System Requirements Specification (Revised)

1. Introduction

1.1 Purpose

This SRS outlines the requirements for the “Textify” system based on a more informed understanding of the current state of AI-enabled Software Engineering. Recognizing that modern AI solutions often rely on mature third-party APIs and iterative training workflows, the primary goal remains delivering robust text translation and OCR capabilities. However, the approach now includes a tighter focus on leveraging proven frameworks, emphasizing incremental releases, and ensuring that the team’s learning goals inform each development cycle.

1.2 Scope

The Textify application shall provide AI-driven text translation and text recognition (OCR), operating as a web-based platform accessible on both desktop and mobile browsers. Unlike the earlier vision, the revised scope narrows the core feature set to those that can be reliably implemented using existing AI APIs and frameworks. Experimental features such as advanced STT/TTS and export formats will be introduced gradually.

1.3 Definitions, Acronåyms, and Abbreviations

API: Application Programming Interface
OCR: Optical Character Recognition
NMT: Neural Machine Translation
TTS: Text-to-Speech
STT: Speech-to-Text
Export: Saving or sharing translation output (e.g., PDF, DOCX, TXT)

1.4 Overview

This document specifies functional and non-functional requirements for Textify’s initial release, emphasizing translation and OCR. Additional features such as TTS, STT, and robust export capabilities are considered longer-term goals. The development approach will be incremental and informed by user testing, performance metrics, and team skill evolution.

1.5 AI-driven Software Engineering Landscape

AI-enabled software engineering has transformed application development by democratizing access to machine learning (ML) and artificial intelligence (AI) through pre-trained models, public APIs, and open-source frameworks. This allows developers to integrate AI without building models from scratch.

Key Trends:

1. API-driven Development:
   Leading AI providers (Google Cloud, Microsoft Azure, AWS) offer AI services (e.g., translation, OCR, speech synthesis) via APIs, accelerating development and enabling teams to focus on UX and customization.

2. Pre-trained Models and Transfer Learning:
   Pre-trained models, fine-tuned for specific use cases, reduce the need for large datasets and computational resources, enhancing Textify's capabilities with minimal overhead.

3. Iterative and Agile Development:
   Incremental rollouts refine features based on feedback, allowing core functionalities (translation, OCR) to stabilize before advanced features (TTS, STT) are added.

4. Modularity and Microservices:
   Modular designs enable independent updates or replacements, promoting scalability and maintainability.

5. Ethical AI and Privacy:
   Compliance with regulations (e.g., GDPR) ensures secure data handling through encryption and anonymization. Textify prioritizes user consent and data protection.

6. Low-code/No-code Solutions:
   Low-code platforms simplify AI integration, enabling rapid deployment without compromising performance.

Strategic Implications for Textify:

- Reduced Complexity: External APIs streamline development.
- Scalability: Modular integration facilitates feature expansion.
- Continuous Adaptation: Iterative releases drive improvements.
- Competitive Edge: Leveraging AI trends ensures responsiveness and market readiness.

Textify’s strategy focuses on proven AI technologies to deliver reliable, scalable, and user-friendly translation and OCR solutions.

2. General System

2.1 Product Overview

Textify will serve as a machine translation and OCR platform accessible via a standard web browser. The system's initial release will focus on:

High-quality translation via a well-established NMT API.
Accurate OCR leveraging a mature OCR library or API.
Subsequent releases will integrate TTS and STT as these capabilities become stable and the team gains the necessary expertise.

2.2 Roles and Responsibilities

Customer: Defines high-level requirements, provides feedback on iterative releases.
Manager: Oversees timelines, manages risks, and facilitates iteration planning.
Designer: Crafts a user interface that is both intuitive and adaptable, integrating user feedback into each design iteration.
Programmers: Implement modular, API-driven solutions, allowing for easier updates and integration of third-party AI services.
Testers: Continuously test new iterations of features, ensuring quality and identifying areas for model improvement, UI refinement, and performance enhancements.

3. Functional Requirements

3.1 Text Translation

Requirement: The system shall accept source text and translate it into a target language using a trusted NMT API.
Acceptance Criteria: Users enter text and select source/target languages. For standard paragraphs (<500 words), translations shall be returned within 2 seconds. Initial language support: English ↔ German, German ↔ English. Future languages introduced iteratively.

3.2 Optical Character Recognition (OCR)

Requirement: The system shall extract text from images using an OCR API with approximately 95% accuracy on standard printed fonts.
Acceptance Criteria:Users upload PNG or JPEG images. The system preprocesses images (de-skew, basic noise reduction) before OCR. Extracted text is displayed and can be translated on request.

3.3 Text-to-Speech (TTS) [Planned for Future Release]

Requirement (Future): Convert translated text into speech via a TTS API.
Acceptance Criteria (Future): Users can listen to the audio playback within the interface. Basic controls (play, pause) will be introduced once stable TTS models are chosen. (This feature will be evaluated after gathering user feedback on the core translation and OCR functionalities.)

3.4 Speech-to-Text (STT) [Planned for Future Release]

Requirement (Future): Convert uploaded speech files into text with approximately 90% accuracy in moderate noise conditions.
Acceptance Criteria (Future): Users upload an audio file, and receive a text transcription. Proper punctuation and capitalization handled by the STT API. (STT integration will be considered once stable, well-documented APIs have been tested.)

3.5 User Interface

Requirement: The UI shall be responsive, simple, and user-centered, reflecting iterative design improvements.
Acceptance Criteria:Core features (translation, OCR) are accessible within one click from the homepage. Pages shall load within 1 second on standard devices.

3.6 Export (Limited Initial Scope)

Requirement: The application shall provide a basic export function for translated or OCR-derived text in at least one common format (e.g., TXT).
Acceptance Criteria: Users can download the final text as a TXT file initially. Additional formats (PDF, Word) will be introduced after validating stability and user demand.

4. Non-Functional Requirements

4.1 Scalability

Requirement: The system shall handle multiple concurrent users.

4.2 Security & Privacy

Requirement: All data transmissions shall be encrypted (SSL/TLS). No unencrypted user data stored. Models may use data for refinement only if explicit user consent is given, aligning with current privacy standards and regulations (e.g., GDPR).

4.3 Maintainability & Modular Architecture

Requirement: The system shall be modular, enabling easy integration or replacement of third-party AI APIs as the technology evolves.
Acceptance Criteria:Each module (Translation, OCR, UI) has clear interfaces and minimal dependencies. Code is documented and version-controlled. Configuration and deployment details are centralized for straightforward updates.

4.4 Reliability

Requirement: The system shall include basic error detection, logging, and actionable error messages.
Acceptance Criteria: Logs include timestamps, error codes. Clear user-facing messages suggest next steps or possible resolutions. Automated tests ensure core functionality before each release.

4.5 Iterative Development and Learning Goals

Requirement: The team shall adopt an iterative approach, learning from each release and integrating user feedback and performance metrics into subsequent sprints.
Acceptance Criteria: Post-iteration reviews determine the next set of features or refinements. Team members share learnings—both technical (e.g., API performance) and non-technical (e.g., user interface improvements)—to continuously improve the product and process.

5. Use Cases

5.1 Use Case: Text Translation

Actors:

- **Primary:** User
- **Secondary:** Translation API

Description:
The user translates input text from a source language to a target language.

Preconditions:

- The user has access to the translation feature.
- A stable internet connection is available.

Postconditions:

- The translated text is displayed to the user.
- The translation is saved in the user's history.

Special Requirements:

- Support for at least 20 languages.
- Translation time must not exceed 2 seconds for texts under 500 words.

5.2 Use Case: OCR (Image-to-Text)

Actors:

- **Primary:** User
- **Secondary:** OCR Engine

Description:
The user uploads an image from which text is extracted.

Preconditions:

- The user has access to the OCR feature.
- The uploaded image format is supported (e.g., PNG, JPEG).

Postconditions:

- The extracted text is displayed to the user.
- The extracted text is saved in the user's history.

Special Requirements:

- Minimum text extraction accuracy of 95% for standard fonts.
- Support for various image formats (PNG, JPEG).

5.3 Use Case: Export Translation

Actors:

- **Primary:** User
- **Secondary:** Export Module

Description:
The user exports the translated or extracted text in various formats.

Preconditions:

- A translation or extracted text is available.
- The user has permission to export the data.

Postconditions:

- The exported file is successfully downloaded or shared.
- The export action is recorded in the user's history.

Special Requirements:

- Support for at least three export formats: PDF, Word (DOCX), TXT.
- Export time must not exceed 2 seconds.

5.4 Use Case: STT (Speech-to-Text)

Actors:

- **Primary:** User
- **Secondary:** STT Engine

Description:
The user transcribes uploaded or live audio recordings into text.

Preconditions:

- The user has access to the STT feature.
- The audio recording is in a supported format (e.g., MP3, WAV).

Postconditions:

- The transcribed text is displayed to the user.
- The transcription is saved in the user's history.

Special Requirements:

- Transcription accuracy of at least 90% in moderately noisy environments.
- Support for at least two audio formats: MP3, WAV.

5.5 Use Case: TTS (Text-to-Speech)

Actors:

- **Primary:** User
- **Secondary:** TTS Engine

Description:
The user converts input or extracted text into an audio file.

Preconditions:

- The user has access to the TTS feature.
- The input text is in a supported format and within the allowable length.

Postconditions:

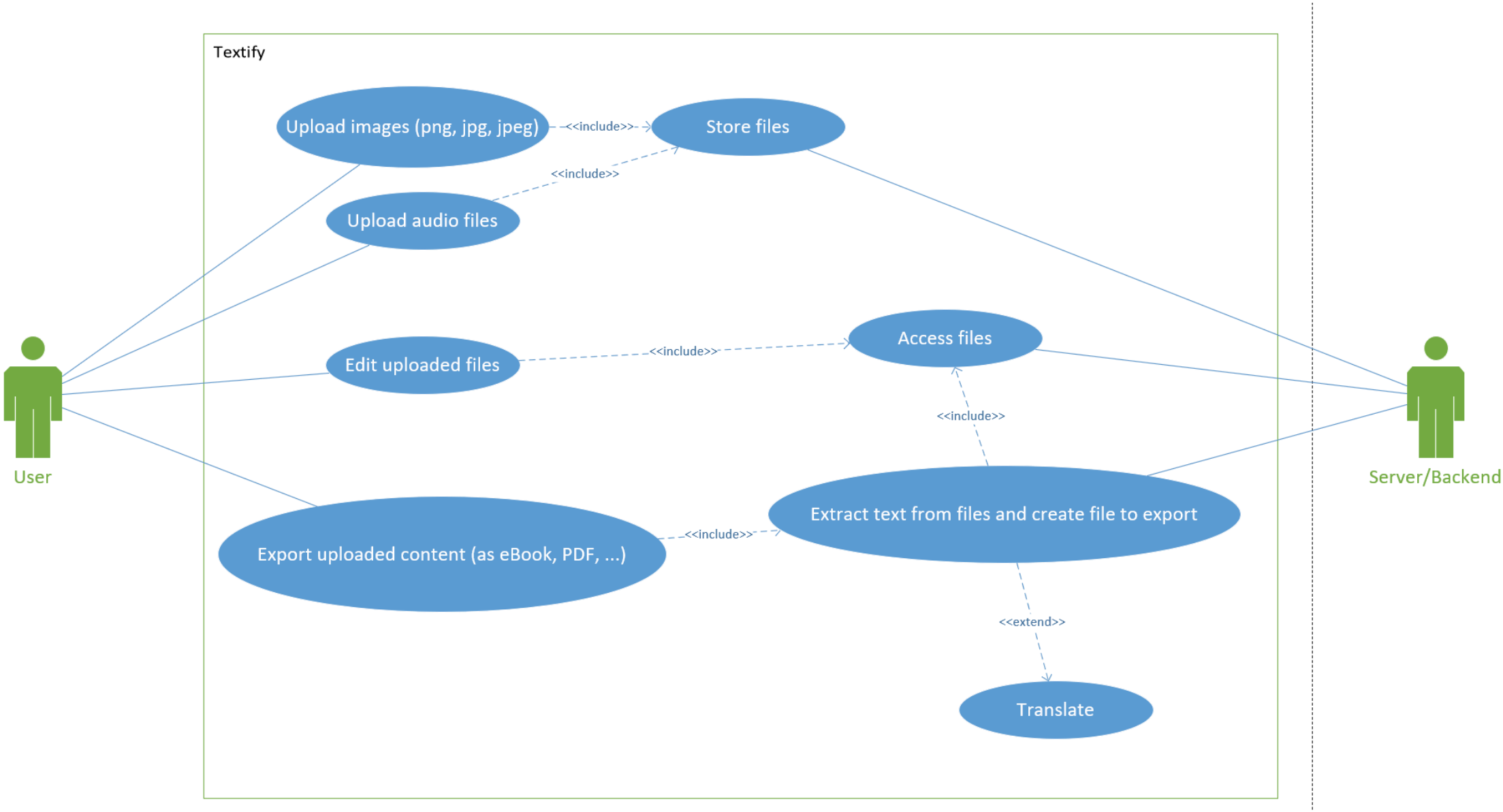
- The generated audio file is provided to the user (e.g., for playback or download).
- The audio file is saved in the user's history.

Special Requirements:

- Support for at least two audio formats: MP3, WAV.
- Adjustable settings for speed and volume.

(Additional use cases for TTS and STT will be defined once these features are near readiness.)

5.4 Use Case Diagram



6. System Design and Architecture

6.1 Architecture Overview

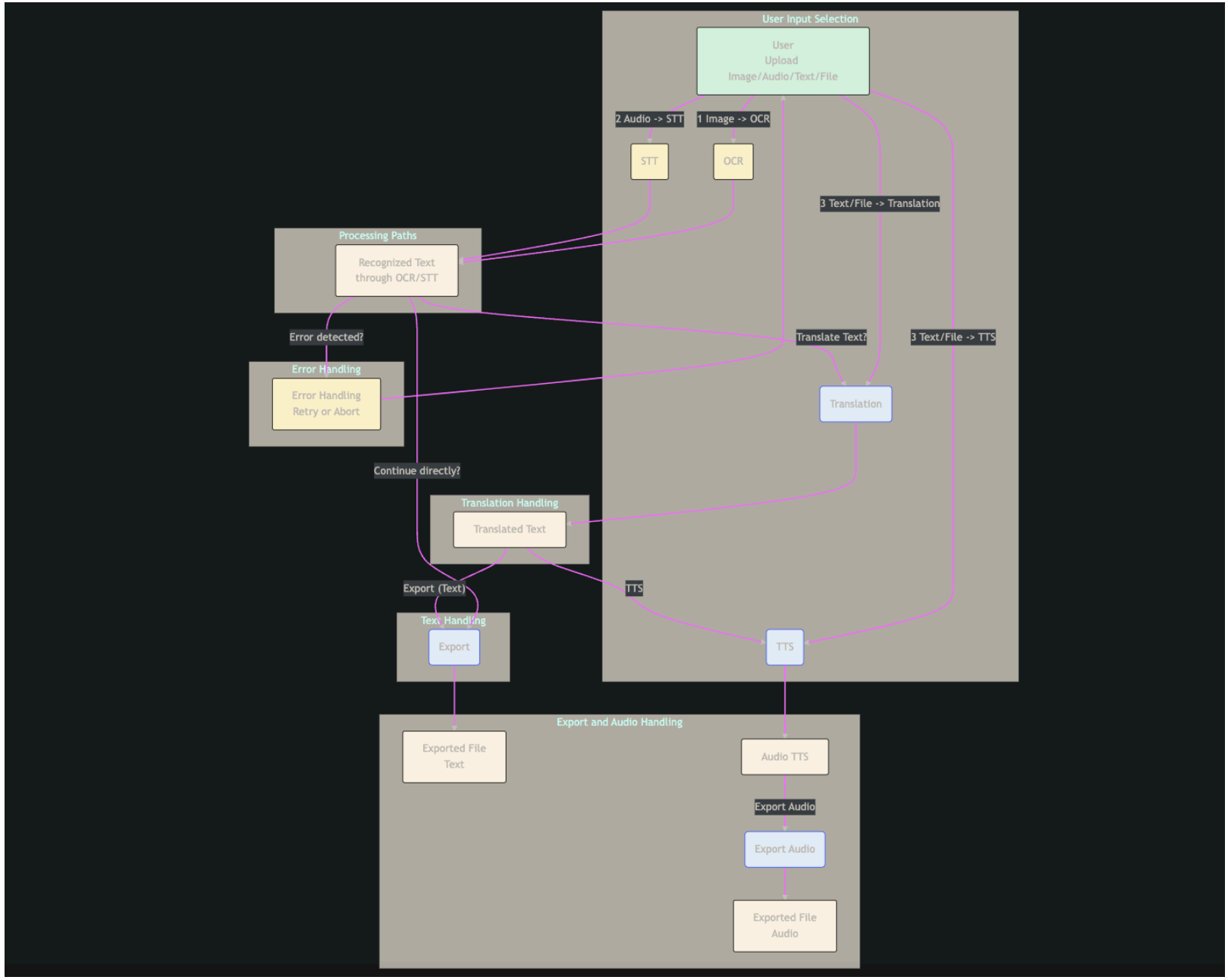
Frontend: Web-based UI, designed to be responsive and user-friendly.
Backend: API-driven approach. The backend acts as an orchestrator, calling external NMT and OCR services.
Database: For user preferences, logs, and optionally user translation histories (if user consents).

6.2 Revised Design Approach

Leverage mature AI APIs (e.g., Google Cloud Translation, Azure OCR) rather than building from scratch.
Incrementally add new services (TTS, STT, advanced export formats) as the team validates quality and performance in real-world conditions.
Focus on modularity to replace or upgrade services without major code rewrites.

Reflections on Revisions:

Focus and Scope: Reduced initial complexity to ensure a polished and maintainable core product (translation and OCR) before adding advanced features.
Teaming and Task Assignments: Clarified roles encourage iterative design and development. Cross-functional teaming improves as developers, designers, and testers continuously align goals.
Learning Goals: Prioritizing incremental releases allows the team to learn from each cycle, improving technical competency in using AI APIs, refining UI/UX, and building confidence in delivering stable features.
Approach: Instead of attempting a fully custom AI model, rely on proven APIs and frameworks. This approach is pragmatic, reduces time-to-market, and ensures the team learns modern integration techniques, leaving room to experiment with custom models in future iterations.


User Input Selection
User
Upload
Image/Audio/Text/File
2 Audio -> STT
1 Image -> OCR
STT
OCR
3 Text/File -> Translation
Processing Paths
Recognized Text
through OCR/STT
Error detected?
Translate Text?
3 Text/File -> TTS
Error Handling
Error Handling
Retry or Abort
Translation
Continue directly?
Translation Handling
Translated Text
Export (Text)
TTS
Text Handling
Export
TTS
Export and Audio Handling
Exported File
Text
Audio TTS
Export Audio
Export Audio
Exported File
Audio

7. Learning Goals

7.1 Technical Proficiency

- Mastering New Technologies – Learning new programming languages, frameworks, and tools.
- AI and Machine Learning – Understanding how to integrate AI models, work with APIs, and manage data pipelines.
- DevOps and Automation – Containerization (Docker, Kubernetes).

7.2 Software Architecture and Design

- Modular Design – Creating systems with interchangeable components for easy maintenance and scalability.
- Microservices – Designing distributed systems with loosely coupled services.
- API Design – Building and consuming REST services effectively.
- Performance Optimization

7.3 Agile and Iterative Development

- Agile Methodologies – Applying Kanban to improve project management.
- Iterative Development – Learning to release small, incremental improvements regularly to gather feedback.

7.4 Collaboration and Team Dynamics

- Code Reviews and Pair Programming – Improving code quality and knowledge sharing through peer reviews.
- Version Control Mastery – Becoming proficient with Git and other version control tools for effective collaboration.

7.5 Quality Assurance and Testing

- Automated Testing – Writing unit, integration, and end-to-end tests to ensure reliability.
- Performance Testing – Identifying bottlenecks and ensuring the system can handle load and stress.

7.6 Security Best Practices

- Secure Coding – Understanding vulnerabilities (e.g., SQL injection) and applying secure coding practices.
- Data Privacy and Compliance.

7.7 Soft Skills and Project Management

- Communication and Documentation – Writing clear documentation and communicating effectively within teams.
- Time Management – Estimating tasks accurately and prioritizing work.

7.8 AI and Software Engineering Intersection (Specific to AI-Enabled Projects)

- AI Model Integration – Understanding how to incorporate pre-trained models via APIs (e.g., OCR, NMT, TTS).
- Data Engineering – Handling data preprocessing, model fine-tuning, and evaluation.