only as a translation tool but also as an advanced linguistic research platform, enabling precise multilingual communication and comprehensive grammatical analysis.

6.2 Examples of Potential Applications

1) AI-Enhanced Language Learning Tool for Multilingual Education

This application serves as an advanced tool for foreign language learners and multilingual users, setting itself apart from conventional translation software by offering a more comprehensive linguistic analysis. Rather than merely providing word-for-word translations, the system facilitates real-time language processing, enabling users to develop a deeper understanding of linguistic structures. Unlike standard translation tools, this system grants instant access to word definitions, grammatical explanations, idiomatic expressions, synonyms, and antonyms, eliminating the need for manual dictionary searches and streamlining the learning process.

For instance, when a user inputs an excerpt from a news article, the system not only translates the text into the selected target language(s) but also delivers contextual vocabulary explanations, idiomatic expressions, grammatical structures, and example sentences. Additionally, by providing synonyms and antonyms, the system allows learners to expand their lexical and expressive range more efficiently. This integrated approach to language acquisition transforms passive translation into an interactive, intuitive learning experience, enabling users to master multiple languages with greater efficiency and precision.

2) Integration with Email Communication

Integrating this system into email platforms would significantly enhance the efficiency and accuracy of composing and interpreting foreign-language correspondence. When receiving an email in a foreign language, users could simply highlight unfamiliar words or phrases, prompting the system to provide not only translations but also synonyms, antonyms, and grammatical insights in real time.

Similarly, while composing emails, users seeking more natural phrasing or precise terminology could utilize the system's real-time translation suggestions and grammar refinement tools, ensuring that their writing is both linguistically accurate and contextually appropriate. By embedding this system within email clients, language barriers can be reduced, facilitating seamless multilingual communication across academic, professional, and global contexts.

3) AI-Powered Language Learning and Translation Model Development

The advancement of AI-driven language learning and translation technologies is crucial in enhancing the contextual accuracy and linguistic sophistication of machine translation systems. While AI-based translation models such as ChatGPT have demonstrated significant progress, they still struggle with contextual nuance, cultural specificity, and the accurate transmission of meaning across languages.

By leveraging this application, researchers can systematically compare real-time translations across multiple languages, detect grammatical inconsistencies, and refine natural expressions through large-scale linguistic data collection. This iterative process enhances AI training methodologies, enabling translation models to achieve higher precision, greater contextual adaptation, and more human-like linguistic output. As a result, this system contributes directly to the development of next-generation AI translation technologies, fostering a future where machine translation approaches near-human proficiency in both accuracy and contextual awareness.

7. Project Timeline

7.1 Project Management Board

The project timeline and updated deliverables will be stored on GitHub:

(Link: <https://github.com/users/srsk-sci/projects/4>)

The screenshot shows a GitHub project management board for the "MultiLingo" project. The board is organized into three columns: "Todo", "In Progress", and "Done".

- Todo Column:** Contains 11 items, all marked as "Draft".
 - Frontend Development - Basic UI Implementation
 - Designing the Primary Translation Interface
 - Core Features - Translation & Word Analysis
 - Real-time Multilingual Translation Testing
 - User Interactivity - Highlight & Instant Lookup
 - Testing & Debugging - Core Features
 - UI/UX Improvements and Usability Testing
 - Finalizing & Optimization
- In Progress Column:** Contains 4 items, all marked as "Draft".
 - System Architecture
 - Prototypes Architecture
 - Backend Development - Database & API
 - Backend Development - API Implementation & Testing
- Done Column:** Contains 2 items, both marked as "Draft".
 - Research & Learning
 - Initial Documentation Process

Figure 26. Project management board for the MultiLingo application, showing task progress across “To Do,” “In Progress,” and “Done” stages using a Kanban-style workflow.

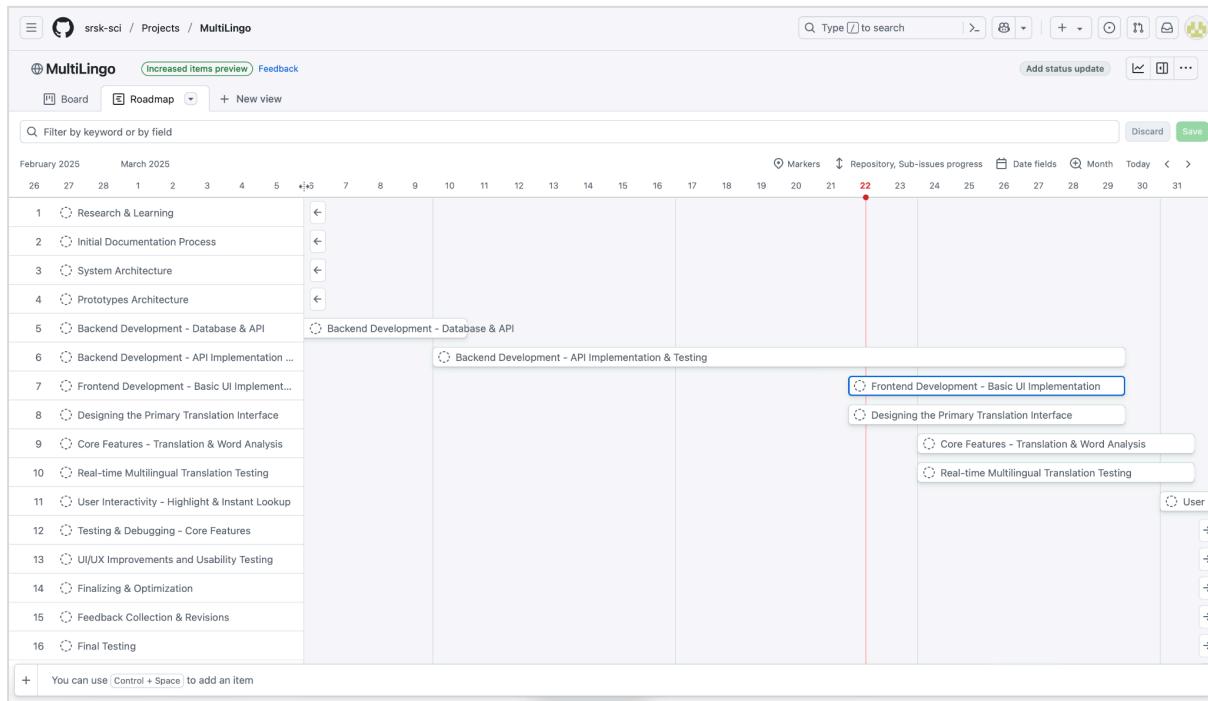


Figure 27. Development roadmap of the MultiLingo application.

7.2 Weekly Tasks and Total Hours Invested

Week	Activities	Objectives
2	Topic exploration and research	Identify potential project ideas and evaluate their feasibility through exploratory research.
3	Idea documentation and refinement	Formalize the selected idea with structured documentation and project planning.
4	Project diagram	Visualize the system architecture using diagrams to define major components and data flow.
5	Initial Django setup and learning	Establish the backend foundation using Django and begin defining scalable data architecture.
6	Django server configuration & testing	Configure and validate Django server functionality while designing UI wireframes to map user flows.
7	Django documentation, demo recording	Document backend implementation and produce a demo for early-stage feedback and validation.
8	Django error resolution, environment setup	Resolve critical server issues and configure the development environment for API integration.
9	<i>System architecture design, backend development, and UI wireframe creation</i>	Design the system's architectural framework, implement backend modules, and create UI wireframes to guide interface development.

10	<i>Backend–frontend integration, API setup</i>	Establish RESTful API communication between frontend and backend, and build interactive UI components for translation and word analysis.
11	Debugging Django–React integration	Resolve integration issues and enhance system interaction between frontend and backend components
12	Multilingual translation, UI refinement	Improve translation accuracy and polish the interface for smoother user experience.
13	AI translation pre-integration, UI optimization	Prepare UI components for AI-based translation features and enhance visual coherence.
14	Finalize integration, error resolution	Optimize system performance, resolve final bugs, and standardize UI elements.
15	Major development push, main page, bug fixing	Finalize main application features and incorporate feedback for overall quality improvement.
16	Final bug fixes, documentation	Conduct final testing, clean up the codebase, and prepare technical documentation for future scalability.

Activity Completed	Total Hours
Supervisor Discussions, Emails	8
Design	3
Coding	186
Testing & Debugging	73
Research, Training, Learning	27
Other	54
Total Hours Logged	351

8. Conclusion and Future Outlook

As international cooperation and exchange between countries continue to expand, the importance of multilingual communication skills and cross-linguistic understanding is becoming increasingly pronounced. In particular, the growing demand for second and non-English foreign languages has led to a steady rise in the number of individuals seeking to learn more than one foreign language. This trend is closely linked to the emergence of

multilingual environments across various sectors, including education, business, and cultural exchange.

Against this backdrop, the application presented in this study offers practical value by enabling language learners to effectively compare and acquire grammatical structures, expressions, and semantic nuances across multiple languages. Furthermore, its functional features suggest strong potential for future scalability and educational applications, with the prospect of broader adoption among global users.

9. Licenses and Source Attribution

This project integrates multiple open-source technologies and external APIs to support its core functionalities. The table below outlines the primary services used, their licensing terms, and their official sources.

9.1 External APIs

API	License / Policy	Source
MyMemory Translation API	Free for non-commercial use, attribution recommended	https://mymemory.translated.net
Free Dictionary API	MIT License	https://dictionaryapi.dev
Datamuse API	Free to use, non-commercial preferred	https://www.datamuse.com/api

The table above summarizes the external APIs integrated into the system to support translation and word-level linguistic features. These services—MyMemory, the Free Dictionary API, and the Datamuse API—are publicly accessible and suitable for educational or non-commercial projects. Each API has been selected for its ease of integration, free licensing model, and relevance to the system's core functionalities, such as multilingual translation, definition lookup, and semantic word expansion. The use of these APIs enables the application to provide robust translation capabilities without relying on commercial services, while maintaining legal clarity through proper attribution.

10. References

- 1) CodeMaple. (n.d.). *Understanding Django MVT Architecture and View Functions: Django Full Course for*

Beginners (Lesson 1). Medium.

<https://medium.com/@CodeMaple/understanding-django-mvt-architecture-and-view-functions-django-full-course-for-beginners-lesson-39c8da093b44>

- 2) Kardelio. (2022, August 26). *3 Different Ways to Start a HTTP Web Server from the Terminal*. DEV Community. <https://dev.to/kardelio/3-different-ways-to-start-a-http-web-server-from-the-terminal-4am3>
- 3) GeeksforGeeks. (n.d.). *How to Connect Django with ReactJS*.
<https://www.geeksforgeeks.org/how-to-connect-django-with-reactjs/>
- 4) Akhand Singh. (2022, October 12). *Building a Django and React Web Application: A Step-by-Step Guide*. Medium.
<https://medium.com/@akhandsinghofficial/building-a-django-and-react-web-application-a-step-by-step-guide-e93000692c9e>
- 5) PedroTech. (2023, March 11). *Full Stack Django + React Tutorial | Build a Full Stack Web App* [Video]. YouTube. https://www.youtube.com/watch?v=ByqKeiD_ue0
- 6) PedroTech. (2023, March 11). *Full Stack Django + React Tutorial | Build a Full Stack Web App* [Video]. YouTube. https://www.youtube.com/watch?v=ByqKeiD_ue0
(Note: Duplicate entry removed if only cited once)
- 7) DevSimplified. (2022, November 2). *Django + React.js Full Stack Web Development for Beginners* [Video]. YouTube. <https://www.youtube.com/watch?v=H2BbvHbnxo>