Coverage for **app/services/book_service.py**: 93%

67 statements | 62 run | 5 missing | 0 excluded


```python
from pathlib import Path
from app.models.book import Book
from app.utils.books_adapter import BXBooksCSVAdapter
from app.utils.data_manager import CSVRepository
from app.schemas.book import BookCreate, BookRead, BookUpdate


class BookService:
    def __init__(self):
        self.repo = BXBooksCSVAdapter()
        self.path = str(Path(__file__).resolve().parents[1] / "data" / "BX_Books.csv")
        self.fields = ["ISBN", "Book-Title", "Book-Author", "Year-Of-Publication", "Publisher", "Image-URL-S", "Image-URL-M", "Image-URL-L"


    def __book_exists(self, isbn: str) -> bool:
        """Check if a book with this ISBN already exists."""
        rows = self.repo.read_all(self.path)
        return any(r["ISBN"] == isbn for r in rows)

    def __load_book_or_none(self, isbn: str):
        """Return the row dict for a book if found."""
        rows = self.repo.read_all(self.path)
        for row in rows:
            if row["ISBN"] == isbn:
                return row
        return None

    def get_all_books(self) -> list[BookRead]:
        """Return all books as BookRead schemas."""
        rows = self.repo.read_all(self.path)
        return [
            BookRead(**Book.from_dict(r).to_api_dict())
            for r in rows
        ]

    def get_book(self, isbn: str):
        """Return a single book or None."""
        row = self.__load_book_or_none(isbn)
```

```
        if not row:
            return None
        return BookRead(**Book.from_dict(row).to_api_dict())

    def create_book(self, data: BookCreate) -> BookRead:
        """Add a new book to Books.csv."""

        if self.__book_exists(data.isbn):
            raise ValueError("Book already exists in the database.")

        book = Book(
            isbn=data.isbn,
            book_title=data.book_title,
            author=data.author,
            year_of_publication=data.year_of_publication,
            publisher=data.publisher,
            image_url_s=data.image_url_s,
            image_url_m=data.image_url_m,
            image_url_l=data.image_url_l
        )

        # Write one row into CSV
        self.repo.append_row(self.path, self.fields, book.to_csv_dict())

        return BookRead(**book.to_api_dict())

    def update_book(self, isbn: str, data: BookUpdate):
        """Update an existing book's fields."""
        rows = self.repo.read_all(self.path)
        updated = False

        for r in rows:
            if r["ISBN"] == isbn:
                # Only update provided fields
                update_data = data.model_dump(exclude_unset=True)

                if "book_title" in update_data:
                    r["Book-Title"] = update_data["book_title"]
                if "author" in update_data:
                    r["Book-Author"] = update_data["author"]
                if "year_of_publication" in update_data:
                    r["Year-Of-Publication"] = update_data["year_of_publication"]
                if "publisher" in update_data:
                    r["Publisher"] = update_data["publisher"]
                if "image_url_s" in update_data:
```

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