

**VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY**



**NGUYỄN PHƯƠNG TÀI – 521H0480
BÙI ANH MINH – 521H0362**

Learn Flutter technology and build an application to support English learning

INFORMATION TECHNOLOGY PROJECT

SOFTWARE ENGINEERING

SUPERVISOR: PH.D LÊ VĂN VANG

HO CHI MINH CITY, 2025

**VIETNAM GENERAL CONFEDERATION OF LABOUR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY**



**NGUYỄN PHƯƠNG TÀI – 521H0480
BÙI ANH MINH – 521H0362**

Learn Flutter technology and build an application to support English learning

INFORMATION TECHNOLOGY PROJECT

SOFTWARE ENGINEERING

SUPERVISOR: PH.D LÊ VĂN VANG

HO CHI MINH CITY, 2025

THANK YOU

First of all, my team would like to sincerely thank Ton Duc Thang University and the Faculty of Information Technology.

The group would like to sincerely thank the Ph.D Lê Văn Vang for enthusiastically conveying his treasure of knowledge to us. Thank you for your dedication to teaching so that we can apply it to this project.

The information technology project below is the group's tireless efforts to repay the teachers' dedication. For the information technology project to be more complete, the group would like to receive comments, contributions, and criticisms from the teachers.

Finally, we would like to thank you for accompanying me throughout this topic and wish you good health and success on the path you are on.

**THE PROJECT IS COMPLETED
AT TON DUC THANG UNIVERSITY**

I hereby declare that this is the product of our project and is guided by Ph.D Lê Văn Vang. The research content and results on this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluation were collected by the author from different sources and clearly stated in the reference section.

In addition, the project also uses several comments, assessments as well as data from other authors and other organizations, all with citations and source notes.

If any fraud is discovered, I will take full responsibility for the content of my project. Ton Duc Thang University is not involved in copyright violations caused by me during the implementation process (if any).

Ho Chi Minh City, date 19 month 01 year 2025

Author

(signaturers and fullnames)

Bùi Anh Minh

Nguyễn Phương Tài

INSTRUCTOR VERIFICATION AND EVALUATION SECTION

Confirmation from the instructor

Ho Chi Minh City, date month year

Author

(signaturer and fullname)

The teacher's evaluation part marks the test

Ho Chi Minh City, date month year

Author

(signaturer and fullname)

SUMMARY

Research on Flutter language and then proceed to build a foreign language learning application

A foreign language learning support application with diverse and comprehensive functions will help users easily access and learn new languages to achieve the desired results, with significant improvements. At the same time, additional and special features will create more fun and motivation for learners, helping them maintain their learning habits and achieve their language goals.

TABLE OF CONTENT

THANK YOU	ii
SUMMARY	v
TABLE OF CONTENT	vi
LIST OF FIGURES.....	x
CHAPTER 1 – INTRODUCTION	1
1.1 Background and reasons for doing the topic.....	1
1.2 Project Objectives	2
1.3 Scope of the topic.....	2
1.4 Research methods	3
1.5 Practical significance	4
1.5.1 Scientific Significance	4
1.5.2 Practical Significance.....	4
1.6 Structure of the topic:.....	5
CHAPTER 2: LITERATURE REVIEW	6
2.1 Related works.....	6
2.2 Related technologies	6
2.2.1 Flutter	6
2.2.2 Development Features	7
2.2.3 Architecture.....	8
2.2.4 Limitations	9
2.2.5 Spring Boot Multiplatform	10
2.2.5.1. Framework Introduction	10
2.2.5.2 Development Features	11
2.2.5.3 Architecture	12

2.2.5.4 Limitations	12
2.2.6 Amazon Web Service	13
2.2.6.1 AWS Services	14
2.2.6.2 Pricing Model	16
2.2.6.3 Conclusion	17
2.3 Mobile Architecture	17
2.3.1 Overview	17
2.3.2 Data Layer	18
2.3.3 Domain Layer	18
2.3.4 Presentation Layer	19
2.4 Backend Architecture	20
2.5 Architect Solutions	24
CHAPTER 3: REQUIREMENTS	28
3.1 User Requirements	28
3.1.1 User Management	28
3.1.2 Course	32
3.1.3 Exercise	34
3.1.4 Notification	35
3.1.5 Subscription	35
3.1.6 Achievement	35
3.1.7 Community	36
3.2 Functional Requirements	36
3.2.1 User Management	36
3.2.2 Course	42
3.2.3 Exercise	45
3.2.4 Notification	47
3.2.5 Subscription	48

3.2.6 Achievement.....	49
3.2.7 Community	50
3.3 Requirement Design.....	51
CHAPTER 4: SOLUTION DESIGN.....	70
4.1 General Solution Design	70
4.2 Solution Design.....	72
4.2.1 User Service	72
4.2.2 Media Service	79
4.2.3 Course Service	81
4.2.4 Exercise Service.....	95
4.2.5 Notification & Notification Dispatcher	102
4.2.6 Subscription Service	107
4.2.7 Achievement Service	111
4.2.8 Community Service	113
4.3 Security Design	116
4.3.1 Authentication and Authorization.....	116
4.3.2 Preventive Measurements Against Attacks	117
CHAPTER 5: DEVELOPMENT AND TESTING	118
5.1 Frameworks.....	118
5.2 Development Environment	118
5.3 Project Structure.....	119
5.4 Deployment Pipeline.....	122
5.5 Implementation Result	123
CHAPTER 6: DEPLOYMENT AND EVALUATION.....	131
6.1 Testing Strategy	131
6.2 Issues Tracking	132
6.3 Testing Evaluation	132

CHAPTER 7: CONCLUSION.....	133
7.1 Achieved Result	133
7.2 Future Direction	133
REFERENCES.....	135

LIST OF FIGURES

Figure 2.1: Flutter Architecture.....	9
Figure 2.2: Spring Boot Architecture.....	12
Figure 2.3: Example Services of Amazon Web Services	14
Figure 2.4: Clean architecture diagram	20
Figure 2.5: Monolithic Architecture	21
Figure 2.6: Service-Oriented Architecture	22
Figure 2.7: Microservices Architecture	23
Figure 2.8: Solution Design	25
Table 3.1: Use cases table	53
Table 3.2: UC00 Description	54
Table 3.3: UC01 Description	55
Table 3.4: UC02 Description	55
Table 3.5: UC03 Description	56
Table 3.6: UC04 Description	56
Table 3.7: UC05 Description	57
Table 3.8: UC06 Description	57
Table 3.9: UC07 Description	58
Table 3.10: UC08 Description	58
Table 3.11: UC09 Description	59
Table 3.12: UC10 Description	59
Table 3.13: UC11 Description	60
Table 3.14: UC12 Description	60
Table 3.15: UC13 Description	61
Table 3.16: UC14 Descripton	61
Table 3.17: UC15 Description	62
Table 3.18: UC16 Description	62

Table 3.19: UC17 Description	63
Table 3.20: UC18 Description	63
Table 3.21: UC19 Description	64
Table 3.22: UC20 Description	64
Table 3.23: UC21 Description	65
Table 3.24: UC22 Description	65
Table 3.25: UC23 Description	66
Table 3.26: UC24 Description	66
Table 3.27: UC25 Description	67
Table 3.28: UC26 Description	67
Table 3.29: UC27 Description	68
Table 3.30: UC28 Description	68
Table 3.31: UC29 Description	69
Table 3.32: UC30 Description	69
Figure 4.1: Architecture Overview	72
Figure 4.2: User Service ERD Design	73
Table 4.3: users table.....	74
Table 4.4: user_achievements table	75
Table 4.5: user_otps table	75
Table 4.6: user_profiles table.....	76
Table 4.7: user_refresh_tokens table	76
Table 4.8: user_subscriptions table.....	77
Table 4.9: user_todo table	78
Figure 4.10: User Service Solution Design.....	79
Figure 4.11: Media Service ERD Design.....	79
Figure 4.12: Media table	80
Figure 4.13: Media Service Solution Design	80

Figure 4.14: Course Service ERD Design	81
Table 4.15: category_course table.....	82
Table 4.16: category table	83
Table 4.17: language_course_learning_content table	84
Table 4.18: language_course table	85
Table 4.19: expression table.....	85
Table 4.20: language_course_learning_content_expressions table	86
Table 4.21: idiom table	86
Table 4.22: language_course_learning_content_idioms table	87
Table 4.23: language_course_learning_content_phonetics table	87
Table 4.24: phonetic table	88
Table 4.25: phrasal_verb table	89
Table 4.26: language_course_learning_content_phrasal_verbs table.....	89
Table 4.27: sentence table	90
Table 4.28: language_course_learning_content_sentences table	90
Table 4.29: tense table.....	91
Table 4.30: tense_form table.....	92
Table 4.31: language_course_learning_content_tenses table	93
Table 4.32: word table.....	93
Table 4.33: language_course_learning_content_words table	94
Figure 4.34: Course Service Solution Design	94
Figure 4.35: Exercise Solution Design	95
Figure 4.36: Pronunciation Learning ERD Design	96
Table 4.37: pronunciation_assessment table.....	96
Table 4.38: pronunciation_learning table	97
Figure 4.39: Quiz Learning ERD Design.....	98
Table 4.40: quiz_learning table.....	98

Figure 4.41: Flashcard Learning ERD Design.....	99
Table 4.42: flash_card_learning table	100
Figure 4.43: Matching Learning ERD Design	101
Table 4.44: matching_learning table	101
Figure 4.45: Notification and Notification Dispatcher Solution Design	102
Figure 4.46: Notification ERD Design	103
Table 4.47: user_notification_credential table	103
Table 4.48: mail_notification_template table	104
Table 4.49: push_notification_template table	104
Table 4.50: sms_notification_template table	105
Table 4.51: notification table	106
Figure 4.52: Subscription ERD Design.....	107
Table 4.53: benefits table	107
Table 4.54: subscription table	108
Table 4.55: subscription_benefits table	109
Table 4.56: payment table	109
Figure 4.57: Subscription Service Solution Design	110
Figure 4.58: User Achievement ERD Design	111
Table 4.59: user_achievement_progress table	111
Table 4.60: achievement table.....	112
Figure 4.61: Achievement Service Solution Design	113
Figure 4.62: Community Service ERD Design.....	114
Table 4.63: chat_session table.....	114
Table 4.64: chat_session_users table	115
Table 4.65: message_user table.....	115
Figure 4.66: Community Service Solution Design	116
Figure 5.1: Mobile Project Structure.....	120

Figure 5.2: Back-end Service Architecture	121
Figure 5.3: Helper Script and Configuration	121
Figure 5.4: AWS Pipeline Runs	122
Figure 5.5: Successful Code Build in Pipeline	122
Figure 5.6: Firebase App Distribution	123
Figure 5.7: Introduction Screen.....	124
Figure 5.8: Login Page	124
Figure 5.9: Registration Page	125
Figure 5.10: Home Page.....	125
Figure 5.11: Category Course	126
Figure 5.12: Profile Page.....	126
Figure 5.13: Flashcard Learning	127
Figure 5.14: Quiz Learning	127
Figure 5.15: Matching Learning	128
Figure 5.16: Pronunciation Learning	128
Figure 5.17: Listening Paragraph.....	129
Figure 5.18: Community Page	129
Figure 5.19: Chat Session	130

CHAPTER 1 – INTRODUCTION

1.1 Background and reasons for doing the topic

In today's world, knowing a foreign language is not just an advantage but a necessity for studying, working, and communicating. The need to learn languages is growing, not only among students but also among professionals, older adults, and even children. However, traditional learning methods still have many weaknesses, making it harder for learners to improve their language skills.

With the rise of technology, online learning platforms, developing a language learning app is not just a trend but an important solution. A good app should do more than teach vocabulary and grammar; it should create a personalized learning experience that helps learners stay motivated and practice effectively.

Although many language learning apps exist, most do not fully meet learners' needs. Many only focus on vocabulary and grammar, ignoring key skills like listening, speaking, reading, and writing. A major issue is the lack of personalized learning paths, as every learner has different abilities and goals. Additionally, many apps do not provide accurate pronunciation feedback or real speaking practice, making it difficult for users to develop fluency. Without a proper system to track progress, learners struggle to understand their strengths and weaknesses. Therefore, real communication practice, and progress tracking will help solve these problems and make learning more effective.

The need for this research comes from three key factors: the growth of online education, the increasing demand for foreign language skills, and the limits of traditional learning methods. AI, big data, and voice recognition technology are changing education, making learning smarter and more interactive. As the world becomes more connected, knowing major languages like English, Chinese, Korean, and Japanese is more important than ever. However, old learning methods using books and expensive

in-person classes no longer fit today's fast-paced lifestyle. A modern, flexible, and interactive app is needed to help learners study more easily and effectively.

We chose this topic because it addresses significant challenges in language learning. By the cutting-edge technology, the proposed application provides a personalized learning journey, provides accurate feedback, and helps maintain learners' motivation. More than just a tool for language proficiency, this project plays a vital role in advancing digital education, transforming the way people learn. The creation of such an application will not only meet the growing demand for language proficiency but also drive innovation in the online education sector.

1.2 Project Objectives

This project aims to develop an online language learning app that makes learning easier, more engaging and more effective. The app will support all four skills (listening, speaking, reading and writing) and allow users to learn anytime, anywhere using only their mobile phones. Unlike traditional classrooms with fixed schedules, the app offers flexibility, allowing learners to choose their own study time. The app will also personalize lessons based on each user's level, providing progress tests and relevant learning content to help them improve effectively. One of the key features is pronunciation training, where learners can record their voice and receive immediate feedback. The app will analyze their pronunciation, score them and suggest improvements to help users speak more naturally. To make learning fun, the app will include interactive exercises, language games and daily challenges to help keep them motivated. Additionally, users can connect with a global learning community, allowing them to practice speaking with others and learn more about different cultures. By combining smart technology with engaging methods, the app aims to provide a more effective and enjoyable way to learn a new language.

1.3 Scope of the topic

Due to limitations in time and development resources, the application is currently only built on the Android operating system platform. This is an operating system with a large number of users, accounting for a high proportion of the market, especially in developing countries, where the demand for foreign language learning is increasing.

Although iOS is also an important platform, because the development requirements on this operating system are more complex and require more costs, the iOS version will be considered and developed in the next stage. When the application achieves stability and receives positive feedback from users on Android, the research team will continue to expand the product to reach iOS users, ensuring that the application can reach a larger number of learners.

1.4 Research methods

During the research and development of the online language learning application, the team used various methods to ensure the product meets users' needs. First, we conducted document research to gather information on effective language learning methods, the use of technology in education, and current market trends. This helps us identify the strengths and weaknesses of existing solutions and find ways to improve them.

Next, we carried out user surveys through questionnaires and direct interviews with students, working professionals, and other learners. The survey results help us understand user needs, study habits, and challenges, allowing us to determine the most important features for the app.

For product development, we chose the Agile model, which allows flexibility by breaking the project into short development cycles (sprints). After each sprint, we tested, evaluated, and adjusted the app based on user feedback to ensure continuous improvement. We also create prototypes to test the design and features with a small group of users before developing the final version. This helps reduce risks and improve user experience from the early stages.

Finally, we integrate speech recognition technology to personalize learning paths, support pronunciation practice, and enhance communication skills. Suggests lessons based on their level and learning speed. By combining these research, analysis, and development methods, we ensure that the app not only meets language learning needs but also provides an effective and engaging learning experience.

1.5 Practical significance

1.5.1 Scientific Significance

Traditional methods mainly focus on vocabulary and grammar but do not fully support listening, speaking, reading and writing skills with that way of learning, learners cannot improve their foreign language skills effectively. Therefore, technological improvements must appear, so to overcome these shortcomings, the application must create a smarter learning process, adapting to the needs and progress of each learner. That is also the goal we want to solve. Help learners to learn effectively.

In addition, this application can help learners to supplement new knowledge not only in language but also in culture. The application helps learners to explore interactive lessons and instant feedback, which can help people learn languages faster and more naturally. The application has completely changed the previous learning habits, especially in improving pronunciation and creating a real conversation experience for learners.

1.5.2 Practical Significance

This research has practical benefits by addressing common problems in language learning. Many learners struggle with poor pronunciation feedback, lack of practice opportunities, and courses that do not match their learning pace. The app offers instant pronunciation corrections, personalized lessons, and interactive speaking exercises, making it easier to improve language skills.

The app will also make learning more engaging and realistic by allowing learners to practice conversations in different situations. This helps build confidence and fluency, making it easier to use the language in real life.

In addition to individual learning, this research supports the larger goal of improving education through technology. It makes language learning more accessible to everyone, including those who cannot attend traditional classrooms. The app provides a flexible and convenient way to learn languages anytime, anywhere. Insights from this research can also be used to improve other online educational tools, helping more people learn more effectively in the digital age.

1.6 Structure of the topic:

- Chapter 1 – Introduction
- Chapter 2 – Literature Review
- Chapter 3 – Requirements Analysis
- Chapter 4 – System Design
- Chapter 5 – Development and Testing
- Chapter 6 – Deployment and Evaluation
- Chapter 7 – Conclusion

CHAPTER 2: LITERATURE REVIEW

2.1 Related works

Many popular language learning apps today, such as Duolingo, Babbel, Rosetta Stone, and Memrise, offer useful features but still have significant limitations. Duolingo makes learning fun with games but lacks real conversation practice. Babbel provides practical dialogues but is not very interactive and requires payment for full access. Rosetta Stone focuses on immersion through images and speech recognition but has a slow learning process and is expensive. Memrise helps with listening and pronunciation using native speaker videos but does not emphasize grammar or writing.

Despite these advantages, most existing apps do not provide effective pronunciation correction, real-time speaking practice, or personalized learning paths. Much focus mainly on vocabulary and grammar but do not fully support all four essential language skills: listening, speaking, reading, and writing. Additionally, learners often struggle with maintaining motivation due to repetitive exercises and a lack of real interaction.

Our app will solve these problems by integrating pronunciation feedback, real-time speaking practice, and a personalized learning path that adapts to each user's progress. Unlike many existing apps, we will create a realistic conversational environment where learners can interact naturally and build their speaking confidence. Additionally, our app will include engaging exercises, daily challenges, and interactive lessons to keep learners motivated and make language learning more effective and fun.

2.2 Related technologies

2.2.1 Flutter

Flutter is an open-source UI software development kit which was developed and is currently being maintained by Google. The framework can be used to develop apps on a variety of platforms, currently Flutter is most popularly known for developing apps on

Android, iOS, Windows and MacOS, besides that, it also supports the development on web platforms, Fuchsia and Linux. All of that on one singular codebase.

Flutter is also utilized in many companies with different domain backgrounds such as pure tech, e-commerce, ride-hailing, ... More specifically, popular apps from Google such as Google Pay and Google Classroom, or more day-to-day types of apps such as Grab, TP Bank and Alibaba.

2.2.2 Development Features

When it comes to the number of features that Flutter offers, there are many that are endorsed and utilized by both developers and business people alike.

First of all, it is a cross-platform framework, which means it can be used to developed on multiple operating system and devices with one code base, which in turns make it very easy for mobile developer to focus on one singular platform to develop rather having to make configuration on specific iOS or Android version to get the exact version of the UI. This is also a very good selling point for the framework to business investors since it can reduce the number of developers needed to be hired, which can reduce cost and have a more efficient work force as compared to the traditional model of having two separated teams of iOS and Android developers to create one singular product.

Secondly, the cross-platform characteristic of the framework also make it more efficient in development time for developer as they will not need to spend little to none time to set up development environment, they just need to focus on the configuration steps of specific platform rather than focus on the development of the user interface, business logic.

Another selling point of the framework is the ability to hot reload the development changes on the go without restarting the app, which means that the changes you made to the user interface or data repository or business logic will take effect immediately, which also further the efficiency of the developers.

It also has a unified user interface package that can be used by developers to have the same interface on both Android and iOS without worrying for the fact that the UI could be dysfunctional upon different devices. Another point is that the Google team has made widgets to either have Material Design or Cupertino aesthetics depending on the developer's choice of user interface kit.

2.2.3 Architecture

The Flutter's architecture is divided into three main layers, which are Framework, Engine and Embedder, and the architecture also layered in a bottom-up approach, which means the lower layer has more native and machine device code rather than abstract view that we got in the higher layer, this is to help developers not to be too rigorous in the development of the app.

- Framework Layer: This layer is responsible for providing user interface widgets, components, gestures and animation that you see on the device, this layer is entirely written in Dart itself and will provide most of the necessary packages and modules to develop the apps.

- Engine Layer: This layer is where all of the rendering of Widgets, input handling and gesture detector are received and handled by the C++ library, one noticeable library we see is the Platform Channels. In mobile or desktop development, sometimes developers need to go to the native platform for native specific features, such as camera handling, voice recording, ... This library will establish a channel with the adequate platform to the native code and will receive and send events through that channel.

- Embedder Layer: This is a platform specific code that interacts with the native operating system and compiles the application.

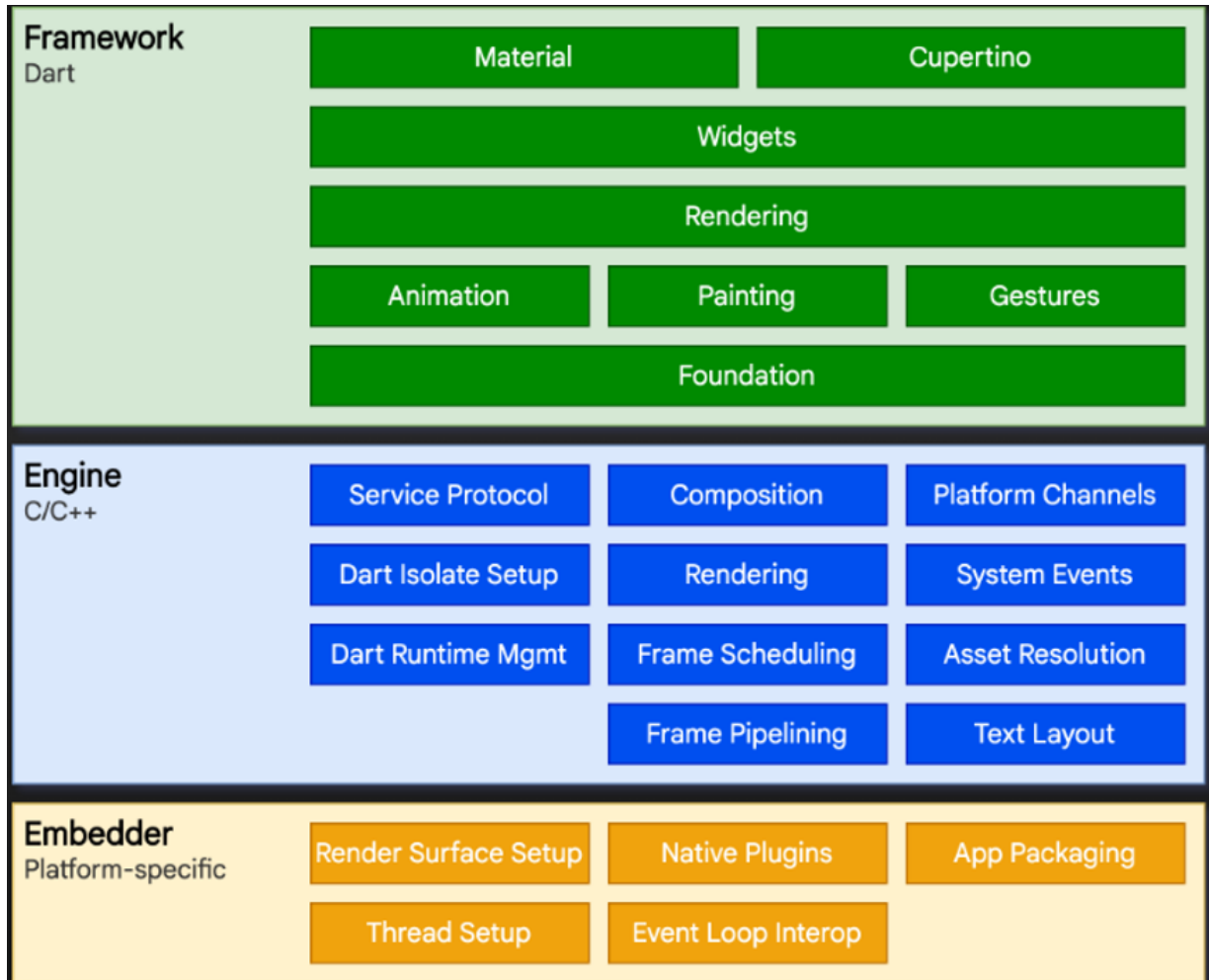


Figure 2.1: Flutter Architecture

2.2.4 Limitations

Whilst the framework has provided numerous features for developers, it is without its flaws of framework when compared to native solutions and scalability problems.

For comparison, when using a native framework such as Android SDK or iOS Swift UI and UI Kit, the solutions provided for native features are better supported and easier to implement as having to go method channel and handling the error provided by the native platform.

Besides that, the limit to your personal is comparatively huge as Dart is a programming language that is only applied to the Flutter framework, when compared to some different languages such as Java, Kotlin, or Javascript, there are much more development framework or development directions that you can take compared to Dart.

In addition, the framework is currently focusing on too many platforms, as compared to two by React Native, which can really hinder the development of the framework in favor of one platform rather than another. For now, Flutter is widely used in mobile applications for Android and iOS, with some support for Windows apps.

2.2.5 Spring Boot Multiplatform

2.2.5.1. Framework Introduction

For the longest time, Spring Boot has been the primary backend development framework for many big tech companies such as Cognizant Technology, JPMorgan Chase Bank, CGI, TymeX Vietnam, ... The list would go on and on as Spring Boot since its initial release in 2014, as a convention-over-configuration extension for the original Spring Java platform, has been one of the most popular framework due to the fact that it has a large Java developers community and provided a scalable and secure structure for large backend system. The framework can be used to build microservices, web applications and console applications.

For the Kotlin programming language, it is a statically typed, general-purpose and high-level programming language which can be used for developing apps for different types of product, as of 2025, Kotlin is the main language and fully supported in Android Mobile Development, Kotlin Multiplatform for both Android and iOS development.

Kotlin itself is designed to be interoperate with Java, which means that the a system can be written in both Java and Kotlin and it is compatible, the language utilize the Java Virtual Machine provided by the Java Development Kit, which now makes backend developers have more choices on what they are going to use for the backend development based on specific need, such as Coroutines, null safety,...

2.2.5.2 Development Features

Spring Boot combined with Kotlin has brought a lot of new and improved features compared to the traditional Java programming language. Especially by the characteristics of the Kotlin programming language such as concise syntax, null safety and coroutines.

Let's first go through the common features of Spring Boot, it is an extension on the traditional Spring Java platform, with heavy emphasis for focusing on the development process rather than having to make cumbersome configuration, enabling developers to focus on what really matters. It embeds Tomcat web server, automatically configures Spring and its third library for developers, and it also provides an optional starter dependencies project which you can find at Spring Initializer at start.spring.io. It is also an out of the box dependency injection that enhances developer performance without the need of code generation that can reduce time to develop significantly.

Now we can go over Kotlin specific characteristics and what it can bring for Spring development. Kotlin itself is an independent programming language and can co-exist within a Java based application, so it is really flexible to have a project that has two programming languages to be used based on specific needs and requirements. Another thing for Kotlin is that it reduces the amount of boilerplate code that Java produces.

The second selling point of Kotlin is that it supports asynchronous in a thread safe manner, prevents blocking when performing asynchronous function from the main thread, and makes sure that the application is smooth in the state of operation.

In Java, the language itself does not support native features for null safety, which for some developers, if they do not handle the error correctly then it can lead to exceptions such as `NullPointerException`, which can crash production server and lead to unexpected behavior on the backend side. Null safety feature actively finds nullable variables and functions so that it can warn the developers to handle the error accordingly and prevent unexpected errors.

2.2.5.3 Architecture

A standard Spring Boot application is designed with a layered architecture, just like Flutter, but the layers are more logic-based rather than programming-based than Flutter. The layers included are Controller – Service – Repository.

First is the Controller layer, this is the layer used to handle requests incoming from client applications or different services in a microservice deployment architecture. It handles the task of returning the response for the caller service or client to receive with information such as status code, response data, messages along with HTTP configuration. The incoming requests must go through a layer of authentication to determine if it is an authorized request, if applicable.

The Service layer is where all of the business logic on the backend is handled, computation, and performing tasks based on input. This layer strictly focuses on business logic handling and not receiving and responding output like the controller layer.

The Repository layer is a layer that is focused on the operation on the database of the service application, it will handle the usual database operations such as querying, inserting, deleting, modifying data inside the database. This layer does not interfere with business logic and just needs to perform database operations.

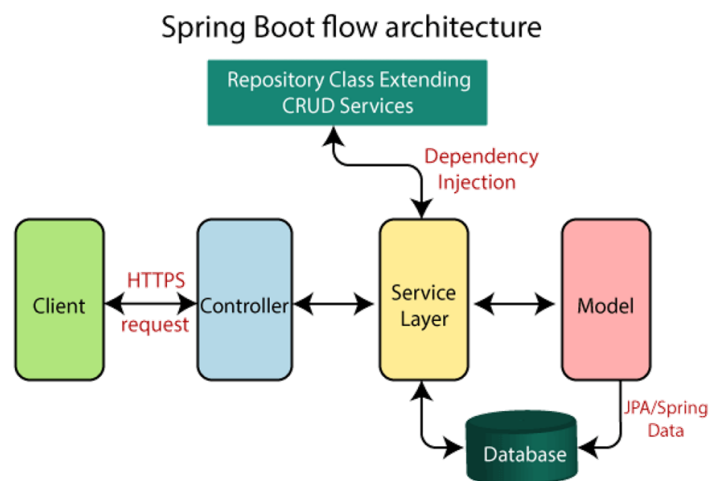


Figure 2.2: Spring Boot Architecture

2.2.5.4 Limitations

Even though Spring Boot with the use of Kotlin has provided users with a lot of conveniences for developers to use, there are still some noticeable pain points that the framework has to address with the growing market of backend development technologies.

One of the first that comes to mind is that some libraries in Spring Boot are still fully written in Java and do not provide sufficient support for the Kotlin language. Therefore, there will be instances where the library will have no effect on the Kotlin file and would require a conversion into Java.

Another point is that understanding coroutines and its specification is a cumbersome task that requires extensive knowledge and an intermediate learning curve for Java developers that want to move into Kotlin.

Finally, converting into Kotlin seems easy at first, but you would need more extensive knowledge for the language, features such as sealed class, object class, lambda function, asynchronous programming knowledge, for references, suspend function, coroutine scope, dispatcher,... to fully integrate into Kotlin.

2.2.6 Amazon Web Service

Amazon Web Service is a comprehensive and widely adopted cloud computing platform used by many big tech companies, if you name a company whose works revolve around technology then there will be a good chance that company has had or already used a service of AWS beforehand. The list of services provided by AWS, though sometimes is considered to be too complex and cumbersome to learn for entry people, has provided a comprehensive service that suits many use cases for different tech companies.

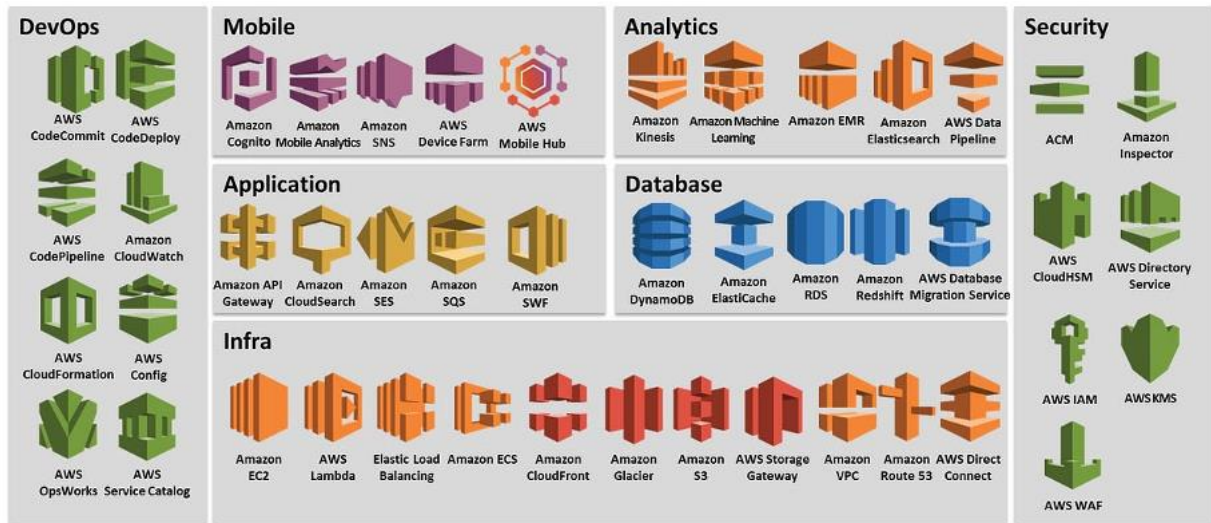


Figure 2.3: Example Services of Amazon Web Services

There are a wide range of services covered by Amazon Web Service which covers most of the aspect of an information technology system, ranging from security such as IAM and ACM, to DevOps in Pipeline, CodeCommit checking. It also provides services for features development purposes such as Amazon S3 for cloud storage of assets, database management system with Amazon RDS and SQS for queue based systems which use case can range from notification queue, message queue. Besides it also has service for deployment of microservices systems either by Amazon Elastic Kubernetes Service and Elastic Container Service, hosting Docker images on Elastic Container Registry.

In total, there are about more than 250 services on AWS that are ready to be deployed for both business and technical use cases alike, it makes AWS flexible in a sense that it can help your business or tech project in some way or another, therefore, it is really common to see AWS being used around in so many day to day action, such as data analytics, database management...

2.2.6.1 AWS Services

Since the first official launch in 2006 with the release of Amazon S3 and Amazon EC2, Amazon Web Service has significantly expanded its number of supported services,

becoming the leading player in the cloud computing and cloud provider market. The service is known for the scalability nature with the ease of adding, removing and modifying services to the suitable need of users.

In details for services provided by AWS, we can have a look through some of the more popular used services.

First is AWS EC2, which is Elastic Computer 2, this is the building block for most of the services that we use on AWS, the goal of this service is to provide a virtual server for applications to be ran on, this service can be used in combination with AWS Relational Database Service to store data, Elastic Kubernetes Service for microservice-type system deployment and configuration. You can also deploy an application on an EC2 instance with all most of the dependencies ready at the go when you initiate one.

Another option for an application deployment for users is AWS Lambda, which is a serverless cloud computing platform that developers only need to worry about the configuration of the app on server rather than having to choose the specification of server, hardware, etc...

Containerized application has received a growing trend following the release of Docker in modern development has also brought in a need for an orchestration tool, AWS also provides multiple solutions based on the specification for different purposes such as Elastic Container Service, a native Amazon solution for container orchestration and management, while Elastic Kubernetes Service is a Kubernetes-based system for container and Fargate is the serverless option of container orchestration service.

Those are the most popular services that come to mind when deploying services. For options in storage service, we have S3, a scalable object storage for developers to store assets of applications on to the cloud. Beside that, we also have Elastic Block Store which is newer and provides more features for object storing such as persistent storing, scalable in the sense of handling a large number of assets without disrupting workloads.

For database management, if you decide to go with a relational database management system, AWS provides you with AWS Relational Database Service for relational databases, such as MySQL, Postgresql,... And if you choose your preferred type of database to buy NoSQL then DynamoDB is a good option or alternative to the popular MongoDB as you can keep the ecosystem within AWS.

For security and access control of a cloud system, AWS provides you with IAM, which is a service that grants user privilege to access to a specific resource or service, or we can also use the access control mechanism to allow EC2 instances to only access specific resources.

AWS also provides a continuous integration and continuous deployment platform called AWS CodePipeline, which will handle all of the automation of code deployment for developers based on their provided script, which makes the development be more efficient. We can also use AWS Code Build for automation for app building service.

2.2.6.2 Pricing Model

AWS provide a flexible model of payment as pay-as-you-go approach, which will have some of the following characteristics:

- Ensure you only pay for what you actually use, it is more flexible than the pay-first model, where we have to pay a fee upfront before using, the model is a great features for company who wanted to reduce the cost on specific services that they do not use and for students, whom just need to pay for the services they use and not paying the full price.

- Another good feature of AWS is the free tier, there are a number of services that can be used for free by students, learners alike in the process of familiarizing themselves with AWS, offering a good amount of access that is not too abundant but good enough for learning purposes.

- They also have cost saving schemes such as reserved instances and saving plans, which offer discounts based on the long-term commitment to the service.

2.2.6.3 Conclusion

From what we have discussed, AWS is a great tool for business owners and developers alike to make development on products of all sizes, from small startup businesses to giant technology companies such as Netflix, LinkedIn, Airbnb,... with various degrees of services and its conveniences. Beside that, a global architecture also makes AWS becoming popular since it can reach every single corner of the globe easily with a high efficiency of network systems.

Whilst the system itself present a challenge in terms of learning curve and the sheer amount of services can make learners overwhelmed at the first glance, and the complexity usually taken as a limitation by their competitor such as Google Cloud Platform and Azure, the advantages of having a large scale cloud computing system and the convenience of deployment makes it outweighs the limitations and becomes an essential tool for modern business to adapt and utilize.

2.3 Mobile Architecture

2.3.1 Overview

In software development, creating an efficient, scalable, and easy-to-read codebase is essential. Developers have devised numerous solutions, including the 3-tier architecture. This architecture divides the code base into three distinct parts: the data layer, the domain (logic) layer, and the presentation (app) layer. Each of these layers has specific functionalities:

Data Layer:

Responsibilities: Communicating with the backend, sending requests, receiving responses, extracting local or external data. This layer is focused solely on data-related tasks and does not interfere with the business side of the application.

Domain Layer:

Responsibilities: Handling business logic related to the application. It defines processes or use cases for specific screens, such as Login or Registration. It serves as an intermediary between the presentation layer and the data layer.

Presentation Layer:

Responsibilities: Managing what users see on the screen. It includes the user interface codebase and the chosen state management for the application.

In clean architecture, each layer is further divided into three subdivisions

2.3.2 Data Layer

The data layer comprises three sub-layers:

Data Source:

Defines the origin of the data and how to interact with it. Typically includes local data sources (e.g., in-app temporary memory) and remote data sources (e.g., APIs).

Model:

Contains the code for model classes, which handle data management.

Repository:

Implements the repository interfaces defined in the Domain layer. Further details will be provided in a later section.

2.3.3 Domain Layer

The domain layer also has three sub-layers:

Entities:

Defines model classes used exclusively for business logic. These differ from data models in the data layer. Entities are utilized for calculations or extracting information.

Repository:

Defines interfaces listing all data-related functions for a particular screen. These interfaces are then implemented by the repository in the data layer.

Use Cases:

Defines all system design use cases, such as changing the dark mode of a screen.

2.3.4 Presentation Layer

The final layer contains the application user interfaces and state management tools:

State Management:

Defines the state management tools used in the application, such as Bloc, GetX, or Riverpod.

Pages: Contains the user interface code.

Widgets: Defines the components of the user interface.

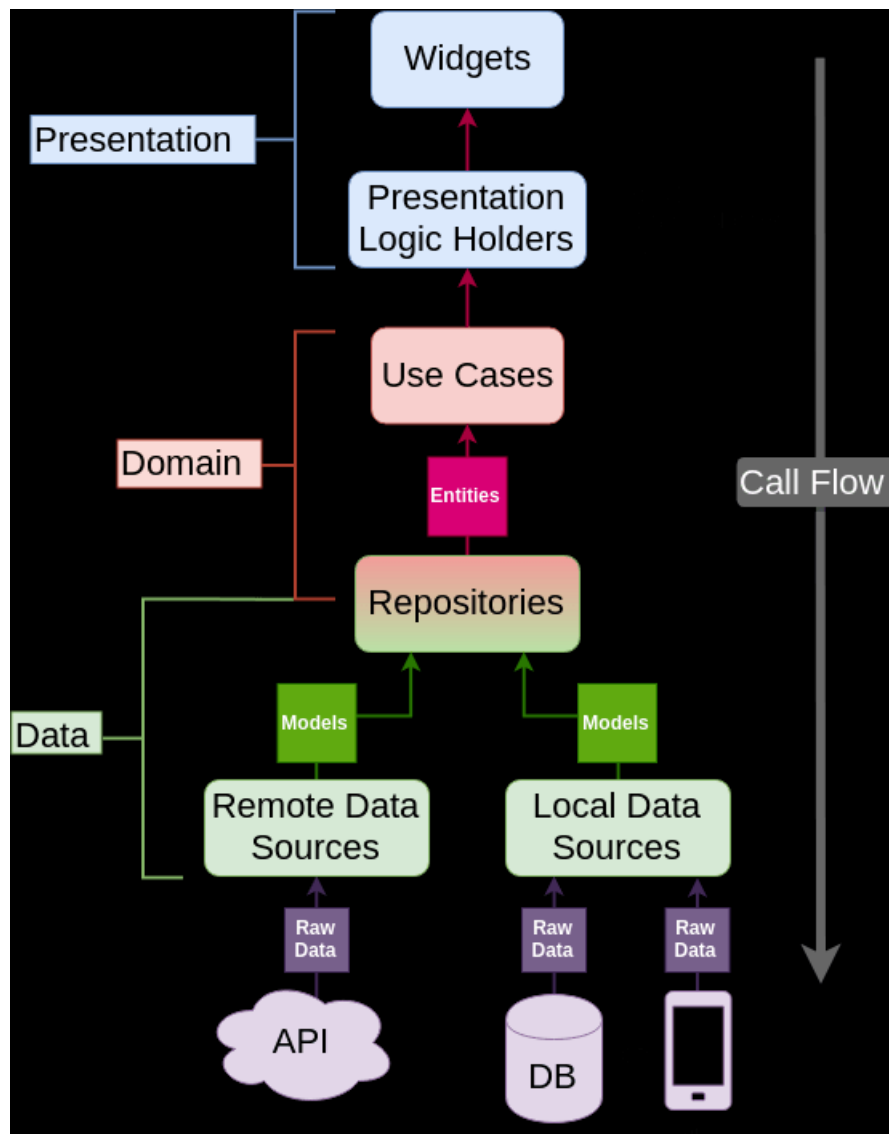


Figure 2.4: Clean architecture diagram

2.4 Backend Architecture

For building backend services, there has been a number of different ways for developers to create an application from earliest days of programming until now, those model are monolithic, service-oriented and microservice. The modern microservice architecture is a collection of small and loosely coupled backends applications (or “services”), they do not necessary be dependent on each other, as a matter of fact, it is a principal to make each service as far as possible in terms of business logic, only having

minimal interaction between them. For example, an app about e-commerce should have at least two services such as Product and Cart, the Cart only need information of the product whenever the User decide to put a Product into the Cart, the reason behind loosely coupled is too keep the client app persistent through the downtime of another service. In a case where the Cart service is down, the User will not be able to put a Product into the Cart, but they can still scroll the Product, which still keeps some service alive enough for the app to function. Now let's talk in more details of the models.

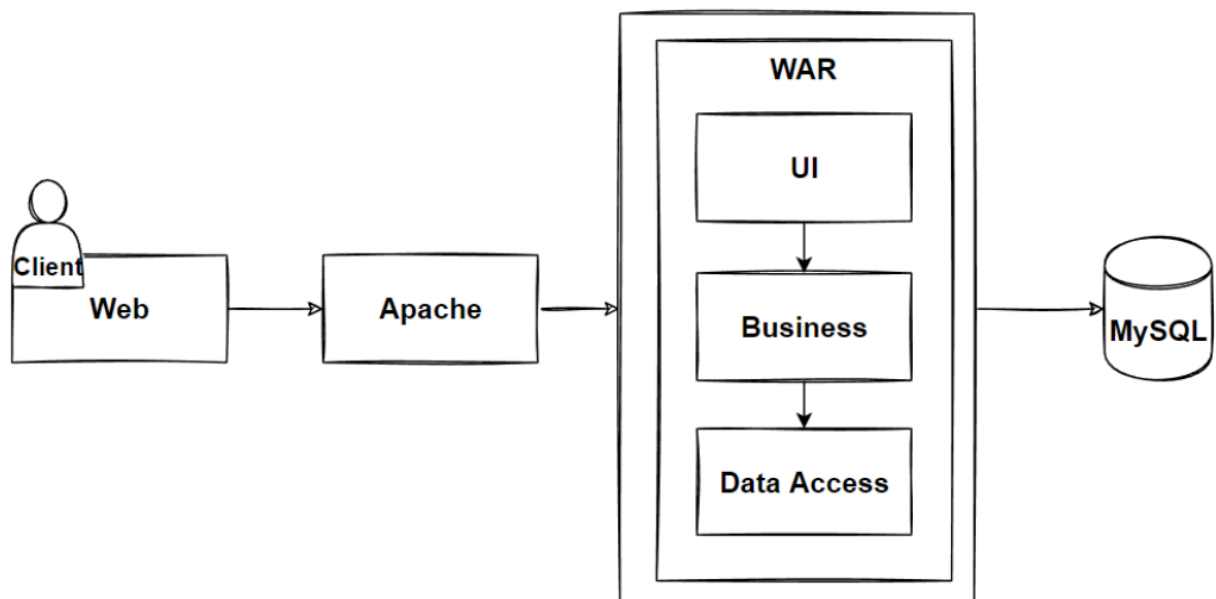


Figure 2.5: Monolithic Architecture

Monolithic Architecture: it refers to a single backend application where all of the functions, services and all business logic are encapsulated into a single codebase.

- Characteristics of this architecture is the centralization of data management, making it very easy for developers to find the source of data and configure accordingly, it has a strong connection between components such as Cart and Product in the previous example and everything is put into a single codebase, making it easier to run and develop.
- The advantages of this model it is fairly easy to develop for a small to medium scale application with not too many developers working on it, it is also very

simple to debug and monitor the status of the app and the infrastructure is going to be cheaper to deploy since the app is only one big application.

- The main disadvantages of the model came to its scalability in terms of “development”, which mean that the larger the codebase, it becomes cumbersome for developers to find the correct module, classes that needed to be debugged. Another pain point for this type of architecture comes from the fact that whenever you have a small change on one particular place, you would need to downtime the application in order to redeploy, which will potentially make the enterprise lose out on profit in some different service.

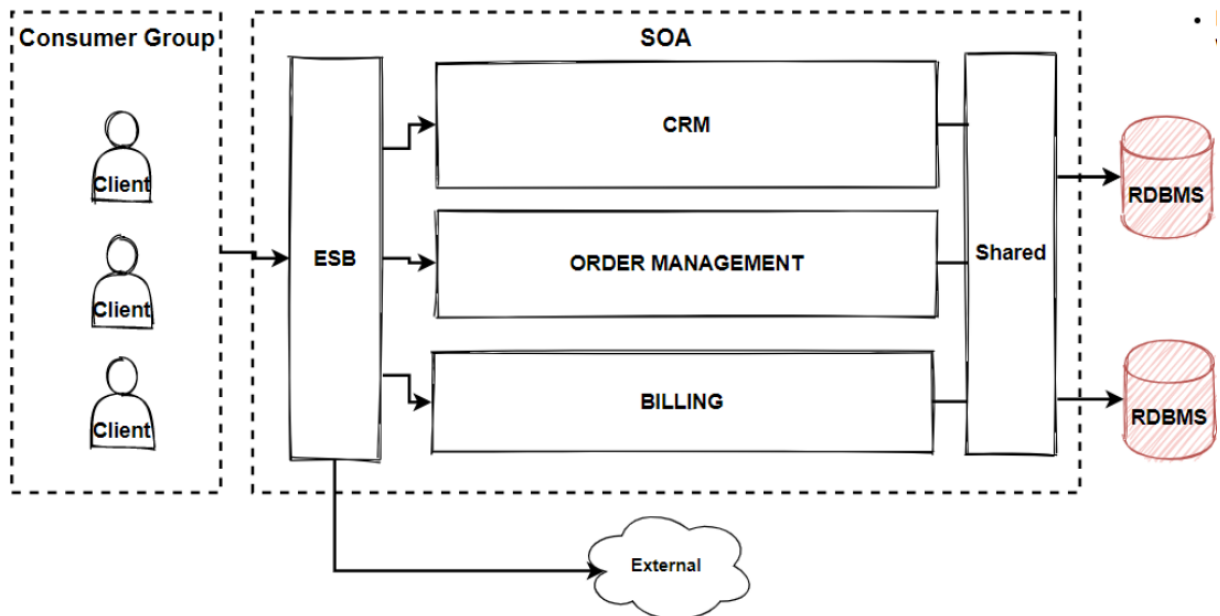


Figure 2.6: Service-Oriented Architecture

Service-Oriented Architecture: This is a type of architecture that arose to address the limitation of monolithic architecture with a heavy emphasize on service-based classes and an access control model of middleware, ensure the correct request will reach the correct destination service.

- Characteristics of this architectural model is the utilization of middleware such as a service bus or RestAPI or XML, SOAP,... and the resuability of services across applications.
- The advantages of this model we can see is that it started to be modularized in terms of services for the ease of management and development where each service can handle the tasks within its domain. The reusable service also make the code less redundant and bring more flexibility when it comes to development.
- But the architecture is without its flaws when it comes to the overhead encountered by middleware, with complex and rigorous infrastructure management and deployment. In addition, there are the latency between communication service due to the fact that it is different service communicate on a network that is centralized.

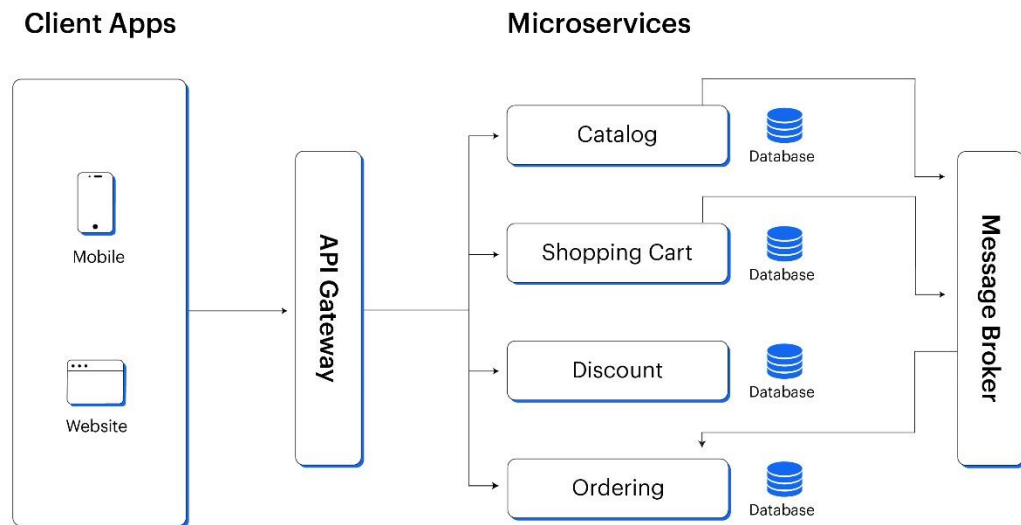


Figure 2.7: Microservices Architecture

Microservice Architecture: It is an extension on the principal of Service-Oriented Architecture, but with extra emphasize on lightweight and decentralized from one another, it typically uses REST API, gRPC or types of event-driven architecture, which means that a service will publish an event and different services will react to it, allowing for the architecture to become loosely coupled.

- The characteristic of this type of architecture is the small and independent services that perform a group of business function that are related to each other, enabling lightweight app packages and reduce management problems which arises from a monolithic style application.
- The advantages of this model is the ease of management, as we can put different services into different repositories for reduced conflicts between code of different team members. In an organization settings, this is very convenience where we can set up multiple development feature team, each team will take one feature and develop it independently from one another. Each service can be deployed and manage easily with little to none impact towards different services in the ecosystem.
- Whilst it has many advantages of developing a microservice backend side, it is noticeable that it is still more efficient to work on a monolithic architecture of the development team is small or if the app does not have too many number of features. The deployment and management process of these services will be expensive and resource-consuming since each of the services will be treated as an independent application.

2.5 Architect Solutions

For now, we have gone over the general ideas of the architecture of backend services, the mobile app architecture and the cloud in the previous part, the next thing is how to assemble a persistent microservice architecture that follows the best practices, guidances of microservice design.

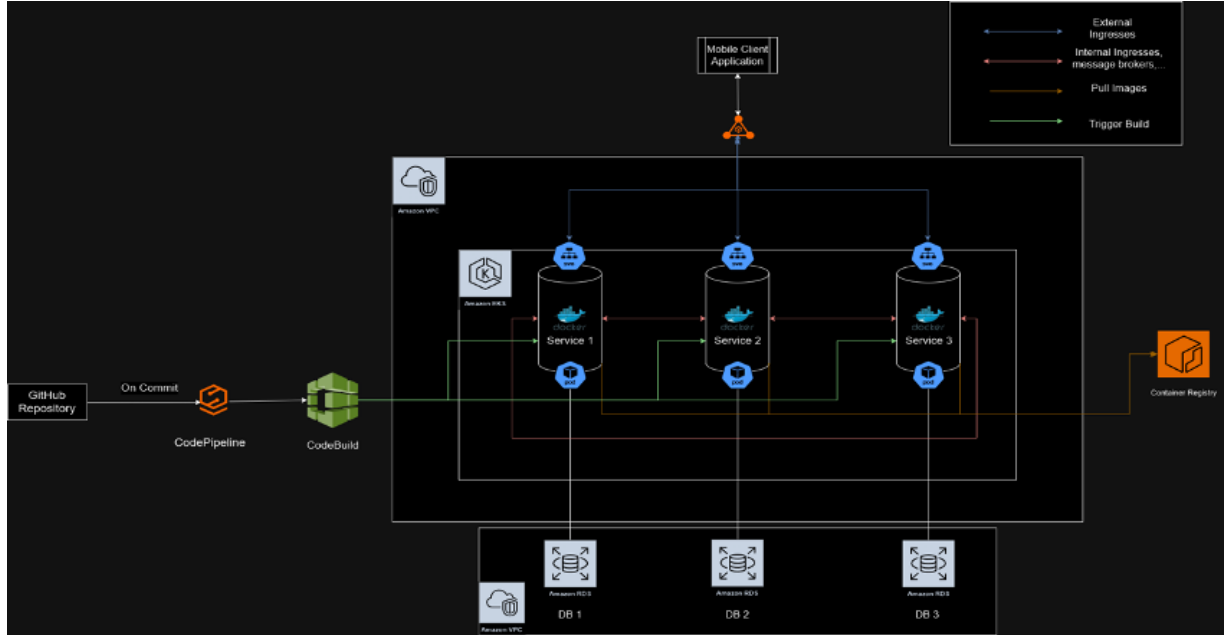


Figure 2.8: Solution Design

To develop a persistent microservice, we first need to decouple the services based on business and functional requirements, for example, user service can only store the information of user and do user/account related features such as registration, login, update profile,... It does not hold directly the external data of different service, image model for example, it just need to hold the reference to the data of different service and it will use that to query for the actual data model.

To ensure each service has a high liveness rate, each of the service will hold a connection with a single database instance, in this way, we can preserve the liveness of the system, imagine a scenario where we need to run migration on a database table, but we have to down the only database, which will terminate the entire system operation.

In terms of network, the deployment services and the database instances are stored inside separated virtual private cloud, which enhances safety as we want to have security access in terms of between the layers and from outside invasion, the principal of the design is to save what is left, if the service penetrated then the database instances are still

safe and database manager can perform retrieve operation and destroy the remnants of the database instances.

On the topic of AWS Elastic Kubernetes Service, each Docker container are put into a division of Kubernetes, which is called **pod**, each pod are the management of container, provide the container with the correct information such as secrets, configuration, connections... pod is an important concept in Kubernetes-based deployment that is needed to be thoroughly understand to deploy such service. To locate these services in terms of network in the deployment, there is **service**, each service will serve a different purpose for your services.

- ClusterIP: The type of service which allows the pod to communicate within a cluster, this is the default type of service.
- NodePort: The type of service that route the incoming traffic into a random port of the pod.
- LoadBalancer: A service that distributes the traffic to pods targetted by service.
- ExternalName: A special type of service that are used to define custom use case such as manual database deployment.

To store all of the Docker containers of service, we can utilize Elastic Container Registry of AWS for this purpose, the image can be stored securely and used by the EC2 instances of EKS if given adequate permissions by user configuration on IAM, there are two most common ways for deploying a container for an EKS service to use is either, build a docker image on your local machine and push directly to Container Registry, after that, use a deployment service to set the configuration. Or, you can setup a continuous integration, continuous deployment to automatically create, upload and deploy the container.

Speaking on setting up a pipeline for development and deployment, we can utilize a CodePipeline service by AWS to build a pipeline for deployment using CodeBuild

trigger, which will trigger the event defined inside the configuration of the build with the popular type of sh or bash command, in summary all of the process is like having a script of command that will perform upon pushing the code to your Github repository.

Now we have covered the general aspect of solutions design, in the next chapter, we are going to take a deeper look in the designing of services, in which are databases, connection, entities, design decision all are defined.

CHAPTER 3: REQUIREMENTS

In this chapter, we are going the features for the users of our English learning app, which are the features they are going to interact with throughout the app. From a high-level perspective, they are going to be separated into the following groups.

- User Management
- Course
- Exercise
- Notification
- Subscription
- Achievement
- Community

3.1 User Requirements

For our application, the sole and only type of users are going to be the learners of our application. They will use the primary features of the app of different group of business cases. We will construct a list of documented of user requirements.

3.1.1 *User Management*

- **Email registration:**
 - Users must be able to register an account using a valid email address, username and password.
 - Users must confirm their password by entering it twice to avoid typos.
 - The system should validate the email format before allowing registration.
 - The password must meet security requirements
- **Social media register/login:**
 - Users must be able to sign up and log in using their Facebook or Google accounts.

- The system should retrieve and store the user's name, email, and profile picture from the selected social login provider.
- Users who sign in via social login should not need to set a password initially.
- Users can set the password for an account by resetting password of their accounts.
- **Email validation:**
 - The system must send a verification email to users after registration. The account needs to be verified to be able to access the app features.
 - The email should contain a unique, time-sensitive 4-digit OTP (One-Time Password).
 - The verification link/OTP should expire after 5 minutes.
 - Users must verify their email before accessing certain features (e.g., resetting password, marketing letters).
- **Phone validation:**
 - The system must send an OTP via SMS when users enter their phone number.
 - The OTP should be a 4-digit numeric code and expire after 5 minutes.
 - Users must enter the correct OTP to verify their phone number.
 - Users should be able to request a new OTP after the previous one expires.
 - A verified phone number must be required for sensitive actions (e.g., password reset).
- **Request for password reset:**
 - Users must be able to request a password reset if they forget their password.
 - The system should provide a way via email to reset a password:

- Users should receive an email containing a unique, time-sensitive reset OTP (One time password).
- The reset OTP should expire after 5 minutes.
- **Resetting for password**
 - Users must enter a new password and enter a confirm one before resetting.
 - The new password must meet security requirements.
- **Change Password**
 - Users must be logged in to access the **change password** feature, which should be available in the **account settings** section.
 - Users must enter their **current password** before setting a new password.
 - Users must enter a **new password** and confirm it by entering it twice.
 - The new password must meet security requirements.
- **Personalize experience**
 - After successful registration and email verification, users must complete their profile by selecting the following information:
 - Native Language: Users must select their native language from a predefined list.
 - This language will be used to tailor the user interface and communication in the app.
 - Learning Language: Users must select the language they wish to learn from a list of available languages.
 - Learning Modes: Users must select their preferred learning mode(s), which could include options such as:
 - Listening
 - Speaking
 - Writing

- Reading
 - Grammar
 - Vocabulary
- Users may select one or maximum of three learning modes based on their preferences.
- **Update user profile, avatar**
 - Users must be able to update their profile information, including:
 - Username: Users can update their username as long as it is unique
 - Native Language: Users can update their native language from a predefined list.
 - Learning Language: Users can change their learning language from the available options.
 - Learning Modes: Users can modify their selected learning mode(s), which could include options such as:
 - Listening
 - Speaking
 - Writing
 - Reading
 - Grammar
 - Vocabulary
 - Users can add or remove one or maximum of three learning modes based on their changing preferences.
 - After making updates, the system should confirm that the profile has been successfully updated and display a success message.
 - All updated information will be saved and used to personalize the user's learning experience moving forward.

- Users should be able to view their updated profile information at any time within the app.

3.1.2 Course

- Course search

- Course Search by Language:
 - Users must be able to search for available courses based on the learning language.
 - The system should return a list of courses that match the selected language, including details such as course name, description, and available levels.
- Course Search by Level (A1 to C2):
 - Users must be able to filter courses by CEFR levels (A1, A2, B1, B2, C1, C2).
 - The system should allow users to select a level from a predefined list of options (A1, A2, B1, B2, C1, C2) and display courses that match the selected level.
 - Courses should indicate which CEFR levels they cover, helping users select the appropriate course based on their proficiency.
- Course learning content view: The users should be able to view the following content of a course:
 - Words: A list of vocabulary words, including their definitions, pronunciations, and example sentences to demonstrate how the word is used in context.
 - Sentences: Sample sentences to demonstrate proper grammar, vocabulary usage, and the context of the words and expressions in various situations.

- Grammar: A section dedicated to grammar rules, explanations, and examples.
- Phrasal Verbs: A collection of phrasal verbs relevant to the course level, including their meanings, example sentences, and pronunciation guides.
- Idioms: Commonly used idioms, their meanings, and examples of how they are used in daily conversations.
- Expressions: A list of useful expressions, including greetings, common conversational phrases, and expressions used in various social and professional contexts.
- Phonetic Transcriptions: The phonetic transcriptions should use an internationally recognized system (e.g., IPA – International Phonetic Alphabet). Show how phonetics are sound in a word

- **View by category courses**

- Users must be able to view a list of course categories, such as:
 - Sport
 - Life
 - Business
 - Travel
 - Culture
 - Health
 - ...
- Categories should be divided into subcategories (e.g., "Sport" category could have subcategories like "Football," "Basketball," "Tennis") to make navigation more efficient.
- Upon selecting a category, users should see a list of courses within that category, displaying:

- Course name
- Number of lessons
- Language

3.1.3 Exercise

- Performing Exercises

- Users must have access to different exercise types to reinforce learning, including:
 - Flashcards – Learn words, phrases, and grammar rules using interactive flashcards.
 - Quiz – Answer multiple-choice and fill-in-the-blank questions to test knowledge.
 - Matching – Match words with their meanings, images, or translations.
 - Pronunciation – Practice speaking with voice recognition and receive feedback.
- Flashcard Learning:
 - Flip cards to reveal meanings, translations, or example sentences.
 - Listen to pronunciations by tapping an audio button.
 - Swipe left to put in not learned category, right to mark as learnt
- Quiz Mode:
 - Multiple-choice questions (word definitions, grammar rules, sentence structures)
 - Matching Exercises: Users must be able to:
 - Drag and drop words or letters to their correct order to form the designated words or sentences.
 - Match text with sound.
- Pronunciation Practice:
 - Practice pronunciation using voice input.

- Get feedback on accuracy and areas of improvement.
- Compare their pronunciation to a native speaker's audio
- **Review their progress**
 - Review incorrect answers with explanations.
 - Retake exercises to improve their scores.

3.1.4 Notification

- **Receiving Notification**
 - User should be able to receive notification on different channel based on use cases:
 - Push Notification (chat notification, ...)
 - Phone (OTP verification, ...)
 - Email (OTP verification for email validation, password reset, ...)

3.1.5 Subscription

- **Subscription purchase**
 - Users are able to buy subscription to access to specialize service
 - Users are able to pay through a payment gateway service
- **Change package with the same price**
 - Users are able to switch to a package that have the same subscription price as the one they currently have

3.1.6 Achievement

- **Earn achievements**
 - Users are able to learn achievement through different process, for example, learn a particular topic, buy subscription packages, chat with different users ... to see their commitment on the application.
 - Achievement should have different categories based on different process.
- **View Achievements**

- Users are able to view Achievements' information such as title, description, point of reward.
- **Thriving for goals:**
 - Users want to make goals that they need to follow
 - The goal should have the information of
 - Title
 - Description
 - Type of process
 - Urgency

3.1.7 Community

- **Artificial Intelligent Helper**
 - Users need an artificial platform to support their study operation during the time on the application
 - Users chat interact with the platform to get help on topics such as finding meanings of a word, grammar explain, fact checks, ...
- **Interaction with different learners**
 - Users need to connect and interact with different learners on the platform.
 - Users can find a group of learners with the same learning goal in terms of language.
 - Users can also initiate a private chat session with a particular users.

3.2 Functional Requirements

From the above requirements of users, we can also divide the app functional requirements based on the general business group of services. For each business group, they will provide the corresponding function to accommodate with users' requirement in the application.

3.2.1 User Management

- **Access token mechanism**

- The application will provide an access-refresh token mechanism with the following expiration marks:
 - Access token: 1 hour
 - Refresh token: 7 days
- If the access token is expired, the mobile app has to send request to renew the access token, by the refresh token.
- If the refresh token is expired, the app will end the session and log the user out.
- The access token will be send through Authorization headers to get authenticated resources.

- **Email registration**

- The service will provide an email registration functionality in which user has to provide the following information:
 - Username (has to be unique)
 - Valid email
 - Password
 - Confirm password (Will be validated on the front end)
- The service will then hashed the password and save the user's information into the database.
- It has to return the access token and refresh token to ensure security when using the app and preserve sessions.

- **Social media login/registration**

- The service will allow users to login with their social media platform, particularly Google and Facebook.
- The front-end will send to the backend the indicator of which type of social login such as:

- Google ID Token
- Facebook Access Code
- The back-end service will query the information of user from Google or Facebook service and save into the database.
- It has to return the access token and refresh token to ensure security when using the app and preserve sessions.
- **Email verification**
 - The system shall allow the user to enter their email address.
 - If the email is valid, the system shall generate a One-Time Password (OTP)
 - The system shall send the OTP or verification link to the provided email
 - The OTP shall be a random numeric code of 4 digits
 - The OTP shall have an expiration time of 5 minutes
 - The email shall include a message instructing the user on how to complete the verification process
 - The user shall enter the OTP in the app to complete verification
 - The system shall validate the OTP against the stored value in the database
 - If the OTP is correct and within the validity period, the system shall mark the email as verified
 - If the OTP is incorrect, the system shall return an error message
 - The system shall allow users to request a new OTP if they did not receive the first one, the cooldown period is 1 minute
- **Phone number verification**
 - The system shall allow the user to enter their phone number
 - If the phone number is valid, the system shall generate a One-Time Password
 - The system shall send the OTP via SMS

- The OTP shall be a random numeric code of 4 digits
 - The OTP shall have an expiration time of 5 minutes
 - The user shall enter the OTP in the app to complete verification
 - If the OTP is correct and within the validity period, the system shall mark the phone number as verified
- **Password reset**
- The system shall allow users to initiate a password reset by entering their registered email or phone number.
 - If valid, the system shall prompt the user to choose the preferred method (Email or SMS) for receiving the OTP
 - The OTP shall be a random numeric code of 4 digits
 - The OTP shall have a validity period of 5 minutes
 - The system shall send the OTP through the method based on user input:
 - Email: An email containing the OTP and reset instructions shall be sent
 - Phone: An SMS containing the OTP and reset instructions shall be sent
 - The system shall allow users to enter the OTP to verify their identity
 - The system shall validate that the OTP is correct, not expired and not used before
- **Change password**
- The system shall allow only **authenticated users** to change their password
 - Users must be **logged in** before accessing the change password feature
 - The system shall prompt the user to enter
 - Current password
 - New password
 - New password confirm
 - The system shall validate that the **current password is correct**

- If the **current password** is incorrect, the system shall return an error message
- If the new password and confirmation password do not match, the system shall return an error message
- If all validations pass, the system shall encrypt and update the user's password in the database
- **Logout**
 - The system shall provide users with a **logout option** in the application
 - When the user selects logout, the system shall invalidate the current session/token
 - Redirect the user to the login page
 - The system shall automatically log out users after the expiration mark of refresh token
- **Registration completion**
 - The system shall require users to complete their profile after initial registration.
 - The profile completion process shall include:
 - Selecting a native language.
 - Selecting a learning language.
 - Choosing at least one and at most three learning modes from the following list:
 - Listening
 - Speaking
 - Writing
 - Reading
 - Grammar
 - Vocabulary

- The system shall provide a fixed list of selectable available languages.
 - The user shall select exactly one native language.
 - The system shall allow users to select one language they want to learn.
 - The system shall display a list of learning modes (Listening, Speaking, Writing, Reading, Grammar, Vocabulary).
 - The user must select at least one and at most three learning modes.
 - The system shall validate the selection and display an error message if the user selects less than one or more than three modes.
 - Upon submission, the system shall save the user's preferences in the database.
- **Update User Profile**
- The system shall allow only **authenticated users** to update their profile.
 - Users must be **logged in** to access the profile update page.
 - The system shall allow users to update the following fields:
 - Username (must be unique)
 - Native language
 - Learning language
 - Learning modes (minimum one, maximum three)
 - The system shall check if the **new username is unique** before saving.
 - The system will provide the list of native language, learning language, learning modes with the format same as that of registration completion.
 - Upon confirmation, the system shall **save the updated profile** to the database and the user should see the updated information inside their app.
- **Update user avatar**
- The system shall allow only **authenticated users** to update their avatar.
 - Users must be **logged in** to access the avatar update feature.

- The system shall allow users to select an image from their device.
- The system shall request permission to access media storage before displaying the file picker.
- Users shall be able to select an image from gallery.
- The system shall upload the image to a secure storage location.
- The system shall store a URL reference to the uploaded image in the user's profile and show the new image on the application.
- If permission is denied, the system shall:
 - Show a message explaining why permission is needed.
 - Open device settings to grant access manually.
- The system shall **only allow the user** to update their own avatar.

3.2.2 Course

- View language courses

- The system shall allow users to browse courses by language.
- The system shall display only available courses for the selected language.
- Inside a selected language, the system shall allow users to filter courses by proficiency level:
 - All (All courses of the language)
 - A1 (Beginner)
 - A2 (Elementary)
 - B1 (Intermediate)
 - B2 (Upper Intermediate)
 - C1 (Advanced)
 - C2 (Proficient)
- The system shall display only courses that match the selected level filter.

- The system shall allow users to view detailed course information, including:
 - Language
 - Level
 - Learning mode
 - Learning process (0 to 100)
- The system shall allow users to join in any courses directly from this page.
- The system shall display courses in a **grid format** for easy browsing.
- The system shall highlight the **selected level filter** to improve user experience.
- The system shall cache frequently accessed courses to improve loading speed.

- **View category courses**

The system shall display a **list of course categories** on the main category page by the **language**

- Each category shall include
 - Category name (e.g., Business English, Travel, Exam Preparation)
 - Category icon or image
 - Category description
- The system shall allow users to click on a category to view its courses
- The system shall navigate users to a new page or section displaying **only courses within the selected category with the corresponding language**
- The system shall display a **list of courses** inside the selected category
 - Each course shall include
 - Course title
 - Language
 - Number of lessons

- Learning progress (0 to 100)
- **View course learning content**
 - Users shall be able to navigate to the learning content from the course page.
 - The system shall organize content into the following categories:
 - Word
 - Idiom
 - Sentence
 - Phrasal Verb
 - Tense
 - Phonetic
 - The system shall provide an audio playback option for all of them.
- **Bookmark courses**
 - The system shall allow users to bookmark a course for later access
 - Users shall be able to bookmark a course by clicking a star icon on the course page
 - The system shall provide a "Bookmarked Courses" section where users can view all saved courses
 - The system shall list bookmarked courses with information the same as language course and category course
 - Users shall be able to **remove a bookmarked course** from the saved list
 - The system shall provide an unstar icon to remove bookmark inside the course details or bookmark list
- **Save course to local database**
 - The system shall allow users to **save a course to Local Database** for offline access
 - The system shall store the course details in Local Database

- The system shall provide an **"Offline Courses" screen** displaying all saved courses
- The system shall fetch cached courses **from Local Database first**, before synching with server if the internet is available
- Users shall be able to **remove a cached course** from Local Database
- Once removed, the course shall no longer appear in the offline courses list

3.2.3 *Exercise*

- **Flashcard Learning**

- The system shall allow users to select **Flashcard Mode** from the course learning content screen after clicking on **Learn**
- Each flashcard shall display:
 - **Front Side:** Word, idiom, sentence, or phrase, and audio pronunciation
 - **Back Side:** Meaning, example, phonetic transcription
- Users shall be able to **flip the card** by tapping
- Users shall be able to **navigate between cards** by swiping left or right or click on **not learned** or **learned** option

- **Quiz Learning**

- The system shall allow users to select **Quiz Mode** from the learning content screen after clicking on **Learn**.
- The system shall generate multiple-choice questions based on learning content.
- Each quiz question shall have:
 - **Question text** (e.g., "What is the meaning of this word?")
 - **Answer options** (one correct answer, rest are incorrect)
 - **Optional audio or image** for enhanced learning

- The system shall randomly shuffle answer choices to prevent memorization.
- Users shall select an answer by tapping on one of the choices.
- The system shall provide immediate feedback:
 - **Correct Answer:** Show correct message and play a success sound (if enabled).
 - **Wrong Answer:** Show incorrect message and play a failure sound (if enabled).
- The system shall display correct versus incorrect answers summary, correct on total at the end of the quiz.
- **Matching Learning**
 - The system shall allow users to select **Matching Learning Mode** from the learning content screen after click on **Learn**
 - Users shall be able to:
 - **Drag and drop** words or letters into the correct order
 - **Tap on items** to move them into the correct position in a sequential order
 - The system shall validate the user's answer once all items are placed and they press on **Check**
 - If the answer is correct:
 - The system shall show success message and play a success sound
 - If the answer is incorrect
 - The system shall show incorrect message and play a failure sound
 - The system shall display correct versus incorrect answers summary, correct on total at the end of the quiz.
- **Pronunciation Learning**

- The system shall allow users to enter **Pronunciation Practice Mode** from the learning content screen.
 - The system shall allow users to **record their voice** using the device's microphone.
 - The system shall request microphone permissions before recording.
 - The system shall analyze the user's pronunciation using **phoneme matching** and provide a **score between 0-100** for each word.
 - The system shall display:
 - **Per-word score (0-100)** to show which words need improvement.
 - **Overall sentence score (0-100)** based on word accuracy.
 - **IELTS Band Score Estimation (1-9 scale)** based on fluency, accuracy, and clarity
 - The system shall display all of the pronunciation overall sentence score of each sentence in the summary.
- **Review exercises**
- The system shall allow users to **view exercises they have completed** within each course.
 - Users shall be able to **navigate to the exercise history** from the user's profile learning history section.
 - The system shall track all completed exercises, including all of the types.
 - Users shall be able to **retake** an exercise from the history screen.

3.2.4 Notification

- **Email notification**
- The system shall send email notifications for the following events:
 - Email verification after registration.
 - Password reset request.

- And more ...
- The system shall use an email service provider (e.g., SendGrid, Amazon SES, Firebase).
- Emails shall be sent **within 5 seconds** of a triggered event.
- **SMS notification**
 - The system shall send phone notifications for the following events:
 - Phone verification
 - Password reset request
 - The system shall deliver phone notifications via SMS channel
 - The system shall use an SMS service provider (e.g., Twilio,...).
- **Push notification**
 - The system shall send push notifications for the following events:
 - Chat notification
 - Achievement receivment notification
 - The system shall use **Firestore Cloud Messaging (FCM)** for Android & iOS push notifications
 - Notifications shall be sent in **real-time**

3.2.5 *Subscription*

- **Subscription purchase**
 - Users shall be able to **view available subscription plans** in the app.
 - Subscription plans shall include:
 - Name
 - Benefits
 - Annual price
 - Users shall only be able to subscribe to **one active plan at a time**.
 - Purchase Flow:

- Select Subscription Plan.
- Proceed to Payment.
- The system processes the payment via an integrated payment gateway (VNPay, Momo ...).
- Upon successful payment, the system shall activate the subscription.
- Users gain access to premium content immediately after payment confirmation.
- Payment:
 - The system shall support multiple payment methods (credit card, debit card ...).
 - Payments shall be processed through **secure gateways**.
 - Transactions shall be **logged** for reference and customer support.
- **Swap subscription plan**
 - Users shall be able to **switch between subscription plans** of the **same price** at any time.
 - The new plan shall **replace** the old plan without extra charges.
 - Flow of changing:
 - User selects a new plan
 - System validates the request
 - Users are able to see new benefits from receiving subscription

3.2.6 Achievement

- **View achievements**
 - Users shall be able to view their achievements, which may include:
 - Topic learning
 - Quiz learning
 - Flashcard learning
 - Matching learning

- Pronunciation learning
- Subscription purchase
- Milestones
- And more ...
- Users shall be able to view achievements in a dedicated section
- Achievements shall be **visually represented** (e.g., badges, trophies, progress bars)
- Achievements shall include:
 - Title
 - Description
 - Date earned
 - Progress tracker
- Users shall receive a **push notification** when unlocking a new achievement.
- The system shall encourage users to **share achievements** on social media.

3.2.7 *Community*

- **AI Chat bot**
 - Users shall be able to access AI chat from the app's menu.
 - The chat shall have a text-based interface.
 - AI chat shall support real-time responses.
 - Users shall be able to create multiple chat sessions with the AI.
 - Each session shall be stored separately, allowing users to continue previous conversations.
 - Users shall be able to delete chat sessions.
 - The chat will support text input.
- **Group chat**

- Users shall be able to create and join group chats with others learning the same language.
- Users shall be able to name the group chat.
- Each group shall have a designated learning language, users of different language can join in as well.
- Users shall be able to leave the group at any time.
- Users shall be able to send and receive messages in real-time.
- The chat shall support text messages.
- Messages shall display timestamps and sender names, avatars.
- Every message should send a push notification to the group member receivers.
- **Private 1-on-1 chat between users**
 - Users shall be able to send chat messages to other users
 - Users shall be able to send and receive real-time messages
 - The chat shall support text input
 - Messages shall display timestamps and sender names, avatars
 - Chats shall be end-to-end encrypted for security
- **To-do management**
 - Users shall be able to **create tasks** with:
 - Title
 - Description
 - To-do topic
 - Priority level
 - Users shall be able to **edit, delete, or mark tasks as completed.**
 - Users shall earn badges for completing tasks.

3.3 Requirement Design

In this section, we are going to get through the designing process for each of the use case we have gone over during the functional requirements

ID	Use Case Name	Description
UC00	Login	Users can log in using their registered email and password.
UC01	Email Registration	Users can create a new account by providing an email address and setting a password
UC02	Social Media Login/Registration	Users can register or log in using third-party social media accounts (Google, Facebook, etc.). The system retrieves necessary user details and automatically creates an account if one does not exist.
UC03	Email Verification	Users must verify their email by entering a code sent to their email to activate their account.
UC04	Phone Number Verification	Users verify their phone number by receiving an OTP via SMS and entering it in the app.
UC05	Password Reset	Users can request a password reset by providing their registered email or phone number. A reset code is sent to the selected method, which users must enter to set a new password.
UC06	Change Password	Users can change their password by providing the old password and setting a new one.
UC07	Logout	Logged-in users can log out of the application, ending their active session.
UC08	Registration Completion	Users complete their profile by selecting their native language, learning language (one only) , and at least one and up to three learning modes from: Listening, Speaking, Writing, Reading, Grammar, or Vocabulary.
UC09	Update User Profile	Users can update their profile details, including username, native language, learning language, and selected learning modes.
UC10	Update User Avatar	Users can upload or change their profile avatar using images from their device. The system requests necessary permissions to access media.
UC11	View Language Courses	Users can browse available courses based on their learning language. They can filter courses by proficiency levels (A1 to C2).
UC12	View Category Courses	Users can view a list of course categories and browse courses within a selected category.
UC13	View Course Learning Content	Users can access learning content, which includes words, idioms, sentences, phrasal verbs, tenses, and phonetics.
UC14	Bookmark Courses	Users can bookmark courses to save them for quick access later.

UC15	Save Course to Local Database	Users can save courses locally for offline access.
UC16	Flashcard Learning	Users can practice vocabulary, idioms, or sentences using flashcards for better retention.
UC17	Quiz Learning	Users participate in multiple-choice quizzes based on their learning content.
UC18	Matching Learning	Users complete matching exercises by dragging or tapping words or letters into the correct order.
UC19	Pronunciation Learning	Users pronounce given words or sentences. The system scores each word from 0 to 100 and provides an IELTS score .
UC20	Review Exercises	Users can review exercises they have completed in each course.
UC21	Email Notification	The system sends important notifications to users via email.
UC22	SMS Notification	The system sends important notifications to users via SMS.
UC23	Push Notification	The system sends push notifications to the user's device for important updates.
UC24	Subscription Purchase	Users can purchase a paid subscription to access premium content. No free trial is offered.
UC25	Swap Subscription Plan	Users can switch to another subscription plan if it has the same price.
UC26	View Achievements	Users can track their progress and view earned achievements based on their learning activities.
UC27	AI Chat Bot	Users can interact with an AI chatbot for language learning support. Multiple chat sessions are allowed with no limitations on conversation topics.
UC28	Group Chat	Users can join group chats with other learners who have the same learning language goal.
UC29	Private 1-on-1 Chat	Users can have private conversations with other learners.
UC30	To-Do Management	Users can manage their learning tasks using a to-do list system.

Table 3.1: Use cases table

Now, we are going to go through the individual feature that the application would offer the users. Starting from Login use case of UC00 until the final UC30, each use case description are going to have the following information:

- Use case ID: Unique identifier of Use case
- Use case name: Name of the use case
- Context: The use case's situation of action

- Description: Use case's full information
- Actors: The primary user
- Triggering Event: Event that fire the use case
- Preconditions: Conditions which must be met before executing the use case
- Result: The desire result of the use case
- System Actor: The system of the use case (app back-end or third-party services)
- Flow of Events: The step by step flow of use case
- Exceptions: The conditions of error

Use Case ID	UC00
Use Case Name	Login
Context	Logging into the application
Description	Users log into the application to access its features.
Actors	User
Triggering Event	User attempts to log into the application.
Preconditions	- The user has access to the application. - The user has a registered account.
Result	The user successfully logs into the application.
System Actor	System
Flow of Events	1. The user selects the "Login" option in the application. 2. The user enters their email/phone number and password. 3. The system verifies the login credentials. 4. If the credentials are valid, the system allows the user to log in. 5. The app store user's token for access to further authorized resources
Exceptions	- If the credentials are incorrect, the system displays an error message.

Table 3.2: UC00 Description

Use Case ID	UC01
Use Case Name	Email Registration
Context	Creating a new user account using an email address
Description	Users can create an account using an email and password.
Actors	User
Triggering Event	User attempts to register a new account using email.
Preconditions	<ul style="list-style-type: none"> - The user has access to a valid email address. - The email is not already registered.
Result	The user successfully registers and verifies their account.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects "Register" in the application. 2. The user enters an email and password. 3. The system checks if the email is already registered. 4. If not registered, the system completes registration.
Exceptions	<ul style="list-style-type: none"> - If the email is already registered, an error message is displayed.

Table 3.3: UC01 Description

Use Case ID	UC02
Use Case Name	Social Media Login/Registration
Context	Registering or logging in with a social media account
Description	Users can register or log in using Google, Facebook, or other third-party authentication.
Actors	User, Third-Party Authentication Provider
Triggering Event	User selects a social media login option from either Google or Facebook
Preconditions	<ul style="list-style-type: none"> - The user has an active social media account.
Result	The user successfully registers/logs in using their social media account.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects "Log in with Google/Facebook" in the app. 2. The system redirects to the social media authentication page. 3. The user grants permission. 4. The system retrieves necessary details and logs the user in.
Exceptions	<ul style="list-style-type: none"> - If authentication fails, an error message is displayed.

Table 3.4: UC02 Description

Use Case ID	UC03
Use Case Name	Email Verification
Context	Verifying the user's email address
Description	Users must verify their email by entering a code sent to their email to activate their account.
Actors	User
Triggering Event	User attempts to verify their email.
Preconditions	<ul style="list-style-type: none"> - The user has entered a valid email during registration. - The email has not been verified yet.
Result	The email is successfully verified.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The system sends a verification email. 2. The user enters the received code. 3. The system verifies the code and activates the account.
Exceptions	<ul style="list-style-type: none"> - If the code is incorrect, the system prompts for re-entry.

Table 3.5: UC03 Description

Use Case ID	UC04
Use Case Name	Phone Number Verification
Context	Verifying the user's phone number
Description	Users verify their phone number by receiving an OTP via SMS and entering it in the app.
Actors	User
Triggering Event	User attempts to verify their phone number.
Preconditions	<ul style="list-style-type: none"> - The user has entered a valid phone number.
Result	The phone number is successfully verified.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The system sends an OTP via SMS. 2. The user enters the OTP. 3. The system verifies the code.
Exceptions	<ul style="list-style-type: none"> - If the code is incorrect, the system prompts for re-entry.

Table 3.6: UC04 Description

Use Case ID	UC05
Use Case Name	Password Reset
Context	Resetting a forgotten password
Description	Users can request a password reset by receiving a verification code via email or SMS.
Actors	User
Triggering Event	User requests password reset.
Preconditions	- The user has a registered account.
Result	The password is successfully reset.
System Actor	System
Flow of Events	1. User enters their email/phone number. 2. The system sends a reset code. 3. User enters the code and sets a new password.
Exceptions	- If the code is incorrect, an error message appears.

Table 3.7: UC05 Description

Use Case ID	UC06
Use Case Name	Change Password
Context	Updating an existing password
Description	Users can change their password by providing the old password and setting a new one. Or by setting a password for a social registered account
Actors	User
Triggering Event	User attempts to change password.
Preconditions	- The user is logged in.
Result	The password is successfully changed.
System Actor	System
Flow of Events	1. User enters the old password. 2. User enters a new password. 3. User enters a new password confirmation. 3. System updates the password.
Exceptions	- If the old password is incorrect, an error is displayed.

Table 3.8: UC06 Description

Use Case ID	UC07
Use Case Name	Logout
Context	Logging out of the application
Description	Users can log out, ending their active session.
Actors	User
Triggering Event	User selects "Logout"
Preconditions	- User is logged in.
Result	User is logged out.
System Actor	System
Flow of Events	1. User selects "Logout." 2. System ends the session.

Table 3.9: UC07 Description

Use Case ID	UC08
Use Case Name	Registration Completion
Context	Completing the user profile after registration
Description	Users complete their profile by selecting their native language, learning language (one only), and at least one and up to three learning modes from: Listening, Speaking, Writing, Reading, Grammar, or Vocabulary.
Actors	User
Triggering Event	The user completes the initial registration process.
Preconditions	- The user has successfully registered an account.
Result	The user profile is completed and saved in the system.
System Actor	System
Flow of Events	1. The system prompts the user to complete their profile. 2. The user selects their native language and learning language. 3. The user selects at least one and up to three learning modes. 4. The system validates the input and saves the data. 5. The user proceeds to the main dashboard.
Exceptions	- If the user does not complete all required fields, an error message is displayed.

Table 3.10: UC08 Description

Use Case ID	UC09
Use Case Name	Update User Profile
Context	Editing personal user information
Description	Users can update their profile details, including username, native language, learning language, and selected learning modes.
Actors	User
Triggering Event	The user selects the "Edit Profile" option.
Preconditions	- The user is logged in.
Result	The user's updated profile information is saved.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user navigates to the profile section. 2. The user selects "Edit Profile." 3. The user updates their details. 4. The system validates the new details. 5. The system saves the updated profile information.
Exceptions	- If required fields are missing, an error message is displayed.

Table 3.11: UC09 Description

Use Case ID	UC10
Use Case Name	Update User Avatar
Context	Changing or uploading a profile picture
Description	Users can upload or change their profile avatar using images from their device. The system requests necessary permissions to access media.
Actors	User
Triggering Event	The user selects the option to update their avatar.
Preconditions	<ul style="list-style-type: none"> - The user is logged in. - The application has permission to access media files.
Result	The new avatar is updated successfully.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user navigates to the profile settings. 2. The user selects "Change Avatar." 3. The system requests media access permission if not already granted. 4. The user selects an image from their device. 5. The system uploads and updates the avatar.
Exceptions	- If the system lacks media permissions, an error message is displayed.

Table 3.12: UC10 Description

Use Case ID	UC11
Use Case Name	View Language Courses
Context	Browsing available courses based on the selected learning language
Description	Users can browse available courses based on their learning language. They can filter courses by proficiency levels (A1 to C2).
Actors	User
Triggering Event	The user navigates to the courses section.
Preconditions	- The user has selected a learning language. - The system has a list of available courses.
Result	The user is presented with a list of language courses filtered by proficiency level.
System Actor	System
Flow of Events	1. The user opens the "Courses" section. 2. The system displays a list of available courses based on the user's learning language. 3. The user selects a proficiency filter (A1–C2). 4. The system updates the list according to the filter.
Exceptions	- If no courses are available, a message is displayed.

Table 3.13: UC11 Description

Use Case ID	UC12
Use Case Name	View Category Courses
Context	Browsing courses by category
Description	Users can view a list of course categories and browse courses within a selected category.
Actors	User
Triggering Event	The user navigates to the course categories section.
Preconditions	- The system has course categories available.
Result	The user is presented with courses grouped by category.
System Actor	System
Flow of Events	1. The user opens the "Course Categories" section. 2. The system displays a list of available categories. 3. The user selects a category. 4. The system displays the courses within that category.
Exceptions	- If no courses are available in a category, a message is displayed.

Table 3.14: UC12 Description

Use Case ID	UC13
Use Case Name	View Course Learning Content
Context	Accessing course materials
Description	Users can access learning content, which includes words, idioms, sentences, phrasal verbs, tenses, and phonetics.
Actors	User
Triggering Event	The user selects a course to view its learning content.
Preconditions	- The selected course has learning content.
Result	The user can view the course content.
System Actor	System
Flow of Events	1. The user selects a course. 2. The system displays the course content. 3. The user navigates through different content sections (words, idioms, etc.).
Exceptions	- If the course has no content, a message is displayed.

Table 3.15: UC13 Description

Use Case ID	UC14
Use Case Name	Bookmark Courses
Context	Saving favorite courses for later access
Description	Users can bookmark courses to save them for quick access later.
Actors	User
Triggering Event	The user selects the option to bookmark a course.
Preconditions	- The user is logged in.
Result	The course is added to the user's bookmarked list.
System Actor	System
Flow of Events	1. The user selects a course. 2. The user taps the "Bookmark" button. 3. The system saves the course to the user's bookmarked list.
Exceptions	- If the user is not logged in, they are prompted to log in first.

Table 3.16: UC14 Descripton

Use Case ID	UC15
Use Case Name	Save Course to Local Database
Context	Offline access to course content
Description	Users can save courses locally for offline access.
Actors	User
Triggering Event	The user selects the option to download a course.
Preconditions	<ul style="list-style-type: none"> - The device has enough storage space. - The user has an internet connection at the time of downloading.
Result	The course is stored in the local database for offline access.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a course. 2. The user taps the "Download" button. 3. The system checks storage availability. 4. The system downloads and saves the course locally.
Exceptions	<ul style="list-style-type: none"> - If there is insufficient storage, an error message is displayed.

Table 3.17: UC15 Description

Use Case ID	UC16
Use Case Name	Flashcard Learning
Context	Practicing vocabulary, idioms, or sentences using flashcards
Description	Users can practice vocabulary, idioms, or sentences using flashcards for better retention.
Actors	User
Triggering Event	The user selects the flashcard learning mode.
Preconditions	<ul style="list-style-type: none"> - The system has flashcard content available.
Result	The user can practice and memorize learning content using flashcards.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a course or topic. 2. The system presents flashcards one by one. 3. The user flips the card to see the meaning, description ... 4. Swipe left to mark as not learn and right as learn 5. The system tracks the user's progress.
Exceptions	<ul style="list-style-type: none"> - If no flashcards are available, a message is displayed.

Table 3.18: UC16 Description

Use Case ID	UC17
Use Case Name	Quiz Learning
Context	Engaging in multiple-choice quizzes
Description	Users participate in multiple-choice quizzes based on their learning content.
Actors	User
Triggering Event	The user starts a quiz session.
Preconditions	- The system has quiz content available.
Result	The user completes a quiz and receives feedback on their performance.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a quiz. 2. The system presents multiple-choice questions. 3. The user selects an answer. 4. The system provides feedback and tracks progress.
Exceptions	- If no quizzes are available, a message is displayed.

Table 3.19: UC17 Description

Use Case ID	UC18
Use Case Name	Matching Learning
Context	Completing matching exercises
Description	Users complete matching exercises by dragging or tapping words or letters into the correct order.
Actors	User
Triggering Event	The user starts a matching exercise.
Preconditions	- The system has matching exercises available.
Result	The user completes a matching exercise and receives feedback.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a matching exercise. 2. The system presents words or letters in a scrambled order. 3. The user drags or taps to arrange them correctly. 4. The system checks the answer and provides feedback.
Exceptions	- If no exercises are available, a message is displayed.

Table 3.20: UC18 Description

Use Case ID	UC19
Use Case Name	Pronunciation Learning
Context	Practicing pronunciation with system scoring
Description	Users pronounce given words or sentences. The system scores each word from 0 to 100 and provides an IELTS score.
Actors	User
Triggering Event	The user selects the pronunciation learning mode.
Preconditions	- The device has a working microphone.
Result	The user receives pronunciation feedback and an IELTS score.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a word or sentence to practice. 2. The system prompts the user to pronounce it. 3. The system records and analyzes pronunciation. 4. The system provides a score and feedback.
Exceptions	- If the microphone is not available, an error message is displayed.

Table 3.21: UC19 Description

Use Case ID	UC20
Use Case Name	Review Exercises
Context	Revisiting completed exercises
Description	Users can review exercises they have completed in each course.
Actors	User
Triggering Event	The user selects the review option in a course.
Preconditions	- The user has completed at least one exercise.
Result	The user can review previous exercises and their performance.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user navigates to the review section. 2. The system displays past exercises with scores. 3. The user selects an exercise to review. 4. The system shows detailed feedback.
Exceptions	- If no exercises have been completed, a message is displayed.

Table 3.22: UC20 Description

Use Case ID	UC21
Use Case Name	Email Notification
Context	Sending important notifications via email
Description	The system sends important notifications to users via email.
Actors	System
Triggering Event	A system event that requires user notification (e.g., account updates, subscription reminders).
Preconditions	- The user has a valid email address registered.
Result	The user receives an email notification.
System Actor	System
Flow of Events	1. The system detects a trigger event. 2. The system formats and sends an email notification. 3. The user receives the email.
Exceptions	- If the email address is invalid, the system logs an error.

Table 3.23: UC21 Description

Use Case ID	UC22
Use Case Name	SMS Notification
Context	Sending important notifications via SMS
Description	The system sends important notifications to users via SMS.
Actors	System
Triggering Event	A system event that requires user notification via SMS.
Preconditions	- The user has a valid phone number registered.
Result	The user receives an SMS notification.
System Actor	System
Flow of Events	1. The system detects a trigger event. 2. The system sends an SMS notification. 3. The user receives the SMS.
Exceptions	- If the phone number is invalid, the system logs an error.

Table 3.24: UC22 Description

Use Case ID	UC23
Use Case Name	Push Notification
Context	Sending push notifications to user devices
Description	The system sends push notifications to the user's device for important updates.
Actors	System
Triggering Event	A system event that requires user notification via push notifications.
Preconditions	- The user has enabled push notifications.
Result	The user receives a push notification.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The system detects a trigger event. 2. The system sends a push notification to the user's device. 3. The user receives the notification.
Exceptions	- If the user has disabled push notifications, the notification is not sent.

Table 3.25: UC23 Description

Use Case ID	UC24
Use Case Name	Subscription Purchase
Context	Allowing users to purchase a paid subscription
Description	Users can purchase a paid subscription to access premium content. No free trial is offered.
Actors	User
Triggering Event	The user selects a subscription plan and proceeds to payment.
Preconditions	- The user has a valid payment method.
Result	The user successfully purchases a subscription.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a subscription plan. 2. The system displays payment options. 3. The user enters payment details and confirms the purchase. 4. The system processes the payment and activates the subscription.
Exceptions	- If the payment fails, an error message is displayed.

Table 3.26: UC24 Description

Use Case ID	UC25
Use Case Name	Swap Subscription Plan
Context	Allowing users to switch subscription plans
Description	Users can switch to another subscription plan if it has the same price.
Actors	User
Triggering Event	The user selects an option to swap their subscription plan.
Preconditions	<ul style="list-style-type: none"> - The user has an active subscription. - The new plan has the same price as the current plan.
Result	The user's subscription is updated to the new plan.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects a new subscription plan. 2. The system checks the eligibility for the swap. 3. If eligible, the system updates the subscription plan. 4. The user is notified of the successful change.
Exceptions	<ul style="list-style-type: none"> - If the new plan has a different price, the swap is not allowed.

Table 3.27: UC25 Description

Use Case ID	UC26
Use Case Name	View Achievements
Context	Tracking learning progress and earned achievements
Description	Users can track their progress and view earned achievements based on their learning activities.
Actors	User
Triggering Event	The user navigates to the achievements section.
Preconditions	<ul style="list-style-type: none"> - The user has completed learning activities that contribute to achievements.
Result	The user views their achievements.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user selects the achievements section. 2. The system retrieves and displays earned achievements.
Exceptions	<ul style="list-style-type: none"> - If no achievements are earned, a message is displayed.

Table 3.28: UC26 Description

Use Case ID	UC27
Use Case Name	AI Chat Bot
Context	Providing AI-powered language learning support
Description	Users can interact with an AI chatbot for language learning support. Multiple chat sessions are allowed with no limitations on conversation topics.
Actors	User, System
Triggering Event	The user initiates a chat with the AI bot.
Preconditions	- The system has an active AI chat feature.
Result	The user engages in a conversation with the AI chatbot.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user starts a chat with the AI bot. 2. The system processes user input and generates responses. 3. The conversation continues until the user exits.
Exceptions	- If the AI chat service is unavailable, an error message is displayed.

Table 3.29: UC27 Description

Use Case ID	UC28
Use Case Name	Group Chat
Context	Allowing users to communicate in group discussions
Description	Users can join group chats with other learners who have the same learning language goal.
Actors	User
Triggering Event	The user selects a group chat to join.
Preconditions	<ul style="list-style-type: none"> - The user has an active account. - The group chat exists for the selected learning language.
Result	The user joins and interacts in a group chat.
System Actor	System
Flow of Events	<ol style="list-style-type: none"> 1. The user navigates to the group chat section. 2. The system displays available group chats. 3. The user selects a chat and joins the conversation.
Exceptions	- If the group chat does not exist, a message is displayed.

Table 3.30: UC28 Description

Use Case ID	UC29
Use Case Name	Private 1-on-1 Chat
Context	Allowing users to have direct conversations with other learners
Description	Users can have private conversations with other learners.
Actors	User
Triggering Event	A user selects another user for a private chat.
Preconditions	- Both users have an active account.
Result	A private chat session is created.
System Actor	System
Flow of Events	1. The user selects another user for private messaging. 2. The system creates a private chat session. 3. The users exchange messages.
Exceptions	- If the other user blocks messages, a notification is shown.

Table 3.31: UC29 Description

Use Case ID	UC30
Use Case Name	To-Do Management
Context	Helping users manage learning tasks
Description	Users can manage their learning tasks using a to-do list system.
Actors	User
Triggering Event	The user accesses the to-do list section.
Preconditions	- The user has an active account.
Result	The user can add, update, or complete tasks.
System Actor	System
Flow of Events	1. The user opens the to-do list. 2. The user adds, updates, or marks tasks as completed. 3. The system saves the changes.
Exceptions	- If a task fails to save, an error message is displayed.

Table 3.32: UC30 Description

CHAPTER 4: SOLUTION DESIGN

4.1 General Solution Design

To create a persistent system architecture and has to be both efficient in terms of development and business logic handling, alongside with the support of multiplatform on mobile application, the system will follow the microservice deployment principal with AWS cloud services for ease of scalability and reliability.

The reasons for selection Microservice can be summarized into the following reasons:

- Ease of management and improve readability for programmers, monolithich architecture tends to be bloated with code if not manage correctly.
- Clear division of business domain of each service can help developers focus on the target features, implementation instead of having to spend too much time navigating the project.

Now that we have the general idea of the project, we will go through the components, services we are using for our development:

- For front-end development, **Flutter** is the chosen framework due to its cross-platform nature to support both Android and iOS, which will help reduce effort and resources to develop mobile application.
- For back-end services, all of the services will use **Kotlin Spring Boot** and follow the microservices design principal
- For database management system, all of our database will be hosted on AWS-supported service of **RDS (Relational Database Service)** as all of our database tables will be written in **SQL**, specifically **PostgreSQL**

- To deploy our service, each of our application containers will be containerized by **Docker**, after that, they will be stored in **ECR (Elastic Container Registry)**, which is an AWS service of hosting containerized images.
- The containers will be orchestrated by Kubernetes with the support of **Elastic Kubernetes Service** on AWS, each of the service will be bind together on a cluster and routed by an **Application Load Balancer (ALB)**, this ensures to separate public and internal endpoint alike for communication between services and to external party. All of the communication between services will utilize **REST API** as well.
- For some special cases, such as Notification Dispatcher, where the requirement is to handle notification by batch and does not have to be synchronous between services, we will use an asynchronous messaging queue called **AWS Simple Queue Service (SQS)**, where the message will be send and does not need to receive a response.
- For media handling, such as images, videos, we are going to use Amazon S3 to save the user uploads and other information related to media.
- The infrastructure will also have to integrate Continuous Integration and Continuos Deployment Pipeline for automation purposes, there for we are going to include:
 - AWS CodePipeline: Pipeline service that will trigger upon conditions (such as pushing code into the main branch)
 - AWS CodeBuild: Service to handle the building of container on AWS, with infrastructure and environment of EC2.
- There are some third party services that we will get into details upon visiting different services in the next section.

From the technologies listed, we can have an overview of architecture in the following simplistic graph, which indicates the position of each technology within the architecture.

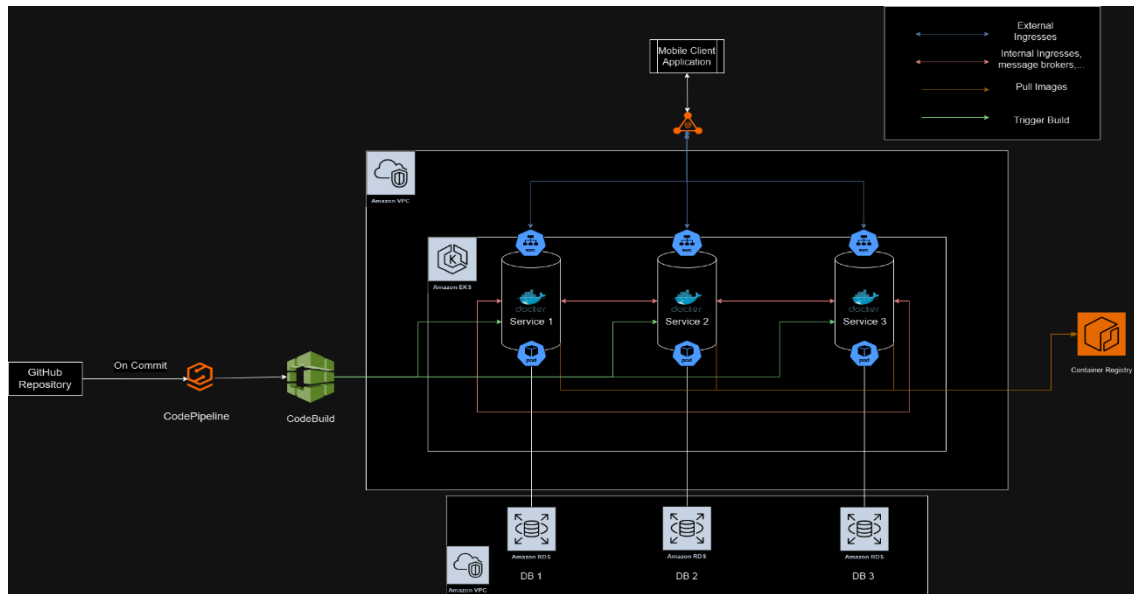


Figure 4.1: Architecture Overview

4.2 Solution Design

In this section, we will explore the solution design of each service, including its entity relationship diagram, cloud service design and the continuous integration, continuous deployment flow.

4.2.1 User Service

For the user service, we are going to store all of User related data whilst keeping references to different resources of different services. Therefore, information such as user subscription and user achievement are references to Achievement and Subscription service respectively.

The user access token are for persistent access-refresh token mechanism to both make the app secure and enhance users' experience.

This service will utilize the Google and Facebook SDK for authentication, it will query the user information for the service.

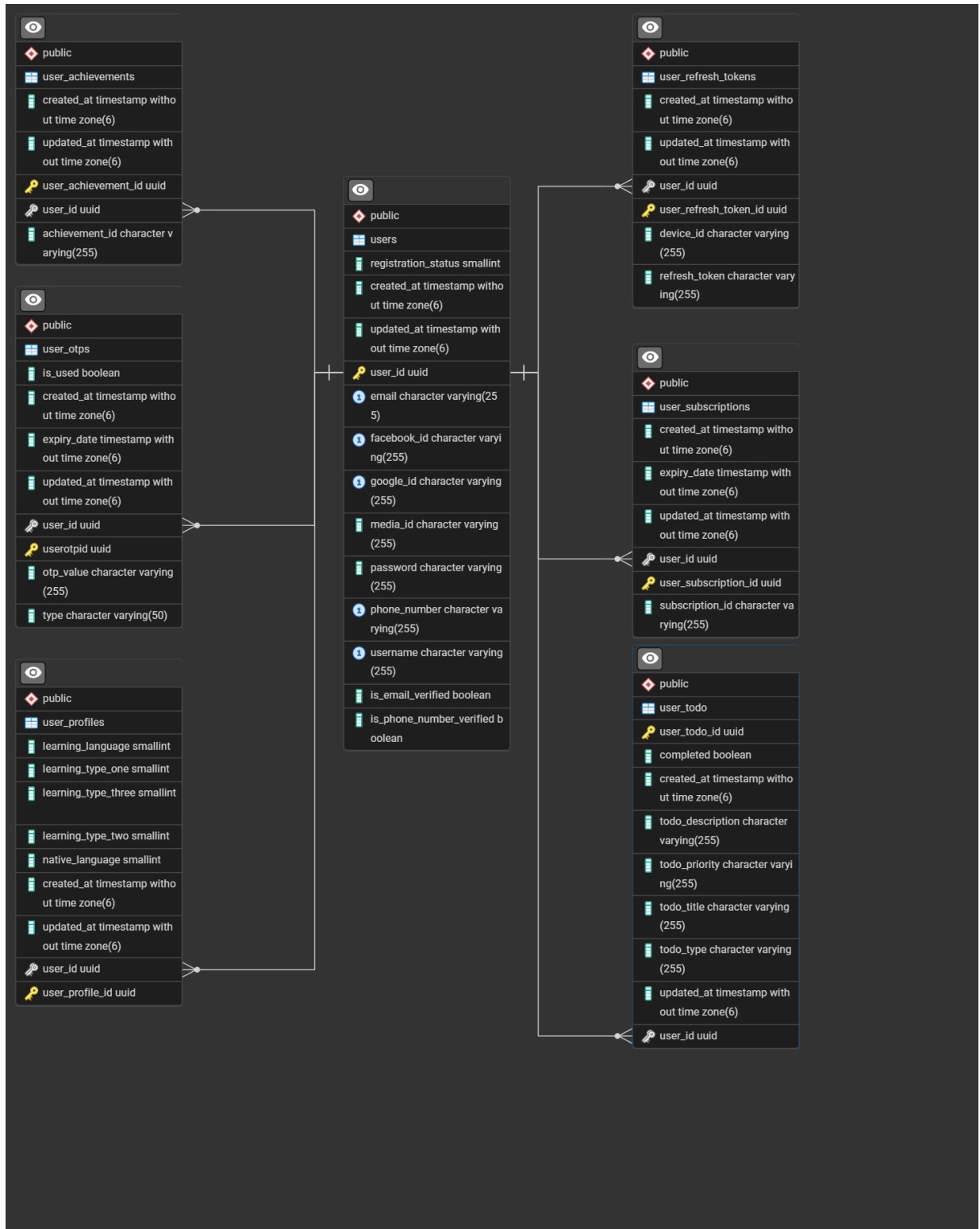


Figure 4.2: User Service ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
registration_status	1	SMALLINT	Status of user registration	1
created_at	2	TIMESTAMP(6)	Record creation timestamp	2024-02-01 10:00:00
updated_at	3	TIMESTAMP(6)	Last update timestamp	2024-02-02 12:00:00
user_id	4	UUID	Unique identifier for the user	550e8400-e29b-41d4-a716-446655440000
email	5	VARCHAR(255)	User email address	user@example.com
facebook_id	6	VARCHAR(255)	User's Facebook ID	1234567890
google_id	7	VARCHAR(255)	User's Google ID	0987654321
media_id	8	VARCHAR(255)	Associated media ID	media12345
password	9	VARCHAR(255)	Hashed password	*****
phone_number	10	VARCHAR(255)	User's phone number	+1234567890
username	11	VARCHAR(255)	Unique username	johndoe
is_email_verified	12	BOOLEAN	Whether email is verified	TRUE
is_phone_number_verified	13	BOOLEAN	Whether phone number is verified	FALSE

Table 4.3: users table

The users table stores essential information about each user, including authentication details, contact information, and verification status. It supports multiple login methods such as email, Facebook, and Google.

Column Name	Ordinal Position	Data Type	Description	Example Value
created_at	1	TIMESTAMP(6)	Record creation timestamp	2024-02-01 10:00:00
updated_at	2	TIMESTAMP(6)	Last update timestamp	2024-02-02 12:00:00
user_achievement_id	3	UUID	Unique identifier for the achievement	550e8400-e29b-41d4-a716-446655440001
user_id	4	UUID	Reference to users table	550e8400-e29b-41d4-a716-446655440000
achievement_id	5	VARCHAR(255)	Identifier for the achievement	550e8400-e29b-41d4-a716-446655440002

Table 4.4: user_achievements table

The user_achievements table tracks achievements earned by users. It links users to specific achievements, providing a historical record of accomplishments.

Column Name	Ordinal Position	Data Type	Description	Example Value
is_used	1	BOOLEAN	Indicates if OTP is used	FALSE
created_at	2	TIMESTAMP(6)	Record creation timestamp	2024-02-01 10:00:00
expiry_date	3	TIMESTAMP(6)	Expiry date of the OTP	2024-02-01 12:00:00
updated_at	4	TIMESTAMP(6)	Last update timestamp	2024-02-02 12:00:00
user_id	5	UUID	Reference to users table	550e8400-e29b-41d4-a716-446655440000
userotpid	6	UUID	Unique identifier for the OTP record	550e8400-e29b-41d4-a716-446655440002
otp_value	7	VARCHAR(255)	One-time password value	123456
type	8	VARCHAR(50)	Type of OTP (email, phone, etc.)	email

Table 4.5: user_otps table

The user_otps table stores one-time passwords (OTPs) generated for user authentication and verification. It includes expiration times and usage status to ensure security.

Column Name	Ordinal Position	Data Type	Description	Example Value
learning_language	1	SMALLINT	User's selected learning language	1
learning_type_one	2	SMALLINT	First preferred learning type	2
learning_type_three	3	SMALLINT	Third preferred learning type	3
learning_type_two	4	SMALLINT	Second preferred learning type	4
native_language	5	SMALLINT	User's native language	5
created_at	6	TIMESTAMP(6)	Record creation timestamp	2024-02-01 10:00:00
updated_at	7	TIMESTAMP(6)	Last update timestamp	2024-02-02 12:00:00
user_id	8	UUID	Reference to users table	550e8400-e29b-41d4-a716-446655440000
user_profile_id	9	UUID	Unique identifier for the profile	550e8400-e29b-41d4-a716-446655440003

Table 4.6: user_profiles table

The user_profiles table stores information about a user's language preferences and learning styles.

Column Name	Ordinal Position	Data Type	Description	Example Value
created_at	1	TIMESTAMP(6)	Record creation timestamp	2024-02-01 10:00:00
updated_at	2	TIMESTAMP(6)	Last update timestamp	2024-02-02 12:00:00
user_id	3	UUID	Reference to users table	550e8400-e29b-41d4-a716-446655440000
user_refresh_token_id	4	UUID	Unique identifier for the refresh token	550e8400-e29b-41d4-a716-446655440004
device_id	5	VARCHAR(255)	Device identifier	device12345
refresh_token	6	VARCHAR(255)	Refresh token value	token12345

Table 4.7: user_refresh_tokens table

The `user_refresh_tokens` table stores refresh tokens for maintaining user authentication across sessions.

Column Name	Ordinal Position	Data Type	Description	Example Value
<code>created_at</code>	1	timestamp(6) without time zone	The timestamp when the subscription was created	2024-02-15 12:34:56.789
<code>expiry_date</code>	2	timestamp(6) without time zone	The expiration date of the subscription	2024-06-15 12:34:56.789
<code>updated_at</code>	3	timestamp(6) without time zone	The timestamp when the record was last updated	2024-02-16 10:00:00.000
<code>user_id</code>	4	uuid	The ID of the user owning the subscription	550e8400-e29b-41d4-a716-446655440000
<code>user_subscription_id</code>	5	uuid	The unique ID of the user subscription record	a111b222-c333-d444-e555-666677778888
<code>subscription_id</code>	6	character varying(255)	The ID of the subscription plan	premium_annual_2024

Table 4.8: `user_subscriptions` table

The `user_subscriptions` table keeps track of users' subscriptions, including when they were created, updated, and their expiration dates. It links to users through the `user_id` field.

Column Name	Ordinal Position	Data Type	Description	Example Value
user_todo_id	1	uuid	The unique ID of the to-do item	123e4567-e89b-12d3-a456-426614174000
completed	2	boolean	Indicates whether the task is completed (true or false)	true
created_at	3	timestamp(6) without time zone	The timestamp when the to-do item was created	2024-02-15 09:00:00.000
todo_description	4	character varying(255)	A brief description of the to-do item	Finish learning 3 topics
todo_priority	5	character varying(255)	The priority level of the task (e.g., High, Medium, Low)	HIGH
todo_title	6	character varying(255)	The title of the to-do item	Learn 3 topics
todo_type	7	character varying(255)	The category/type of the task	TOPIC
updated_at	8	timestamp(6) without time zone	The timestamp when the record was last updated	2024-02-16 11:00:00.000
user_id	9	uuid	The ID of the user who created the task	550e8400-e29b-41d4-a716-446655440000

Table 4.9: user_todo table

The user_todo table stores users' to-do items, including task descriptions, priorities, and statuses. It also tracks when the task was created and last updated.

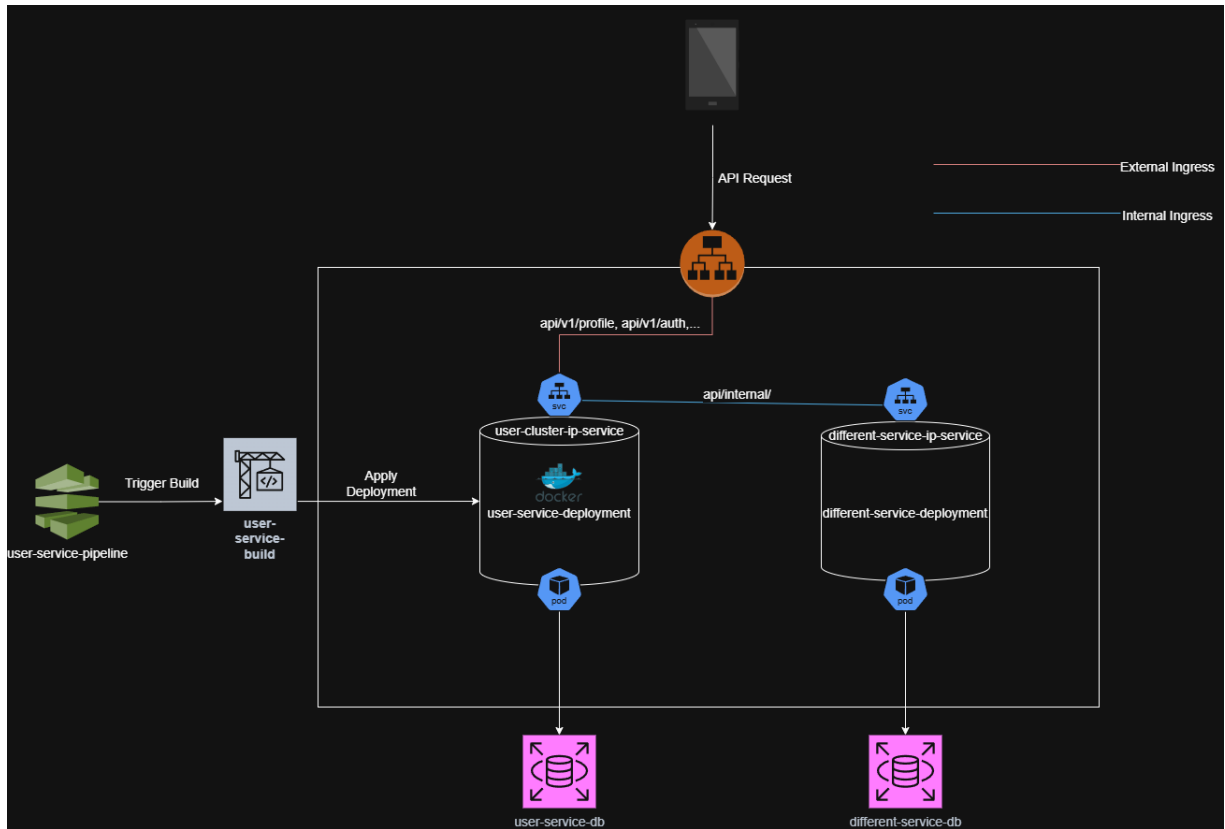


Figure 4.10: User Service Solution Design

4.2.2 Media Service

The media service is the center point of all media related information such as user avatar, course images, category images, learning content media, and much more, it is a center for all of services to get the necessary media.

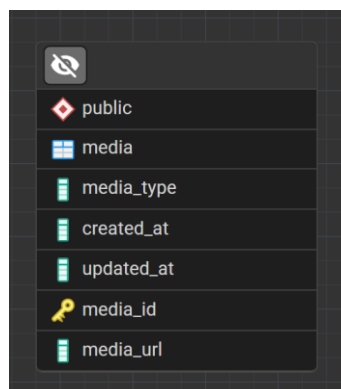


Figure 4.11: Media Service ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
media_type	1	smallint	The type of media (e.g., 1 for image, 2 for video)	1
created_at	2	timestamp(6) without time zone	The timestamp when the media was created	2024-02-15 10:00:00.000
updated_at	3	timestamp(6) without time zone	The timestamp when the media was last updated	2024-02-16 12:30:00.000
media_id	4	uuid	The unique ID of the media record	550e8400-e29b-41d4-a716-446655440000
media_url	5	character varying(255)	The URL where the media is stored	https://example.com/media/image.jpg

Figure 4.12: Media table

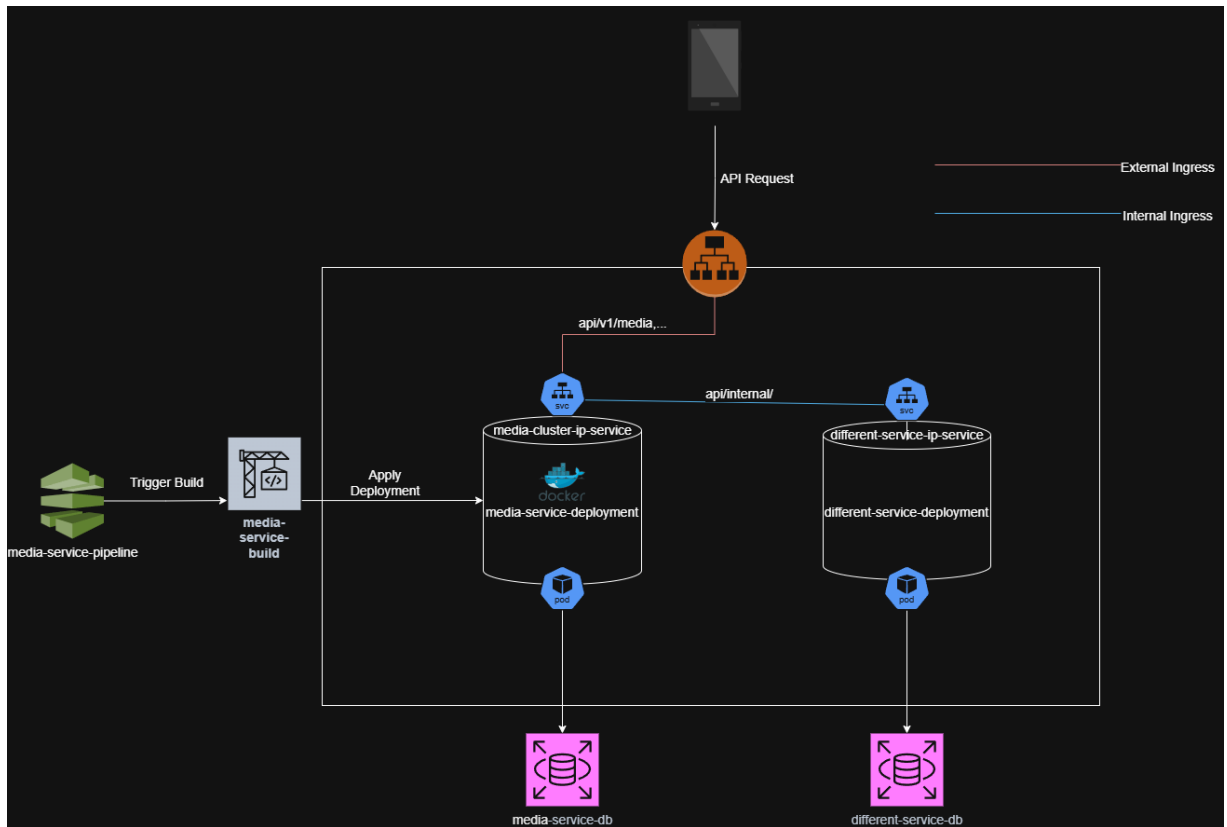


Figure 4.13: Media Service Solution Design

4.2.3 Course Service

The course service will hold two types of lessons, one is the regular language lesson that you would typically see, with level based learning from A1 to C2, and learning by category are lesson that has contents from specific topics, such as sport, life, education ...

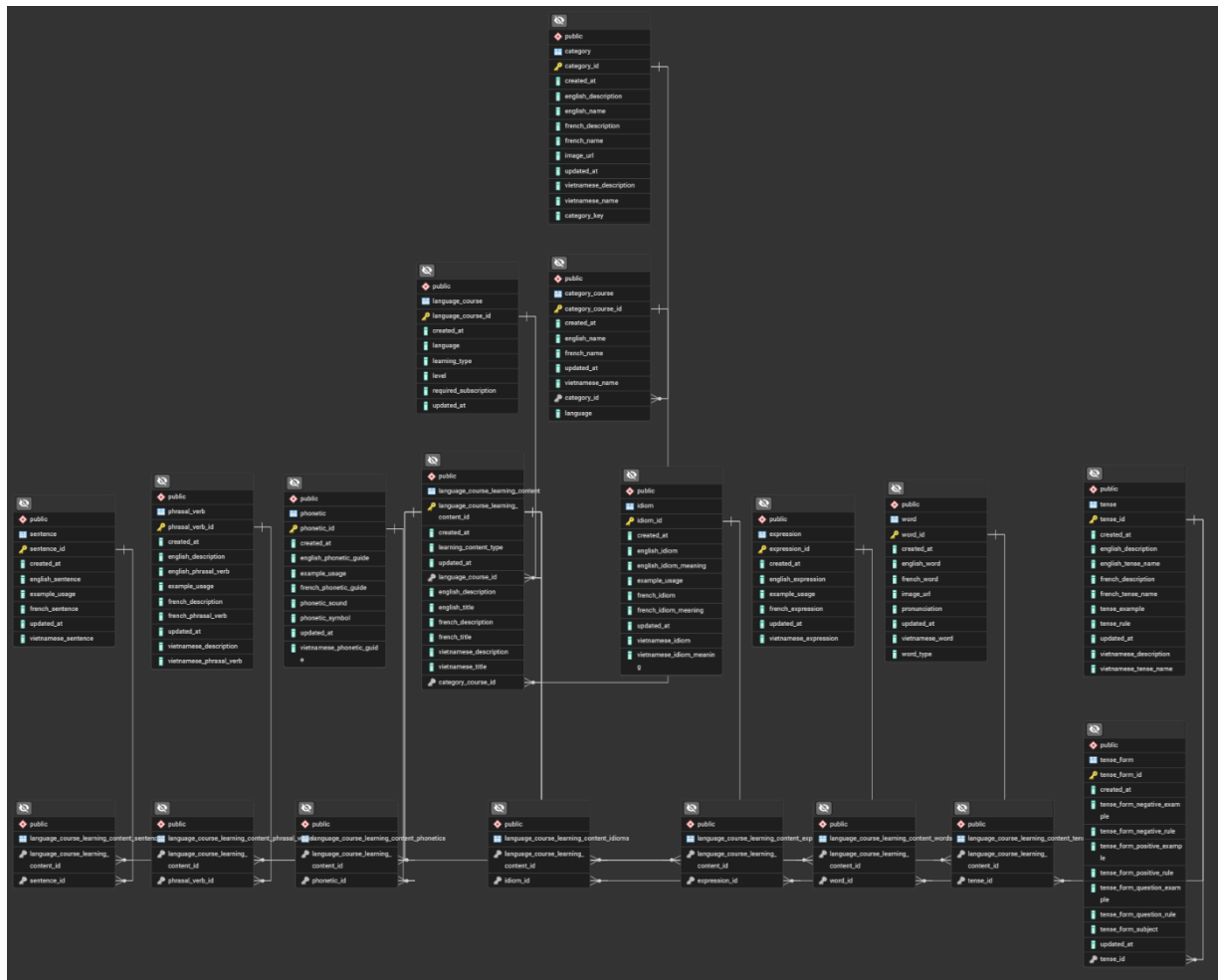


Figure 4.14: Course Service ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
category_course_id	1	uuid	The unique ID of the category-course record	550e8400-e29b-41d4-a716-446655440000
created_at	2	timestamp(6) without time zone	The timestamp when the record was created	2024-02-15 10:00:00.000
english_name	3	character varying(255)	The name of the category in English	Programming Basics
french_name	4	character varying(255)	The name of the category in French	Bases de Programmation
updated_at	5	timestamp(6) without time zone	The timestamp when the record was last updated	2024-02-16 12:30:00.000
vietnamese_name	6	character varying(255)	The name of the category in Vietnamese	Cơ bản lập trình
category_id	7	uuid	The ID of the related category	123e4567-e89b-12d3-a456-426614174000
language	8	character varying(255)	The language of the category (English, French, Vietnamese, etc.)	English

Table 4.15: category_course table

The category_course table links courses to categories while supporting multiple languages.

Column Name	Ordinal Position	Data Type	Description	Example Value
category_id	1	uuid	The unique ID of the category	123e4567-e89b-12d3-a456-426614174000
created_at	2	timestamp(6) without time zone	The timestamp when the category was created	2024-02-15 09:30:00.000
english_description	3	character varying(255)	The description of the category in English	A beginner course for programming
english_name	4	character varying(255)	The name of the category in English	Programming Basics
french_description	5	character varying(255)	The description of the category in French	Un cours pour débutants en programmation
french_name	6	character varying(255)	The name of the category in French	Bases de Programmation
image_url	7	character varying(255)	The URL of the category image	https://example.com/images/category.png
updated_at	8	timestamp(6) without time zone	The timestamp when the record was last updated	2024-02-16 11:30:00.000
vietnamese_description	9	character varying(255)	The description of the category in Vietnamese	Khóa học dành cho người mới bắt đầu
vietnamese_name	10	character varying(255)	The name of the category in Vietnamese	Cơ bản lập trình
category_key	11	character varying(255)	A unique identifier string for the category	programming_basics

Table 4.16: category table

The category table defines categories with multilingual support and includes descriptions, names, and an image URL.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	The unique ID of the learning content	550e8400-e29b-41d4-a716-446655440000
created_at	2	timestamp(6) without time zone	The timestamp when the content was created	2024-02-15 10:00:00.000
learning_content_type	3	character varying(255)	The type of learning content (e.g., video, article, quiz)	Video
updated_at	4	timestamp(6) without time zone	The timestamp when the content was last updated	2024-02-16 12:30:00.000
language_course_id	5	uuid	The ID of the related language course	123e4567-e89b-12d3-a456-426614174000
english_description	6	character varying(255)	The description of the content in English	Introduction to grammar
english_title	7	character varying(255)	The title of the content in English	Grammar Basics
french_description	8	character varying(255)	The description of the content in French	Introduction à la grammaire
french_title	9	character varying(255)	The title of the content in French	Bases de grammaire
vietnamese_description	10	character varying(255)	The description of the content in Vietnamese	Giới thiệu về ngữ pháp
vietnamese_title	11	character varying(255)	The title of the content in Vietnamese	Cơ bản ngữ pháp
category_course_id	12	uuid	The ID of the associated category course	987e6543-e89b-12d3-a456-426614174999

Table 4.17: language_course_learning_content table

The language_course_learning_content table holds learning materials for courses and supports multiple languages. It has the foreign key to both language_course and category_course, the reason for this is to scale the content of learning for some cases but it does not require to have both on one instance.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_id	1	uuid	The unique ID of the language course	123e4567-e89b-12d3-a456-426614174000
created_at	2	timestamp(6) without time zone	The timestamp when the course was created	2024-02-15 09:30:00.000
language	3	character varying(255)	The language of the course	English
learning_type	4	character varying(255)	The type of learning method (e.g., online, offline, hybrid)	Online
level	5	character varying(255)	The difficulty level of the course (e.g., beginner, advanced)	Beginner
required_subscription	6	boolean	Indicates whether a subscription is required	true
updated_at	7	timestamp(6) without time zone	The timestamp when the course was last updated	2024-02-16 11:30:00.000

Table 4.18: language_course table

The language_course table stores details about different language courses, including their type, level, and whether a subscription is required.

Column Name	Ordinal Position	Data Type	Description	Example Value
expression_id	1	character varying(255)	The unique ID of the expression	123e4567-e89b-12d3-a456-426614174001
created_at	2	timestamp(6) without time zone	The timestamp when the expression was created	2024-02-15 08:00:00.000
english_expression	3	character varying(255)	The expression in English	"Break the ice"
example_usage	4	character varying(255)	Example sentence using the expression	"He used a joke to break the ice."
french_expression	5	character varying(255)	The expression in French	"Briser la glace"
updated_at	6	timestamp(6) without time zone	The timestamp when the expression was last updated	2024-02-16 10:00:00.000
vietnamese_expression	7	character varying(255)	The expression in Vietnamese	"Phá băng"

Table 4.19: expression table

The expression table stores phrases and their translations in English, French, and Vietnamese alongside with the example_usage field to provide a sample sentence for better understanding.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	The ID referencing the learning content	550e8400-e29b-41d4-a716-446655440000
expression_id	2	character varying(255)	The ID referencing an expression	123e4567-e89b-12d3-a456-426614174001

Table 4.20: language_course_learning_content_expressions table

This table acts as a many-to-many relationship between learning content and expressions, allowing multiple expressions to be linked to a single content item.

Column Name	Ordinal Position	Data Type	Description	Example Value
idiom_id	1	character varying(255)	The unique ID of the idiom	550e8400-e29b-41d4-a716-446655440001
created_at	2	timestamp(6) without time zone	The timestamp when the idiom was created	2024-02-15 08:00:00.000
english_idiom	3	character varying(255)	The idiom in English	"Hit the sack"
english_idiom_meaning	4	character varying(255)	The meaning of the idiom in English	"Go to bed"
example_usage	5	character varying(255)	Example sentence using the idiom	"I'm exhausted; it's time to hit the sack."
french_idiom	6	character varying(255)	The idiom in French	"Aller au lit"
french_idiom_meaning	7	character varying(255)	The meaning of the idiom in French	"Aller dormir"
updated_at	8	timestamp(6) without time zone	The timestamp when the idiom was last updated	2024-02-16 10:00:00.000
vietnamese_idiom	9	character varying(255)	The idiom in Vietnamese	"Đi ngủ"
vietnamese_idiom_meaning	10	character varying(255)	The meaning of the idiom in Vietnamese	"Đi ngủ, nghỉ ngơi"

Table 4.21: idiom table

The idiom table stores idioms in English, French, and Vietnamese, along with their meanings and example usage.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	The ID referencing the learning content	550e8400-e29b-41d4-a716-446655440000
idiom_id	2	character varying(255)	The ID referencing an idiom	550e8400-e29b-41d4-a716-446655440001

Table 4.22: language_course_learning_content_idioms table

The language_course_learning_content_idioms table links idioms to learning content, supporting multiple idioms per content item.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	The ID referencing the learning content	550e8400-e29b-41d4-a716-446655440000
phonetic_id	2	uuid	The ID referencing a phonetic guide	123e4567-e89b-12d3-a456-426614174000

Table 4.23: language_course_learning_content_phonetics table

The language_course_learning_content_phonetics table links phonetics to learning content, allowing multiple phonetic guides per content item.

Column Name	Ordinal Position	Data Type	Description	Example Value
phonetic_id	1	uuid	The unique ID of the phonetic guide	123e4567-e89b-12d3-a456-426614174000
created_at	2	timestamp(6) without time zone	The timestamp when the phonetic guide was created	2024-02-15 08:00:00.000
english_phonetic_guide	3	character varying(255)	The phonetic guide for English pronunciation	"ə'baʊt"
example_usage	4	character varying(255)	Example word or phrase using the phonetic	"about"
french_phonetic_guide	5	character varying(255)	The phonetic guide for French pronunciation	"a.bu"
phonetic_sound	6	character varying(255)	The pronunciation audio file (URL)	"https://example.com/sound.mp3"
phonetic_symbol	7	character varying(255)	The IPA symbol representation	"ə"
updated_at	8	timestamp(6) without time zone	The timestamp when the phonetic guide was last updated	2024-02-16 10:00:00.000
vietnamese_phonetic_guide	9	character varying(255)	The phonetic guide for Vietnamese pronunciation	"a'bawt"

Table 4.24: phonetic table

The phonetic table stores multilingual phonetics, including IPA symbols, pronunciation audio, and example usage.

Column Name	Ordinal Position	Data Type	Description	Example Value
phrasal_verb_id	1	uuid	The unique ID of the phrasal verb	123e4567-e89b-12d3-a456-426614174000
created_at	2	timestamp(6) without time zone	The timestamp when the phrasal verb was created	2024-02-15 08:00:00.000
english_description	3	character varying(255)	A brief explanation of the phrasal verb meaning in English	"To quit or abandon something"
english_phrasal_verb	4	character varying(255)	The actual phrasal verb in English	"give up"
example_usage	5	text	Example sentences demonstrating usage	"She decided to give up smoking."
french_description	6	character varying(255)	A brief explanation of the phrasal verb in French	"Arrêter de faire quelque chose"
french_phrasal_verb	7	character varying(255)	The phrasal verb in French	"abandonner"
updated_at	8	timestamp(6) without time zone	The timestamp when the phrasal verb was last updated	2024-02-16 10:00:00.000
vietnamese_description	9	character varying(255)	A brief explanation of the phrasal verb in Vietnamese	"Từ bỏ điều gì đó"
vietnamese_phrasal_verb	10	character varying(255)	The phrasal verb in Vietnamese	"từ bỏ"

Table 4.25: phrasal_verb table

The phrasal_verb table stores multilingual phrasal verbs, including descriptions and example usages.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	The ID referencing the learning content	550e8400-e29b-41d4-a716-446655440000
phrasal_verb_id	2	uuid	The ID referencing a phrasal verb	123e4567-e89b-12d3-a456-426614174000

Table 4.26: language_course_learning_content_phrasal_verbs table

The `language_course_learning_content_phrasal_verbs` table links phrasal verbs to learning content, allowing multiple phrasal verbs per content item. This structure supports phrasal verb learning across multiple languages.

Column Name	Ordinal Position	Data Type	Description	Example Value
<code>sentence_id</code>	1	uuid	The unique ID of the sentence	123e4567-e89b-12d3-a456-426614174000
<code>created_at</code>	2	timestamp(6) without time zone	The timestamp when the sentence was created	2024-02-15 08:00:00.000
<code>english_sentence</code>	3	character varying(255)	The sentence in English	"The quick brown fox jumps over the lazy dog."
<code>example_usage</code>	4	text	Example usage of the sentence	"This sentence contains every letter in the English alphabet."
<code>french_sentence</code>	5	character varying(255)	The sentence in French	"Le renard brun rapide saute par-dessus le chien paresseux."
<code>updated_at</code>	6	timestamp(6) without time zone	The timestamp when the sentence was last updated	2024-02-16 10:00:00.000
<code>vietnamese_sentence</code>	7	character varying(255)	The sentence in Vietnamese	"Con cáo nâu nhanh nhẹn nhảy qua con chó lười biếng."

Table 4.27: sentence table

The sentence table stores multilingual sentences, including example usages.

Column Name	Ordinal Position	Data Type	Description	Example Value
<code>language_course_learning_content_id</code>	1	uuid	The ID referencing the learning content	550e8400-e29b-41d4-a716-446655440000
<code>sentence_id</code>	2	uuid	The ID referencing a sentence	123e4567-e89b-12d3-a456-426614174000

Table 4.28: `language_course_learning_content_sentences` table

The `language_course_learning_content_sentences` table links sentences to learning content, allowing multiple sentences per content item.

Column Name	Ordinal Position	Data Type	Description	Example Value
tense_id	1	uuid	Unique ID for the tense	550e8400-e29b-41d4-a716-446655440000
created_at	2	timestamp(6) without time zone	Timestamp when the tense was created	2025-02-15 08:00:00.000
english_tense_name	3	character varying(255)	The name of the tense in English	"Present Simple"
english_description	4	character varying(255)	Description of the tense in English	"Used for general facts and routines."
french_tense_name	5	character varying(255)	The name of the tense in French	"Présent simple"
french_description	6	character varying(255)	Description of the tense in French	"Utilisé pour les faits et habitudes."
vietnamese_tense_name	7	character varying(255)	The name of the tense in Vietnamese	"Hiện tại đơn"
vietnamese_description	8	character varying(255)	Description of the tense in Vietnamese	"Dùng cho các sự thật chung và thói quen."
tense_example	9	character varying(255)	Example usage of the tense	"She eats breakfast every morning."
tense_rule	10	character varying(255)	The grammatical rule of the tense	"Subject + Verb(s/es) + Object"
updated_at	11	timestamp(6) without time zone	Timestamp when the tense was last updated	2025-02-16 10:00:00.000

Table 4.29: tense table

The tense table stores tense definitions across English, French, and Vietnamese, including rules and examples.

Column Name	Ordinal Position	Data Type	Description	Example Value
tense_form_id	1	uuid	Unique ID for the tense form	123e4567-e89b-12d3-a456-426614174000
created_at	2	timestamp(6) without time zone	Timestamp when the tense form was created	2025-02-15 08:00:00.000
tense_form_subject	3	character varying(255)	The subject in the sentence structure	"She"
tense_form_positive_example	4	character varying(255)	Positive example of the tense	"She eats breakfast."
tense_form_positive_rule	5	character varying(255)	Rule for forming the positive sentence	"Subject + Verb(s/es) + Object"
tense_form_negative_example	6	character varying(255)	Negative example of the tense	"She does not eat breakfast."
tense_form_negative_rule	7	character varying(255)	Rule for forming the negative sentence	"Subject + does not + Verb(base) + Object"
tense_form_question_example	8	character varying(255)	Question example of the tense	"Does she eat breakfast?"
tense_form_question_rule	9	character varying(255)	Rule for forming the question	"Does + Subject + Verb(base) + Object?"
updated_at	10	timestamp(6) without time zone	Timestamp when the tense form was last updated	2025-02-16 10:00:00.000
tense_id	11	uuid	Foreign key referencing tense_id in tense table	550e8400-e29b-41d4-a716-446655440000

Table 4.30: tense_form table

The tense_form table provides examples and formation rules for affirmative, negative, and interrogative sentence structures. A one-to-many relationship exists where one tense can have multiple tense forms.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	uuid	Foreign key referencing language_course_learning_content_id in the language_course_learning_content table.	550e8400-e29b-41d4-a716-446655440000
tense_id	2	uuid	Foreign key referencing tense_id in the tense table.	76e6c81a-1b2c-4a87-9e78-2b4312f26a47

Table 4.31: language_course_learning_content_tenses table

This table establishes a many-to-many relationship between language_course_learning_content and tense. It links different learning content pieces (such as lessons or exercises) to the tenses they cover. Each record represents an association between a specific learning content item and a tense, allowing the system to manage and categorize grammar lessons efficiently.

Column Name	Ordinal Position	Data Type	Description	Example Value
word_id	1	UUID	Unique identifier for each word.	a2c54b33-9d0f-4f34-bde6-16b6ef0e1325
created_at	2	TIMESTAMP(6)	Timestamp when the word entry was created.	2024-02-15 14:30:00
english_word	3	VARCHAR(255)	The word in English.	apple
french_word	4	VARCHAR(255)	The word in French.	pomme
image_url	5	VARCHAR(255)	URL to an image representing the word (optional).	https://example.com/apple.png
pronunciation	6	VARCHAR(255)	Phonetic pronunciation of the word.	/'æp.əl/
updated_at	7	TIMESTAMP(6)	Timestamp when the word entry was last updated.	2024-02-16 10:45:00
vietnamese_word	8	VARCHAR(255)	The word in Vietnamese.	táo
word_type	9	VARCHAR(255)	The category/type of the word (e.g., noun, verb, adjective).	noun

Table 4.32: word table

The word table stores vocabulary words along with their translations, pronunciation, and associated metadata. It helps in structuring vocabulary lessons by providing words in multiple languages and categorizing them by type.

Column Name	Ordinal Position	Data Type	Description	Example Value
language_course_learning_content_id	1	UUID	Foreign key referencing language_course_learning_content_id	550e8400-e29b-41d4-a716-446655440000
word_id	2	UUID	Foreign key referencing word_id in the word table.	a2c54b33-9d0f-4f34-bde6-16b6ef0e1325

Table 4.33: language_course_learning_content_words table

This table establishes a many-to-many relationship between language_course_learning_content and word. It links different learning content items (such as lessons or exercises) to specific words that are taught or practiced. This allows structured organization and retrieval of vocabulary associated with particular lessons.

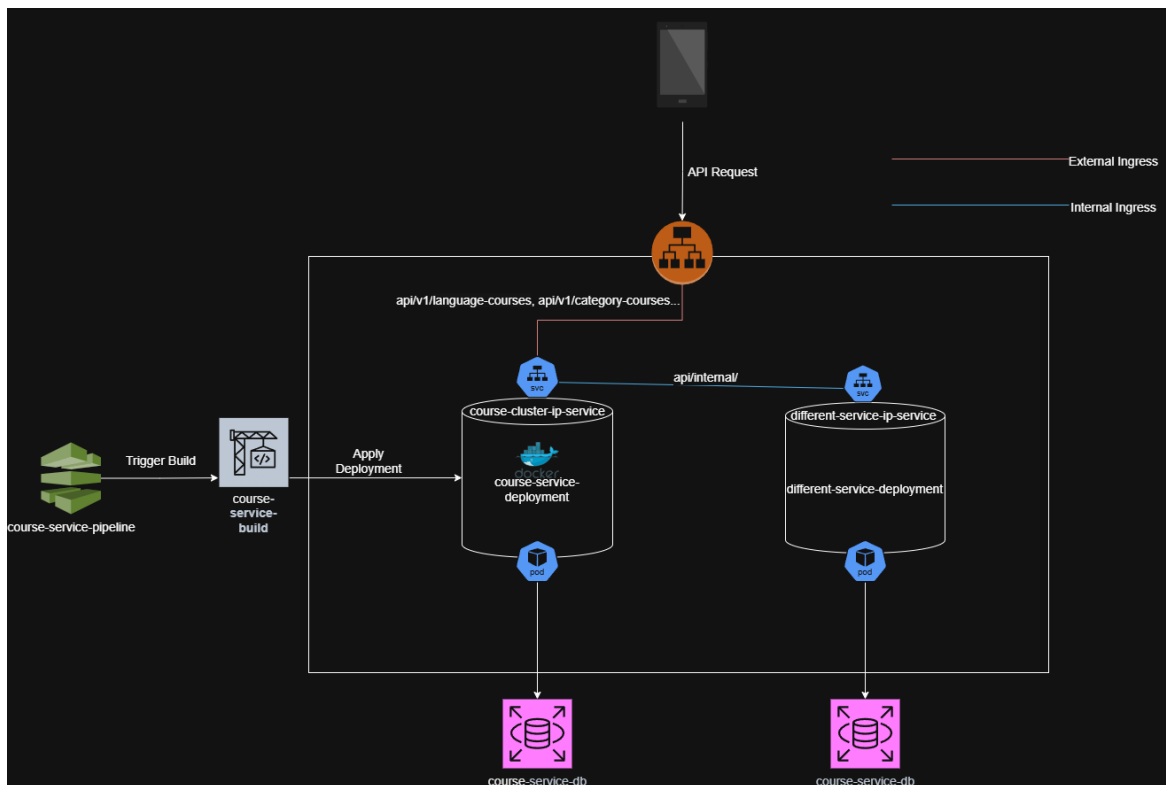


Figure 4.34: Course Service Solution Design

4.2.4 Exercise Service

Exercise service provide users a handful of ways to learn the courses, it is also used to track all of the progress of users when they finish the course in the following 4 types of learning:

- Flashcards
- Matching
- Quiz
- Pronunciation

The service has to be able to store all four learning components, at the mean time, it also need to execute external API to provide user pronunciation assessment in the article of Language Confidence (<https://rapidapi.com/language-confidence-language-confidence-default/api/pronunciation-assessment1>)

The matching animation will utilize Flutter’s animation API for actions such as drag to fill.

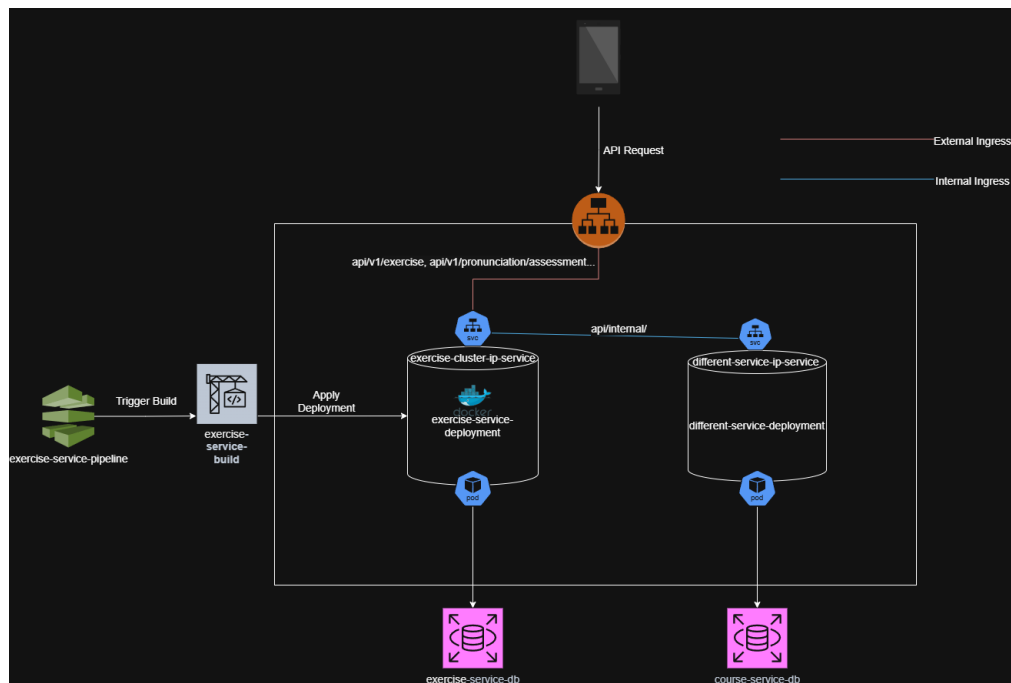


Figure 4.35: Exercise Solution Design

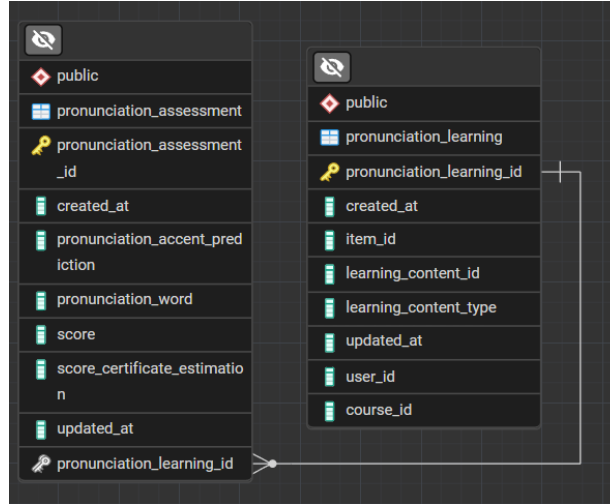


Figure 4.36: Pronunciation Learning ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
pronunciation_assessment_id	1	UUID	Unique identifier for each pronunciation assessment.	f1d5a9c4-6e3f-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the assessment was recorded.	2024-02-15 09:30:00
pronunciation_accent_prediction	3	VARCHAR(1024)	AI-predicted accent of the pronunciation (optional).	"British"
pronunciation_word	4	VARCHAR(1024)	The word being assessed for pronunciation.	"apple"
score	5	INTEGER	Pronunciation score, typically on a predefined scale (e.g., 0-100).	85
score_certificate_estimation	6	VARCHAR(1024)	Estimated certificate level based on pronunciation performance.	"B2"
updated_at	7	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 10:00:00
pronunciation_learning_id	8	UUID	Foreign key referencing pronunciation_learning_id in the pronunciation_learning table.	a5b9e3f8-8d1e-11ee-b962-0242ac120002

Table 4.37: pronunciation_assessment table

This table stores the results of pronunciation assessments, linking them to pronunciation learning sessions. It records the pronunciation word, accent prediction, score, and certificate estimation for learners.

Column Name	Ordinal Position	Data Type	Description	Example Value
pronunciation_learning_id	1	UUID	Unique identifier for each pronunciation learning session.	a5b9e3f8-8d1e-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the pronunciation learning session was created.	2024-02-15 08:45:00
item_id	3	UUID	Foreign key referencing a specific word, sentence, or phrasal verb being learned.	f9c5b7d2-8d1e-11ee-b962-0242ac120002
learning_content_id	4	UUID	Foreign key referencing the learning content item.	e8b1c6f0-8d1e-11ee-b962-0242ac120002
learning_content_type	5	SMALLINT	Indicates the type of learning content (e.g., 1 = word, 2 = sentence, etc.).	1
updated_at	6	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 09:30:00
user_id	7	UUID	Foreign key referencing the user learning pronunciation (optional).	d7a3b5e4-8d1e-11ee-b962-0242ac120002
course_id	8	UUID	Foreign key referencing the associated course (optional).	c6b2a1f6-8d1e-11ee-b962-0242ac120002

Table 4.38: pronunciation_learning table

This table tracks users' pronunciation learning activities. It stores information about the learning content, user association, and course context.

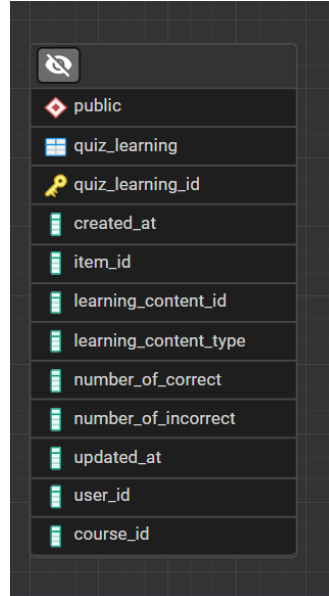


Figure 4.39: Quiz Learning ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
quiz_learning_id	1	UUID	Unique identifier for each quiz learning session.	a1b2c3d4-e5f6-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the quiz session was started.	2024-02-15 08:30:00
item_id	3	UUID	Foreign key referencing a specific learning item (word, sentence, etc.).	f9c5b7d2-8d1e-11ee-b962-0242ac120002
learning_content_id	4	UUID	Foreign key referencing the learning content item.	e8b1c6f0-8d1e-11ee-b962-0242ac120002
learning_content_type	5	SMALLINT	Indicates the type of learning content (e.g., 1 = word, 2 = sentence).	2
number_of_correct	6	INTEGER	The number of correct answers in the quiz session.	5
number_of_incorrect	7	INTEGER	The number of incorrect answers in the quiz session.	2
updated_at	8	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 09:00:00
user_id	9	UUID	Foreign key referencing the user taking the quiz (optional).	d7a3b5e4-8d1e-11ee-b962-0242ac120002
course_id	10	UUID	Foreign key referencing the associated course (optional).	c6b2a1f6-8d1e-11ee-b962-0242ac120002

Table 4.40: quiz_learning table

This table records users' quiz learning activities, tracking the number of correct and incorrect answers for each learning content type within a course. It helps in assessing user progress and performance.

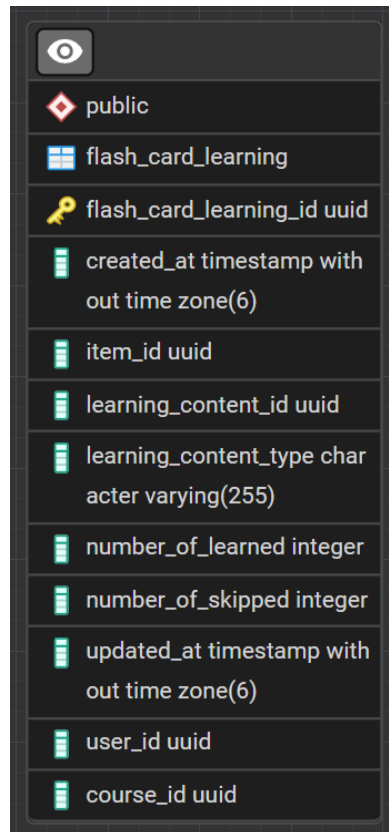


Figure 4.41: Flashcard Learning ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
flash_card_learning_id	1	UUID	Unique identifier for each flashcard learning session.	a1b2c3d4-e5f6-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the flashcard learning session was started.	2024-02-15 08:30:00
item_id	3	UUID	Foreign key referencing a specific learning item (word, sentence, etc.).	f9c5b7d2-8d1e-11ee-b962-0242ac120002
learning_content_id	4	UUID	Foreign key referencing the learning content item.	e8b1c6f0-8d1e-11ee-b962-0242ac120002
learning_content_type	5	VARCHAR(255)	Type of learning content (e.g., word, sentence, tense).	"word"
number_of_learned	6	INTEGER	The number of flashcards learned in the session.	10
number_of_skipped	7	INTEGER	The number of flashcards skipped in the session.	2
updated_at	8	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 09:00:00
user_id	9	UUID (Nullable)	Foreign key referencing the user learning the flashcards.	d7a3b5e4-8d1e-11ee-b962-0242ac120002
course_id	10	UUID (Nullable)	Foreign key referencing the associated course.	c6b2a1f6-8d1e-11ee-b962-0242ac120002

Table 4.42: flash_card_learning table

This table tracks users' learning activities using flashcards. It records how many flashcards were learned and skipped, helping measure user engagement and progress within a course.

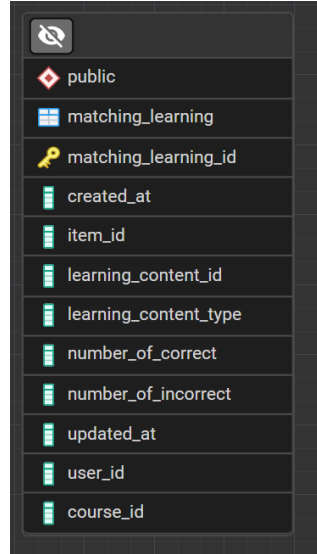


Figure 4.43: Matching Learning ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
matching_learning_id	1	UUID	Unique identifier for each matching learning session.	a1b2c3d4-e5f6-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the matching learning session was created.	2024-02-15 08:30:00
item_id	3	UUID	Foreign key referencing a specific learning item (word, sentence, etc.).	f9c5b7d2-8d1e-11ee-b962-0242ac120002
learning_content_id	4	UUID	Foreign key referencing the learning content item.	e8b1c6f0-8d1e-11ee-b962-0242ac120002
learning_content_type	5	SMALLINT	Type of learning content (e.g., word, sentence, tense).	1 (representing "word")
number_of_correct	6	INTEGER	Number of correct matches made by the user.	8
number_of_incorrect	7	INTEGER	Number of incorrect matches made by the user.	3
updated_at	8	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 09:00:00
user_id	9	UUID (Nullable)	Foreign key referencing the user who participated in the matching exercise.	d7a3b5e4-8d1e-11ee-b962-0242ac120002
course_id	10	UUID (Nullable)	Foreign key referencing the associated course.	c6b2a1f6-8d1e-11ee-b962-0242ac120002

Table 4.44: matching_learning table

This table stores users' learning progress for matching exercises, tracking the number of correct and incorrect matches made. It helps assess user performance and engagement in a language-learning course.

4.2.5 Notification & Notification Dispatcher

In the dispatcher service, the channel for notification are diverse, therefore we will provide each a channel a notification provider:

- SMS Notification (Twilio API)
- Mail (Java Mail Sender)
- Push Notification (Firebase Push Notification)

While the notification service is the place where all of the notifications are hold, the sender only have one mission and that is to send the notification through corresponding channel, and notify if the message was sent or not.

The asynchronous communicating mechanism is also utilized for this as the notification will be send by batch from Notification to Notification Dispatcher and will be implemented by AWS Simple Queue Service (SQS).

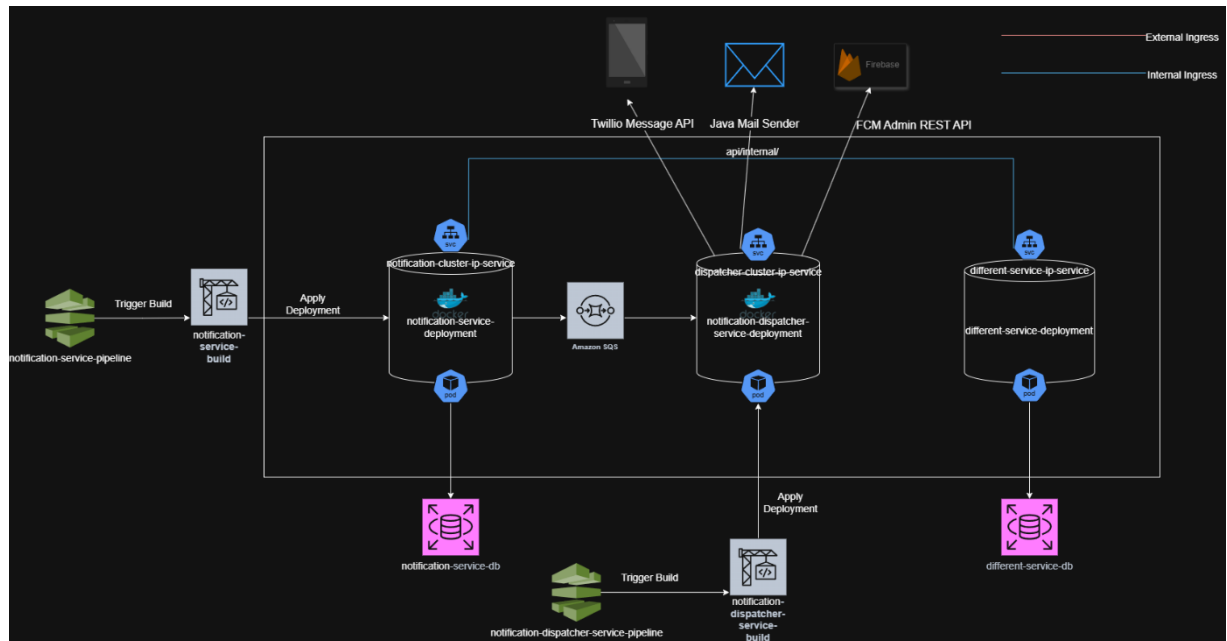


Figure 4.45: Notification and Notification Dispatcher Solution Design

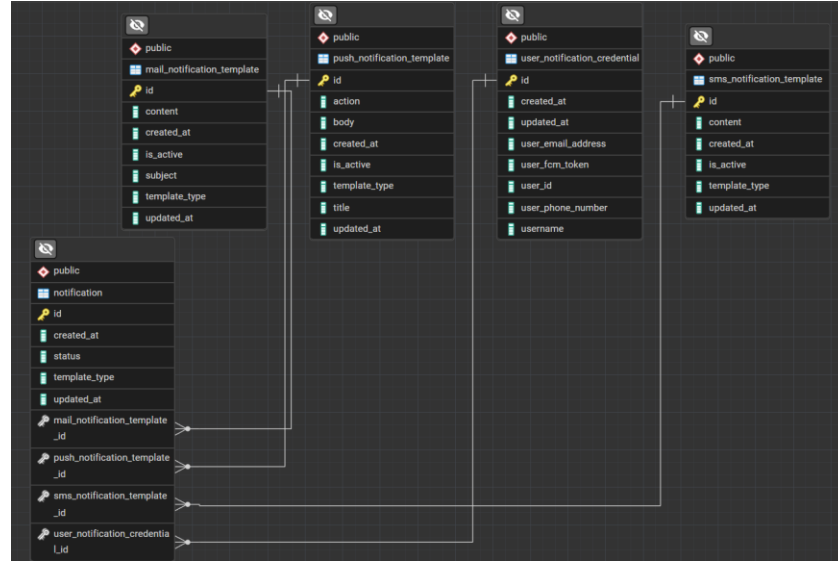


Figure 4.46: Notification ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
id	1	UUID	Unique identifier for the user notification credential record.	a1b2c3d4-e5f6-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the record was created.	2024-02-15 08:00:00
updated_at	3	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-15 09:00:00
user_email_address	4	TEXT	User's email address used for notifications.	user@example.com
user_fcm_token	5	TEXT	Firebase Cloud Messaging (FCM) token for push notifications.	fcm_token_123456789abcdef
user_id	6	UUID	Foreign key referencing the user associated with this record.	d7a3b5e4-8d1e-11ee-b962-0242ac120002
user_phone_number	7	TEXT	User's phone number used for SMS notifications.	+1234567890
username	8	VARCHAR(255)	User's display name or username.	john_doe

Table 4.47: user_notification_credential table

This table stores user notification credentials, including email, phone number, and FCM (Firebase Cloud Messaging) token. It is used to send notifications via different channels such as email, push notifications, and SMS.

Column Name	Ordinal Position	Data Type	Description	Example Value
id	1	UUID	Unique identifier for the mail notification template.	a1b2c3d4-e5f6-11ee-b962-0242ac120002
content	2	TEXT	The body content of the email template.	Hello {username}, welcome to our platform!
created_at	3	TIMESTAMP(6)	Timestamp when the template was created.	2024-02-15 08:00:00
is_active	4	BOOLEAN	Indicates whether the template is active or not.	true
subject	5	TEXT	The email subject line.	Welcome to Our Platform
template_type	6	VARCHAR(255)	Type/category of the email template.	welcome_email
updated_at	7	TIMESTAMP(6)	Timestamp when the template was last updated.	2024-02-15 09:00:00

Table 4.48: mail_notification_template table

This table stores email notification templates, which can be used for sending predefined emails to users. Each template includes content, subject, type, and an active status flag.

Column Name	Ordinal Position	Data Type	Description	Example Value
id	1	UUID	Unique identifier for the push notification template.	d7a3b5e4-8d1e-11ee-b962-0242ac120002
action	2	VARCHAR(255)	The action associated with the notification.	open_app
body	3	TEXT	The message content of the push notification.	You have a new message!
created_at	4	TIMESTAMP(6)	Timestamp when the template was created.	2024-02-15 08:30:00
is_active	5	BOOLEAN	Indicates whether the template is active or not.	true
template_type	6	VARCHAR(255)	Type/category of the push notification template.	message_alert
title	7	TEXT	The title of the push notification.	New Message
updated_at	8	TIMESTAMP(6)	Timestamp when the template was last updated.	2024-02-15 09:30:00

Table 4.49: push_notification_template table

This table stores push notification templates, which can be used for sending predefined push notifications to users. Each template includes a title, body content, type, action, and an active status flag.

Column Name	Ordinal Position	Data Type	Description	Example Value
id	1	UUID	Unique identifier for the SMS notification template.	b3f8e6d2-9a1b-11ee-b962-0242ac120002
content	2	TEXT	The message content of the SMS notification.	Your verification code is 123456.
created_at	3	TIMESTAMP(6)	Timestamp when the template was created.	2024-02-15 10:00:00
is_active	4	BOOLEAN	Indicates whether the template is active or not.	true
template_type	5	VARCHAR(255)	Type/category of the SMS notification template.	otp_verification
updated_at	6	TIMESTAMP(6)	Timestamp when the template was last updated.	2024-02-15 11:00:00

Table 4.50: sms_notification_template table

This table stores SMS notification templates, which can be used for sending predefined text messages to users. Each template includes message content, type, an active status flag, and timestamps for creation and updates.

Column Name	Ordinal Position	Data Type	Description	Example Value
id	1	UUID	Unique identifier for the notification.	c7e4a1d2-9a1b-11ee-b962-0242ac120002
created_at	2	TIMESTAMP(6)	Timestamp when the notification was created.	2024-02-15 12:00:00
status	3	VARCHAR(255)	Status of the notification (e.g., sent, failed, delivered).	sent
template_type	4	VARCHAR(255)	Type of notification (email, push, or sms).	sms
updated_at	5	TIMESTAMP(6)	Timestamp when the notification was last updated.	2024-02-15 12:05:00
mail_notification_template_id	6	UUID	Reference to the mail_notification_template table (if applicable).	NULL
push_notification_template_id	7	UUID	Reference to the push_notification_template table (if applicable).	NULL
sms_notification_template_id	8	UUID	Reference to the sms_notification_template table (if applicable).	a2b3c4d5-9a1b-11ee-b962-0242ac120002
user_notification_credential_id	9	UUID	Reference to the user_notification_credential table (recipient user).	d4e5f6g7-9a1b-11ee-b962-0242ac120002

Table 4.51: notification table

This table stores records of notifications sent to users. It tracks the notification status, template type, associated templates (mail, push, or SMS), and the user's notification credentials. This helps manage and monitor notifications effectively.

4.1.6 Subscription Service

For the design of Subscription service, this will be the service where all of the information of subscriptions come from, plus this is the source of truth when it comes to payment information of User.

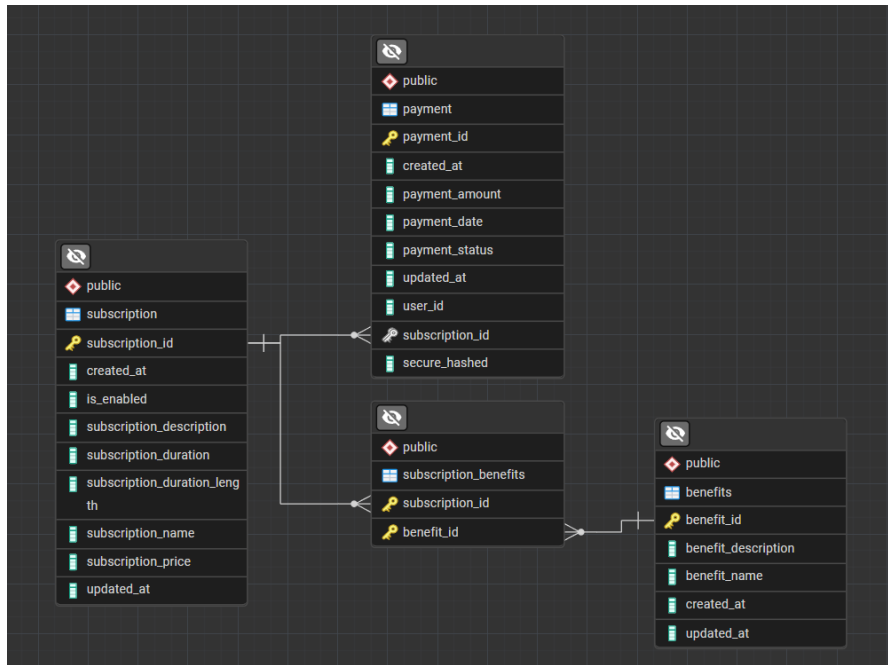


Figure 4.52: Subscription ERD Design

Column Name	Ordinal Position	Data Type	Description	Example Value
benefit_id	1	UUID	Unique identifier for the benefit.	e1a2b3c4-d5f6-7890-1234-56789abcdef0
benefit_name	2	VARCHAR(255)	Name of the benefit.	Premium Support
benefit_description	3	VARCHAR(255)	Description of the benefit.	24/7 priority customer support
created_at	4	TIMESTAMP(6)	Timestamp when the benefit was created.	2024-02-15 10:30:00
updated_at	5	TIMESTAMP(6)	Timestamp when the benefit was last updated.	2024-02-16 14:45:00

Table 4.53: benefits table

The benefits table stores information about different benefits that may be associated with a product, service, or user subscription. Each benefit has a unique identifier, name, description, and timestamps to track its creation and updates.

Column Name	Ordinal Position	Data Type	Description	Example Value
subscription_id	1	UUID	Unique identifier for the subscription.	f47ac10b-58cc-4372-a567-0e02b2c3d479
subscription_name	2	VARCHAR(255)	Name of the subscription plan.	Premium Plan
subscription_description	3	VARCHAR(255)	Description of the subscription.	Access to premium features and content
subscription_price	4	DOUBLE PRECISION	Price of the subscription.	9.99
subscription_duration	5	INTEGER	Duration of the subscription (numeric value).	12
subscription_duration_length	6	VARCHAR(255)	Unit of duration (e.g., days, months, years).	months
is_enabled	7	BOOLEAN	Indicates if the subscription is currently active.	TRUE
created_at	8	TIMESTAMP(6)	Timestamp when the subscription was created.	2024-02-15 10:30:00
updated_at	9	TIMESTAMP(6)	Timestamp when the subscription was last updated.	2024-02-16 14:45:00

Table 4.54: subscription table

The subscription table stores information about different subscription plans available to users. Each subscription has a unique ID, name, description, price, duration, and status (enabled/disabled). It also tracks creation and update timestamps.

Column Name	Ordinal Position	Data Type	Description	Example Value
subscription_id	1	UUID	Foreign key referencing subscription_id.	f47ac10b-58cc-4372-a567-0e02b2c3d479
benefit_id	2	UUID	Foreign key referencing benefit_id.	e1a2b3c4-d5f6-7890-1234-56789abcdef0

Table 4.55: subscription_benefits table

The subscription_benefits table establishes a many-to-many relationship between subscriptions and benefits. Each subscription can have multiple benefits, and each benefit can be associated with multiple subscriptions.

Column Name	Ordinal Position	Data Type	Description	Example Value
payment_id	1	VARCHAR(255)	Unique identifier for the payment	TXN123456789
user_id	2	UUID	Foreign key referencing the user who made the payment.	550e8400-e29b-41d4-a716-446655440000
subscription_id	3	UUID	Foreign key to the subscription purchased.	f47ac10b-58cc-4372-a567-0e02b2c3d479
payment_amount	4	DOUBLE PRECISION	The amount paid.	9.99
payment_date	5	TIMESTAMP(6)	The date and time when the payment was processed.	2024-02-15 10:30:00
payment_status	6	VARCHAR(255)	Status of the payment (e.g., pending, completed, failed).	completed
secure_hashed	7	TEXT	A hashed value for security or verification purposes.	a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
created_at	8	TIMESTAMP(6)	Timestamp when the record was created.	2024-02-15 10:30:00
updated_at	9	TIMESTAMP(6)	Timestamp when the record was last updated.	2024-02-16 14:45:00

Table 4.56: payment table

The payment table stores details of transactions made by users for subscriptions or other purchases. It keeps track of payment amounts, statuses, timestamps, and associated user and subscription information.

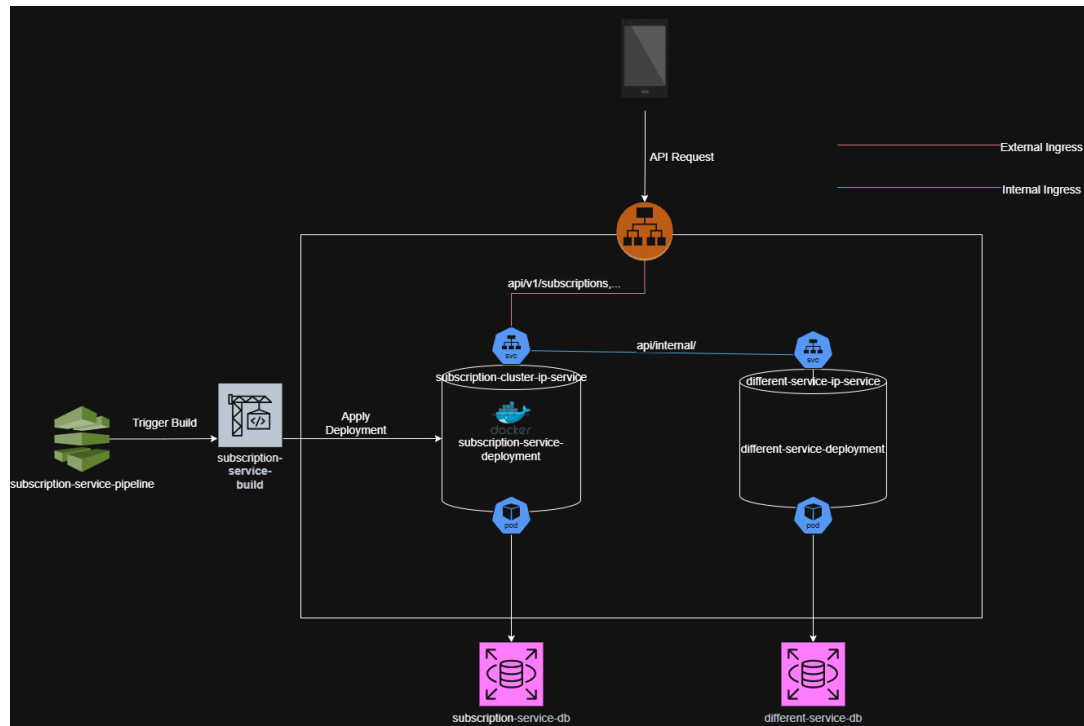


Figure 4.57: Subscription Service Solution Design

4.2.7 Achievement Service

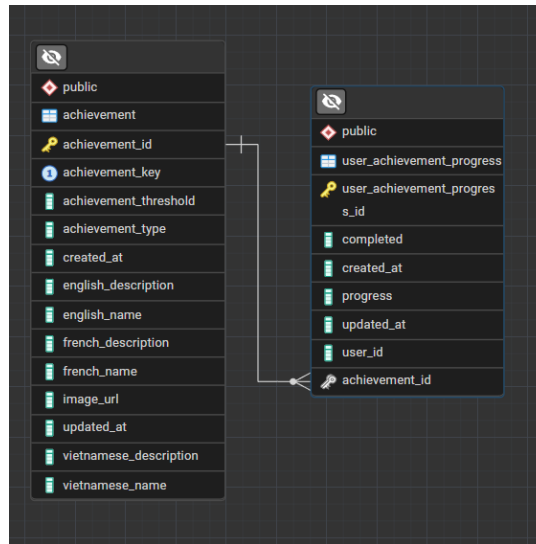


Figure 4.58: User Achievement ERD Design

Column Name	Data Type	Description	Example Value
user_achievement_progress_id	UUID	Unique identifier for the user progress entry.	123e4567-e89b-12d3-a456-426614174000
user_id	UUID	Foreign key referencing users(user_id).	550e8400-e29b-41d4-a716-446655440000
achievement_id	UUID	Foreign key referencing achievement(achievement_id)	f47ac10b-58cc-4372-a567-0e02b2c3d479
progress	INTEGER	The user's current progress.	3
completed	BOOLEAN	Indicates if the achievement is completed.	false
created_at	TIMESTAMP(6)	Timestamp when progress was recorded.	2024-02-15 10:30:00
updated_at	TIMESTAMP(6)	Timestamp when progress was last updated.	2024-02-

Table 4.59: user_achievement_progress table

This table tracks users' progress towards earning achievements.

Column Name	Data Type	Description	Example Value
achievement_id	UUID	Unique identifier for the achievement.	550e8400-e29b-41d4-a716-446655440000
achievement_key	VARCHAR(255)	A unique key for identifying the achievement.	FIRST_PURCHASE
achievement_threshold	INTEGER	The required progress value to unlock the achievement.	5
achievement_type	VARCHAR(255)	Type/category of achievement.	purchase
image_url	VARCHAR(255)	URL of the achievement image.	https://example.com/achievement1.png
english_name	VARCHAR(255)	Achievement name in English.	First Purchase
english_description	VARCHAR(255)	Description in English.	Complete your first purchase.
french_name	VARCHAR(255)	Achievement name in French.	Premier Achat
french_description	VARCHAR(255)	Description in French.	Effectuez votre premier achat.
vietnamese_name	VARCHAR(255)	Achievement name in Vietnamese.	Mua Hàng Đầu Tiên
vietnamese_description	VARCHAR(255)	Description in Vietnamese.	Hoàn thành lần mua hàng đầu tiên.
created_at	TIMESTAMP(6)	Timestamp when the achievement was created.	2024-02-15 10:30:00
updated_at	TIMESTAMP(6)	Timestamp when the achievement was last updated.	2024-02-16 14:45:00

Table 4.60: achievement table

The achievement table stores different achievements that users can earn. It includes multilingual descriptions and an `achievement_threshold` defining the requirement for completion.

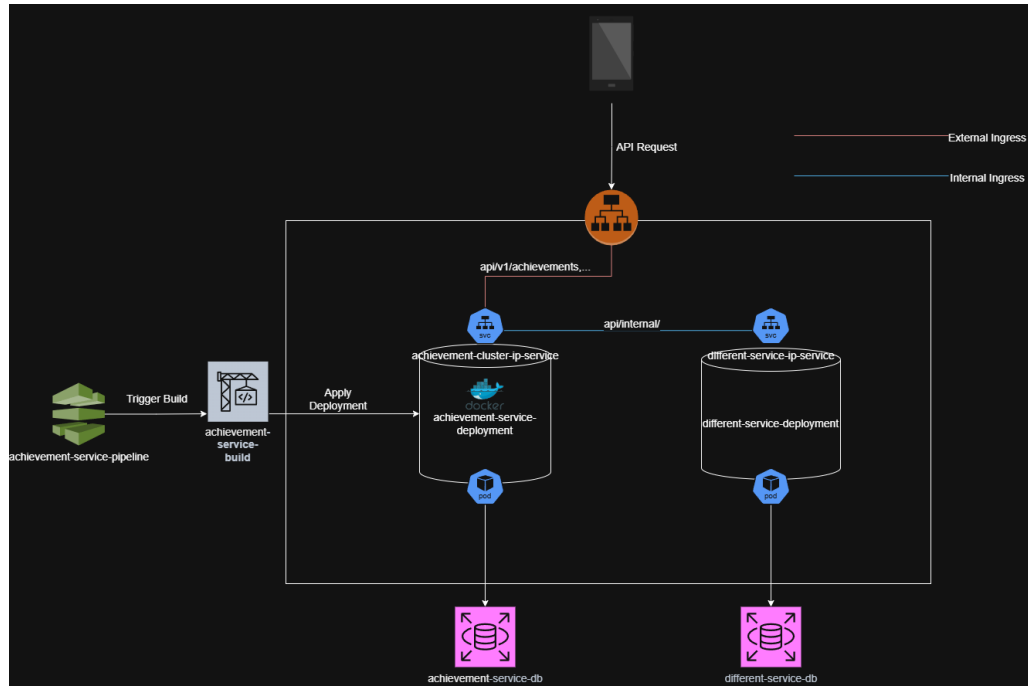


Figure 4.61: Achievement Service Solution Design

4.2.8 Community Service

The community service will aim to solve three use cases:

- AI Chat bot session
- Group chat session
- Private 1-on-1 chat

Therefore, we are going to need to locate session for each of the chat type, each chat will have multiple messages and each of the message belongs to a user.

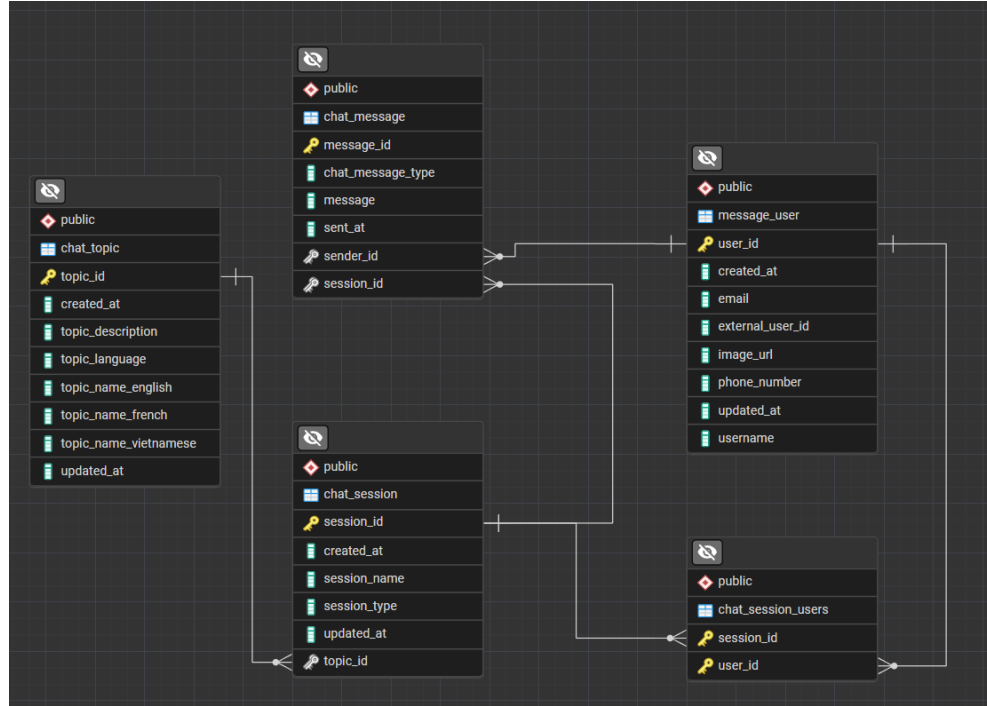


Figure 4.62: Community Service ERD Design

Column Name	Data Type	Description	Example Value
session_id	UUID	Unique identifier for the chat session	550e8400-e29b-41d4-a716-446655440000
session_name	VARCHAR(255)	Name of the chat session	"General Chat"
session_type	SMALLINT	Type of session (e.g., private, group)	1 (1 = private, 2 = group)
created_at	TIMESTAMP(6)	Timestamp when the session was created	2024-02-15 10:30:00
updated_at	TIMESTAMP(6)	Timestamp of the last update	2024-02-15 10:45:00
topic_id	UUID	ID of the related topic (if applicable)	f3d9a2b5-3f76-48c1-a13a-50c2c0b5d276

Table 4.63: chat_session table

The session_type column is important for distinguishing between different kinds of sessions (like group chats or private messages). The topic_id links this session to a particular topic, if applicable.

Column Name	Data Type	Description	Example Value
session_id	UUID	References the chat session that the user is part of. This column is linked to the chat_session table.	550e8400-e29b-41d4-a716-446655440000
user_id	UUID	Unique identifier for the user in the chat session. It links to a specific user in the message_user table.	f4a3b9d5-9c2a-46d8-b8d4-123456789abc

Table 4.64: chat_session_users table

This table serves as a many-to-many relationship between chat_session and message_user. It stores which users belong to which sessions. It enables the system to track users in both private and group chats.

Column Name	Data Type	Description	Example Value
user_id	UUID	Unique identifier for the user. This links to users in the message_user table.	f4a3b9d5-9c2a-46d8-b8d4-123456789abc
username	VARCHAR(255)	The username displayed for the user in the chat. It can be used to identify the user in the system.	"JohnDoe"
email	VARCHAR(255)	The user's email address. This is a unique field for identifying users in the system.	"johndoe@example.com"
phone_number	VARCHAR(255)	The user's phone number. Like email, this is unique and important for contact purposes.	" +1234567890"
image_url	VARCHAR(255)	The URL pointing to the user's profile picture, if available. This allows for visual customization in chat interfaces.	"https://example.com/john.jpg"
external_user_id	VARCHAR(255)	Optional field for storing the ID of the user in external systems (if applicable).	"EXT-USER-001"
created_at	TIMESTAMP(6)	Timestamp when the user profile was created in the system. This can be used for auditing.	2024-02-15 09:30:00
updated_at	TIMESTAMP(6)	Timestamp of the last update made to the user's profile. This helps keep track of user activity.	2024-02-15 10:30:00

Table 4.65: message_user table

The message_user table stores the profile data of each user involved in messaging. The combination of user_id, username, and email ensures each user is uniquely

identified. The `external_user_id` is useful for integration with external services (e.g., social login).

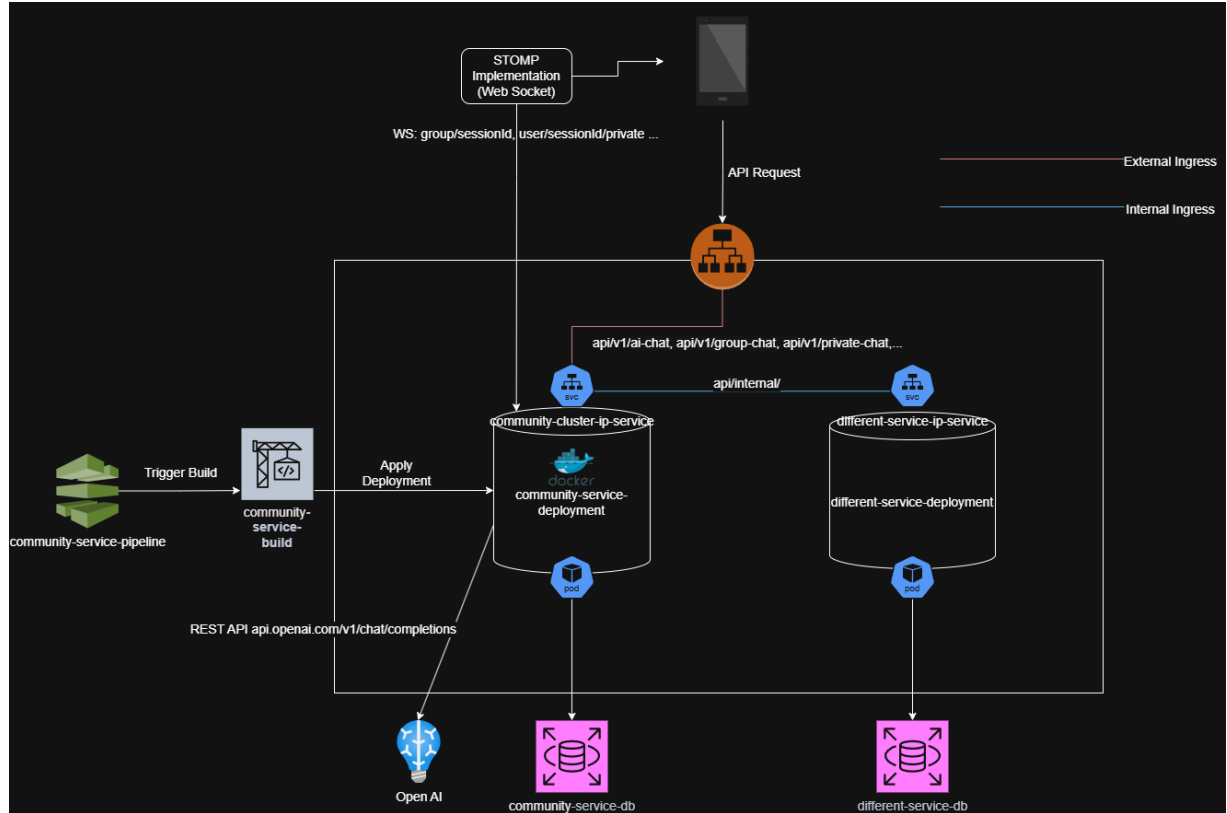


Figure 4.66: Community Service Solution Design

4.3 Security Design

This section outlines the security measures implemented in our project both on back-end and front-end to protect against various threats such as network attacks and data breaches.

4.3.1 Authentication and Authorization

For authentication purposes, we will have the following measurement of security:

- **User Authentication:** The application uses JWT (JSON Web Tokens) for secure user authentication. When a user logs in, the backend generates a token that is sent to the frontend (Flutter mobile app). The token is then

used for subsequent requests to authenticate the user for restricted resources.

- **Role-Based Authorization:** The system follows a Role-Based Access Control (RBAC) model. Users are assigned roles (e.g., subscribed-user, user) upon login, and the system checks their roles to determine access to specific resources or actions. This ensures that users can only access the data and features they are authorized to use, such as the community services after subscription.

4.3.2 Preventive Measurements Against Attacks

SQL Injection Prevention: The system uses parameterized queries and prepared statements in the Spring Boot microservices to prevent SQL injection attacks. This ensures that user input is properly validated and sanitized before being executed in the database.

The API Keys are store securely on Kubernetes secrets and can not be accessed without the correct permissions given by IAM, we don't want to store the keys on the front-end as it could be intercepted by reverse engineering.

The system also deploy the service of AWS CloudWatch to see the explicit activities of different deployment, to watch out for services that have unusual number of requests.

CHAPTER 5: DEVELOPMENT AND TESTING

In this chapter, we are going to take a focus on the scope of development of our system, which include:

- Frameworks
- Development Environment (Tools, Integrated Development Environment, Command line interface ...)
- Project Structure
- Deployment Pipeline

5.1 Frameworks

This is going to be a primarily mobile application that will be developed and test on Android and iOS, the frameworks are:

- Flutter for cross-platform development
- Kotlin Spring Boot for back-end development
- AWS for cloud deployment and services related to pipeline, network infrastructure:
 - Elastic Container Registry
 - Elastic Kubernetes Service
 - Simple Queue Service
 - IAM
 - CodePipeline
 - CodeBuild
 - CloudWatch ...
- Firebase for logging, monitoring, notification, app testing on Android ...

5.2 Development Environment

These are the development setup from personal computer to start the development of the project:

- Android Studio (The latest version)
- Java JDK (17)
- Flutter SDK (3.27.0)
- IntelliJ (The latest version)
- Git Command Line Interface
- Github
- AWS Command Line Interface (aws)
- Kubernetes Command Line Interface (kubectl)
- Firebase Project
- pgAdmin for database administration
- Docker
- Xcode (for MacOS)
- SonarQube for code quality assessment on local

5.3 Project Structure

To keep the project well-organized and allow independent development of different features, the codebase is structured into multiple **Flutter packages and modules** contains different content in the layer of clean architecture, alongside with that, there will be some helper module such as localization and services to provide utilities functionalities.

All of these different modules will be link together through a module management library called melos. There are in 7 modules in total:

- app (Presentation Layer, main entryptpoint of the app)
- data (Data Layer)
- logic (Domain Layer)
- initializer (provide initialization logic for all of the modules)
- localizaiton (provide localization for text resources)

- services (provide utilities services)
- design (provide common design resources used throughout the app, e.g. colors, button, input ...)

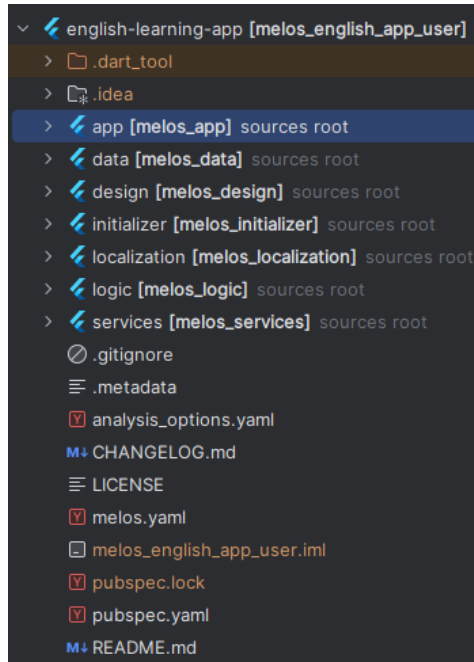


Figure 5.1: Mobile Project Structure

For the state management of the project, we decided to go with the Flutter provided `ChangeNotifier` in combination with the dependencies inject of `Provider` package, and the structure of the app layer will follow a `Model-View-ViewModel` as it is one of the most used architecture in mobile development due to its manageability, easy-to understand principal.

For the back-end service, each service will have three layers of `Controller-Service-Repository`. With some different folders such as `config` to provide configuration for the service, `entities` for data entities, `model` for handling response and request data ...

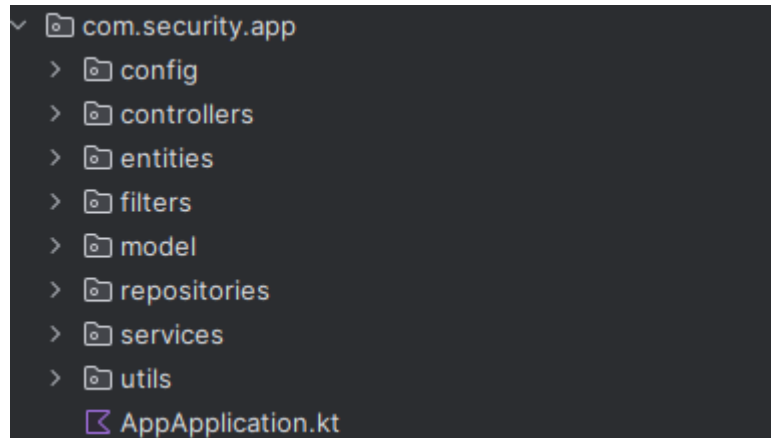


Figure 5.2: Back-end Service Architecture

Beside, each service will have a pipeline that will be triggered on the commit into the main branch, so if we look at the below structure:

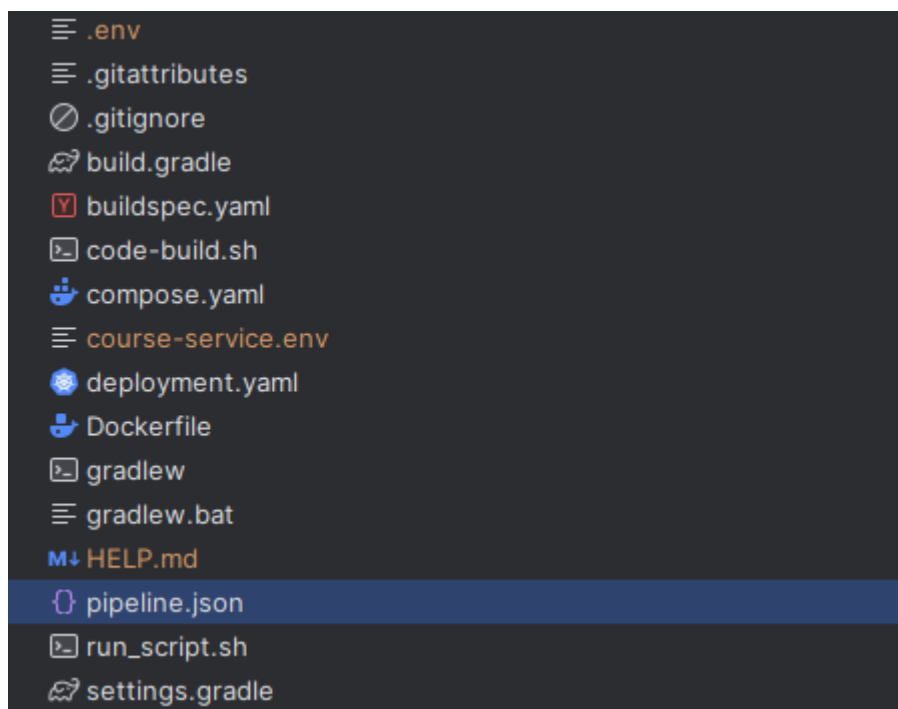


Figure 5.3: Helper Script and Configuration

- The .env files: Secrets of the service both on local and on cloud
- pipeline.json: Configuration of the pipeline
- deployment.yaml: Store the configuration of the deployment

- buildspect.yaml: Step for creating builds, upload to Container Registry and deploy on Kubernetes Service
- code-build.sh: Command line to create CodeBuild project on AWS
- The rest are configuration for local deployment to Docker

5.4 Deployment Pipeline

For the deployment of the backend services, upon any git commit into the main branch, the pipeline will start a Code Build run to build and upload the image into AWS Container Registry, after that, it will apply the deployment.yaml file to deploy the instance on to AWS Kubernetes Service

<input type="radio"/>	community-service-pipeline (Type: V2 Execution mode: SUPERSEDED)	✓ Succeeded	SourceAction – 84aa365c init pipeline	3 days ago	✓ ✗ View details
<input type="radio"/>	achievement-service-pipeline (Type: V2 Execution mode: SUPERSEDED)	✓ Succeeded	SourceAction – f4686273 update print	3 days ago	✓ ✓ ✗ ✗ View details
<input type="radio"/>	subscription-service-pipeline (Type: V2 Execution mode: SUPERSEDED)	✓ Succeeded	SourceAction – c72d5c25 init pipeline	3 days ago	✓ ✗ View details
<input type="radio"/>	course-service-pipeline (Type: V2 Execution mode: SUPERSEDED)	✓ Succeeded	SourceAction – 84ec95c3 update printing	3 days ago	✓ ✓ ⌂ ✗ View details

Figure 5.4: AWS Pipeline Runs

achievement-service-pipeline

Source

Build

BuildAction AWS CodeBuild

Pipeline execution details

BuildAction

Logs

Summary

Input

Output

Status

✓ Succeeded

Action execution ID

2fa47ae8-2cc6-496d-83f6-b86f1ce05a2e

Message

-

Execution details

[View in CodeBuild](#)

Figure 5.5: Successful Code Build in Pipeline

The same pipeline instance on the mobile app repository will be handle the process of building an apk and upload it into Firebase App Tester for our group to run test on them.

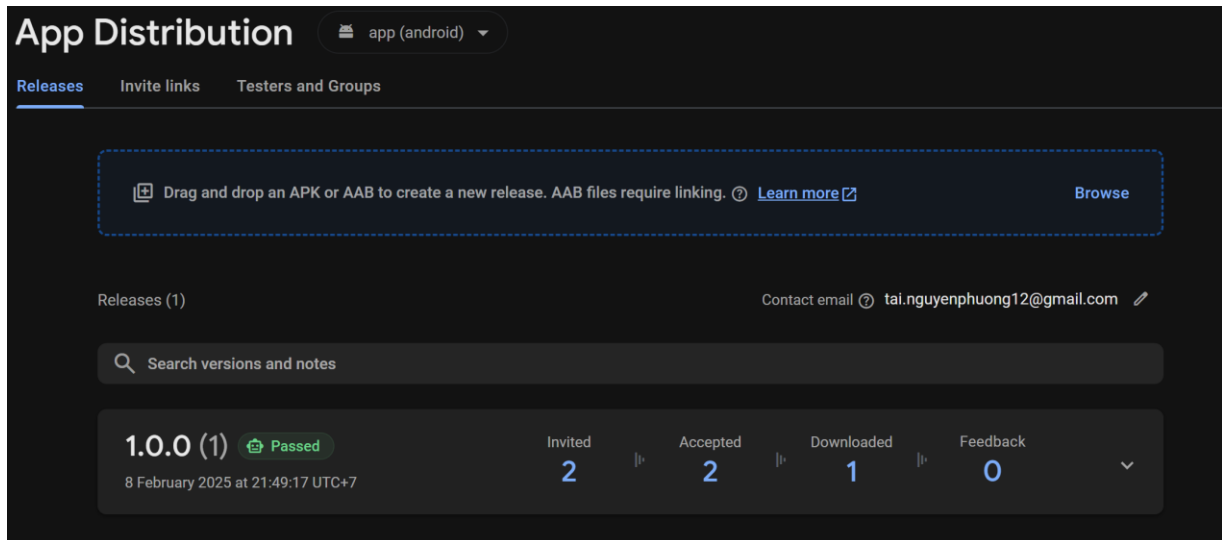


Figure 5.6: Firebase App Distribution

5.5 Implementation Result

These are the important pages have been implemented inside our project, not every screen will be making an entrance, just the notable ones.

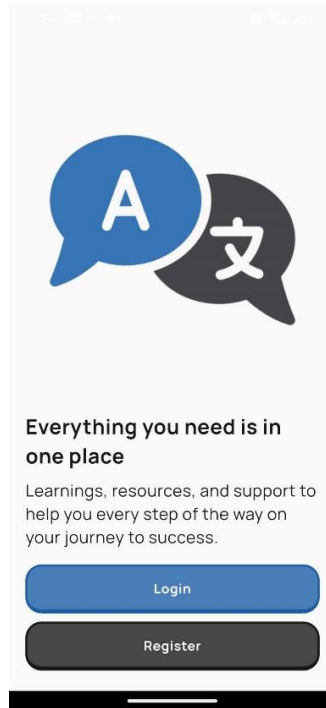


Figure 5.7: Introduction Screen

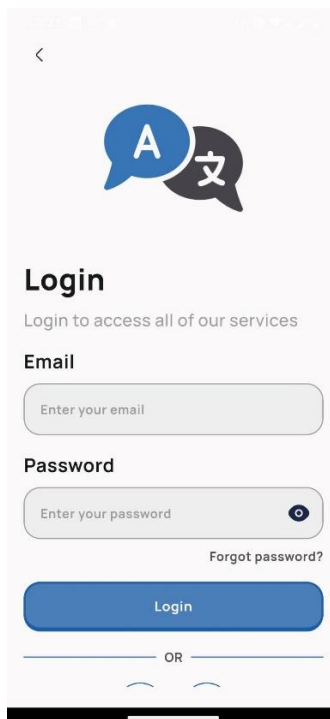
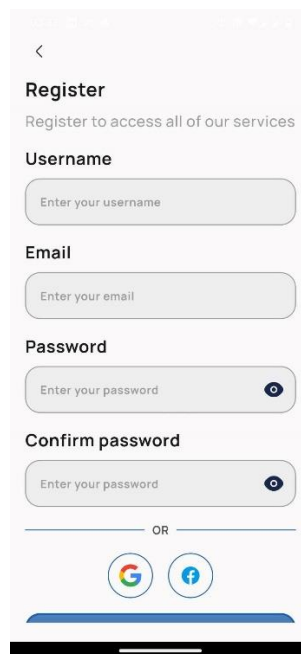


Figure 5.8: Login Page



The registration page features a clean, light gray background. At the top, there's a back arrow and a status bar. The main heading is "Register" in bold, followed by a subtext "Register to access all of our services". Below this are four input fields: "Username", "Email", "Password", and "Confirm password", each with a placeholder text "Enter your...". The password fields have an eye icon to toggle visibility. At the bottom, there's an "OR" separator and two circular icons for Google and Facebook login.

Figure 5.9: Registration Page



The home page has a light gray background. At the top, there's a status bar and a welcome message "Welcome, Phuongtai0542" with a user icon. Below this is a section titled "Choose what to learn today ?" with a subtext "Choose the language". There are three language selection buttons: "English" with a UK flag, "Vietnam..." with a Vietnamese flag, and "Fren" with a French flag. Below the language selection is a "Community Hub" section with a message "You have joined a community! Start learning with other learners." and a "Join community" button. At the bottom, there's a "My in-progress courses" section with two course cards: "English" (Level: A1) and "Vietna..." (Level: A2). The bottom navigation bar includes icons for Home, Courses, AI Chatbot, To do, and Profile.

Figure 5.10: Home Page



Figure 5.11: Category Course

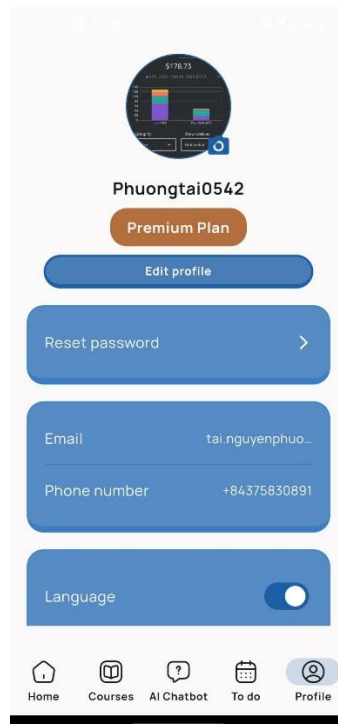


Figure 5.12: Profile Page

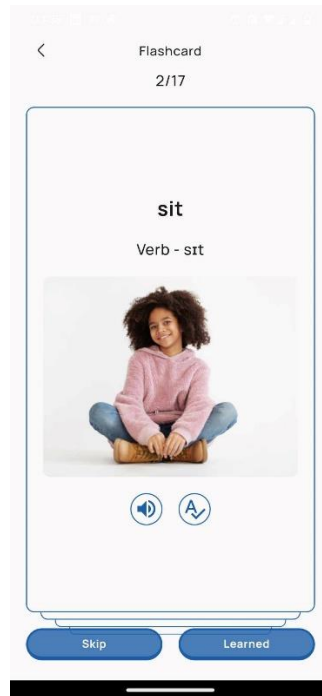


Figure 5.13: Flashcard Learning



Figure 5.14: Quiz Learning

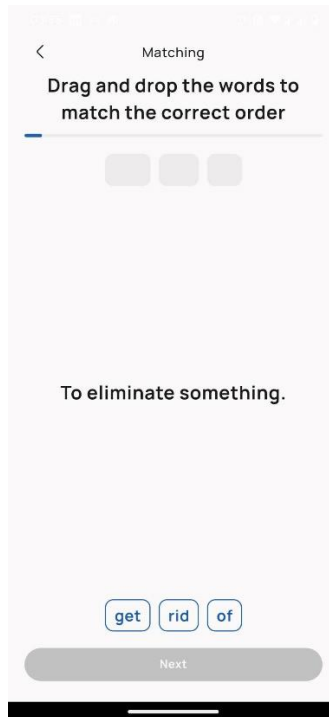


Figure 5.15: Matching Learning

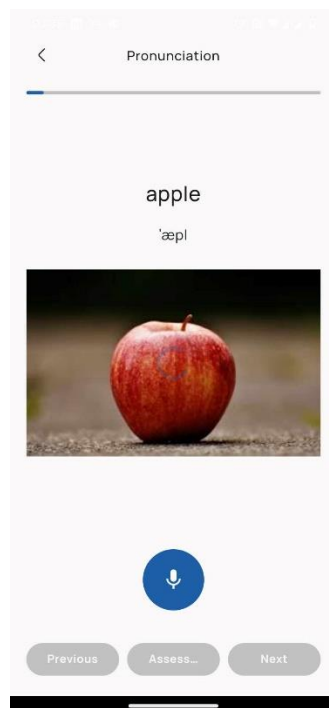


Figure 5.16: Pronunciation Learning

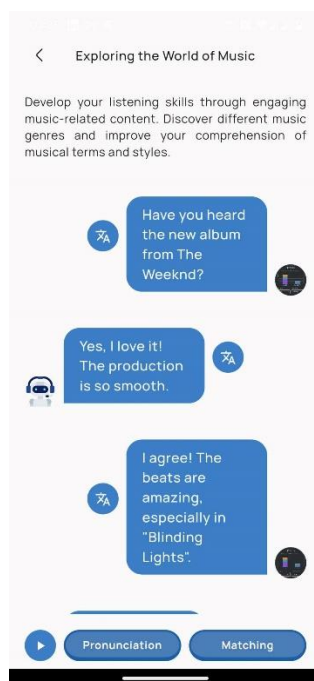


Figure 5.17: Listening Paragraph



Figure 5.18: Community Page

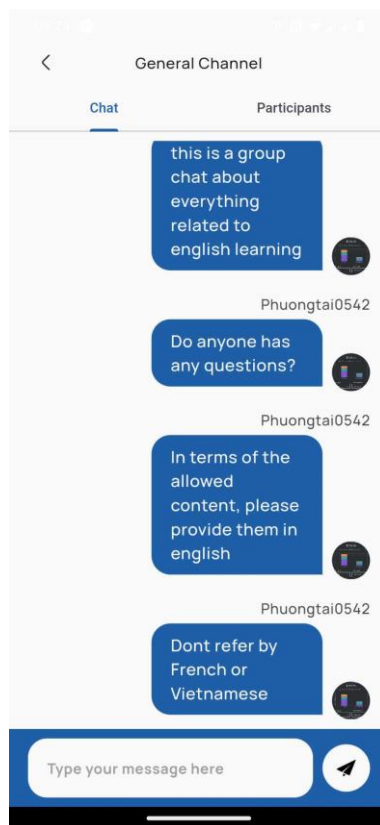


Figure 5.19: Chat Session

CHAPTER 6: DEPLOYMENT AND EVALUATION

This chapter focuses on the testing activities performed to ensure that the our application on mobile and back-end functions correctly and remains stable. Testing is not only for detecting errors but also for verifying that the system meets the defined requirements.

6.1 Testing Strategy

The system follows a structured testing approach to ensure software reliability. Currently, unit testing is implemented to validate individual components of the application.

Unit testing:

- Each module is tested in isolation to verify that it behaves as expected.
- Mocking is used to isolate dependencies and test business logic.
- In Flutter, `flutter_test` and `mocktail` are used for writing and executing unit tests, including validating use cases, model format, entity format, repository testing.
- In the backend (Kotlin Spring Boot), JUnit is used for testing services and repositories.
- It is integrated in the pipeline of each service and inside the application

Crash tracking:

- The application is currently tracked by Firebase Crashlytics, the goal of Crashlytics is to logs the instance when the app crashes and the error that comes with it.

Integration testing for future implementation:

- Integrate Crashlytics with a message tool, such as Slack or Jira to log all of the crash errors in real-time tracking.

- Tools like Testcontainers for PostgreSQL and WireMock for simulating external APIs may be used.

System testing for future implementation:

- Will cover end-to-end testing to validate the complete application workflow.
- Possible tools: Postman for API testing, Selenium/Appium for UI testing.

6.2 Issues Tracking

Bugs and issues are tracked using GitHub Issues, allowing the team to manage and prioritize fixes efficiently. Each issue follows a structured reporting format, including:

- Description of the bug
- Steps to reproduce
- Expected vs. actual behavior
- Assigned developer and status updates

6.3 Testing Evaluation

Since testing is an ongoing process, future improvements will focus on:

- Expanding test coverage with **integration and end-to-end tests**.
- Automating more testing procedure to have a clear overview before going to deployment.
- Collecting user feedback through logs testing to refine the application availability and usability for the users.

CHAPTER 7: CONCLUSION

7.1 Achieved Result

The project has been implemented and deployed thoroughly compared to the initial plan for the application. It has provided the users with adequate learning experience with different models and allow users to connect with eachother through community features such as group or private chat, seeking help from AI Chat, they can also revise the progress by visiting the progress section and manage their favorite courses. It has become a fully functional app with most of the essential features for learning, engaging and getting hooked upon entering the app. We have achieved the desired goal of creating a language learning application that has most of the features we planned out to achieve.

The app is ready to be commercialized, if wished, and would provide users with a pleasant learning experience, we are providing features that may not rise up to the standards of Duolingo or Memrise, but we have create a comprehensive learning application.

But the system has some flaws that we might need to address for this large of infrastructure, such as the lack of testing between services, most of the progress currently is testing on the go, and we can only evaluate the useability of a service through the deployment of it on the cloud, we have not had a concrete testing procedure for microservices. The other thing is the coverage of unit testing, due to the large scale of the project, plus the shortage of manpower (2 people) has made the process of integrating testing more difficult.

7.2 Future Direction

For now the app has some of the following goals for development and business ideas:

- Integrate fully self-designed assets, such as images, icons for commercialization.

- Fully expand the features for different sections, such as community to have better quality of chat optimization, security (end to end encryption).
- Increase coverage of unit testing on each service and mobile app, integrate testing for microservice, architecture.
- Improve security measurements, such as database source of truth should be secured and can only be accessed by the services, and super admin.
- Move the support for mobile app into web, and desktop applications to reach to the wider audiences.
- Integrate SonarQube to ensure code quality on the deployment pipeline, tracking the unit testing coverage before uploading to ensure the services are thoroughly tested by developers first.
- Create more meaningful analytics with the migration from Firebase Analytics to MoEngage or Segment, create more comprehensive graph to see the users' behavior.
- Dashboard for easier interaction with database
- Research and development in the service of pronunciation assessment to give users the most accurate output, rather than having to rely on a third party API that can go out of production anytime.
- Better management of different courses for users, currently, users can only bookmark and put into cache, but we need to give users to options to organize their courses into different folders.
- Better management for learning, currently users does not have any mean to limit the number of questions they have to take in the learning process ,making them learn everything is not resourceful, moreover the system currently does not have a progress tracking for the progress of users of taking, therefore, there might be a case where they proceed to the last question and misclicked back, all of their progress will be gone.

REFERENCES

English

- [1]. Marco L. Napoli [2019], Beginning Flutter: A Hands On Guide To App Development, 1st Edition, Wrox Press, New Jersey.
- [2]. Eric Windmill [2019], Flutter in Action, 1st Edition, Manning Publications, New York.
- [3]. Prajyot Mainkar, Salvatore Giordano [2019], Google Flutter Mobile Development Quick Start Guide, 1st Edition, Packt, United Kingdom.
- [4]. Ivo Balbaert, Dzenan Ridjanovic [2015], Learning Dart, 2nd Edition, Packt, United Kingdom.
- [5]. Gilad Bracha [2015], The Dart Programming Language, Addison-Wesley, Boston.