**Phone Book Management Application Report**

Xinye Zhu (300367809)

**Overview**

The Phone Book Management Application is designed to manage contacts efficiently. It is constituted of three main files: contacts.py, phone\_book.py, and app.py. Each file contributes to the overall functionality of the application, from defining data structures to implementing core functionalities and initializing the application. It also has a database.json file that stores added contacts information and keeps track of history of changes, and a phonebook\_log.log file that records all operations performed in the application along with timestamps.

**1. Data Structures**

**1.1 Contact Class (contacts.py)**

* **Attributes:**
  + \_first\_name: The first name of the contact.
  + \_last\_name: The last name of the contact.
  + \_phone\_number: The contact's phone number.
  + \_email\_address: The contact's email address.
  + \_address: The contact's physical address.
  + \_create\_time: The timestamp when the contact was created.
  + \_update\_time: The timestamp when the contact was last updated.
  + \_history: A list of Change objects representing the history of changes made to the contact.
* **Constructor:**
  + Use provided arguments to create a new contact.
  + For new contacts created by the user, use user input to create the contact and sets its \_create\_time and \_update\_time to be the current time.
  + For contacts imported from database.json, use existing read-in information to create the contact.
* **Methods:**
  + Getters for each attribute
  + Setters for each attribute: If the provided argument would change corresponding argument, make changes and update \_update\_time. If not, output message to indicate no changes are made.
  + print\_history(): Prints history of change of the contact.
  + to\_dict(): Converts the contact object to a dictionary. Used in PhoneBook export\_contacts\_to\_json() to write the Contact object to a json file.
  + from\_dict(): Creates a contact object from a dictionary. Used in PhoneBook import\_contacts\_to\_json() to read a Contact object from a json file.

**1.2 Change Class (contacts.py)**

* **Attributes:**
  + operation: The type of operation performed (“Created” or “Updated”).
  + message: Only used to record information of operation “Created”.
  + field: The field that was changed. Only used for operation “Updated”.
  + old\_value: The old value of the field. Only used for operation “Updated”.
  + new\_value: The new value of the field. Only used for operation “Updated”.
  + change\_time: The timestamp when the change was made. Should be the current time when the Change object is created.
* **Methods:**
  + print(): Prints the time, operation, and content of the change. Called in Contact print\_history().
  + to\_dict(): Converts the change object to a dictionary. Used for writing into json file.
  + from\_dict(): Creates a change object from a dictionary. Used for reading from json file.

**1.3 PhoneBook Class (phone\_book.py)**

* **Attributes:**
  + contacts: A list of Contact objects. Stores all the contacts added by the user.
* **Methods:**
  + print\_all\_contacts(): Prints all contacts in a tabulated format.
  + print\_contact(): Prints a single contact in a tabulated format.
  + print\_contact\_list(): Prints a given list of contacts in a tabulated format.
  + is\_valid\_email(): Checks whether input email address is valid using regex.
  + is\_valid\_phone\_number(): Checks whether input phone number is valid using regex. This application forces the user to enter phone number in the format (###) ###-####.
  + is\_contact\_exist(): Checks whether a contact with a specific name exists in the contacts list.
  + input\_mandatory\_field(): Prompts the user for a valid input if the input is empty. Used for mandatory fields such as first name, last name, and phone number.
  + create\_contact(): Prompts the user to create new contacts manually or batch create from a CSV file.
  + search\_contact(): Searches for contacts based on user input. Enables the user to select and view contact’s history of changes when complete the search.
  + update\_contact(): Updates an existing contact based on user input.
  + delete\_contact(): Deletes contacts based on user input or batch deletes from a CSV file.
  + sort\_contacts(): Sorts contacts based on various criteria.
  + group\_contacts(): Groups contacts by the first letter of their last name.
  + export\_contacts\_to\_json(): Exports the contacts list to a JSON file.
  + import\_contacts\_from\_json(): Imports contacts from a JSON file.

**1.4 Application Initialization (app.py)**

* **Logging Initialization:**
  + init\_logging(): Initializes logging with a specific format and log level. Configs the logger to log into the file phonebook\_log.log.
* **Main Function:**
  + main(): The entry point of the application. It initializes logging, creates an instance of PhoneBook, imports contacts from a JSON file (database.json), and provides a menu for the user to interact with the phone book management system through the created PhoneBook object. When the user quits the application, exports contacts to database.json.

**2. Functionalities**

**2.1 Application Overview**

The Phone Book Management Application supports CRUD operations on contacts, searching, sorting, grouping, update history recording, and logging. It also supports partial match in searching and validation in certain required fields. Users can access app functionalities through running main.py. The application stores contacts information and history of changes in database.json and keeps logs in phonebook\_log.log.

**2.2 Rules**

* This application uses full name (First Name + Last Name) to uniquely identify contacts. Duplicated contacts with the same name (case insensitive) are not allowed when creating and updating.
* For each contact, First Name, Last Name, and Phone Number are mandatory fields. Email Address and Address are optional, could be empty.
* Phone Number must be entered in the format (###) ###-####. This application does not accept any other format.
* Email Address should be in the format (user\_name)@(domain\_name).(top-leveldomain).
* Date should be in the format yyyy-mm-dd.
* Batch create: For csv file, each row should be: first\_name, last\_name, phone\_number, email\_address, address. The file should not contain header.
* Batch delete: For csv file, each row should be: first\_name, last\_name. The file should not contain header. Batch delete does not support partial match, contact names should exactly match (case insensitive).
* Partial match supported in: Search Contact by name or phone number, search for contact to update in Update Contact, search for contact to be manually deleted in Delete Contact
* For every functionality, the application will prompt until the user enters a valid input or chooses to quit.

**2.3 Functionality details**

Following functionalities are realized by corresponding PhoneBook methods.

**[1] Print Contact**

* Prints all the contacts in the system in a tabulate format.
* Prints notification if there’s no contact in the system.

**[2] Create Contact**

Users can choose from creating contacts one by one manually or batch create with a csv file.

* Manually
  + Performs input validation. Creates contacts when all valid information is provided.
  + Performs duplicated contacts validation.
* Batch Create
  + Performs validations, output messages indicating success or failure for creating contact with each row.
  + Outputs total number of successful contact creation.

**[3] Search Contact**

Users can search by contact name, phone number, or date created.

* Name: Supports partial match, case insensitive.
* Phone number: Supports partial match. Application will process the query to only use digitals for searching.
* Date created: Users may enter two dates to search for contacts created between them (inclusively) or enter only one date to search for contacts created within that date.

**[4] Update Contact**

Users first perform a search for contacts to be updated, then select the contact they want to modify and the attribute to be updated.

* Search for contacts by name (first name, last name, or full name). Supports partial match.
* Attributes to be updated: First Name, Last Name, Phone Number, Email Address, Address
* Performs input validation and duplicated contacts validation.

**[5] Delete Contact**

Users can choose to delete contacts one by one manually, batch delete using csv, or delete all contacts.

* Manually
  + Users first perform a search for contacts to be deleted, then select the contact they want to delete.
  + Search for contacts by name (first name, last name, or full name). Supports partial match.
* Batch delete
  + Does not support partial match, contact names should exactly match (case insensitive).
  + Outputs messages indicating success or failure for creating contact with each row.
  + Outputs total number of successful contact deletion.
* Delete all
  + Ask users to confirm before performing the deletion.

**[6] Sort Contact**

Users first select how to sort the contacts (by first name, last name, phone number, create time, or update time), then select the sorting order (ascending or descending). Sorting outcome will be saved in the database.

**[7] Group Contact**

Group contacts by the first letter or last name, then print the groups in a tabulate format. This won’t be written into database.

**3. Important Files**

* database.json: Stores all contacts information and history of changes. To be imported from at the start of application and exported when user quit the application. Please don’t modify.
* phonebook\_log.log: Keeps track of all application activities. Please don’t modify.
* add\_old.csv: Used to store contacts into database for demonstration purposes.
* add\_new.csv: Used to test batch create contacts.
* add\_invalid.csv: Used to test batch create contacts validation.
* delete.csv: Used to test batch delete.

**Conclusion**

The Phone Book Management Application is a comprehensive system for managing contacts. It utilizes well-defined data structures and provides a wide range of functionalities, including reading, creating, searching, updating, deleting, sorting, and grouping contacts. The application is user-friendly and can be easily extended to include additional features as needed.