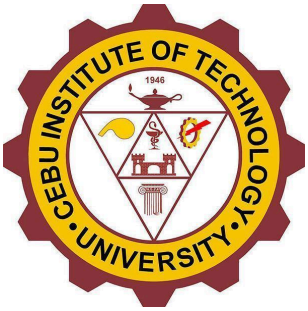

CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

AudioScholar

Version 1.0

February 04, 2025
Prepared by

Biacolo, Math Lee L.
Terence, John Duterte
Orlanes, John Nathan

Change History

Table of Contents

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) document defines the requirements for the AudioScholar application. AudioScholar is designed to assist students, professionals, and lifelong learners in transforming lecture audio into actionable insights. This document serves as a comprehensive guide for developers, testers, and stakeholders, outlining the functionality, performance, and design constraints of the software. It aims to ensure that all parties have a clear and shared understanding of the project goals and requirements, facilitating efficient development and successful delivery of the final product. This SRS covers both the mobile (Android) and web platform components of AudioScholar.

1.2. Scope

AudioScholar is an intelligent application that supports both mobile and web platforms. The mobile app (developed using Android Kotlin) allows users to record audio in real-time, while the web platform (developed with ReactJS) provides users the ability to upload pre-recorded audio files for summarization and content recommendations. The software will also feature secure user authentication and a cloud-hosted database solution for storing user data and processed content. Key functionalities include:

- Real-time recording, automated summarization, and content recommendations for the mobile app.
- Audio upload, summarization, and content recommendations for the web platform.
- Secure login using JWT-based authentication for both platforms.

AudioScholar aims to provide an efficient and accessible tool for managing and reviewing lecture content, ultimately enhancing the learning experience. However, the application will not provide advanced audio editing or collaboration features at this stage.

1.3. Definitions, Acronyms and Abbreviations

- **AudioScholar**: The software application being developed.
- **AI**: Artificial Intelligence, which is used for text summarization and content recommendations.
- **Android Kotlin**: A programming language used to build the mobile application.
- **ReactJS**: A JavaScript framework used to build the web frontend.
- **JWT**: JSON Web Token, a secure authentication method for user login.
- **NLP**: Natural Language Processing, a technology used to generate summaries from recorded audio.

1.4. References

- *[1] "Develop for Android," Android Developers. https://developer.android.com/develop*
- *[2] "Getting Started – React," reactjs.org, 2024. https://legacy.reactjs.org/docs/getting-started.html*
- *[3] "Kotlin docs | Kotlin," Kotlin Help. https://kotlinlang.org/docs/home.html*
- *[4] Spring, "Spring Projects," Spring.io, 2019. https://spring.io/projects/spring-boot*
- *[5] Firebase, "Documentation | Firebase," Firebase, 2019. https://firebase.google.com/docs*

2. Overall Description

2.1. Product perspective

AudioScholar is a standalone application designed to provide users with tools for recording, summarizing, and augmenting their learning experience from audio content. It comprises two main modules: a mobile application for Android devices and a web platform accessible through standard web browsers. The application adheres to a client-server architecture, with the mobile app and web platform acting as clients and a Java Spring Boot backend handling data processing and storage. The system integrates with external services via APIs to enhance its functionality. The system is designed to be data-driven, providing personalized recommendations and insights based on user activity and content analysis.

Module 1: Mobile App (Android Kotlin)

- **1.1 Real-Time Audio Recording:** The app allows users to record audio in real-time.
- **1.2 Automated Summarization:** The app automatically generates summaries of recorded audio.
- **1.3 Content Recommendations:** The app provides recommendations for supplementary learning materials.
- **1.4 Pause and Resume Recording:** Users can pause and resume audio recording as needed.
- **1.5 Login:** Users can create accounts and log in to the app.
- **1.6 Audio Playback:** Users can playback recorded audio files.

Module 2: Web Platform (ReactJS)

- **2.1 Audio Upload:** Users can upload audio files from their devices.
- **2.2 Automated Summarization:** The platform automatically generates summaries of uploaded audio.
- **2.3 Content Recommendations:** The platform provides recommendations for supplementary learning materials.
- **2.4 Login:** Users can create accounts and log in to the web platform.
- **2.5 Audio Playback:** Users can playback uploaded audio files.
- **2.6 User Profile Management:** Users can manage their profiles, including settings and preferences.

2.2. User characteristics

- **Students**: Users who are recording lectures and meetings for review. They have the ability to create accounts, record audio, view summaries, and receive learning content recommendations.
- **Educators**: Teachers who use the platform to upload lectures or presentations for students to access. They can also benefit from automated summaries and content suggestions.
- **Lifelong Learners**: Individuals who want to review learning materials, access summaries, and receive personalized recommendations.
- **Admin Users**: Users who manage accounts, monitor usage, and ensure the smooth operation of the system. They have elevated privileges for managing the application.

2.3. Constraints

- **Hardware Limitations**: The mobile app should be optimized for various Android devices with different hardware specifications (e.g., storage, processing power).
- **Regulatory Policies**: The system must adhere to data privacy regulations such as GDPR.
- **Interfaces to Other Applications**: Integration with external APIs for content recommendations (e.g., YouTube, TED Talks).
- **Reliability Requirements**: Ensure the system has minimal downtime and high availability, particularly for real-time recording features.
- **Security**: User data must be securely stored and transmitted, and authentication processes must be protected using secure methods like JWT.

2.5. Assumptions and dependencies

- **Assumptions**:

    1. The system assumes that users have access to modern smartphones and computers with stable internet connections to facilitate real-time audio recording, upload functionality, and seamless interaction with the platform.
    2. It is assumed that the Firebase Realtime Database or Cloud Firestore will be available for managing user data, recordings, and other content storage needs.
    3. The mobile app assumes that Android OS versions 8.0 (Oreo) and higher are available to ensure compatibility with Kotlin and required APIs.

4. The web platform assumes that users will have access to modern web browsers (e.g., Chrome, Firefox, Safari) to use the features such as audio upload and viewing content recommendations.

- **Dependencies**:

1. The **Speech-to-Text API** and **AI Summarization API** are essential dependencies for real-time transcription and summarization of audio content. The functionality of these features depends on the availability and reliability of these external services.
2. The **JWT Authentication Service** (or equivalent) must be available to securely handle user authentication across both mobile and web platforms.
3. **Firebase Realtime Database or Cloud Firestore** must be fully integrated and operational to ensure real-time storage and retrieval of user data, audio metadata, and processed summaries.
4. For content recommendations, external APIs or services (e.g., YouTube, TED Talks) are required for fetching relevant educational content based on user preferences and summarized data.

3. Specific Requirements

3.1. External interface requirements

3.1.1. Hardware interfaces

The application interacts with various hardware components of the user's device.

Mobile Devices: The Android mobile app will interact with the device's microphone to record audio. The app will also require storage space on the device to save recorded audio and metadata before it is uploaded or processed.

- The app supports mobile devices running Android version 8.0 (Oreo) and higher, which should support Kotlin and the required APIs.
- The app requires access to device capabilities such as Wi-Fi for internet connectivity, camera and microphone for audio recording, and local storage for temporarily saving files.

Web Platform: The web platform will interact with the user's device through standard web browsers (e.g., Chrome, Firefox, Safari). It will allow users to upload audio files, view summaries, and access content recommendations.

- Users will require access to a computer or laptop with an internet connection.
- The system will support common input devices (e.g., keyboard, mouse, touchpad) and will provide a responsive UI optimized for different screen sizes.

3.1.2. Software interfaces

The application interacts with several software components and external services.

Operating System:

- The mobile application will be built for Android OS, specifically versions 8.0 (Oreo) and above.
- The web platform will require a modern web browser for interaction, such as Chrome, Firefox, or Safari, to ensure compatibility with the ReactJS framework.

Database System:

- The system will use **Firebase Realtime Database** or **Cloud Firestore** to store user authentication details, audio metadata, processed summaries, and personalized recommendations.

- The backend will be integrated with these databases to allow the mobile and web apps to interact with the data.

External APIs:

- **Speech-to-Text API**: The system will utilize an external API (undecided) to convert audio recordings into transcribed text. This service will be invoked by the mobile and web platforms for transcription purposes.
- **AI Summarization API**: The system will integrate an AI summarization API (undecided) to process the transcribed text and generate concise summaries.
- **Content Recommendation API**: The system will pull content recommendations (e.g., educational articles, YouTube videos) using external APIs based on the content of the audio summaries.

3.1.3. Communications interfaces

The application communicates with the backend server and external services over the internet.

Local Network Protocols:

- The mobile app will communicate with the backend server via REST APIs over HTTPS. The communication will use standard HTTP methods (GET, POST, PUT, DELETE) to send and receive data between the mobile app and the server.
- The web platform will also use HTTPS to securely communicate with the backend through REST APIs.

Cloud Communication:

- The backend server, built using Java Spring Boot, will communicate with Firebase Realtime Database or Cloud Firestore through cloud services. This will enable real-time data storage and synchronization across users' devices.
- Data encryption in transit and at rest will be ensured to maintain security across all interfaces.

3.2. Functional requirements

Mobile App

1. **Real-Time Audio Recording:**

    - The application shall allow users to record audio in real-time. The recording should support a variety of audio formats (e.g., MP3, AAC).
2. **Automated Summarization:**
    - The application shall automatically generate summaries of recorded audio using a natural language processing (NLP) algorithm. The user should be able to adjust the summarization level (e.g., short, medium, long).
3. **Content Recommendations:**
    - The application shall provide recommendations for supplementary learning materials (e.g., articles, videos, research papers) based on the content of the recorded audio. The recommendations should be personalized to the user's interests and learning history.
4. **Pause and Resume Recording:**
    - The application shall allow users to pause and resume audio recording seamlessly. The system should maintain the audio quality and integrity during pause and resume operations.
5. **Login:**
    - The application shall require users to create an account and log in to access its features. The login process should support username/password authentication and third-party authentication (e.g., Google, Facebook).
6. **Audio Playback:**
    - The application shall allow users to playback recorded audio files. The playback interface should include controls for play, pause, stop, rewind, and fast forward.

Web Platform

1. **Audio Upload:**
    - The platform shall allow users to upload audio files from their devices. The platform should support a variety of audio formats (e.g., MP3, WAV, AAC) and file sizes.
2. **Automated Summarization:**
    - The platform shall automatically generate summaries of uploaded audio using a natural language processing (NLP) algorithm. The user should be able to adjust the summarization level (e.g., short, medium, long).
3. **Content Recommendations:**
    - The platform shall provide recommendations for supplementary learning materials (e.g., articles, videos, research papers) based on the content of the uploaded audio. The recommendations should be personalized to the user's interests and learning history.
4. **Login:**
    - The platform shall require users to create an account and log in to access its features. The login process should support username/password authentication and third-party authentication (e.g., Google, Facebook).
5. **Audio Playback:**
    - The platform shall allow users to playback uploaded audio files. The playback interface should include controls for play, pause, stop, rewind, and fast forward.
6. **User Profile Management:**

- The platform shall allow users to manage their profiles, including updating their personal information, changing their password, and adjusting their preferences.

3.4 Non-functional requirements

Performance

- *The application should respond to user requests within 2 seconds.*
- *The audio summarization process should complete within 30 seconds for a 60-minute audio file.*
- *The application should be able to handle a large number of concurrent users without performance degradation.*

Security

- *The application should protect user data from unauthorized access.*
- *The application should use secure communication protocols (HTTPS) to protect data in transit.*
- *The application should comply with data privacy regulations.*

Reliability

- *The application should be available 99.9% of the time.*
- *The application should be able to recover from failures gracefully.*
- *The application should be thoroughly tested to ensure its stability and reliability.*