**Address Book Project**

Project Overview:

Build an address book application that allows users to store and manage contact information.

Features:

Add a Contact:

Allow users to add a new contact with details such as name, phone number, email, and address.

View All Contacts:

Display a list of all contacts in the address book.

Search for a Contact:

Implement a search functionality to find a contact by name.

Update Contact Information:

Allow users to update the details of an existing contact.

Delete a Contact:

Provide an option to delete a contact from the address book.

Save/Load Address Book to/from File:

Implement functionality to save the current state of the address book to a file (e.g., in JSON format) and load it back.

**Guidelines:**

Use Classes:

Create a Contact class to represent an individual contact with attributes like name, phone number, email, and address.

Use a Dictionary for the Address Book:

Use a dictionary to store contacts, where the key is a unique identifier (e.g., name) and the value is an instance of the Contact class.

**File Handling:**

Implement functions to save the address book to a file and load it back. You can use JSON for serialization and deserialization.

**User Interface:**

Create a simple text-based interface that allows users to interact with the address book (e.g., using a command-line interface).

**Function Decomposition:**

Decompose the functionalities into functions/methods. For example, you might have functions like add\_contact, view\_contacts, search\_contact, update\_contact, delete\_contact, save\_to\_file, and load\_from\_file.

**Example Structure:**

class Contact:

def \_\_init\_\_(self, name, phone, email, address):

# Initialize contact attributes

class AddressBook:

def \_\_init\_\_(self):

# Initialize an empty dictionary to store contacts

def add\_contact(self, contact):

# Add a new contact to the address book

def view\_contacts(self):

# Display all contacts

def search\_contact(self, name):

# Search for a contact by name

def update\_contact(self, name, new\_phone, new\_email, new\_address):

# Update contact information

def delete\_contact(self, name):

# Delete a contact

def save\_to\_file(self, filename):

# Save address book to a file

def load\_from\_file(self, filename):

# Load address book from a file

# Main program logic

address\_book = AddressBook()

# Implement a loop to continuously interact with the address book until the user chooses to exit.

# Inside the loop, you can present a menu to the user and take appropria

**Project: Personal Library System**

**Project Overview:**

Build a simple personal library system that allows users to manage their book collection. This project will involve creating classes for books, managing them in a library, and using inheritance to handle different types of books.

Classes:

Book:

Base class for all types of books.

Attributes: title, author, publication\_year.

EBook (inherits from Book):

Represents an electronic book.

Additional attribute: file\_format (PDF, EPUB, etc.).

PaperBook (inherits from Book):

Represents a physical book.

Additional attributes: ISBN, number\_of\_pages.

Library:

Manages a collection of books (both EBooks and PaperBooks).

Methods: add\_book, remove\_book, display\_books.

Guidelines:

Book Class:

Implement a Book class with attributes for title, author, and publication year.

EBook and PaperBook Classes:

Create two subclasses, EBook and PaperBook, inheriting from the Book class.

Add additional attributes specific to each type of book.

Library Class:

Implement a Library class that can store both EBooks and PaperBooks.

Include methods to add a book, remove a book, and display the list of books in the library.

File Handling:

Save the library data to a file and load it back when the program starts.

String Representation:

Implement a \_\_str\_\_ method in each class to provide a meaningful string representation when printing objects.

Example Usage:

# Create library

my\_library = Library()

# Add books to the library

ebook1 = EBook("Python Basics", "John Doe", 2020, "PDF")

paperbook1 = PaperBook("Data Structures", "Jane Smith", 2019, "1234567890", 300)

my\_library.add\_book(ebook1)

my\_library.add\_book(paperbook1)

# Display books in the library

my\_library.display\_books()

# Remove a book from the library

my\_library.remove\_book(ebook1)

# Display updated library

my\_library.display\_books()