**Assignment: Library Management System**

**Problem Statement**

Create a Python class LibraryManagementSystem that simulates a simple library system. The class should have the following properties and methods:

1. **Properties**:
   * books (list of dictionaries): A list to store book information. Each book is represented as a dictionary with keys: title, author, isbn, and available (boolean).
2. **Methods**:
   * add\_book(title, author, isbn): Adds a new book to the library.
   * borrow\_book(isbn): Marks a book as borrowed if it is available.
   * return\_book(isbn): Marks a book as returned.
   * search\_book(title=None, author=None, isbn=None): Searches for books by title, author, or ISBN.
   * list\_available\_books(): Returns a list of all available books.

Write unit tests to verify the correctness of the LibraryManagementSystem class.

**Assignment Instructions**

1. **Implement the LibraryManagementSystem class** as described above.
2. **Write unit tests** for all methods in the LibraryManagementSystem class.
   * Ensure you test both valid and invalid inputs (e.g., borrowing an unavailable book, adding a duplicate ISBN).
   * Use the unittest framework to write and run the tests.
3. **Measure code coverage** using the coverage library.
   * Run the tests and generate a coverage report.
   * Ensure that your tests achieve **100% code coverage**.
4. **Submit the following**:
   * The LibraryManagementSystem class implementation.
   * The unit test file.
   * A screenshot or text output of the coverage report.

**Expected Output**

When the tests are run, they should pass, and the coverage report should show 100% coverage for the LibraryManagementSystem class.

bash

$ coverage run -m unittest test\_library\_management.py

$ coverage report -m

**Implementation of the LibraryManagementSystem Class**

python

class LibraryManagementSystem:

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

self.books = []

def add\_book(self, title, author, isbn):

if not title or not author or not isbn:

raise ValueError("Title, author, and ISBN are required.")

for book in self.books:

if book["isbn"] == isbn:

raise ValueError("Book with this ISBN already exists.")

self.books.append({

"title": title,

"author": author,

"isbn": isbn,

"available": True

})

def borrow\_book(self, isbn):

for book in self.books:

if book["isbn"] == isbn:

if book["available"]:

book["available"] = False

return f"Book '{book['title']}' borrowed successfully."

else:

return f"Book '{book['title']}' is not available."

return "Book not found."

def return\_book(self, isbn):

for book in self.books:

if book["isbn"] == isbn:

if not book["available"]:

book["available"] = True

return f"Book '{book['title']}' returned successfully."

else:

return f"Book '{book['title']}' is already available."

return "Book not found."

def search\_book(self, title=None, author=None, isbn=None):

results = []

for book in self.books:

if (title and title.lower() in book["title"].lower()) or \

(author and author.lower() in book["author"].lower()) or \

(isbn and isbn == book["isbn"]):

results.append(book)

return results

def list\_available\_books(self):

return [book for book in self.books if book["available"]]

**Grading Criteria**

1. **Correctness of Implementation** (40%):
   * The LibraryManagementSystem class should work as expected.
2. **Quality of Unit Tests** (40%):
   * Tests should cover all methods and edge cases.
3. **Code Coverage** (20%):
   * Achieve 100% code coverage for the LibraryManagementSystem class.