

Coverage for **app/services/user_service.py**: 86%



143 statements 123 run 20 missing 0 excluded

« prev ^ index » next coverage.py v7.11.3, created at 2025-11-21 23:38 -0800

```
1 import csv
2 from datetime import datetime, timedelta
3 from pathlib import Path
4 from typing import Optional, Dict
5 from app.repositories.csv_repository import CSVRepository
6
7 USER_CSV = Path(__file__).resolve().parents[1] / "data" / "Users.csv"
8
9 FIELDNAMES = ["id", "username", "email", "password_hash", "is_admin", "is_suspended", "suspended_until"]
10
11 class CSVUserService:
12     def __init__(self, repo: CSVRepository, path: str = USER_CSV):
13         self.repo = repo
14         self.path = path
15
16     def _norm(self, s: str) -> str:
17         return s.strip().lower()
18
19     def _convert_row(self, row: dict) -> dict:
20         r = dict(row)
21         r["id"] = int(r["id"])
22
23         raw_flag = r.get("is_admin", "False")
24
25     def _get_next_id(self) -> int:
26         rows = self.repo.read_all(self.path)
27         if not rows:
28             return 1
29         return max(int(r["id"]) for r in rows) + 1
30
31
```

```
32 | def _check_suspension_expired(self, user: dict) -> dict:
33 |     suspended_until = user.get("suspended_until", "")
34 |     if not suspended_until:
35 |         return user
36 |
37 |     suspended_until_dt = datetime.fromisoformat(suspended_until)
38 |     if datetime.now() >= suspended_until_dt:
39 |         user["is_suspended"] = "false"
40 |         user["suspended_until"] = ""
41 |
42 |         rows = self.repo.read_all(self.path)
43 |         for r in rows:
44 |             if int(r["id"]) == user["id"]:
45 |                 r["is_suspended"] = "false"
46 |                 r["suspended_until"] = ""
47 |                 break
48 |
49 |         with open(self.path, "w", newline="", encoding="utf-8") as f:
50 |             writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
51 |             writer.writeheader()
52 |             writer.writerows(rows)
53 |
54 |     return user
55 |
56 | def _is_admin(self, user_id: int) -> bool:
57 |     rows = self.repo.read_all(self.path)
58 |
59 |     for r in rows:
60 |         if int(r["id"]) == int(user_id):
61 |             return str(r.get("is_admin", "False")).strip().lower() in {"true", "1", "yes"}
62 |
63 |     return False
64 |
65 | def get_by_username(self, username: str) -> Optional[Dict]:
66 |     username_norm = self._norm(username)
67 |
68 |     for row in self.repo.read_all(self.path):
```

```

69 |         if self._norm(row["username"]) == username_norm:
70 |
71 |             row["id"] = int(row["id"])
72 |             row["is_admin"] = row["is_admin"].lower() == "true"
73 |             row["is_suspended"] = row["is_suspended"].lower() == "true"
74 |
75 |             row = self._check_suspension_expired(row)
76 |
77 |             return row
78 |
79 |         return None
80 |
81 |     def create_user(
82 |         self,
83 |         username: str,
84 |         email: str,
85 |         password_hash: str,
86 |         is_admin: bool = False,
87 |     ) -> Dict:
88 |         users = self.repo.read_all(self.path)
89 |
90 |         username_norm = self._norm(username)
91 |         email_norm = self._norm(email)
92 |
93 |         for u in users:
94 |             if self._norm(u["username"]) == username_norm:
95 |                 raise ValueError("username_taken")
96 |             if self._norm(u["email"]) == email_norm:
97 |                 raise ValueError("email_taken")
98 |
99 |         new_id = self._get_next_id()
100 |
101 |         user = {
102 |             "id": new_id,
103 |             "username": username,
104 |             "email": email,
105 |             "password_hash": password_hash,

```

```

106         "is_admin": is_admin,
107         "is_suspended": False,
108     }
109
110     users.append(user)
111
112     with open(self.path, "w", newline="", encoding="utf-8") as f:
113         writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
114         writer.writeheader()
115         writer.writerows(users)
116
117     return user
118
119 def update_user(
120     self,
121     user_id: int,
122     username: str | None = None,
123     email: str | None = None,
124     is_admin: bool | None = None,
125 ) -> Dict:
126
127     users = self.repo.read_all(self.path)
128     updated_user = None
129
130     for u in users:
131         if int(u["id"]) == int(user_id):
132
133             if username is not None:
134                 new_norm = self._norm(username)
135                 for other in users:
136                     if int(other["id"]) != int(user_id) and self._norm(other["username"]) == new_norm:
137                         raise ValueError("username_taken")
138                 u["username"] = username
139
140             if email is not None:
141                 new_norm = self._norm(email)
142                 for other in users:

```

```
143         if int(other["id"]) != int(user_id) and self._norm(other["email"]) == new_norm:
144             raise ValueError("email_taken")
145         u["email"] = email
146
147         if is_admin is not None:
148             u["is_admin"] = "true" if is_admin else "false"
149
150         updated_user = u
151         break
152
153     if updated_user is None:
154         raise ValueError("user_not_found")
155
156     with open(self.path, "w", newline="", encoding="utf-8") as f:
157         writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
158         writer.writeheader()
159         writer.writerows(users)
160
161     updated_user["id"] = int(updated_user["id"])
162     updated_user["is_admin"] = str(updated_user.get("is_admin", "false")).lower() in {"true", "1", "yes"}
163     updated_user["is_suspended"] = str(updated_user.get("is_suspended", "false")).lower() in {"true", "1", "yes"}
164
165     return updated_user
166
167
168     def suspend_user(self, admin_id: int, target_id: int, duration_minutes: int):
169         if not self._is_admin(admin_id):
170             raise PermissionError("Admin privileges required")
171
172         rows = self.repo.read_all(self.path)
173         target = None
174
175         suspended_until_dt = datetime.now() + timedelta(minutes=duration_minutes)
176         suspended_until_str = suspended_until_dt.isoformat()
177
178         for u in rows:
179             if int(u["id"]) == int(target_id):
```

```
180         u["is_suspended"] = "true"
181         u["suspended_until"] = suspended_until_str
182         target = u
183         break
184
185     if target is None:
186         raise ValueError("user_not_found")
187
188     with open(self.path, "w", newline="", encoding="utf-8") as f:
189         writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
190         writer.writeheader()
191         writer.writerows(rows)
192
193     return target
194
195 def unsuspend_user(self, admin_id: int, target_id: int) -> Dict:
196     if not self._is_admin(admin_id):
197         raise PermissionError("Admin privileges required")
198
199     users = self.repo.read_all(self.path)
200     unsuspended_user = None
201
202     for u in users:
203         if int(u["id"]) == int(target_id):
204             u["is_suspended"] = "false"
205             u["suspended_until"] = ""
206             unsuspended_user = u
207             break
208
209     if unsuspended_user is None:
210         raise ValueError("user_not_found")
211
212     with open(self.path, "w", newline="", encoding="utf-8") as f:
213         writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
214         writer.writeheader()
215         writer.writerows(users)
216
```

217 | **return** unsuspended_user

« prev ^ index » next coverage.py v7.11.3, created at 2025-11-21 23:38 -0800