A. Core Python & Data Handling

1. Modify the project so that all data (books, members) is auto-saved every time an operation happens, without requiring manual save().

def save\_data():

data = {

"books": {isbn: asdict(book) for isbn, book in books.items()},

"members": {mid: asdict(m) for mid, m in members.items()}

}

with open("library.json", "w") as f:

json.dump(data, f, indent=4)

print("Data auto-saved!")

def add\_book(book: Book):

books[book.isbn] = book

save\_data()

->this code indicates-> now the book is auto saved after adding.

2. Add a feature to search books by partial title or author name (case-insensitive).

Use .lower() to ignore cases while searching.

def search\_books(self, keyword: str):

keyword = keyword.lower()

return [book for book in self.books.values()

if keyword in book.title.lower() or keyword in book.author.lower()]

3. Implement sorting of books by title, author, or availability using Python’s sorted() and custom key functions.

def sort\_books(self, by: str = "title"):

if by == "title":

return sorted(self.books.values(), key=lambda b: b.title)

elif by == "author":

return sorted(self.books.values(), key=lambda b: b.author)

elif by == "availability":

return sorted(self.books.values(), key=lambda b: b.available, reverse=True)

4. Use list comprehensions to fetch all currently borrowed books.

borrowed\_books = [book for book in books.values() if not book.is\_available]

5. Add a feature to export all library data to a CSV file.

import csv

with open('library\_export.csv', 'w', newline='') as f:

writer = csv.writer(f)

writer.writerow(['ISBN', 'Title', 'Author', 'Available'])

for b in books.values():

writer.writerow([b.isbn, b.title, b.author, b.is\_available])

6. Convert the book and member collections into dictionaries of dataclasses instead of normal classes.

from dataclasses import dataclass

@dataclass

class Book:

isbn: str

title: str

author: str

is\_available: bool = True

@dataclass

class Member:

member\_id: str

name: str

borrowed\_books: list

->This lines is stored in dictionaries keyed by ISBN or member ID:

books: dict[str, Book] = {}

members: dict[str, Member] = {}

7. Use zip() to pair members with the books they borrowed for a custom report.

for member, book in zip(members.values(), borrowed\_books):

print(member.name, book.title)

8. Write a function that uses regular expressions to validate ISBN numbers.

import re

def validate\_isbn(isbn: str) -> bool:

pattern = r'^(97(8|9))?\d{9}(\d|X)$'

return re.match(pattern, isbn.replace('-', '')) is not None

B. Advanced OOP Concepts

9. Introduce a StaffMember subclass with permission to remove books, while normal members cannot.

@dataclass

class StaffMember(Member):

def remove\_book(self, isbn: str):

if isbn in books:

del books[isbn]

10. Implement operator overloading (\_\_eq\_\_, \_\_lt\_\_) so two books can be compared by ISBN.

@dataclass

class Book:

isbn: str

title: str

author: str

is\_available: bool = True

def \_\_eq\_\_(self, other):

return self.isbn == other.isbn

def \_\_lt\_\_(self, other):

return self.isbn < other.isbn