System and Software Architecture Description (SSAD)

LINGGGO

Team 3

Chicheng Ren – Software Architect

Dahai Li – Quality Focal Point

Dashun Wen – Life Cycle Planner

Kraingkrai Bumroungruksa – Prototyper

Siming Ye – Feasibility Analyst

Shiqi Wei – Operational Concepts Engineer

Yiting Xiao – Project Manager

11/30/15

Version History

Date	Author	Version	Changes made	Rationale
10/17/15	KB	1.0	• Documented sections 1-2	• Initial draft for use with Instructional ICM-Sw v1.0
10/19/15	DL	1.1	• Updated Users to Language Learners	• Change to consistent stakeholder names
11/30/15	CR	1.2	• All document sections	• Final draft

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

The purpose of this document is to provide the whole picture and a deep understanding of LINGGGO's behavior and functionalities. It shows all of the actors interacting with the system. Also, all important system artifacts have been shown in detail.

1.2 Status of the SSAD

The current version contains the system's behavior, actor, and functionalities.

2. System Analysis

2.1 System Analysis Overview

The main purpose of LINGGGO is to enable Language Learners who want to learn other languages by practicing with native speakers of that particular language. The system matches language learners according to their language preferences. For example, English speakers who want to learn Chinese language will be matched to Chinese speakers who want to learn English. Then, the users(language learners) can send messages to each other and decide which communication tool is comfortable with them(e.g. Skype).

2.1.1 System Context

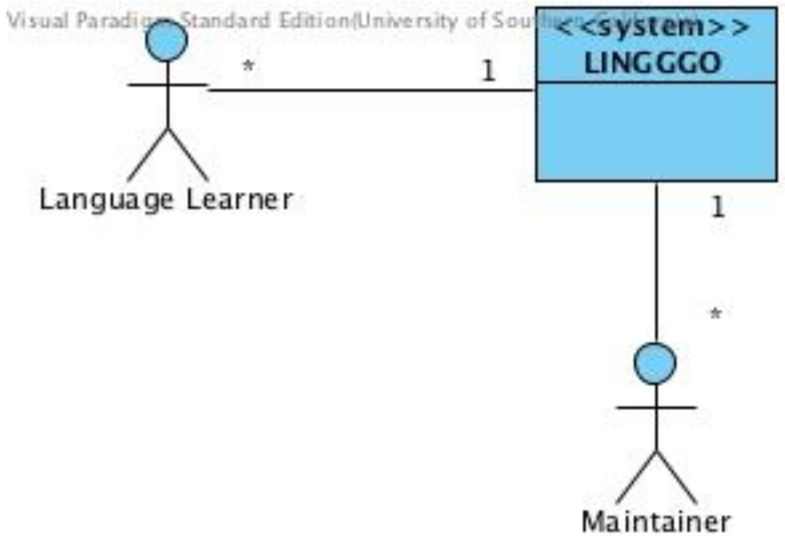

Visual Paradigm Standard Edition(University of Southern California)
<<system>>
LINGGGO
*
1
Language Learner
1
*
Maintainer

Figure 1: System Context Diagram

Table 1: Actors Summary

Actor	Description	Responsibilities
Language Learner	Language learners use the system to find other languages learners to learn/teach new languages	• Create profile and set language preferences • Contact other language learners that they are interested

2.1.2 Artifacts & Information

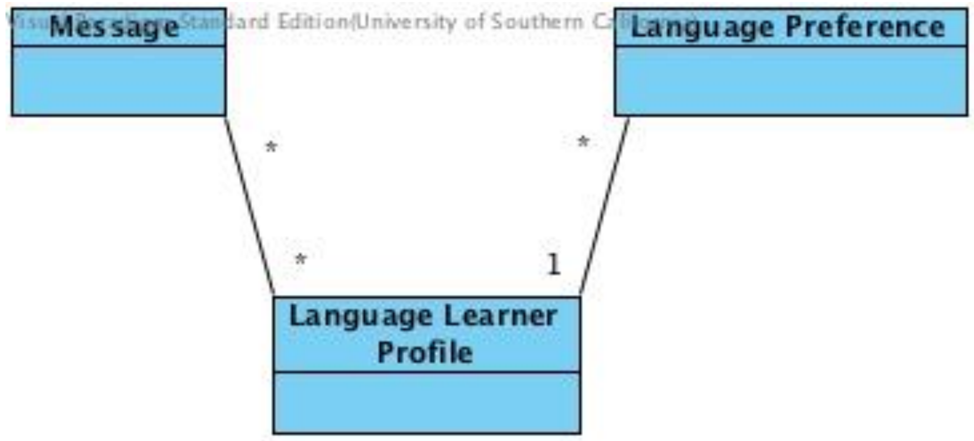

Message
Language Preference
Language Learner Profile
*
*
*
1
Visual Paradigm Standard Edition(University of Southern California)

Figure 2: Artifacts and Information Diagram

Table 2: Artifacts and Information Summary

Artifact	Purpose
Language Learner Profile	Contains all language learner registration information including sex, location, and native language
Language Preference	Contains Language Learners' desire to learn languages including priority
Message	Contains all messages of Language Learner's communication

2.1.3 Behavior

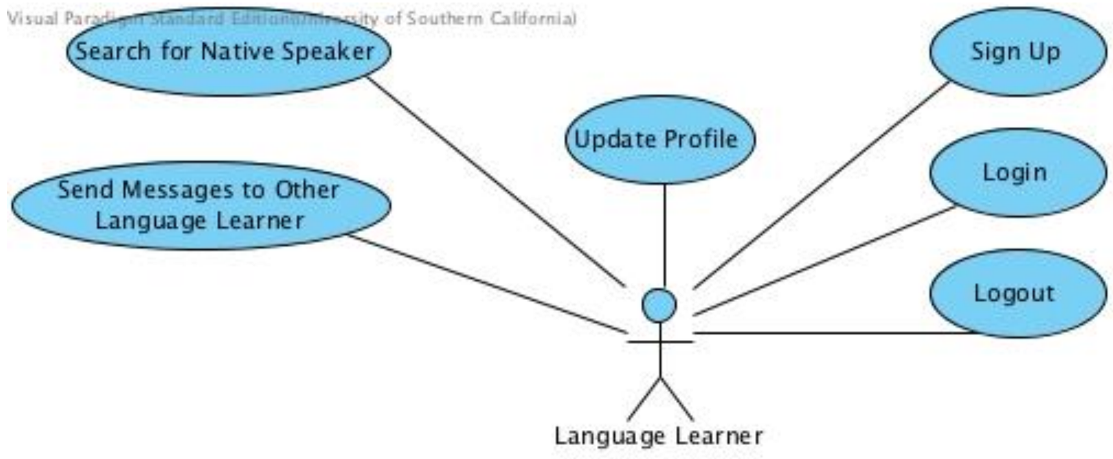

Visual Paradigm Standard Edition(University of Southern California)
Search for Native Speaker
Send Messages to Other Language Learner
Update Profile
Sign Up
Login
Logout
Language Learner

Figure 3: Process Diagram

2.1.3.1 Capability

2.1.3.1.1 Process Login

Table 3: Process Description

Identifier	Login
Purpose	To log language Learner into the system and get their profile information
Requirements	Account is already created
Development Risks	Security in user credentials
Pre-conditions	Language Learner is at the login page
Post-conditions	Language Learner is logged into the system and can see their profile information

Table 4: Typical Course of Action – Login Successfully

Seq#	Actor's Action	System's Response
1	Language Learner inputs their username and password	
2	Language Learner clicks login button	
3		Check whether username and password match the record in the database

4		Return profile information and show login successful message

2.1.3.1.2 Process Logout

Table 5 : Process Desscription

Identifier	Logout
Purpose	To log Language Learner out from the system and clear their session data
Requirements	Authentication & Authorization
Development Risks	If the system does not clear the session data, other people can use that computer to access Language Learner information.
Pre-conditions	Language Learner has already logged in
Post-conditions	Clear session, bring user back to home page, and show message logout successfully

Table 6 : Typical Course of Action – Logout Successfully

Seq#	Actor's Action	System's Response
1	Language Learner clicks on logout button	
2		System clears Language Learner session from the server
3		Return Language Learner to home page and show logout successful message

2.1.3.2 Capability Profile Management

2.1.3.2.1 Process Profile Registration

Table 7 : Process Description

Identifier	Signup
Purpose	To create Language Learner account in the system
Requirements	None
Development Risks	None
Pre-conditions	Language Learner is at the signup page
Post-conditions	Language Learner's account is created with their input information and the account is logged into the system automatically

Table 8 : Typical Course of Action – Signup Successfully

Seq#	Actor's Action	System's Response
1	Language Learner inputs all required fields	

2	Language Learner clicks submit button	
3		System checks whether username/email already exist in the system or not
4		System checks whether Language Learner inputs all required fields
5		System creates account for Language Learner based on their inputs
6		System logs Language Learner into the system
7		Return Language Learner to home page and show signup successfully message

Table 9 : Alternate Course of Action – Username/ email Already Exist

Seq#	Actor's Action	System's Response
1	Language Learner inputs all required fields	
2	Language Learner clicks submit button	
3		System checks whether username/email already exist in the system or not
4		System shows message username/email is already existed

Table 10 : Alternate Course of Action – Not Imputting All Required Fields

Seq#	Actor's Action	System's Response
1	Language Learner inputs some of required fields	
2	Language Learner clicks submit button	
3		System checks whether username/email already exist in the system or not
4		System checks whether Language Learner inputs all required fields
5		System shows message "Please input all required fields"

2.1.3.2.2 Process Profile Update

Table 11 : Process Description

Identifier	Update Profile
Purpose	To update profile information
Requirements	Already have account in the system
Development Risks	Security
Pre-conditions	Language Learner is at the profile setting page
Post-conditions	Language Learner's account is updated with their new information

Table 12 : Typical Course of Action – Profile Update Successfully

Seq#	Actor's Action	System's Response
1	Language Learner inputs all required fields	
2	Language Learner clicks submit button	
3		System checks whether Language Learner inputs all required fields
4		System updates Language Learner information
5		Show update successful message

Table 13 : Alternate Course of Action – Not Inputting All Required Fields

Seq#	Actor's Action	System's Response
1	Language Learner inputs some required fields	
2	Language Learner clicks submit button	
3		System checks whether Language Learner inputs all required fields
4		System shows message "Please input all required fields"

2.1.3.3 Capability Matching Language Learners

2.1.3.3.1 Process Match Language Learners

Table 14 : Process Description

Identifier	Search for Native Speaker
Purpose	To find a native speaker of the language that Language Learners want to learn
Requirements	Authentication & Authorization
Development	None

Risks	
Pre-conditions	- Language Learner has already logged in - The language preference has been already set - Language Learner is at the matching page
Post-conditions	Return a list of Language Learners who are native speakers of the desired language. In case of no matching Language Learners, return no data found.

Table 15 : Typical Course of Action – Match Successfully

Seq#	Actor's Action	System's Response
1	Language Learner chooses desired to learn language	
2	Language Learner clicks match button	
3		System checks language preferences for each language learner in the database to find the matching
4		Show a list of Language Learners who are native speakers of the desired language

Table 16 : Alternate Course of Action – No Data Found

Seq#	Actor's Action	System's Response
1	Language Learner chooses desired to learn language	
2	Language Learner clicks match button	
3		System checks language preferences for each language learner in the database to find the matching
4		Show message no matching found

2.1.3.4 Capability Messaging

2.1.3.4.1 Process Send Message to Language Learners

Table 17 : Process Description

Identifier	Send messages to other language learners
Purpose	To send message to other language learner to have a conversation
Requirements	Authentication & Authorization
Development Risks	None
Pre-conditions	Language Learner has already logged in

Post-conditions	Return a list of Language Learners who are native speakers of the desired language. In case of no matching Language Learners, return no data found.

Table 18 : Typical Course of Action – Send Message Successfully

Seq#	Actor's Action	System's Response
1	Go to the profile of other language learner	
2	Click send message button	
3		System brings Language Learner to send message page
4	Type message content	
5	Click send button	
6		System check that message is not empty
7		System saves message into database and show send successful message

Table 19 : Alternate Course of Action – Missing Required Field

Seq#	Actor's Action	System's Response
1	Go to the profile of other language learner	
2	Click send message button	
3		System brings Language Learner to send message page
4	Click send button	
5		System check that message is not empty
6		System shows message "Please fill in the content"

2.1.4 Modes of Operation

The system will operate in only one mode, so nothing further need be said of modes of operation.

2.2 System Analysis Rationale

There is 1 type of actors in the system: Language Learner and Maintainer.

1. **Language Learner**: A user who wants to learn new language.

3. Technology-Independent Model

3.1 Design Overview

3.1.1 System Structure

Figure 4 : Hardware Class Diagram (TI)

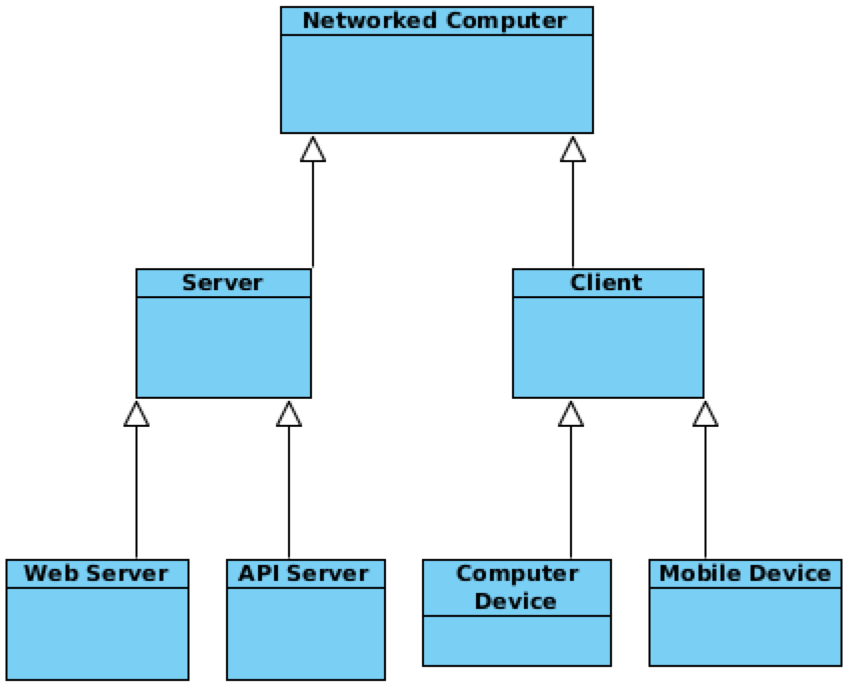


Table 20 : Hardware Component Class Description (TI)

Hardware Component	Description
Networked Computer	The computer/mobile device connected to the Internet
Server	Servers where we deploy the LINGGGO system
Client	Client devices that are able to connect to the LINGGO server
Web Server	The server which serves all the front-end element for the clients
API Server	The server which provides the access to back-end APIs
Computer Devices	Client's devices running desktop OS, such as: Windows and etc.
Mobile Devices	Client's devices running mobile OS, such as: iOS and etc.

Figure 5 : Software Component Class Diagram (TI)

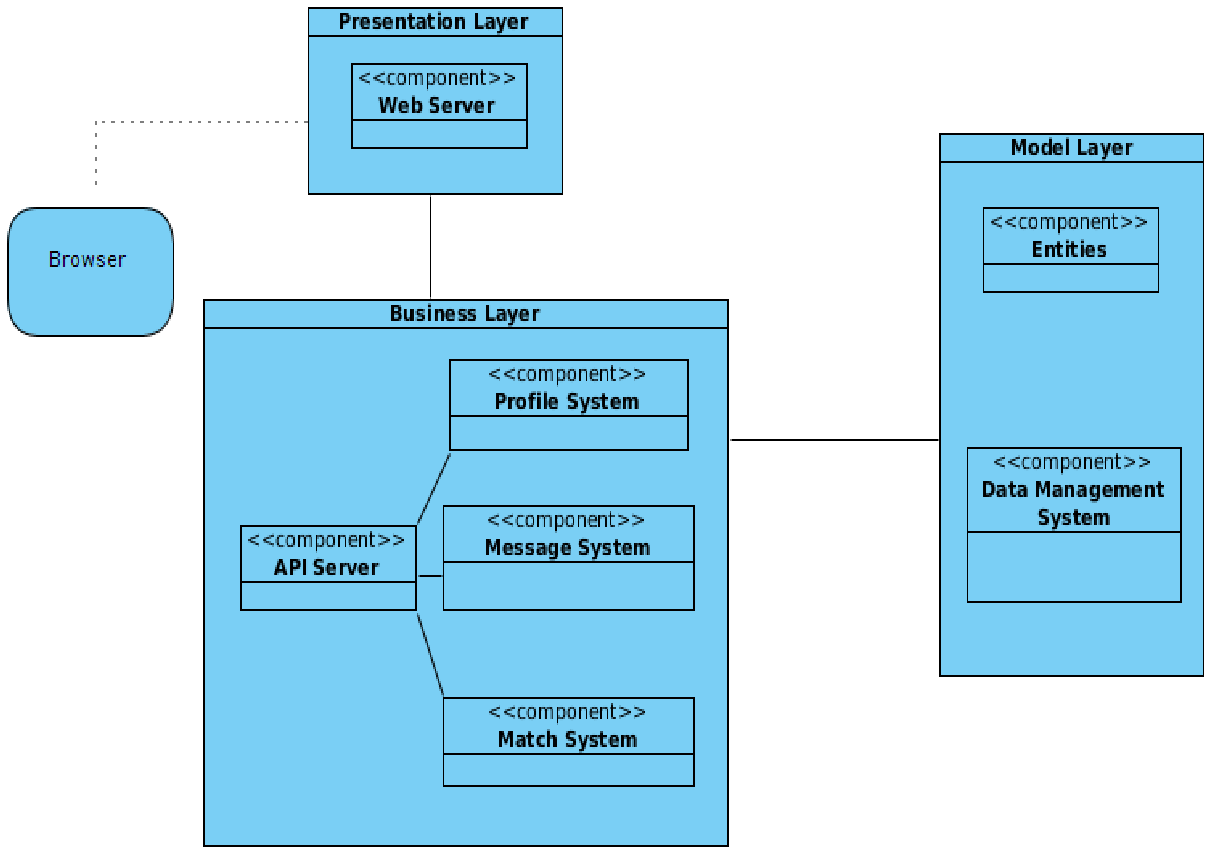
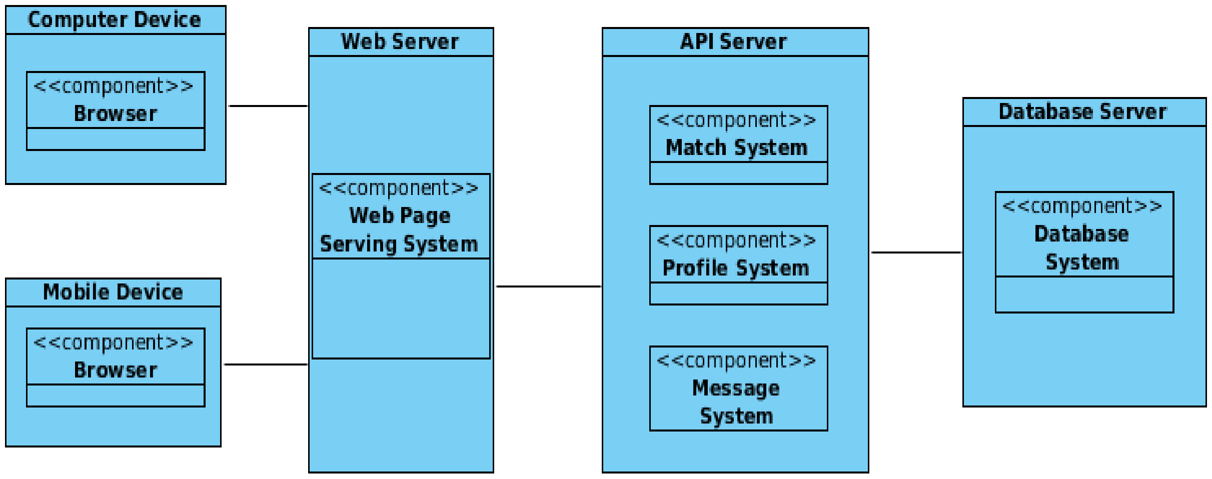


Table 21: Software Component Class Description (TI)

Software Component	Description
Presentation Layer	This layer provides the access to all the resources of the front-end elements in the LINGGGO system, including HTML pages, images, CSS and javascript. This client (Language Learner) directly interacts with this layers.
Business Layer	This layer provides the business logic of the LINGGGO system. It defines the APIs which can be later called to access the resources in the back-end system (back-end server and database). This layer is not directly accessible to the client themselves but should be called by the presentation layer.
Model Layer	This layer provides the storage and management of different data in the LINGGGO system, in the form of tables. It also gives the APIs so that the business layer can access the data in the data management system.

Figure 6 : Deployment Diagram (TI)



3.1.2 Design Classes

*: Since we're using REST system architecture, each entity has its own class method to modify its data. It serves both the role of Entity and Controller.

3.1.2.1 Profile Management

Figure 7 : Profile Management Class Diagram (TI)

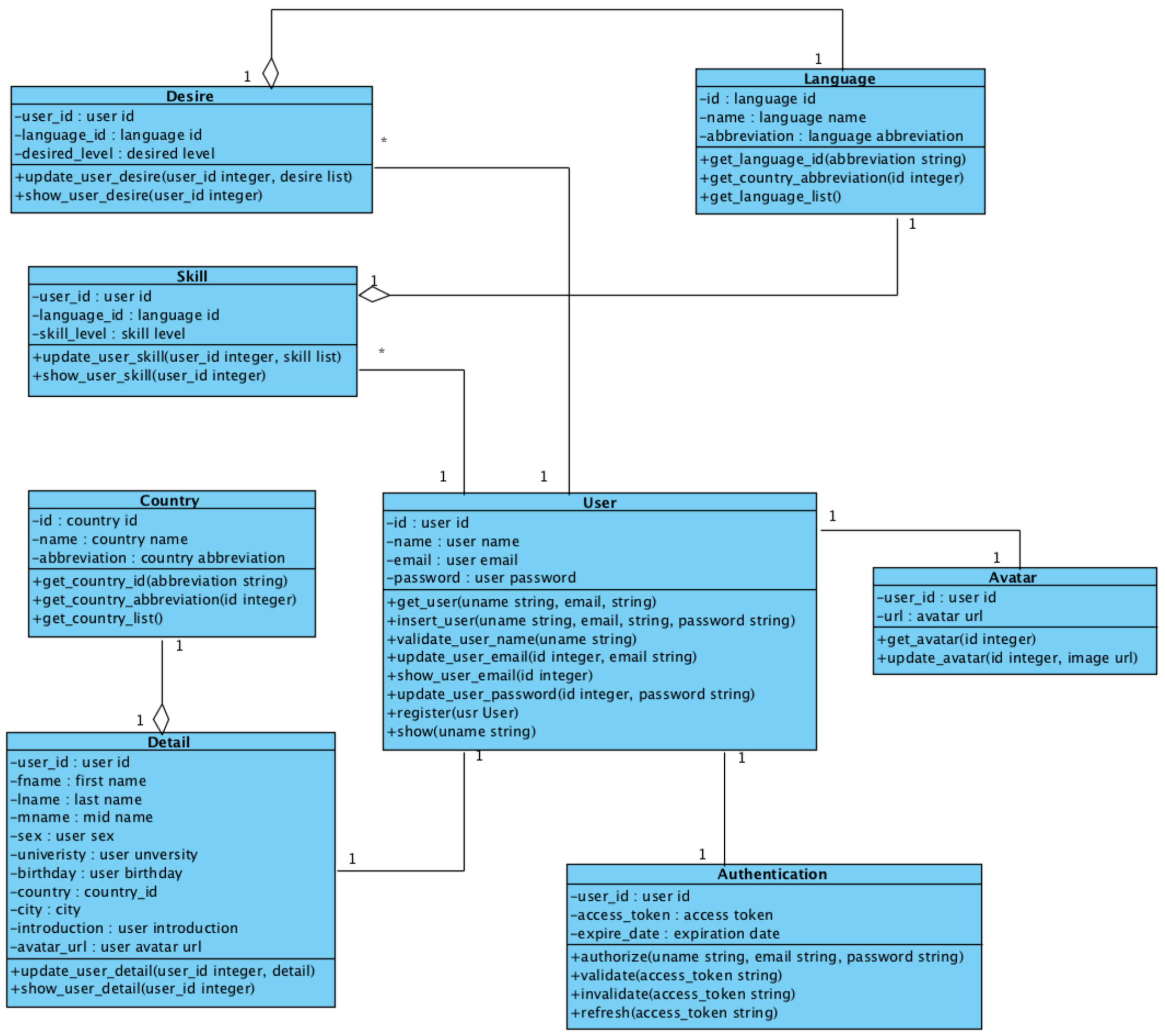

Desire
-user_id : user id
-language_id : language id
-desired_level : desired level
+update_user_desire(user_id integer, desire list)
+show_user_desire(user_id integer)
Language
-id : language id
-name : language name
-abbreviation : language abbreviation
+get_language_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_language_list()
Skill
-user_id : user id
-language_id : language id
-skill_level : skill level
+update_user_skill(user_id integer, skill list)
+show_user_skill(user_id integer)
Country
-id : country id
-name : country name
-abbreviation : country abbreviation
+get_country_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_country_list()
User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
Avatar
-user_id : user id
-url : avatar url
+get_avatar(id integer)
+update_avatar(id integer, image url)
Detail
-user_id : user id
-fname : first name
-lname : last name
-mname : mid name
-sex : user sex
-univeristy : user unversity
-birthday : user birthday
-country : country_id
-city : city
-introduction : user introduction
-avatar_url : user avatar url
+update_user_detail(user_id integer, detail)
+show_user_detail(user_id integer)
Authentication
-user_id : user id
-access_token : access token
-expire_date : expiration date
+authorize(uname string, email string, password string)
+validate(access_token string)
+invalidate(access_token string)
+refresh(access_token string)
1
1
*
1
1
*
1
1
1
1
1
1
1
1
1
1
1
1

Table 22 : Profile Management Class Description (TI)

Class	Type	Description
Desire	Entity, Controller	It shows user's desired language. One user can have multiple desired language.
Skill	Entity, Controller	It shows user's skilled language. One user can have multiple skilled language

User	Entity, Controller	It stores user's authentication information including user id, user name, user email and password.
Detail	Entity, Controller	It stores user's detail information.
Authentication	Entity, Controller	It stores user's authentication for logging into the system
Avatar	Entity, Controller	It stores user's avatar information.
Language	Entity, Controller	It stores all the languages supported in the system, including: language id, name and abbreviation.
Country	Entity, Controller	It stores all the countries supported in the system, including: country id, name and abbreviation

3.1.2.2 Match System

Figure 8 : Match System Class Diagram (TI)

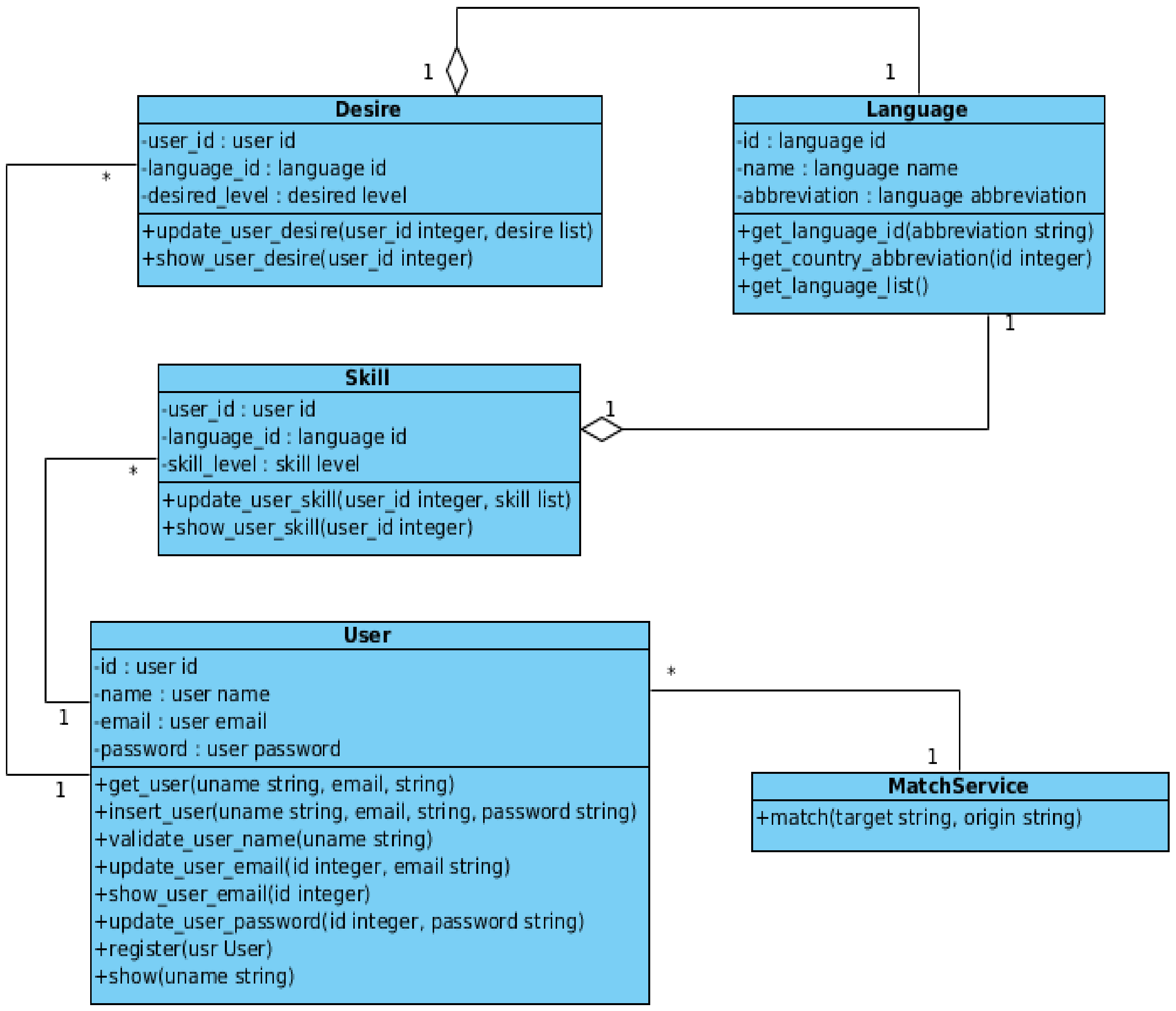

Desire
-user_id : user id
-language_id : language id
-desired_level : desired level
+update_user_desire(user_id integer, desire list)
+show_user_desire(user_id integer)
Language
-id : language id
-name : language name
-abbreviation : language abbreviation
+get_language_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_language_list()
Skill
-user_id : user id
-language_id : language id
-skill_level : skill level
+update_user_skill(user_id integer, skill list)
+show_user_skill(user_id integer)
User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
MatchService
+match(target string, origin string)
1
1
*
1
1
*
1
1
*
1

Table 23 : Match System Class Diagram (TI)

Class	Type	Description
User	Entity, Controller	It stores user's authentication information including user id, user name, user email and password.
Desire	Entity, Controller	It shows user's desired language. One user can have multiple desired language.
Skill	Entity, Controller	It shows user's skilled language. One user can have multiple skilled language
Language	Entity, Controller	It stores all the languages supported in the system, including: language id, name and abbreviation.
MatchService	Controller	It retrieves the data from database and generated the matched user for a API call.

3.1.2.3 Message System

Figure 9 : Message System Class Diagram (TI)

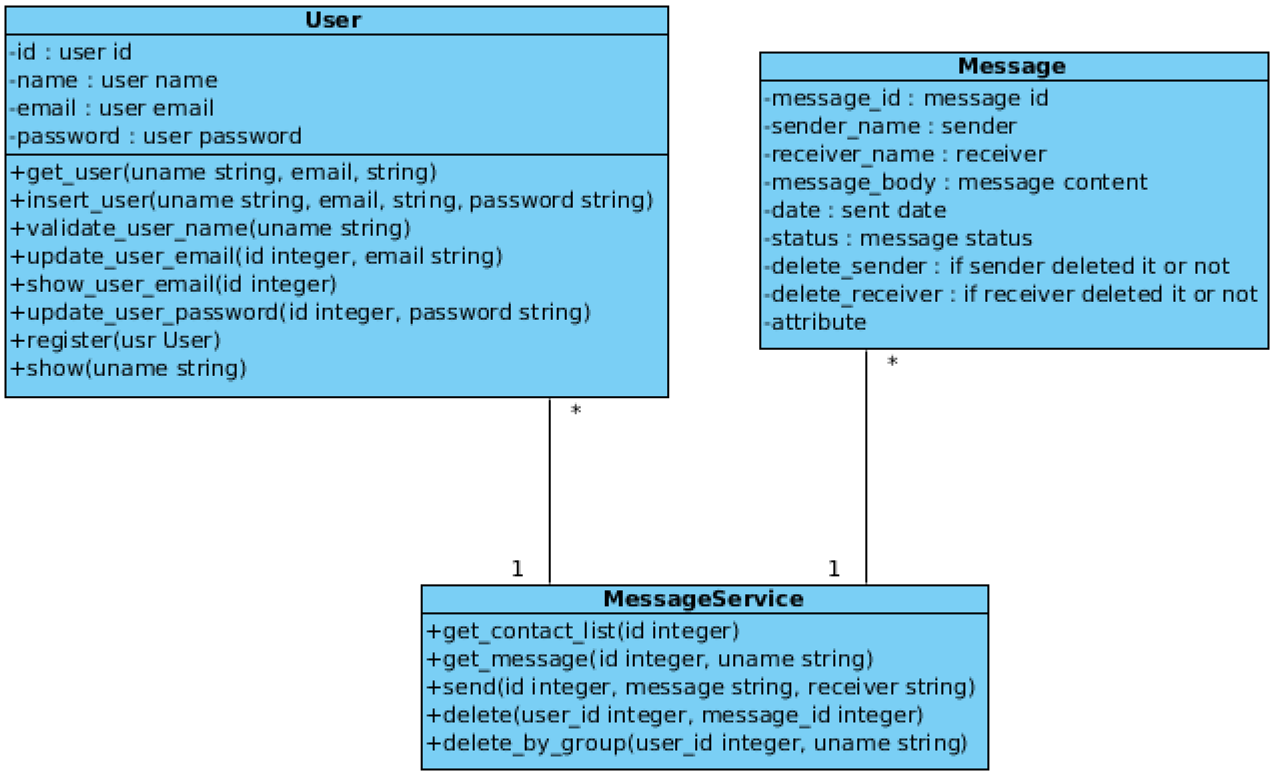

User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
Message
-message_id : message id
-sender_name : sender
-receiver_name : receiver
-message_body : message content
-date : sent date
-status : message status
-delete_sender : if sender deleted it or not
-delete_receiver : if receiver deleted it or not
-attribute
*
*
1
1
MessageService
+get_contact_list(id integer)
+get_message(id integer, uname string)
+send(id integer, message string, receiver string)
+delete(user_id integer, message_id integer)
+delete_by_group(user_id integer, uname string)

Table 24 : Message System Class Description (TI)

Class	Type	Description
User	Entity, Controller	It stores user's authentication information including

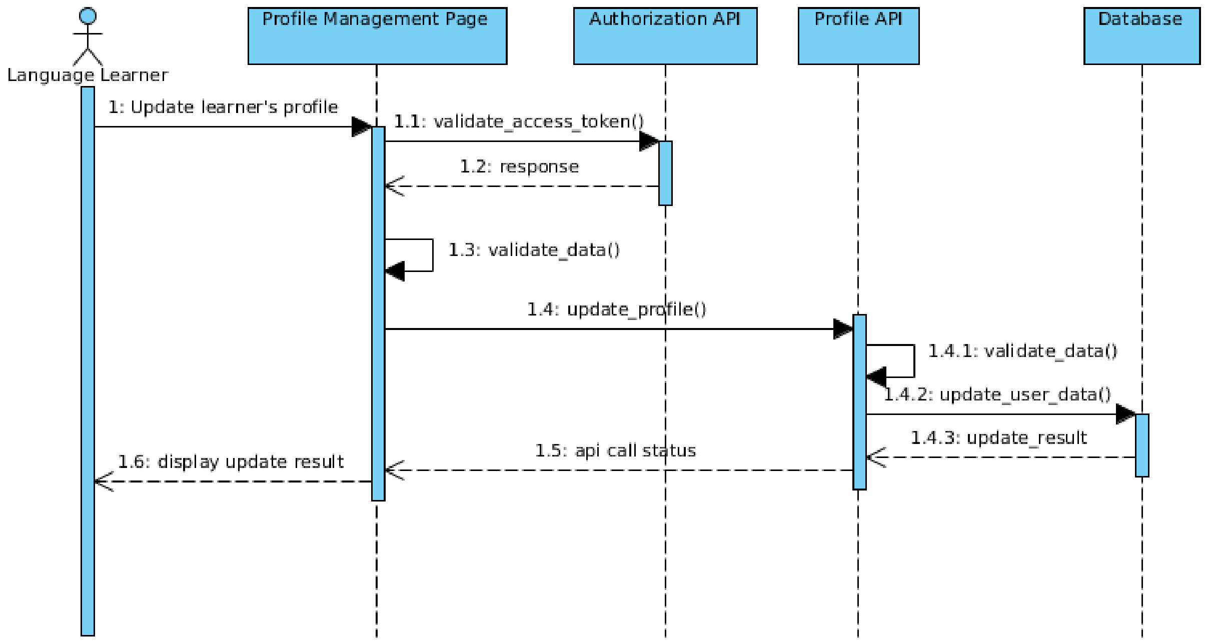
		user id, user name, user email and password.
Message	Entity	It keeps the record of one message.
MessageService	Controller	It controls the logic for a language

3.1.3 Process Realization

In this section, we will show 3 sequence diagrams which accords with our three main functionalities in our system.

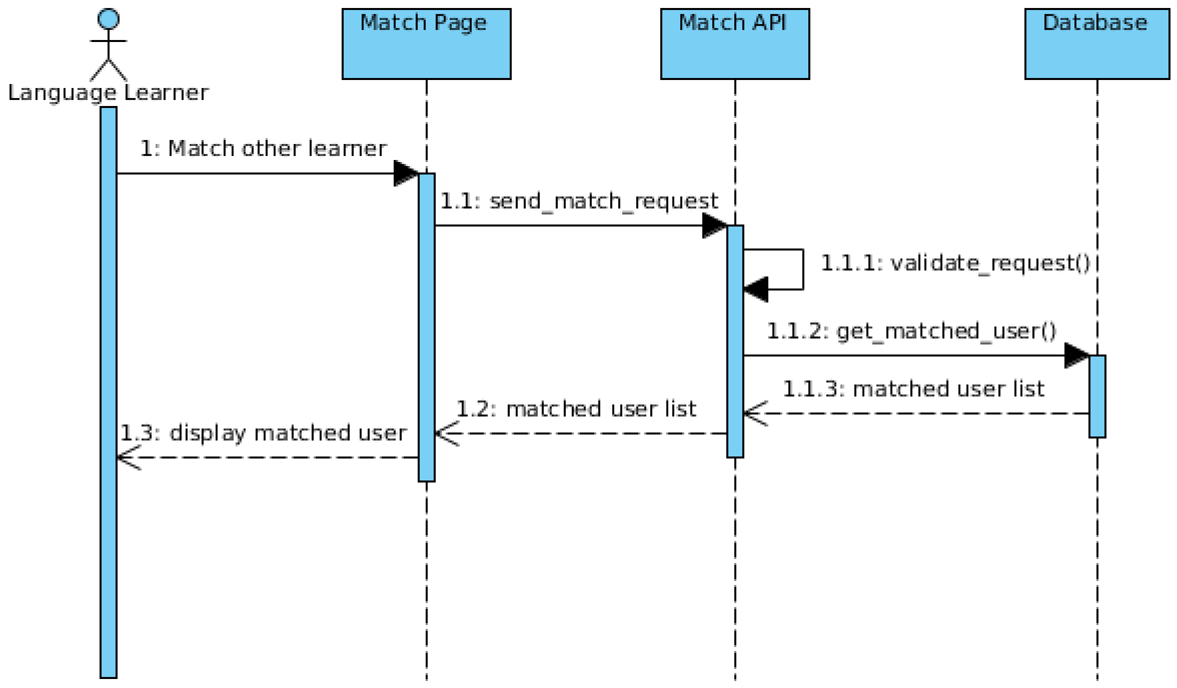
The language learners use the profile management page to manage and update their profiles.

Figure 10 : Profile Management Sequence Diagram (TI)



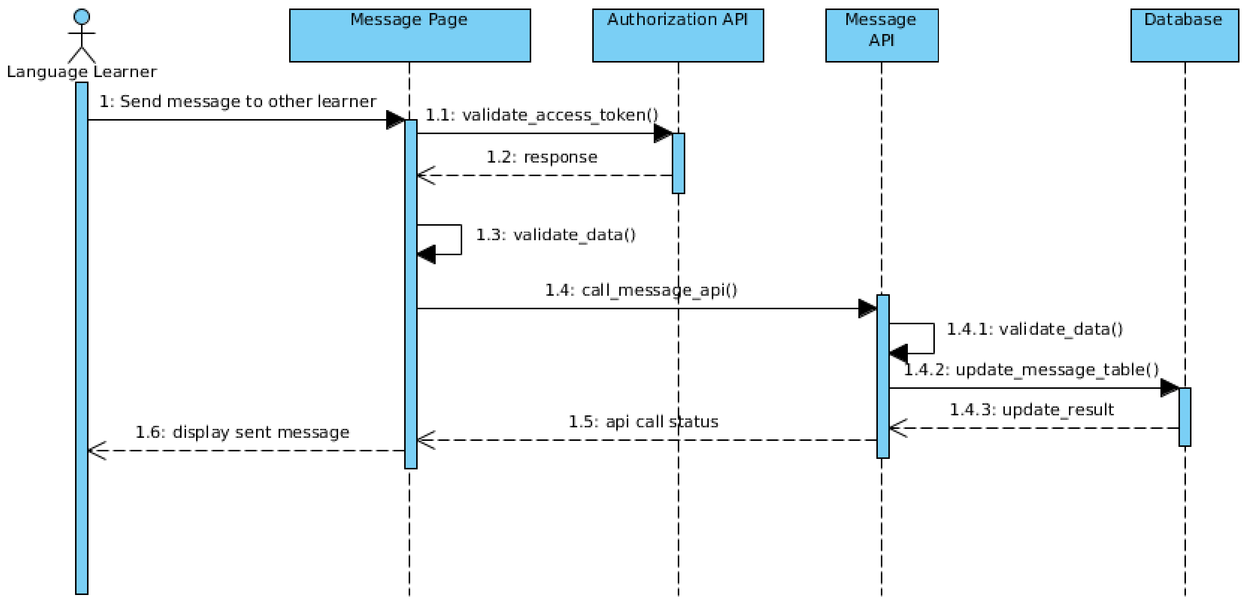
The language learner uses the match page to match others people with the corresponding desire.

Figure 11 : Match Sequence Diagram (TI)



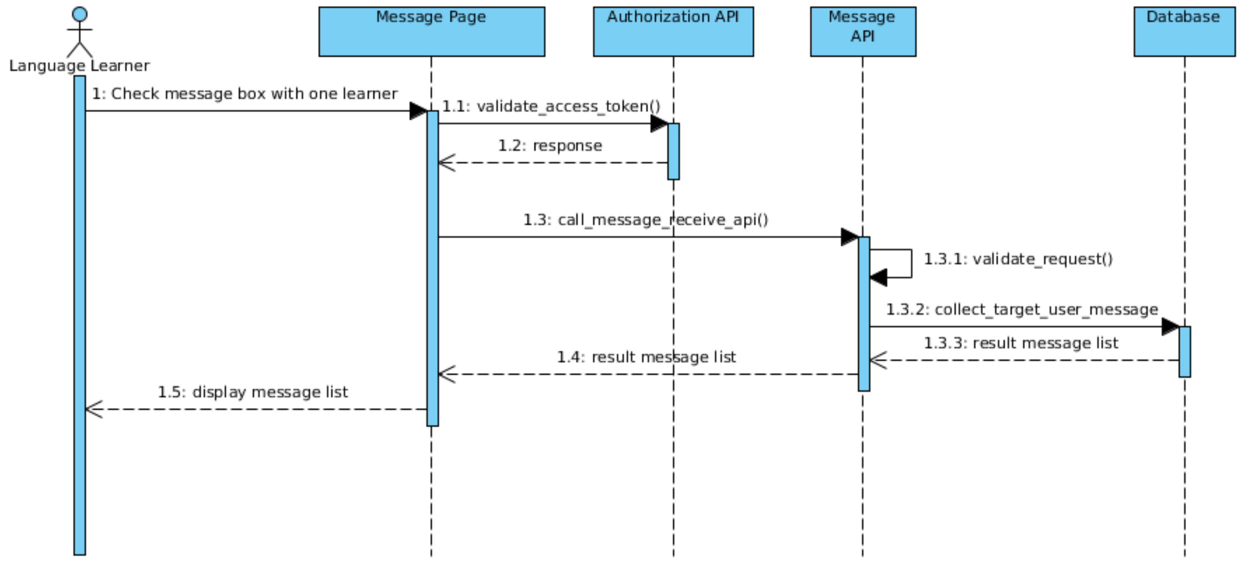
The language learner uses the message interface to send the message to one other language learner.

Figure 12 : Send Message Sequence Diagram (TI)



The language learner uses the message interface to receive the message to one other language learner.

Figure 13 : Receive Message Sequence Diagram (TI)


Language Learner
Message Page
Authorization API
Message API
Database
1: Check message box with one learner
1.1: validate_access_token()
1.2: response
1.3: call_message_receive_api()
1.3.1: validate_request()
1.3.2: collect_target_user_message
1.3.3: result message list
1.4: result message list
1.5: display message list

3.2 Design Rationale

As the LINGGGO project is meanly web-based application and we want to reserve the potential to later extend its capabilities for other platform development. We adopt the idea of REST and we mainly use 3-tier architecture in our system. The presentation layer is done by the web server and client browser and this tier is only in charge of maintaining communication between the client and the server and little business logic is implemented here to keep the client simple (but heavy user interface setup). The second and third tier are currently combined together and lay on the API server. The API server is responsible for processing client's request and sending back corresponding result. This layer is only in charge of business logic and data management and it has nothing to do with how to present the data to the client. This kind of architecture has several advantages: a. This system architecture reduces the overhead between the client and server by eliminating unnecessary data transfer between the client and server as we avoid transferring duplicate HTML and other static resources b. This system architecture reduces the cohesion of the development process between the front-end and back-end development so that both parts of the team can focus on their own development environment as AJAX technology is platform-independent and REST system architecture determines the correct how to parse the AJAX data c. This system architecture can also be easily converted into distributed structure if we encounter a bottleneck on a single server which runs the entire API services.

4. Technology-Specific System Design

4.1 Design Overview

4.1.1 System Structure

Figure 14 : Hardware Component Diagram (TS)

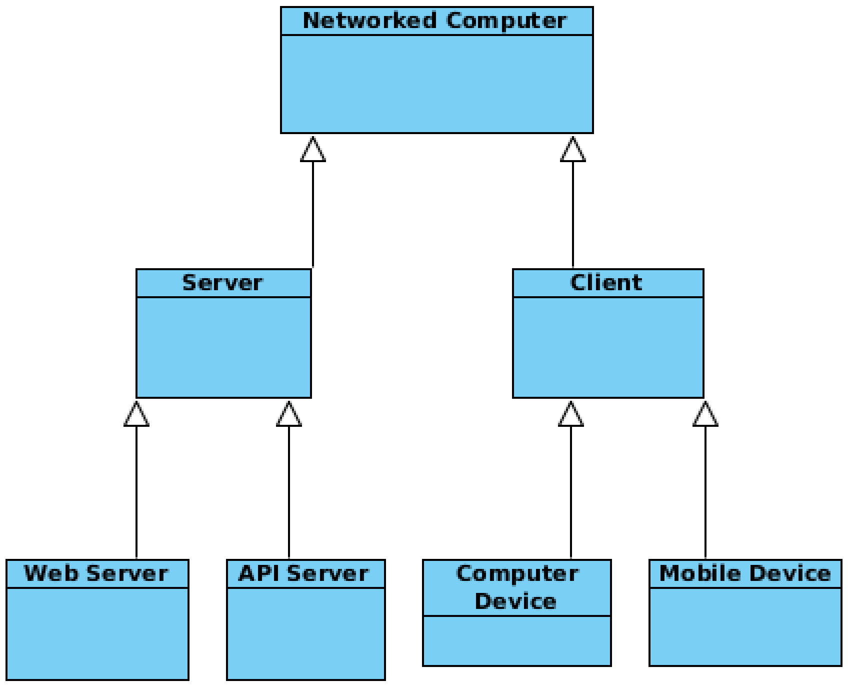


Table 25 : Hardware Component Description (TS)

Hardware Component	Description
Networked Computer	The computer/mobile device connected to the Internet
Server	Servers where we deploy the LINGGGO system
Client	Client devices that are able to connect to the LINGGO server
Web Server	The server which serves all the front-end element for the clients
API Server	The server which provides the access to back-end APIs
Computer Devices	Client’s devices running desktop OS, such as: Windows and etc.
Mobile Devices	Client’s devices running mobile OS, such as: iOS and etc.

Figure 15 : Software Component Diagram (TS)

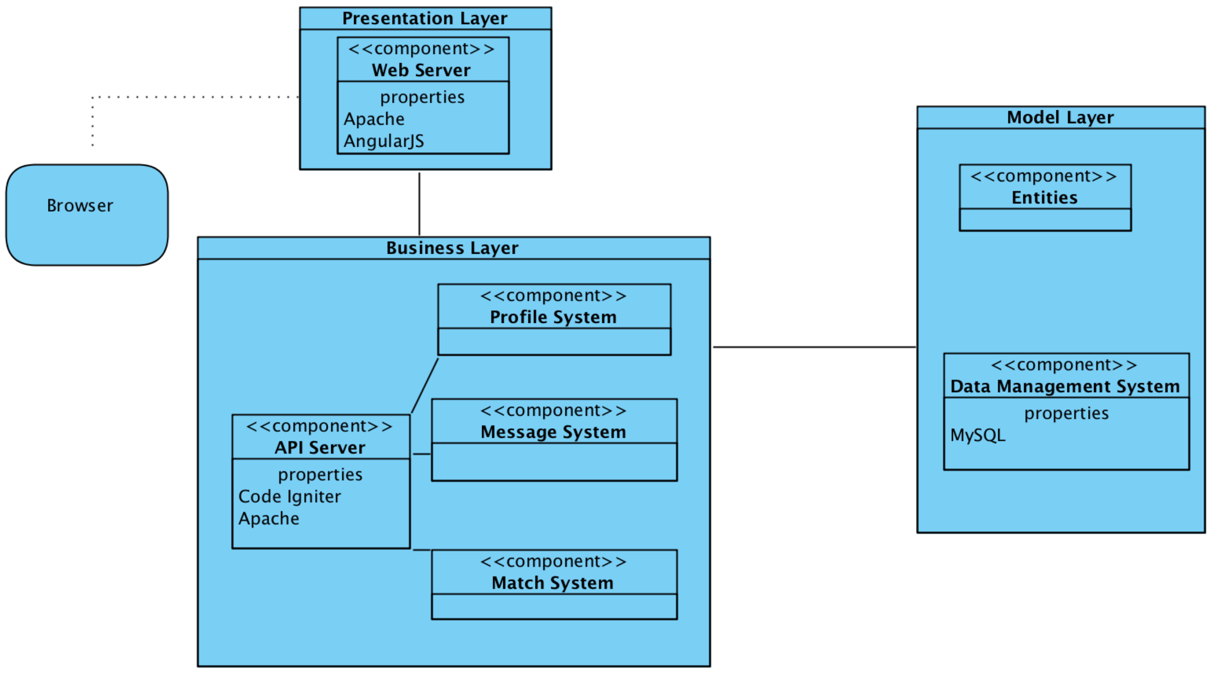
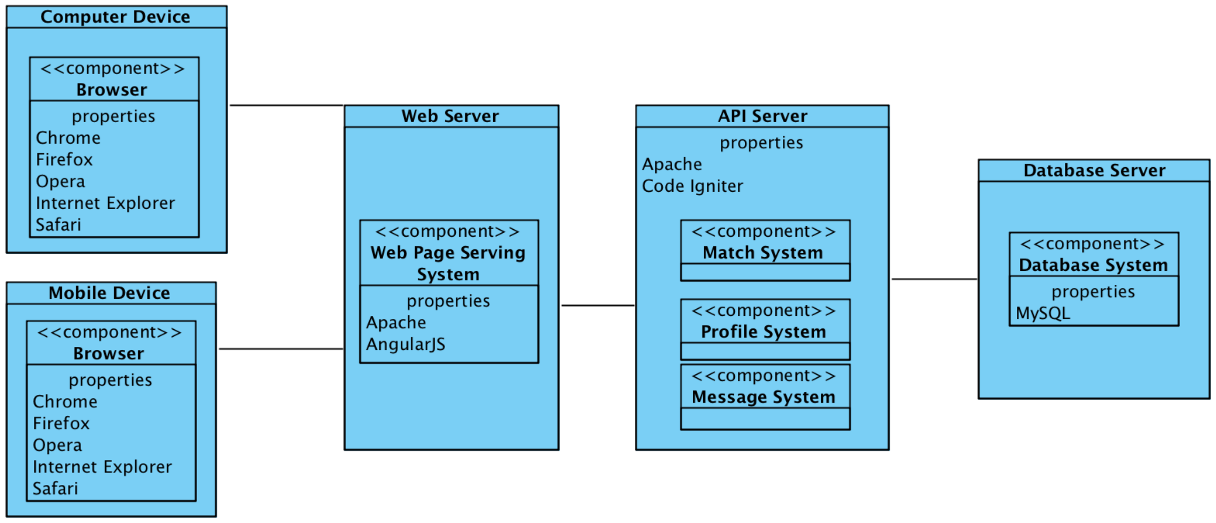


Table 26 : Software Component Description (TS)

Software Component	Description
Presentation Layer	This layer utilizes the Apache web server to provide the access to all the resources of the front-end elements in the LINGGGO system, including HTML pages, images, CSS and javascript. This client (Language Learner) directly interacts with this layers through different common web browser such as Chrome, Firefox, Safari, Internet Explorer and Opera.
Business Layer	This layer utilizes the Apache Server (with PHP mod) and PHP framework ‘Code Igniter’ to provide the business logic of the LINGGGO system. It defines the APIs which can be later called to access the resources in the back-end system (back-end server and database). This layer is not directly accessible to the client themselves but should be called by the presentation layer.
Model Layer	This layer utilizes the MySQL database to provide the storage and management of different data in the LINGGGO system, in the form of tables. It also gives the APIs so that the business layer can access the data in the data management system.

Figure 16 : Deployment Diagram (TS)



4.1.2 Design Classes

*: Since we're using REST system architecture, each entity has its own class method to modify its data. It serves both the role of Entity and Controller.

4.1.2.1 Profile Management

Figure 17 : Profile Management Class Diagram (TS)

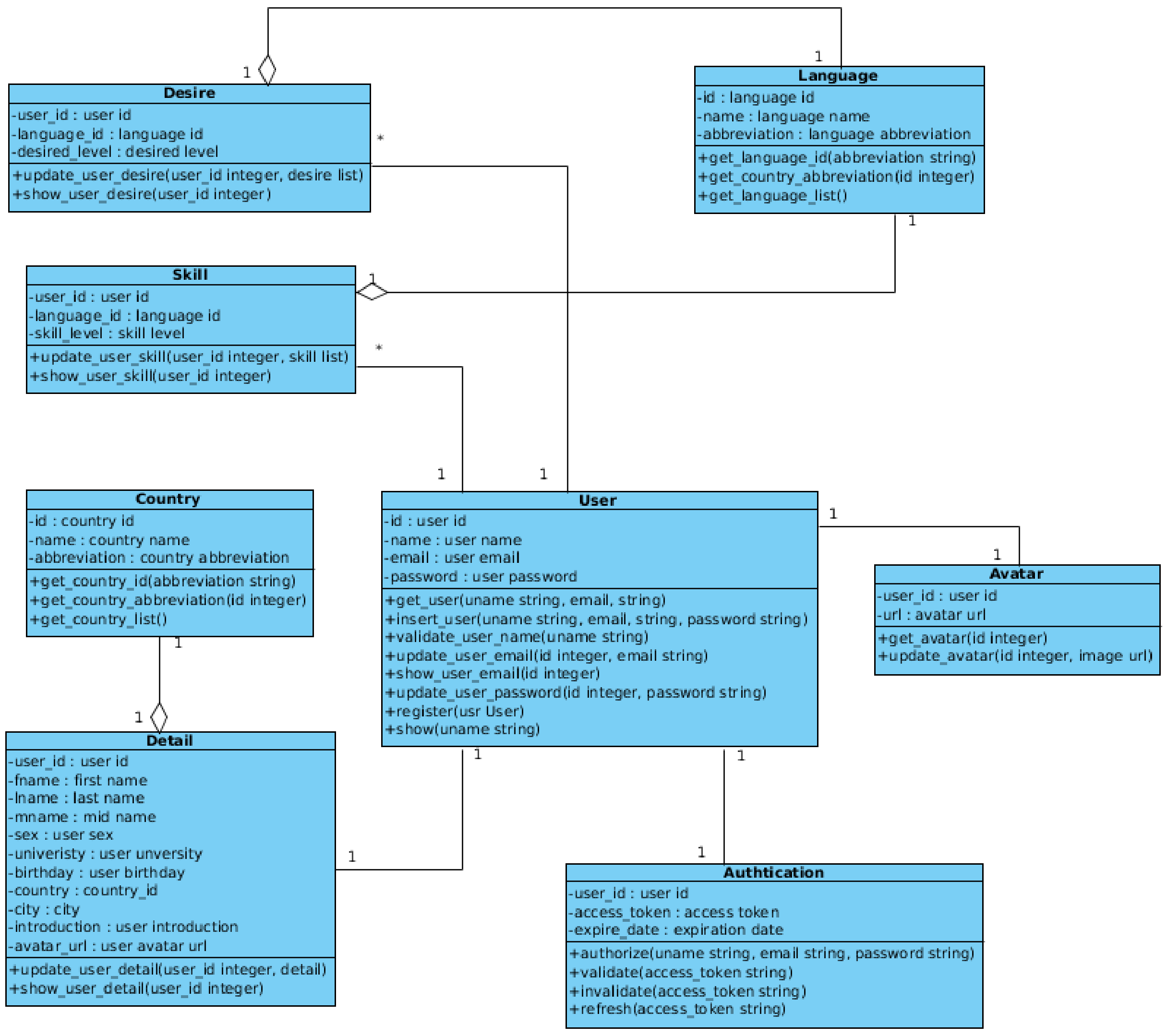

Desire
-user_id : user id
-language_id : language id
-desired_level : desired level
+update_user_desire(user_id integer, desire list)
+show_user_desire(user_id integer)
Language
-id : language id
-name : language name
-abbreviation : language abbreviation
+get_language_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_language_list()
Skill
-user_id : user id
-language_id : language id
-skill_level : skill level
+update_user_skill(user_id integer, skill list)
+show_user_skill(user_id integer)
Country
-id : country id
-name : country name
-abbreviation : country abbreviation
+get_country_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_country_list()
User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
Avatar
-user_id : user id
-url : avatar url
+get_avatar(id integer)
+update_avatar(id integer, image url)
Detail
-user_id : user id
-fname : first name
-lname : last name
-mname : mid name
-sex : user sex
-univeristy : user unversity
-birthday : user birthday
-country : country_id
-city : city
-introduction : user introduction
-avatar_url : user avatar url
+update_user_detail(user_id integer, detail)
+show_user_detail(user_id integer)
Authtication
-user_id : user id
-access_token : access token
-expire_date : expiration date
+authorize(uname string, email string, password string)
+validate(access_token string)
+invalidate(access_token string)
+refresh(access_token string)
1
*

Table 27 : Profile Management Class Description (TS)

Class	Type	Description
Desire	Entity, Controller	It shows user's desired language. One user can have multiple desired language.
Skill	Entity, Controller	It shows user's skilled language. One user can have multiple skilled language
User	Entity, Controller	It stores user's authentication information including user id, user name, user email and password.
Detail	Entity, Controller	It stores user's detail information.
Authentication	Entity, Controller	It stores user's authentication for logging into the system
Avatar	Entity, Controller	It stores user's avatar information.
Language	Entity, Controller	It stores all the languages supported in the system, including: language id, name and abbreviation.
Country	Entity, Controller	It stores all the countries supported in the system, including: country id, name and abbreviation

Figure 18 : Match Class Diagram (TS)

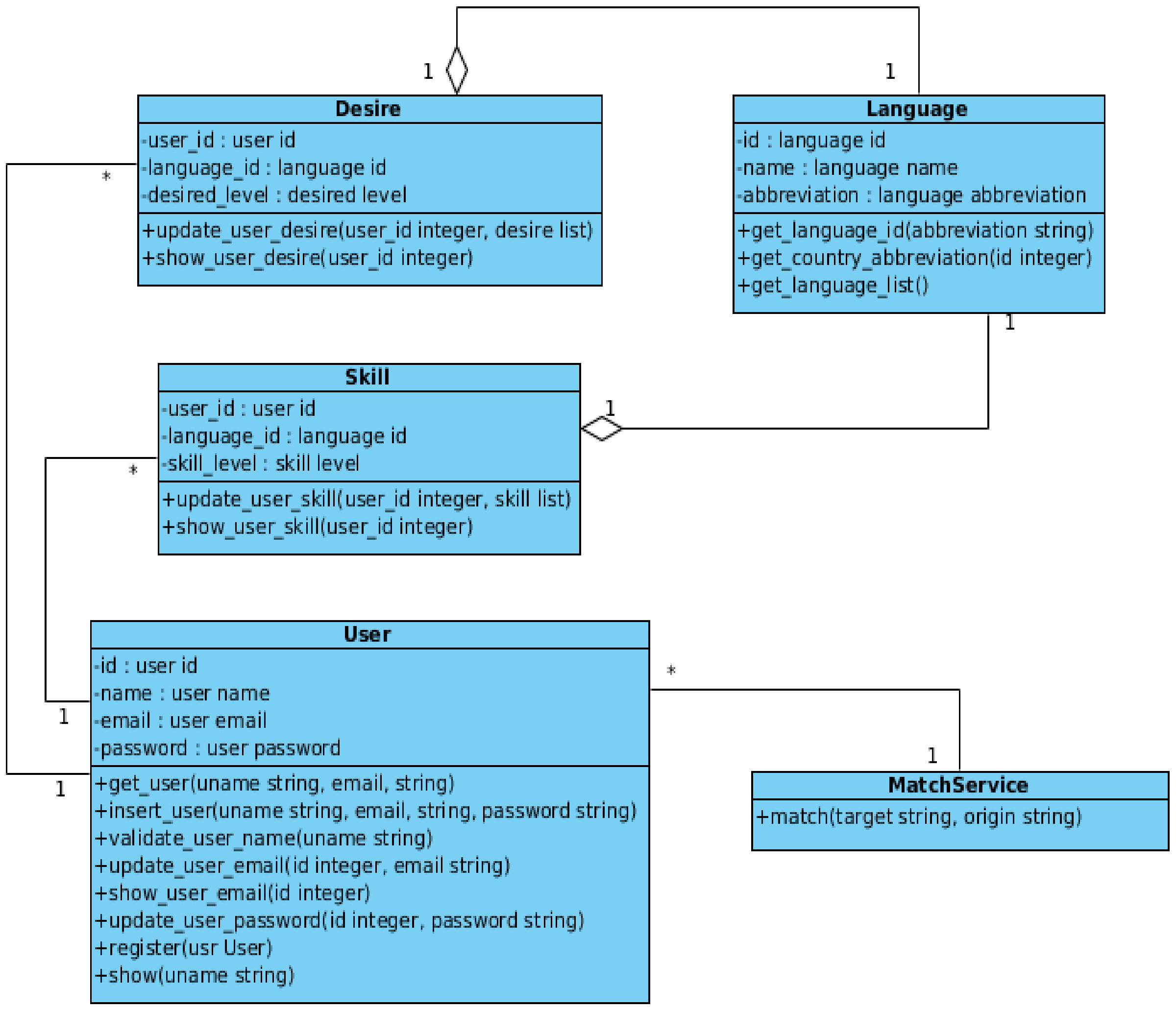

Desire
-user_id : user id
-language_id : language id
-desired_level : desired level
+update_user_desire(user_id integer, desire list)
+show_user_desire(user_id integer)
Language
-id : language id
-name : language name
-abbreviation : language abbreviation
+get_language_id(abbreviation string)
+get_country_abbreviation(id integer)
+get_language_list()
Skill
-user_id : user id
-language_id : language id
-skill_level : skill level
+update_user_skill(user_id integer, skill list)
+show_user_skill(user_id integer)
User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
MatchService
+match(target string, origin string)
1
1
*
1
1
*
1
1
*
1

Table 28 : Match Class Description (TS)

Class	Type	Description
User	Entity, Controller	It stores user's authentication information including user id, user name, user email and password.
Desire	Entity, Controller	It shows user's desired language. One user can have multiple desired language.
Skill	Entity, Controller	It shows user's skilled language. One user can have multiple skilled language
Language	Entity, Controller	It stores all the languages supported in the system, including: language id, name and abbreviation.
MatchService	Controller	It retrieves the data from database and generated the

		matched user for a API call.

Figure 19 : Message Class Diagram (TS)

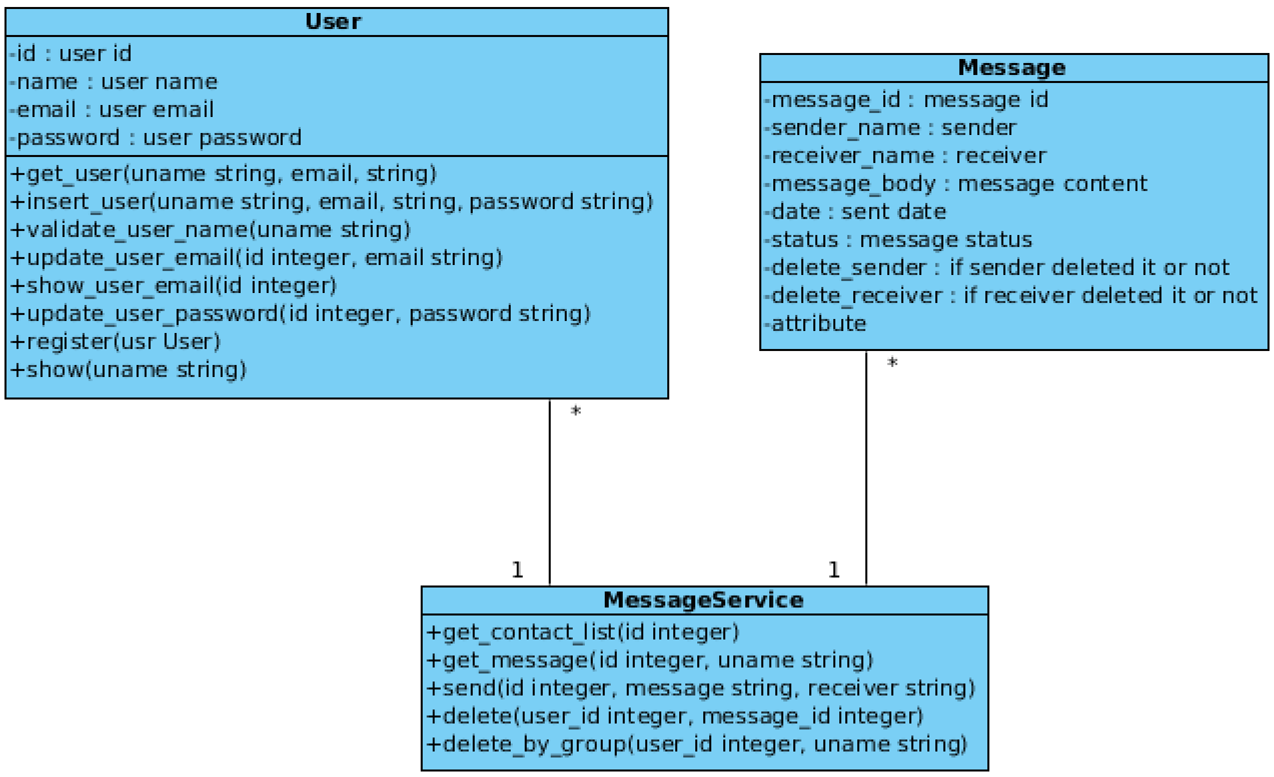

User
-id : user id
-name : user name
-email : user email
-password : user password
+get_user(uname string, email, string)
+insert_user(uname string, email, string, password string)
+validate_user_name(uname string)
+update_user_email(id integer, email string)
+show_user_email(id integer)
+update_user_password(id integer, password string)
+register(usr User)
+show(uname string)
Message
-message_id : message id
-sender_name : sender
-receiver_name : receiver
-message_body : message content
-date : sent date
-status : message status
-delete_sender : if sender deleted it or not
-delete_receiver : if receiver deleted it or not
-attribute
*
*
1
1
MessageService
+get_contact_list(id integer)
+get_message(id integer, uname string)
+send(id integer, message string, receiver string)
+delete(user_id integer, message_id integer)
+delete_by_group(user_id integer, uname string)

Table 29 : Message Class Description (TS)

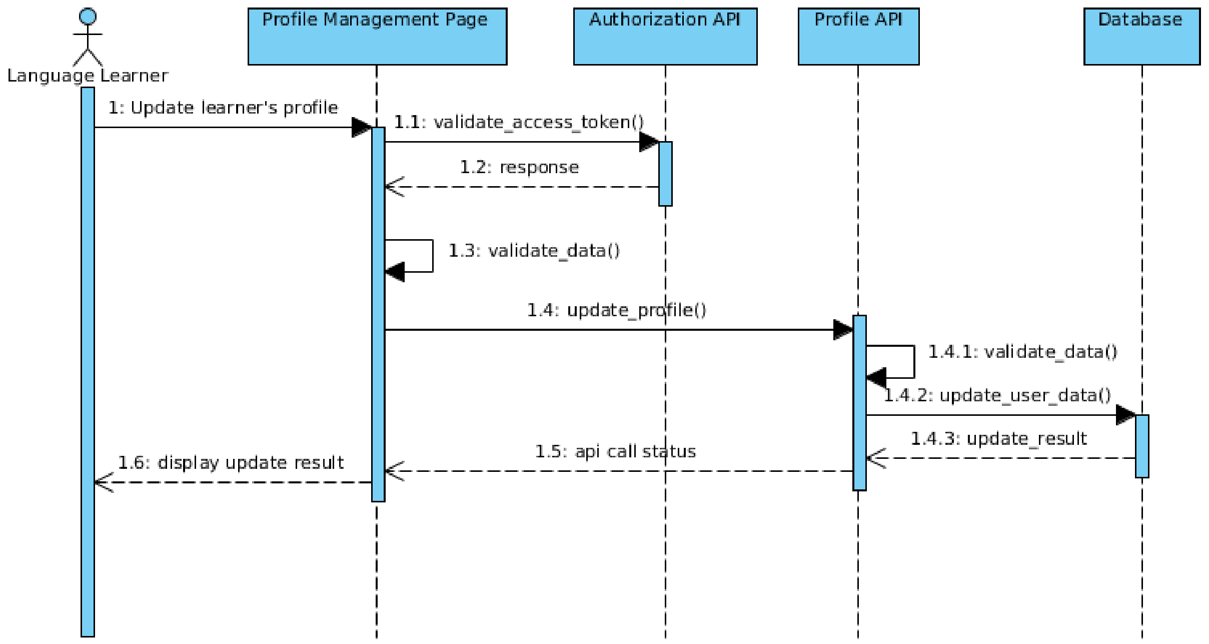
Class	Type	Description
User	Entity, Controller	It stores user's authentication information including user id, user name, user email and password.
Message	Entity	It keeps the record of one message.
MessageService	Controller	It controls the logic for a language

4.1.3 Process Realization

In this section, we will show 3 sequence diagrams which accords with our three main functionalities in our system.

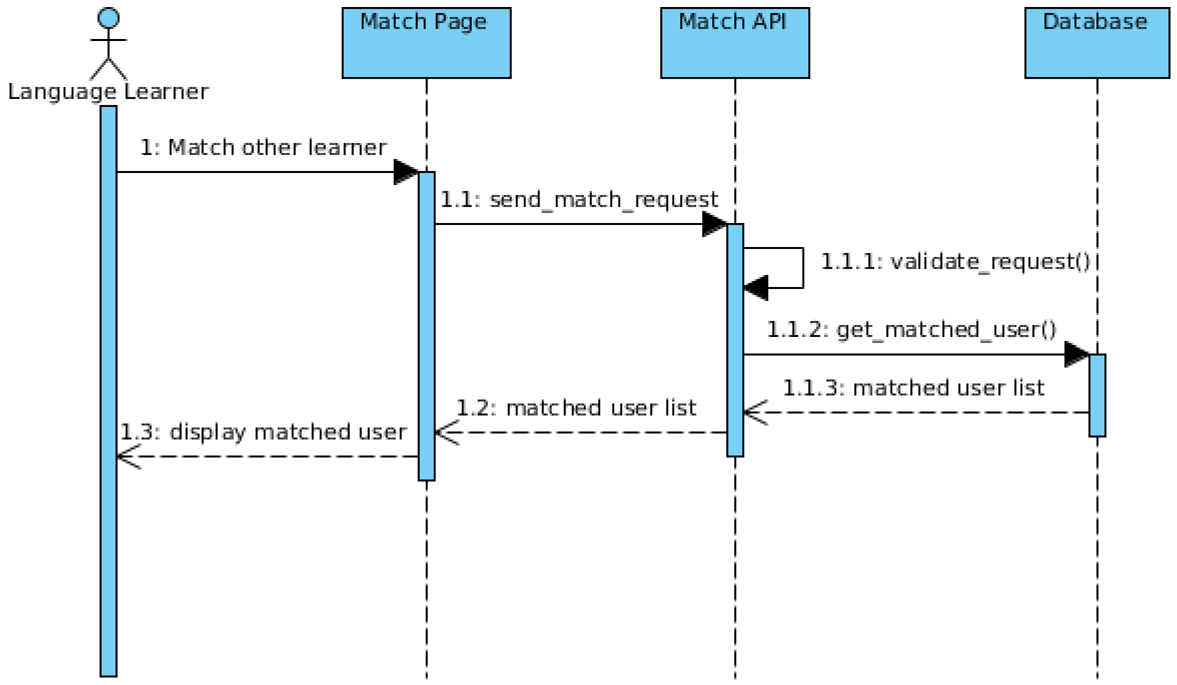
The language learners use the profile management page to manage and update their profiles.

Figure 20 : Profile Management Sequence Diagram (TS)



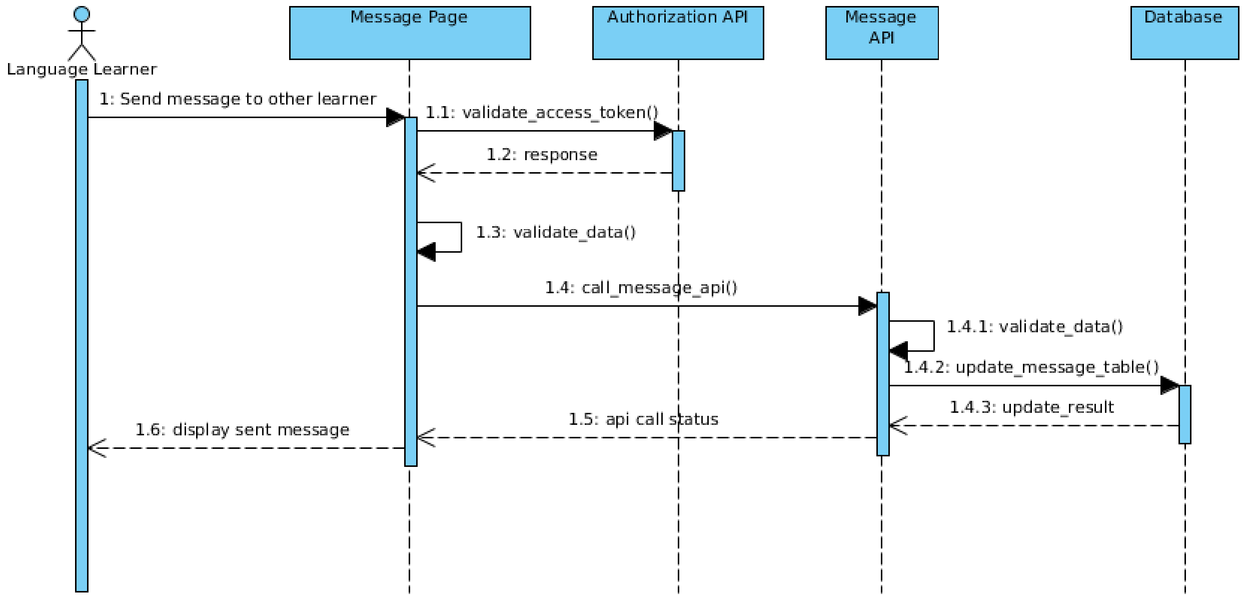
The language learner uses the match page to match others people with the corresponding desire.

Figure 21 : Match Sequence Diagram (TS)



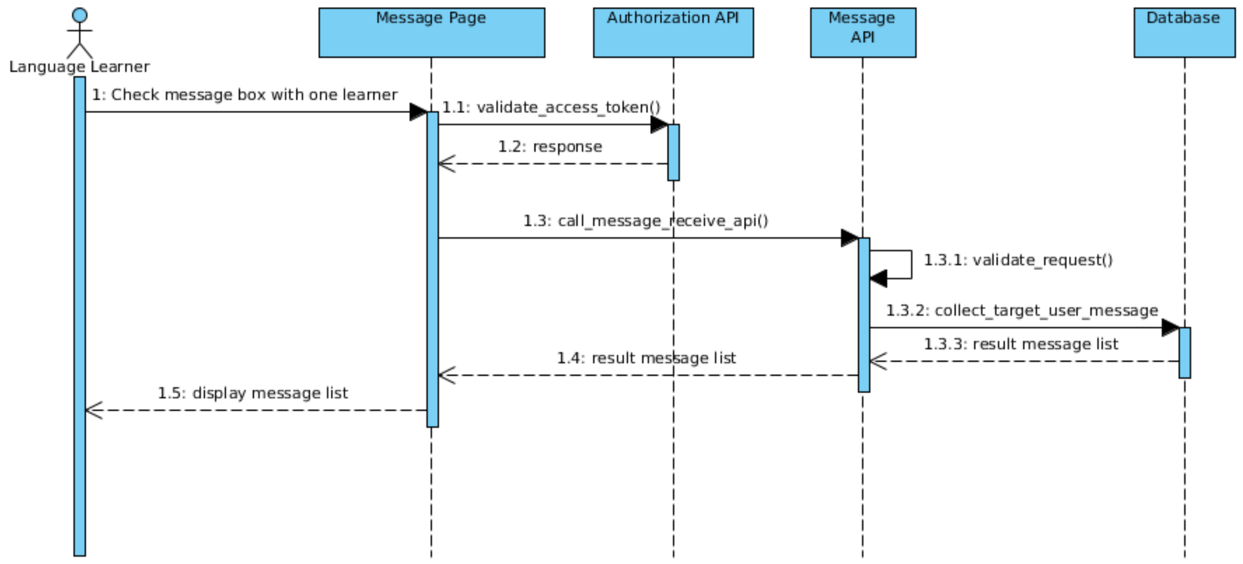
The language learner uses the message interface to send the message to one other language learner.

Figure 22 : Send Message Sequence Diagram (TS)



The language learner uses the message interface to receive the message to one other language learner.

Figure 23 : Receive Message Sequence Diagram (TS)


Language Learner
Message Page
Authorization API
Message API
Database
1: Check message box with one learner
1.1: validate_access_token()
1.2: response
1.3: call_message_receive_api()
1.3.1: validate_request()
1.3.2: collect_target_user_message
1.3.3: result message list
1.4: result message list
1.5: display message list

4.2 Design Rationale

As the LINGGGO project is meanly web-based application and we want to reserve the potential to later extend its capabilities for other platform development. We adopt the idea of REST and we mainly use 3-tier architecture in our system. The presentation layer is done by the web server and client browser and this tier is only in charge of maintaining communication between the client and the server and little business logic is implemented here to keep the client simple (but heavy user interface setup). The second and third tier are currently combined together and lay on the API server. The API server is responsible for processing client's request and sending back corresponding result. This layer is only in charge of business logic and data management and it has nothing to do with how to present the data to the client. This kind of architecture has several advantages: a. This system architecture reduces the overhead between the client and server by eliminating unnecessary data transfer between the client and server as we avoid transferring duplicate HTML and other static resources b. This system architecture reduces the cohesion of the development process between the front-end and back-end development so that both parts of the team can focus on their own development environment as AJAX technology is platform-independent and REST system architecture determines the correct how to parse the AJAX data c. This system architecture can also be easily converted into distributed structure if we encounter a bottleneck on a single server which runs the entire API services.

5. Architectural Styles, Patterns and Frameworks

Table 30 : Architectural Styles, Patterns and Frameworks

Name	Description	Benefits, Costs, and Limitations
Code Igniter	PHP Model-View-Controller framework	Benefits: 1. Easy connection to the MySQL database system 2. Easy control of the traditional Model-View-Controller architecture setup 3. Most of the back-end team members are familiar with PHP language and Code Igniter framework. Cost: 1. Free Limitations: 1. Performance Issue with PHP language
3-tier architecture	The system consists of three main parts: presentation, business and model layer	Benefits: 1. Clear structure and responsibility of each layer Cost: 1. Free, no patent on this architecture Limitations: 1. Overhead between the layers. The layers have to send extra data during communication 2. Tend to increase the complexity of the system architecture
AngularJS	Javascript framework	Benefits: 1. Clear structure for the front-end system development 2. Support for multiple platform Cost: 1. Free, open-sourced Limitations: 1. Some original libraries may perform as what the developer

		expect in the system, not reliable 2. Learning curve 3. Code-style problem from different programmers
REST	Representational state transfer (REST) is an software architecture style which defines the way to access a system.	Benefits: 1. Distributed Resources System which has high scalability 2. Platform-independent 3. Reduce the overhead between the client and server Cost: 1. Free Limitations: 1. Error state detection 2. Detailed documentation required 3. Good API design