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

56 statements | 54 run | 2 missing | 0 excluded

 coverage.py v7.11.3, created at 2025-11-21 23:38 -0800

```python
from pathlib import Path
from app.models.review import Review
from app.repositories.csv_repository import CSVRepository
from app.schemas.review import ReviewCreate, ReviewRead, ReviewUpdate


class ReviewService:
    def __init__(self):
        self.repo = CSVRepository()
        self.path = Path(__file__).resolve().parents[1] / "data" / "Reviews.csv"
        self.fields = ["ReviewID", "UserID", "ISBN", "Comment", "Time"]

    def __read_rows(self):
        return self.repo.read_all(self.path)

    def __write_rows(self, rows):
        self.repo.write_all(self.path, self.fields, rows)

    def __generate_next_id(self) -> int:
        rows = self.__read_rows()
        if not rows:
            return 1
        ids = [int(r["ReviewID"]) for r in rows if r["ReviewID"].isdigit()]
        return max(ids, default=0) + 1

    def __already_reviewed(self, user_id: int, isbn: str) -> bool:
        rows = self.__read_rows()
        return any(r["UserID"] == str(user_id) and r["ISBN"] == isbn for r in rows)

    def create_review(self, user_id: int, data: ReviewCreate, isbn: str) -> ReviewRead:
        """
```

```
        1 review per user per book.
        """
        if self.__already_reviewed(user_id, isbn):
            raise ValueError("This user has already reviewed this book.")

        next_id = self.__generate_next_id()

        review = Review(
            review_id=next_id,
            user_id=user_id,
            isbn=isbn,
            comment=data.comment,
        )

        self.repo.append_row(self.path, self.fields, review.to_csv_dict())
        return ReviewRead(**review.to_api_dict())

    def get_all_reviews(self, isbn: str) -> list[ReviewRead]:
        rows = self.__read_rows()
        filtered = [r for r in rows if r["ISBN"] == isbn]
        return [ReviewRead(**Review.from_dict(r).to_api_dict()) for r in filtered]

    def edit_review(self, review_id: int, data: ReviewUpdate) -> ReviewRead:
        rows = self.__read_rows()
        found_row = None

        for r in rows:
            if r["ReviewID"] == str(review_id):
                r["Comment"] = data.comment
                found_row = r
                break

        if not found_row:
            raise ValueError("Review not found")

        self.__write_rows(rows)
        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
```