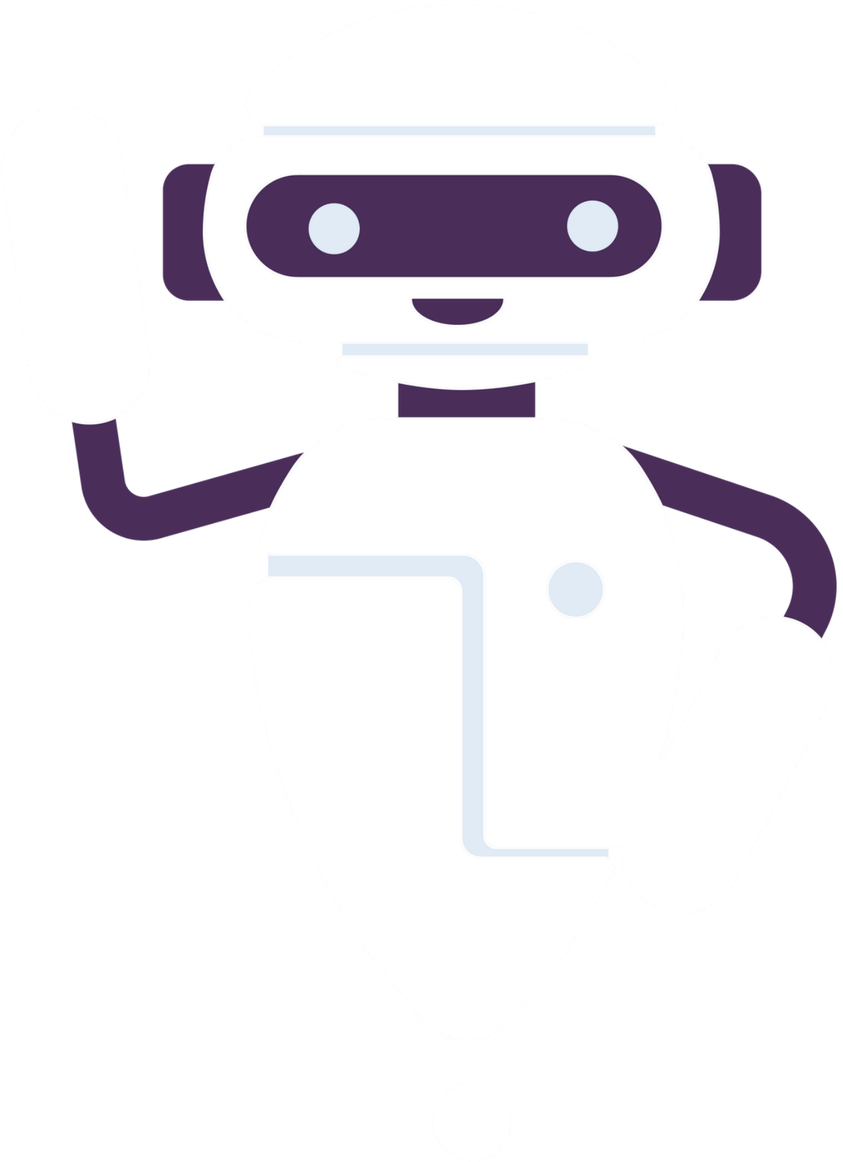
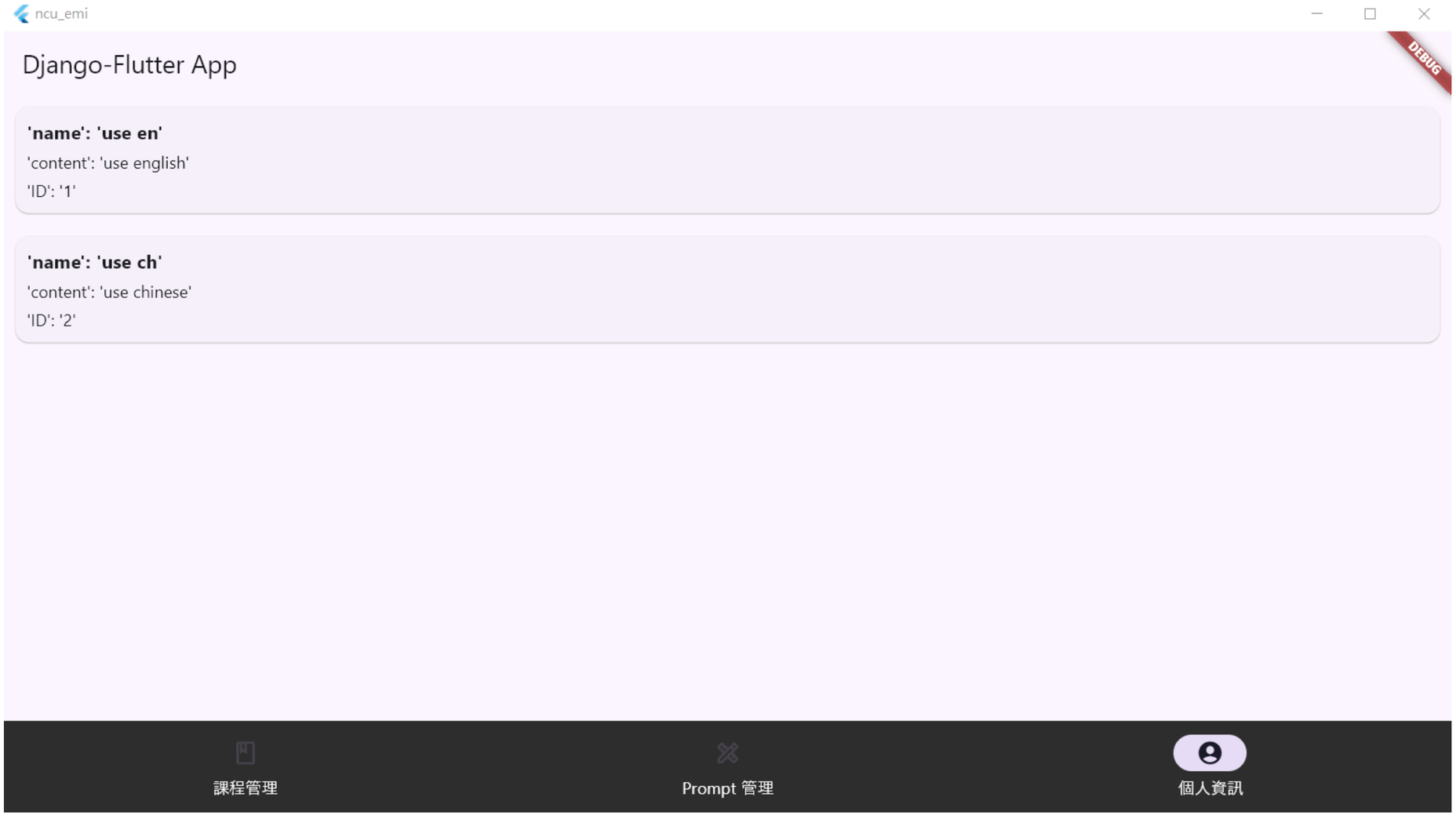
EMI教學助手



get


ncu_emi
Django-Flutter App
DEBUG
'name': 'use en'
'content': 'use english'
'ID': '1'
'name': 'use ch'
'content': 'use chinese'
'ID': '2'
課程管理
Prompt 管理
個人資訊

```python
api_endpoint = "https://api.openai.com/v1/vector_stores"
api_key = os.environ.get('OPENAI_API_KEY')

# 定義請求標頭
headers = {
    "Authorization": f"Bearer {api_key}",
    "OpenAI-Beta": "assistants=v2"
}

# 發送GET請求以獲取所有的vector stores
response = requests.get(api_endpoint, headers=headers)

vector_id = None
# 檢查請求是否成功
if response.status_code == 200:
    vector_stores = response.json()
    # 列出所有的vector stores
    for store in vector_stores['data']:
        print(f"ID: {store['id']}, Name: {store['name']}")
        vector_id = store['id']
else:
    print(f"Failed to retrieve vector stores: {response.status_code} {response.text}")
```

✓ 0.2s

```
ID: vs_ZbiGyuz5pe1HnV6xXZ5HXPBS, Name: Algorithm
```

```python
vector_store = client.beta.vector_stores.create(name="Algorithm")

file_paths = ["改良QuickSort作業說明_更正.pdf","Eclipse安裝與輸出說明.pdf","chapter5.pdf"]
file_streams = [open(file_path, "rb") for file_path in file_paths]

file_batch = client.beta.vector_stores.file_batches.upload_and_poll(
  vector_store_id=vector_store.id, files=file_streams
)

print(file_batch.status)
print(file_batch.file_counts)
```

```
completed
FileCounts(cancelled=0, completed=3, failed=0, in_progress=0, total=3)
```

```python
assistant = client.beta.assistants.update(
  assistant_id=assistant.id,
  tool_resources={"file_search": {"vector_store_ids": [vector_id]}},
)
```

✓ 0.3s

```python
thread = client.beta.threads.create(
  messages=[
    {
      "role": "user",
      "content": "briefly explain the homework assignment."
    }
  ]
)
```

```python
run = client.beta.threads.runs.create_and_poll(
    thread_id=thread.id, assistant_id=assistant.id
)

messages = list(client.beta.threads.messages.list(thread_id=thread.id, run_id=run.id))
print(messages[0].content[0].text.value)
```

```
The homework assignment involves implementing a modified version of the Quicksort algorithm in Java, based on pseudocode provided in the c

1. **Implementation**:
   - Implement the Quicksort and Insertion Sort algorithms in Java using the pseudocode from the textbook (mentioned sources: Chapter 5, p
   - Ensure that when the subarray size is less than or equal to 4, the algorithm switches from Quicksort to Insertion Sort to enhance eff

2. **Functions**:
   - Create three functions: `quickSort()`, `hoarePartition()`, and `insertionSort()`.
   - Additional helper functions may be defined but must include proper comments explaining their use.

3. **Printing Requirements**:
   - Print the array content before starting and after finishing the sorting process.
   - After each call to `hoarePartition()` or `insertionSort()`, print the method used and the current array content.
   - Use specific print statements like "Use partition:" or "Use Insertion:" followed by the array content, ensuring conformity to the giv

4. **Input Arrays**:
   - Use the provided arrays as input, and ensure the output matches the given results exactly.

5. **Submission**:
   - Name your project "Quicksort_StudentID" and the main class "Main".
   - Use JavaSE-17 for compiling the project.
   - Encode text files in UTF-8 to avoid character misinterpretation.
   - Submit the completed project as a zip file named "Quicksort_StudentID.zip" through the EE-Class workspace by the specified deadline (

6. **Coding Standards**:
   - Add appropriate comments to your code for readability and ease of grading.
   - Do not copy or plagiarize code from classmates or the internet.

By following these guidelines, you will ensure that your implementation meets the assignment requirements and receives full credit  .
```