**📌 Voice-to-Text Meeting Minutes Generator**

**🔍 Introduction**

In today’s fast-paced corporate environment, documenting meeting minutes manually is inefficient and error-prone. This project automates the process by converting meeting audio into accurate, structured summaries using AWS and NLP tools. When a voice recording is uploaded to an S3 bucket, AWS Transcribe converts speech to text, followed by post-processing with NLP models like NLTK and GPT to eliminate fillers, silences, and noise. A clean, concise summary is then generated and displayed via a user-friendly dashboard, helping stakeholders quickly review discussions—even if they missed the meeting.

⏱️ **Key Benefits**:

* + For those who **missed the meeting** – get the full summary in seconds.
  + For **busy executives** – review meetings faster with crisp, action-focused points.
  + Enables **automated, scalable, and clear documentation** with zero manual effort.

**💼 Use Case**

* Automatically generate meeting minutes from voice recordings.
* Eliminate silences, "um"s, gaps, and irrelevant audio using NLP.
* Provide fast, precise summaries for decision-makers and absentees.
* Boost productivity and knowledge retention across distributed teams.

**🛠️ Tools & Technologies**

* **Programming & Frameworks**: Python, Flask (Dashboard & API)
* **AWS Services**:
  + **S3**: Audio storage
  + **Lambda**: Event-driven transcription job trigger
  + **Transcribe**: Speech-to-text conversion
* **NLP Stack**: NLTK (noise removal), TextRank/GPT (summary generation)
* **IDE**: Visual Studio Code (VS Code)
* **Other Tools**: Boto3 (AWS SDK), JSON, REST APIs

**☁️ Cloud & Data Engineering Aspects**

* Serverless architecture using **AWS Lambda** for scalability and cost-efficiency.
* **S3 buckets** as secure, versioned audio input/output storage.
* **Event-driven pipelines** triggered on S3 file upload.
* **ETL-like flow**: Audio → Text (Transcribe) → Cleaned Transcript (Lambda) → Summarized Output (GPT).
* Reusability, modularity, and fault tolerance via microservices.

**🚀 Future Scope**

* Add **real-time transcription** and speaker identification.
* Integrate with **meeting platforms** (Zoom, Teams, Google Meet).
* Enable **multilingual transcription** and **translation** support.
* Generate **action items, sentiment analysis**, and **keyword extraction**.
* Store summaries in **RDS or DynamoDB** with search capability.

**DATA FLOW OVERVIEW**

**End-to-end modular flow from uploading audio ➝Lambda invokes transcribing ➝ processing ➝ summarization ➝ dashboard + download**

**Clearly maps to your project modules:**

**Module 1 → Audio Upload & S3**

**Module 2 →Lambda Function Automatically Invokes AWS Transcribe**

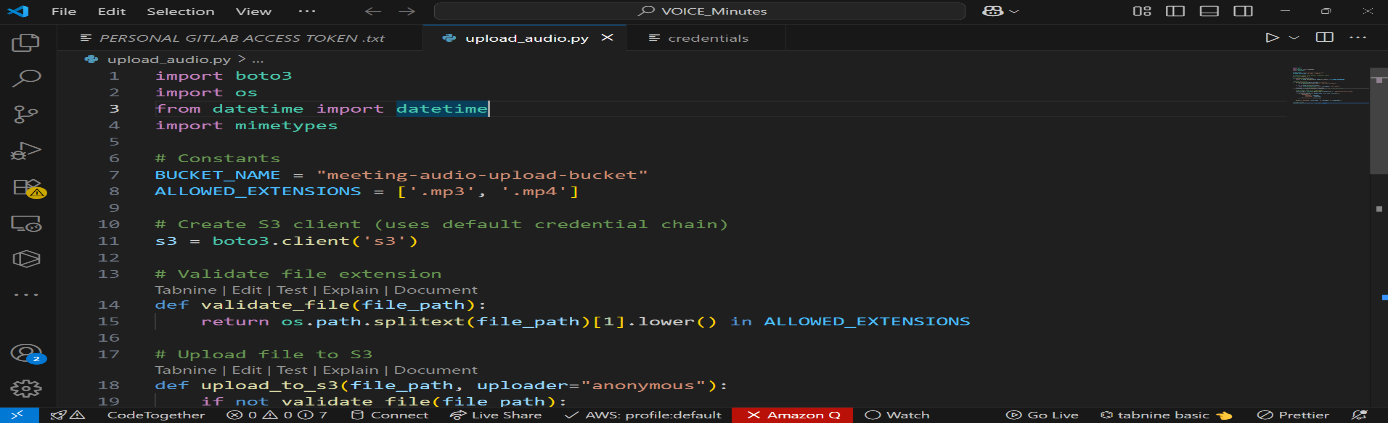
**Module 3 → Transcript Post-Processing**

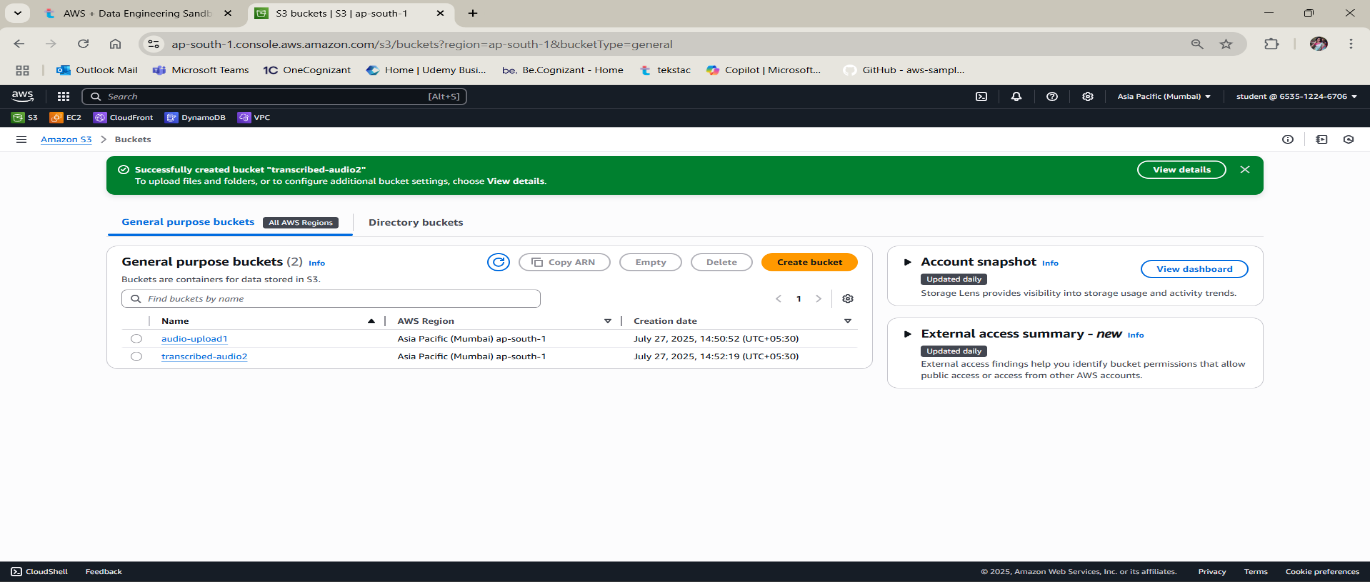
**Module 4 → Summary Generation**

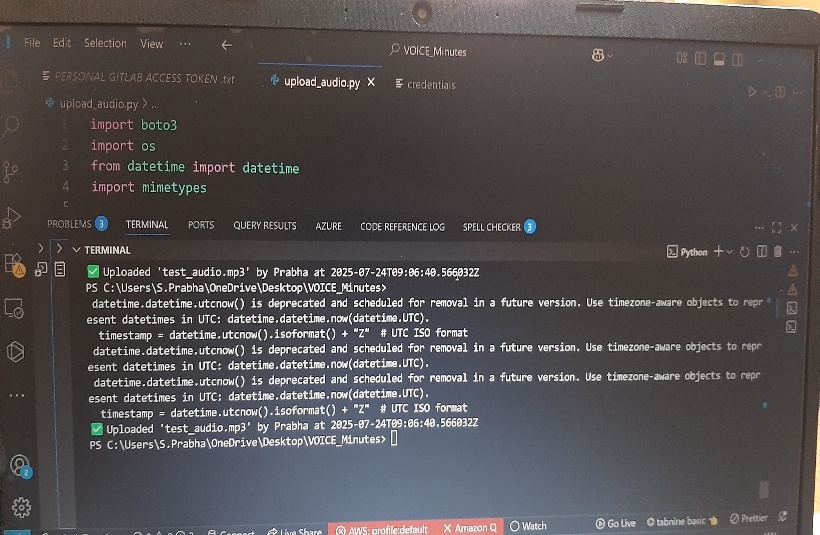
**Module 5 → Dashboard & Downloadable Reports**

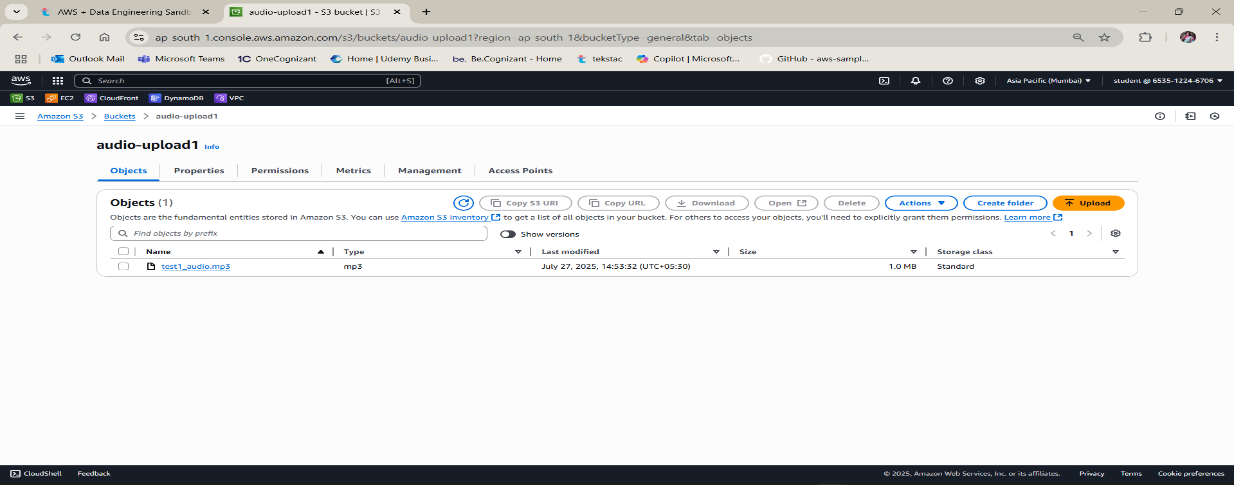
**Modules and Functionalities**

**🧩 Module 1: Audio Upload & S3 Integration**

* **✅ Built a Python interface (or CLI) to upload .mp3/.mp4 files.**
* **✅ Uploaded audio stored in S3 bucket audio\_upload1.**
* **✅ Validates audio format and file naming (e.g., test\_audio.mp3).**
* **✅ Stores metadata (filename, timestamp).**
* **✅ Upload status shown to user via console or CLI.**
* **🔧 AWS Used: S3  
  📁 Bucket Created: audio\_upload1**



**Data upload in bucket**: **audio\_upload1**  
  


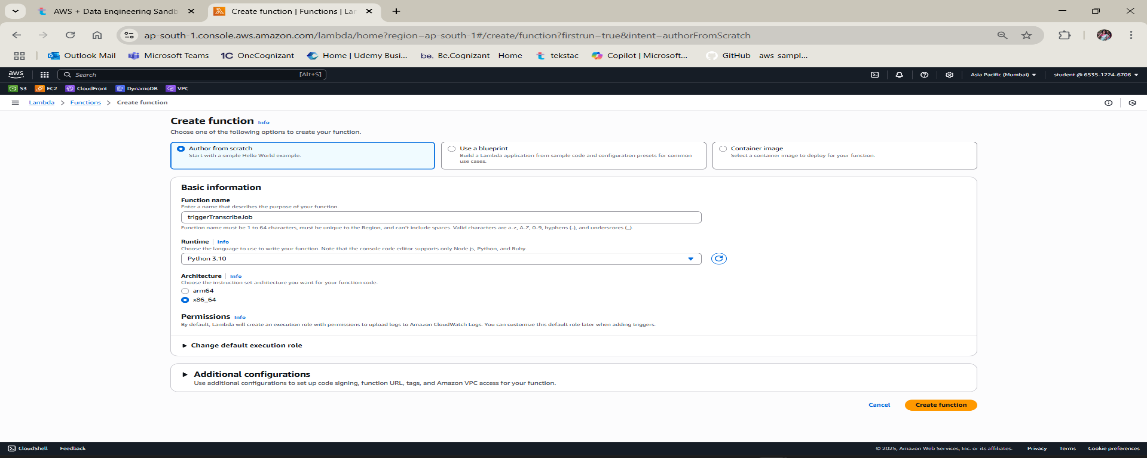


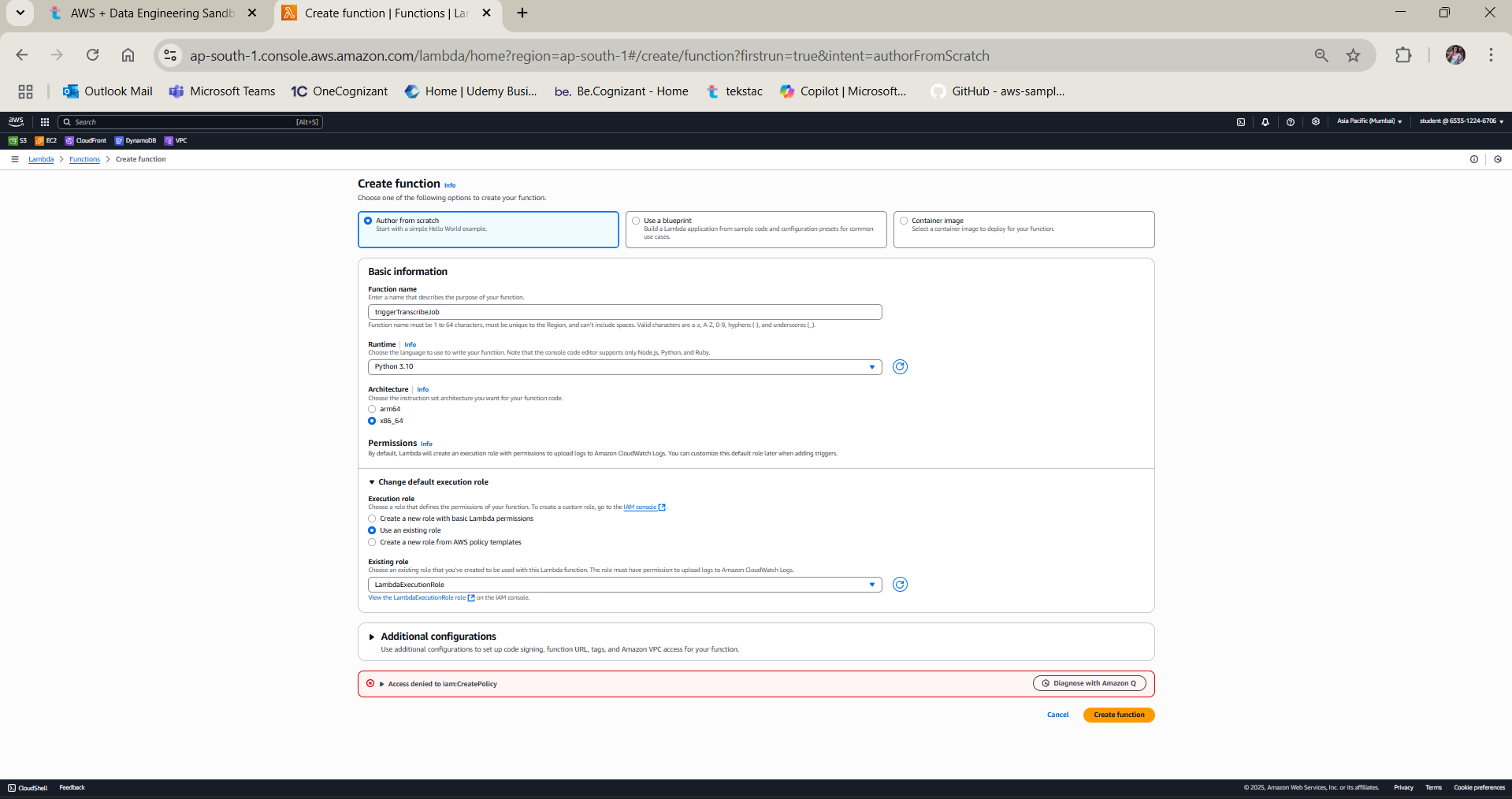
**🧩 Module 2: AWS Transcribe Setup**

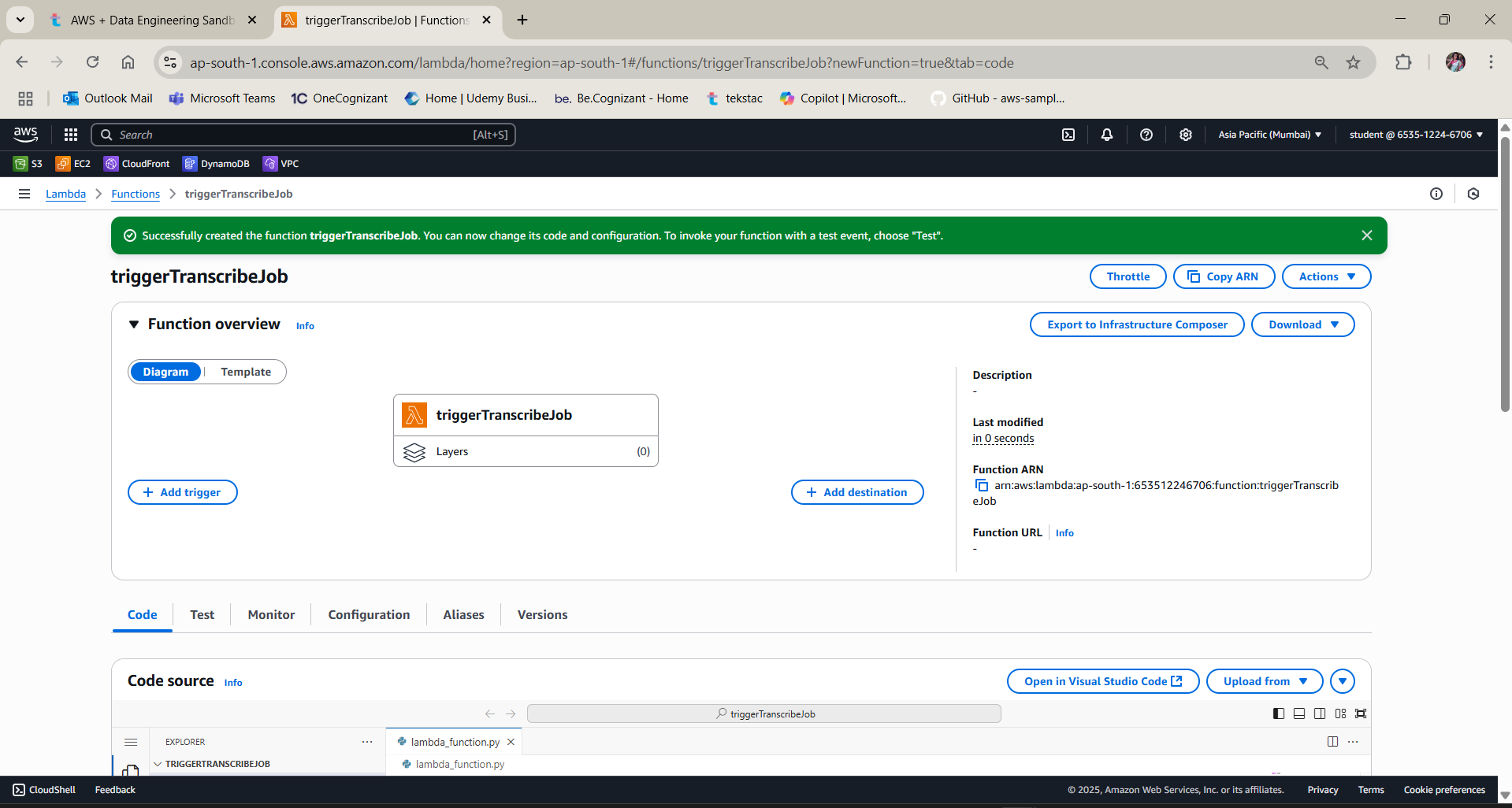
* **✅ Lambda automatically triggers AWS Transcribe when new audio is uploaded.**
* **✅ Transcribe converts audio to text with speaker labels.**
* **✅ Job monitored by Lambda and result stored in transcribed\_audio2 bucket.**
* **✅ Output format: raw JSON transcript.**

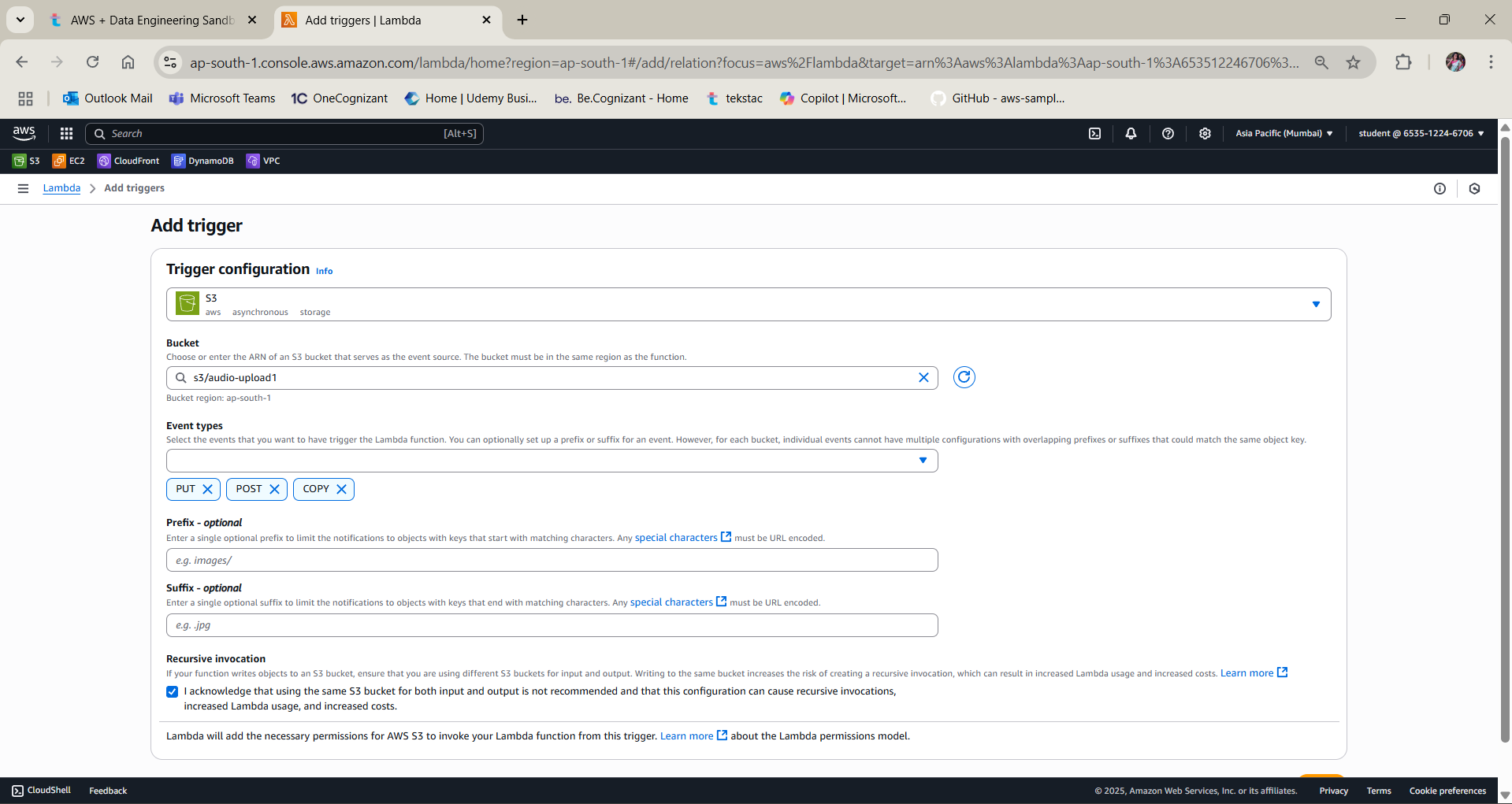
**🪄 Services Used:**

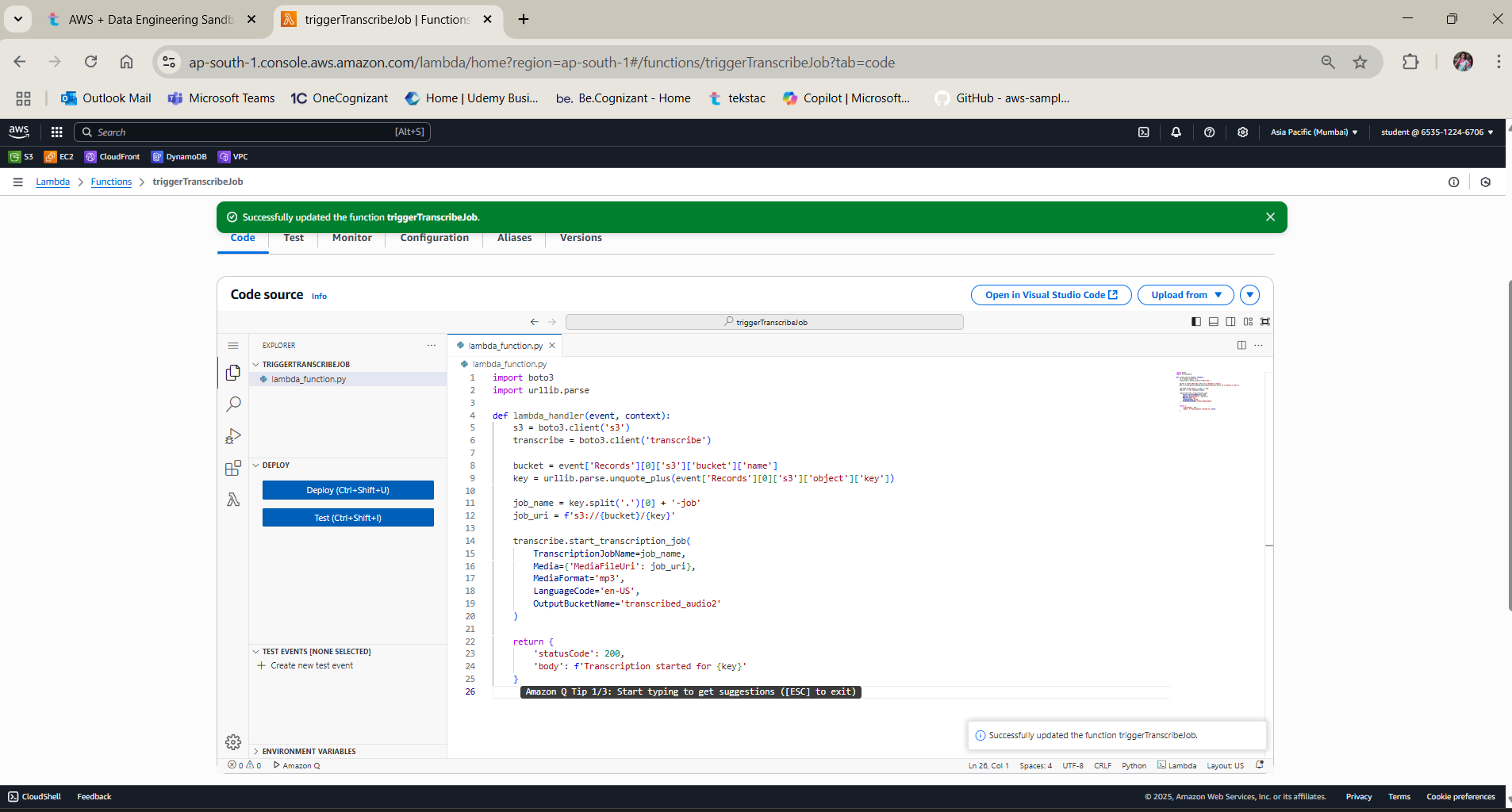
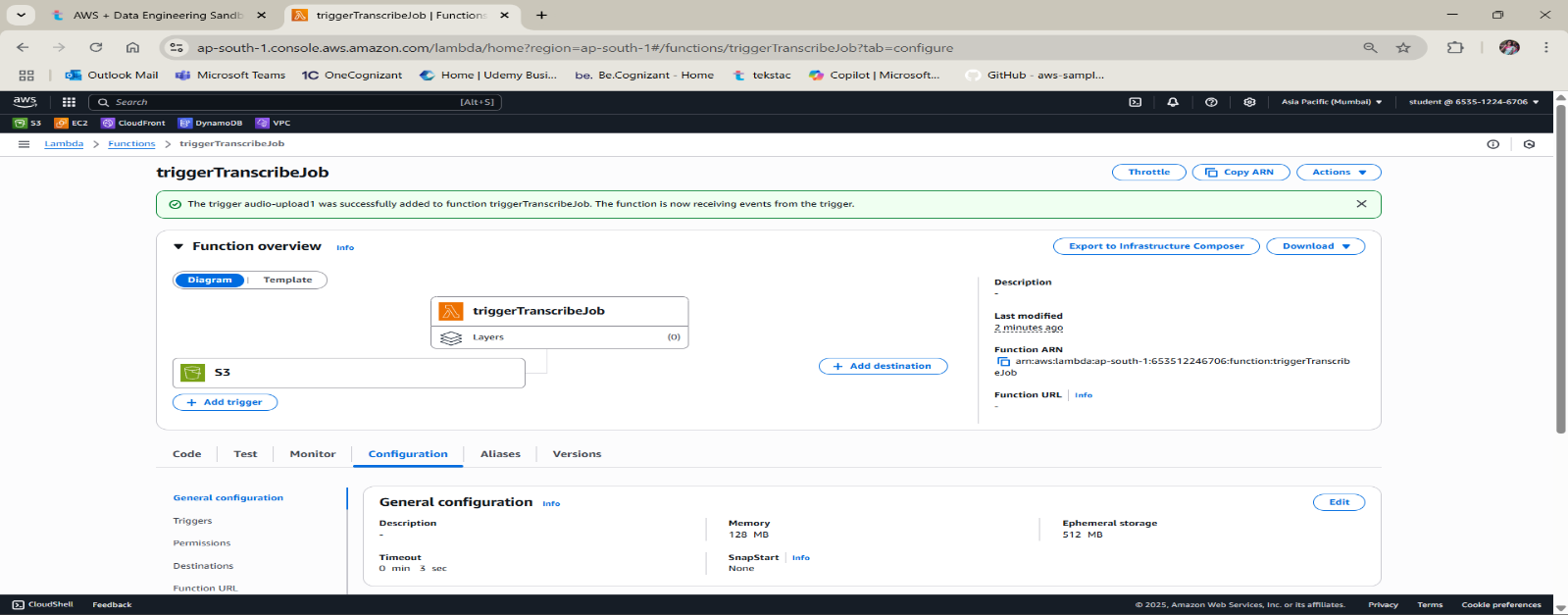
* **Lambda (trigger)**
* **Amazon Transcribe**
* **S3 (transcribed\_audio2)**



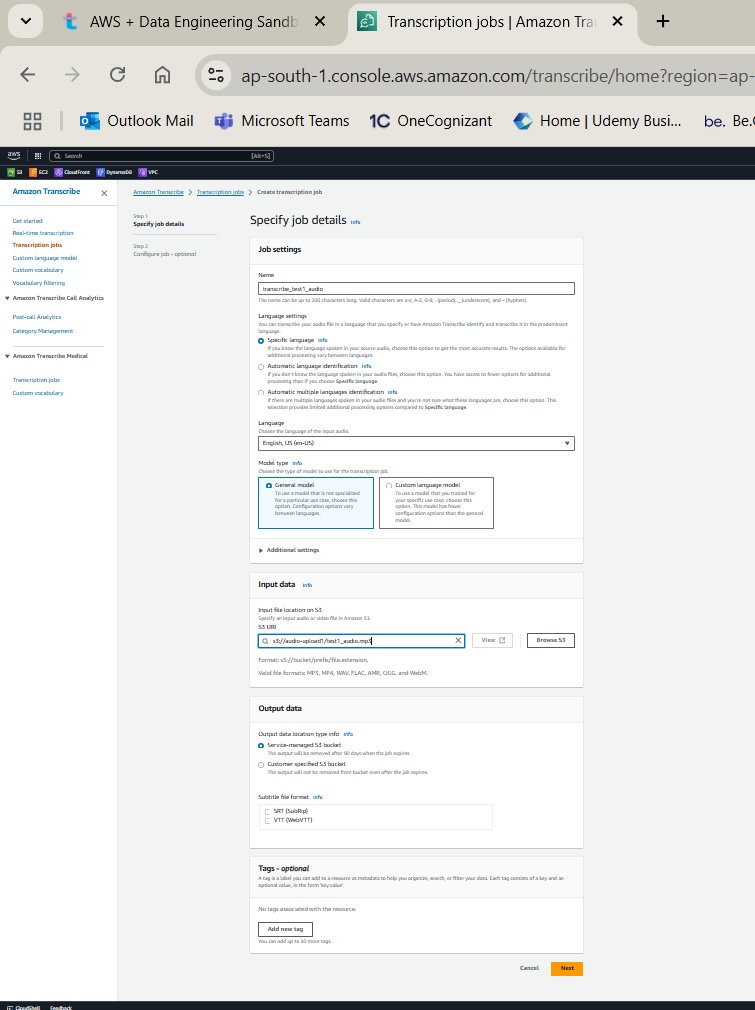


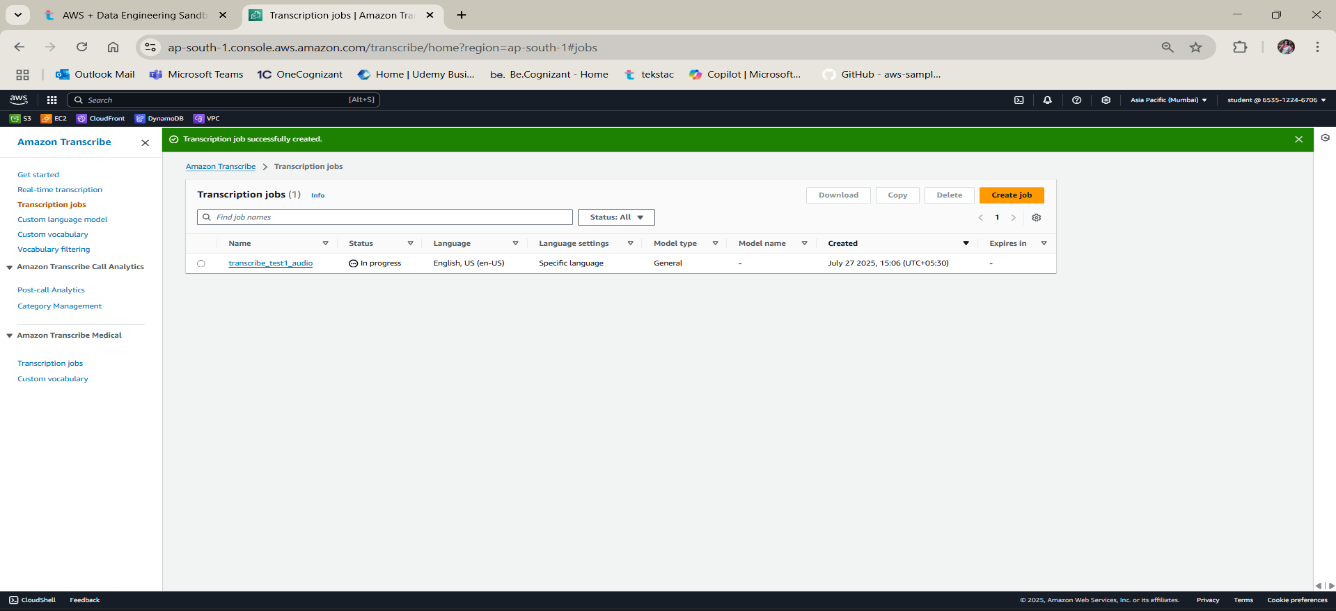


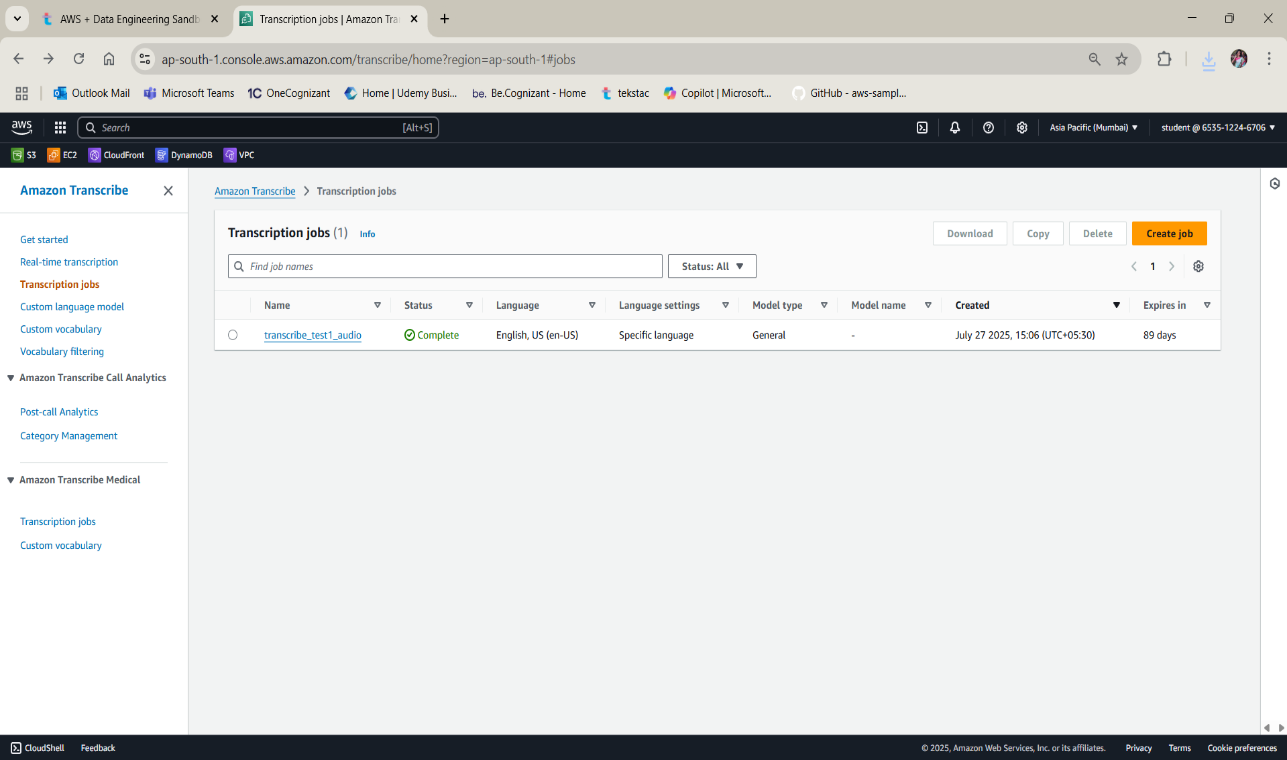


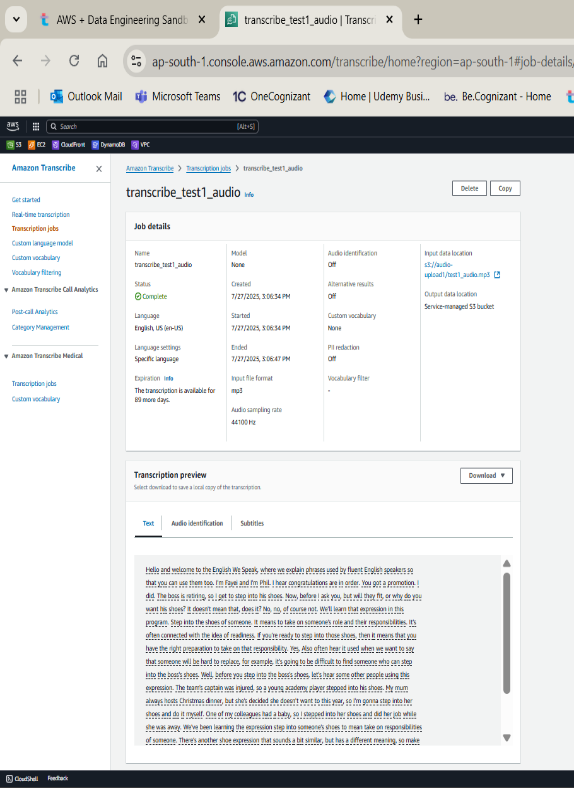
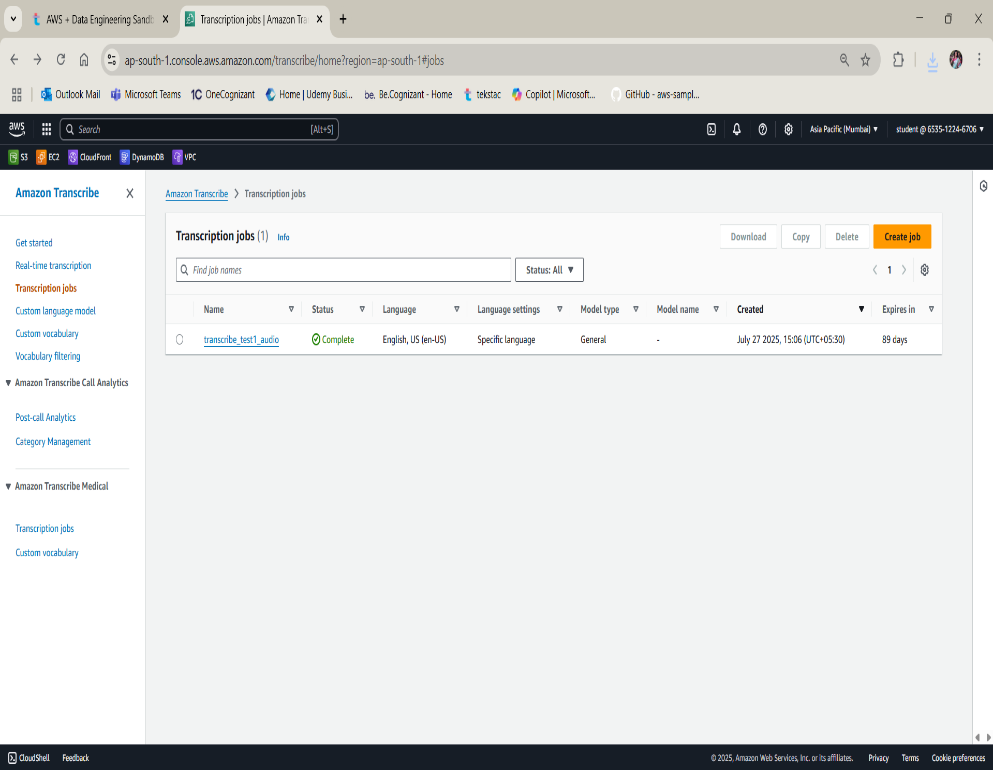


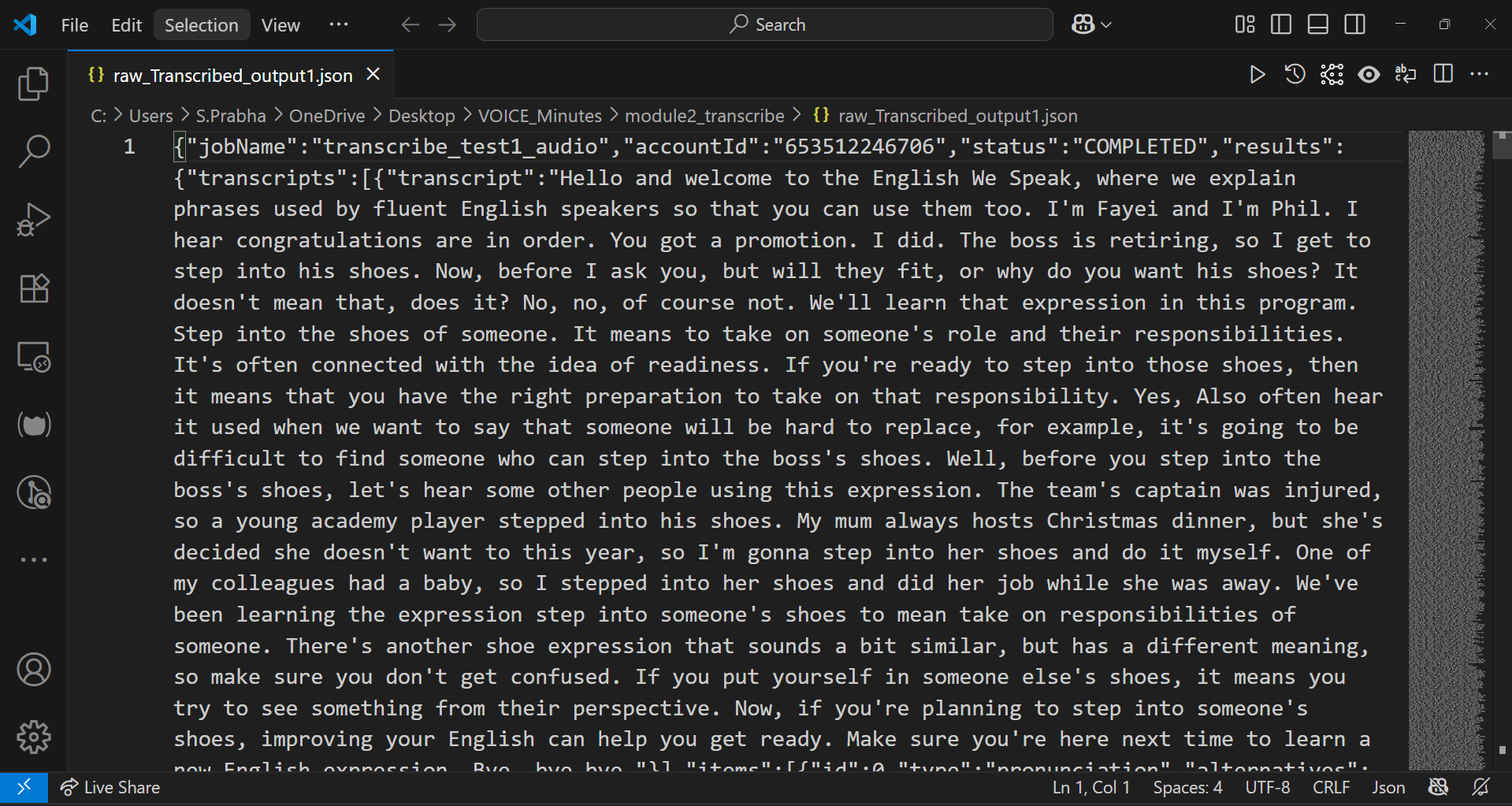
**AWS Transcribe Automatically Triggered Whenever Data is Uploaded in S3 Bucket:-**









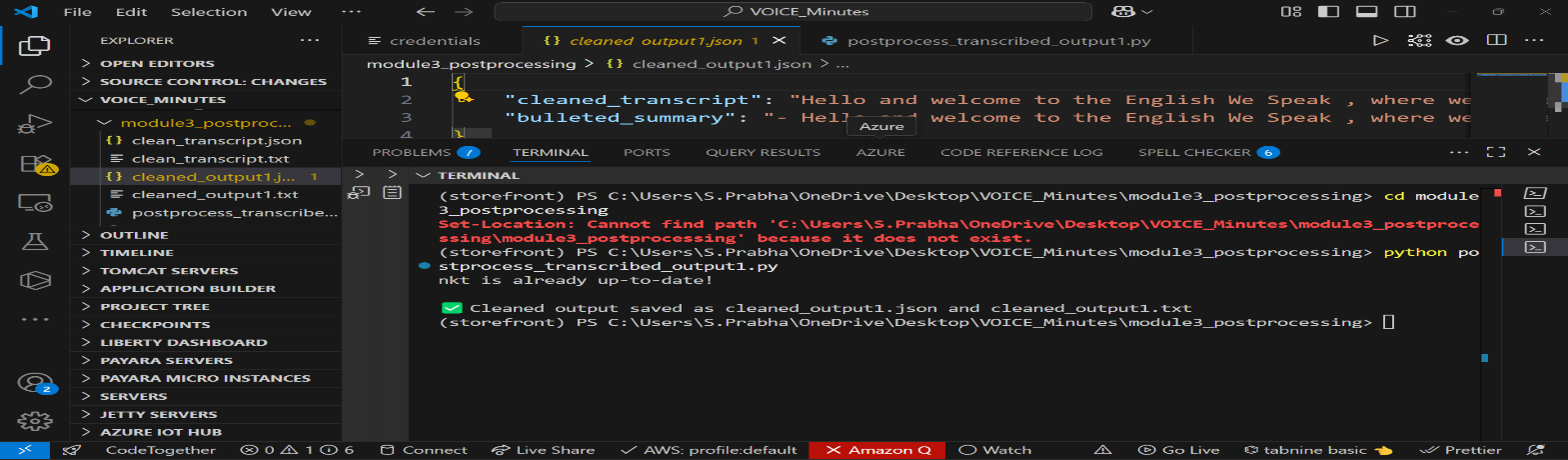
**🧩 Module 3: Transcript Post-Processing (5 mins)**

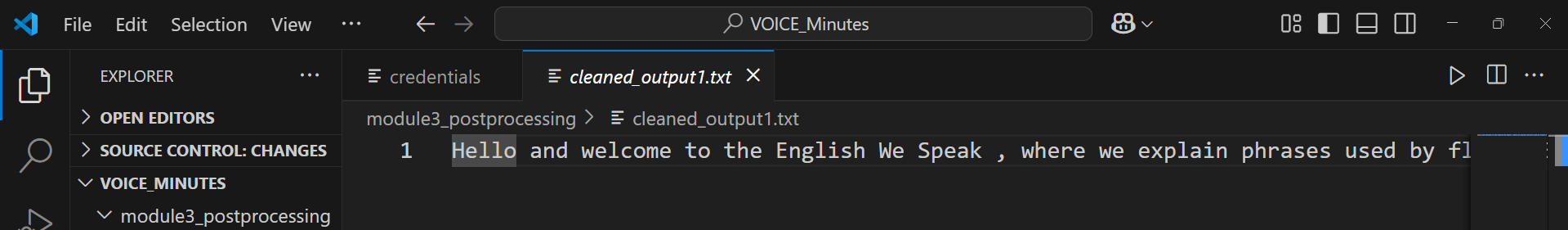
* **✅ Input: Raw JSON transcript.**
* **✅ Cleaned filler words (uh, um, like…).**
* **✅ Formatted text into paragraphs and bullet points.**
* **✅ Extracted speaker labels, timestamps if available.**
* **✅ Output saved as both .json and .txt.**

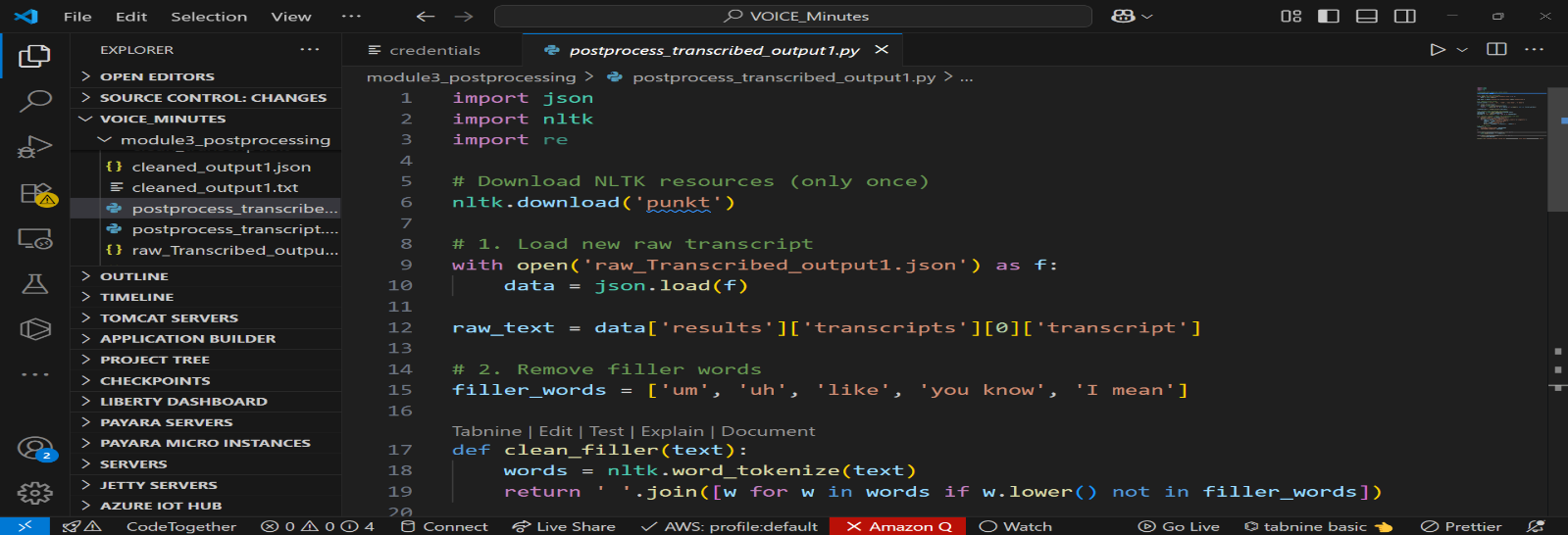
**📁 Files:**

* **clean\_transcript.json, clean\_transcript.txt**
* **cleaned\_output1.json, cleaned\_output1.txt (for 2nd audio)**

**🧠 NLP Used: NLTK (for tokenizing, sentence segmentation)**

****

****



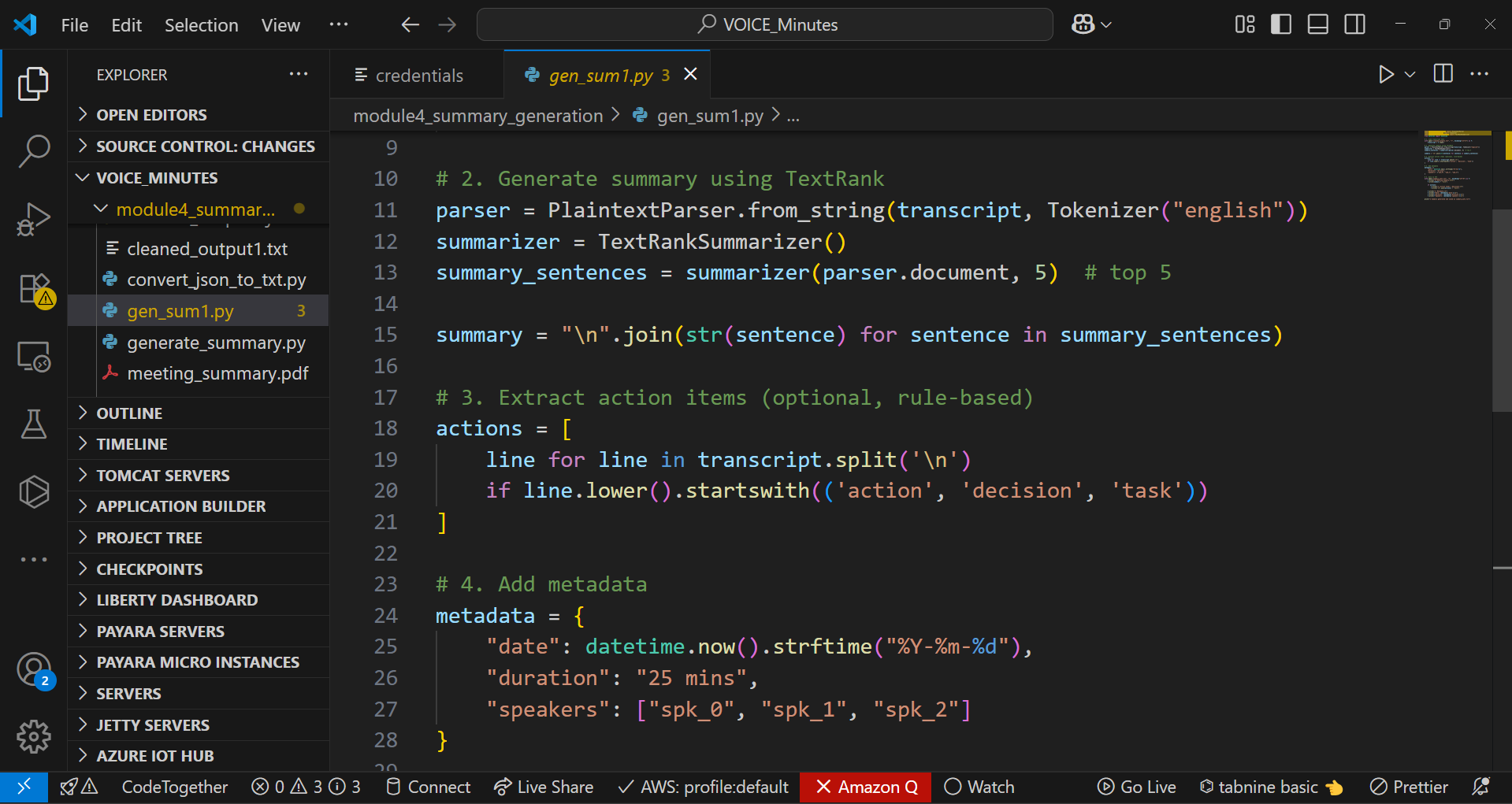
**🧩 Module 4: Meeting Summary Generation (6 mins)**

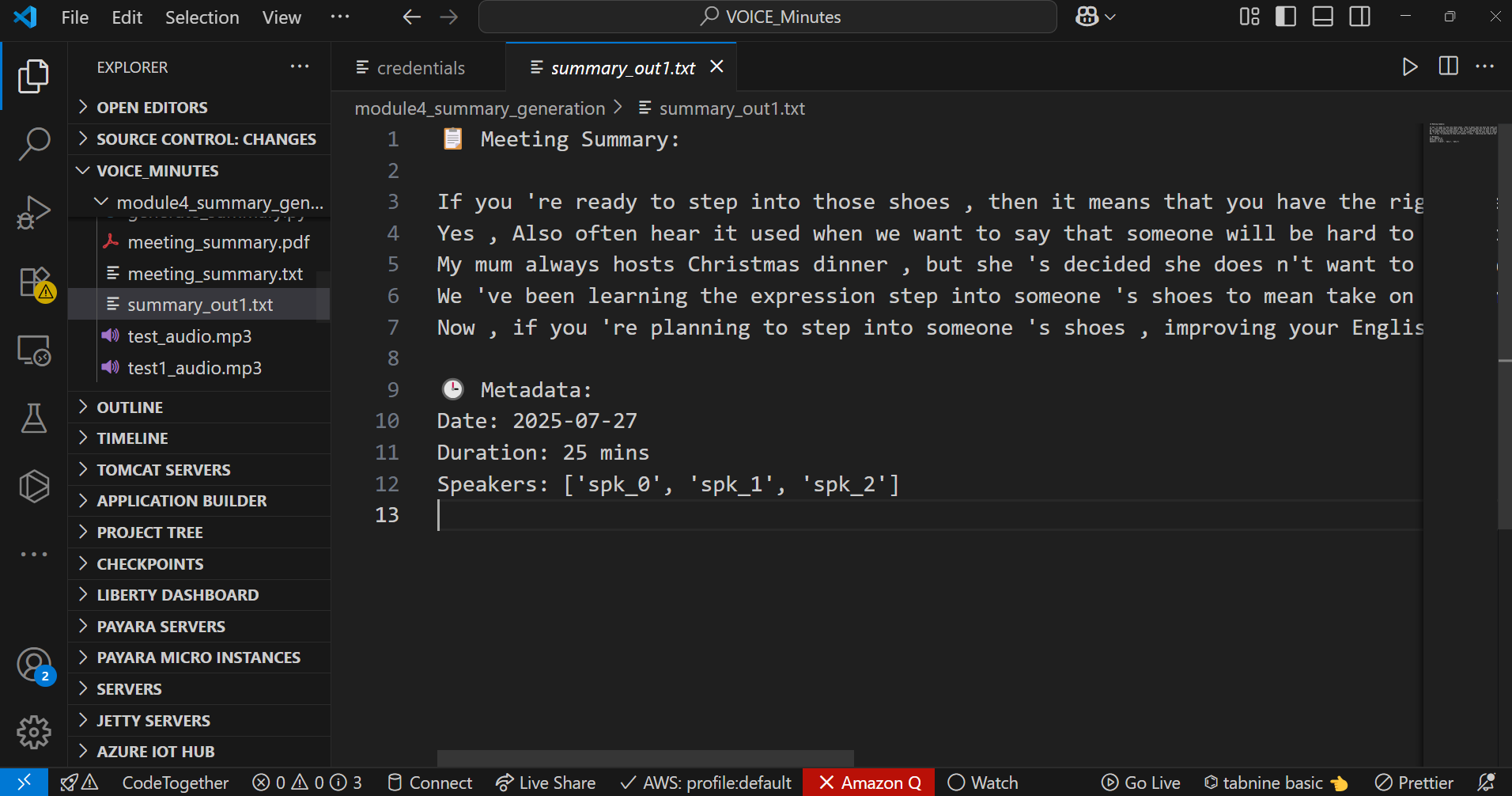
* **✅ Used TextRank via sumy to extract key points.**
* **✅ Generated:**
  + **Short Summary**
  + **Action items (if available)**
  + **Metadata (date, duration, speakers)**
* **✅ Output saved as .txt (PDF optional, but skipped per project scope)**

**📁 Files:**

* **meeting\_summary.txt ← from test\_audio.mp3**
* **summary\_out1.txt ← from test1\_audio.mp3**

**🧠 NLP Used: Sumy (TextRank), datetime for metadata**

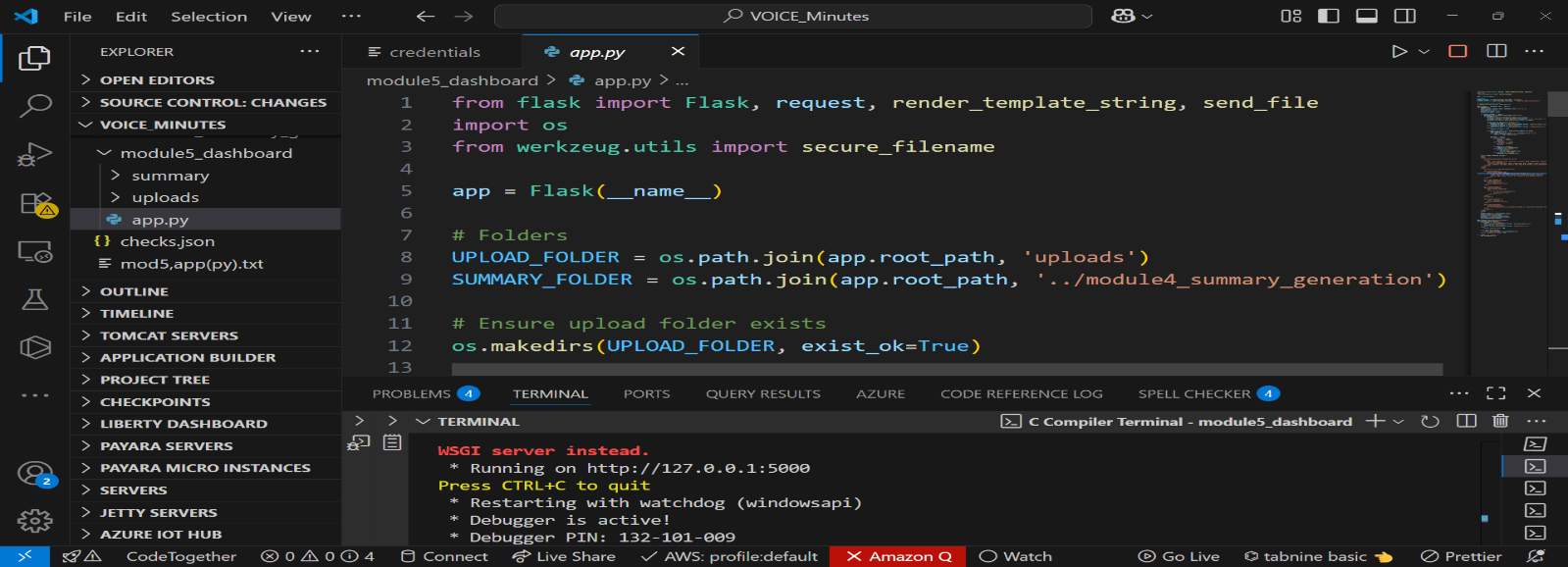
****

****

**🧩 Module 5: Dashboard & Downloadable Reports (7 mins)**

* **✅ Built with Flask (single app.py file).**
* **✅ Upload .mp3 → Dashboard fetches matching .txt summary.**
* **✅ Dynamically shows:**
  + **Summary**
  + **Action Items**
  + **Metadata**
* **✅ Download options (TXT only).**
* **✅ Works with multiple audios: test\_audio.mp3 → meeting\_summary.txt, test1\_audio.mp3 → summary\_out1.txt.**

**🌐 Tech Used: Flask, HTML templating  
📁 Folder: module5\_dashboard/uploads, links to module4\_summary\_generation**

****

