# Requirements Document – Phase 2

## Project Name: Speech2SpeechBuffer

### Team Number: 44

### Mentors: Satish Kathirisetti (Product Labs, IIITH), Nikunj (Mentor)

## 1. Introduction

This document outlines the Phase-2 requirements of the Speech-to-Speech System, focusing on integrating dedicated Python servers for reliable API communication with sandbox services (ASR, MT and TTS). These servers manage translation and synthesis through efficient API handling with retries and error control. Additionally, a buffering mechanism ensures smooth playback by maintaining 4-5 audio chunks, while Persistent Messages and Durable queue management protects against data loss and system failures.

## 2. System Requirements

### 2.1 Software Requirements

- **Operating System:** Platform Independent (Mac/Linux/Windows)
- **Development Environment:** VS Code, GitHub
- **Programming Language:** Python
- **Message Queue System:** RabbitMQ
- **Python Libraries:**
    - Pika (RabbitMQ communication)
    - FastAPI (API framework)
    - Asyncio (asynchronous processing)
    - Regex (re) (pattern matching)
    - JSON (data handling)

- Logging (error monitoring)
- Collections (data structures)
- OS (file handling)
- requests (Api calls)

# 3. User Profiles

## 3.1 System Administrators (Audio Engineers & Developers)

- **Usage Mode:** Managing real-time audio processing in applications.
- **Technical Familiarity:** Experienced in RabbitMQ, FastAPI, and audio processing.

## 3.2 End Users (General Public & Interpreters)

- **Usage Mode:** Receiving processed audio output.
- **Technical Familiarity:** Basic; primarily relies on system performance.

# 4. Feature Requirements (Phase 2)

## 4.1 System Features (Phase 2)

| Use Case No. | Feature Name | Description |
|---|---|---|
| UC-9 | Error Recovery | Handle **system failures, internet issues, malformed data, and timeouts**. |
| UC-10 | Data Persistence | Ensure **no data loss** during processing through Persistent Messages and Durable queues. |

## 4.2 Use Cases Implemented in Both Phase 1 & Phase 2

| Use Case No. | Feature Name | Description |
|---|---|---|

| | | |
|---|---|---|
| UC-2 | Error Monitoring | Access error logs, view system alerts, and manage failed message retries. |
| UC-4 | Server Management | Run/Stop any Python server at any time. |
| UC-11 | API-Managment | Processes messages from the input queue via API calls to sandbox services (MT, TTS) and forwards responses to the output queue. |

### 4.3 General Features Implemented in Phase 2

| Feature No. | Feature Name | Description |
|---|---|---|
| GF-2 | Reliability | Ensure **continuous operation** even during partial system failures. |
| GF-3 | Error Handling | Handle **system errors and edge cases** effectively. |

### 4.4 Technical Features Implemented in Phase 2

| Feature No. | Feature Name | Description |
|---|---|---|
| TF-1 | FastAPI Integration | High-performance **async API handling**. |
| TF-3 | System Integration | Integration with **ASR, MT, and TTS services**. |

**5. Use Case Descriptions**

## 5.1 UC-11: API Management

- **Actors:** Backend System
- **Pre-condition:** Input message is available in the RabbitMQ queue, and sandbox services (ASR, MT, TTS) are accessible.
- **Flow:**

    - The system reads a message from the input queue.
    - Sends the message to the sandbox API (ASR, MT or TTS) via an API call.
    - Waits for the response from the sandbox API.
    - Receives the processed message from the sandbox API.
    - Pushes the processed message to the output queue.
- **Alternative Flow:**
    - If the sandbox API does not respond within the specified timeout, the system retries the request.
    - If the sandbox API fails or returns an error, the system logs the error and moves the message to an error handling flow.
- **Post-condition:** The processed message is successfully sent to the output queue, or an error is logged if processing fails.

## 5.2 UC-2: Error Monitoring

- **Actors:** System Administrators
- **Pre-condition:** The system is running.
- **Flow:**
    - The system continuously logs errors into an error file on the admin's local device.
    - The administrator manually opens the error log file to review recorded errors.
    - The admin analyzes the logged errors and takes appropriate action (e.g., resolving issues or retrying failed processes).
- **Post-condition:** Errors are logged, reviewed, and resolved.

## 5.3 UC-4: Server Management

- **Actors:** System Administrators
- **Pre-condition:** Python servers are deployed and registered in the system.
- **Flow:**
    - The concerned authority logs into the device where the target server is running.
    - Navigates to the directory containing the server code or executable.
    - Uses the appropriate command to start or stop the server process.
- **Alternative Flow:**

    - If the device is inaccessible (e.g., due to network issues or permissions), the authority must first gain access before proceeding.
- **Post-condition:** Selected servers are successfully started or stopped.

## 5.4 UC-9: Error Recovery

- **Actors:** Backend System
- **Pre-condition:** System is running, and error logging is active.
- **Flow:**
    - The system detects an error during processing.
    - Logs the error and retries the failed task.
    - If recovery fails, continuous retries are attempted.
- **Post-condition:** Errors are resolved, retried, or logged for review.

## 5.5 UC-10: Data Persistence

- **Actors:** System Administrators, Backend System
- **Pre-condition:** Backup file is configured and accessible.
- **Flow:**
    - Backup files are created when the server starts.
    - The current translation session is stored in the backup file.
    - In case of queue deletion, data is restored from backups.
- **Post-condition:** Data is successfully saved or retrieved.

# 6. Conclusion

Phase-2 enhances the system with robust API handling for sandbox services (ASR, MT and TTS), continuous audio delivery through buffering and reliability even in case of message processing failures. These improvements ensure seamless, real-time speech processing with high system stability and minimal interruptions.