

Coverage for app/services/readinglist_service.py: 85%

114 statements 97 run 17 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.request import Request
3 from app.schemas.readinglist import ReadingListCreate, ReadingListDetail, ReadingListSummary
4 from app.models.readinglist import ReadingList
5 from app.utils.data_manager import CSVRepository
6
7 class ReadingListService:
8     def __init__(self):
9         self.repo = CSVRepository()
10        self.path = Path(__file__).resolve().parents[1] / "data" / "ReadingLists.csv"
11        self.fields = ["ListID", "UserID", "Name", "ISBNs", "IsPublic"]
12
13    def __generate_next_id(self) -> int:
14        rows = self.repo.read_all(self.path)
15        if not rows:
16            return 1
17        ids = [int(r["ListID"]) for r in rows if r["ListID"].isdigit()]
18        return max(ids, default=0) + 1
19
20    def __already_added(self, list_id: int, isbn: str) -> bool:
21        rows = self.repo.read_all(self.path)
22
23        found = False
24        for row in rows:
25            if row["ListID"] == str(list_id):
26                books = row["ISBNs"].split(" | ") if row["ISBNs"] else []
27                if isbn in books:
28                    found = True
29                    break
30
31        return found
32
33    def __number_of_readinglist(self, user_id)-> int:
34        rows = self.repo.read_all(self.path)
35        count = 0
```

```
35     for row in rows:
36         if str(user_id) == row["UserID"]:
37             count += 1
38
39     return count
40
41 def get_all_readinglist(self, user_id: int):
42     rows = self.repo.read_all(self.path)
43     result = []
44     for r in rows:
45         if r["UserID"] == str(user_id):
46             rl = ReadingList.from_dict(r)
47             result.append(
48                 ReadingListSummary(
49                     list_id=rl.list_id,
50                     name=rl.name,
51                     total_books=len(rl.books),
52                     is_public=rl.is_public
53                 )
54             )
55
56     return result
57
58 def create_list(self, data: ReadingListCreate, user_id: int) -> ReadingListDetail:
59     next_id = self.__generate_next_id()
60     num_readinglist = self.__number_of_readinglist(user_id=user_id)
61     if num_readinglist >= 10:
62         raise ValueError("You can only have 10 reading lists")
63
64     rows = self.repo.read_all(self.path)
65     for r in rows:
66         if r["UserID"] == str(user_id) and r["Name"].lower() == data.name.lower():
67             raise ValueError(f'A reading list named "{data.name}" already exists.')
68
69     readinglist = ReadingList(list_id=next_id,
70                               user_id=user_id,
71                               name=data.name)
72
73     self.repo.append_row(self.path, self.fields, readinglist.to_csv_dict())
74     return ReadingListDetail(**readinglist.to_api_dict())
75
```

```
76     def delete_list(self, list_id: int, user_id: int):
77
78         rows = self.repo.read_all(self.path)
79         original_count = len(rows)
80
81         updated_rows = [
82             r for r in rows if not (int(r["ListID"]) == list_id and int(r["UserID"]) == user_id)
83         ]
84         if len(updated_rows) == original_count:
85             return False
86
87         for i, row in enumerate(updated_rows, start=1):
88             row["ListID"] = str(i)
89
90         self.repo.write_all(self.path, self.fields, updated_rows)
91
92     return True
93
94     def rename(self, list_id: int, user_id: int, new_name: str) -> bool:
95         rows = self.repo.read_all(self.path)
96         all_names = [r["Name"].lower() for r in rows]
97             if r["UserID"] == str(user_id) and r["ListID"] != str(list_id)]
98
99         if new_name.lower() in all_names:
100             raise ValueError(f'A reading list named "{new_name}" already exists.')
101
102     renamed = False
103     for r in rows:
104         if r["UserID"] == str(user_id) and r["ListID"] == str(list_id):
105             readinglist = ReadingList.from_dict(r)
106             readinglist.rename(new_name)
107             r.update(readinglist.to_csv_dict())
108             renamed = True
109             break
110
111     if not renamed:
112         return False
113     self.repo.write_all(self.path, self.fields, rows)
114     return True
115
116     def toggle_visibility(self, list_id: int, user_id: int):
```

```
117 rows = self.repo.read_all(self.path)
118
119     for r in rows:
120         if r["ListID"] == str(list_id) and r["UserID"] == str(user_id):
121             rl = ReadingList.from_dict(r)
122             rl.is_public = not rl.is_public
123             r.update(rl.to_csv_dict())
124             self.repo.write_all(self.path, self.fields, rows)
125             return {
126                 "list_id": list_id,
127                 "is_public": rl.is_public,
128                 "message": "Visibility toggled successfully",
129             }
130     return False
131
132 def add_book(self, list_id: int, user_id: int, isbn: str) -> bool:
133     rows = self.repo.read_all(self.path)
134
135     for r in rows:
136         if r["ListID"] == str(list_id) and r["UserID"] == str(user_id):
137
138             rl = ReadingList.from_dict(r)
139
140             if isbn in rl.books:
141                 raise ValueError(f"Book {isbn} already in the reading list.")
142
143             rl.add_book(isbn)
144
145             r.update(rl.to_csv_dict())
146
147             self.repo.write_all(self.path, self.fields, rows)
148             return True
149
150     return False
151
152 def remove_book(self, list_id: int, user_id: int, isbn: str) -> bool:
153     rows = self.repo.read_all(self.path)
154
155     for r in rows:
156         if r["ListID"] == str(list_id) and r["UserID"] == str(user_id):
157             rl = ReadingList.from_dict(r)
```

```
158
159     if isbn not in rl.books:
160         raise ValueError(f"Book {isbn} not found in the reading list.")
161
162     rl.remove_book(isbn)
163
164     r.update(rl.to_csv_dict())
165
166     self.repo.write_all(self.path, self.fields, rows)
167     return True
168
169 return False
170
```

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