Coverage for **app/services/book_service.py**: 91%

66 statements | 60 run | 6 missing | 0 excluded

« prev ^ index » next coverage.py v7.11.3, created at 2025-11-19 02:12 +0000

```
1  from pathlib import Path
2  from app.models.book import Book
3  from app.utils.data_manager import CSVRepository
4  from app.schemas.book import BookCreate, BookRead, BookUpdate
5
6
7  class BookService:
8      def __init__(self):
9          self.repo = CSVRepository()
10         self.path = Path(__file__).resolve().parents[1] / "data" / "Books.csv"
11         self.fields = ["ISBN", "Book-Title", "Book-Author", "Year-Of-Publication", "Publisher", "Image-URL-S", "Image-URL-M", "Image-URL-L"
12
13
14     def __book_exists(self, isbn: str) -> bool:
15         """Check if a book with this ISBN already exists."""
16         rows = self.repo.read_all(self.path)
17         return any(r["ISBN"] == isbn for r in rows)
18
19     def __load_book_or_none(self, isbn: str):
20         """Return the row dict for a book if found."""
21         rows = self.repo.read_all(self.path)
22         for row in rows:
23             if row["ISBN"] == isbn:
24                 return row
25         return None
26
27     def get_all_books(self) -> list[BookRead]:
28         """Return all books as BookRead schemas."""
29         rows = self.repo.read_all(self.path)
30         return [
31             BookRead(**Book.from_dict(r).to_api_dict())
32             for r in rows
33         ]
34
35     def get_book(self, isbn: str):
36         """Return a single book or None."""
37         row = self.__load_book_or_none(isbn)
38         if not row:
39             return None
40         return BookRead(**Book.from_dict(row).to_api_dict())
```

```
41
42      def create_book(self, data: BookCreate) -> BookRead:
43          """Add a new book to Books.csv."""
44
45          if self.__book_exists(data.isbn):
46              raise ValueError("Book already exists in the database.")
47
48          book = Book(
49              isbn=data.isbn,
50              book_title=data.book_title,
51              author=data.author,
52              year_of_publication=data.year_of_publication,
53              publisher=data.publisher,
54              image_url_s=data.image_url_s,
55              image_url_m=data.image_url_m,
56              image_url_l=data.image_url_l
57          )
58
59          # Write one row into CSV
60          self.repo.append_row(self.path, self.fields, book.to_csv_dict())
61
62          return BookRead(**book.to_api_dict())
63
64      def update_book(self, isbn: str, data: BookUpdate):
65          """Update an existing book's fields."""
66          rows = self.repo.read_all(self.path)
67          updated = False
68
69          for r in rows:
70              if r["ISBN"] == isbn:
71                  # Only update provided fields
72                  update_data = data.model_dump(exclude_unset=True)
73
74                  if "book_title" in update_data:
75                      r["Book-Title"] = update_data["book_title"]
76                  if "author" in update_data:
77                      r["Book-Author"] = update_data["author"]
78                  if "year_of_publication" in update_data:
79                      r["Year-Of-Publication"] = update_data["year_of_publication"]
80                  if "publisher" in update_data:
81                      r["Publisher"] = update_data["publisher"]
82                  if "image_url_s" in update_data:
83                      r["Image_S"] = update_data["image_url_s"]
84                  if "image_url_m" in update_data:
85                      r["Image_M"] = update_data["image_url_m"]
86                  if "image_url_l" in update_data:
87                      r["Image_L"] = update_data["image_url_l"]
```

```
88
89                  updated = True
90                  break
91
92          if not updated:
93              return None
94
95          self.repo.write_all(self.path, self.fields, rows)
96
97          row = self.__load_book_or_none(isbn)
98          return BookRead(**Book.from_dict(row).to_api_dict())
99
100     def delete_book(self, isbn: str) -> bool:
101         """Delete a book by ISBN."""
102         rows = self.repo.read_all(self.path)
103         new_rows = [r for r in rows if r["ISBN"] != isbn]
104
105         if len(new_rows) == len(rows):
106             return False  # Book was not found
107
108         self.repo.write_all(self.path, self.fields, new_rows)
109         return True
```