Phone Book Manager Application Documentation

GITHUB LINK:

https://github.com/F-saiyed/Full_stack/tree/main/Assignment_1

Overview

The **Phone Book Manager** is a command-line application that allows users to manage a collection of contacts. Users can perform operations such as adding new contacts, viewing contacts, searching for contacts, updating contact details, deleting contacts, and importing contacts from a CSV file. The application is built using Python and consists of three main components:

- Contact Class: Represents individual contact entries.
- Phonebook Class: Manages the collection of contacts.
- A command-line interface (main.py) for user interaction.

```
#Step 1: Setting Up the Contact Class
```

The **Contact** class represents an individual entry in the phonebook. It contains the contact's basic information such as first name, last name, phone number, email, and address. Additionally, it maintains timestamps for when the contact was created and last updated.

Attributes:

- first_name (str): The first name of the contact.
- last_name (str): The last name of the contact.
- phone_number (str): The phone number of the contact.
- email (str, optional): The email address of the contact.
- address (str, optional): The physical address of the contact.
- created_at (datetime): The timestamp when the contact was created.
- updated_at (datetime): The timestamp when the contact was last updated.

Methods:

- __init__(self, first_name, last_name, phone_number, email=None, address=None): Initializes a new contact object with the provided attributes.

- update (self, first_name=None, last_name=None, phone_number=None, email=None, address=None): Updates the contact's information. If a parameter is provided, it updates the corresponding field; if not, the old value remains unchanged. The updated_at timestamp is refreshed after any update.
- __str__(self): Returns a string representation of the contact in the format: First Last Name, Phone: <phone_number>, Email: <email>, Address: <address>.

```
#Step 2: Setting Up the Phonebook Class
```

The **Phonebook** class is responsible for storing and managing a collection of contacts. It provides methods for adding, updating, deleting, and listing contacts. Additionally, it includes functionality to search for contacts based on specific fields and to import contacts from CSV files.

Attributes:

- contacts (list): A list that stores instances of the Contact class.

Methods:

- __init__(self): Initializes an empty PhoneBook.
- add_contact(self, contact): Adds a new contact to the phone book.
- import_contacts_from_csv(self, file_path): Imports contacts from a CSV file. Each row in the CSV is converted into a Contact object and added to the phone book.
    - **CSV Format**:
        - The CSV file should contain the following columns:

```
First_Name,Last_Name,Phone_Number,Email_Address,Address
Uzair,Saiyed,(123) 456-7890,uzair.saiyed@gmail.com,123 Elm Street
Saniyah,Saiyed,(987) 654-3210,saniyah.saiyed@gmail.com,456 Oak Avenue
Vibhu,Puri,(343) 678-3452,vibhu.puri@uottawa.ca,1200 Bank Street
Mohit,Upadhyay,(613) 900-9453,mohit.upa@uottawa.ca,Lincoln Field
Fehmida,Saiyed,(343) 456-2856,fsaiyed@uottawa.ca,Huntclub Road
Avani,Purohith,(613) 677-9898,apuro56@uottawa.ca,Merivale Club
```

- find_contact (self, **kwargs): Searches for contacts in the phone book based on provided keyword arguments (e.g., first_name, last_name, phone). It returns a list of matching contacts.

- update_contact (self, contact, **kwargs): Updates an existing contact's details. Accepts keyword arguments to specify which fields (first_name, last_name, phone_number, email, address) to update.
- delete_contact (self, contact): Deletes the specified contact from the phone book.
- list_contacts (self): Returns a sorted list of all contacts, ordered by last name.
- log_operation (self, operation): Logs operations (e.g., adding, deleting contacts) to a phonebook.log file with a timestamp.

```
#Step 3: Command-Line Interface
```

The **main.py** file provides the user interface, enabling interaction with the phonebook through the terminal. It uses a simple text-based menu system to guide the user through various operations, such as adding new contacts, viewing contacts, searching, and importing contacts from a CSV file.

Menu Options:

1. **Add Contact**:
    - Prompts the user to enter first name, last name, phone number, email, and address.
    - Creates a new Contact object and adds it to the phone book.
2. **View Contacts**:
    - Displays a list of all contacts in the phone book in a tabular format.
    - If there are no contacts, a message is displayed.
3. **Search Contacts**:
    - Prompts the user to specify a search field (first name, last name, or phone) and a search value.
    - Uses the find_contact method to search for matching contacts and displays the results.
4. **Update Contact**:
    - Prompts the user to search for a contact by first name, last name, or phone.
    - If a matching contact is found, the user is prompted to enter new details (or leave blank to keep the current information) for first name, last name, phone number, email, and address. The contact is then updated.

5. **Delete Contact**:
    - Prompts the user to search for a contact by first name, last name, or phone.
    - If a match is found, the contact is removed from the phone book.
6. **Import Contacts from CSV**:
    - Prompts the user for the file path of a CSV file.
    - Imports contacts from the CSV file and adds them to the phone book.
7. **Exit**:
    - Exits the Phone Book Manager.

Utility Functions (main.py):

- display_contacts (contacts): Displays a formatted list of contacts with columns for ID, first name, last name, phone number, email, and address. If no contacts are found, it prints "No contacts found."

Run the application:

- In your terminal or command prompt, navigate to the directory where the files are located and run the main.py script:

  python main.py

Conclusion:

The **Phone Book Manager** is a fully functional, easy-to-use contact management system. It allows users to perform essential contact operations efficiently, with potential for further enhancement. This modular, well-organized application is suitable for managing small to medium-sized collections of contacts directly from the command line.