

Multi-Agent AI Application

Enter Technical Requirements

Create a python module to create fake data for user address book information

Submit Requirements

Status:

Status: Starting

- D You can implement the Python module for creating fake user address book information using the Faker library. Below is a sample code snippet for creating this module:

```
from faker import Faker

fake = Faker()

def generate_address_book_entries(num_entries):
    address_book = []
    for _ in range(num_entries):
        entry = {
            'name': fake.name(),
```

```
'address': fake.address(),
'phone_number': fake.phone_number(),
'email': fake.email()
}
address_book.append(entry)
return address_book

if __name__ == '__main__':
    num_entries = 10
    address_book_data = generate_address_book_entries(num_entries)
    for entry in address_book_data:
        print(entry)
```

You can install the Faker library by running `pip install faker` in your Python environment. This code snippet uses the Faker library to generate fake name, address, phone number, and email for each entry in the address book. The `generate_address_book_entries` function creates a list of user address book entries and returns it.

You can modify the fields or add more fields as per your requirements. You can also adjust the number of entries generated by changing the `num_entries` variable.

R Improvements: The code is well-structured and follows best practices. One potential improvement could be to add type hints for the parameter and return types in the function definition to enhance code readability and maintainability.

Status: Reviewer is reviewing code

D Based on the improvements provided, I have added type hints to the function definition to enhance code readability and maintainability. Here is the updated code:

```
from faker import Faker
from typing import List, Dict

fake = Faker()

def generate_address_book_entries(num_entries: int) -> List[Dict[str, str]]:
    address_book = []
    for _ in range(num_entries):
        entry = {
```

```
        'name': fake.name(),
        'address': fake.address(),
        'phone_number': fake.phone_number(),
        'email': fake.email()
    }
    address_book.append(entry)
return address_book

if __name__ == '__main__':
    num_entries: int = 10
    address_book_data = generate_address_book_entries(num_entries)
    for entry in address_book_data:
        print(entry)
```

By adding type hints, it becomes clearer what types of parameters the function expects and what type of value it returns. This can help in understanding the code more easily and in detecting errors during development.

R Improvements: Add type hints to function definition for better code readability and maintainability.

Status: Reviewer is reviewing code

D Your code has been successfully updated with the requested improvements. The type hints added to the function definition enhance code readability and maintainability. The code is now improved to include type hints for function parameters and return types.

R Improvements: The code snippet provided adequately demonstrates the use of type hints in function definitions which is a good programming practice. No additional improvements are necessary at this time.