Bilkent University
Department of Computer Engineering

Senior Design Project
2336
Lingui

Detailed Design Report

21902051, Mustafa Oğuz DUMAN, oguz.duman@ug.bilkent.edu.tr
21902356, Deniz Berkant DEMİRÖRS,
berkant.demirors@ug.bilkent.edu.tr
21903103, Emirhan Nadir KARAMAN,
nadir.karaman@ug.bilkent.edu.tr
21903031, Olcaytu GÜRKAN, olcaytu.gurkan@ug.bilkent.edu.tr
21901789, Enis ÖZER, enis.ozer@ug.bilkent.edu.tr

Uğur GÜDÜKBAY

Erhan DOLAK
Tağmaç TOPAL
Selim AKSOY

13.03.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

Detailed Design Report

Lingui: A Personalized Language Learning App Using Videos and Spaced Repetition

1 Introduction

Most people use the same approach when it comes to learning and teaching new languages: the skill-building approach. This approach is quite intuitive at first, suggesting that you learn the language by consciously learning grammar and applying it when producing output [1]. However, fluency requires more than that: a person must be able to conjugate verbs, choose the correct words, put the words in the correct order, etc. in a very short time, which can only be achieved by acquiring the language. Therefore, we reject the skill-building approach all the other language learning apps and the conventional language learning methods follow and accept a completely different approach: the Comprehensible Input Hypothesis, which can be summarized as follows: “We all acquire language in the same way: by understanding messages” [1].

To speak a language fluently, one needs to effortlessly construct new sentences. This instinct can be developed by acquiring the language, not by consciously learning the grammar rules [2, 3, 4]. Most native speakers do not know the grammar of their native language, despite being fluent in the language. So, we know that fluency is not achieved by learning the grammar of the language. This brings us to our first point: fluency is only possible by developing an instinct for the language.

Learning the definition is not enough to develop an instinct for the words in a language. You also need to see the word in hundreds of different contexts to have a feel for how the word should be used. Every time you hear a word and understand it, the better your instinct becomes, and eventually, you know how the word is used so well that you can use it in your sentences, too. In other words, you acquire the word. Here is our second point: the meaning of a word is acquired when it is seen and understood many times in different contexts.

We did not learn our native language by checking the dictionary each time we heard a word we did not know. We heard it in different contexts, and eventually, we understood the word. After we developed an instinct for the language, we started to speak. You can get the meaning of a word you do not know, or a grammatical structure from context, if you understand the meaning of the overall sentence [1, 3, 4]. To sum it up, the third point is that we can understand the meaning of a word if it is in a sentence where we understand the general meaning.

Lingui combines this approach and the three ideas mentioned. We want our users to watch YouTube videos in such a way that the language level of the video is never too easy or too difficult for the user. We will achieve this by tracking the vocabulary of every user. To ensure that the user understands the meaning of an unknown word in

context, we want to give the word in sentences that do not contain any other unknown words. After seeing the words in such sentences, a certain number of times, a spaced repetition system will be used to test the user if they learned the word correctly, and to retain this vocabulary knowledge.

Our goal is not to teach the users some phrases they can use in situations they encounter. Language books and other language-learning apps already do that. Our goal is something bigger: to make our users fluent.

1.1 Purpose of the System

Our system aims to play a supplementary role in language learning alongside traditional methods for learning foreign languages by benefiting from spaced repetition.

The primary use-case of our system is to make the users hear the vocabulary tailored to themselves in recommended videos. This method aims both to achieve learning by giving context to the corresponding words and also making the process of learning less boring and more entertaining by using carefully picked YouTube content.

Also, our system will test the users' understanding and long-term learning of the vocabulary via cloze tests.

To conclude, our system will serve users to learn vocabulary in a juiced up way. We believe that our system will make the process of learning new languages less boring and more connected to daily life.

1.2 Design Goals

1.2.1 Usability

The UI of the application will be Turkish for the initial users to understand. Additional languages will be supported as the application user base grows. Also, the user interface has to be very easy to understand both in order to make the user learn quicker and also for better user experience.

1.2.2 Scalability

Even though the application will start with very few users, it is intended to have many. In this case, the servers must be able do the complex tasks of video finding for every user separately.

1.2.3 Performance

Even though our servers will not be finding videos corresponding to the users' "words to learn list" in real time, performance is still an issue. Firstly, the video and the subtitles must be synchronous. Also, the app must have a low response time in order to achieve a good user experience.

1.2.4 Efficiency

Apps with video content can use big amounts of energy. The application must be energy efficient both for the environment and the user device's battery. In order to do that, the application will not perform the calculation for finding videos on device but rather on our server and only send the corresponding video information to the device.

1.2.5 Security

Even though the data that this app gathers is not sensitive, it is still user data and needs to be protected. To do that, we will use industry standard technologies in our servers and provide user authentication.

1.2.6 Extensibility

Even though Lingui will start its lifecycle as a mobile app, it may also have a website and/or desktop application. Also, the app will probably grow with additional features. In order to achieve those, the documentation needs to be systematic and open to change, and platform specific tools should be used as little as possible during development. The source code will be written in a tidy manner, refactored as much as possible and comments will be added when necessary..

1.2.7 Accessibility

Lingui will be available both on Google Play Store and App Store. The language of the app will be in Turkish in order for our initial users to use it easily. As the app grows, additional language support will be added. Also, the interface design of the app favors easy usage.

1.2.8 Maintainability

The features given in in extensibility (refactoring, comments, etc.) will also provide maintainability for the application.

1.2.9 Legality

The user data gathered from the application will only be used in our own servers for transaction related to our application and will not be shared with third parties.

1.3 Definitions, Acronyms and Abbreviations

- SRS: Spaced Repetition System.
- AWS: Amazon Web Services
- API: Application Programming Interface
- UI: User Interface

1.4 Overview

Lingui will be a mobile application designed for both Android and iOS users to learn vocabulary and practice their desired language by being exposed to words and sentences. For both accessibility and feedback purposes the application will be free and available on the mobile application stores of the given operating systems. For future profits, it is planned to put ads on different parts of the app.

In order to give you the reader a good preliminary understanding of the proposed system, we created a brief overview of the system using simpler terms. The application will require an account from each user in order to keep track of their learning process. Each new user will be assessed on their personal language knowledge and be presented with personalized content according to that assessment. The users will also have lists of words that they “do not know”, “have learnt” and are “in progress to learn”. According to these lists of words the application will find YouTube videos that contain these words. The user will be presented with these words in order both to teach new words and make the memory of the learnt words more concrete through a spaced repetition of the word. These videos will feature clickable subtitles separate from YouTube. While watching these videos, the user will also be able to select the words in the video that they do not know. If the user selects a word, they will first be presented with a small window showing the meaning of the word. The user will be able to add this word to a list if they wish to. After their learning process, the users will also be able to take short written quizzes in order to test their comprehension of the vocabulary.

The application will be developed using Flutter framework for frontend, and a microservices architecture which uses several different frameworks for different services in the backend. Also, several different technologies such as AWS and Docker will be used in the project.

In order to achieve the security of the user data, all accounts will be authenticated. A further explanation of the requirements of the application will be given in the next chapters.

2 Current Software Architecture

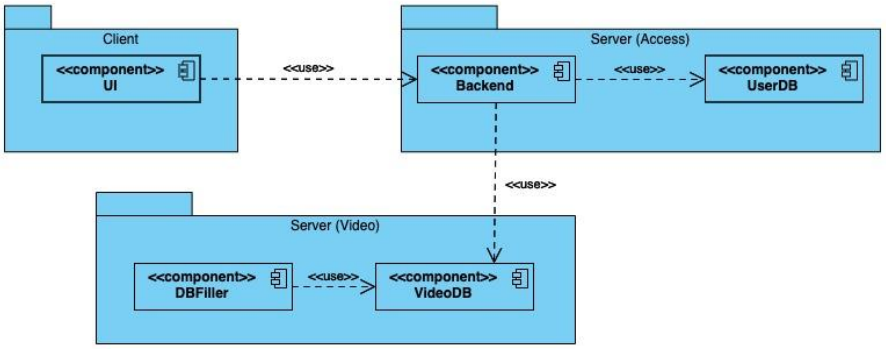
Client side is using the MVC architectural pattern that separates the data, control and presentation layers. For the server side, we are using microservice architecture so that when one service is down, the application does not fail.

3 Proposed Software Architecture

3.1 Overview

To provide a user-friendly and efficient language learning experience, Lingui's architecture is designed to enable seamless interactions between its various subsystems. To achieve this, each subsystem's access control and architecture are defined and analyzed through decomposition and protocol usage at both hardware and software levels. This includes the analysis of data control and dependencies, authorization and access control, and global software control. By utilizing this approach, Lingui ensures the design of a secure and efficient language learning app.

3.2 Subsystem Decomposition



Lingui is a client-server app with a client-side UI written in Flutter for mobile devices (Android & iOS) and a server-side divided into two subsystems.

The first subsystem, which grants access to the UI, has two components: a vocabulary subsystem that provides videos and cloze tests, and a user authentication subsystem that handles login and user data management. The second subsystem is a video subsystem that scans YouTube for videos, analyzes the transcripts to determine which words they contain, and fills the database with video data.

The video subsystem has two components: a video analysis engine that performs a scanning algorithm on the transcripts, and a database interface that manages the storage and retrieval of video data. Communication between the client and server is handled via HTTP requests and responses. The server-side uses a three-tier architecture with a routing tier, a logic tier, and a data tier. The routing tier handles communication between the client and server, the logic tier contains the business logic of the app and handles user authentication, and the data tier manages the storage and retrieval of data from the database.

The whole server side is deployed using AWS and the data management is performed using PostgreSQL. Overall, Lingui's subsystem decomposition provides a clear and organized breakdown of its architecture, enabling effective management and development of the app.

3.3 Hardware Software Mapping

Lingui is a client-server application that runs entirely on cloud-based infrastructure provided by Amazon Web Services (AWS). The client side of the app is a mobile application built with Flutter, which runs on iOS and Android devices. The server side of the app is divided into two subsystems. The first subsystem provides access to the user interface, including features such as video playback and cloze tests. This subsystem is built using the Flask web framework, which is written in Python and runs on AWS.

The second subsystem is responsible for video processing and storage. It scans YouTube for relevant videos and analyzes their transcripts to identify vocabulary words. The subsystem then stores the videos and their associated metadata in a PostgreSQL database, which runs on an Amazon RDS instance. This subsystem is built using a combination of Python scripts and AWS Lambda functions.

The entire system communicates using REST APIs. The client side of the app communicates with the server using HTTP requests and JSON responses, while the server side of the app communicates with the database using the psycopg2 Python library.

To conclude, in terms of hardware, the Lingui app runs entirely on AWS resources.

3.4 Persistent Data Management

Lingui being an application that aims to teach languages through watching videos and cloze tests, it needs to have a large stack of videos, their sentences/subtitles and the words in the sentences/subtitles accessible in a way that does not affect user

experience. Since we are using videos from Youtube, we do not need to store them in any custom file storage but we store the ids with other relevant information on the database. To not overload the system, after executing a query the results will be cached on the client's device for a small period of time so repetitive requests will not pressure the back-end system.

3.5 Access Control and Security

Lingui's authentication system consists two types of users:

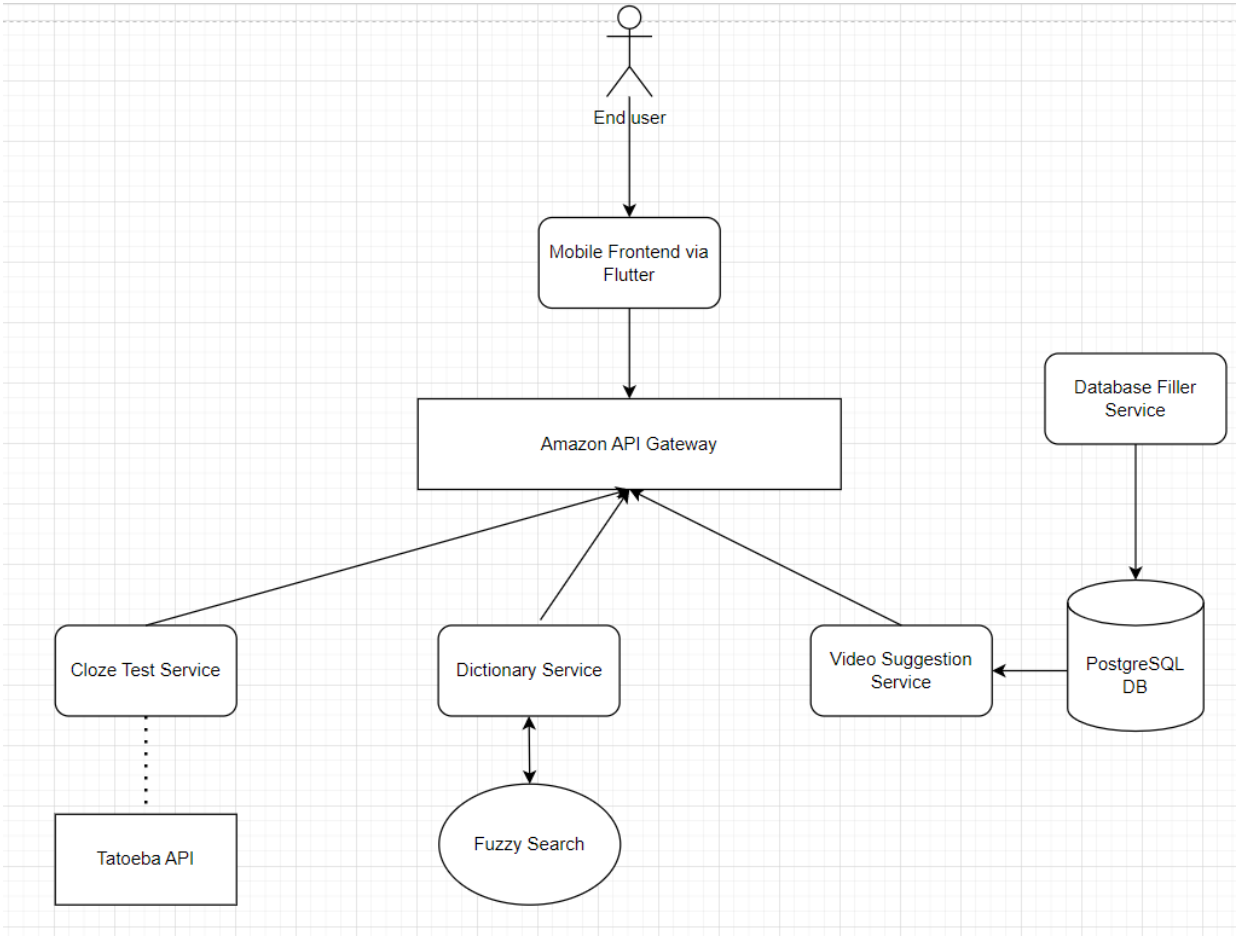
- Server Administrator/Developer: Server side developers
- Customer client: Mobile phone of the users

Client side of the application will use HTTP requests. To verify the requests coming from a valid client, we will use Amazon Cognito's authentication system that allows users to sign up via Google or Apple accounts. After the sign-up/sign-in, the system will provide a JWT token which the server side checks the validity before doing any kind of read/write operations.

Since Lingui is a B2C application, it may collect information such as geospatial data, contacts, usage data, search history and diagnostics. Therefore, permission to collect such data will be asked in terms of agreement.

4 Subsystem Services

In the diagram below, rounded rectangles denote our subsystem services:



Firstly, we have our frontend service as a whole. The reason it is not divided into subsystems is due to the fact that we have only one person working at frontend, Deniz. The rest of the rounded rectangles belong to the backend team. Database Filler Service fills our database with the specifically chosen language-to-language videos. Cloze Test Service interacts with Tatoeba's external API to present the user the cloze test designed for him/her. Dictionary Service uses the Fuzzy Search method (implemented by us) to present a smoother dictionary experience to the end user. Video Suggestion Service interacts with our database to present the most suitable video for the language learning for the end user. Those backend services were divided among the backend team with respect to each member's workload.

5 Test Cases

Tests Related to /list-user-words Endpoint

Parameters:

- user_id: string
- language: string
- page_number: int
- rows_per_page: int

Test ID: F-LUW-01
Type / Category: Functional
Summary: Wrong user_id (user with user_id does not exist)
Procedure:

- Send a GET request to /list-user-words, with a user_id that is known not to exist:

Expected Outcome: Error: user with user_id does not exist
Priority / Severity: Critical

Test ID: F-LUW-02
Type / Category: Functional
Summary: Language is not valid
Procedure:

- Send the GET request above, with language = any string with more than 2 letters. (Language codes are 2-lettered, so this way it is guaranteed to be invalid)

Expected Outcome: []
Priority / Severity: Critical

Test ID: F-LUW-03
Type / Category: Functional
Summary: Page number is invalid.
Procedure:

- Send the GET request above, with a non-integer page_number.

Expected Outcome: Error: page number is not an integer.
Priority / Severity: Critical

Test ID: F-LUW-04
Type / Category: Functional
Summary: rows_per_page is invalid.
Procedure:

- Send the GET request above, with a non-integer rows_per_page.

Expected Outcome: Error: rows_per_page is not an integer.
Priority / Severity:

Test ID: F-LUW-05
Type / Category: Functional
Summary: page_number is not positive.
Procedure:

- Send the GET request above, with page_number <= 0.

Expected Outcome: Error: page number should be positive.
Priority / Severity: Critical

Test ID: F-LUW-06
Type / Category: Functional
Summary: rows_per_page is not positive.
Procedure:

- Send the GET request above, with rows_per_page <= 0.

Expected Outcome: Error: rows_per_page should be positive.
Priority / Severity: Critical

Test ID: F-LUW-07
Type / Category: Functional
Summary: page_number is a very big number.
Procedure:

- Send the GET request above, with a very big number.

Expected Outcome: Success: []
Priority / Severity: Critical

Test ID: F-LUW-08
Type / Category: Functional
Summary: Language does not exist in the user's list of languages.
Procedure:

- Send the GET request above, with a language that does not exist in the user's list of languages.

Expected Outcome: Error: language does not exist in the user's list of languages.
Priority / Severity: Critical

Test ID: F-LUW-09
Type / Category: Functional
Summary: All input parameters are valid.
Procedure:

- Send a GET request to /list-user-words endpoint with valid parameters.

Expected Outcome: Success: a list of words that the user has started learning with pagination.
Priority / Severity: Mid

Tests Related to /check-answer Endpoint
Parameters:

- user_id: string
- word: string
- language: string
- answer_is_correct: boolean

Test ID: F-CA-01
Type / Category: Functional
Summary: User with user_id does not exist.
Procedure:

- Send a POST request to /check-answer.

Expected Outcome: Error: user with user_id does not exist.
Priority / Severity: Critical

Test ID: F-CA-02
Type / Category: Functional
Summary: (word, language) pair does not exist in the user's vocabulary.
Procedure:

- Send the POST request above, with a (word, language) pair that does not exist in the vocabulary list of the user.

Expected Outcome: Error: word does not exist in user's vocabulary.
Priority / Severity: Critical

Test ID: F-CA-03
Type / Category: Functional
Summary: The answer of the user in the cloze test is correct, and 1 <= strength <= 7.
Procedure:

- Send the POST request above, with answer_is_correct = True.

Expected Outcome: Success: The strength of the word is incremented by one.
Priority / Severity: Critical

Test ID: F-CA-04
Type / Category: Functional
Summary: The answer of the user in the cloze test is correct, and strength = 8.
Procedure:

- Send the POST request above, with answer_is_correct = True.

Expected Outcome: Error: word is already mastered.
Priority / Severity: Minor

Test ID: F-CA-05
Type / Category: Functional
Summary: The answer of the user in the cloze test is incorrect.
Procedure:

- Send the POST request above, with answer_is_correct = False.

Expected Outcome: Success: The strength is decreased to 1.
Priority / Severity: Critical

Tests Related to /get-unknown-sentences Endpoint
Parameters:

- user_id: string
- language: string
- page_number: int
- rows_per_page: int

Test ID: F-GUS-01
Type / Category: Functional
Summary: User with user_id does not exist.
Procedure:

- Send a GET request to /get-unknown-sentences.

Expected Outcome: Error: User with user_id does not exist.
Priority / Severity: Critical

Test ID: F-GUS-02
Type / Category: Functional
Summary: Language does not exist in the user's list of learned languages.
Procedure:

- Send the GET request above.

Expected Outcome: Error: language does not exist in the user's list of learned languages.
Priority / Severity: Minor

Test ID: F-GUS-03
Type / Category: Functional
Summary: Language is not valid.
Procedure:

- Send the GET request above, with len(language) >= 3.

Expected Outcome: Error: Language is not supported.
Priority / Severity: Minor

Test ID: F-GUS-04
Type / Category: Functional
Summary: max_unknown_count is smaller than 0.
Procedure:

- Send the GET request above, with max_unknown_count < 0.

Expected Outcome: Error: max_unknown_count should be an integer >= 0.
Priority / Severity: Critical

Test ID: F-GUS-05
Type / Category: Functional
Summary: max_unknown_count is equal to 0.

Procedure:

- Send the GET request above, with max_unknown_count = 0.

Expected Outcome: Success: If the user is new, [].
Priority / Severity: Critical

Test ID: F-GUS-06
Type / Category: Functional
Summary: max_unknown_count is a number between 1-15.
Procedure:

- Send the GET request above, with 0 < max_unknown_count <= 15.

Expected Outcome: Success: If user vocab is empty, all sentences of word count = max_unknown_count.
Priority / Severity: Critical

Test ID: F-GUS-07
Type / Category: Functional
Summary: max_unknown_count is a very big number.
Procedure:

- Send the GET request above, with max_unknown_count = 100.

Expected Outcome: Success: All sentences in that language.
Priority / Severity: Critical

Test ID: F-GUS-08
Type / Category: Functional
Summary: quality = True, language = a Latin-script language.
Procedure:

- Send the GET request above, with quality = True, and language = ‘en’.

Expected Outcome: Success: the expected English sentences, starting with a capital letter and ending with an exclamation mark, a question mark, or a period.
Priority / Severity: Critical

Tests Related to /learn-word Endpoint
Parameters:

- user_id: string
- word: string
- language: string

Test ID: F-LW-01
Type / Category: Functional
Summary: quality = True, language = ‘ru’
Procedure:

- Send the GET request above, with quality = True, and language = ‘ru’.

Expected Outcome: Success: the expected Russian sentences, starting with a Cyrillic capital letter, and ending with an exclamation mark, a question mark, or a period.
Priority / Severity: Critical

Test ID: F-LW-02
Type / Category: Functional
Summary: quality = False
Procedure:

- Send the GET request above, with quality = False.

Expected Outcome: Success: The expected sentences in that language.
Priority / Severity: Critical

Test ID: F-LW-03
Type / Category: Functional
Summary: Page number is invalid.
Procedure:

- Send the GET request above, with a non-integer page_number.

Expected Outcome: Error: page number is not an integer.
Priority / Severity: Critical

Test ID: F-LW-04
Type / Category: Functional
Summary: rows_per_page is invalid.
Procedure:

- Send the GET request above, with a non-integer rows_per_page.

Expected Outcome: Error: rows_per_page is not an integer.
Priority / Severity: Critical

Test ID: F-LW-05
Type / Category: Functional
Summary: page_number is not positive.
Procedure:

- Send the GET request above, with page_number <= 0.

Expected Outcome: Error: page number should be positive.
Priority / Severity: Critical

Test ID: F-LW-06
Type / Category: Functional
Summary: rows_per_page is not positive.
Procedure:

- Send the GET request above, with rows_per_page <= 0.

Expected Outcome: Error: rows_per_page should be positive.
Priority / Severity: Critical

Test ID: F-LW-07
Type / Category: Functional
Summary: page_number is a very big number.
Procedure:

- Send the GET request above, with a very big number.

Expected Outcome: Error: Page number out of bounds
Priority / Severity: Critical

Test ID: F-LW-08
Type / Category: Functional
Summary: User with user_id does not exist.
Procedure:

- Send a POST request to /learn-word, with an unexisting user_id.

Expected Outcome: Error: user with user_id does not exist.
Priority / Severity: Critical

Test ID: F-LW-09
Type / Category: Functional
Summary: Language does not exist in the user's list of learned languages.
Procedure:

- Send the POST request above, with a language the user isn't learning.

Expected Outcome: Error: Language does not exist in user's list of learned languages.
Priority / Severity: Critical

Test ID: F-LW-10
Type / Category: Functional
Summary: User has already learned the word given in the parameters.
Procedure:

- Send the POST request above, with a word that the user with user_id has already learned.

Expected Outcome: Error: User has already learned that word.
Priority / Severity: Mid

Tests Related to /add_word_to_user Endpoint
Parameters:

- user_id: string
- word: string
- language: string

Test ID: F-AWTU-01
Type / Category: Functional
Summary: User given with user_id parameter does not exist.
Procedure:

- Send the request above with an invalid user_id.

Expected Outcome: Error: User does not exist.
Priority / Severity: Critical.

Test ID: F-AWTU-02
Type / Category: Functional
Summary: The languages that the user is aiming to learn does not contain the language in the parameters.
Procedure:

- Send the request above with an invalid user_id.

Expected Outcome: Error: Language does not exist in the user's languages list.
Priority / Severity: Mid.

Test ID: F-AWTU-03
Type / Category: Functional
Summary: User given with user_id parameter has not learned the word
Procedure:

- Send the request above with a word
- **Expected Outcome:** Success: user starts learning that word with initial level of that word = 1

Priority / Severity: Critical.

Test ID: F-AWTU-04
Type / Category: Functional
Summary: The user has already started learning the word
Procedure:

- Send the request above with a word that the user has already started learning.

Expected Outcome: Success: The word is completely learned if the level goes above 7 otherwise the level is incremented by one.
Priority / Severity: Critical.

Tests Related to /add_language_to_user Endpoint
Parameters:

- user_id
- language

Test ID: F-ALTU-01
Type / Category: Functional

Summary: User given with user_id parameter does not exist.
Procedure:

- Send the GET request above, with an invalid user id.

Expected Outcome: Error: Invalid user id.
Priority / Severity: Critical

Test ID: F-ALTU-02
Type / Category: Functional
Summary: The language does not exist in the user's list of languages.
Procedure:

- Send a POST request to /add-language-to-user with an existing user_id.

Expected Outcome: Success: New language is added to the user's language list.
Priority / Severity: Critical

Test ID: F-ALTU-03
Type / Category: Functional
Summary: The language already exists in the user's list of languages.
Procedure:

- Send a POST request to /add-language-to-user with an existing user_id.

Expected Outcome: Error: Language already exists in the user's list of languages.
Priority / Severity: Critical

Tests Related to /transcript Endpoint
Parameters:

- video_id

Test ID: F-T-01
Type / Category: Functional
Summary: The video_id does not exist.
Procedure:

- Send a GET request to /transcript with a non-existing video_id.
    - video_id: string

Expected Outcome: Error: video_id does not exist
Priority / Severity: Mid

Test ID: F-T-02
Type / Category: Functional
Summary: The video_id does not exist.
Procedure:

- Send a GET request to /transcript with an existing video_id.
    - video_id: string

Expected Outcome: Success: The transcript of the video will be returned as response.
Priority / Severity: Mid

Tests Related to /add-video-to-blacklist Endpoint
Parameters:

- video_id

Test ID: F-AVTB-01
Type / Category: Functional
Summary: The video_id does not exist.
Procedure:

- Send a GET request to /add-video-to-blacklist with a non-existing video_id.

Expected Outcome: Error: video_id does not exist
Priority / Severity: Mid

Test ID: F-AVTB-02
Type / Category: Functional
Summary: The video_id exists.
Procedure:

- Send a GET request to /add-video-to-blacklist with an existing video_id.

Expected Outcome: Success: video is added to the blacklist
Priority / Severity: Mid

Tests Related to /search_word Endpoint
Parameters:

- user_input
- language

Test ID: F-SW-01
Type / Category: Functional
Summary: User_input is blank
Procedure:

- Send a GET request to /search_video with blank user_input and a valid language.

Expected Outcome: Success: an empty list is returned.
Priority / Severity: Mid

Test ID: F-SW-02
Type / Category: Functional

Summary: Both user_input and language are valid.
Procedure:

- Send a GET request to /search_video with a non-empty user_input and valid language.

Expected Outcome: Success: a list of at most 10 words similar to user_input in that language are returned.
Priority / Severity: Mid

Test ID: F-SW-03
Type / Category: Functional
Summary: Language is invalid.
Procedure:

- Send a GET request to /search_video with an invalid language.

Expected Outcome: Error: Invalid language.
Priority / Severity: Mid

Tests Related to /tatoeba Endpoint
Parameters:

- word
- native_language
- target_language

Test ID: F-TAT-01
Type / Category: Functional
Summary: Word does not exist in the target_language
Procedure:

- Send a GET request to /tatoeba with a non-existing word in the target_language parameter.

Expected Outcome: Success: an empty list is returned.
Priority / Severity: Mid

Test ID: F-TAT-02
Type / Category: Functional
Summary: Word exists in the target_language
Procedure:

- Send a GET request to /tatoeba with a non-existing word in the target_language parameter.

Expected Outcome: Success: An example sentence in target_language, translation in native_language and word itself.
Priority / Severity: Mid

Tests Related to /learn-multiple-words Endpoint
Parameters:

- user_id: string
- word_list: string[]
- language: string
- page_number: int
- rows_per_page: int

Test ID: F-LMW-01
Type / Category: Functional
Summary: All input parameters are valid.
Procedure:

- Send a GET request to /learn-multiple-words endpoint with valid parameters.

Expected Outcome: Success: words are added to the word_strength table with strength = 8.
Priority / Severity: Critical

Test ID: F-LMW-02
Type / Category: Functional
Summary: The language parameter is invalid
Procedure:

- Send a GET request to /learn-multiple-words endpoint with invalid language parameter, all other parameters are valid.

Expected Outcome: Error: Language does not exist.
Priority / Severity: Critical

Test ID: F-LMW-03
Type / Category: Functional
Summary: All input parameters are in a valid format but user is not learning the given language.
Procedure:

- Send a GET request to /learn-multiple-words endpoint with valid parameters.

Expected Outcome: Erorr: User is not learning the given language
Priority / Severity: Critical

Test ID: F-LMW-04
Type / Category: Functional
Summary: All input parameters are valid.

Procedure:

- Send a GET request to /learn-multiple-words endpoint with valid parameters.

Expected Outcome: Success: words are added to the word_strength table with strength = 8.
Priority / Severity: Critical

Test ID: NF-LMW-01
Type / Category: Non-functional
Summary: All input parameters are valid, 50 words are being learned at once.
Procedure:

- Send a GET request to /learn-multiple-words endpoint with valid parameters and 50 words.

Expected Outcome: Success with request taking less than 2 seconds.
Priority / Severity: Critical

Test ID: NF-LMW-02
Type / Category: Non-functional
Summary: All input parameters are valid, 100 words are being learned at once.
Procedure:

- Send a GET request to /learn-multiple-words endpoint with valid parameters and 100 words.

Expected Outcome: Success with request taking less than 4 seconds.
Priority / Severity: Critical

6 Consideration of Various Factors in Engineering Design

Our system's main purpose is to teach English to the native Turkish speakers (Initially, this was our aim. Now our app is more extensive). Since language learning is a domain that is not really affected by the general ongoing of a person's daily life, our system does get affected on things like public health and safety. Public welfare can affect our system, because if public welfare is too low, learning a language might not be the first priority of a person. So, our system should be able to run on old phones and should not consume too much internet. Global factors do not really affect our system, but English being the global language of the world makes our app more desirable. Also, since there are more languages supported in our app, it is even more desirable right now. In the Turkish education system, English is taught by a hard emphasis on rules, grammar and test questions. Since education is strongly connected with cultural and social factors, this is the main paradigm in Turkish society. We want to change the emphasis on rules and grammar with an exposure system, but to keep

the cultural familiarity, we will keep questions in the form of cloze tests. Also, socially, people are more inclined to watch short videos because of the decline of the attention span. Thus, we might want to keep our videos short.

Table 1: Factors that can affect analysis and design.

	Effect level	Effect
Public health	0	Public health has no effect on our system.
Public safety	0	Public safety has no effect on our system.
Public welfare	5	Public welfare can affect the usage of our system because if the general welfare is very low, learning a new language might not be the priority of the people.
Global factors	4	English being the global language of the world can attract/lure users to our system because it is a desired language to learn.
Cultural factors	6	In the Turkish education system, English is not taught as we will teach in our system. Since education and culture are closely related, it might take a time for users to get used to our system.
Social factors	6	Since people like short videos more due to the decline of the attention span, we might choose videos with short durations.

7 Teamwork Details

7.1 Contributing and Functioning Effectively on the Team

Since our team's mindset is like "From each according to his ability, to each according to his needs", we did not have any problem such as forcing a teammate to contribute to the project. Each member was proficient in an area before the project, so we just distributed people to the right parts, and each one of us functioned really well. Deniz is experienced and proficient in frontend development, so nearly carried all the work in the front side of the project. He used Flutter to create a very eye-appealing mobile app. Enis is interested and experienced in DevOps and infrastructure, so we let

him deal with all of the work on that part of the project. He deployed our project's servers to AWS. Emirhan, Oğuz and Olcaytu are proficient in backend development. So they did a great job of effectively dividing the workload on backend, and created functioning backend services using Flask and Spring.

7.2 Helping Creating a Collaborative and Inclusive Environment

Every member of the team has been friends since freshman year, and been in numerous projects before together. So, working as a team with this group of people was not anything new, or surprising. Before even picking a group meeting time, every member's availability was asked, and every member tried to create availability to match the other members availability. Since some members worked part-time jobs, if a member was tight on his other schedules, other members picked up his work with a premise that the busy member will do "overwork" in the future for that member. Even in small matters like choosing the authentication framework, we took all member's ideas and respected their past experiences with the frameworks so that we can pick the most fit tools for our project, aiming that every team member would be comfortable in the project domain.

7.3 Taking Lead Role and Sharing Leadership on the Team

We do not have any specified leader. Every member has the full leadership responsibility on the project, and is willing to call the shots if necessary. If a service is needed for the frontend, Deniz sets a deadline for the backend team for that service. If a service needs to be deployed to AWS EC2, the backend team collaboratively sets a deadline for Enis. If the deadline of a report is coming, whoever notices it first alerts the team through WhatsApp group and sets a meeting. While doing such stuff, we also try to respect other member's schedules and availability, which prevents toxic leadership amongst the team. Since this team has been on numerous projects before, we hardly take offense from each other when taking "orders", and this helps us to get things done faster, and clearer.

8 Glossary

SRS: Spaced Repetition System.
AWS: Amazon Web Services
API: Application Programming Interface
UI: User Interface

9 References

[1] Krashen, S. (2002). The comprehension hypothesis and its rivals. Selected papers from the Eleventh International Symposium on English Teaching/Fourth Pan Asian Conference. pp. 395-404. Taipei: Crane Publishing Company.

[2] Krashen, S. (2004). Applying the comprehension hypothesis: Some suggestions. Paper presented at 13th international symposium and book fair on language teaching (English Teachers Association of the Republic of China), Taipei, Taiwan. Retrieved on 17 Oct 2022 from http://www.sdkrashen.com/content/articles/eta_paper.pdf.

[3] Rodrigo, V (2006). The amount of input matters: Incidental acquisition of grammar through listening and reading. The International Journal of Foreign Language Teaching, 2 (1). 10-13.

[4] Ponniah, R. J. (2008). Free voluntary reading and the acquisition of grammar by adult ESL students. The International Journal of Foreign Language Teaching, 4 (1) 20-22.