CENG 407

Innovative System Design and Development 1

2025-2026 Fall

Project:

Syntagma: AI Powered Spaced Repetition System Based Language Learning App

Team Members:

202211403 İlker Feza Çoban

202111027 İlayda Sümer

202111007 Elif Su Tufan

202211077 Alp Yılmaz

202211032 Sultan Gürbüz

202111417 Ömerhan Yiğit

Advisor: Abdul Kadir Görür

Table of Contents

Table of Figures

Abstract

Syntagma is a smart language learning app that uses artificial intelligence (AI) to help people learn foreign languages more effectively. The project's main goal is to turn regular media consumption, like watching videos or reading online, into an effective way to learn vocabulary and achieve fluency.

Our philosophy is based on the idea that people learn languages best through "comprehensible input"—meaning content they understand, but which contains just enough new information to push them forward. Therefore, Syntagma avoids traditional grammar drills and textbooks, focusing instead on fun, interesting media to keep users engaged and reduce learning stress.

To ensure users remember what they learn, the app uses a system called Spaced Repetition (SRS). This system schedules reviews for words at the perfect moment—right before the user would normally forget them—to maximize memory retention while minimizing study time. We use a modern, advanced algorithm called FSRS, which uses machine learning to analyze the user's memory patterns and predict the ideal time to review each flashcard.

The application has two main parts:

1. The Server (Backend): This robust system is built using reliable tools like Spring Boot and PostgreSQL (a high-performance database). The whole setup is packaged using Docker to ensure the app works consistently everywhere.

2. The User Interface and Content Tool: A Chrome Extension is used to extract subtitles and text from websites and video platforms. This extracted text is processed using Natural Language Processing (NLP) tools that automatically identify important words, filter out irrelevant content, and create high-quality, context-aware flashcards.

The mobile app is designed to be highly motivating. It includes gamification features like points, streaks, and badges to encourage users to practice daily and build learning habits. The system also uses personalization to adapt content and review schedules to the individual learner's progress. By combining media consumption with efficient, data-driven memory training, Syntagma delivers a personalized and engaging path toward language fluency.

1. Introduction

The Syntagma project introduces a new, smart application designed to make learning foreign languages easier and more effective. Many popular learning platforms, like Duolingo, focus on making lessons fun games, but they often do not present the deep, contextual study needed for fluency. Other powerful tools, such as the combination of Yomitan and Anki, are highly effective but are difficult and time-consuming for most learners to set up and maintain. Syntagma aims to solve this gap by bringing together the fun parts of learning with the efficiency of advanced study techniques. Our approach is built on the educational framework known as Stephen Krashen’s Monitor Model. This theory suggests that fluency comes from natural acquisition (like a child learning their first language) rather than just the conscious study of grammar rules. Our main goal is to reduce stress (the "affective filter") and allow users to learn from content they genuinely enjoy, such as videos or dramas. To make this learning system efficient, Syntagma focuses on providing comprehensible input (i + 1)—content that is mostly understood but contains just enough new information to drive progress. We achieve this through two main technical components. First, a Chrome Extension is used to extract subtitles and text directly from external media sources. This raw text is then processed using smart tools, known as Natural Language Processing (NLP), to automatically create high-quality, contextual flashcards. Second, to ensure that vocabulary is remembered long-term, the system uses an advanced Spaced Repetition System (SRS). This system employs the FSRS algorithm, which uses machine learning to accurately predict a user's memory state and schedule reviews at the exact moment needed to maximize retention while minimizing study time. Finally, the mobile application uses gamification elements, like streaks and points, along with personalization, to keep users motivated and help them build strong, consistent learning habits.

Section 1: Literature Research

1. Existing Works:

1.1 Existing Language Learning Applications

These are the tools that are widely used for foreign language acquisition.

Gamified Platforms (Duolingo [1], Memrise [2]):

Duolingo is a market leader in terms of user numbers, targeting casual beginners with short, game-like lessons, points, streaks, etc. Their approach is fundamentally different from Syntagma's, prioritizing engagement and habit formation through extrinsic motivators over deep, contextual learning.

Curriculum-Based Platforms (Babbel [3], Busuu [4], Rosetta Stone [5]):

These applications provide more structured, course-like experiences that are closer to traditional classroom learning.

Language Reactor:

Operating as a freemium browser extension or a website with integrated content from sites such as YouTube or Netflix. However, there are no way to upload your own content, or study those materials effectively.

LingQ:

LingQ is built around a reading-centric approach [6]. It boasts an extensive library of texts with synchronized audio and tracks vocabulary progress. While its methodology is very similar to Syntagma's, it's focus is more on text.

Yomitan + Anki:

For technically proficient learners, these combinations of free, open-source tools are free to use. Yomitan serves as a pop-up dictionary extension, and Anki is used for spaced repetition [7]. These tools can replicate much of Syntagma's core functionality however these tools are very hard to set up and can cost a lot of time for language learners.

1.2 Foreign Language Acquisition Theories

The input hypothesis and i + 1:

The philosophical foundation of Syntagma is a direct application of Stephen Krashen's monitor model, particularly his influential input Hypothesis. This theory has several claims that are included in the application's design.

The Acquisition-Learning Distinction:

Krashen distinguishes between “acquisition” which is a subconscious process similar how a child learns their first language and “learning”, the conscious study of grammar rules. He hypothesizes that fluency comes from acquisition, not learning. Syntagma’s rejection of traditional textbooks and grammar drills in favor of fun, interesting media consumption is an embodiment of this principle.

Comprehensible Input (i + 1) :

The core of this hypothesis is that “acquisition” occurs when learners are exposed to “comprehensible input” that is just one step beyond their current level of understanding, previously stated as (i + 1).

The input must be understood but also contain new elements to drive progress. Syntagma implements this concept directly by tracking the user’s vocabulary and making a “comprehension score” of each piece of media. The suite of dictionary, translation, AI explanation tools also serve to make difficult (i + 2, i + 3 etc) sentences into more easily digestible (i + 1) sentences.

The Affective Filter Hypothesis:

Krashen also proposed that emotional factors like anxiety, stress, and low motivation can create an “affective filter” that blocks comprehensible input from being processed.

Syntagma’s emphasis on allowing users to learn from content they genuinely enjoy such as YouTube videos, dramas, anime is a practical strategy for lowering this filter and making the learning process engaging.

1.3 Spaced Repetition System (SRS)

Spaced repetition systems (SRS) schedule reviews at expanding intervals to exploit the spacing effect and the forgetting curve: easier items are scheduled less often, harder items more often, so a learner maintains a target recall level while minimising reviews.

1.3.1 SM-2 (SuperMemo-2)

SM-2 is a lightweight algorithm originally used in SuperMemo that updates a card’s easiness factor and interval based on the user’s grade for each review.

Key variables (per card):

EF = easiness factor (initial 2.5 by default)

- I[n] = interval after nth successful repetition (in days)
- n = repetition count (consecutive successful reviews)
- q = quality response from user, typically 0–5 (in Anki it's mapped to 4 choices)

Core SM-2 rules (text):

1. If q < 3 (failure), set n = 0, schedule the card for immediate/short repeat (e.g., 0 or 1 day).
2. If q >= 3:
    a. If n = 0 → I[1] = 1
    b. If n = 1 → I[2] = 6
    c. If n >= 2 → I[n+1] = I[n] * EF
    d. Update EF: EF := EF + (0.1 - (5 - q) * (0.08 + (5 - q) * 0.02))
    e. If EF < 1.3 then EF := 1.3 (lower bound)

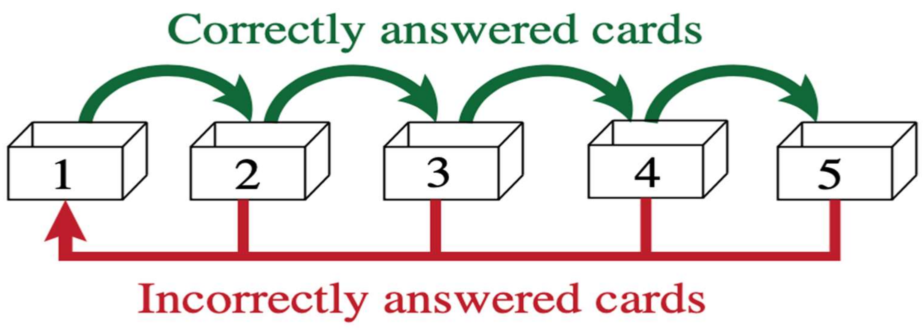

Figure 1 SM-2

1.3.2 FSRS

FSRS aims to learn your memory patterns and schedule reviews more efficiently than SM-2.

FSRS is based on the Three Component Model of Memory [8]. The model asserts that three variables are sufficient to describe the status of a unitary memory in a human brain. These three variables include:

- Retrievability (R): The probability that the person can successfully recall a particular information at a given moment. It depends on the time elapsed since the last review and the memory stability (S).
- Stability (S): The time, in days, required for R to decrease from 100% to 90%. For example, S = 365 means that an entire year will pass before the probability of recalling a particular card drops to 90%.

- Difficulty (D): The inherent complexity of a particular information. It represents how difficult it is to increase memory stability after a review.

In FSRS, these three values are collectively called the "memory state". The value of R changes daily, while D and S change only after a card has been reviewed. Each card has its own DSR values, in other words, each card has its own memory state. To accurately estimate the DSR values, FSRS analyzes the user's review history and uses machine learning to find parameters that provide the best fit to the review history.

2.System Architecture Components:

2.1 Backend System:

2.1.1 Gradle

Gradle is a build automation tool used to compile, test, and package Java projects (like Spring Boot applications) [9].
It manages dependencies — the external libraries your project needs — and automates repetitive tasks such as building .jar files, running tests, or deploying the app.

Gradle uses Groovy or Kotlin scripts instead of XML (like Maven), which makes it faster and more flexible.
It also supports incremental builds, meaning it only rebuilds the parts of the project that changed, saving time.

In this project, Gradle is used for:

- Manage dependencies (Spring Boot, Hibernate, Lombok, etc.)
- Build and run the backend application

2.1.2 Hibernate + Spring Data JPA

Hibernate is an Object-Relational Mapping (ORM) tool that connects Java objects to relational database tables [10].
It automatically generates SQL queries from your Java code, so you don't need to write raw SQL for CRUD operations.

Spring Data JPA builds on Hibernate to simplify database operations even further.
By defining repository interfaces, developers can perform complex queries and manage entities without writing SQL or boilerplate code.

In this project, Hibernate is used for:

- Eliminates manual SQL query writing

- Reduces boilerplate code
- Manages entity relationships and transactions automatically

2.1.3 Lombok

Lombok is a Java library that reduces boilerplate code [11]by automatically generating commonly used methods such as getters, setters, constructors, toString(), and equals/hashCode().

It integrates with IDEs and the build system, allowing to write cleaner and shorter code while maintaining full functionality [12].

In this project, Lombok is used for:

- Eliminates repetitive code for entity classes and other Java objects
- Simplifies maintenance and improves readability
- Speeds up development by reducing manual coding of boilerplate methods [11]

2.1.4 Mockito

Mockito is a popular Java library used for unit testing by creating mock objects [13]. Mocking allows to simulate the behavior of external dependencies such as databases, APIs, or services, so that tests can focus only on the component being tested [14].

In this project, Mockito is used for:

- Eliminates the need to access real databases or external services during testing
- Makes unit tests faster, more reliable, and isolated
- Allows developers to control and verify interactions with dependencies

2.1.5 Spring Boot:

Spring Boot is a sub-framework derived from the Spring ecosystem. In the modern trend of separating the front-end and back-end layers, Spring Boot allows developers to build server-side applications quickly and efficiently. Its auto-configuration mechanism significantly reduces the need for manual setup and enhances development productivity [15]

Traditional Spring projects often required large and complex configuration files, but Spring Boot eliminates this problem by introducing an automated and lightweight configuration process. Through Spring Boot starters, the framework automatically manages the necessary dependencies for the application. With the help of build automation tools such as Maven or Gradle, developers can easily add essential modules, including web and data access components. This approach enables them to focus on implementing business functionality rather than spending time on extensive configuration tasks.

Spring Boot's microservices architecture enhances the scalability and maintainability of these applications, allowing for the seamless addition of microservices to meet evolving service requirements.

Spring Boot's role in creating a Service-Oriented Architecture (SOA)-based REST service API at the Atmospheric Radiation Measurement (ARM) Data Center showcases its utility in bridging the divide between frontend user interfaces and backend databases [16]

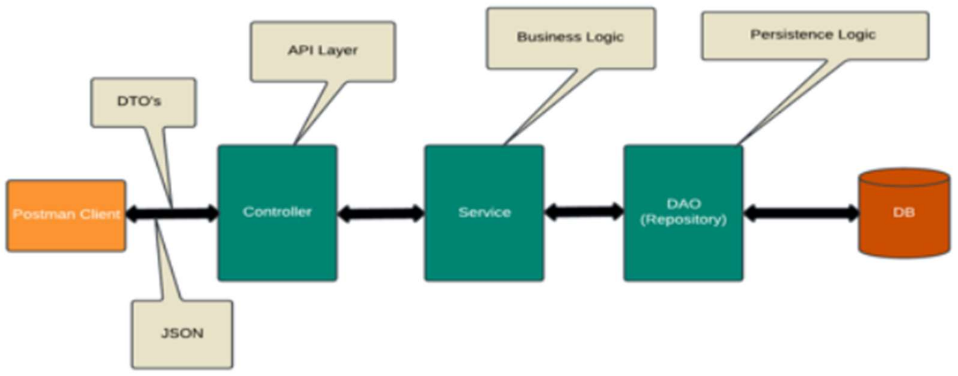

Figure 2 Spring Boot working diagram

2.1.6 SLF4J:

The Simple Logging Facade [17] for Java, serves as a simple facade or abstraction for various logging frameworks. SLF4J allows the end-user to plug in the desired logging framework at deployment time [18].

In this project, SLF4J is used for:

- SLF4J is facade and we can write code once but switch to another logging framework without changing any line of code.
- Reduces coupling [19] between application and logging framework

2.1.7 Logback:

Logback is a logging framework that works perfectly with Spring-Boot applications [20].

In this project, Logback is used for:

- Built-in support for SLF4J

2.1.8 PostgreSQL:

Several studies have compared relational databases in the context of e-learning and high-performance data management applications, often highlighting the advantages of PostgreSQL over conventional systems like MySQL.

Jeyaraj et al. (2022) proposed a system designed to promote the adoption of PostgreSQL by demonstrating its strengths and advantages compared to other relational databases, particularly MySQL [21]. The authors highlight that PostgreSQL, as an object-relational database management system (ORDBMS), offers advanced SQL compliance and superior performance capabilities. However, they also noted that its complex textual interface and lack of beginner-friendly tools limit its widespread use. To address this, their study developed a visual and interactive learning platform that helps users understand PostgreSQL's structure, query manipulation, and relational modeling. The research emphasizes PostgreSQL's potential as the most advanced open-source RDBMS and its value for educational and industrial database systems.

Further substantiating this position, Salunke and Ouda (2024) conducted a comprehensive benchmarking study comparing the performance of PostgreSQL and MySQL under real-world, high-load scenarios. Their research introduced a Python-based benchmarking framework to evaluate query execution time, latency, and concurrent processing capabilities. Experimental results demonstrated that PostgreSQL achieved significantly lower latency than MySQL—up to faster in select queries and faster in conditional selects—while maintaining stable performance during simultaneous operations [22]. These findings confirm PostgreSQL's suitability for applications requiring low data latency and high concurrency, making it an optimal choice for large-scale backend systems such as learning platforms or continuous authentication environments. This collective body of research indicates a growing trend toward adopting PostgreSQL for scalable and educational database systems due to its superior architectural and performance advantages.

2.1.9 DBeaver:

DBeaver is a universal database management tool. DBeaver allows for easy data editing in a spreadsheet-like interface [23].

DBeaver was chosen for the Spring Boot-based development process to visualize and manage the database structure more efficiently. This tool significantly simplifies the processes of reviewing, editing, and managing database designs within the project.

2.1.10 Docker:

Mhatre (2023) examined the role of Docker and Kubernetes within microservices-based software architectures, emphasizing their contributions to scalability, flexibility, and reliability in modern development. [24] Docker ensures consistent runtime environments across development, testing, and production stages, eliminating deployment discrepancies, while Kubernetes provides automated orchestration, scaling, and load balancing for containerized applications.

In this project, however, Docker was preferred over Kubernetes because, although Kubernetes offers powerful orchestration capabilities for large-scale distributed systems, it introduces higher complexity, greater resource consumption, and longer setup time for small to medium-sized projects. Docker provides a simpler configuration, faster deployment, and sufficient scalability for the project's backend and database services. Moreover, using Docker Compose, multiple services (e.g., Spring Boot backend and PostgreSQL database) can be easily linked within the same container network.

Thus, Docker was selected as the primary containerization technology to simplify deployment, ensure portability, and improve development efficiency.

2.1.11 Swagger:

Lama (2025) emphasizes in his study that API documentation is a critical component in software projects that directly impacts interteam communication and development efficiency.

He stated that traditional, static documentation methods lead to problems such as outdatedness, inconsistency, and lack of interaction, which in turn prolongs development time and leads to integration errors.

The study recommends the automated, standardized, and interactive documentation structure provided by Swagger to address these issues.

Swagger integrates with Java-based frameworks such as Spring Boot, automatically keeping API endpoints, request/response models, and data schemas up-to-date.

This approach both streamlines the flow of information between developers and enables new team members to quickly adapt to the system.

The study's results demonstrate that Swagger provides transparency, accuracy, and speed in API management and forms the foundation of interactive API ecosystems in modern software development [25].

2.1.12 Postman:

Kore et al. (2022) presented an overview of Postman, a leading platform for API testing and automation widely adopted by developers and enterprises [26]

The study outlines how traditional manual API testing methods fail to scale in modern software ecosystems where applications interact with hundreds of APIs simultaneously.

Postman simplifies this process by providing a graphical interface for designing, executing, and validating RESTful API requests (GET, POST, PUT, DELETE) while offering integrated support for headers, parameters, and authentication.

Additionally, the paper discusses Newman, Postman's CI/CD add-on, which enables automated and continuous testing as part of DevOps pipelines.

Their implementation in a banking and financial services project demonstrated how Postman enhances collaboration, efficiency, and test repeatability, making it a valuable tool for verifying API reliability and security.

2.2 Chrome Extension with NLP-Based Flashcard Generation

Since most of our media consumption is done on browsers, it makes sense to embed our learning tools directly into the user's web experience. The Chrome extension acts as the gateway between web content and Syntagma's personalized language learning system.

2.2.1 Subtitle and Text Extraction

The extension will interface with video platforms and general websites to extract readable text content.

Technologies can be used:

- yt-dlp: A command-line tool to download videos and subtitles from platforms like YouTube [27].
- netflix-to-srt: A GitHub tool that allows subtitle extraction from Netflix videos [28].
- Readability.js: Mozilla's tool for cleaning articles by stripping cluttered HTML from web pages [29]

2.2.2 Dictionary Lookup and Interaction

Users can click or hover on words in subtitles or website text. The extension displays a pop-up with dictionary definitions, translations, and example usage.

2.2.3 NLP-Driven Flashcard Creation

This stage transforms selected or parsed sentences into high-quality flashcards using multiple NLP components:

a. Sentence Segmentation and Tokenization

- wink-nlp: A fast JavaScript NLP library supporting sentence segmentation and word tokenization [30].
- compromise.js: Provides tokenization, lemmatization, and POS tagging for browser environments [31].

b. Lemmatization and POS Tagging

These help reduce words to their root forms and identify their grammatical roles.

c. Frequency and Difficulty Filtering

SUBTLEX word lists will be used to filter low-frequency or less relevant vocabulary [32].

e. Named Entity Recognition (NER)

spaCy [33] (for Python) and compromise.js (for JavaScript) to filter out named entities that don't require flashcard creation.

f. Audio Extraction and TTS Generation

Native audio can be extracted using yt-dlp.
If unavailable, synthetic audio can be generated using Google TTS [34].

2.3 Frontend and Design

2.3.1. User Interface (UI) Design and User Experience (UX)

The front-end design of mobile language learning applications is one of the most critical components directly affecting the user experience. The User Interface (UI) in these applications is not merely a visual element but a key factor that determines the effectiveness of the learning process [35].

Research on leading applications such as Duolingo, Memrise, Babbel, Busuu, and LingQ shows that all of them commonly adopt a philosophy of simple, user-friendly, and engagement-oriented design. The interfaces of these applications are mostly button-based, aiming to allow users to act quickly and intuitively during the learning process. Menu structures are intentionally kept minimal and guiding, thereby avoiding unnecessary distractions.

Each application establishes a unique identity through its distinctive color palette and iconography. This consistent visual theme strengthens user attachment to the application and reinforces visual coherence.

UI/UX design principles focus on avoiding complex transitions and information overload. Instead, designers prioritize visual elements that support learning:

- Clear Icons: Symbols whose meaning is immediately comprehensible.
- Consistent Typography: Font styles that enhance readability and reflect brand identity.
- Intuitive Navigation: Components that enable users to easily reach their desired location.

Furthermore, instant feedback mechanisms play a vital role in the user experience. Users immediately see whether their answers are correct or incorrect, reinforcing learning through immediate response and repetition. This fast feedback loop creates an interactive and supportive learning environment [35].

2.3.2 Gamification and Motivation

Gamification is one of the most widely used strategies to enhance user motivation in mobile language learning applications. This approach integrates game-like elements into the learning process.

Key Gamification Elements:

- Rewards and Points (XP): Ensures users see a tangible return for their efforts.
- Badges and Achievements: Creates a sense of accomplishment upon reaching specific milestones.
- Levels and Streaks: Establishes continuous progression and a habit of regular practice.

These elements increase both the frequency of engagement and the retention of acquired knowledge [36]. Through gamification, users tend to open the app more frequently, set specific goals, and feel a sense of accomplishment upon completing them. These mechanisms provide immediate motivation in the short term, while aiding in the construction of sustainable learning habits over the long term.

Applications like Duolingo [1], Memrise [2], and Busuu [4] extensively apply these elements. For example, in Duolingo, users who practice daily without skipping sessions earn "streak" points or receive badges for completing milestones. These systems not only make learning enjoyable but also foster a positive emotional connection between the user and the app.

Significantly, gamification encourages voluntary engagement rather than forced participation. Studies have shown that voluntary users interact with language learning applications more frequently and for longer durations compared to mandatory users. This highlights that the inclusion of motivation-enhancing features is critical for maintaining long-term user engagement and consistent application use [36].

2.3.3 Personalization and Adaptive Learning

Personalization is another defining feature of modern mobile language learning applications. These systems analyze user data—such as learning habits, progress pace, and areas of strength or weakness—to create individualized recommendations, reminders, and review schedules [37].

This adaptive design allows each learner to control their own learning journey. For instance, LingQ enables learners to progress at their own pace while providing relevant vocabulary and content recommendations tailored to their interests and current proficiency level. This makes the learning experience more relevant and efficient [37].

Push Notifications also play a critical role in sustaining user engagement [38]. Regularly sent reminders encourage users to return to the app, helping maintain consistency in their learning routine. These features not only enhance user commitment but also help prevent interruptions in the learning process [37].

2.3.4. Accessibility and Mobile Learning Advantages

Mobile language learning apps make education flexible, personalized, and available anywhere [37]. Unlike traditional classroom settings, Mobile-Assisted Learning allows users to study anytime using their smartphones or tablets. This is ideal, especially for individuals with limited time or restricted access to formal education [37].

A key benefit of mobile accessibility is ubiquitous access. Users can find learning opportunities even during short daily breaks, such as commuting. Apps like Duolingo and Memrise offer short "micro-lessons" that fit easily into busy routines. Furthermore, many applications support an offline mode, increasing inclusivity for users in areas with poor internet connectivity [37].

Finally, multimodal learning and push notifications sustain engagement. Combining text, audio, and visuals strengthens vocabulary and pronunciation [39], while reminders and progress tracking encourage consistent study habits [37]. Overall, mobile accessibility enables a flexible, inclusive, and continuous approach to language learning.

Comparison Table of Major Language Learning Apps:

Application	UI/UX Design	Gamification Level	Personalization & Adaptive Learning	Notification Usage	Vocabulary / Word Management	Additional Notes
Duolingo	Colorful, minimal, icon-based; fast tap-interaction	Very high: streaks, XP, badges, levels	Moderate: adjusts lesson difficulty based on progress	Frequent daily reminders	Basic vocabulary tracking; limited customization	Highly gamified; widely used; simple micro-lessons
Memrise	Visual-focused; simple layout; includes user-generated video clips	Medium: points, levels	Low–medium: adjusts pace slightly	Frequent motivational reminders	Strong flashcard system; users can create their own decks	Uses real native-speaker videos; community-driven content
Babbel	Clean, serious, professional UI; lesson-focused	Low: minimal gamification	Strong: structured lesson paths based on level	Moderate notifications	Strong review system; spaced repetition	More academically structured; requires subscription
Busuu	Modern design; strong community integration	Medium: achievements and goals	High: personalized study plan + AI reviews	Regular reminders	Personal vocabulary list; community corrections	Social learning: natives correct your writing/speaking
LingQ	Text-heavy, simple UI; color-coded words	Very low	Very high: personalized content, smart suggestions, vocabulary-based adaptation	Moderate	One of the most advanced vocab systems (highlighting, stats, word tracking)	Best for reading & listening immersion; highly customizable

2.3.5. Push Notification Frequency and User Behavior

Push notifications play a central role in sustaining user engagement in mobile learning environments by prompting users to return to the application and maintain consistent study habits. However, notification frequency is a decisive factor that can significantly influence user behavior, perceived app intrusiveness, and long-term retention. The empirical findings of Wohllebe et al. (2021) provide one of the most comprehensive experimental analyses on this topic, making it directly relevant for mobile-based language learning applications [40].

In their large-scale experiment involving 17,500 real app users, five different notification frequency groups were observed over a seven-week period. The results demonstrated that higher notification frequency leads to a substantial increase in uninstall rates. Specifically, users receiving two notifications per week exhibited noticeably higher uninstall percentages compared to groups receiving weekly or bi-weekly notifications [40]. This indicates that excessive notifications can be perceived as intrusive, ultimately harming user retention.

Regarding user interaction patterns, the study reported that direct open rates (opening the app by tapping the notification) decrease significantly as notification frequency increases. Users become less responsive to notifications when they feel overwhelmed by their frequency. Conversely, indirect open rates (opening the app within 12 hours without tapping the notification) did not show a statistically significant decline. This suggests that while users may ignore overly frequent notifications, they do not entirely lose interest in the app content.

Parallel findings in the broader literature also support this relationship between excessive notifications and reduced user receptiveness. Prior research shows that users typically tolerate up to three notifications per day before negative reactions occur, especially in learning or health-related applications — beyond that threshold, interaction rates begin to drop and perceived annoyance rises [40].

Overall, push notifications remain an essential mechanism for activating and re-engaging users; however, the evidence shows that overly frequent, non-personalised notifications can lead to a rapid decline in direct engagement and increase app abandonment. For language learning applications, notification strategies must therefore be:

- optimally timed,
- moderate in frequency,
- context-aware, and
- personalized to user behavior and progress.

Such a balanced approach helps reinforce learning habits while avoiding the user fatigue and churn associated with excessive notification frequency.

2.3.6. React Native in Mobile App

React is an open-source JavaScript library developed by Facebook, focused exclusively on building user interfaces (UIs) and forming the foundation of modern, fast, and scalable web applications [41]. The core strength of React lies in its philosophy of breaking down complex interfaces into small, independent, and reusable "components," a modular architecture that significantly speeds up development and prevents code duplication. Its most critical advantage is the Virtual DOM (Document Object Model); instead of directly manipulating the real browser structure, React efficiently updates only the changed parts of the UI, ensuring superior performance and a highly responsive user experience, which is essential for data-intensive and dynamic projects. Ultimately, adopting React for your project provides robust scalability, easier debugging due to its declarative and one-way data flow, a massive supportive community, and future-proof potential, including cross-platform development via React Native, all of which contribute to reduced technical debt and long-term project success.

System Architecture and Technology Stack

The project relies on a robust backend and specialized front-end components:

Component Category	Key Technologies	Function and Rationale
Backend Framework	**Spring Boot**	Allows quick, efficient construction of server-side a, creating a **Service-Oriented Architecture (SOA)-based REST service API**.
Data Persistence	**PostgreSQL**	Selected for its advanced SQL compliance and **superior performance capabilities** in high-concurrency environments, demonstrating significantly lower latency than alternatives like MySQL.
Data Management	**Hibernate + Spring Data JPA**	Acts as an **Object-Relational Mapping (ORM)** tool, automatically generating SQL queries and eliminating the need to write raw SQL for CRUD operations.
Development Tools	**Gradle, Lombok, Mockito, Swagger, Postman**	**Gradle** manages dependencies and automates tasks; **Lombok** reduces boilerplate code for better readability; **Mockito** is used for faster, isolated unit testing by simulating dependencies; **Swagger** standardizes and automates interactive API documentation; and **Postman** is used for designing, executing, and validating RESTful API requests.
Deployment	**Docker**	Used for containerization to ensure a **consistent runtime environment** and simplify deployment, offering sufficient scalability for small to medium-sized projects compared to Kubernetes.
Content Generation	**Chrome Extension & NLP**	The extension extracts text/subtitles (using tools like yt-dlp and Readability.js). **Natural Language Processing (NLP)** components, including **tokenization**, **lemmatization**, **POS tagging**, and filtering using **SUBTLEX word lists**, are then applied to create high-quality, context-aware flashcards.
Mobile Application	**React Native, Gamification**	**React Native** is used to build the mobile application. The UI/UX is designed to be simple and engagement-oriented. **Gamification** elements (streaks, badges, points) are included to enhance motivation and build continuous study habits.
User Engagement	**Personalization & Notifications**	Systems analyze user data to create individualized recommendations. However, the research stresses that **high Push Notification Frequency leads to increased uninstall rates and decreased direct open rates**, requiring notifications to be moderate, personalized, and optimally timed.

Section 2: Product Vision & Scope: Syntagma

1. Product Vision

Our product vision is to build an effective and time-efficient language acquisition platform for language learners. We do this by combining passive media consumption with active vocabulary recall sessions.

While watching content or browsing the web, our extension will parse the content and give the user a comprehension score. The user can then judge the difficulty of the content and decide whether or not the engage with the content.

Easy flashcard creation based on the content the user consumes allows for later active vocabulary recall sessions.

2. Scope Boundaries

2.1 In-Scope Features

Target Language: English only

Native Language: Turkish only

Platforms: Chrome Desktop, iOS & Android (Mobile App)

AI Integration: translation, example sentence generation.

Core Workflow:

1. Parse: An NLP engine parses the website or video subtitles.
2. Providing a percentage understanding rate.
3. Capture: User can click to generate a flashcard with audio + sentence + screenshot if available.
4. Review: User can then use the mobile app to review flashcards with FSRS algorithm.

2.2 Out of scope:

1. Advanced Economy & Marketplace (Complex Gamification) While simple motivators like Streaks and XP are included, complex "virtual economies" are out of scope.

- **Excluded:** Virtual currencies (coins/gems), avatar customization, purchasing items, or an in-app store.

2. Social & Community Features The application is designed as a solitary learning tool.

- **Excluded:** Adding friends, user-to-user messaging, global leaderboards, or sharing decks with other users.

3. Real-time AI Tutoring (Chatbot) The AI integration is strictly limited to backend content processing (parsing/translating).

- **Excluded:** Conversational AI interfaces (chatting with a bot), voice-based roleplay scenarios, or grammar correction chatbots.

4. Mobile Content Creation The mobile application is a "Companion App" focused solely on the review process.

- **Excluded:** Parsing websites or creating new flashcards directly within the mobile app. All content generation happens via the Chrome Extension.

5. Multi-Language Support (MVP) The system architecture supports Internationalization, but the initial release is locked to a single pair.

- **Excluded:** Support for source languages other than English or native languages other than Turkish in the initial release.

3. Feature Breakdown

3.1 Chrome Extension

1. Content Parser

   Injects itself as an interactive overlay into the DOM of the browser.

2. Message Passing

   Secure passing of “flashcard” information to the backend.

3. Intuitive and non-obtrusive UI

   A pop-up interface when words are hovered or clicked, displaying the dictionary definition and “Create a flashcard” button.

3.2 NLP Engine

3.2.1 Text Processing

The module is responsible for ingesting raw text from the content parser.

- Tokenization

  Splits paragraphs and sentences into tokens based on the target language rules.

- Lemmatization

  Reduces words to their roots to accurately match the user’s known vocabulary database.

- Stop-word filtering

  Identifies and separates common grammatical structures (like “the”,” at”,” is”) to focus difficulty scoring on substantive vocabulary.

3.2.2 Language Configuration Layer

Abstracts language-specific logic to allow for future scalability (e.g., switching the Target Language from English to Spanish).

- Locale Handler: Dynamically loads the correct tokenizer and lemmatizer models based on the selected target language.

3.3 Backend

3.3.1. APIs

- Authentication & Users
  Handles account creation, secure login, and user identity verification. It uses JWT. Flashcards, collections, SRS review history, and preferences are always protected and managed by the correct authenticated user.

- Difficulty scoring
  Analyzes the user's collection of cards and returns a percentage score representing how many unique words in content the user knows.

- Cards
  Manages CRUD operations related to flashcards.

- Collections of flashcards
  Manages all operations related to a group of flashcards.

- Review
  User reviews flashcard to perform the Spaced Repetition System.

- Statistics of usage
  Users can track their stats.
    - Accuracy
    - Progress
    - Streaks
- Synchronization to enable offline usage.

- Storage for media data manages media data to store.
    - Screenshots
    - Audio files

- AI API (Translation & Sample Sentence Generation)
  It connects the desired AI to translate the word and get an example sentence with it.

- Health Check
  To synchronize the necessary Space Repetition data from offline mode. It does not create a new flashcard because offline mode is only available for mobile applications, and the mobile application does not create flashcards.

3.3.2. Database

Relational SQL database that stores users, flashcards, reviews, synchronization changes, and media references. The database stores:

- User accounts and authentication information.
- Statistics for each user.
  - Streak
  - Last login
- Groups of flashcards.
- Information about flashcards.
- The information that is important for the Spaced Repetition System
  - Is the word in flashcard remembered or not
  - When was the flashcard last time reviewed
  - Repetition number
- Data that needs to be synchronized between platforms.
- References for media data for the cards.

3.3.3. Spaced Repetition System

- SRS Scheduling Algorithm (FSRS or SM-2)
- Difficulty scoring
- Date for the next review calculation

3.3.4. Logging

We have a logging feature to track application behavior, debug issues, and monitor errors.

Technologies used:

- SLF4J
- Logback

3.4 Mobile App

3.4.1. SRS Review Experience

It is the heart of the application, built upon the FSRS algorithm.

- Interface: A simple, button-based, and fast touch interaction structure is used, similar to applications like Duolingo.

- Content: The word, sound, screenshot of the scene if available, and an example sentence are presented on a single screen.

- Operation: The user flips the card and selects how well they know it. This feedback allows the algorithm (FSRS) to calculate the next review time.

3.4.2. Flashcard Viewer

The card detail screen provides the user with all information about the word in an integrated manner.

- Context: It displays not just the meaning of the word, but also its usage within an example sentence and its translation on the same screen.

3.4.3. Dashboard

It is the main screen that motivates the user.

- Operation: It shows the total number of words reviewed daily and allows for the review of learned words.
- Design: Clear icons and easy-to-read fonts are used to increase readability.

3.4.4. Meaning / Dictionary Integration

It is the feature that allows looking up words without leaving the application.

- When the user clicks on an unknown word in a subtitle or text, the meaning and usage examples open in a "pop-up" window.

3.4.5. Gamification

Game-like elements are used to increase motivation.

- Streak System: A "don't break the chain" mechanism to ensure the user logs in every day.
- Badges: Achievement-based rewards that are unlocked when users reach specific proficiency levels. Each badge represents a clearly defined level (e.g., A2, B1), and the system visually indicates the user's current progress within that level, such as 60% completion toward B1. This helps users understand both their current level and how close they are to the next one.

3.4.6 Push Notifications

Reminders that call the user back to the application.

- **Strategy:** Notifications are sent at the right time and are personalized for the user.

- **Frequency Setting:** Since sending notifications too frequently causes users to uninstall the app, they are sent at a non-intrusive frequency (less than 3 per day).

3.4.7. Offline Review Support

Allows studying even without the internet.

- Data is saved to the phone locally to allow studying in places like subways, planes, or where there is no internet reception.
- Progress is automatically saved when the internet becomes available.

4. User Stories:

Teacher:

As a teacher teaching English, I want to keep track of my students known words, so that I can assess their comprehension levels and assign reading material accordingly.

Self-taught learner:
As a self-taught learner, I want to have instant translations available for words I do not know, so that I can understand the plot and learn passively.

Commuter Learner:
As a learner that commutes for hours, I want to open my phone and review my flashcards, so that I can study during my downtime.

Busy Professional:
As a busy user, I want quick daily notifications that remind me to review so I don’t break my learning streak.
Power User:
As a dedicated learner, I want my data to sync across devices so that I can study seamlessly on Chrome and a mobile application.

Section 3: System Architecture & Design Decisions

1. Functional Requirements

Chrome Extension

FR1.1 Login Interface: The extension shall provide a login form accepting email/password.

FR 1.2 Session Storage: The extension shall store JWT (JSON Web Token) in chrome.storage.local to maintain user sessions across browser restarts.

FR 1.3 Known Word Sync: The extension shall cache a local bloom filter or hash map of the user's "Known Words" to enable syntax highlighting.
FR 1.4 Known Words Browser: The user shall be able to view a list of all words and their statuses, known, ignored, learning words

FR 1.5 Subtitle Ingestion: The extension shall detect active caption tracks provided by the streaming service and ingest them as text.
FR 1.6 Dual Subtitles: The extension shall support rendering two subtitle tracks at the same time, the target language and native language.
FR 1.7 Subtitle Hiding/Blurring: The user shall be able to toggle a blur effect on subtitles, only revealing them on mouse hover or pause.
FR 1.8 Auto-Pause: The video player shall automatically pause at the end of every sentence, if the "Auto-Pause" setting is on.
FR 1.9 Sentence Navigation: The user shall be able to press hotkeys to seek exactly to the start or end timestamp of the current, previous, next subtitle line.

FR 1.10 Page Analysis: On page load, the extension shall traverse the DOM of the active tab, identify text nodes, and send the text to the parser engine.
FR 1.10 Syntax Highlighting: The system shall highlight words with specific CSS classes based on the user's knowledge status.
FR 1.11 Hover Logic: Hovering over any parsed word shall trigger a popup menu without requiring a click.

FR 1.12 Card Creation: The user shall be able to create a flashcard directly from the Video Overlay or Dictionary Popup with a single button press.

FR 1.13 Frequency Lists: The user shall be able to select a frequency list to determine how important the word is to learn.
FR 1.14 Ignore List: The user shall be able to ignore certain words so that it doesn't show up as unknown word.

NLP

FR 2.1 Raw Text Ingestion: The NLP module shall ingest raw text received from external media sources, such as websites or video subtitles.

FR 2.2 Segmentation and Tokenization The system must split paragraphs and sentences into fundamental units, performing sentence segmentation and word tokenization based on the rules of target language.

FR 2.3 Lemmatization and POS Tagging: The NLP module shall reduce words to their root forms (lemmatization) accurately match the users known vocabulary database and identify their grammatical roles.

FR 2.4 Vocabulary Filtering: The system must filter out irrelevant or non-substantive content by identifying and separating common grammatical structures to focus difficulty scoring on substantive vocabulary by using Named Entity Recognition to exclude name entities from flashcard creation.

FR 2.5 Frequency and Difficulty Filtering The system shall utilize SUBTLEX word lists to filter out low-frequency or less relevant vocabulary before a comprehension score is generated.

FR 2.6 Audio Generation: The system must be able to extract native audio or, if native audio is unavailable, generate synthetic audio using services for the flashcards.

FR 2.7 Language Configuration Layer Management: The NLP Engine shall maintain a Language Configuration Layer that abstracts language-specific logic, such as dynamically loading the correct tokenizer and lemmatizer models, to allow for future scalability of target languages.

FR 2.8 Flashcard Data Preparation: The NLP process shall prepare the necessary structured data for flashcard creation, including the word's lemma, the source sentence, and references for audio or screenshots, transforming the processed input into high-quality, context-aware flashcards.

Backend

FR-3.1: User Authentication & Authorization

- FR-3.1.1: The system must allow users to create accounts.
- FR-3.1.2: The system must allow users to securely log in with their email address and password.

- FR-3.1.3: The system should generate a JWT (JSON Web Token) for the user after a successful login.

- FR-3.1.4: The system must perform JWT validation on all protected API requests.

- FR-3.1.5: The system should process each request only under the name of the corresponding authenticated user.

FR-3.2: User Profile & Preferences Management

- FR-3.2.1: The system must store the user's basic profile information (user ID, email, preferences).

- FR-3.2.2: The system should be able to save the user's learning preferences (notification settings, target daily word count, etc.).
- FR-3.2.3: The system needs to update the user's last login time and active streak information.

FR-3.3: Flashcard Management

- FR-3.3.1: The system must support creating new flashcards from data received via the Chrome Extension.

- FR-3.3.2: Each flashcard should contain the following information:
    - Word (lemma)
    - Source of word's sentence
    - Example sentence (OPTIONAL)
    - Translation
    - Screenshot if available
    - Sentence audio if available

- FR-3.3.3: The system must be able to perform CRUD operations (create, read, update, delete) on the user's flashcards.

- FR-3.3.4: The system must ensure that flashcards are accessible only by the authorized user.

FR-3.4: Flashcard Collections

- FR-3.4.1: The system should allow users to group flashcards under collections.

- FR-3.4.2: The system must support creating, deleting, and updating collections.
- FR-3.4.3: The system must support adding or removing a flashcard from a collection.

FR-3.5: AI Integration (Translation & Example Generation)

- FR-3.5.1: The system should be able to generate AI-based translations for the selected word.
- FR-3.5.2: The system should generate an example sentence containing the word.
- FR-3.5.3: AI services should only be called from the backend (the frontend cannot access them directly).

FR-3.6: Spaced Repetition System (SRS)

- FR-3.6.1: The system must store the SRS parameters (stability, difficulty, retrieval) for each flashcard.

- FR-3.6.2: The system should run the FSRS algorithm based on the user's review result. Only yes or no.

- FR-3.6.3: The system must calculate the next review date for the flashcard after each review.

- FR-3.6.4: The system should be able to generate a daily review list for the user.

FR-3.7: Review History Tracking

- FR-3.7.1: The system must record each review process along with the date, result, and the option used.

- FR-3.7.2: The system should be able to use historical review data to generate statistics.

FR-3.8: Statistics & Progress Tracking

- FR-3.8.1: The system should calculate the following user statistics:
    - Daily review count
    - Accuracy rate
    - Number of words learned
    - Active streak duration

- FR-3.8.2: The system should provide these statistics via API.

FR-3.9: Synchronization & Offline Support

- FR-3.9.1: The system should be able to synchronize reviews made by the mobile application in offline mode.

- FR-3.9.2: The system should resolve conflicts during synchronization based on timestamps.
- FR-3.9.3: Creating new flashcards in offline mode should not be allowed.

FR-3.10: Media Storage Management

- FR-3.10.1: The system must store media references for audio and screenshot files on flashcards.

- FR-3.10.2: The system must ensure that media files can be accessed securely from the backend.

FR-3.11: Health Check & Monitoring

- FR-3.11.1: The system should provide a health check endpoint to verify that the backend services are up and running.

- FR-3.11.2: This endpoint should be used for synchronization and the mobile application.

FR-3.12: Logging & Error Tracking

- FR-3.12.1: The system must log all critical backend operations.

- FR-3.12.2: Errors should be logged at appropriate log levels (INFO, WARN, ERROR).

- FR-3.12.3: Logging should be done in a way that does not disclose user data.

Mobile App

FR 4.1: User Authentication: The mobile application shall allow users to log in using their existing account credentials.
Authenticated users shall only be able to access their own flashcards, review history, and statistical data.

FR 4.2: Flashcard Synchronization: The mobile application shall synchronize flashcards, review schedules, and user progress data with the backend system.
This synchronization shall ensure data consistency between the Chrome extension and the mobile application.

FR 4.3: SRS-Based Review Sessions: The mobile application shall provide vocabulary review sessions based on a Spaced Repetition System (SRS – FSRS algorithm).
Flashcards presented during review sessions shall be selected according to the review schedule determined by the system.

FR 4.4: Flashcard Detail View: The mobile application shall provide a detailed view for each flashcard.
This view shall display the word's meaning, audio pronunciation (if available), example sentence, and usage context on a single screen.

FR 4.5: User Self-Assessment: The mobile application shall allow users to indicate whether they remember a flashcard or not.
The feedback provided by the user shall be used to update the review schedule of the SRS algorithm.

FR 4.6: Offline Operation Support (Offline Mode): The mobile application shall be able to operate in environments without an internet connection (e.g., subway, airplane).
For this purpose, required data shall be stored locally on the device and automatically synchronized when an internet connection becomes available.

FR 4.7: User Progress Dashboard: The mobile application shall provide a progress dashboard that displays the user's daily reviewed flashcard count, learning streak information, and overall learning and progress statistics.

FR 4.8: Streak System: The mobile application shall provide a streak system that tracks usage and review activities performed on consecutive days in order to encourage daily application usage.

FR 4.9: Badge and Achievement System: The mobile application shall unlock badges when users reach specific language proficiency levels (e.g., A2, B1).
The user's progress within the current proficiency level shall be visually represented.

FR 4.10: Notification System: The mobile application shall send push notifications to remind users of scheduled review times.
Notification frequency shall be configurable by the user and limited to a non-intrusive level.

FR 4.11: Local Data Storage: The mobile application shall store flashcards, review history, and synchronization-related data locally on the device to support offline usage.

FR 4.12: Synchronization and Error Handling: The mobile application shall be able to detect errors that occur during synchronization.
When the connection is restored, failed synchronization operations shall be retried automatically.
User progress shall not be lost due to temporary connectivity issues.

FR 4.13: Flashcard Knowledge Status Tracking
The mobile application shall allow users to assign a knowledge status to each flashcard as "known," "learning," "unknown," or "ignored."
The selected knowledge status shall be stored in the backend system and synchronized across all platforms.

2. Use Case Diagrams:

Chrome Extension:

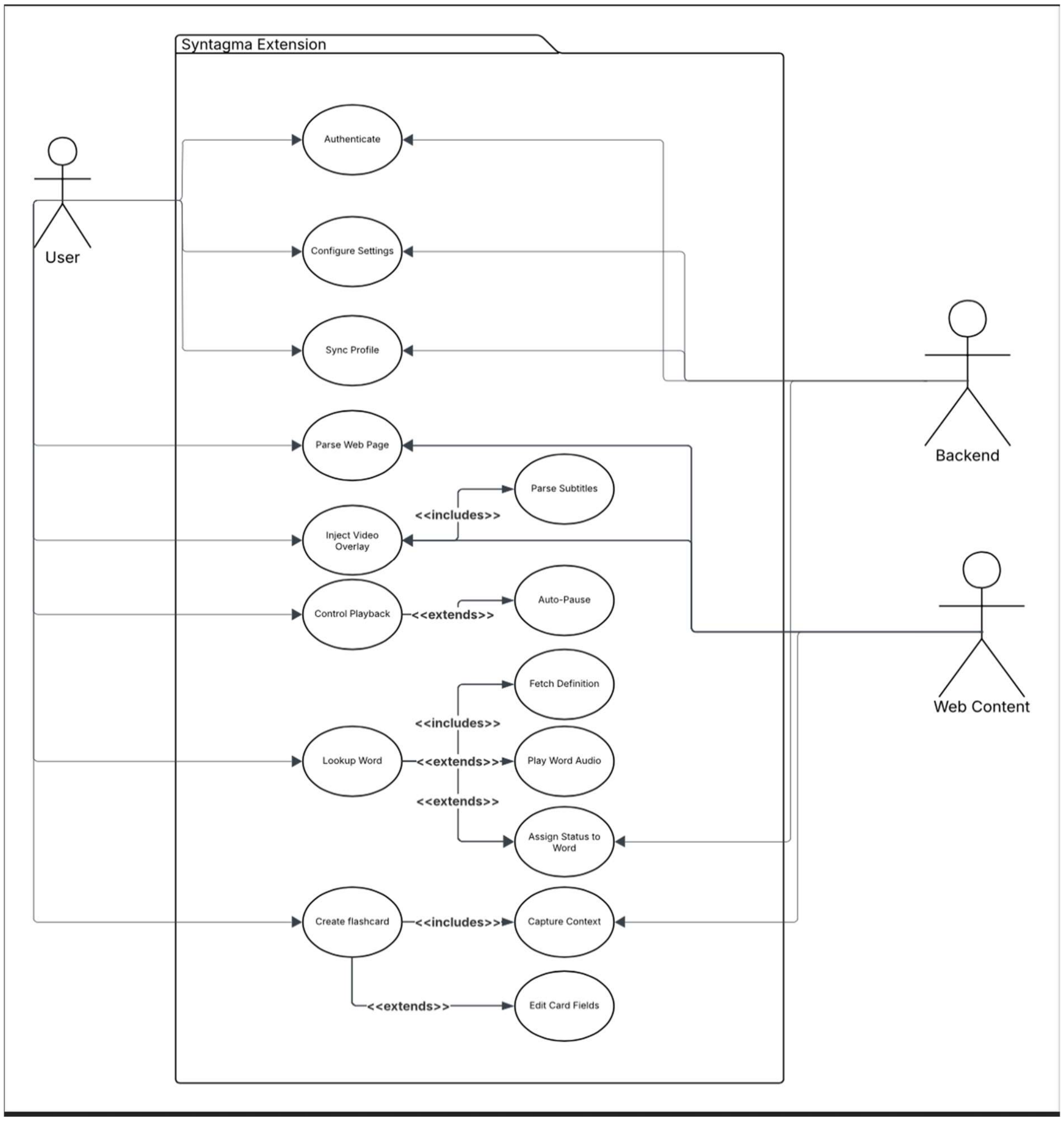

Figure 3 Use Case Diagram of Chrome Extension

Mobile Application:

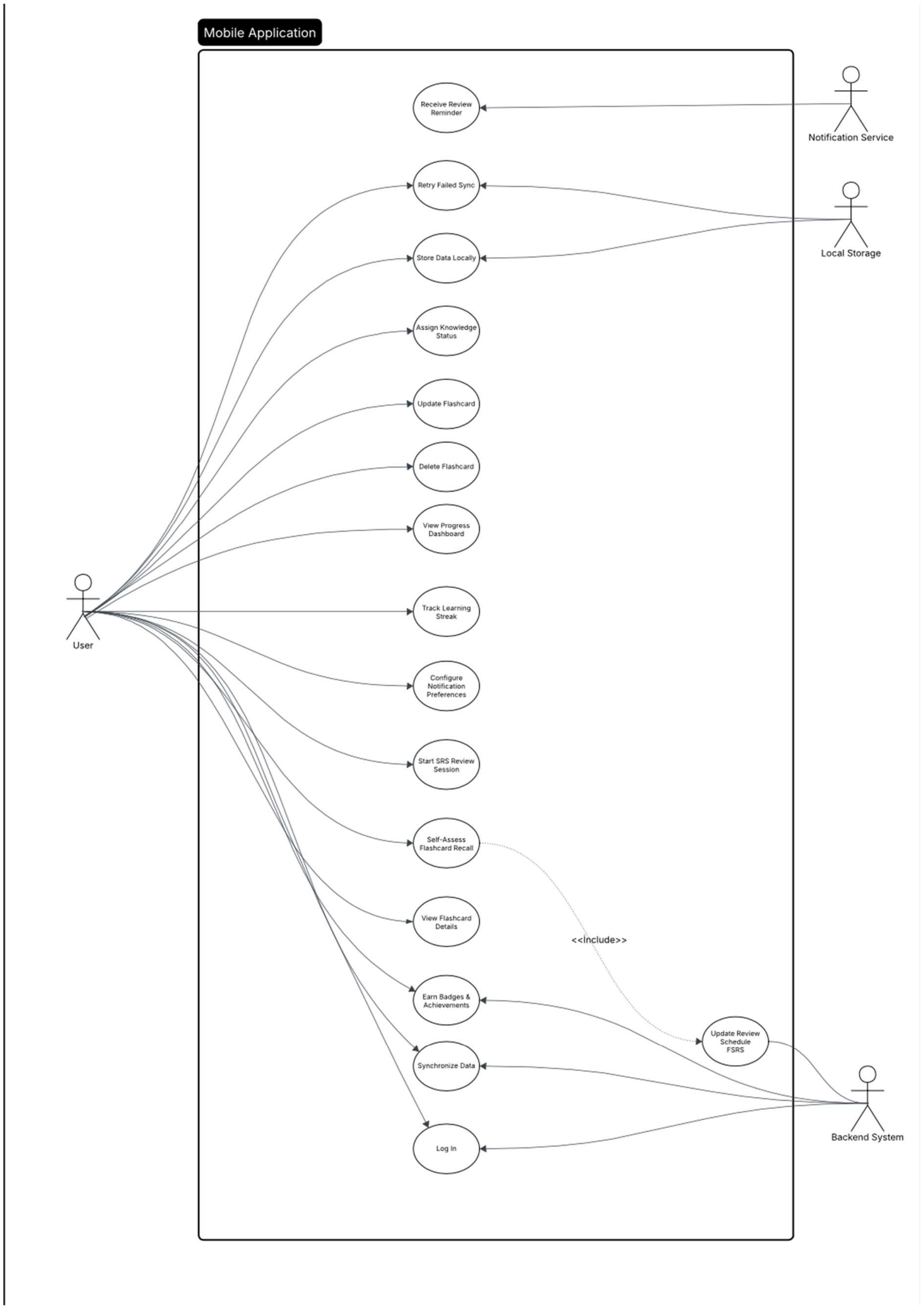

Figure 4 Use Case Diagram of Mobile Application

3. Data Flow Diagrams:

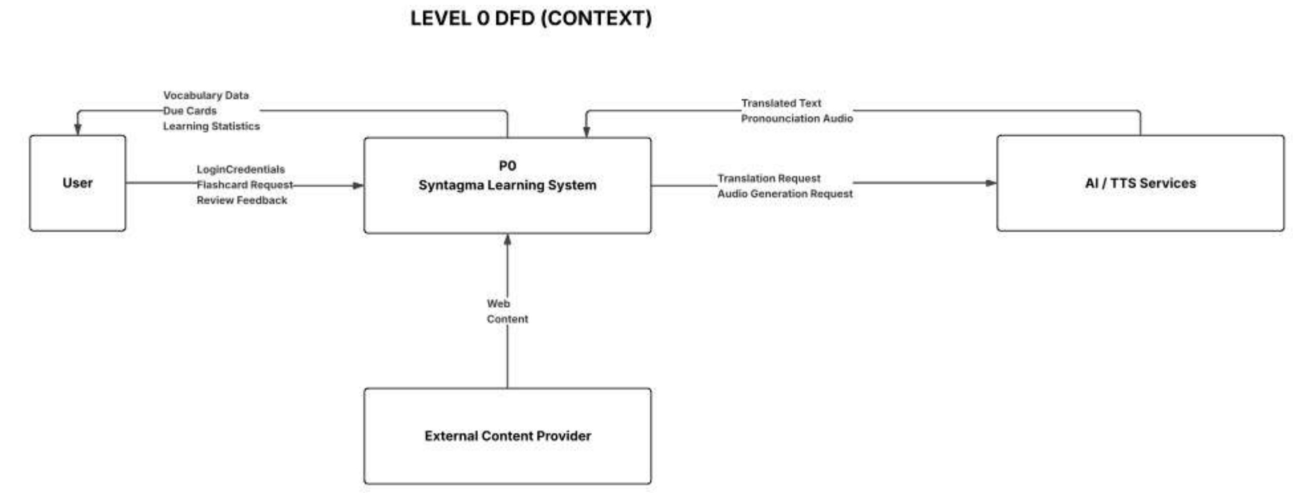

Figure 5 Data Flow Diagram of Syntagma Learning System Level 0

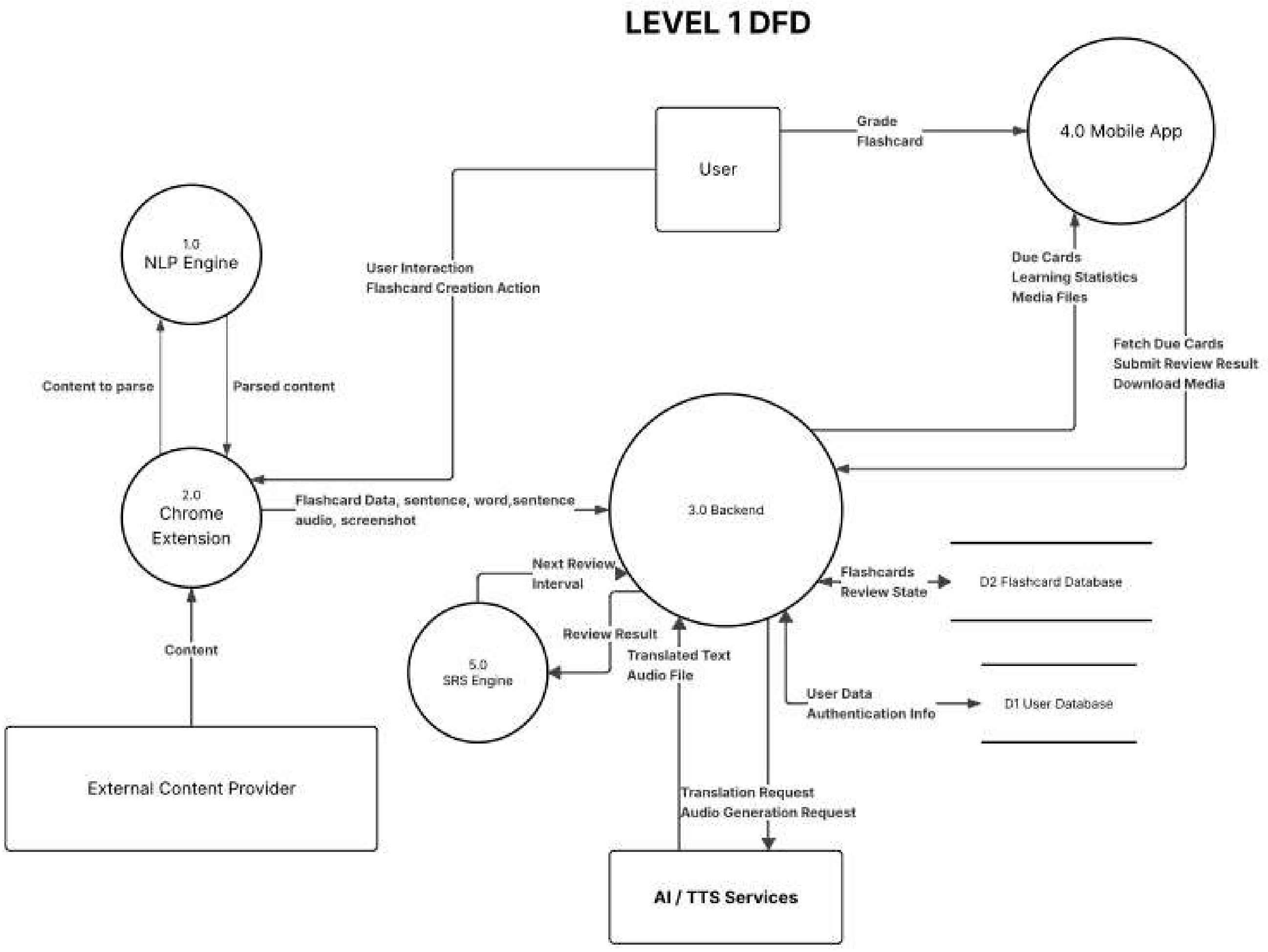

Figure 6 Data Flow Diagram of Syntagma Learning System Level 1

References

[1] Duolingo, “Main Page,” [Online]. Available: https://tr.duolingo.com/. [Accessed 4 Nov 2025].

[2] Memrise, “Memrise : Learn a Language,” [Online]. Available: https://www.memrise.com. [Accessed 11 Nov 2025].

[3] Babbel, “Babbel – Language Learning,” [Online]. Available: https://www.babbel.com. [Accessed 11 Nov 2025].

[4] Busuu, “Busuu: Learn Languages,” [Online]. Available: https://www.busuu.com. [Accessed 11 Nov 2025].

[5] Rosetta Stone Ltd., “Rosetta Stone: Language Learning,” [Online]. Available: https://www.rosettastone.com. [Accessed 11 Nov 2025].

[6] LingQ Inc., “LingQ – Language Learning App,” [Online]. Available: https://www.lingq.com. [Accessed 7 Nov 2025].

[7] Anki, “What spaced repetition algorithm does Anki use?,” 2023. [Online]. Available: https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html. [Accessed 4 Nov 2025].

[8] SuperMemo Guru, “Three component model of memory,” [Online]. Available: https://supermemo.guru/wiki/Three-component_model_of_memory. [Accessed 27 Nov 2025].

[9] Gradle, “Gradle User Manual,” [Online]. Available: https://docs.gradle.org/current/userguide/userguide.html. [Accessed 4 Nov 2025].

[10] Hibernate, “Introduction to Hibernate ORM,” [Online]. Available: https://docs.hibernate.org/orm/7.1/introduction/html_single/. [Accessed 4 Nov 2025].

[11] Amazon Web Services, “What Is Boilerplate Code?,” [Online]. Available: https://aws.amazon.com/what-is/boilerplate-code/. [Accessed 27 Nov 2025].

[12] Baeldung, “Introduction to Project Lombok,” [Online]. Available: https://www.baeldung.com/intro-to-project-lombok. [Accessed 4 Nov 2025].

[13] GeeksforGeeks, “Software Engineering | Mock Introduction,” [Online]. Available: https://www.geeksforgeeks.org/software-engineering/software-engineering-mock-introduction/. [Accessed 27 Nov 2025].

[14] Baeldung, “Mockito Annotations,” [Online]. Available: https://www.baeldung.com/mockito-annotations. [Accessed 4 Nov 2025].

[15] C. A. H. A. a. M. P.-G. F. J. Palacios-Hidalgo, “Origins, concept and didactic applications – systematic review of literature (2012–2019),” *Technology, Knowledge and Learning,* vol. 25, no. 4, p. 853–879.

[16] R. D. a. K. K. K. Guntupally, “Spring Boot based REST API to improve data quality report generation for big scientific data: ARM Data Center example,” in *Proc. IEEE Int. Conf. Big Data*, Seattle, WA, USA, 2018.

[17] Wikipedia, “Facade pattern,” [Online]. Available: https://en.wikipedia.org/wiki/Facade_pattern. [Accessed 27 Nov 2025].

[18] QOS.ch, “Name of Web Page:,” [Online]. Available: https://www.slf4j.org/manual.html. [Accessed 27 Nov 2025].

[19] Wikipedia, “Coupling (computer programming),” [Online]. Available: https://en.wikipedia.org/wiki/Coupling_(computer_programming). [Accessed 2025 Nov 2025].

[20] QOS.ch, “Logback Manual,” [Online]. Available: https://logback.qos.ch/manual/index.html. [Accessed 27 Nov 2025].

[21] M. N. Jeyaraj, S. Sucharitharathna, C. Senarath, Y. Kanagaraj and I. Udayakumara, “Cognitive visual-learning environment for PostgreSQL,” arXiv, 2022.

[22] S. V. Salunke and A. Ouda, “A performance benchmark for the PostgreSQL and MySQL databases,” *Future Internet,* vol. 16, no. 10, p. 382, 2024.

[23] DBeaver Community, “DBeaver: Free Universal Database Tool,” [Online]. Available: https://dbeaver.io. [Accessed 27 Nov 2025].

[24] A. L. Mhatre, “Microservices architecture using Docker and Kubernetes,” *Int. J. for Multidisciplinary Research,* vol. 5, no. 5, p. 1–8, 2023.

[25] A. Lama, “Documenting APIs with Swagger: Enhancing Developer Experience,” *International Journal of Science and Technology,* vol. 15, no. 2, pp. 165-169, 2025.

[26] P. P. Kore, M. J. Lohar, M. T. Surve and S. Jadhav, “API testing using Postman tool,” *Int. J. Research in Applied Science and Engineering Technology,* vol. 10, no. 12, p. 841–843, 2022.

[27] y.-d. Developers, “yt-dlp: A Feature-Rich Video Downloader (Fork of youtube-dl),” GitHub, 2021. [Online]. Available: https://github.com/yt-dlp/yt-dlp. [Accessed 11 Nov 2025].

[28] I. Bernat, “netflix-to-srt: Export Netflix subtitles to SRT format,” GitHub Repository, 2019. [Online]. Available: https://github.com/isaacbernat/netflix-to-srt. [Accessed 7 Nov 2025].

[29] Mozilla, “Readability,” Github, [Online]. Available: https://github.com/mozilla/readability. [Accessed 7 11 2025].

[30] Graype Systems, “WinkNLP – Developer Friendly NLP in Node.js,” WinkJS Documentation, 2023. [Online]. Available: https://winkjs.org. [Accessed 11 Nov 2025].

[31] S. K. Small, “compromise.js,” Github, 2015. [Online]. Available: https://github.com/spencermountain/compromise.

[32] B. N. M. Brysbaert, “SUBTLEXus: Word frequencies based on 51 million words from American English television subtitles,” *Behavior Research Methods,* vol. 41, no. 3, p. 978–990, 2009.

[33] I. M. S. V. L. a. A. B. M. Honnibal, “spaCy: Industrial-strength Natural Language Processing in Python,” 2020. [Online]. Available: https://spacy.io. [Accessed 7 Nov 2025].

[34] Google, “Text-to-Speech API,” Google, [Online]. Available: https://cloud.google.com/text-to-speech. [Accessed 11 Nov 2025].

[35] X. Z. a. Y. G. D. He, “Mobile-assisted language learning: A review of recent developments,” *Journal of Information Technology & Software Engineering,* vol. 9, no. 2, p. 1–6, 2019.

[36] M. W. Y. Reinders, “The effects of gamification on language learning: A study using a mobile language learning app,” *Computer Assisted Language Learning,* vol. 36, no. 2, p. 195–216, 2023.

[37] R. Gafni, D. B. Achituv and G. J. Rachmani, “Learning foreign languages using mobile applications,” *Journal of Information Technology Education: Research,* vol. 16, p. 301–317, 2017.

[38] R. B. a. J. Nielsen, “User Interface Design for Mobile Applications,” *2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC),* 2019.

[39] O. Viberg and Å. Grönlund, “Mobile assisted language learning: A literature review,” *11th World Conf. Mobile and Contextual Learning (mLearn 2012),* vol. 955, p. 9–16, Oct 2012.

[40] A. Wohllebe, D.-S. Hübner, U. Radtke and S. Podruzsik, “Mobile Apps in Retail: Effect of Push Notification Frequency on App User Behavior,” *Innovative Marketing,* vol. 17, no. 2, p. 102–111, 2021.

[41] B. Eisenman, Learning React Native: Building Native Mobile Apps with JavaScript, Sebastopol, CA, USA: O’Reilly Media, 2016.