

## **AI 620 Emerging Topics in Artificial Intelligence**

### **HOS03A Universal Translator**

02/20/2023 Developed by Yared Shewarade

09/17/2024 Updated by Anh Nguyen

10/17/2024 reviewed by Jonathan Koerber

School of Technology and Computing (STC) @City University of Seattle (CityU)

### **Before You Start**

- The directory path shown in screenshots may be different from yours.
- Some steps are not explained in the tutorial. If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask the courses student worker for help.

### **Learning Outcomes**

Students will be able to learn:

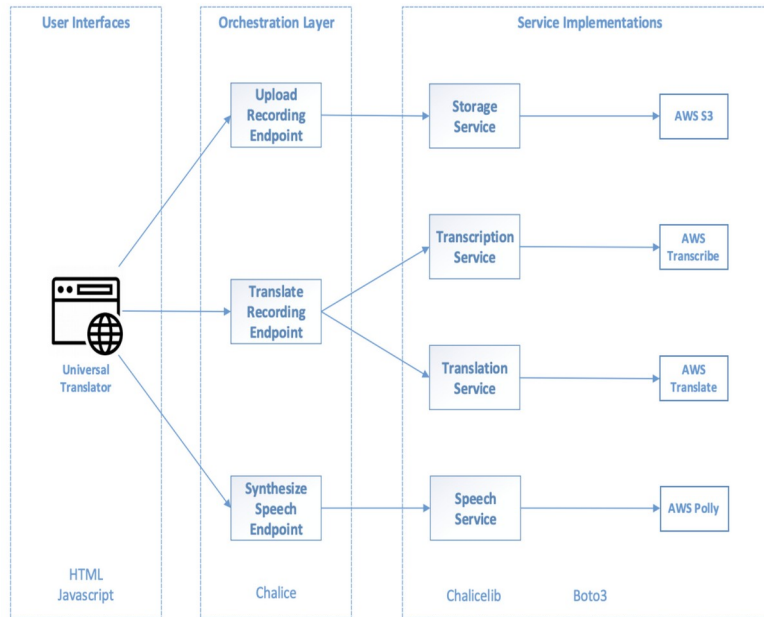
- Introduction to Universal Translator
- Setting up universal translator application project structure

### **Resources**

- Tripuraneni, S., & Song, C. (2019). Hands-on artificial intelligence on amazon web services: Decrease the time to market for AI and ML applications with the power of AWS (1st ed.). Packt.

## **Introduction to Universal Translator**

Universal Translator application will provide a web user interface for the users to record a phrase in one language and then translate that phrase to another language. The architecture design highlights the layers and services of our application.



The web user interface will interact with three RESTful endpoints in the orchestration layer:

- Upload Recording Endpoint will delegate the audio recording upload to our Storage Service, which provides an abstraction layer to AWS S3.
- Translate Recording Endpoint will use both the Amazon Transcription Service and the Amazon Translation Service. It first gets the transcription of the audio recording, and then translates the transcription text to the target language. The transcription service and the translation service abstract the Amazon Transcribe and Amazon Translate services respectively.
- Synthesize Speech Endpoint will delegate the speech synthesis of the translated text to the Speech Service, which is backed by the Amazon Polly service.

## Setting up universal translator project structure

**Note:** For submission, take the screenshot for all steps and save it in your local repository.

1. In Visual Studio, Open your working directory (AI620-Fall2024-HOS03)
2. Type the following to create the root project directory

```
$ mkdir UniversalTranslator
$ cd UniversalTranslator
```

3. Type the following to create placeholders for the web fronted by creating a directory (website) and within this directory, create the index.html and scripts.js files.

```
$ mkdir Website
$ touch Website/index.html
$ touch Website/scripts.js
```

4. Type the following to create a Python virtual environment

```
python -m pip install --user virtualenv
python -m venv pipenv
```

Type the following to activate the virtual environment

- Windows:

```
pipenv\Scripts\activate
```

- OSX

```
pipenv/bin/activate
```

In case you face the permission denied issue while using the above command. Please run the below commands. Below are images showing the above command and ways to handle the permission denied issue by using any of the below commands.

```
. pipenv/bin/activate
```

(Note: *space between dot and 'venv/bin/activate'*)

Or

```
source pipenv/bin/activate
```

5. Type the following scripts to create a Python 3 virtual environment with pipenv in the project's root directory and install boto3 and chalice packages.

**Note:** if pipenv doesn't work, you can use pip/pip3

```
$ pipenv install boto3
$ pipenv install chalice
```

6. Next, while still in the virtual environment, type the following scripts to create the orchestration layer as an AWS [chalice](#) project named [Capabilities](#).

```
$ chalice new-project Capabilities
```

7. Type the following to create the [chalicelib](#) Python package.

```
cd Capabilities
mkdir chalicelib
touch chalicelib/__init__.py
cd ..
```

8. The initial project structure for Universal Translator should look like the following. This is the Universal Translator project structure which contains a user interface, orchestration, and the service implementation layers of the AI application architecture

#### Project Structure

```
-----
├─ UniversalTranslator/
│   ├── Capabilities/
│   │   ├── .chalice/
│   │   │   ├── config.json
│   │   ├── chalicelib/
│   │   │   ├── __init__.py
│   │   ├── app.py
│   │   └── requirements.txt
│   ├── Website/
│   │   ├── index.html
│   │   └── script.js
│   ├── Pipfile
│   └── Pipfile.lock
```

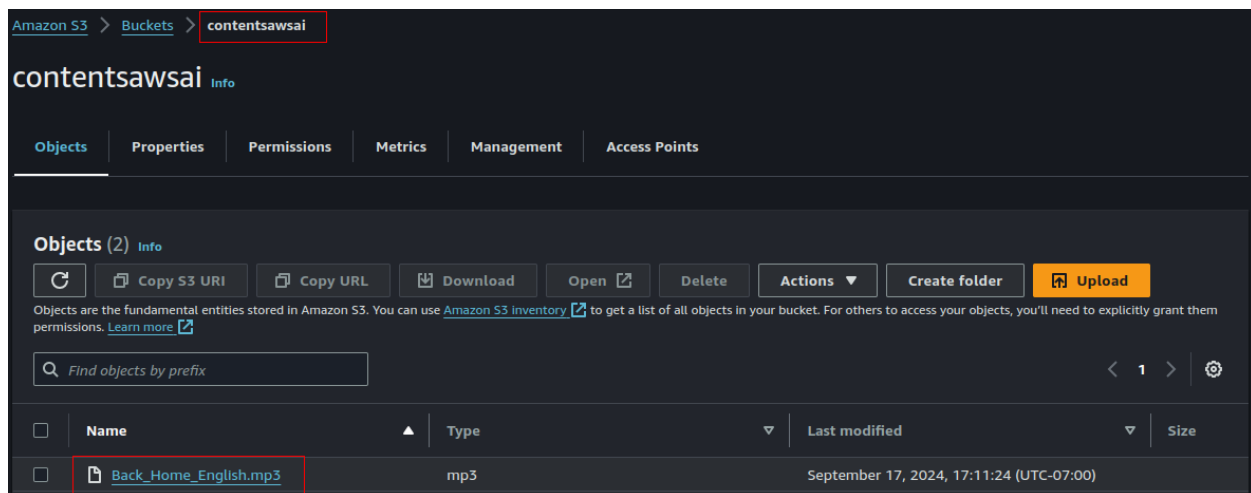
## Implementation of the Universal Translator on AWS

1. Upload audio file to Amazon S3 bucket

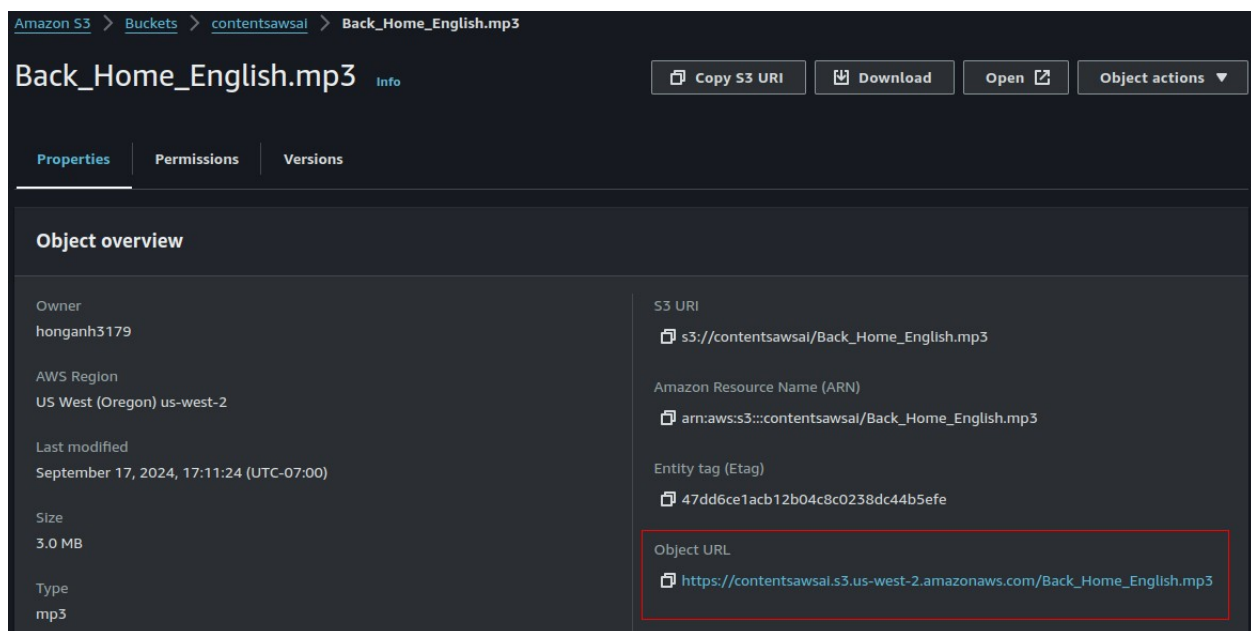
Log in to your AWS Console > Go to S3.

Use the bucket in your HOS02, upload the provided audio file to your S3 bucket.

Then copy the Object URL.



tr



## 2. Set up AWS configuration with Access Key and Secret Key ID

In HOS02, you have downloaded a CSV file containing the Access Key and the Secret Key ID of an IAM user with Administrator access.

Open your terminal/command prompt and type this command:

```
aws configure
```

- AWS Access Key ID: Your Access Key ID
- AWS Secret Access Key: Your Secret Access Key
- Default region name: Enter your S3 bucket region. Eg, us-west-2
- Default output format: json
- Press Enter to save all of the above configuration.

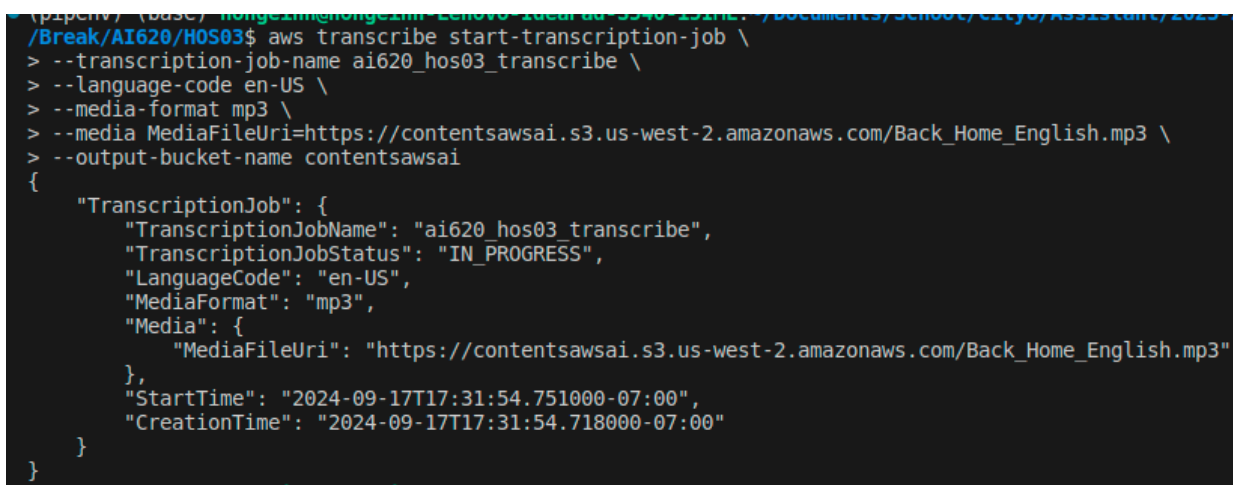
### 3. Transcription service – speech-to-text

- a. Open AWS CLI and type the following command to start a transcription. Replace the text in red with your own Object URL and your bucket name.

```
aws transcribe start-transcription-job \
--transcription-job-name ai620_hos03_transcribe \
--language-code en-US \
--media-format mp3 \
--media MediaFileUri=<Object URL you copied in previous step> \
--output-bucket-name <Your bucket name>
```

**Note:** - The job name should be unique ID for each transcription job, and the output bucket name specifies in which S3 bucket the transcript

The output should look like this:



```
(pipenv) (base) nongezim@nongezim-Lenovo-Idea-15-82: /Documents/School/City/Assistant/2023-24$ aws transcribe start-transcription-job \
> --transcription-job-name ai620_hos03_transcribe \
> --language-code en-US \
> --media-format mp3 \
> --media MediaFileUri=https://contentsawsai.s3.us-west-2.amazonaws.com/Back_Home_English.mp3 \
> --output-bucket-name contentsawsai
{
  "TranscriptionJob": {
    "TranscriptionJobName": "ai620_hos03_transcribe",
    "TranscriptionJobStatus": "IN_PROGRESS",
    "LanguageCode": "en-US",
    "MediaFormat": "mp3",
    "Media": {
      "MediaFileUri": "https://contentsawsai.s3.us-west-2.amazonaws.com/Back_Home_English.mp3"
    },
    "StartTime": "2024-09-17T17:31:54.751000-07:00",
    "CreationTime": "2024-09-17T17:31:54.718000-07:00"
  }
}
```

- b. Type the following in your AWS CLI to check the status of the job

```
aws transcribe get-transcription-job --transcription-job-name
ai620_hos03_transcribe
```

The output result looks like the following.

```

/HOS03$ aws transcribe get-transcription-job --transcription-job-name ai620_hos03_transcribe
{
  "TranscriptionJob": {
    "TranscriptionJobName": "ai620_hos03_transcribe",
    "TranscriptionJobStatus": "COMPLETED",
    "LanguageCode": "en-US",
    "MediaSampleRateHertz": 44100,
    "MediaFormat": "mp3",
    "Media": {
      "MediaFileUri": "https://contentsawsai.s3.us-west-2.amazonaws.com/Back_Home_English.mp3"
    },
    "Transcript": {
      "TranscriptFileUri": "https://s3.us-west-2.amazonaws.com/contentsawsai/ai620_hos03_transcribe.json"
    },
    "StartTime": "2024-09-17T17:31:54.751000-07:00",
    "CreationTime": "2024-09-17T17:31:54.718000-07:00",
    "CompletionTime": "2024-09-17T17:32:21.342000-07:00",
    "Settings": {
      "ChannelIdentification": false,
      "ShowAlternatives": false
    }
  }
}

```

4. [Optional Question] Speech Service – text-to-speech
  - a. Type the following in your AWS CLI to generate an audio speech

```

aws polly start-speech-synthesis-task \
--output-format mp3 \
--output-s3-bucket-name <Your bucket name> \
--text "Testing testing 1 2 3 Text to Speech" \
--voice-id Ivy

```

```

/HOS03$ aws polly start-speech-synthesis-task \
> --output-format mp3 \
> --output-s3-bucket-name contentsawsai \
> --text "Testing testing 1 2 3 Text to Speech" \
> --voice-id Ivy
{
  "SynthesisTask": {
    "TaskId": "8867877f-9af6-4439-bc58-e73eb37bb124",
    "TaskStatus": "scheduled",
    "OutputUri": "https://s3.us-west-2.amazonaws.com/contentsawsai/8867877f-9af6-4439-bc58-e73eb37bb124.mp3",
    "CreationTime": "2024-09-17T17:40:58.412000-07:00",
    "RequestCharacters": 36,
    "OutputFormat": "mp3",
    "TextType": "text",
    "VoiceId": "Ivy"
  }
}

```

- b. Type the scripts to check the status of the task

```

aws polly get-speech-synthesis-task --task-id <Task ID from the
previous output>

```

```
$ aws polly get-speech-synthesis-task --task-id e68d1b6a-4b7f-4c79-9483-2b5a5932e3d1

{
  "SynthesisTask": {
    "TaskId": "e68d1b6a-4b7f-4c79-9483-2b5a5932e3d1",
    "TaskStatus": "completed",
    "OutputUri": "https://s3.us-east-1.amazonaws.com/<bucket>/<task id>.mp3",
    "CreationTime": 1552754991.114,
    "RequestCharacters": 21,
    "OutputFormat": "mp3",
    "TextType": "text",
    "VoiceId": "Ivy"
  }
}
```

## HOS submission instructions:

1. Please install the GitHub Desktop: [https://cityuseattle.github.io/docs/git/github\\_desktop/](https://cityuseattle.github.io/docs/git/github_desktop/)
2. Clone, organize, and submit your work through GitHub Desktop:  
<https://cityuseattle.github.io/docs/hoporhos>