Software Requirements Specification (SRS)

Speech-to-Text Assistant Web Application

Technology Stack: Python Flask + Whisper Model + Render Deployment

1. Introduction

1.1 Purpose

This document specifies requirements for a web-based Speech-to-Text Assistant that converts spoken audio into text using an AI speech recognition model. It serves as a reference for developers, testers, and stakeholders.

1.2 Scope

The system allows users to record or upload audio, convert speech to text, view, copy, and download transcripts, and access the service through a browser without installing software.

1.3 Definitions

STT: Speech to Text

API: Application Programming Interface

UI: User Interface

Whisper Model: AI model used for speech recognition

2. Overall Description

2.1 Product Perspective

The system is a standalone web application consisting of frontend interface, backend server, speech recognition engine, and cloud hosting environment.

2.2 Product Functions

The application will accept audio input, process audio through speech recognition model, display transcribed text, provide download option, and handle errors and invalid input.

2.3 User Classes

General User: Uploads audio and receives transcription.

Admin: Maintains server and monitors logs.

2.4 Operating Environment

Web browser, Python runtime server, and cloud hosting platform.

2.5 Constraints

Requires internet connection. Audio length limits. Processing time depends on file size.

2.6 Assumptions

Users have microphone access if recording live. Server has sufficient RAM for model inference.

3. Functional Requirements

3.1 Audio Input

System shall allow audio upload in WAV, MP3, and M4A formats and support live microphone recording.

3.2 Processing

System shall send audio to speech recognition model, preprocess audio, and generate transcription.

3.3 Output

System shall display transcription, allow copy to clipboard, and download as TXT file.

3.4 Error Handling

Invalid file types show error message. Processing failure shows retry option. Server timeout notifies user.

3.5 User Interface

Minimal UI with upload button, record button, output display, and download button.

4. Non-Functional Requirements

4.1 Performance

Audio under 30 seconds processed within 10 seconds. Support at least 20 concurrent users.

4.2 Reliability

System uptime >= 99%. Automatic retry for failed requests.

4.3 Security

Uploaded files deleted after processing. HTTPS encryption required.

4.4 Usability

UI should be simple and beginner-friendly. No login required for basic usage.

4.5 Scalability

Architecture must support scaling to multiple instances.

5. External Interface Requirements

5.1 User Interface

Upload audio button, record audio button, transcription output box, download transcript button, and status indicator.

5.2 Hardware Interface

Microphone optional for recording.

5.3 Software Interface

Python Flask backend, speech recognition model integration, and hosting service APIs.

6. System Architecture

Frontend: HTML, CSS, JavaScript

Backend: Flask server, audio processor, model inference module

```
User Audio -> Flask Server -> Audio Preprocessing -> Model -> Text -> Response -> UI
```

7. Data Requirements

Input Data: Audio file

Output Data: Text transcript

Temporary Storage: Uploaded audio files stored briefly and deleted after processing.

8. Deployment Requirements

Platform: Render

Environment variables: Model path, Temp directory

Required packages: Flask, Torch, Audio processing libraries

9. Future Enhancements

Multi-language transcription, speaker identification, real-time streaming, login system, transcript history, translation feature.

10. Acceptance Criteria

System is complete when audio uploads work, speech converts accurately, output displays within acceptable time, and deployment runs successfully online.

11. Risks and Mitigation

Slow processing -> Use smaller model

Large uploads -> Limit file size

Server overload -> Add queue system

12. Appendix

Recommended Folder Structure:

```
project/
■
■■■ app.py
■■■ templates/
■■■ static/
■■■ uploads/
■■■ requirements.txt
```