# Table of Contents


1. User Manual
    1. Introduction – ALFRED Requirement Generation System
    2. How to Run the System
        - Step-by-Step Access Guide
            - i. Space Access
            - ii. System Interface
            - iii. OpenAI API Key Setup
            - iv. API Key Creation
            - v. API Key Input
            - vi. Pipeline Execution
            - vii. Log Output Monitoring
            - viii. Results & Downloads
    3. More Details About How the System Processed
        - System Deployment
        - JSON-based Persona Requirements
        - API Configuration
        - System Output
    4. Output Files
        - JSON Structure
        - User Story Attributes
    5. Troubleshooting
        - Common Issues
    6. Contact / Support
    7. Acknowledgment

2. Folder Structure
    - Project Overview
    - Input Files
    - Processing Modules

- Output Structure

# User manual

## Introduction – ALFRED Requirement Generation System

Context: ALFRED system is virtual assistant platform designed to support multiple older adults in living independently, maintaining their physical and mental health, and staying socially connected with other older adults in the system, and receiving effective and efficient care

This application is a solution to automate the generation of system requirements (a.k.a user stories) and usage the information provided from different personas, which can be:

Devlopers/App Creators: They are responsible for designing, developing, and maintaining the features of the ALFRED system

Caregivers/Medical Staff: They use the ALFRED as a way to provide appropriate healthcare to their patients

Older adults: They are the end-users of the ALFRED system (e.g. patients, olders, ...), which needs help to maintain the health physically and mentally

After generating the user stories for different personas, the application will check for any potential conflicts/inconsistencies between the user stories, then resolve them appropriately to ensure a consistent, comprehensive list of the ALFRED system requirements

## How to Run the System

Run immediately by accessing Huggingface with link given:

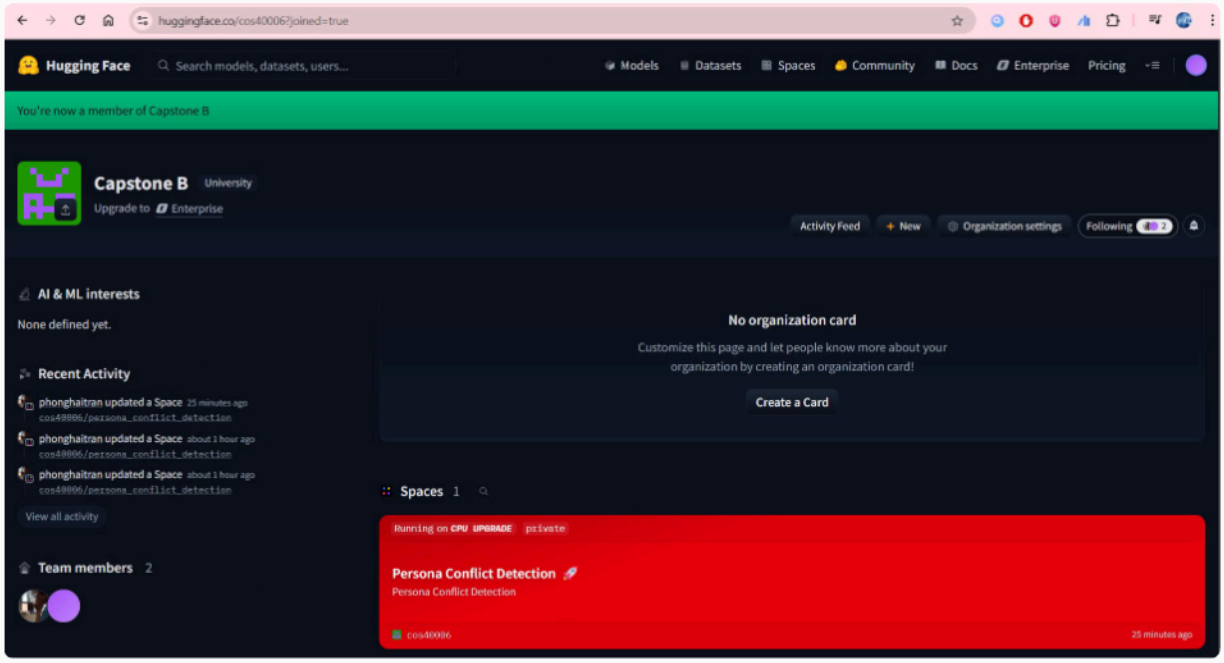
Our organisation page: https://huggingface.co/cos40006

i. Access to our organization page, then please click on the Space named Persona Conflict Detection, or you can access the space by hit the link below:

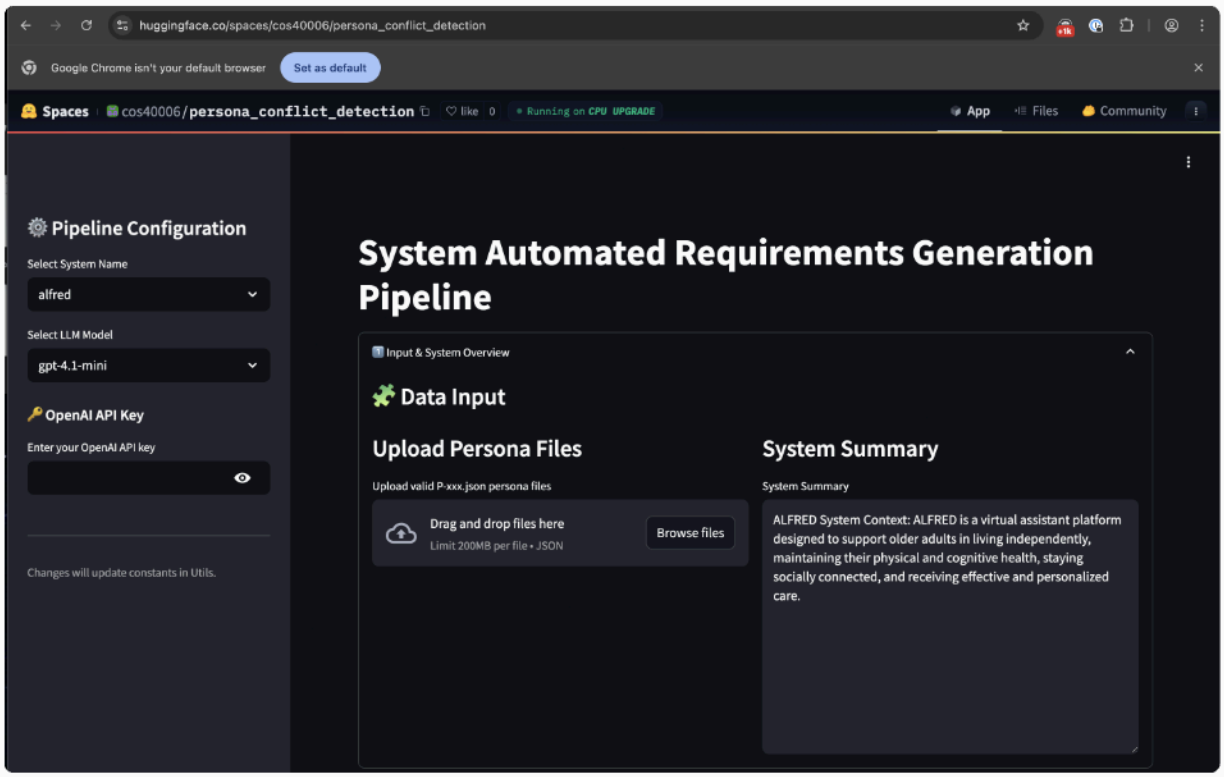
**Persona Conflict Detection - a Hugging Face Space by cos40006**

Persona Conflict Detection

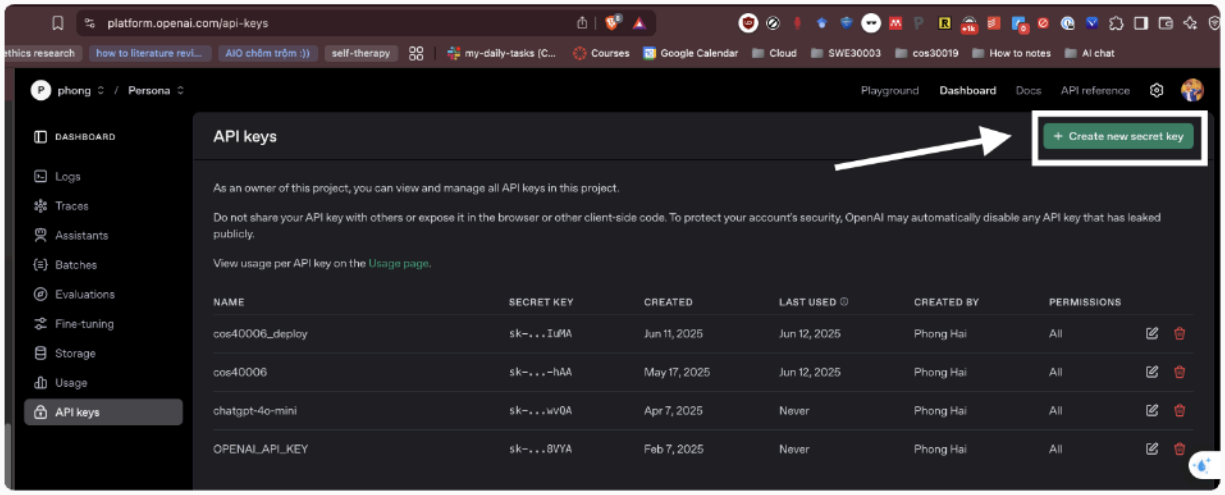
https://huggingface.co/spaces/cos40006/persona_conflict_detection



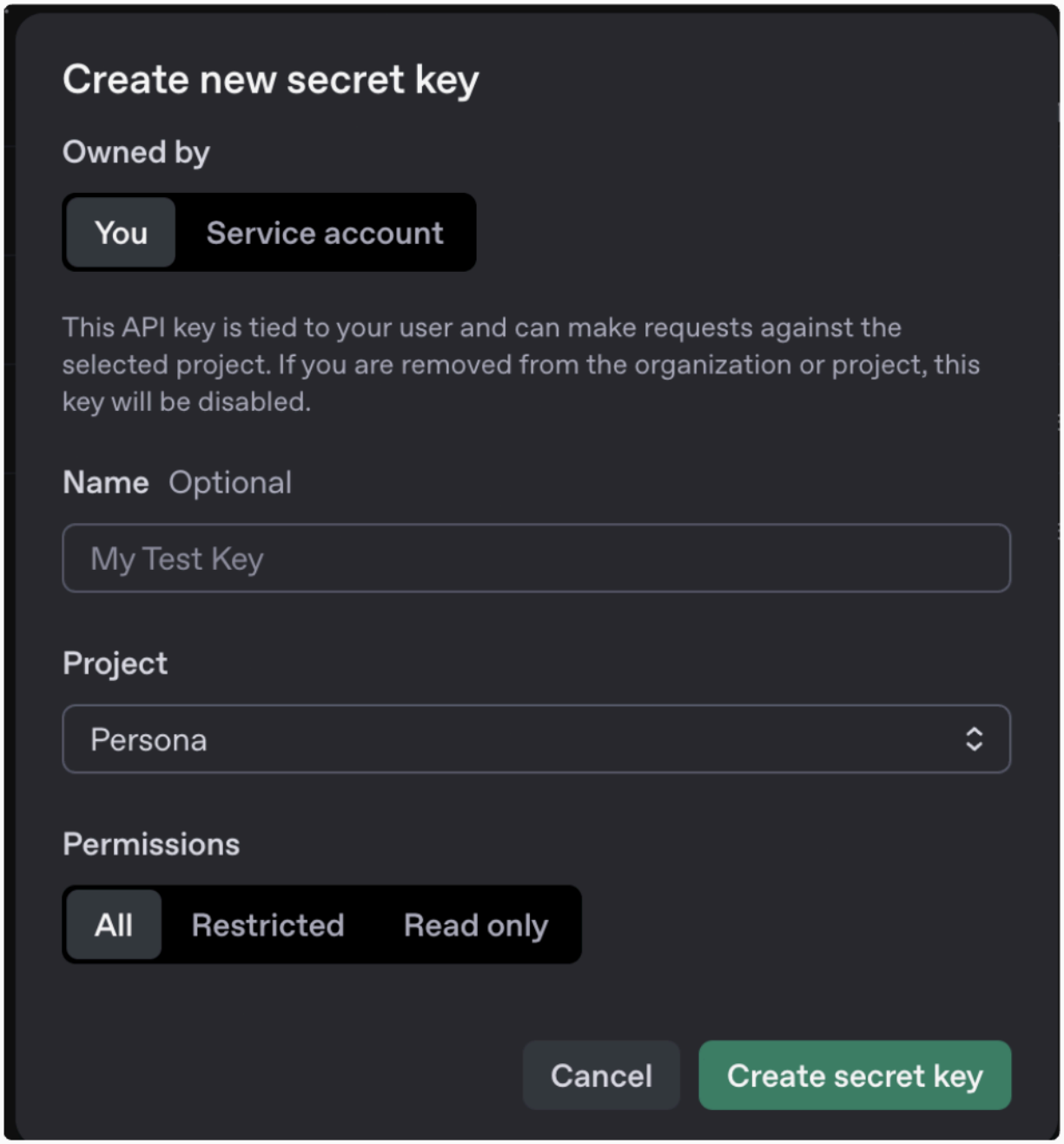

ii. You can see the image below that the system that has been deployed before, so you can access and operate the running system.



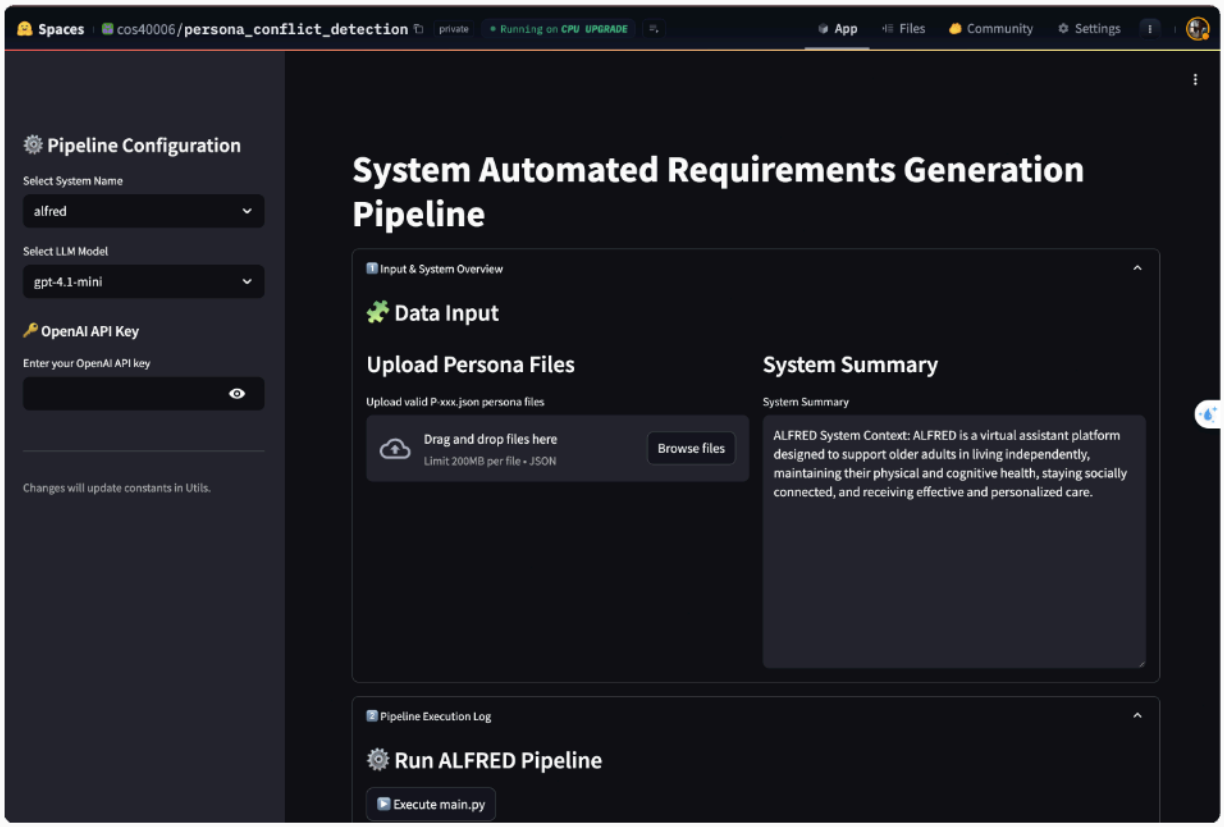

iii. Input your own API key from OpenAI platform at: https://platform.openai.com/api-keys, with the interface below that the website accessing the API Keys at the left sidebar with pointing out that click on the Create new secret key to show the pop up.



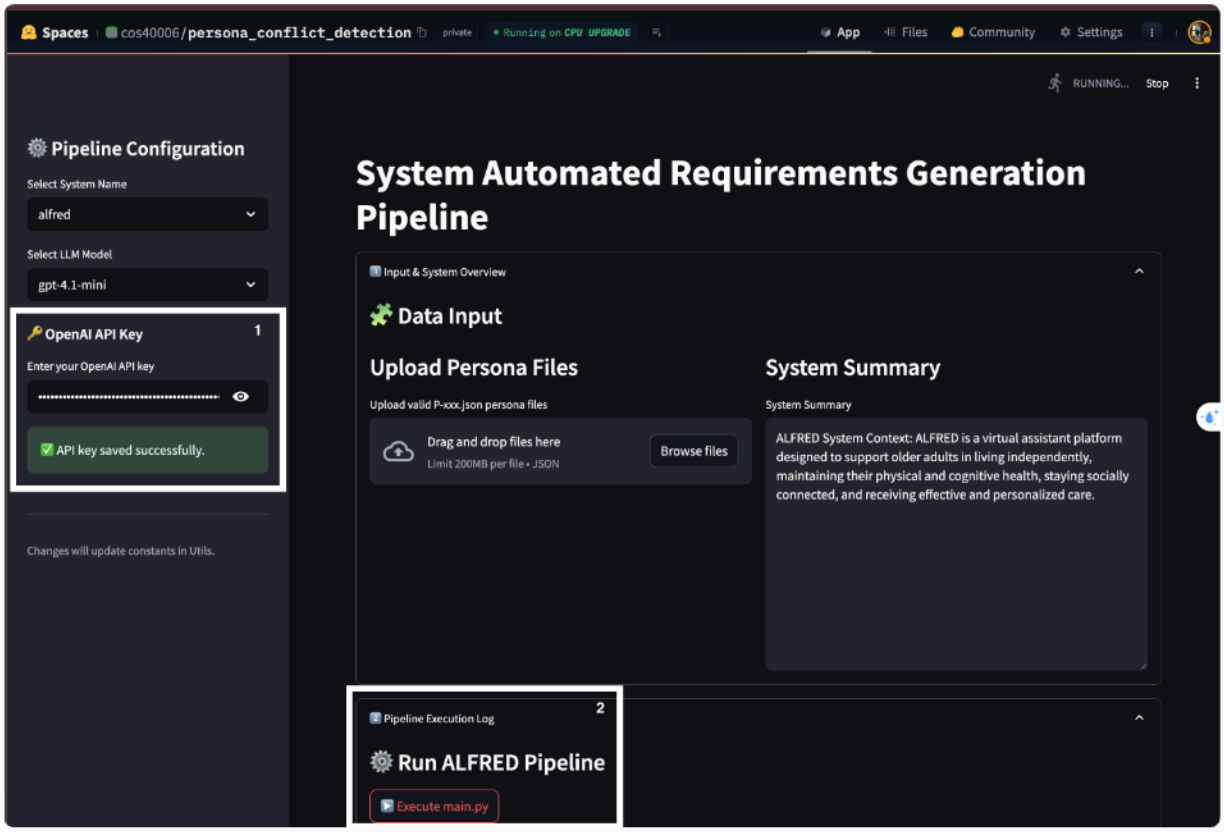

iv. Fulfill all details that the form said, exactly like the image given below.



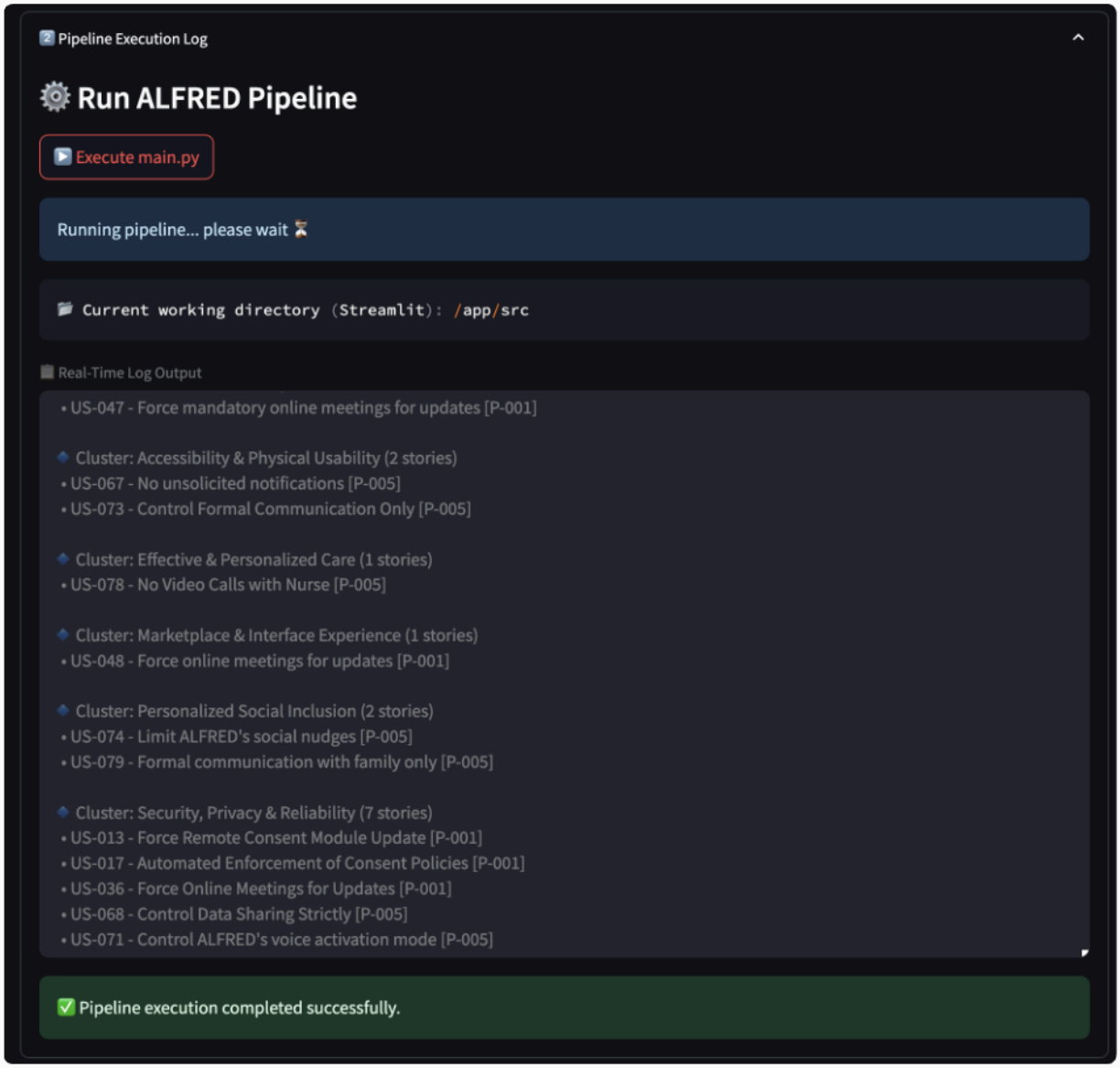
v. After having the secret key, at the space on the Huggingface go the input at the OpenAI API Key and Enter the OpenAI API Key at the sidebar as your created API key.



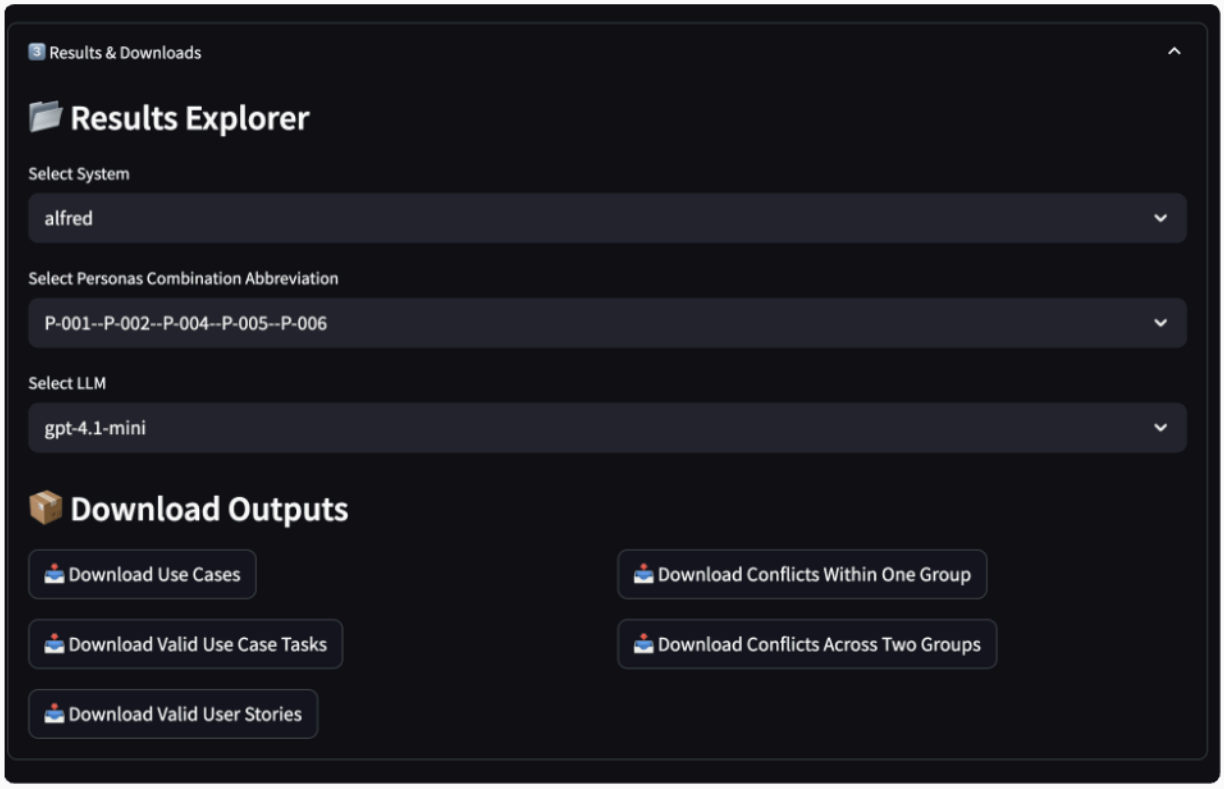

vi. After fulfilling the OpenAI API Key with input box, it is optional to make a change on other input form like Upload Persona Files, System Summary. Then click Execute main.py to run the pipeline.




vii. After executing, you can see the Log Output has been displayed with the process sequence of pipeline. Then display Pipeline execution completed successfully when finishing all Log Output




viii. After completing successfully the log output running. Results & Downloads will display all dropdown with given Select System, Personas combination, LLM model to show all the Download Ouputs that can be able to see the actual result in JSON.




## More details about how the system processed

1. The system is deployed as a web application. The user needs to visit to the website's link, upload the JSON-based personas.

2. The JSON-based personas must includes the following attributes:

a. Id (string)
b. Name (string)
c. Role (string)
d. Tagline (string)
e. DemographicData (dist)
f. CoreCharacteristics (list)
g. CoreGoals (list)
h. TypicalChallenges (list)
i. Singularities (list)
j. WorkingSituation (list)
k. PlaceOfWork (list)
l. Expertise (list)

3. The OpenAI's API (with available credits) key must be typed correctly to a fieldset

4. The system returns of list of consistent and comprehensive system requirements (user stories) for the ALFRED system, as a JSON file. The user may click on the "Download"

button to save it to the local device.

## Output files

The final output files are a JSON-based ones ("Download Valid User stories"), containing the lists of user stories (after handling all conflicts) for all personas, each has following attributes:

- Id: Unique identification of the user story
- Title: The title of the User story
- Type: Functional or non-functional user story
- Cluster: The cluster that the user story is classified into
- Summary: The description of the User story
- Priority: The priority of the user story, according to the ALFRED's definintion (from 1 to 5)
- Pillar (optional): The pillar in which the user story belongs to
- UserGroup: The user group of the persona, specifying in PersonaId
- PersonaId: The identification of the persona who the user story belongs to
- useCases: Associated use case(s) that the user story is generated from

Besides, there are also downloadable JSON files, saving the generated use cases, extracted tasks from use cases, and handled conflicts before retrieving the final lists of valid user stories

## Troubleshooting

The following issues are common:

- API key not found: Ensure that the API key exists, provided by appropriate service (e.g. OpenAI)
- Not sufficient credits: Ensure that the account associated with the provided API key contains enough credits
- Output is empty or generic: Double-check input persona for missing fields

## Contact / Support

For any questions or unresolvable issues, please contact one of the following emails:


- 104053642@student.swin.edu.au (Trung Kien Nguyen)

- 103830572@student.swin.edu.au (Quoc Co Luc)
- phonghaitran.work@gmail.com (Phong Tran)


## Acknowledgment


This User Manual was developed by the project development team with limited assistance from OpenAI's GPT-4.1-mini model


# Folder Structure

```
cos40006/
├── .github/workflows/deploy.yml
├── src/
│   ├── data/
│   │   ├── alfred/
│   │   │   ├── personas/ (sample & uploaded) ← Input persona files
│   │   │   ├── pillars/ (Pi-001 to Pi-006)
│   │   │   ├── use_case_rules/
│   │   │   ├── user_groups/ (UG-001 to UG-003)
│   │   │   ├── user_story_conflict_rules/
│   │   │   └── user_story_rules/
│   │   └── llm_response_language_proficiency_level.txt
│   ├── pipeline/
│   │   ├── result_analysis/ (6 modules)
│   │   ├── ui/ (5 modules)
│   │   ├── use_case/ (6 modules)
│   │   ├── user_story/ (8 modules)
│   │   ├── user_story_conflict/ (10 modules)
│   │   ├── main.py
│   │   ├── test.py
│   │   └── utils.py
│   ├── results/alfred/P-001--P-002--P-004--P-005--P-006/gpt-4.1-mini/ ← Output
│   │   ├── result_analysis/ (analysis files)
│   │   ├── tasks/ (extracted & unique)
│   │   ├── use_cases/ (UC-001 to UC-015)
│   │   ├── user_stories/ (duplicated & unique)
│   │   └── user_story_conflicts/ (conflicts & resolutions)
│   └── api_key.txt ← Required API key
├── .gitignore
├── README.md
└── requirements.txt
```