

Coverage for **app/services/review_service.py**: 96%



56 statements

54 run

2 missing

0 excluded

« prev ^ index » next coverage.py v7.11.3, created at 2025-11-20 19:17 -0800

```
1 from pathlib import Path
2 from app.models.review import Review
3 from app.utils.data_manager import CSVRepository
4 from app.schemas.review import ReviewCreate, ReviewRead, ReviewUpdate
5
6
7 class ReviewService:
8     def __init__(self):
9         self.repo = CSVRepository()
10        self.path = Path(__file__).resolve().parents[1] / "data" / "Reviews.csv"
11        self.fields = ["ReviewID", "UserID", "ISBN", "Comment", "Time"]
12
13    def __read_rows(self):
14        return self.repo.read_all(self.path)
15
16    def __write_rows(self, rows):
17        self.repo.write_all(self.path, self.fields, rows)
18
19    def __generate_next_id(self) -> int:
20        rows = self.__read_rows()
21        if not rows:
22            return 1
23        ids = [int(r["ReviewID"]) for r in rows if r["ReviewID"].isdigit()]
24        return max(ids, default=0) + 1
25
26    def __already_reviewed(self, user_id: int, isbn: str) -> bool:
27        rows = self.__read_rows()
28        return any(r["UserID"] == str(user_id) and r["ISBN"] == isbn for r in rows)
29
30    def create_review(self, user_id: int, data: ReviewCreate) -> ReviewRead:
31        """
32        1 review per user per book.
33        """
34        if self.__already_reviewed(user_id, data.isbn):
```

```

35         raise ValueError("This user has already reviewed this book.")
36
37     next_id = self.__generate_next_id()
38
39     review = Review(
40         review_id=next_id,
41         user_id=user_id,
42         isbn=data.isbn,
43         comment=data.comment,
44     )
45
46     self.repo.append_row(self.path, self.fields, review.to_csv_dict())
47     return ReviewRead(**review.to_api_dict())
48
49 def get_all_reviews(self, isbn: str) -> list[ReviewRead]:
50     rows = self.__read_rows()
51     filtered = [r for r in rows if r["ISBN"] == isbn]
52     return [ReviewRead(**Review.from_dict(r).to_api_dict()) for r in filtered]
53
54 def edit_review(self, review_id: int, data: ReviewUpdate) -> ReviewRead:
55     rows = self.__read_rows()
56     found_row = None
57
58     for r in rows:
59         if r["ReviewID"] == str(review_id):
60             r["Comment"] = data.comment
61             found_row = r
62             break
63
64     if not found_row:
65         raise ValueError("Review not found")
66
67     self.__write_rows(rows)
68     updated_review = Review.from_dict(found_row)
69     return ReviewRead(**updated_review.to_api_dict())
70
71 def delete_review(self, review_id: int) -> bool:
72     rows = self.__read_rows()
73     original_count = len(rows)
74     filtered = [r for r in rows if r["ReviewID"] != str(review_id)]
75

```

```
76 |         if len(filtered) == original_count:
77 |             return False
78 |
79 |         for i, row in enumerate(filtered, start=1):
80 |             row["ReviewID"] = str(i)
81 |
82 |         self.__write_rows(filtered)
83 |
84 |         return True
```

« prev ^ index » next coverage.py v7.11.3, created at 2025-11-20 19:17 -0800