Software Requirements Specification (SRS) Document

Team Number - 44
Project Name - Speech2SpeechBuffer
Project Mentors – Satish Kathirisetti, Product Labs, IIITH (client), Nikunj (Mentor)

Team Members:

1. Akash Sigullapalli (2023101104)
2. Varun Alokam (2023101046)
3. Arya Mahendrakar (2023101041)
4. Karthik Malavathula (2023111025)
5. Prashanth (2023115006)

Brief Project Description:

- **Real-Time Audio Processing System Using RabbitMQ Queues**

This project involves creating a real-time audio processing backend system using RabbitMQ for queue-based communication and using Fast API for efficiency. We also need to ensure reliability and minimize disruptions, a buffering mechanism is implemented to maintain continuous audio delivery, even if system failures occur.

System requirements:

- Operating System: Platform independent (Can be Mac, Linux or Windows).
- Development Environment – VS code, GitHub
- Programming language:

    1. Backend – Python.
    2. EMQ – Rabbit MQ.

- Python Libraries Used:

    1. Pika
    2. FastAPI
    3. Asyncio
    4. Regex(re)
    5. Json
    6. Logging
    7. Collections
    8. Os (file handling)

9. requests

Users profile:

System Administrators (Audio Engineers & Developers)

- **Mode of Use:** Integrating real-time audio processing into applications or workflows.
- **Technical Familiarity:** Experienced with audio processing, RabbitMQ and FastAPI.

End Users (General Public & Non-Technical Users)

- **Mode of Use:** Engaging with processed audio through integrated applications such as music streaming, podcasts, or live translations.
- **Technical Familiarity:** Basic; minimal direct interaction with the system but relies on its performance for an enhanced experience.

- Our application mainly focuses on the backend part of a Speech-to-Speech Translation machine so we do not exactly know any information regarding who will be using this as of now.
- We were told “Subashini Company” will be using this as their backend part in the Speech-to-Speech Translation Machine.

Feature requirements (described using use cases):

1. Features for System Administrators

No.	Use Case Name	Description	Release
UC-1	Queue Management	Monitor and manage RabbitMQ queues, view queue statuses.	R1
UC-2	Error Monitoring	Access error logs, view system alerts, and manage failed message retries	R1, R2
UC-3	System Configuration	Configure buffer sizes and queue settings	R1, R2
UC-4	Server Management	Run/Stop any python server at any time.	R1, R2

2. Features for End Users (General Public, Interpreters/Professional Users)

No.	Use Case Name	Description	Release
UC-5	Continuous Translation	Receive translated audio output without significant interruption	R1
UC-6	Extended Sessions	Handle long-duration translation sessions with stable performance	R2

3. System Features (Backend)

No.	Use Case Name	Description	Release
UC-7	Buffer Management	Maintain 4-5 chunks buffer for smooth playback	R2

UC-8	Queue Processing	Process audio chunks through ASR→MT→TTS pipeline	R1
UC-9	Error Recovery	Handle system failures, Internet Issues, Malformed Data and Timeouts	R1, R2
UC-10	Data Persistence	Ensure no data loss by using Persistent Messages and Durable queues.	R1, R2
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.	R2

4. General Features

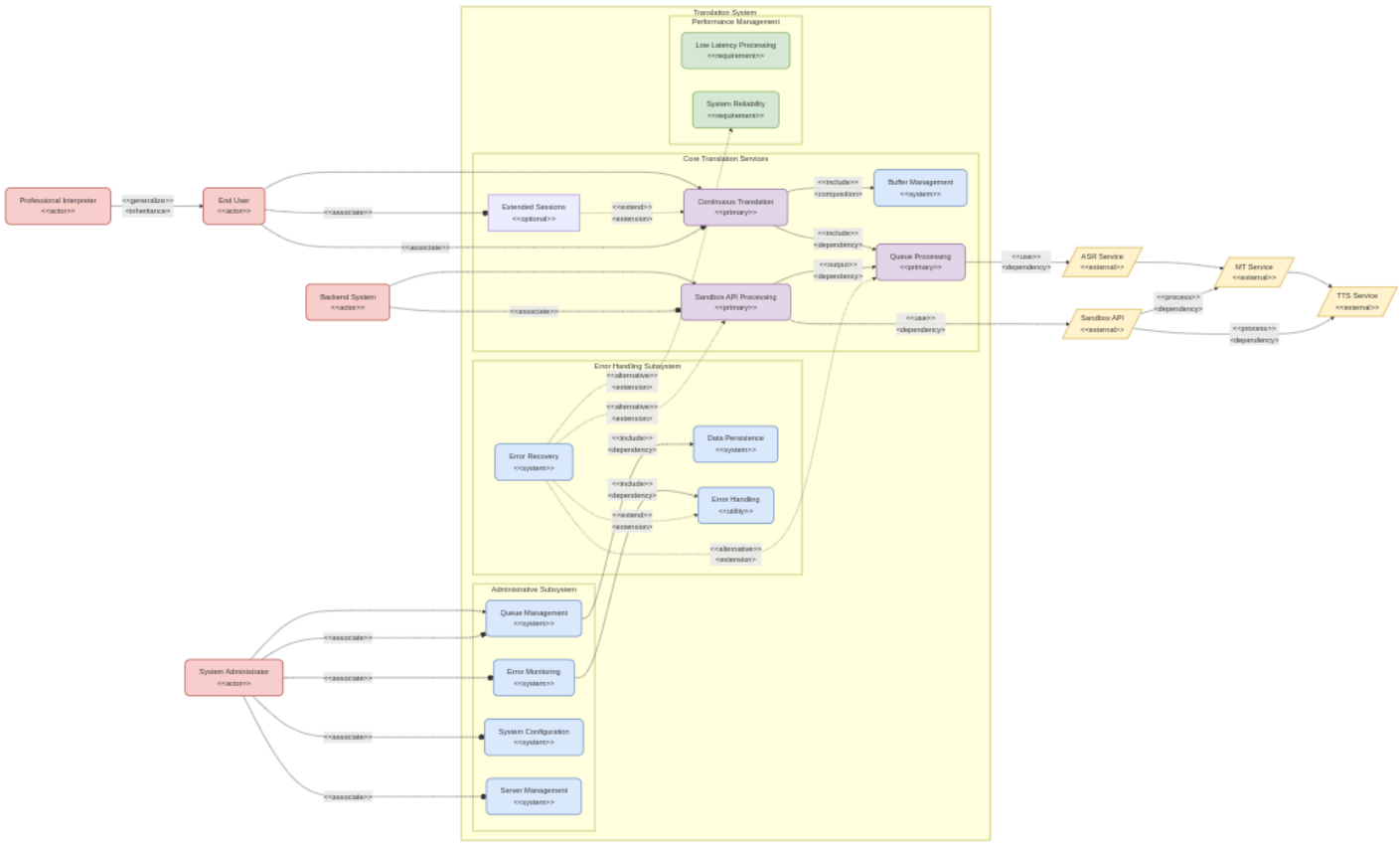
No.	Feature Name	Description	Release
GF-1	Low Latency Processing	Maintain minimal delay in translation pipeline	R1
GF-2	Reliability	Ensure continuous operation even during partial system failures.	R1, R2
GF-3	Error Handling	Handling of system errors and edge cases	R1, R2

5. Technical Features

No.	Feature Name	Description	Release
TF-1	FastAPI Integration	High-performance async API handling	R1, R2
TF-2	RabbitMQ Implementation	Reliable message queue system	R1
TF-3	System Integration	Integration with ASR, MT, and TTS services	R2

Use case diagram:

[Image link]



Use case description:

Use Case: UC-1 Queue Management

Use Case Number	UC-1
Use Case Name	Queue Management
Overview	Monitor and manage RabbitMQ queues, view queue statuses.
Actors	System Administrators
Pre-condition	Rabbit MQ server is running
Flow	**Main Flow**: 1. The Administrator logs into the dashboard. 2. View the list of active queues and their current statuses. 3. Manages the queue **Alternative Flow**: - If RabbitMQ is down, the system displays an error message and logs the issue.
Post-condition	The selected queues are successfully monitored or managed.

Use Case: UC-2 Error Monitoring

Use Case Number	UC-2
Use Case Name	Error Monitoring
Overview	Access error logs, view system alerts, and manage failed message retries.
Actors	System Administrators
Pre-condition	The system is running
Flow	**Main Flow:** 1. The system continuously logs errors into an error file on the admin's local device. 2. The administrator manually opens the error log file to review recorded errors. 3. 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.

Use Case: UC-3 System Configuration

Use Case Number	UC-3
Use Case Name	System Configuration
Overview	Configure buffer sizes and queue settings.
Actors	System Administrators
Pre-condition	The server code is accessible to the admins
Flow	**Main Flow**: 1. The system administrator accesses the server code on their local machine or server. 2. Locates the section of the code responsible for buffer size or queue parameters. Modifies the necessary configuration values in the code. Runs the server again after all the changes. **Alternative Flow**: 1.If the modifications contain syntax errors or invalid values, the server may fail to start or behave unexpectedly. The admin must debug the code, correct the issues, and restart the server.
Post-condition	Buffer sizes and queue settings have been successfully updated.

Use Case: UC-4 Server Management

Use Case Number	UC-4
Use Case Name	Server Management
Overview	Run/Stop any Python server at any time.
Actors	System Administrators
Pre-condition	Python servers are deployed and registered in the system.
Flow	**Main Flow**: 1. The concerned authority logs into the device where the target server is running. Navigates to the directory containing the server code or executable. 2. 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.

Use Case: UC-5 Continuous Translation

Use Case Number	UC-5
Use Case Name	Continuous Translation
Overview	Receive translated audio output without significant interruption.
Actors	End Users (General Public, Interpreters)
Pre-condition	Audio input is being captured, and the translation pipeline is operational.
Flow	**Main Flow**: 1. The user streams or uploads audio input. 2. The system processes the input through the ASR→MT→TTS pipeline. 3. Translated audio is delivered to the user in real-time. **Alternative Flow**: - If translation fails, the system retries processing or notifies the user.
Post-condition	Translated audio is delivered seamlessly to the user.

Use Case: UC-6 Extended Sessions

Use Case Number	UC-6
Use Case Name	Extended Sessions
Overview	Handle long-duration translation sessions with stable performance.
Actors	End Users (General Public, Interpreters)
Pre-condition	System resources (CPU, memory) are sufficient for session requirements.
Flow	**Main Flow**: 1. The user initiates a long-duration translation session. 2. The system checks the current API usage against rate limits. 3. If API limited is reached system cannot make additional API calls until the time window resets. 4. The session continues with stable performance until the user ends it. **Alternative Flow**:
Post-condition	Long-duration sessions are completed without interruptions.

Use Case: UC-7 Buffer Management

Use Case Number	UC-7
Use Case Name	Buffer Management
Overview	Maintain 4-5 chunks buffer for smooth playback.
Actors	Backend System
Pre-condition	Audio processing pipeline is active.
Flow	**Main Flow**: 1. The system continuously processes audio chunks. 2. Maintains a buffer of 4-5 chunks for playback. 3. Pleays audio from the buffer while processing new chunks or during the time it takes for our system to recover after a failure.
Post-condition	Audio playback is smooth and uninterrupted.

Use Case: UC-8 Queue Processing

Use Case Number	UC-8
Use Case Name	Queue Processing
Overview	Process audio chunks through ASR→MT→TTS pipeline.
Actors	Backend System
Pre-condition	Input audio chunks are available in the RabbitMQ queue.
Flow	**Main Flow**: 1. The system picks audio chunks from the queue. 2. Processes chunks through the ASR→MT→TTS pipeline. 3. Outputs translated audio for playback or download.
Post-condition	Audio chunks are processed and ready for playback or delivery.

Use Case: UC-9 Error Recovery

Use Case Number	UC-9
Use Case Name	Error Recovery
Overview	Internet issues, malformed data, and timeouts.
Actors	Backend System
Pre-condition	The system is running, and error logging is being done.
Flow	**Main Flow**: 1. The system detects an error during processing. 2. Logs the error and retries the failed task. **Alternative Flow**: - If recovery fails, the system continuously retries the failed task.
Post-condition	Errors are resolved, retried, or logged for review.

Use Case: UC-10 Data Persistency

Use Case Number	UC-10
Use Case Name	Data Persistency
Overview	Ensure no data loss during processing through Persistent Messages and Durable queues.
Actors	Backend System
Pre-condition	None
Flow	**Main Flow**: - Even in Case of system failures, Messages and queues are restored. **Alternative Flow**: - None
Post-condition	Data and queues are successfully restored.

Use Case: UC-11 API Management

Use Case Number	UC-11
Use Case Name	API Managment
Overview	Process messages from the input queue by making API calls to sandbox services (ASR, MT, TTS), await the response, and forward the processed message to the output queue.
Actors	Backend System
Pre-condition	Input message is available in the RabbitMQ queue, and sandbox services (ASR, MT, TTS) are accessible.
Flow	**Main Flow:** 1. The system reads a message from the input queue. 2. Sends the message to the sandbox API (ASR, MT or TTS) via an API call. 3. Waits for the response from the sandbox API. 4. Receives the processed message from the sandbox API. 5. 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.