⚙ SQS Background Operations: Step-by-Step Execution Guide

This document provides a highly detailed, step-by-step breakdown of every background task run asynchronously by the 12 SQS queues in consumer.service.ts.

---

1. Python CUDA Result Queue (`pythonCUDAQueue` / `QUEUE_URL`)

This queue processes high-CPU calculations like video stitching, GPT transcript evaluation, and interview grading. It has a parallel processing batch size of `5`.

Operation 1.1: Roleplay Summary & Grading Process

- **Trigger**: Message contains the `roleplay` configuration object.
- **Step-by-step Logic**:
  1. The worker extracts the user's roleplay transcript, user audio duration, and bot configs.
  2. It sends a POST request to `${process.env.SELF_ENDPOINT}/ai-examiner/ai_summary_process`.
  3. **Inside Endpoint**: The endpoint retrieves the ideal conversation template from MongoDB. It queries OpenAI/Gemini to analyze user responses against the guidelines, calculates scores for various competencies, and updates the final `RoleplayResult` document in MongoDB.
  4. The worker deletes the message from the queue on success.

Operation 1.2: Roleplay Video Generation

- **Trigger**: Message contains the `roleplayVideo` key.
- **Step-by-step Logic**:
  1. Invokes `${process.env.SELF_ENDPOINT}/chapters/generate-roleplay-video` via POST.
  2. **Inside Endpoint**: Gathers the user's speech audio recordings and visual slides. It spawns a background child process calling **FFmpeg** to merge the avatar slides with the compiled audio tracks, outputs a final `.mp4` video, and uploads the video file to AWS S3.

Operation 1.3: Interview Evaluation & Grading

- **Trigger**: Message contains `evaluationType === 'interview'`.
- **Step-by-step Logic**:
  1. The worker dispatches the request payload to `${process.env.SELF_ENDPOINT}/user-interview/interviewMainSqs`.
  2. **Inside Endpoint**: The endpoint evaluates the user's mock job interview. It grades answers based on professionalism and resume details, saves the assessment, and marks the user's interview status as `completed`.

Operation 1.4: Normal Training Evaluation

- **Trigger**: Message contains user scripts but is not an interview type.
- **Step-by-step Logic**:
  1. POST request is made to `${process.env.SELF_ENDPOINT}/user-interview/interviewSqs`.
  2. **Inside Endpoint**: Compares user responses against training module criteria, logs evaluation outcomes, and triggers push/email results.

---

2. Coach Training Queue (`coachTrainQueue`)

This queue compiles training datasets, text translations, and vector databases for custom bots.

Operation 2.1: Translation of Bot Metadata

- **Trigger**: Message `type === 'translation'`.
- **Step-by-step Logic**:
  1. Initiates a POST request to `${process.env.SELF_ENDPOINT}/bot/translation/process`.
  2. **Inside Endpoint**: Uses the translation engine (OpenAI or Google Translate) to translate bot name, instructions, and training materials into target locales (e.g. Hindi, Spanish) and saves the translated schemas.

Operation 2.2: AI Coach Reference Data Vectorization

- **Trigger**: Standard training trigger message.
- **Step-by-step Logic**:
  1. Issues a POST request to `/bot/generate_embeddings-by-chapter-main`.
  2. **Inside Endpoint**: Downloads the reference documents (PDF/DOCX) from AWS S3, extracts text, cuts text into chunks, and uses `@xenova/transformers` (local) or OpenAI to generate vector embeddings. The resulting index is saved back as a `.json` vector file on S3 for semantic searches during roleplay.

---

3. PowerPoint to Video Queue (`pptVideoGenQueue`)

Converts uploaded training PPT slides into high-fidelity training video files. It processes `5` messages in parallel.

Operation 3.1: PPT Rendering & TTS Synthesis

- **Step-by-step Logic**:
  1. Dispatches message body to `${process.env.SELF_ENDPOINT}/bot/generate-ppt-video-main`.
  2. **Inside Endpoint**:
     - Extracts text notes from each slide.
     - Calls Azure TTS to generate spoken voice tracks for each slide's content.
     - Converts slide images and matching audio files, rendering them together into an output video segment via FFmpeg, and merges all segments into a single presentation video.

---

4. Roleplay Video Assembly Queue (`genRoleplayVideoQueue`)

Coordinates the generation of avatar review videos for completed sessions.

Operation 4.1: Avatar Review Video Generation

- **Step-by-step Logic**:
  1. Sends a POST request with the session metrics to `/chapters/generate-roleplay-video`.
  2. **Inside Endpoint**: Merges the avatar imagery with synthesized audio and user inputs, generating a unified visual review record.

---

5. Question Generation Queue (`questionGenerationQueue`)

Generates assessment question banks and MCQ quizzes.

Operation 5.1: Module/Course Creation

- **Trigger**: Message contains the `modules` parameter.
- **Step-by-step Logic**:
  1. Posts request data to `${process.env.SELF_ENDPOINT}/chapters/generate-module-main`.
  2. **Inside Endpoint**: Uses LLMs to generate course sections, text summaries, and chapter structures based on training parameters, creating the modules in MongoDB.

Operation 5.2: MCQ Quiz Generation

- **Trigger**: No modules parameter in message.
- **Step-by-step Logic**:
  1. Invokes `${process.env.SELF_ENDPOINT}/evaluations/generate-evaluation-question` via POST.
  2. **Inside Endpoint**: Evaluates the training modules, generates multiple-choice questions (MCQs), options, and correct answers using OpenAI schemas, saving them under the quiz collection.

---

6. Embeddings Queue (`embeddingsQueue`)

Coordinates core vector indexing tasks.

Operation 6.1: Video Cues & Prompts Generation

- **Trigger**: Message contains `ppt` key.
- **Step-by-step Logic**:
  1. Calls `${process.env.SELF_ENDPOINT}/bot/generate-video-prompt` via POST.
  2. **Inside Endpoint**: Generates detailed visual prompts matching slide narratives.

Operation 6.2: Global Bot Custom Training

- **Trigger**: Message contains `globalBot` key.
- **Step-by-step Logic**:
  1. POST request is sent to `${process.env.SELF_ENDPOINT}/chat/train-bot-main`.
  2. **Inside Endpoint**: Processes reference documents to generate global vectors.

Operation 6.3: Conversation Paths Generation

- **Trigger**: Message contains `generateGlobalBotTopics` key.
- **Step-by-step Logic**:
  1. POST request is made to `${process.env.SELF_ENDPOINT}/chat/generate-bot-topics-main`.
  2. **Inside Endpoint**: Structures chat paths and conversation trees.

---

7. Reports & Bulk Exports Queue (`reportsQueue`)

Generates large Excel/CSV analytics data sheets.

Operation 7.1: User Roleplay Result Reports Compile

- **Trigger**: Message `reportType === 'reports'`.
- **Step-by-step Logic**:
  1. Dispatches message to `/users/fetch-roleplay-reports-download`.
  2. **Inside Endpoint**: Extracts roleplay scorecards, parses grades, formats columns in an **ExcelJS** sheet, uploads the file to AWS S3, and sends the download link to the user.

Operation 7.2: Activity Report Mailing

- **Trigger**: Message `reportType === 'activities'`.
- **Step-by-step Logic**:
  1. POST to `/users/send-activity-report-mail`.
  2. **Inside Endpoint**: Compiles a CSV of user activities and sends it as an email attachment via AWS SES.

Operation 7.3: Employee bulk data Export

- **Trigger**: Message `reportType === 'employee-export'`.
- **Step-by-step Logic**:
  1. POST to `/users/process-employee-export`.
  2. **Inside Endpoint**: Exports a list of employee profiles, details, and access keys.

---

8. AI Analysis Queue (`aiAnalysisQueue`)

Generates deep qualitative reports of conversation logs.

Operation 8.1: Chat History Analysis

- **Step-by-step Logic**:
  1. Sends request to `/ai-examiner/ai_analysis`.
  2. **Inside Endpoint**: Analyzes dialog logs, extracts soft skills/objection handling parameters using OpenAI, and compiles an analysis summary report.

---

9. Video Checkpoints Queue (`videoInteractionQueue`)

Processes interactive training video checkpoints.

Operation 9.1: Log Video Watch Interactions

- **Step-by-step Logic**:
  1. POST request to `/video-interaction/process-internal`.
  2. **Inside Endpoint**: Records watch metrics, completed checkpoints, and answers to questions popped up during video watching.

---

10. Result Regenerator Queue (`resultRegenerateQueue`)

Regenerates evaluation sheets when administrative scoring rules change.

Operation 10.1: Bulk Scoring Update

- **Step-by-step Logic**:
  1. POST request to `/ai-examiner/ai_analysis`.
  2. **Inside Endpoint**: Re-evaluates target scripts and updates scores in the DB.

---

11. Scheduled Reminders Queue (`cronJobsQueue`)

Dispatches nudges and automated reminder notifications.

Operation 11.1: Training Expiry Nudge

- **Trigger**: Body contains `trainingExpiry`.
- **Step-by-step Logic**:
    1. Dispatches to `/users/send-training-expiry-nudge-main`.
    2. **Inside Endpoint**: Identifies users with pending deadlines and sends emails/push alerts.

Operation 11.2: Chat Auto Nudges (Smart/Auto/Manager Nudges)

- **Trigger**: Body contains `smartNudge` or `autoNudge` or `managerNudge`.
- **Step-by-step Logic**:
    1. Routes to the respective endpoint (e.g. `/chat/send-nudges-main` or `/users/send-manager-nudges-main`).
    2. **Inside Endpoint**: Fires scheduled notification reminders over SMS, Firebase Push Notifications, or Meta WhatsApp templates.

---

12. Dead Letter Queue (`deadLetterQueue`)

Handles crashed tasks to prevent infinite processing loops.

Operation 12.1: Crash Email Alerts Dispatch

- **Step-by-step Logic**:
    1. The worker parses the crashed job payload.
    2. It constructs an HTML diagnostic page displaying the exact raw message details.
    3. It calls `this.emailService.sendMail` to deliver the report to `tech.support@mple.ai` for debugging, then purges the message from SQS.