

**DEVELOPING MULTI TRANSLATION CHAT APPLICATION USING  
DJANGO FRAMEWORKS AND M2M100 MODEL**

by

**Muhammad Sayyid Tsabit Anfaresi**

**A Thesis**

*Submitted to the Faculty of Nanjing Xiaozhuang University  
in Partial Fulfillment of the Requirements for the degree of*

**Bachelor of Software Engineering**



School of Information Engineering

Fangshan, Nanjing

June 2023

To: Dean Zheng Hao  
School of Information Engineering

This thesis, written by Muhammad Sayyid Tsabit Anfaresi and entitled *DEVELOPING MULTI TRANSLATION CHAT APPLICATION USING DJANGO FRAMEWORKS AND M2M100 MODEL*, having been approved with respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommended its approval.

---

type the name of committee member here

---

type the name of committee member here

---

type the name of committee member here

Date of Defense: June 6, 2023

The thesis of Muhammad Sayyid Tsabit Anfaresi has been approved.

---

Dean Zheng Hao  
School of Information Engineering

---

Zhou Hong  
Chief of Foreign Affairs Office

Nanjing Xiaozhuang University, 2023

## ACKNOWLEDGMENTS

*Bismillahirrahmanirrahim*, I would like to express my sincere gratitude to God and everyone who contributed to the completion of this work.

First and foremost, I am extremely grateful to my supervisor Xiao Wenjie for her invaluable guidance, expertise, and unwavering support throughout this research. Their insightful feedback and constructive criticism greatly improved the quality of this work.

I would also like to thank the faculty at Nanjing Xiaozhuang University and Universitas Islam Indonesia for their knowledge, encouragement, and commitment to academic excellence. Their teaching and guidance played an important role in shaping my understanding of the subject. We sincerely thank all the participants who generously donated their time and shared their insights for the purpose of this study. And for all contributions contributed to the success of this research project.

I would like to thank my family and friends for their support, unwavering belief in me, and encouragement in this endeavor. Their love, understanding and encouragement are a source of strength and motivation to me.

Finally, I would like to thank for every channel in YouTube platform which I listened to for accompanying me in writing this thesis till the end.

Although I take sole responsibility for any errors or omissions in this research, I am grateful for the collective support and inspiration that has guided me on this academic journey. Thank you for your valuable contribution.

# TABLE OF CONTENTS

LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
ABSTRACT.....	1
1 INTRODUCTION .....	2
1.1 Background.....	2
1.2 Research Question .....	3
1.3 Objectives .....	3
1.4 Scope.....	3
1.5 Methodology.....	4
2 LITERATURE REVIEW .....	6
2.1 Chat App History .....	6
2.2 Hugging Face Hub .....	7
2.3 M2M100 Translation Model.....	8
2.4 Technology .....	10
2.4.1 Frontend Technologies .....	10
1 HTML.....	10
2 CSS.....	11
3 Javascript .....	11
2.4.2 Backend Technologies .....	12
1 Python.....	12
Why Chooses Python? .....	13
2 Pytorch.....	14
Why Pytorch?.....	14
3 SQLite.....	14
Why SQLite? .....	15
4 Django Frameworks .....	16

Why Chooses Django?.....	16
5 Ajax .....	17
Why Chooses Ajax? .....	18
3 SYSTEM DESIGN .....	19
3.1 Requirement Analysis .....	19
3.1.1 Functional Requirements .....	19
1 Registration.....	19
2 Profile details .....	19
3 Message notification.....	20
4 Sending messages .....	20
5 Receive messages .....	20
6 Searching and adding friends .....	20
7 Auto translation .....	20
3.1.2 Non-functional Requirements.....	20
1 Low latency .....	21
2 High availability .....	21
3 No lag .....	21
3.2 Use Case Diagram.....	21
3.3 User Interface.....	22
3.3.1 User Interface Mockup .....	23
3.4 Database Design.....	25
3.4.1 Auth user.....	26
3.4.2 Profile.....	26
3.4.3 Friends.....	27
3.4.4 Profile friends.....	27
3.4.5 Chat message .....	27
3.5 Project Structure.....	28
4 CODE AND TEST .....	29
4.1 Models Creation.....	29
4.2 Features .....	30

4.2.1	URL Configuration .....	31
4.2.2	Writing views .....	32
4.3	Features Testing .....	33
4.3.1	Sign Up new user .....	33
4.3.2	User details.....	34
4.3.3	User login.....	34
4.3.4	Translation process .....	34
5	CONCLUSIONS.....	35
	REFERENCES .....	36

## LIST OF TABLES

Table 1 Properties of Auth model.....	26
Table 2 Properties of chapp_profile model.....	26
Table 3 Properties of Friends model.....	27
Table 4 Properties of profile friends' model.....	27
Table 5 Properties of chat message model.....	27
Table 6 API List.....	31
Table 7 API List inside <i>urls.py</i> file.....	31

## LIST OF FIGURES

Figure 1 M2M100 model with language-specific parameters .....	9
Figure 2 SQLite's architecture .....	15
Figure 3 Django framework logo.....	16
Figure 4 Use case diagram of multilingual chat app.....	22
Figure 5 Registration.....	23
Figure 6 Details page .....	23
Figure 7 Login page .....	24
Figure 8 User section .....	24
Figure 9 Chat section .....	25
Figure 10 Database design for chat app.....	25
Figure 11 Project structure.....	28
Figure 12 Model creation for Profile .....	29
Figure 13 Project setting for models' creation with Django.....	30
Figure 14 Database created by Django model .....	30
Figure 15 url.py setting on root folder .....	32
Figure 16 Example of views in Django .....	33



## ABSTRACT

The significance of language as a powerful communication tool and its evolution in the context of modern communication platforms. While chat applications have provided convenient means for global interactions, language barriers and misinterpretations often arise due to the difficulty in capturing the intended meaning. Although translation technologies like Google, Yandex, and Bing exist, constantly switching between chat and translation applications can hamper user productivity. To address this limitation, some chat applications integrate translation functions, enabling users to translate texts without leaving the application.

However, these functions have certain limitations, such as the need to select and translate specific chats and the inability to translate emoticons or stickers. To overcome these limitations, the development of a web-based chat application capable of performing direct translation. By using the utilization of the M2M100 NLP model in chat application development.

The M2M100 model stands out by offering translation capabilities for a wide range of languages, surpassing other translation APIs such as those from Google and Microsoft which require subscription costs. Being an open-source model, M2M100 proves to be a favorable choice for initiating the development of the chat application. Despite the longer process involved, the model's results are deemed satisfactory

.

# 1 INTRODUCTION

## 1.1 Background

Language is the most reliable and powerful communication tool for conveying information in society[1]. Man uses language in all aspects of his life. Language has become an important part of human life. Along with the time, communication change from face-to-face into several media with smartphone platforms including web platforms. With many chats application giving advantage to a person can interact and communicate from local and long distances only by using the internet network[2], [3]. However, in the global context of information delivery, there are often misinterpretations of meaning because the language used is sometimes difficult for a country to capture. Not everyone has the opportunity or ability to learn a new language.

The problem is aided by the presence of translation technology such as *Google*, *Yandex*, and *Bing*. By *copying* and *pasting* the translation platform, we can quickly obtain translation results. However, this will greatly consume user productivity because they have to exchange chat and translation applications. Therefore, some chat applications implement additional functions to translate text, such as those found in *WeChat* applications[2].

This additional function is very useful for users to directly translate chat texts from the application, so that they do not need to leave the application to translate languages from other countries. Although this function is very useful, we need to press the chat we want to translate first, and then use the translation function. Another limitation is that the translation process stops when other users send emoticons or stickers. Create the text after it fails to translate.

For this reason, we need to develop a chat application that can perform the translation process directly. This thesis describes the process of developing web-based chat applications. The writing of the thesis also applies as a graduation requirement for bachelor's degree in software engineering.

## 1.2 Research Question

The formulation of the problem in this study is as follows:

1. What do we need to build a chat application?
2. What are the functions required to build a chat application?
3. How does the translation process on the application take place?

## 1.3 Objectives

The objectives of this study are as follows:

1. To identify what is needed to build and develop a chat app.
2. Create a chat app with a special function to automatically translate the text.
3. To understand how the translation process occurs inside an application.

## 1.4 Scope

The present study aims to develop a web-based chat application using Django framework and Python programming language. Unlike other existing chat apps, this study did not use the translation APIs of popular search engines, such as *Google*, *Yandex* or *Bing*. Instead, the study implemented a state-of-the-art NLP model, specifically the M2M100, obtained from the Hugging Face Hub. The use of the M2M100 model is new and applied only in the development of translation and chatbot applications. Therefore, the study sought to explore the feasibility of integrating this model into a multilingual chat application. The study used a mixed approach including comprehensive literature review, system design, application development, and testing. Research has developed an NLP-based chatbot that can translate and interpret multiple languages in real time. The application uses many features such as user authentication, chat history, and chat moderation. Furthermore, the application is designed to be extensible, secure, and user-friendly.

Research results show that the M2M100 model is an effective tool for developing multilingual chat applications. The application developed in this research has achieved a high level of

accuracy and efficiency in translating and interpreting different languages. The research contributes to the development of studies on the application of NLP models in chatbot development. Search results provide useful information for developers and researchers working on similar projects.

## 1.5 Methodology

The Waterfall method was used to design the chat application. The Waterfall method was introduced by Winston Royce in 1970, adopted later by software project managers, and then developed again through teaching in every software project. Waterfall methodology is a widely used project management method with a linear approach. In waterfall methodology, each stage of the workflow must be completed before moving on to the next step. While there are various types of project management methodologies, waterfalls are well suited for projects where the objectives are clearly outlined from the beginning[4].

The classical waterfall approach models start with the analysis stage which includes the analysis for requirements[4]. The model is considered offering well-defined set of criteria and the requirement indications before even starting the designing and implementation phase of the project, at the end it provides a basic plan of the project before starting in orderly sequence of the project. There are five phases of the waterfall methodology[5].

**Analysis phase.** During this phase, we outline the big picture of our project's requirements. The key aspect of the waterfall methodology is that all requirements are gathered at the beginning of the project both functional and non-functional requirements, allowing every other phase to be planned without further correspondence until the product is complete. It is assumed that all requirements can be gathered at this waterfall.

**Design phase.** The design phase of the waterfall process divided into two subphases: logical design and physical design. The logical design subphase happen, when possible, solutions are brainstormed and theorized. The physical design subphase happen, when those theoretical

ideas and schemas are made into concrete specifications. It means that the software developers and the designers are going to define the plan for a solution, and it includes algorithm design, software architecture design, logical diagram scheme, etc.

**Implementation phase.** The implementation phase happens, when programmers assimilate the requirements and specifications from the previous phases and produce actual code. This is where the real code is written and compiled into operational application, from where the database and text files were created.

**Verification phase.** This phase is also known as verification and validation, happens when the user gives reviews to the product to make sure that it meets the requirements laid out at the beginning of the waterfall project. In this phase the bugs and system glitches are found, and they are corrected. This is done after releasing the completed product to the user.

**Maintenance phase.** When the user is regularly using the product during the maintenance phase, discovering bugs, inadequate features and other errors that occurred during production. The production team must apply these fixes until the user satisfied.

## 2 LITERATURE REVIEW

### 2.1 Chat App History

The first instant messaging program was called “Talkomatic,” and it was developed in 1973 by a team of students at the University of Illinois. It was designed for use on the PLATO computer system and allowed up to five users to chat in real-time[2].

Later, in 1988, Internet Relay Chat (IRC) created allowing users to connect to networks with client software to chat with groups in real-time. its ability to support large communities of users, its flexibility and customizability, and its low bandwidth requirements[6]. It notes that IRC has been used for a variety of purposes, including online gaming, technical support, and social networking. However, there are challenges associated with using IRC, such as the potential for abuse and harassment, as well as its lack of built-in security features.

Instant messaging has evolved significantly since the 1996 release of the first widely used instant messaging program, *ICQ*. Yahoo! launched its Messenger in 1998 as Yahoo! Pager[3]. Microsoft released MSN Messenger in 1999, renaming it Windows Live Messenger in 2005[7]. Messaging platforms have changed the way people communicate, with many now preferring to use them over traditional phone calls and emails. However, there are challenges associated with their use, including privacy issues, cyberbullying and harassment, and the risk of addiction. As messaging apps continue to evolve and integrate more with other forms of communication, they will continue to play an important role in how we interact with each other.

Today, instant messaging has played a large role in bringing people together. In Asia-Pacific the rise of messaging apps has been dramatic. According to Statista, social media consumption in the region has grown dramatically, with the number of social media users expected to reach 1.4 billion by 2025<sup>1</sup>. Messaging apps have gained a strong foothold in both popularity and

---

<sup>1</sup> Social media in the Asia-Pacific region - statistics & facts. <https://www.statista.com/topics/6606/social-media->

influence across the world's most connected region, and the momentum will only continue to grow. In fact, a survey conducted by Google found that messaging apps are the most popular type of app in Asia-Pacific<sup>2</sup>.

## 2.2 Hugging Face Hub

The Hugging Face Hub is a platform with over 120000 models, 20000 datasets, and 50000 demo apps (Spaces), all open source and publicly available, in an online platform where people can easily collaborate and build ML together[8]. The Hub works as a central place where anyone can explore, experiment, collaborate and build technology with Machine Learning.

We can discover and use dozens of thousands of open-source ML models shared by the community. Inside the model repos we can find Model Cards to inform users of each model's limitation and biases. We can also include additional metadata such as their tasks, languages, and metrics. By adding an inference widget, we can play with the model directly in the browser, or for production settings, an API is provided to instantly serve the model.

Other things we can find in Hugging Face Hub are datasets. The Hub is home to over 5000 datasets in more than 100 languages that can be used for a broad range of tasks across NLP[9], Computer Vision, and Audio[8]. To make it simple to find, download, and upload datasets. The Hub is accompanied by extensive documentation in the form of Dataset Card and Dataset Preview, while many datasets are public, organizations and individuals can create private datasets to comply with license or privacy issues.

What makes Hugging Face Hub different is spaces. Spaces is a simple way to host ML demo apps on the Hub[10]. They allow user to build their ML portfolio, showcase their projects at

---

[in-asia-pacific/](#)

<sup>2</sup> Message received: What APAC's messaging app developers and marketers ....  
<https://www.thinkwithgoogle.com/intl/en-apac/consumer-insights/consumer-trends/message-received-what-apacs-messaging-app-developers-and-marketers-need-to-know-about-their-users/>.

conferences or to stakeholders, and work collaboratively with other people in the ML ecosystem.

### **2.3 M2M100 Translation Model**

M2M100 is a new many-to-many multilingual translation model that can translate between the 9900 directions of 100 languages. The underlying dataset was mined from CommonCrawl using a novel strategy which exploits language groupings to avoid mining every possible direction while maintaining good accuracy[11].

M2M100 model uses a single unified architecture and a shared vocabulary across all languages, which allows it to take advantage of transfer learning and improve translation quality for low-resource languages[12]. The model was trained on a massive dataset of parallel sentences from various sources, including web pages, books, and subtitles. The M2M100 model is available for use through the Azure Cognitive Services platform, which provides a range of machine learning and AI services for developers and businesses.

With many-to-many setting provided by M2M100: the selection of the 100 languages, the evaluation benchmark, and the construction of a large-scale training set through data mining[13] and backtranslation[14] that training data thousands of directions. M2M100 capable to create a multilingual benchmark, covering the language matrix by mining relevant parallel data, augmenting bitext data with backtranslation, and balancing languages in a many-to-many setting.

Comparing to different types of models in performance on different types of directions, likely, any language to English (To English), English to any language (From English), and all the directions not involving English (Non-English)[15]. The results of the research show that current multilingual MT systems are heavily skewed towards English, with most of the research and development resources dedicated to improving English-centric MT. This results in a lack of support for non-English languages and a bias towards English-speaking users. The study



also highlights the challenges of MT for low-resource languages, where there is limited training data and resources available for MT systems.

The proposed framework suggests a more inclusive approach to multilingual MT, where resources and research efforts are distributed more evenly across different languages, including low-resource languages. The framework also suggests incorporating the linguistic and cultural context of the languages in the MT system, to improve the accuracy and relevance of translations.

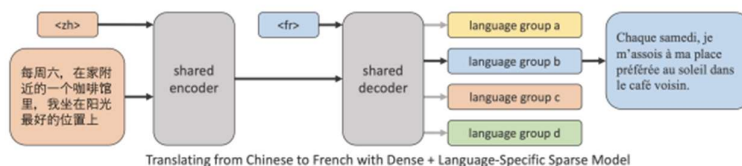


Figure 1 M2M100 model with language-specific parameters

Here's how it works:

- 1) The input text is first tokenized and embedded into a sequence of vectors. The encoder then processes this sequence of vectors, producing a set of hidden states that capture the meaning of the input text.
- 2) The attention mechanism helps the decoder focus on the most relevant parts of the input text when generating the output translation. This is especially important when translating long sentences or text that contains complex structures.
- 3) The decoder then uses the hidden states and attention weights to generate a sequence of output tokens, which are then converted into the final translated text.
- 4) The M2M100 model also uses a shared vocabulary across all languages, which helps it take advantage of transfer learning and improve translation quality for low-resource languages.

Overall, the M2M100 model is a powerful and flexible machine translation model that can handle a wide range of languages and translation tasks.

## 2.4 Technology

The architecture of the application consists of the back end and the front end, both of them having their own set dependencies (libraries and frameworks). The front end is the presentation layer that the end user sees when they enter the site. The back end provides all the data and part of the logic and it is running behind the scenes.

### 2.4.1 Frontend Technologies

#### *1 HTML*

HTML, an acronym for Hyper Text Markup Language, emerged as a programming language in 1980 and was then standardized in 1995[16]. Since then, it has become one of the most widely used languages. in the world, especially in the development of websites and the Web. websites. HTML is primarily focused on defining the structure of web documents, including things like headings, paragraphs, and images[17]. Considered the fundamental building block of a programming language, HTML serves as an essential foundation for developers around the world. Its importance goes beyond its stand-alone use, as it is often combined with other programming languages such as JavaScript and CSS to enhance display properties.

The syntax of HTML revolves around the use of opening and closing tags, indicated by curly braces (<html></html>). These tags include specific elements in the web document, allowing developers to specify their structural features. By using these tags, developers define the site's hierarchy, creating a logical framework that defines how content is presented and organized[18]. Additionally, HTML files have the .html file extension, which indicates their dedicated format.

The popularity of HTML as a platform programming language underscores its pivotal role in shaping the digital landscape[18]. Its structural capabilities and integration with other languages contributed to its enduring popularity and widespread adoption by developers. Therefore, HTML is an essential tool for creating organized and cohesive web documents, facilitating effective communication of information and engaging user experience.

## 2 CSS

CSS, which stands for Cascading Style Sheets, represents an evolved version of HTML and provides a more user-friendly approach to HTML. CSS is used to improve the presentation of web pages, including design aspects, colors, layout, and fonts, among other things. It integrates seamlessly with HTML, ensuring compatibility between the two[16]. The structure of CSS follows a defined set of rules that include selectors, properties, declaration blocks, and corresponding values. It allows developers to specify which elements of the website should be styled and define the desired look and feel. In the field of web design, CSS proves to be an indispensable tool, as it helps to separate content and presentation. This separation promotes modularity and flexibility, allowing web designers to change the visual aspects of a website without altering its underlying content structure. CSS allows designers to create visually appealing and cohesive web pages by applying style rules evenly across multiple pages. In addition, CSS supports three methods for embedding style sheets in HTML code[17]:

External CSS, Internal CSS, and Inline CSS. The external CSS method involves linking an external CSS file to an HTML document, providing a centralized location for style definitions. The internal CSS method allows styling definitions to be embedded within the HTML document itself, making it suitable for smaller scale projects. Finally, the inline CSS approach involves directly applying style rules to specific HTML elements.

By using CSS, web designers and developers can have greater control over the visual aspects of a website, ensuring a consistent and engaging user experience[7]. Its file extension, .css, stands for CSS stylesheet-specific format. With its versatility, compatibility, and diverse implementation options, CSS remains an essential component of modern web development and design practices.

## 3 Javascript

JavaScript, originally developed by Brendan Eich in 1995, has become a widely used general-purpose programming language for web development[16]. JavaScript, often abbreviated as JS, acts as a client-side scripting language, enabling dynamic and interactive functionality in web

pages. It complements HTML and CSS by adding behavioral aspects to the web page, improving user experience and facilitating responsive interactions.

As a programming language, JavaScript provides many functions, allowing developers to manipulate and control various elements of a web page[16], [17]. JavaScript allows handling of user events, such as mouse clicks and keyboard input, facilitating interactivity and responsiveness. It can automatically modify the content and structure of web pages, allowing to add, remove or modify elements seamlessly without requiring a full page reload.

JavaScript also makes it simple to incorporate third-party libraries and APIs, giving access to a wide range of extra features and resources[16]. It is suitable for creating web applications that require real-time updates or communication with databases because it enables data to be retrieved and sent to servers.

JavaScript allows programmers to create modular, reusable code thanks to its syntax, which is made up of functions, variables, and objects. It supports functional programming paradigms and object-oriented programming models, allowing for flexibility and extensibility in the way that code is organized[17].

## **2.4.2 Backend Technologies**

### *1 Python*

Python is a high-level, interpreted, and general-purpose programming language that emphasizes readability and simplicity[19]. It supports multiple programming paradigms, such as object-oriented, procedural, functional, and imperative. Python has a large and comprehensive standard library that provides built-in modules for common tasks, such as data structures, file handling, networking, database access, and web development. Python also has a rich set of third-party libraries and frameworks that extend its functionality and enable rapid development of applications in various domains, such as data science, machine learning, web

scraping, automation, and game development[20]. Python is widely used by programmers of all levels of experience and backgrounds, due to its ease of use, versatility, and expressiveness[21].

### **Why Choose Python?**

Python is easy to learn and use. It has a simple and expressive syntax that resembles natural language. It does not require semicolons or curly braces to define blocks of code. It also supports multiple programming paradigms, such as procedural, object-oriented, functional, and imperative.

Python is free and open source. Anyone can download, modify, and distribute the source code of Python without any restrictions. Python also has a large and active community of developers who contribute to its improvement and maintenance.

Python has a rich set of libraries and frameworks that provide ready-made solutions for various tasks and domains[21]. For example, NumPy and Pandas for data analysis, Django and Flask for web development, TensorFlow and PyTorch for machine learning, PyGame and Tkinter for GUI development, etc.

Python is portable and cross-platform. It can run on different operating systems, such as Windows, Linux, Mac OS, etc., without requiring any changes in the code. Python also supports multiple implementations, such as CPython, Jython, IronPython, PyPy, etc., that can integrate with other languages and platforms[21].

Python is interpreted and dynamically typed. This means that Python code is executed line by line at run time, without needing prior compilation. This makes Python code easier to debug and test. It also means that Python can handle different types of data without explicit declaration[22].

## 2 *Pytorch*

PyTorch is an open-source deep learning and machine learning library developed by Facebook, Inc. provides seamless GPU utilization and a deep learning platform that delivers maximum flexibility and speed[20]. As a library, PyTorch provides an excellent means for researchers and practitioners to develop and train deep learning experiments at scale while providing a neat abstraction for several building blocks yet being flexible for deep customization. In the next few chapters, while practically implementing deep learning models, you will see how PyTorch takes care of so many things in the background and thus equips the user with the speed and required agility for accelerated experiments at scale.[22]

### **Why Pytorch?**

Pytorch provides an extremely easy to use, extend, develop, and debug framework. Because it is Pythonic, it is easy for the software engineering community to embrace. It is equally easy for researchers and developers to get tasks done. PyTorch also makes it easy for deep learning models to be productionized. It is equipped with a high-performance C++ runtime that developers can leverage for production environments while avoiding inference via Python[20].

## 3 *SQLite*

SQLite is an embedded database. Rather than running independently as a standalone process, it symbiotically coexists inside the application it serves—within its process space. Its code is intertwined, or embedded, as a part of the program that hosts it. To an outside observer, it would never be apparent that such a program had a relational database management system (RDBMS) on board.[23]

SQLite architecture consists of eight separate modules grouped with three major subsystems. These modules divide query processing into discrete tasks that work like an assembly line. The interface is the first layer of the SQLite architecture. Then there is the compiler, where the compilation process starts with the tokenizer and parser. Where SQLite's tokenizer is hardcoded. Its parsers generated by SQLite's custom parser generator, which is called Lemon.

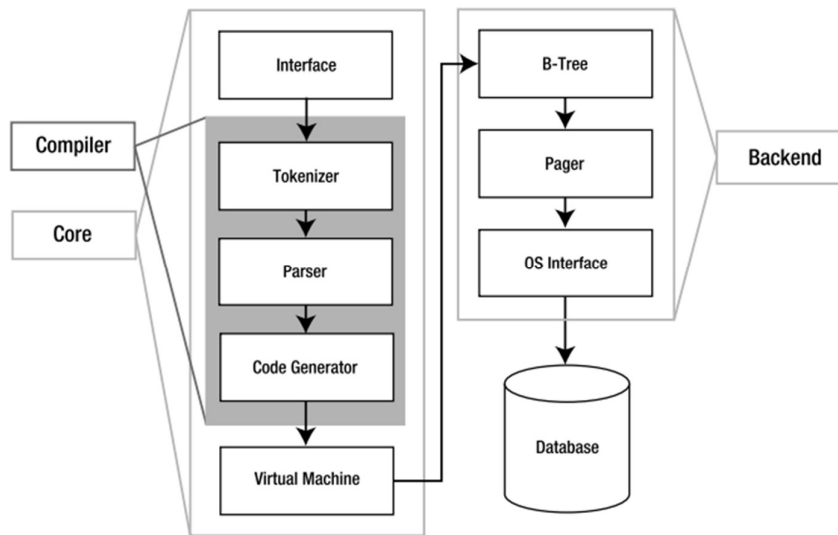


Figure 2 SQLite's architecture

### Why SQLite?

SQLite comes with some advantages features and capabilities despite its small size. From its initial conception, SQLite has a simple configuration so developer don't have to setting up outside. SQLite does not need to be "installed" before it can be used[23]. There is no "configuration" procedure. There is no need to start, stop or configure any server processes. Administrators do not need to create new database instances or assign access rights to users. SQLite does not use any configuration files. There is no need to do anything to tell the system that SQLite is running. No action is required to recover from a system crash or power failure. There is nothing to fix the problem.

Another reason is Django framework supported SQLite for the model creation and API. SQLite is easy to integrate into chat applications due to its simple API and support for various programming languages[24]. It provides a seamless connection between the app and the database, allowing efficient storage and retrieval of chat messages and user information.

Because it is a lightweight and embedded database engine, eliminating the need for a separate database server and reducing resource consumption. This simplifies integration into chat

applications and ensures efficient storage and retrieval of chat messages and user information. SQLite is cost-effective as an open-source database, making it an ideal choice for smaller-scale projects or startups with limited resources. Additionally, it provides built-in encryption capabilities, ensuring data security and confidentiality of chat conversations.

Lastly, SQLite's cross-platform compatibility allows chat apps to run seamlessly on various operating systems, enabling wider reach and accessibility for users. Overall, these advantages make SQLite a favorable option for developing chat apps that require efficient performance, data security, and cost-effectiveness.

#### 4 Django Frameworks



Figure 3 Django framework logo

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites[25]. It takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. Django provides full MVC Framework that includes the whole things. While Django alone could be used to make a RESTful API, it is one of the frameworks that sows fantastical creations. It features filled extension towards Django framework[24].

#### **Why Chooses Django?**

Django is a popular and powerful web framework that offers many benefits for building web applications[24], [25]. One of its strongest points is its ability to build common web application



components, such as user authentication and database integration, quickly and easily with minimal configuration. This allows developers to focus more on the specific functionality of their application, rather than the underlying infrastructure. Django also has a large and active developer community, providing access to a wealth of resources and troubleshooting and debugging support.

Another important advantage of Django is its extensibility. Django is designed to handle large and complex web applications, with features like automatic administration interfaces, object-relational mapping, and automatic caching that optimize performance and reduce load on users' server. Additionally, Django is highly modular and extensible, allowing developers to add new features and functionality as needed without having to rewrite the entire application.

Lastly, Django offers robust security features, including built-in protection against common web vulnerabilities, such as cross-site scripting (XSS) and SQL injection. This helps ensure that applications built with Django are secure and protected from potential attacks.

## 5 *Ajax*

Ajax technologies have revolutionized the way developers build web applications by delivering improved performance and interactivity[26]. With Ajax, developers can create web applications that provide desktop software functionality, including complex user interfaces and advanced capabilities. This has allowed developers to build applications that were previously impossible on the web, and as a result users can now access many advanced features and tools from the comfort of their browsing behaviors.

The use of Ajax also allows web applications to be more responsive and user-friendly, improving the overall user experience. It's no surprise that Ajax has become an essential technology in modern web application development, allowing developers to create highly interactive and responsive software that is both efficient and reliable. With the continued advancements in web development technology, it is likely that Ajax will remain the cornerstone

of web application development for many years to come.

### **Why Chooses Ajax?**

One of the main advantages of Ajax technology is its ability to create web applications that provide a more seamless and efficient user experience. Using Ajax allows the creation of web pages that can be dynamically updated without requiring a full-page refresh. This means that users can interact with web applications in real time without having to wait for the entire page to reload every time they make a request. This can greatly improve the usability of web applications, as users can quickly and easily access the information they need without annoying delays.

Another advantage of Ajax technology is its ability to enable rich and interactive user interfaces to be created. By allowing web applications to respond to user actions in real time, Ajax provides a much more engaging user experience. With the help of Ajax, developers can create web applications that provide functionality that was previously only available in desktop software. This has greatly expanded the capabilities of web applications, making them more flexible and adaptable to a wide range of user needs[26].

In addition, the use of Ajax technology can help improve the performance of web applications. By minimizing the need to refresh the entire page and optimizing data transmission, Ajax can help reduce the amount of bandwidth and server resources required to deliver web content. This can lead to faster load times and smoother performance, even for highly interactive and data-intensive applications.

---

## 3 SYSTEM DESIGN

In this section all of the requirement for the chat application are described. Also, all technologies that will be used in the development process are mentioned.

### 3.1 Requirement Analysis

Requirement analysis is a process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: *Functional* and *Non-functional requirements*[4].

#### 3.1.1 Functional Requirements

These are the requirements that the end user specifically demands as a basic facility that the system should offer[4]. All the functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the first development.

##### 1 Registration

Users must first register to use the software. Ideally, the users should authenticate their accounts via their phone number, email address or social media accounts. Since we made it from scratch user need to register with their username, first name, last name, email, and password. After that the user should click the button to go to details page. The user will be stored in the database

##### 2 Profile details

User need to fill up the detail form to complete the registration. In this part, user will need their username, picture, and their language preferences. The language will determine their chat application to receive other language as their language.

### *3 Message notification*

Users able to see the messages they receive immediately, or as soon as they go online. With that, users will be knowing who is currently online, or when was the last time they were present online or not. With this functionality user will have indication when a person on the other end is typing and display message statuses, whether they have been delivered, read, edited, or failed to deliver.

### *4 Sending messages*

User can send the messages through messages field and click the button nearby the messages field to send the messages. After that user need to wait the translation process. After the translation is done, the messages will be stored in the database.

### *5 Receive messages*

User will get the messages from the other user. The messages get from the translated chat database. When the user not open the chat, the user will receive a notification.

### *6 Searching and adding friends*

User able to find friends by typing of another user username in search bar. The username will show up after user type their name. Then, user can add them as their friends by clicking the add button.

### *7 Auto translation*

The auto translation function is cannot be turned off. This function will automatically translate to user preferred language from the detail form before. The translation will use NLP algorithm to do the job. Then the translated chat will be stored in the database.

## **3.1.2 Non-functional Requirements**

Nonfunctional requirements describe user-level requirements that are not directly related to functionality. This includes usability, reliability, performance, supportability, implementation, interface, operational, packaging, and legal requirements[4].

### *1 Low latency*

The chat application needs to have a low latency because it needs to look real-time so while user sending a message, the other person should immediately be able to see that message.

### *2 High availability*

Chat app should have high availability because the system should not go down no matter what happens.

### *3 No lag*

There should be no lag as it needs to be a real-time system where one could send and receive messages instantly.

## **3.2 Use Case Diagram**

Use Case Diagram is one of the diagrams of UML (Unified Model Language) which demonstrates the interaction of the users (called Actors) with each other within the system[4]. In other words, a use case diagram represents the activity of a system between actors, elements and their roles. The use case diagram consists of a system which is a whole scenario in which all events and flows of the different elements are interacted with each other. Actors can be users or internal elements in the system and the used cases that made connection with elements in the system that is always represented by an oval shape in Figure 4.

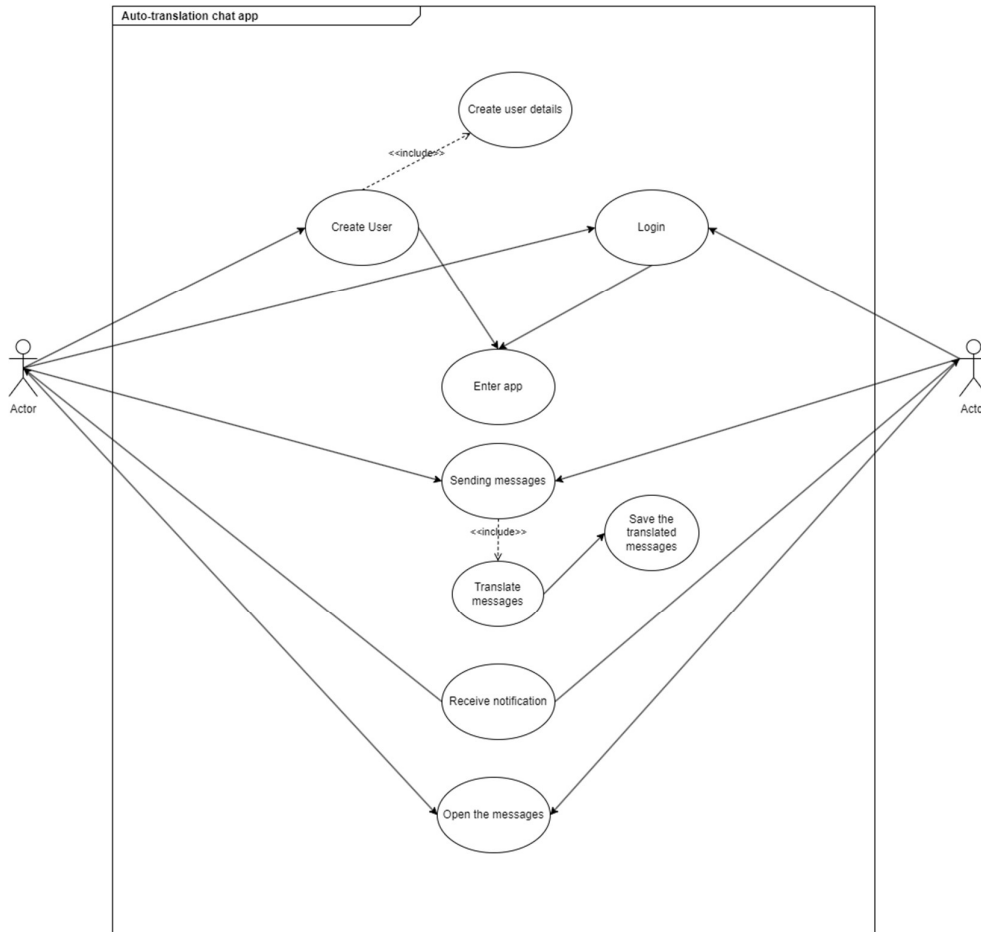


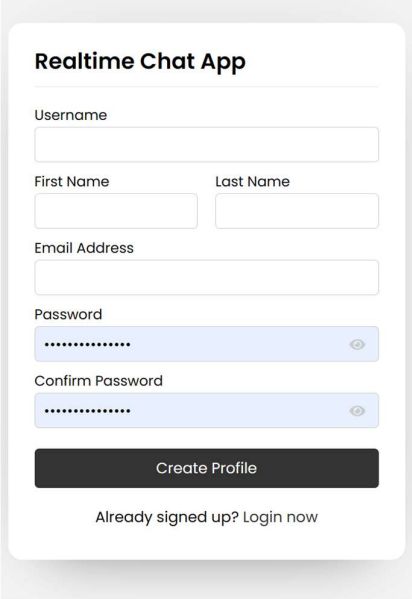
Figure 4 Use case diagram of multilingual chat app

### 3.3 User Interface

A User Interface (UI) is an integral component of any electronic device that enables users to interact with it directly[4]. The primary goal of a UI is to provide a communication bridge between the user and the system. This allows electronic devices, such as computers, tablets, and smartphones, to be operated efficiently and effectively. The UI provides users with a platform to input commands and receive feedback in a clear and understandable way. Without a well-designed UI, electronic devices would be challenging to use and operate. As such, the UI plays a vital role in enhancing the overall user experience and ensuring that users can interact with electronic devices with ease. Therefore, designing a user-friendly UI is crucial for any electronic device to succeed in the market.

### 3.3.1 User Interface Mockup

In order to create the UI design, we choose the modern approach.

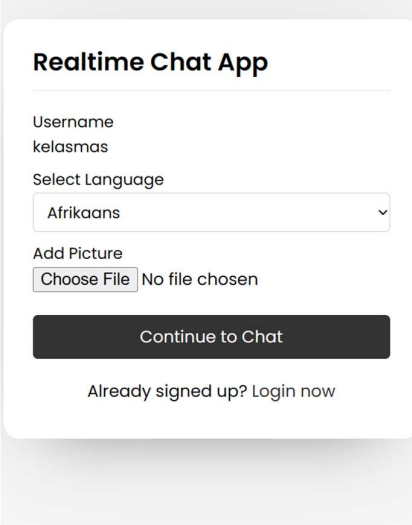


The registration form for the Realtime Chat App includes the following fields and elements:

- Username:** A single text input field.
- First Name:** A text input field.
- Last Name:** A text input field.
- Email Address:** A text input field.
- Password:** A text input field with a blue background and a toggle icon on the right.
- Confirm Password:** A text input field with a blue background and a toggle icon on the right.
- Create Profile:** A dark grey button.
- Already signed up? Login now:** A link at the bottom of the form.

Figure 5 Registration

The registration page is the page where user can create a new account, in order to use the application. This page need user to input their username, first name, last name, email address, password, and confirm password. After all the requirements full filled then user can press the button to go the next phase of registration.



The details page for the Realtime Chat App includes the following fields and elements:

- Username:** A text input field containing the value "kelasmas".
- Select Language:** A dropdown menu with "Afrikaans" selected.
- Add Picture:** A section containing a "Choose File" button and the text "No file chosen".
- Continue to Chat:** A dark grey button.
- Already signed up? Login now:** A link at the bottom of the form.

Figure 6 Details page

The details page is the page where user can manage their preference language and profile picture. This page will only show once after the user complete their registration. After the user already set their preferences then user can continue to chat app by pressing the button.

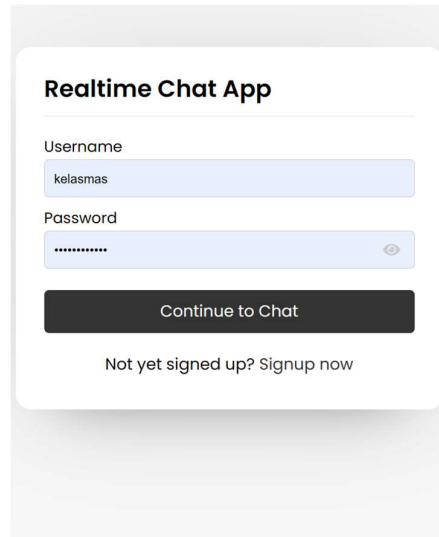


Figure 7 Login page

Other page that we create is the login page. Login page is a must for an app that use authentication. In the page only provide the user to input their username and password only. Then they can enter the chat app.

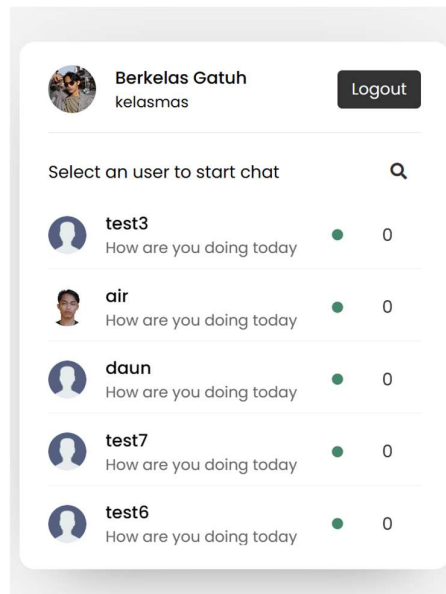


Figure 8 User section

The user section page is the main place where the information of database taking place. Here we can find the logout button, profile picture with username and their full name. We also create the search bar to find another user. This will help user to find their friends easily.



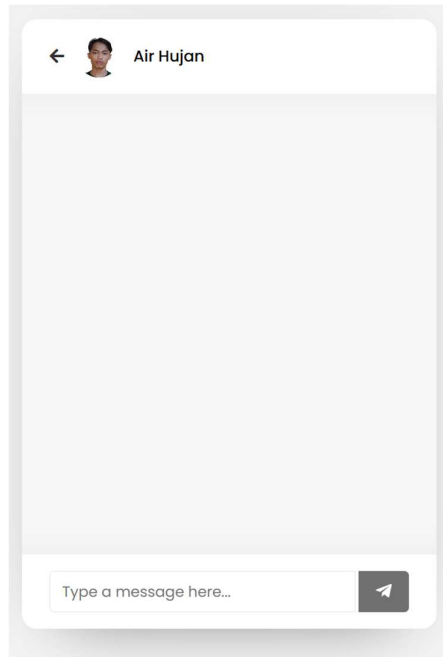


Figure 9 Chat section

Last, we created the chat section page. Here where user and their friends make a conversation each other. The chat section has the form to type the message they want to send and a button to send the message. And there is a back button for user to get back to user section page.

### 3.4 Database Design

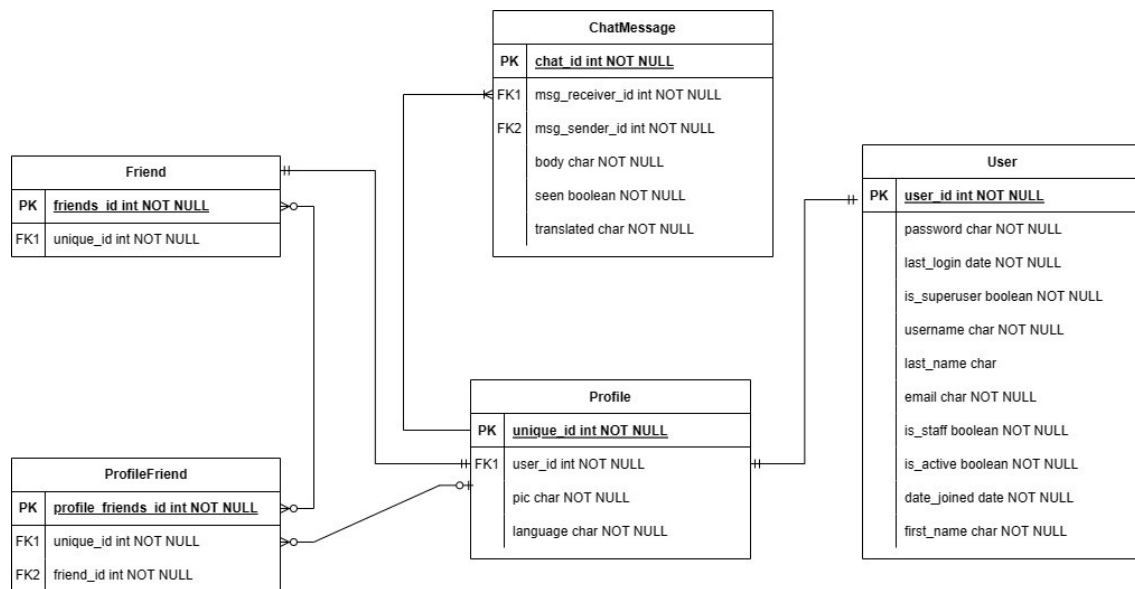


Figure 10 Database design for chat app

As shown in figure 6 of the chat application developed, it has a database design with five models. The database contains many-to-many relationship and one-to-many relationship

### 3.4.1 Auth user

The auth user model is the default model of the Django framework. It has the following properties:

NAME	TYPE
id	PK, INT
password	CHAR
last_login	DATE
is_superuser	BOOLEAN
username	CHAR
last_name	CHAR
email	CHAR
is_staff	BOOLEAN
is_active	BOOLEAN
date_joined	DATE
first_name	CHAR

Table 1 Properties of Auth model

Using this built-in model, we can use some of the functionality that Django has provided for authentication.

### 3.4.2 Profile

Model profile is the child of model auth user. We're just deriving the properties of the auth user model and adding some property attributes to the Profile Detail page. The following properties are added.

NAME	TYPE
unique_id	PK, INT
user_id	FK, INT
pic	CHAR
language	CHAR

Table 2 Properties of chapp\_profile model

### 3.4.3 Friends

The friends' model is a model for turning successfully registered users into friends by taking unique\_id from chat app profile. This model has only two properties:

NAME	TYPE
id	PK, INT
profile_id	FK, INT

Table 3 Properties of Friends model

### 3.4.4 Profile friends

This model is useful for storing friendship relationships between users. After creating a relationship, it is time we finally can chat with other users. The model properties are:

NAME	TYPE
id	PK, INT
profile_id	FK, INT
friends_id	FK, INT

Table 4 Properties of profile friends' model

### 3.4.5 Chat message

This model is a place to store messages from users. After the user sends a message into the system, the message will be translated with the configured NLP model. This model has the following properties:

NAME	TYPE
id	PK, INT
body	CHAR
seen	BOOLEAN
msg_receiver_id	FK, INT
msg_sender_id	FK, INT
translated	CHAR

Table 5 Properties of chat message model

### 3.5 Project Structure

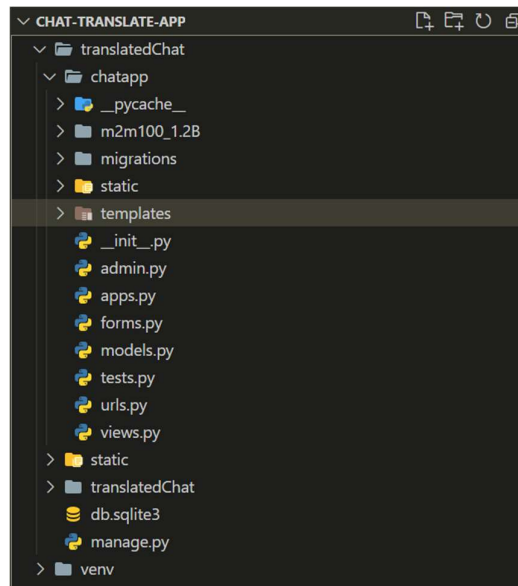


Figure 11 Project structure

In the picture above, we defining the root folder as *translatedChat* then inside the root folder we can find another main installation file from Django when we start a project which are: *chatapp* folder, *static* folder, *translatedChat* folder, *db.sqlite3*, and *manage.py*.

*Chatapp* folder is the main area where we working on our project. Inside this folder we can find our M2M100 NLP model inside the *m2m100\_1.2B* folder. The *migrations* folder is a place to save our database migration from our model file. The rest is *static* and *templates* where we create our user interface and saving CSS file.

## 4 CODE AND TEST

This chapter details the most relevant parts of the application development, decisions taken and algorithms. We have divided this chapter into three sections: models' creation, features (the most important ones) and a brief overview on how we tested our features.

### 4.1 Models Creation

A key defining aspect of any database-dependent application is its database structure. The database design can vary depending on many different factors, such as the number of reads over writes or the values that the user is likely to request the most. That is because as full stack developers we want the database to have the best performance, which can often be achieved by focusing the optimizations on the most common actions.

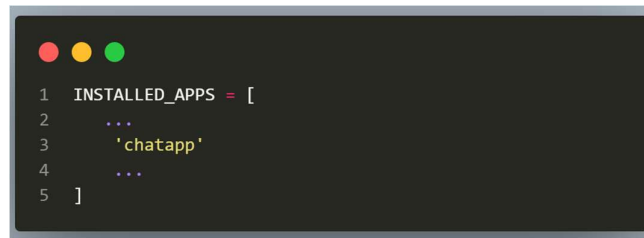
A model is the only source of accurate information about your data. It contains the essential fields and behavior of the data you store. In general, each model corresponds to a database table. With the help of the Django framework, Python model creation can be assisted with the *django.db.models.Model* class. In Django Model class, each attribute of the model represents a database field. With this, Django will give an automatically-generated database-access API.

```
1 from django.db import models
2 from django.contrib.auth.models import User
3 import random, time
4
5 from django.urls import reverse
6 # Create your models here.
7 class Profile(models.Model):
8
9     random_id = random.randrange(100000000, int(time.time()))
10    unique_id = models.IntegerField(default= random_id, primary_key=True)
11    user = models.OneToOneField(User, on_delete=models.CASCADE)
12    pic = models.ImageField(upload_to='images', blank=True, null=True)
13    language = models.CharField(max_length=3)
14    friends = models.ManyToManyField('Friend', related_name = "my_friends")
```

Figure 12 Model creation for Profile

Once we defined models, we need to tell Django that we are going to use those models. By this by editing our setting file and changing the `INSTALLED_APPS` setting to add the name of the module that contains our `models.py`.

In this app, the models for our application live in the module `chatapp.models` so we should write inside `INSTALLED_APPS` like this:



```
1 INSTALLED_APPS = [
2     ...
3     'chatapp'
4     ...
5 ]
```

Figure 13 Project setting for models' creation with Django

When we finish adding new apps to `INSTALEED_APPS`, be sure to run `manage.py migrate`, optionally making migrations for them first with `manage.py makemigrations` command. Afterward we can see the result of migration in our database editor

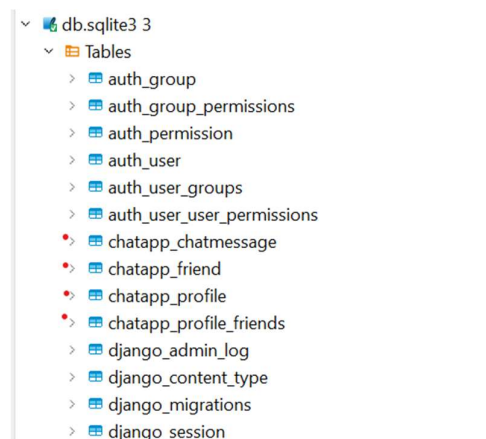


Figure 14 Database created by Django model

As we can see in the Figure 14, the model that we created will get the format like this `appname_modelname` except the model from Django which is `auth_user`.

## 4.2 Features

In building features in web-based application development, we will start by creating a list of APIs based on functional requirements that we have designed from the beginning of the chat

application development process. This API list will be very useful so that developers are able to maintain stability in the development process, so that there are not many changes.

API Name	Method
chatapp/	GET
chatapp/signup	POST
chatapp/signup/2	POST
chatapp/login	POST
chatapp/logout	GET
chatapp/user	GET
chatapp/friend/<str:pk>	GET
chatapp/sent_msg/<str:pk>	POST
chatapp/rec_msg/<str:pk>	POST
chatapp/notification	GET

Table 6 API List

#### 4.2.1 URL Configuration

After we create the API list, we need to write it inside Django. To do that, first we need to create our *urls.py* file inside our project file, at this project inside /chatapp folder, so we will create the file inside that folder.

```

1  from django.urls import path
2  from . import views
3  urlpatterns = [
4      path('chatapp/', views.chatapp, name='chatapp'),
5      path('chatapp/signup', views.signupPage, name='signup'),
6      path('chatapp/signup/2', views.detailPage, name='signup2'),
7      path('chatapp/login', views.loginPage, name='login'),
8      path('chatapp/logout', views.logoutUser, name='logout'),
9      path('chatapp/user', views.chatPage, name='user'),
10     path('chatapp/friend/<str:pk>', views.chatDetail, name='detail'),
11     path('chatapp/sent_msg/<str:pk>', views.sendMessage, name = "sent_msg"),
12     path('chatapp/rec_msg/<str:pk>', views.receivedMessage, name = "rec_msg"),
13     path('chatapp/notification', views.chatNotification, name = "notification"),
14 ]

```

Table 7 API List inside *urls.py* file

Inside the *urls.py* file, remember to import path from *django.urls* module. This file will contain every URLs that our application needs, then pass it into our main root folder. To do that we will do some configuration inside our *urls.py* on root folder.

A screenshot of a code editor window with a dark background and light-colored text. The code is Python and defines Django URL patterns. It includes imports for 'admin', 'include', 'path', 'settings', and 'static' from various Django modules. The 'urlpatterns' list contains two entries: one for the root path that includes 'chatapp.urls' and another for the 'admin/' path that includes 'admin.site.urls'.

```
1 from django.contrib import admin
2 from django.urls import include,path
3
4 from django.conf import settings
5 from django.conf.urls.static import static
6
7
8 urlpatterns = [
9     path('', include('chatapp.urls')),
10    path('admin/', admin.site.urls),
11 ]
```

Figure 15 url.py setting on root folder

#### 4.2.2 Writing views

Python functions that accept a web request and return a web response are referred to as view functions or simply view. This response could be anything at all, including the HTML code for a web page, a redirect, a 404 error, an XML file, an image, etc. Whatever arbitrary logic is required to return that result is contained within the view itself. As long as it is on your Python path, you can put this code wherever you like. There is no other prerequisite—no, sort of, "magic" As a matter of convention, views are typically placed in a file called views.py that is located in your project or application directory.

To create a view in Django. First, we open our Django project in your code editor and navigate to the app directory where we want to create your view. Create a new Python file named "views.py" in your app directory. This file contains your view functions or classes. If the file already exists, open it for editing.

At the top of your views.py file, import the required modules and classes. Normally you should import the *HttpResponse* class from the *django.http* module. This class allows you to build and return HTTP responses. Define the view function. A view function receives an *HttpRequest*



object as a parameter and returns an *HttpResponse* object. Inside your view function, you can implement the necessary logic to process user requests and generate responses. This includes retrieving data from databases, handling form submissions, rendering templates, etc.



```
1 def chatPage(request):
2     user = request.user.profile
3     friends = user.friends.all()
4     context = {"user": user, "friends": friends}
5     print(user.pic.url)
6     return render(request, 'chatapp/user.html', context)
```

Figure 16 Example of views in Django

Save the views.py file to ensure your changes are saved. After creating the view, you need to map it to a URL in Django's URL configuration. To do this, you'll need to specify a URL pattern in your app's urls.py file and map it to the appropriate view function or class. This defines the URL where the view can be accessed within the project.

### 4.3 Features Testing

In this part, test cases of the application are examined

#### 4.3.1 Sign Up new user

- Test Step  
Go to the Chat App page and sign up with required field then click “Create Profile” button.
- Expected Result  
If the registration complete, the user will go to the profile details page. If failed, there will be error notification and user will remain in the registration page.
- Actual Result  
There is no notification for user if they write wrong input, only remain in the registration page.

#### **4.3.2 User details**

- **Test Step**  
Choose the preference language for translation and input picture. After that click “Continue to Chat” button.
- **Expected Result**  
It should allow user to enter Chat App. If failed, there will be error notification and user will remain in the user detail page.
- **Actual Result**  
There is no notification for failed action, only remain in the registration page.

#### **4.3.3 User login**

- **Test Step**  
User assign their username and password to the system, then click “Continue to Chat” button.
- **Expected Result**  
It should allow user to enter Chat App. If failed, there will be error notification and user will remain in the user detail page.
- **Actual Result**  
User can enter chat app, but there is no failed notification when user put wrong input.

#### **4.3.4 Translation process**

- **Test Step**  
User sent a message to other user with their main language inside message field, then click the button or press enter. The system will do the translation before it was sent. Other user, will receive the translation result as a message.
- **Expected Result**  
New message able to translate into other user preference language and sent to another user.
- **Actual Outcome**  
Message able to translate and show to another user.

## 5 CONCLUSIONS

M2M100 is an NLP model that successfully translates as many as 100 languages in the world. By making this model for chat application development, this is certainly a differentiator with the use of translation APIs on the internet, such as Google, Microsoft, and others that require subscription costs. M2M100 is an open-source model so it is very good for the beginning of the development of this chat application. Although the process carried out by the M2M100 is longer, the results are not a problem.

With the Django framework, web development can be faster than having to code from scratch. We are assisted by the modules provided by Django, such as modeling, views, and templates. Therefore, Django is a promising choice.

However, this chat app still has many features that need to be developed again. Because the use of chat apps is not intended for translation needs only. As for the development of this application in the next stage, we will focus on some of the latest technology, improved features, and improved efficiency of the translation process.

## REFERENCES

- [1] J. S. Y. Park, “Language as pure potential,” *J Multiling Multicult Dev*, vol. 37, no. 5, pp. 453–466, Jul. 2016, doi: 10.1080/01434632.2015.1071824.
- [2] T. Barot and E. Oren, “Guide to Chat Apps,” *Columbia Journalism School*, 2015.
- [3] A. Dana, A. Zuhdi, and G. Santoso, “Dospemku Chat Application Prototype with Threading Feature using Cordova Framework for Android Based Competency Consultation,” *INTELMATICS*, vol. 3, no. 1, pp. 23–32, 2023.
- [4] S. Demirel and R. Das, “Software Requirement Analysis: Research Challenges and Technical Approaches,” in *International Symposium on Digital Forensic and Security*, Mar. 2018.
- [5] H. K. Aroral, “Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis,” *International Journal of Applied Business and Management Studies*, vol. 6, no. 1, pp. 91–99, Apr. 2021.
- [6] C. Kalt, “Internet Relay Chat: Architecture,” *The Internet Society*, 2000.
- [7] S. John, “CHAT APP WITH REACT JS AND FIREBASE,” Vaasan Ammattikorkeakoulu University of Applied Science, 2010.
- [8] Hugging Face, “Hugging Face – The AI community building the future.,” *The AI community building the future*, 2016.
- [9] R. Morgan and R. Garigl, “Hugging face - Natural language processing with Transformers,” *Endeavour*, vol. 19, no. 1. 2022.
- [10] S. M. Jain, “Hugging Face,” in *Introduction to Transformers for NLP*, 2022. doi: 10.1007/978-1-4842-8844-3\_4.
- [11] L. Xue *et al.*, “mT5: A massively multilingual pre-trained text-to-text transformer,” Oct. 2020, [Online]. Available: <http://arxiv.org/abs/2010.11934>
- [12] A. Fan *et al.*, “Beyond English-Centric Multilingual Machine Translation,” Oct. 2020, [Online]. Available: <http://arxiv.org/abs/2010.11125>
- [13] C. Raffel *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” Oct. 2019, [Online]. Available: <http://arxiv.org/abs/1910.10683>
- [14] O. Ogundepo, A. Oladipo, M. Adeyemi, K. Ogueji, and J. Lin, “AfriTeVa: Extending ‘Small Data’ Pretraining Approaches to Sequence-to-Sequence Models,” in *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, Association for Computational Linguistics, Jun. 2022, pp. 126–135. [Online]. Available: <https://data.statmt.org/cc-100/>
- [15] J. Sánchez Martínez, “Combining Multilingual Machine Translation and other NLP tasks to Learn Intermediate Language Representations (Enhancing Intermediate Representations by Jointly Learning Multilingual Translation and Part-Of-Speech Tagging),” Universitat Politècnica de Catalunya, 2021. [Online]. Available: <https://mt.cs.upc.edu/people/>
- [16] S. Junlabuddee, “Learning Web Design: A Beginner’s Guide to HTML, CSS, JavaScript, and Web.,” *Journal of Information Science*. 2020.
- [17] M. Sholikhhan, S. Kom, and M. Kom, “HTML, CSS dan Javascript,” *Penerbit Yayasan*

*Prima Agus Teknik*, 2022.

- [18] Electron, “Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS.,” *Electron*. 2017.
- [19] L. Tulchak and A. Marchuk, “History of Python.”
- [20] N. Ketkar and J. Moolayil, *Deep Learning with Python*. Apress, 2021. doi: 10.1007/978-1-4842-5364-9.
- [21] M. F. Sanner, “PYTHON: A PROGRAMMING LANGUAGE FOR SOFTWARE INTEGRATION AND DEVELOPMENT”, [Online]. Available: <http://www.python.org/doc/Comparisons.html>
- [22] K. J. Millman and M. Aivazis, “Python for scientists and engineers,” *Computing in Science and Engineering*, vol. 13, no. 2. 2011. doi: 10.1109/MCSE.2011.36.
- [23] Michael. Owens, *The Definitive Guide to SQLite*. Apress, 2006.
- [24] H. Gore *et al.*, “Django: Web development simple & fast,” *Ann Rom Soc Cell Biol*, vol. 25, no. 6, 2021.
- [25] Django Software Foundation, “Django: The Web framework for perfectionists with deadlines,” *Djangoproject.Com*, 2013.
- [26] L. Paulson, “Building Rich Web Applications with Ajax,” in *IEE Computer Society*, L. Garber, Ed., 2005.