Tap-Vision

(As a part of ACTS Management System)

Functional Requirement Specification Document

**Functional Requirements Document:**

|  |  |
| --- | --- |
| **Name of Document** | Functional Requirement Specification of Accessible Text Magic Studio |
| **Author** | Aditya Kumar, Khushi Srivastava, Hrishikesh Kanade, Varad kodgire, Abhishek pal |
| **Description of Content** | Functional, Technical and Operational Requirements |
| **Reference** |  |
| **Approved by** | Aishwarya pandey |
| **Date of Approval** |  |

**Distribution:**

|  |  |
| --- | --- |
| **Copy No.** | **Distributed to** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

**Document Version Control:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Version Date** | **Prepared by** | **Reviewed by** | **Particulars/**  **Description for Change** | **Remarks** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**List of Requirements Elicitation Discussions:**

|  |  |  |
| --- | --- | --- |
| **SN.** | **Date** | **Participating Institutes/ Departments** |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |
| 4. |  |  |
| 5. |  |  |
| 6. |  |  |
| 7. |  |  |
| 8. |  |  |
| 9. |  |  |
| 10. |  |  |
| 11. |  |  |
| 12. |  |  |
| 13. |  |  |
| 14. |  |  |
| 15. |  |  |

|  |  |  |
| --- | --- | --- |
| 16. |  |  |
| 17. |  |  |
| 18. |  |  |
| 19. |  |  |
| 20. |  |  |

**Abbreviations:**

|  |  |
| --- | --- |
| **TTS** | Text-to-Speech |
| OCR | Optical Character Recognition |
| API | Application Programming Interface |
| **PDF** | Portable Document Format |
| DOCX | Document Open XML (Microsoft Word format) |
| gTTS | Google Text-to-Speech |
| SR | Speech Recognition |
| UX | User Experience |
| **SDK** | Software Development Kit |

|  |  |  |
| --- | --- | --- |
|  | **INDEX** |  |
| **Contents** |  | **Page. No** |
| **1. Introduction** |  | **1** |
| 1.1 Purpose |  | 1 |
| 1.2 Background |  | 1 |
| 1.3 Scope |  | 1 |
| 1.4 References |  | 1 |
| 1.5 Document Overview |  | 1 |
| **2. Functional Requirements** |  | **2** |
| 2.1 User Management |  | 2 |
| 2.1.1 Description |  | 2 |
| 2.1.2 Functional Requirements |  | 2 |
| 2.1.3 Field Validations |  | 2 |
| 2.2.2 Functional Requirements |  | 2 |
| 2.2.3 Field Validations |  | 3 |
| 2.3.1 Description |  | 3 |
| 2.4 Reports and Analytics |  | 4 |
| 2.4.1 Description |  | 4 |
| * + 1. Functional Requirements |  | 4 |
| 3 [External Interfaces 4](#_TOC_250001) Pre-Requisites, Assumptions, and Dependencies |  | 4  4 |

[External Interfaces 4](#_TOC_250001)

1. [Pre-Requisites, Assumptions, and Dependencies 4](#_TOC_250000)

1. **Introduction**
   1. **Purpose**

This application extracts text from a variety of file formats and URLs, including PDFs, Word documents, ePub files, plain text, and images. The extracted text can be converted into speech, using either an online or offline Text-to-Speech (TTS) engine, depending on the user's internet availability.

**1.2 Background**

The **Text Extraction and Text-to-Speech Application** extracts text from various file formats (PDF, DOCX, images) and converts it into speech. It supports OCR, web scraping, and voice command features, enhancing accessibility and convenience. Built with Python libraries, it offers both online and offline text-to-speech functionality for diverse user needs.

* 1. **Scope**

The scope of the **Text Extraction and Text-to-Speech Application** includes

**Text Extraction :** Extract text from various file formats (PDF, DOCX, TXT, EPUB, images via OCR)Web scraping for text extraction from URLs.

* **Text-to-Speech (TTS):** Convert extracted text to speech using online (gTTS) and offline (pyttsx3) methods.

 **Voice Command Integration:** Recognize speech commands to trigger text-to-speech conversion.

* **Accessibility :** Designed for users with visual impairments or those who prefer auditory content consumption.
* **Multi-format Support :** Handles a wide range of document formats and media types for flexible content interaction.
* **Offline and Online Functionality :** Works both with and without internet access, providing seamless usage in different environments

**1.4 References**

* Documentation: https://docs.streamlit.io/
* File Upload: https://docs.streamlit.io/library/api-reference/widgets/st.file\_uploader

**2. Optical Character Recognition (OCR) with Tesseract**

* pytesseract Documentation: <https://pypi.org/project/pytesseract/>
* Tesseract OCR Installation Guide: <https://github.com/tesseract-ocr/tesseract>

**3. Text Extraction from Files**

* PyMuPDF (for PDF text extraction): <https://pymupdf.readthedocs.io/en/latest/>
* python-docx (for Word documents): <https://python-docx.readthedocs.io/en/latest/>
* ebooklib (for ePub files): <https://pypi.org/project/EbookLib/>

**4. Web Scraping with BeautifulSoup**

* BeautifulSoup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
* Handling HTTP Errors: https://docs.python-requests.org/en/latest/user/quickstart/#errors-and-exceptions

**5. Speech Recognition with Google Speech API**

* SpeechRecognition Library: <https://pypi.org/project/SpeechRecognition/>
* Microphone Input Handling: https://realpython.com/python-speech-recognition/

**6. Language Translation with Helsinki-NLP Models**

* MarianMTModel (Hugging Face): https://huggingface.co/Helsinki-NLP
* Transformers Library: https://huggingface.co/docs/transformers/index

**7. Text-to-Speech (TTS)**

* gTTS (Google Text-to-Speech): <https://pypi.org/project/gTTS/>
* pyttsx3 (Offline TTS): <https://pypi.org/project/pyttsx3/>

**8. Socket Programming (Internet Connection Check)**

* Python Socket Module: <https://docs.python.org/3/library/socket.html>
  1. **Document Overview**

This document describes the functional aspects of the Text Extraction and Text-to-Speech Application. It provides details about the system's capabilities, requirements, and validation criteria. The document is structured to offer a clear understanding of how the system processes user input, extracts text from various sources, and converts it into speech. Additionally, it includes functional and field validation tables that outline expected system behaviors. References to external sources and libraries used in the implementation are provided for further study and verification.

# Functional Requirements

* 1. **User Management**
     1. **Description**

The application allows users to upload a file or enter a URL to extract text. The extracted text can be translated and converted into speech using AI-based models. Users can interact via the UI or voice commands.

* + 1. **Functional Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **SN.** | **Functionality** | **Process** | **Remarks** |
| FR 1.1 | File Upload | User uploads PDF, DOCX, EPUB, TXT, or image | Supports OCR for image files |
| FR1.2 | Text Extraction | Extracts text from uploaded files | Uses PyMuPDF, Python-Docx |
| FR1.3 | Extract Text from Web Pages | Fetches webpage content using requests and parses text using BeautifulSoup   |  | | --- | |  | | Displays error for blocked websites (403 error) |
| FR1.4 | Speech Recognition (Voice Commands) | Listens to user commands using speech\_recognition | Recognizes "convert to speech" and language selection |
| FR 1.5 | Language Translation | Uses MarianMT Transformer models for translation | Supports English, Hindi, French, and German |
| FR 1.6 | Text-to-Speech (Online Mode - gTTS) | Converts text to speech using Google TTS | Requires internet connection |
| FR 1.7 | Text-to-Speech (Offline Mode - pyttsx3) | Converts text to speech using pyttsx3 (offline engine) | Works without an internet connection |
| FR 1.8 | Audio Playback | Plays the generated speech file using Streamlit audio playe | Supports MP3 format |
| FR 1.9 | Error Handling for Unsupported Formats | Displays an error message for unsupported file types | Simple and user-friendly design |

* + 1. **Fields Validations**

This table lists the various fields that will be there in the form for the requirement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Field Name** | **Field Description** | **Validations** | **Remarks** |
| 1 | File Upload,  URL Input, | Upload a file for text extraction | Supported formats: PDF, DOCX, EPUB, TXT, JPG, JPEG, PNG | |  |  | | --- | --- | |  | Displays error  for unsupported  formats | |
| 2 | Speech Command | |  |  | | --- | --- | |  | Recognizes voice  commands for TTS | | Recognizable phrases like "convert to speech | |  |  | | --- | --- | |  | Warns on  Unrecognized  commands | |
| 3 | Language Selection | Target language for translation | Supported: en, hi, fr, de | |  | | --- | | Defaults to  English for  invalid entries |  |  | | --- | |  | |
| 4 | File Uploader | Upload component for files | |  |  | | --- | --- | |  | File format validation | | Shows upload progress |
| 5 | Network Check | |  |  | | --- | --- | |  | Verifies internet  availability | | Socket connection timeout | Fallback to offline mode |

**Reports and Analytics**

**User Interaction Analytics**

* **File Uploads**: Track the number and types of files uploaded by users to better understand the most common use cases.
* **Language Preferences**: Analyze the most selected languages for translation and TTS to enhance future versions of the application with language support.

**Performance Metrics**

* **Text Extraction Speed**: Measure the time taken to extract text from each file type (PDF, DOCX, etc.) and optimize processing time if needed.
* **TTS Processing Time**: Track how long it takes to convert text to speech using both online (gTTS) and offline (pyttsx3) engines, and ensure responsiveness.

**Error Tracking**

* Keep logs of errors encountered, such as unsupported file formats, failed translations, or unrecognized speech commands, to continuously improve the system's robustness.

# External Interfaces

# The Text Extraction and Text-to-Speech Application interfaces with multiple external components:

# User Interface (UI): Streamlit-based interface for file uploads, URL inputs, and voice commands.

# File System: Local system for processing PDF, DOCX, EPUB, TXT, and image files.

# Web Scraping: Extracts text from URLs using HTTP requests and BeautifulSoup.

# Translation Service: Utilizes Hugging Face MarianMT models for language translation.

# TTS Engines: Google TTS (online) and pyttsx3 (offline) for text-to-speech conversion.

# Speech Recognition: Recognizes voice commands for TTS actions.

# Internet Connectivity: Determines TTS engine based on available internet.

# Pre-Requisites, Assumptions, and Dependencies

The application requires Python 3.7+, libraries like Streamlit, pytesseract, pyttsx3, and gTTS, and Tesseract OCR for image text extraction. Users must have a microphone for voice input. Dependencies include internet access for translation and TTS services, and proper installation of necessary libraries and APIs for functionality.