1. Introduction

1.1 Purpose

This System Requirements Specification (SRS) outlines the requirements for the “Textify” system, developed by a small team of three students with limited resources. The primary goal is to deliver robust text translation and Optical Character Recognition (OCR) capabilities by leveraging locally integrated pre-trained models and open-source frameworks. This document ensures that the project remains feasible within the constraints of the team’s expertise and available hardware.

1.2 Scope

Textify will function as a web-based platform providing AI-driven text translation and OCR services, accessible via desktop and mobile browsers. The initial release focuses on core functionalities achievable with local pre-trained models. Advanced features such as Text-to-Speech (TTS), Speech-to-Text (STT), and additional export formats will be introduced incrementally based on stability and user feedback.

1.3 Definitions, Acronyms, 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 details the functional and non-functional requirements for the initial release of Textify, emphasizing translation and OCR. Future enhancements, including TTS, STT, and advanced export functionalities, are outlined as long-term objectives. The development strategy employs an iterative approach, incorporating user feedback, performance metrics, and the team’s evolving skills.

1.5 AI-driven Software Engineering Landscape

AI-enabled software engineering has revolutionized application development by providing access to pre-trained models, open-source frameworks, and local integration capabilities. This democratization allows developers to incorporate sophisticated AI functionalities without developing models from scratch.

Key Trends:

- **Local Integration of Pre-trained Models:** Utilizing open-source models and frameworks (e.g., Hugging Face, TensorFlow, PyTorch) to implement AI features without relying on external APIs.
- **Optimization for Limited Resources:** Adapting and optimizing pre-trained models to run efficiently on constrained hardware.
- **Iterative and Agile Development:** Enhancing functionalities based on feedback and stabilizing core features before adding advanced capabilities.
- **Modularity:** Facilitating scalability and maintainability through modular system designs.
- **Ethical AI and Privacy:** Ensuring compliance with data protection regulations (e.g., GDPR) through secure data processing.
- **Low-code/No-code Solutions:** Simplifying AI integration without extensive programming expertise.

Strategic Implications for Textify:

- **Reduced Dependencies:** Utilizing local models eliminates reliance on external API services.
- **Cost Efficiency:** Avoiding API usage fees through local processing.
- **Adaptability:** Enabling customization and fine-tuning of models to meet specific requirements.
- **Competitive Edge:** Staying current with AI trends ensures market readiness and responsiveness.

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

---

2. General System

2.1 Product Overview

Textify serves as a platform for machine translation and OCR, accessible through a standard web browser. The initial version will concentrate on:

- **High-Quality Translation:** Utilizing a locally integrated pre-trained NMT model.
- **Accurate OCR:** Employing a mature open-source OCR library.
- **Future Integration of TTS and STT:** Incorporating these features once they are stable and the team gains the necessary expertise.

2.2 Roles and Responsibilities

- **Customer:** Defines high-level requirements and provides feedback on iterative releases.
- **Project Manager:** Oversees timelines, manages risks, and facilitates iteration planning.
- **Designer:** Develops a user-friendly interface and integrates user feedback into each design iteration.
- **Developers:** Implement modular solutions using pre-trained models and open-source frameworks, ensuring easy updates and integration.
- **Testers:** Continuously test new feature iterations to ensure quality and identify improvement areas.

---

3. Functional Requirements

The functional requirements are tailored to utilize locally integrated pre-trained models, considering the team's limited hardware resources.

3.1 Text Translation

- **Requirement:** The system shall accept source text and translate it into a target language using a locally integrated pre-trained NMT model.
- **Acceptance Criteria:**
    - Users input text and select source/target languages.
    - Translations for standard paragraphs (<500 words) are returned within 5 seconds, depending on available hardware.
    - Initial language support: English ↔ German. Additional languages introduced iteratively.
    - Utilizes open-source frameworks like Hugging Face Transformers with models such as MarianMT.
    - Models are hosted locally and optimized to minimize storage and computational requirements.

3.2 Optical Character Recognition (OCR)

- **Requirement:** The system shall extract text from images using a locally integrated open-source OCR library, achieving approximately 95% accuracy on standard printed fonts.
- **Acceptance Criteria:**
    - Users upload PNG or JPEG images.
    - The system preprocesses images (e.g., de-skewing, noise reduction) before performing OCR using libraries like Tesseract.
    - Extracted text is displayed and can be translated upon request.
    - OCR processes are optimized to minimize runtime on limited hardware.

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

- **Requirement (Future):** Convert translated text into speech using a locally integrated pre-trained TTS model.
- **Acceptance Criteria (Future):**
    - Users can listen to audio playback within the interface.
    - Basic controls (Play, Pause) are introduced once stable TTS models are selected.
    - Utilizes open-source TTS models like Tacotron or WaveNet, optimized for local execution.

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

- **Requirement (Future):** Convert uploaded audio files into text with approximately 90% accuracy in moderately noisy environments.
- **Acceptance Criteria (Future):**
  - Users upload an audio file and receive a text transcription.
  - Proper punctuation and capitalization are handled by the STT model.
  - Utilizes open-source STT models like Whisper by OpenAI, optimized for local use.

3.5 User Interface

- **Requirement:** The UI shall be responsive, simple, and user-centered, with iterative design improvements.
- **Acceptance Criteria:**
  - Core features (translation, OCR) are accessible with a single click from the homepage.
  - Pages load within 2 seconds on standard devices.
  - UI adjustments reflect local processing latencies transparently (e.g., loading indicators).

3.6 Export (Limited Initial Scope)

- **Requirement:** The application shall provide a basic export function for translated or OCR-extracted text in at least one common format (e.g., TXT).
- **Acceptance Criteria:**
  - Users can download the final text initially as a TXT file.
  - Additional formats (PDF, Word) are introduced after validating stability and user demand.
  - Utilizes libraries like jsPDF or python-docx for export functionality.

3.7 Model Management

- **Requirement:** Manage and update the locally integrated pre-trained models efficiently.
- **Acceptance Criteria:**
  - Mechanisms for easy replacement or updating of models without extensive code changes.
  - Documentation of the models used and their versions.
  - Regular checks for model updates and timely integration as needed.

3.8 Resource Management

- **Requirement:** Efficiently utilize available hardware resources through model and process optimization.
- **Acceptance Criteria:**
  - Implement techniques such as model compression, quantization, or distillation to make models more resource-efficient.
  - Monitor system resources during model execution to prevent bottlenecks.
  - Adjust model parameters based on available hardware to balance performance and resource usage.

---

4. Non-Functional Requirements

4.1 Scalability

- **Requirement:** The system shall support a moderate number of concurrent users, tailored to the team’s limited hardware resources.
- **Acceptance Criteria:**
  - Scalability is designed for approximately 10 concurrent users.
  - Future scalability through optimization and resource augmentation is feasible.

4.2 Security & Privacy

- **Requirement:** All data transmissions shall be encrypted (SSL/TLS). No unencrypted user data will be stored. Data may only be used for model refinement with explicit user consent, adhering to current privacy standards and regulations (e.g., GDPR).
- **Acceptance Criteria:**
  - Implementation of SSL/TLS for all data transfers.
  - No storage of unencrypted user data.
  - Clear user consent mechanisms for data usage.

4.3 Maintainability & Modular Architecture

- **Requirement:** The system shall be modular to allow easy integration or replacement of pre-trained models and open-source libraries.
- **Acceptance Criteria:**
  - Each module (Translation, OCR, UI) has clear interfaces and minimal dependencies.
  - Code is well-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 user-friendly error messages.
- **Acceptance Criteria:**
  - Logs include timestamps and error codes.
  - User-facing messages suggest next steps or possible resolutions.
  - Automated tests ensure core functionality before each release.

4.5 Hardware Requirements

- **Requirement:** The system shall run effectively on the team’s available hardware resources.
- **Acceptance Criteria:**
  - Minimum requirements for development and testing environments (e.g., CPU, RAM, storage).
  - Specifications for the production server or chosen hosting platform.

    - Recommendations for local development setups to test performance in advance.

4.6 Software Dependencies

- **Requirement:** The system shall be built on specific software stacks and frameworks.
- **Acceptance Criteria:**
    - List of programming languages used (e.g., Python, TypeScript).
    - Enumeration of necessary frameworks and libraries (e.g., TensorFlow, PyTorch, React).
    - Versioning of dependencies to ensure compatibility.
    - Documentation of installation and configuration steps.

4.7 Deployment Strategy

- **Requirement:** The system shall be deployed efficiently and reliably.
- **Acceptance Criteria:**
    - Description of the deployment process (e.g., Docker/Docker-Compose).
    - Use of containerization technologies like Docker to simplify deployment.

4.8 Data Management

- **Requirement:** Effective management and storage of data.
- **Acceptance Criteria:**
    - Choice and structure of the database (e.g., SQL vs. NoSQL).
    - Handling of sensitive user data in compliance with privacy policies (e.g., GDPR).
    - Mechanisms for data anonymization and encryption.

---

5. Use Cases

The use cases have been adjusted to reflect the use of locally integrated pre-trained models, altering secondary actors and accounting for local processing.

5.1 Use Case: Text Translation

- **Actors:**
  - **Primary:** User
  - **Secondary:** Local pre-trained NMT model
- **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 (for initial model loading, if required).
- **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 5 seconds for texts under 500 words.
  - Local processing of translations without external API calls.

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

- **Actors:**
  - **Primary:** User
  - **Secondary:** Local pre-trained OCR model
- **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).
  - Local processing of OCR without external API calls.

5.3 Use Case: Export Translations

- **Actors:**
  - **Primary:** User
  - **Secondary:** Export module
- **Description:** The user exports translated or OCR-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.
  - Utilization of local libraries for file creation without external services.

5.4 Use Case: STT (Speech-to-Text) [Planned for Future Release]

- **Actors:**
  - **Primary:** User
  - **Secondary:** Local pre-trained STT model
- **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.
  - Local processing of transcription without external API calls.

5.5 Use Case: TTS (Text-to-Speech) [Planned for Future Release]

- **Actors:**
  - **Primary:** User
  - **Secondary:** Local pre-trained TTS model
- **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.
  - Local processing of TTS without external API calls.

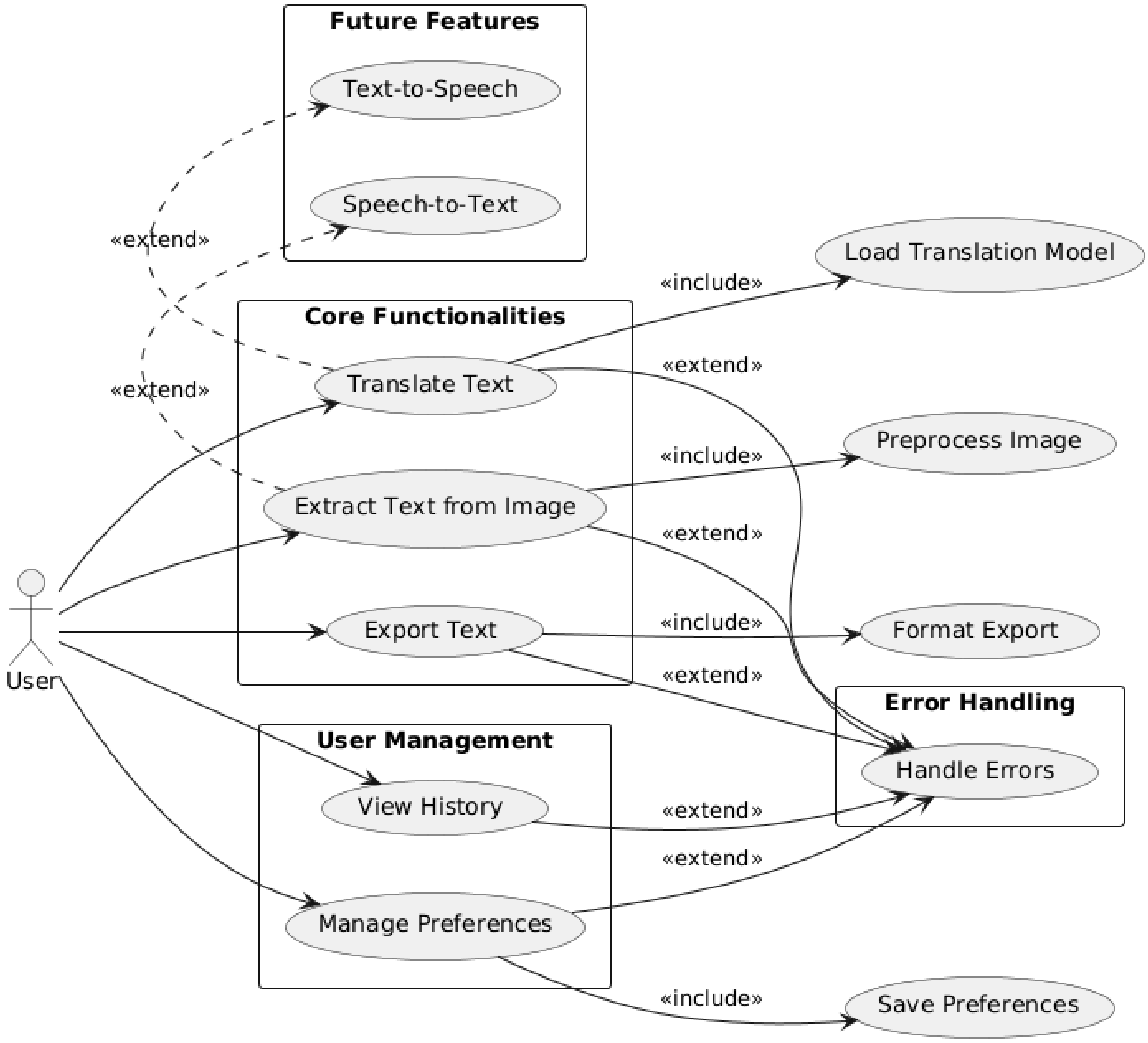
5.6 Use Case: View Translation and OCR History

- **Actors:**
  - **Primary:** User
  - **Secondary:** System Database
- **Description:** The user views a history of their past translations and OCR results.
- **Preconditions:**
  - The user has performed translations or OCR operations previously.
  - User consent has been given to store history data.
- **Postconditions:**
  - A list of past translations and OCR results is displayed to the user.
  - The user can select individual entries to view detailed results.
- **Special Requirements:**
  - History should be accessible from the main interface.
  - Users can search, filter, or sort their history entries.
  - Option to delete individual history entries or clear all history.

5.7 Use Case: Handle Errors and Provide Feedback

- **Actors:**
  - **Primary:** User
  - **Secondary:** System
- **Description:** The system detects errors during processing and provides appropriate feedback to the user.
- **Preconditions:**
  - An error occurs during text translation, OCR, export, STT, or TTS operations.
- **Postconditions:**
  - The user is informed of the error with a clear and actionable message.
  - The system logs the error details for further analysis.
- **Special Requirements:**
  - User-friendly error messages that avoid technical jargon.
  - Suggestions for resolving common issues (e.g., unsupported file format, network issues).
  - Logging of error details with timestamps and error codes for troubleshooting.

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


Future Features
Text-to-Speech
Speech-to-Text
«extend»
«extend»
Load Translation Model
Core Functionalities
«include»
Translate Text
«extend»
Preprocess Image
«include»
Extract Text from Image
«extend»
User
Export Text
«include»
Format Export
«extend»
Error Handling
User Management
Handle Errors
View History
«extend»
«extend»
Manage Preferences
«include»
Save Preferences

6. System Design and Architecture

6.1 Architecture Overview

- **Frontend:** Web-based user interface, designed to be responsive and user-friendly.
- **Backend:** Integration of pre-trained models and open-source libraries. The backend handles requests and executes AI functions locally.
- **Database:** Stores user preferences, logs, and optionally user translation histories (with user consent).

6.2 Revised Design Approach

- **Local Utilization of Pre-trained Models:** Employ open-source models (e.g., Hugging Face for NMT, Tesseract for OCR) to implement core functionalities.
- **Incremental Integration of New Services:** Add TTS and STT functionalities after validating their quality and performance under existing hardware conditions.
- **Optimization for Limited Resources:** Adapt and optimize models to run efficiently on available hardware resources.
- **Focus on Modularity:** Ensure easy replacement or updating of models and libraries without extensive code modifications.

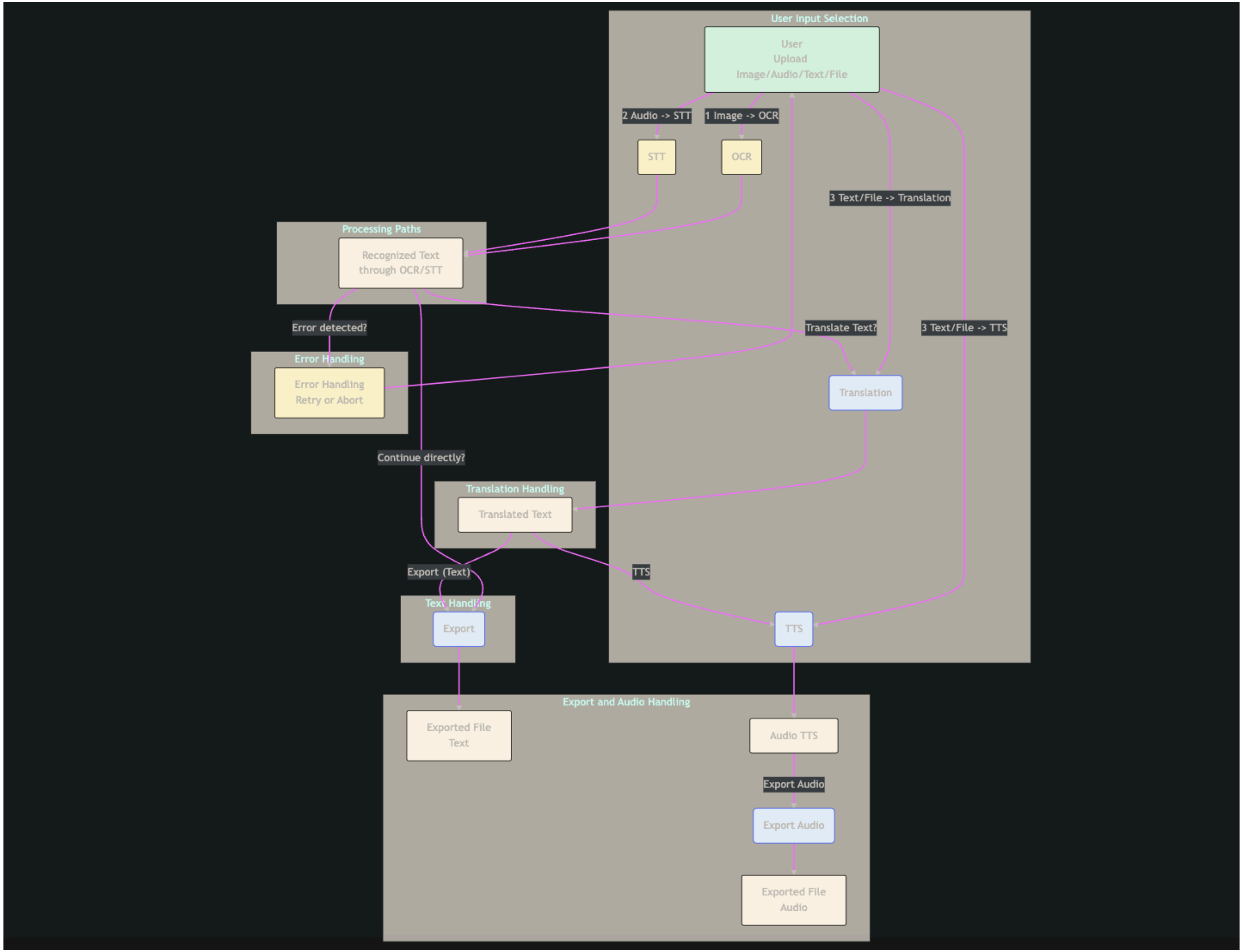
6.3 Reflections on Revisions:

- **Focus and Scope:** Reduced dependency on external services by utilizing local models, ensuring independence and cost efficiency.
- **Teamwork and Task Distribution:** Clear roles promote iterative design and development. Cross-functional teamwork improves through continuous alignment among developers, designers, and testers.
- **Learning Goals:** Prioritizing iterative releases allows the team to learn from each cycle, enhancing technical skills in handling pre-trained models, optimizing UI/UX, and ensuring stability.
- **Approach:** Leveraging proven pre-trained models and open-source frameworks instead of developing custom APIs simplifies the development process and allows the team to focus on modern integration techniques.

6.4 ADR’s for technological Decisions

See https://github.com/platofan23/textify/tree/Documents/ADR's

(These are written as MD-Files as is better readable)


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 Competence

- **Mastering New Technologies:** Learning new programming languages, frameworks, and tools.
- **AI and Machine Learning:** Understanding the integration of pre-trained models, working with open-source libraries, and managing data pipelines.
- **DevOps and Automation:** Utilizing containerization technologies (e.g., Docker).

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:** Effectively creating and consuming REST services.
- **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:** Enhancing 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 system performance under load.

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:** Creating clear documentation and communicating effectively within the team.
- **Time Management:** Accurately estimating tasks and prioritizing work.

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

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

---

8. Complementary Notes

- **Prototyping and Feedback Loops:** Creating early prototypes and regularly obtaining feedback from potential users can help improve usability and functionality continuously.
- **Performance Monitoring:** Implementing monitoring tools to track system performance and detect issues early in the production environment.
- **Team Communication and Collaboration:** Utilizing tools for effective team communication and collaboration (e.g., Slack, Trello, GitHub Issues) to enhance workflows and ensure all team members are aligned.