# 

**TOBB ETU**

**Economy & Technology University**

**BIL 481**

**Software Requirements Specification (SRS)**

## **Lexora - AI-Powered Dictionary**

**Team Members**Gülsüm Yıldırım (221404031)Nurefşan Olfaz (211301008)Ozan Bayer (211201024)

## **TABLE OF CONTENTS**

1. **Introduction**  
    1.1 Purpose  
    1.2 Scope  
    1.3 Definitions, Acronyms, and Abbreviations  
    1.4 References
2. **Functional Requirements**
3. **Non-Functional Requirements**
4. **Effort Estimations**  
    4.1 Estimation Method  
    4.2 Effort Estimation by Phase  
    4.3 Effort Estimation by Task Type  
    4.4 Effort per Person Mapped to Responsibilities
5. **Task Assignments**  
    5.1 Team Members and Roles  
    5.2 Functional Requirements Assignment  
    5.3 Rationale for Task Assignment
6. **Document-Specific Task Matrix**
7. **References**

## **1. Introduction**

### **1.1 Purpose**

The purpose of this document is to define the functional and non-functional requirements of the Lexora – AI-Powered Dictionary system. It serves as a formal reference for the project team during design, implementation, and testing phases. The document specifies what the system will do, the scope of its capabilities, and the quality attributes expected from the final product. It will be used by developers, project supervisors, and evaluators to ensure that the implemented system aligns with the initially agreed specifications.

### **1.2 Scope**

Lexora is an AI-powered, web-based language learning dictionary designed to help users expand and manage their vocabulary efficiently. The system automatically generates word definitions, explanations, example sentences, and synonyms using artificial intelligence. It also enables users to create personalized vocabulary libraries, track their daily learning activity through a streak tracking system, and generate podcasts from word groups using text-to-speech technology.  
The system will be developed using **Python Flask** for backend logic and **HTML, and JavaScript** for the frontend interface. All data, including words and their generated meanings, will be stored locally in **JSON format** within a lightweight database. The application will operate for a single user only and will feature a responsive web design suitable for both desktop and mobile browsers.

### **1.3 Definitions, Acronyms, and Abbreviations**

| **Term / Acronym** | **Definition** |
| --- | --- |
| **AI** | Artificial Intelligence – technology used to generate word definitions, explanations, and examples automatically. |
| **TTS** | Text-to-Speech – API used to convert text content (word and definition) into synthesized speech for podcast generation. |
| **JSON** | JavaScript Object Notation – data format used to store definitions, examples, and related words locally in the database. |
| **Flask** | A lightweight Python web framework used for backend development and server-side routing. |
| **Frontend** | The client-side part of the application, built using HTML, and JavaScript, where users interact with the system. |
| **Backend** | The server-side logic, database management, and API integration layer developed in Python Flask. |
| **Streak** | A consecutive count of active learning days, measured by the number of words searched within 24-hour periods. |

## **2. Functional Requirements**

|  |  |
| --- | --- |
| **FR ID** | **Description** |
| **FR-1** | The system **must** generate a concise and accurate **definition and explanation** for any user-entered word using an external AI service. (This functionality is testable by comparing the AI-generated output against a standard dictionary definition for correctness and conciseness.) |
| **FR-2** | The system **must** generate at least one **contextually appropriate example sentence** for the searched word using an external AI service to demonstrate its correct usage. |
| **FR-3** | The system **must** retrieve and display a list of **related words and synonym suggestions** for the searched word. |
| **FR-4** | The system **must** store all newly searched words, their AI-generated definitions, and example sentences in the internal database using the **JSON data format**. |
| **FR-5** | The system **must** allow the single user to **add** and permanently **delete** word entries within their Personal Vocabulary Library. (A user account system is explicitly excluded; all data management is for a single, non-authenticated user.) |
| **FR-6** | The system **must** implement a **Streak Tracking System** that automatically records the total number of **words** searched by the user in a 24-hour period. (The streak counter must be updated upon each successful word search.) |
| **FR-7** | The system **must** allow the user to select one or more word entries from their library and request the generation of an audio file (**Podcast**). For each selected word, the audio **must** synthesize the pronunciation of the word, followed by its definition (Word + Definition) using a **Text-to-Speech (TTS) API**. |
| **FR-8** | The system **must** provide a function to **export the selected word list and their definitions** from the Personal Vocabulary Library into a **PDF file format** for offline use. |
| **FR-9** | The application **must** display the user's ongoing **Streak Count** (the number of consecutive days with recorded learning activity) on the main interface. |
| **FR-10** | The system **must** provide an in-browser audio player interface to play the newly generated Podcast (FR-7) and an option to **download the audio file** (e.g., as MP3). |
| **FR-11** | The system **must** allow the user to select the **accent** (e.g., British or American) for the synthesized voice **before** generating the Podcast audio (FR-7). |
| **FR-12** | The system **must** limit the user to generating a Podcast containing a **maximum of 10 words** per day to manage external API usage costs. (If the user selects more than 10, the system must display a warning.) |

## **3. Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **Exclusion ID** | **Excluded Feature** | **Rationale** |
| **EX-1** | **User Registration and Authentication** | The application is designed for a single, local user only. No features related to account creation, login, password management, or user roles will be implemented. |
| **EX-2** | **Multi-User Functionality** | The system will not support multiple distinct users, shared data, or personalized data specific to different logins. |
| **EX-3** | **Live Data Synchronization** | There will be no cloud synchronization or remote backup mechanism for the user's vocabulary data. Data integrity relies solely on the local storage/database instance. |
| **EX-4** | **Word Editing Functionality** | While adding and deleting words is included (FR-5), the ability to modify or *edit* the content (definition, example sentences) of an existing word entry is excluded. |
| **EX-5** | **Flashcard or Review System** | Dedicated spaced repetition, flashcard browsing interfaces, learning sessions, or quiz features are excluded from the initial scope. |
| **EX-6** | **Advanced Profile Settings** | Customization options beyond the included features (e.g., changing interface theme, language, notifications, or displaying user-specific statistics beyond the Streak Count) are excluded. |
| **EX-7** | **Advanced Search Filtering** | The search functionality (if implemented for the local library) will be based on simple text matching; advanced filtering by date, category, or other metadata is excluded. |
| **EX-8** | **Automatic Data Scraping** | The system will not include any functionality to automatically retrieve definitions or information from web sources other than the designated AI service (FR-1). |

## **4. Effort Estimations**

### **4.1 Estimation Method**

We have used **Expert Judgment** and **Timeboxed Estimation** (Bütçelenmiş Zaman Tahmini) to align with strict time constraints. Our estimates are based on:

* Prioritizing core features (MVP - Minimum Viable Product).
* Aggressively time-boxing development to meet the 5-week deadline (27oct – 3dec).
* Assuming maximum code reuse and focusing only on the included FRs.

**Assumptions:**

* Work will be distributed among 3 team members.
* Each team member can contribute exactly **5 hours per week**.
* Project duration is exactly **5 weeks**.
* Total available effort for the project is **75 person-hours**.

### 

### **4.2 Effort Estimation by Phase**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Tasks Included** | **Effort (Person-Hours)** |
| Requirements & Design | Scope Finalization, Simple UI Mockups, Database Schema (FR-4) | **9 hours** |
| Core Development | All Feature Implementation (FR 1-12) | **44 hours** |
| Testing & Integration | Unit Testing, Feature Integration, Bug Fixing | **11 hours** |
| Deployment & Documentation | Server Setup, Final Report, Technical Write-up | **11 hours** |
| **Total Project Effort** | (Sum of all above) | **75 person-hours** |

### **4.3 Effort Estimation by Task Type**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Allocated Effort** | **Weekly Average** |
| **Gülsüm Yıldırım** (Backend & Data) | **25 hours** | 5 hours/week |
| **Nurefşan Olfaz** (Frontend & UI/UX) | **25 hours** | 5 hours/week |
| **Ozan Bayer** (AI/TTS Integration) | **25 hours** | 5 hours/week |
| **Total Team Effort** | **75 person-hours** |  |

### **4.4 Effort per Person Mapped to Responsibilities**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Primary Role** | **Secondary Responsibilities** |
| **Gülsüm Yıldırım** | Backend Developer & Database Administrator | Requirements Analyst, User Documentation |
| **Nurefşan Olfaz** | Frontend Developer & UI/UX Designer | System Architect, Deployment Specialist |
| **Ozan Bayer** | AI/TTS Integration Specialist | Testing Coordinator, Logic Implementer |

## **5. Task Assignments**

**5.1 Team Members and Roles**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Primary Role** | **Secondary Responsibilities** |
| **Nurefşan Olfaz** | Frontend Developer & UI/UX Designer | Requirements Analyst, User Documentation |
| **Gülsüm Yıldırım** | Backend Developer & Database Administrator | System Architect, Deployment Specialist |
| **Ozan Bayer** | AI/TTS Integration Specialist | Testing Coordinator, Logic Implementer |

**5.2 Functional Requirements Assignment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assigned To** | **FR Group Name** | **Requirement ID** | **Requirement Name** | **Est. Effort (Hours)** |
| **Nurefşan Olfaz** | **Frontend & User Interface (UI)** | **FR-5** | **Word Addition and Deletion (Library Management)** | **5** |
|  |  | **FR-9** | **Streak Count Display on Interface** | **8** |
|  |  | **FR-10** | **Podcast Player and Download Option (UI)** | **6** |
|  |  | **UI Development & Responsive Design** | **Core HTML/CSS for Mobile/Desktop** | **6** |
|  |  | **Subtotal** |  | **25 hours** |
| **Gülsüm Yıldırım** | **Backend & Data Management** | **FR-4** | **JSON Data Storage and Database Implementation** | **8** |
|  |  | **FR-8** | **Converting Word List to PDF Format** | **5** |
|  |  | **FR-12** | **Daily Podcast Generation Limit Logic** | **6** |
|  |  | **Backend API & Deployment** | **Core API, Server Setup, Deployment** | **6** |
|  |  | **Subtotal** |  | **25 hours** |
| **Ozan Bayer** | **AI & Core Feature Integration** | **FR-1** | **AI Definition/Explanation Generation** | **5** |
|  |  | **FR-2/FR-3** | **AI Example Sentences & Synonym Suggestions** | **5** |
|  |  | **FR-6** | **Streak Tracking Logic (Total Words Searched)** | **10** |
|  |  | **FR-7/FR-11** | **Podcast Generation & Accent Selection (TTS API)** | **5** |
|  |  | **Subtotal** |  | **25 hours** |
| **Total FR Effort** | **(Sum of the above)** |  |  | **75 hours** |

**5.3 Rationale for Task Assignment**

### Distribution Approach

Our task assignment strategy is rooted in architectural specialization and time-boxed efficiency. Given the strict 8-week deadline and the 96-hour total effort budget, tasks were grouped into cohesive blocks (Frontend, Backend/Data, Integration) to ensure maximum focus and expertise application for the Minimum Viable Product (MVP).

### Role-Based Assignment and Competency Alignment

The 12 Functional Requirements (FRs) are distributed equally (32 hours/person) by aligning the technical requirements with each team member's core competencies and developmental interests:

* Nurefşan Olfaz (Frontend Lead & UI/UX):
  + Focus: User Experience (UX) and Client-Side Implementation. Nurefşan is assigned all tasks related to the visual interface and user interaction (FR-5, FR-9, FR-10). This capitalizes on her demonstrated strengths in responsive design (HTML) and visual fidelity, ensuring the web application is intuitive, accessible, and highly usable on both mobile and desktop devices.
  + Assigned FRs: Library Management (FR-5), Streak Display (FR-9), Podcast Player (FR-10).
* Gülsüm Yıldırım (Backend Lead & Database):
  + Focus: Data Integrity and Infrastructure Control. Gülsüm is responsible for the core data persistence layer (FR-4) and critical server-side logic (FR-8, FR-12). Her expertise ensures the robust storage of JSON data and the reliable enforcement of key business rules, such as the daily 10-word Podcast limit (FR-12) and the professional PDF export functionality (FR-8).
  + Assigned FRs: Database Setup (FR-4), PDF Export (FR-8), Daily Limit Logic (FR-12).
* Ozan Bayer (AI/TTS Integration Specialist):
  + Focus: External API Integration and Complex Logic. Ozan is assigned all tasks requiring connection to external services and implementing the project's unique features (FR-1, FR-6, FR-7/FR-11). His problem-solving acumen is essential for integrating the AI Definition/Synonym generation and mastering the Text-to-Speech (TTS) API for accurate podcast creation with accent selection.
  + Assigned FRs: AI Generation (FR-1, FR-2/FR-3), Streak Logic (FR-6), Podcast Generation/Accent Selection (FR-7/FR-11).

### **Effectiveness and Project Management**

1. Parallel Development Strategy: The functional grouping allows team members to work largely independently during the initial development weeks (First 4 weeks). This approach drastically minimizes dependency conflicts and enables rapid progress toward integration milestones.
2. Balanced Workload: The assignments strictly adhere to the 32-hour budget per member. This guarantees that the project is feasible within the 8-week, 4 hours/week constraint, demonstrating effective resource management.
3. Cohesion and Integration Points: Tasks are strategically paired to ensure seamless integration. For example, Ozan's work on Podcast Generation (FR-7) directly feeds the output required by Gülsüm's Podcast Player UI (FR-10), establishing clear integration points for collaborative testing.
4. Risk Mitigation: Assigning critical non-functional components (Data, Deployment, Testing Coordination) to specialized leads minimizes the risk of failure in foundational areas.

This distribution not only aligns with industry best practices but also maximizes efficiency by pairing high-complexity features with the appropriate expertise, ensuring successful delivery of the MVP.

**6. Document Specific Task Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Description** | **Responsible Team Member** | **Status** | **Completion Date** |
| **Scope Definition (Included/Excluded)** | All team members | Completed | Oct 27, 2025 |
| **FR Group 1 Definition (FR-5, FR-9, FR-10)** | Nurefşan Olfaz | Completed | Oct 29, 2025 |
| **FR Group 2 Definition (FR-4, FR-8, FR-12)** | Gülsüm Yıldırım | Completed | Oct 28, 2025 |
| **FR Group 3 Definition (FR-1, FR-2/3, FR-6, FR-7/11)** | Ozan Bayer | Completed | Oct 27, 2025 |
| Non-Functional Requirements Specification (NFR) | All team members | Completed | Oct 31, 2025 |
| **Effort Calculation and 96h Budget Finalization** | All team members | Completed | Nov 3, 2025 |
| Task Distribution Rationale Write-up (4.4) | All team members | Completed | Oct 31, 2025 |
| Task Matrix (4.2 & 4.3) Creation | Ozan Bayer | Completed | Nov 27, 2025 |
| System Architecture/Design Overview Write-up | Gülsüm Yıldırım | Completed | Oct 28, 2025 |
| Document Formatting and Consistency Check | Nurefşan Olfaz | Completed | Oct 28, 2025 |
| Final Proofreading and Quality Check | All team members | Completed | Nov 3, 2025 |

### **7. References**

The following sources and materials were referenced during the preparation of this Software Requirements Specification document:

* OpenAI API Documentation – for AI-based text generation and example sentence creation.
* Flask Official Documentation – for backend development and routing structure.
* W3C HTML and CSS Standards – for frontend structure, styling, and responsive design compliance.

BIL 481 course project templates and guidelines provided by instructors

**Prepared by:** Gülsüm Yıldırım, Nurefşan Olfaz, Ozan Bayer