Bilkent University
CS491 Senior Design Project 1
Fall 2025
Project Specification Document

Project Name: StreamLang
Team ID: T2533

Team Members:
22202995 - Mehmet Emin Avşar
22203632 - Uygar Bilgin
22202913 - Göktuğ Ozan Demirtaş
22203805 - Ruşen Ali Yılmaz
22203239 - Can Tücer

1. Introduction

1.1. Description

Language learners require listening content to complement their studies; however, it is a pain to find content that is suitable for one's level. If you find some listening content that is pushing you to grow in terms of understanding complex grammar structures, it might be too easy in terms of vocabulary, or vice versa. Our solution allows the language learner to select precise grammar structures, specific vocabulary, content type, speaking speed and voice which are then used to create a listening stream uniquely tailored for the user's growth. Selected vocabulary and grammar structures are sprinkled into the listening stream using the scientifically proven *Spaced Repetition System* to optimize memory recall. Initial listening stream configuration templates are available as well as a Dynamic Mode which updates the vocabulary and grammar level automatically in time, enabling a passive learning experience.

1.2. High Level System Architecture & Components of Proposed Solution

StreamLang will be composed of a microservice architecture, with logical and performance-based separation of modules. [Fig 1.] shows the architecture of the StreamLang product.

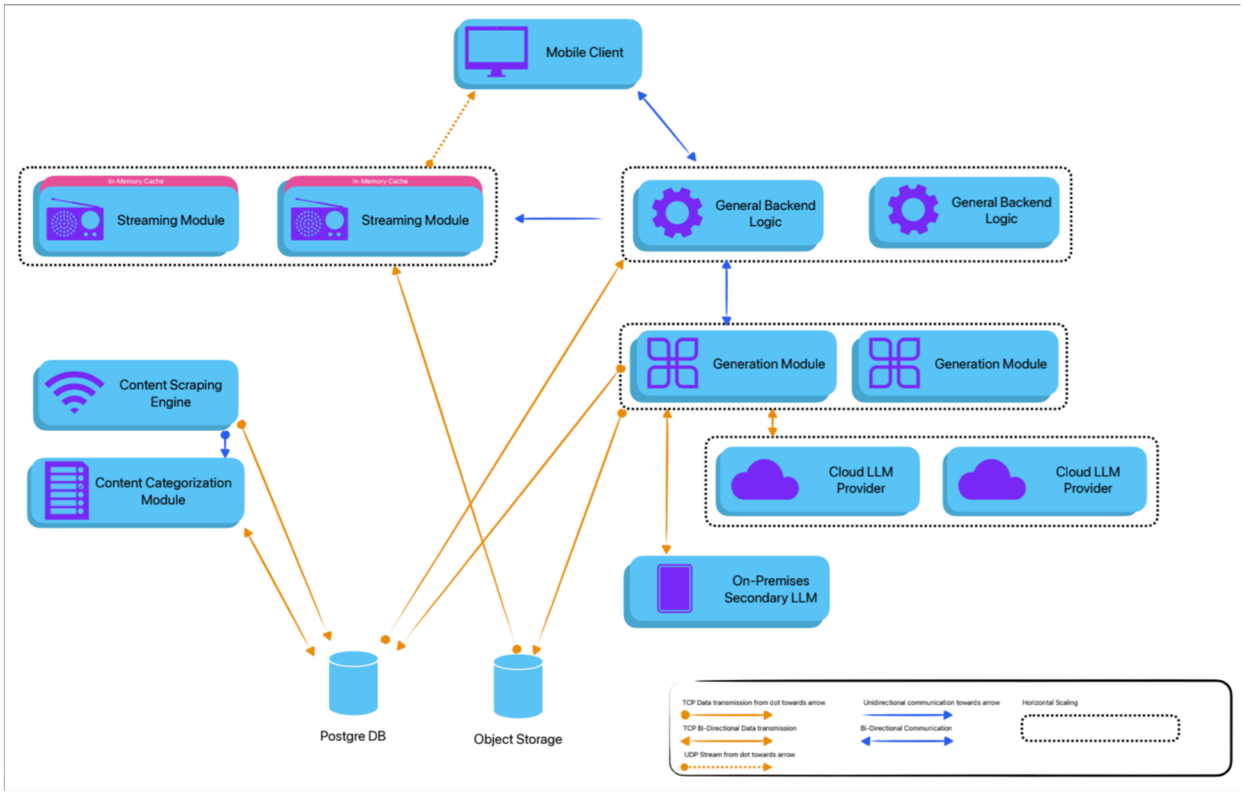

Fig 1. The Decomposition of The Architecture

The Mobile Client: This module is where the user experience begins. Its a cross platform mobile application developed using Google's flutter framework. We chose to build an application instead of a web app for performance and local caching reasons. An application solution has the benefit of persistent storage and higher performance due to a lower level of access than web.
General Backend Logic: This module is the main orchestrator of the backend. It handles IAM and overall logic. It is a horizontally scaling serverless microservice. When a user wants to listen to an audio stream, this module checks the database for cached responses, communicates with the "Generation Module" for the generation of new sentences and audio responses. It also informs the "Streaming Module" to indicate which audio streams, in which order, should be streamed to the user.
Generation Module: When a user wants an audio stream, the generation of the textual content of the desired audio stream is orchestrated here. This generation module does not, itself, generate the content: it manages its delegation. It communicates with, and analyses the responses of a Cloud LLM Provider and an on-premises secondary LLM.
LLM Providers: We separate the LLM providers for cost management. Cloud handles the generation of text generation, while on premises LLM handles the checking of those responses, to make sure the response is actually within the context requested by the user.
Object Storage and PostgreSQL DB: Object Storage is used to store generated audio segments. This is done to avoid the performance cost of text-to-speech audio generation. Once the audio segment for a sentence has been generated, it will be stored in this object storage. When the same sentence is to be streamed, the previously generated audio segment from the object storage will be streamed, bypassing the text-to-speech system in the generation module. PostgreSQL DB will be handling the metadata related to the audio streams. It will also be the main database for user & any other related data. It was chosen because of the pgvector extension, which enables PostgreSQL to be used as a vector database, useful for AI applications.
Streaming Module: These streaming modules transmit the audio data to the client. They do this by streaming the audio segments to their local cache, joining them in real-time, and streaming to the user using a UDP audio stream.
Content Scraping Engine & Content Categorization Module: To ensure we can generate content related to the user's request, we have a content scraping engine. This engine crawls through legally usable content, scrapes and cleans it. This data is then saved to the Postgre DB, waiting for categorization. The Content Categorization Module analyzes the scraped content, categorizes, and classifies it depending on the topic, language rules, and the level of language used.

1.3. Constraints

1.3.1. Implementation Constraints

We are faced with several challenges when it comes to implementation. One of the most important challenges is the performance. Generative AI is an exceptionally performance heavy task, and providing customers with a fast, responsive, and accurate product requires a lot of constraints.

All modules, except the content scraping engine and the content categorization module, must be serverless.

The generation must be done asynchronously with the streaming, such that the audio stream can start while the audio segments are still being generated.

Audit trail for LLM calls, mostly needed for cost analysis, but also for cost analysis.
No user-controller code execution in any service.

Cleaning the input for any kind of injection type of attacks (SQLInjection, Prompt Injection).

Bottleneck prevention for streaming, meaning if an audio segment is still under generation when it's needed, another audio segment should be streamed so the service still appears fast, even when it is bottlenecked by generation.

1.3.2. Economic Constraints

The cost for cloud LLM providers must not exceed more than 40% of the income from a subscribed customer.

The object storage must be segmented such that audio segments that have not been used for more than 14 days are moved to colder storage, up to the archive storage to reduce costs.

No service, except the Generation Module, should use more than 1 GigaByte of system ram, so that cloud hosting costs are reduced to a minimum.

1.3.3. Ethical Constraints

There are various ethical constraints StreamLang faces. One constraint is regarding the storage of sensitive user data. Users will share their personal preferences and interests to find the streams they would like; thus, we will have to store some personal data users would most likely want to keep private. Here, to deal with this constraint and ensure data safety, we will have to encrypt data at rest and in transit. Additionally, we will also have to pay attention to account safety procedures, and possibly use measures like two-factor authentication. Also, we must provide users the option to delete their data from our databases whenever they want, even if they want to keep using StreamLang. Furthermore, this action should be automatically done after account deletions.

Another ethical constraint is illustrated with our stream content generation mechanism. As previously discussed, to improve the quality and enjoyability of the streamed content, we are planning to use scraped text data, such as book chapters, news journals, and real interviews. However, while doing this, we will have to ensure that the content owner allows the redistribution of their content for commercial purposes. Thus, we will have to analyze the license of contents we are planning to scrape in detail. Whenever necessary, we will have to give credit to content providers. To overcome this issue, we may prioritize using license-free contents, like classical books and publicly available information.

The third ethical constraint is also related to content generation. As we will scrape the data from external sources, and we won’t have any human intervention in the content selection stage, some of the content we are providing may be inappropriate for some people. In this regard, by analyzing the scraped data and developing content moderation algorithms, we can redact and remove obviously unsuitable content. However, there is always the risk of an individual being uncomfortable due to some content that would be credited as okay by the majority. For those cases, we are planning to allow users to block some content from being used in their streams, so they won't have a similar experience again.

Similarly, the scraped content may include some outdated/biased/incorrect information. Users may tend to believe and embrace the generated streams irresponsibly. This is dangerous for StreamLang as it may harm and expel our users. To overcome this, we will again have to carefully analyze the contents we are scraping. We may also allow users to give feedback on the generated streams. By evaluating the feedback either automatically or with human intervention, we can spot and remove such contents from our database without it affecting a large user base.

1.4. Professional and Ethical Issues

Some of the important professional and ethical issues StreamLang will face, like copyright, data safety, and user protection were previously discussed. Additionally, there may be some professional and ethical issues that do not arise from or affect our design and implementation choices directly. In this section, we will briefly mention those broader scope issues and will share the solutions we are planning to apply.

Firstly, and most importantly, at all times while developing StreamLang, we will adhere to the IEEE Standards Association Code of Ethics [1]. As given in the code,

- We will prioritize user well-being by applying best practices for interaction design and learning management.
- We will accept positive criticism by implementing multiple feedback mechanisms both for the generated contents and the application itself.
- We will make realistic claims and we will give honest estimations for all of our implementations and expectations by doing detailed planning.
- We will abide by the necessary laws of our target countries by doing research on necessary concepts like copyright infringements and user data management.
- We will maintain our technical competence by studying and following known best practices for any technical task and we will update our product accordingly throughout its lifecycle.
- We will treat every user equally, and we will refrain from doing any implementation that may harm any user group for any reason.

Outside this code, we will also follow some of our own rules to ensure StreamLang becomes the successful project we are willing to develop. While working on StreamLang, we will make sure that all of our members have similar, if not equal, workloads and every member gets proper credit for their work. As our work will include some data analysis to showcase the functionality and usability of StreamLang, we will make sure to be transparent and honest with all of our findings. Finally, of course, we will make sure every member of our team follows all of the rules mentioned above.

Moreover, as StreamLang aims to help individuals learn, there will be some teaching concepts we will have to follow and thus our project will include much more than some basic content generation and streaming logic. In a way, our project will include some psychological background. We will have to analyze and study how individuals learn better, especially in the context of language learning. We will strive to do research on topics like the concept of learning, memorizing methods, positive reinforcement, forming habits, etc. At every step, we will have to analyze the outcomes we find, and we will have to keep making adjustments until StreamLang becomes a viable alternative to other language learning methods.

1.5. Standards

For the Backend we use the REST API standard [2]. For modelling we use UML 2.5.1. We use the "Effective Dart" style guide for the frontend [3]. We use the Google Python style guide for the backend [4]. For the referencing style, we use IEEE 830 in our documentation. We write our internal documentation in markdown format, and give references using links.

2. Design Requirements

2.1. Functional Requirements

The application's main focus is creating structured listening streams for language learners that will help them attain fluency faster compared to listening to content not personally curated for them. The application will have a strong grammar-centric architecture where not only the vocabulary, but also the grammar structures that the user is exposed to will be tracked and then reintroduced at optimal times for memory retention according to SRS (Spaced Repetition System) principles, a method that has been shown to be effective in retaining information [5].

Dynamic Mode -for users who are willing to use it- will remove the need for the users to set up configuration templates themselves. In this mode, the user will first go through a brief assessment. According to the results of this assessment, a listening stream configuration will be set up, and then the configuration will automatically be updated in time as the user grows in their language abilities. User's growth in language skills will be assessed with intermittent quizzes administered through audio; if the user replies correctly to a question regarding the past tense, for example, their predicted skill in that grammar structure will increase and they will start to hear less of past tense and more of other, not yet learned grammar structures in their listening stream.

The specific scenarios that the user must be able to perform are listed as follows:

- Select the level of the vocabulary
- Select the type of content they are going to listen to
    - Article, story, individual sentences, translation of uploaded document, or some general text in the vein of the uploaded document
- Select whether they want to hear the translations

    - They can select to hear the translations after each sentence (only option for “individual sentences” type of content), or after the entire content
- Select the grammar structures they want to study.
    - Ex.: Past tense, present tense, perfect tense etc.
    - Select whether they want “Drill” mode or “Roaming” mode. In Drill mode, every sentence will cycle through the selected grammar structures. In “Roaming” mode these grammar structures will be interspersed throughout the listening.
- Select which voice they want to hear.
- Select the speed of the voice.
- Select whether they want the app to ask questions.
    - Select the frequency of the questions –after every sentence, after X sentences.
    - Select the type of questions they want the app to ask: comprehension questions, vocabulary questions (what is the meaning of X), “Construct this sentence in your target language” questions.
- Select which Anki account to integrate (optional)
    - Choosing vocabulary from Anki deck overrides the “level of vocabulary” selection.
    - Select whether they want card status in their Anki deck to be updated when:
        - (A) just by the listening stream going on for some time
        - (B) after answering “what is the meaning of this word” questions from the app
    - Select whether new cards should be created in their Anki deck for words that they frequently come across during the listening stream.

Listening stream configuration templates catering to users with different skill sets and levels in the language will be available.

2.2. Non-Functional Requirements

2.2.1. Usability

A quality application should be easy to use. Session tokens shall be cached on the user’s device to prevent launching the app from being tedious. Any audio to be streamed to the device shall download faster than its being played and all downloaded audio shall be fully cached for one playback session to facilitate scrubbing.

2.2.2. Reliability

Assuming continuous third-party service availability, the servers shall have at least 95% uptime and work without disturbance when not in maintenance. All data shall be constantly updated to ensure no data loss in case of a failure. The server data shall always be in a valid state. In case of malformed data, any requested operations using such data shall be denied. To avoid loss/corruption of data, any modifications to the same entry on server data shall be sequenced and not run in parallel. In case of data corruption or an invalid operation, the server shall be able to roll back the database to a valid state.

2.2.3. Performance

A smooth user experience is key for a quality application. The mobile client shall never show stutters, and any performance-heavy operation shall be completed in the background with a loading indicator. The server should also be Constantly/routinely operating modules shall not meaningfully impact the operating speed of other modules. Some user requests may require content to be generated using AI models. Any request that does not generate content shall respond to user requests under one second (excluding client-side network delays). Any content generating request shall respond to the user under one second after AI's work is done.

2.2.4. Maintainability

StreamLang's server's modular architecture design allows for easy maintenance. When modules in need of maintenance can function independently of each other, application functionality that is isolated from any affected modules may keep running unaffected, while any fixes/rollbacks are carried out on problematic modules. In the case of horizontally scalable modules, when a module instance has an outage/must be taken down, another instance may take its place to serve users until the instance is back up. Server operations shall be logged to make bug-fixing easier. Client maintenance is a non-issue since the entire application is already built and on the device of a user. Any updates shall be delivered through the device's application store.

2.2.5. Scalability

The client is a mobile application programmed in Flutter, so there will be no extra servers for the client application, meaning it can effectively be scaled infinitely. The backend will be separated into multiple modules to facilitate horizontal scaling. Any module that will be directly facing the user and is running strictly locally on our own servers shall have the ability to run in multiple instances and can be scaled infinitely. Modules using third-party services shall be scaled horizontally so long as the service provides such an option. With this architecture we should be able to adequately scale our application as user numbers change.

3. Feasibility Discussions

3.1. Market & Competitive Analysis

StreamLang has three main competitors; Taalhammer, Languia and Beelinguapp. The features of these competitors can be summarized as follows.

Taalhammer:

- Curated content library that aims to improve language fluency by comprehensible input
- Spaced repetition system for vocabulary and entire sentences optimizing review time

- Listening mode with configurable narrator speed
- Sentence building practice available but without a rigorous grammar focus
- Users can create and track their own content alongside the curated content [6]

Lenguia:

- Has the ability to create personalized stories or podcasts administered through audio, specified based on difficulty or interest level
- Focuses on comprehensible input rather than textbook-like drills
- Spaced Repetition System for vocabulary only (not grammar), that inserts the vocabulary in new stories / contexts as the time for reviews come
- AI tutor feature available for answering questions [7]

Beelinguapp:

- Reading and listening based content library segregated based on level and spanning various categories like novels, news or lyrics
- Displays translated and native text of the content being listened to, synchronized with the narrator
- Narrations are not by AI but by native speakers, an edge over StreamLang
- Assessment of level of comprehension by intermittent quizzes [8]

StreamLang has many differentiators with regards to its competitors. StreamLang's grammar-centric configuration capabilities is a feature that no competitor offers, especially considering the fine-grained choices that the user can make between Drill Mode (repeating sentences with the same grammar structures again and again for a more brute-force approach) and Roaming Mode (interspersing the grammar structure into the content for a more natural flow). Another differentiator of StreamLang is the Dynamic Adaptive mode. The listening stream configuration under Dynamic Mode in StreamLang updates in a way that requires minimal intervention from the user, increasing ease of use and shadowing the competitors. Streamlang's deep integration with Anki -a very popular flash card app with 2.1k reviews with 4.1 score on AppStore [9]- a feature that no competitor possesses. We also have an intelligent and dynamic question generation system to assess user-level, created based on the specific vocabulary and grammar structures that the user is studying at a given moment -further increasing StreamLang's edge.

3.2. Academic Analysis

The main aim of our project is language learning. Thus, it is imperative that we research and utilize different scientific approaches and their effects on language learning, and incorporate those strategies in our process. To address our core problem, we can look to the Monitor Model of Stephen Krashen for second language acquisition [10]. According to Krashen, there are two distinct processes for learning a language, acquisition and learning. Here, acquisition is the subconscious process of learning a language, mainly through exposure. Learning is the conscious effort of learning a language, such as studying vocabulary or grammatical structures.

It is implied that the acquisition method is more effective for learning than the conscious learning method. This can be seen in children, which naturally pick up the language spoken around them. Our project aims to enable users to naturally acquire language through listening experience. Although, there is an important point here to be aware of. We can see that through Krashen’s Input Hypothesis. It states that language learners learn effectively when the content they are exposed to is comprehensible, but slightly above their current language level. In our project, we aim to address it by gradually increasing the difficulty of the listening stream as the user learns the language, while testing them occasionally to ensure that they learned the previous topics.

Another approach we utilize for language learning is the Spaced Repetition System(SRS). This method allows learners, for any topic, to memorize things more effectively [11]. The basis of the idea comes from Ebbinghaus’ idea of a “forgetting curve” [12]. This idea theorizes that over time, there are time frames where new information should be repeated to be more effectively memorized. SRS utilizes this theory to improve learning. We can utilize this method in our project to ensure that the listeners learn the concepts correctly.

Traditional learning approaches focus on learning a specific grammar structure, or a vocabulary list. Instead, we offer a different approach, focusing on communication in a language. These two approaches are categorized by Michael H. Long as Focus on Forms(traditional learning approach) versus Focus on Form(communication based approach) [13]. It is theorized that with the Focus on Form approach, due to learners having limited attentional resources and that they prioritize semantic resources over linguistic structures, a more effective learning method can be constructed, given that the learners have some knowledge of the content, which ties into Krashen’s Input Hypothesis. In our project, we will prioritize the Focus on Form approach to effectively teach language.

4. References

[1] IEEE, “IEEE Code of Ethics,” [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8. (accessed Nov. 3, 2025).

[2] IBM, “OpenAPI 3.0,” ibm.com. https://www.ibm.com/docs/en/app-connect/13.0.x?topic=apis-openapi-30 (accessed Nov. 20, 2025).

[3] Dart, “Effective Dart,” dart.dev. https://dart.dev/effective-dart (accessed Nov. 20, 2025).

[4] “Google Python Style Guide,” google.github.io. https://google.github.io/styleguide/pyguide.html (accessed Nov. 20, 2025).

[5] D. W. Price et al., “The effect of spaced repetition on learning and knowledge transfer in a large cohort of practicing physicians,” Academic Medicine, vol. 100, no. 1, pp. 94–102, Sep. 2024. doi:10.1097/acm.0000000000005856

[6] “Home,” Taalhammer, https://www.taalhammer.com/ (accessed Nov. 7, 2025).

[7] Lenguia, “Learn languages with comprehensible input,” Lenguia, https://www.lenguia.com/ (accessed Nov. 7, 2025).

[8] “Language learning app: How does beelinguapp work?,” Beelinguapp, https://beelinguapp.com/language-learning-app (accessed Nov. 7, 2025).

[9] A. P. Ltd, “Ankimobile Flashcards App,” App Store, https://apps.apple.com/us/app/ankimobile-flashcards/id373493387#productRatings (accessed Nov. 7, 2025).

[10] S. D. Krashen, *The Input Hypothesis: Issues and Implications*. United States: Linguistic Society of America, 1988, pp. 171-173. Accessed: Nov. 20, 2025. [Online]. Available: https://www.jstor.org/stable/414800?origin=crossref.

[11] P. Smolen, Y. Zhang, and J. H. Byrne, “The right time to learn: mechanisms and optimization of spaced learning,” *Nat. Rev. Neurosci.*, vol. 17, pp. 77–88, Jan. 2016. Accessed Nov. 20, 2025. doi:https://doi.org/10.1038/nrn.2015.18. [Online]. Available: https://www.nature.com/articles/nrn.2015.18

[12] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*. United States: Annals of Neurosciences, 2013, pp. 155-156. Accessed: Nov. 20, 2025. [Online]. Available: https://annalsofneurosciences.org/journal/index.php/annal/article/view/540.

[13] S. Loewen, “Focus on Form Versus Focus on Forms,” in The TESOL Encyclopedia of English Language Teaching, John Wiley & Sons, Ltd, 2018, pp. 1–6. doi:https://doi.org/10.1002/9781118784235.eelt0062.