AI 620 Emerging Topics in Artificial Intelligence

HOS04A Contact Organizer Application

02/20/2023 Developed by Yared Shewarade

09/17/2024 Updated by Anh Nguyen

10/17/2025 review 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 Contact Organizer Application
- Setting up contact organizer application project structure

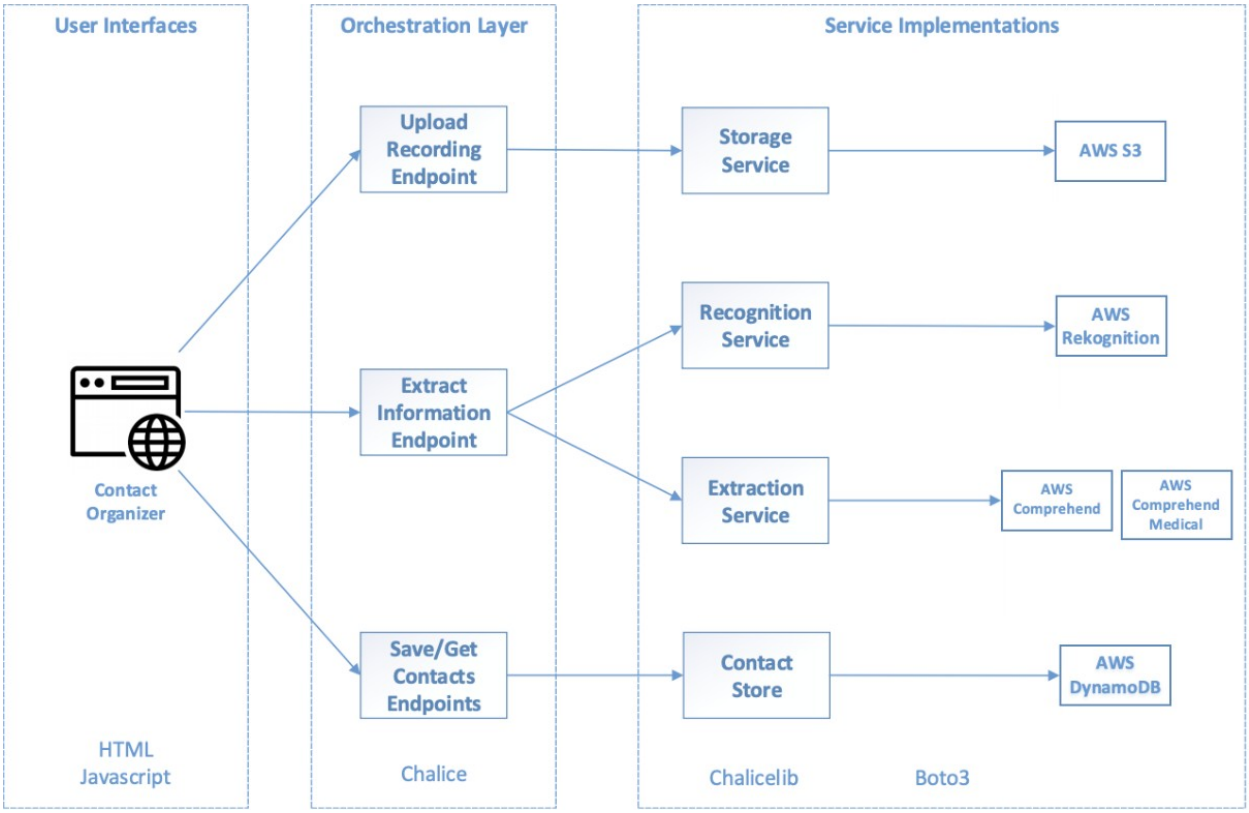
Resources

- 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 Contact Organizer Application

The Contact Organizer application will provide a web user interface for users so that they can upload an image of a business card. The contact information will be extracted and categorized by the application. The automatically extracted contact information will then be displayed to the user in the web user interface. The user can review and correct the information before saving it to a permanent contact store.

The following diagram shows the architecture's design, highlighting the layers and services of the Contact Organizer application.



In this application, the web user interface will interact with three RESTful endpoints in the orchestration layer:

- **Upload Recording Endpoint** will delegate the image upload to our storage service.
- **Extract Information Endpoint** will use the recognition service and extraction Service:
- **Save/Get Contacts Endpoints** will write to/read from the **Contact Store**, which is backed by the AWS DynamoDB NoSQL database.

Setting up universal translator project structure

1. In Visual Studio, Open your working directory (AI620-Spring2023-HOS04)
2. Type the following to create the root project directory

```
$ mkdir ContactOrganizer
$ cd ContactOrganizer
```

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 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.

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

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

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 ..
```

The initial project structure for Contact 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
------------
├── ContactOrganizer/
    ├── Capabilities/
        ├── .chalice/
            ├── config.json
        ├── chalicelib/
            ├── __init__.py
        ├── app.py
        ├── requirements.txt
    ├── Website/
        ├── index.html
        ├── script.js
    ├── Pipfile
    ├── Pipfile.lock
```

Implementation of the Contact Organizer Application on AWS

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

Extraction Service – contact information extraction

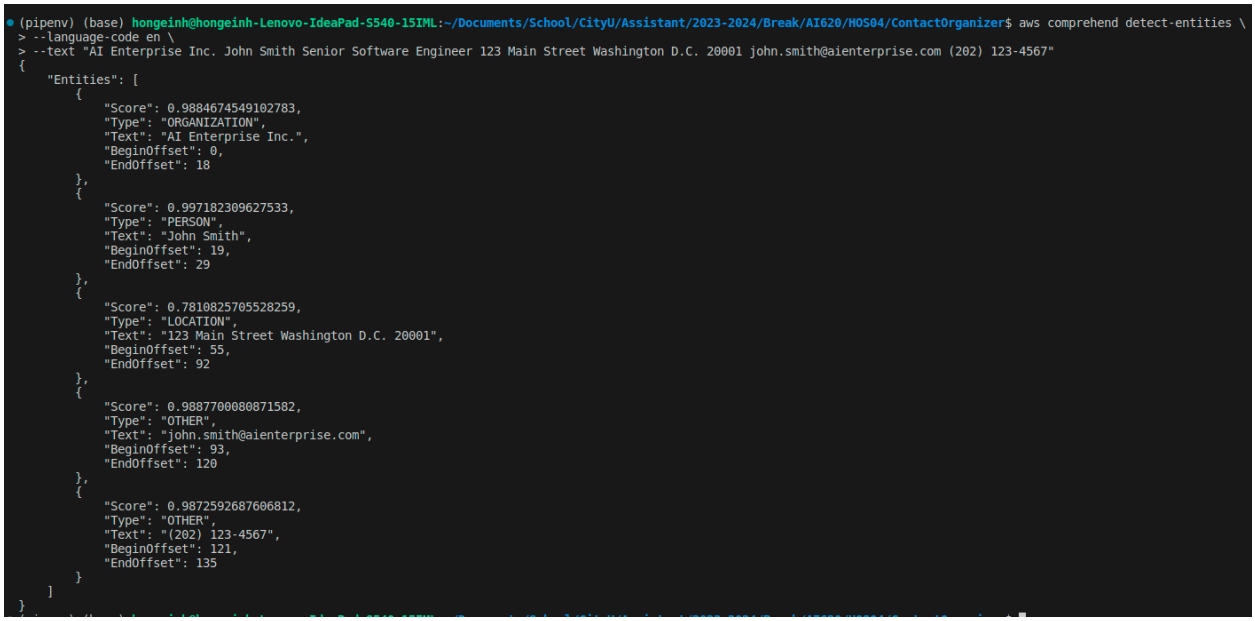
Let’s extract the following contact information to leverage Amazon Comprehend

```
AI Enterprise Inc.
John Smith
Senior Software Engineer
123 Main Street Washington D.C. 20001
john.smith@aienterprise.com
(202) 123-4567
```

a. Open your terminal / command prompt and type the following command to start a transcription.

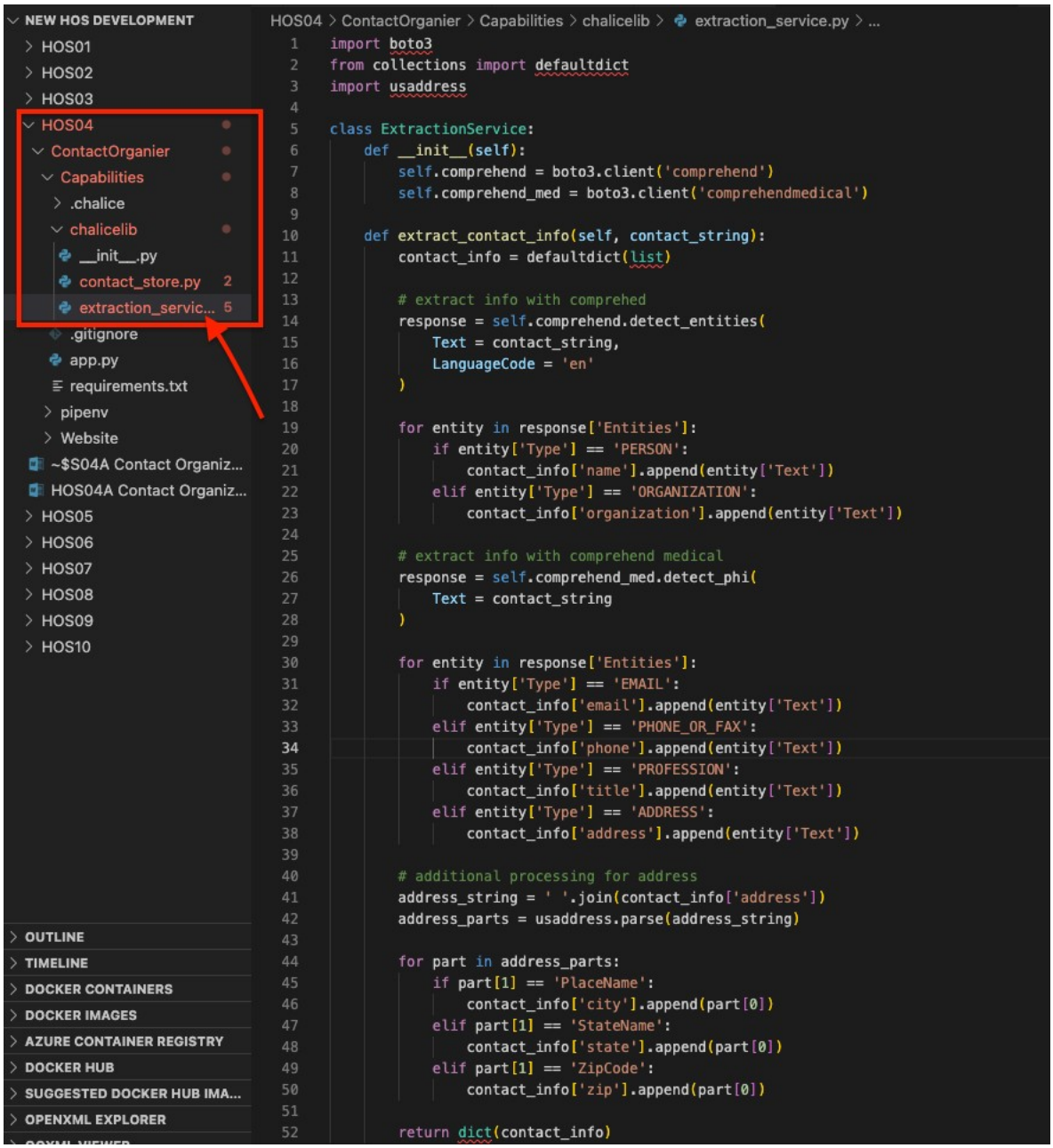
```
aws comprehend detect-entities \
--language-code en \
--text "AI Enterprise Inc. John Smith Senior Software Engineer
123 Main Street Washington D.C. 20001
john.smith@aienterprise.com (202) 123-4567"
```

Amazon Comprehend extracted some pieces of information like job title (PROFESSION), the phone number (PHONE_OR_FAX), the email (EMAIL), and address (LOCATION).



Implement Extraction Service

a. Go to the `chalicelib` directory, create a new file `extraction_service.py` and type the following to create a Python class named `ExtractionService`.



- The `extract_contact_info()` method calls both variants of Amazon Comprehend through `boto3`. The results from both calls are processed and stored in the `contact_info` dictionary.
- `contact_info` is declared as a `defaultdict(list)` ,which is a dictionary data structure where the values are defaulted to an empty list.

HOS submission instructions

1. Please install the 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