

Coverage for **app/utils/data_manager.py**: 93%

30 statements 28 run 2 missing 0 excluded

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

```
1 import csv
2 import os
3 import threading
4 from typing import List, Dict
5 class CSVRepository:
6     """Thread-safe CSV handler shared across all models."""
7     _locks: Dict[str, threading.Lock] = {}
8
9     def __get_lock(self, path: str) -> threading.Lock:
10         if path not in self._locks:
11             self._locks[path] = threading.Lock()
12         return self._locks[path]
13
14     def read_all(self, path: str) -> List[Dict]:
15         """
16             Reads all rows from a CSV file and returns them as a list of dictionaries.
17         """
18         if not os.path.exists(path):
19             return []
20         lock = self.__get_lock(path)
21         with lock, open(path, 'r', newline='', encoding='utf-8') as f:
22             return list(csv.DictReader(f))
23
24     def write_all(self, path: str, fieldnames: List[str], rows: List[Dict]):
25         """
26             Overwrites the entire CSV file with the given rows.
27         """
28         lock = self.__get_lock(path)
29         with lock, open(path, 'w', newline='', encoding='utf-8') as f:
30             writer = csv.DictWriter(f, fieldnames=fieldnames)
```

```
31     writer.writeheader()
32     writer.writerows(rows)
33
34     def append_row(self, path: str, fieldnames: List[str], row: Dict):
35         """
36             Add a single row to the CSV file safely.
37         """
38
39         file_exists = os.path.exists(path)
40         lock = self._get_lock(path)
41         with lock, open(path, 'a', newline='', encoding='utf-8') as f:
42             writer = csv.DictWriter(f, fieldnames=fieldnames)
43             if not file_exists:
44                 writer.writeheader()
45             writer.writerow(row)
```

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