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

143 statements | 123 run | 20 missing | 0 excluded


```python
import csv
from datetime import datetime, timedelta
from pathlib import Path
from typing import Optional, Dict
from app.repositories.csv_repository import CSVRepository

USER_CSV = Path(__file__).resolve().parents[1] / "data" / "Users.csv"

FIELDNAMES = ["id", "username", "email", "password_hash", "is_admin", "is_suspended", "suspended_until"]

class CSVUserService:
    def __init__(self, repo: CSVRepository, path: str = USER_CSV):
        self.repo = repo
        self.path = path

    def _norm(self, s: str) -> str:
        return s.strip().lower()

    def _convert_row(self, row: dict) -> dict:
        r = dict(row)
        r["id"] = int(r["id"])

        raw_flag = r.get("is_admin", "False")

    def _get_next_id(self) -> int:
        rows = self.repo.read_all(self.path)
        if not rows:
            return 1
        return max(int(r["id"]) for r in rows) + 1


```

```
    def _check_suspension_expired(self, user: dict) -> dict:
        suspended_until = user.get("suspended_until", "")
        if not suspended_until:
            return user

        suspended_until_dt = datetime.fromisoformat(suspended_until)
        if datetime.now() >= suspended_until_dt:
            user["is_suspended"] = "false"
            user["suspended_until"] = ""

            rows = self.repo.read_all(self.path)
            for r in rows:
                if int(r["id"]) == user["id"]:
                    r["is_suspended"] = "false"
                    r["suspended_until"] = ""
                    break

            with open(self.path, "w", newline="", encoding="utf-8") as f:
                writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
                writer.writeheader()
                writer.writerows(rows)

        return user

    def _is_admin(self, user_id: int) -> bool:
        rows = self.repo.read_all(self.path)

        for r in rows:
            if int(r["id"]) == int(user_id):
                return str(r.get("is_admin", "False")).strip().lower() in {"true", "1", "yes"}

        return False

    def get_by_username(self, username: str) -> Optional[Dict]:
        username_norm = self._norm(username)

        for row in self.repo.read_all(self.path):
```

```
            if self._norm(row["username"]) == username_norm:

                row["id"] = int(row["id"])
                row["is_admin"] = row["is_admin"].lower() == "true"
                row["is_suspended"] = row["is_suspended"].lower() == "true"

                row = self._check_suspension_expired(row)

                return row

        return None

    def create_user(
        self,
        username: str,
        email: str,
        password_hash: str,
        is_admin: bool = False,
    ) -> Dict:
        users = self.repo.read_all(self.path)

        username_norm = self._norm(username)
        email_norm = self._norm(email)

        for u in users:
            if self._norm(u["username"]) == username_norm:
                raise ValueError("username_taken")
            if self._norm(u["email"]) == email_norm:
                raise ValueError("email_taken")

        new_id = self._get_next_id()

        user = {
            "id": new_id,
            "username": username,
            "email": email,
            "password_hash": password_hash,
```

```
            "is_admin": is_admin,
            "is_suspended": False,
        }

        users.append(user)

        with open(self.path, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
            writer.writeheader()
            writer.writerows(users)

        return user

    def update_user(
        self,
        user_id: int,
        username: str | None = None,
        email: str | None = None,
        is_admin: bool | None = None,
    ) -> Dict:

        users = self.repo.read_all(self.path)
        updated_user = None

        for u in users:
            if int(u["id"]) == int(user_id):

                if username is not None:
                    new_norm = self._norm(username)
                    for other in users:
                        if int(other["id"]) != int(user_id) and self._norm(other["username"]) == new_norm:
                            raise ValueError("username_taken")
                    u["username"] = username

                if email is not None:
                    new_norm = self._norm(email)
                    for other in users:
```

```
                        if int(other["id"]) != int(user_id) and self._norm(other["email"]) == new_norm:
                            raise ValueError("email_taken")
                    u["email"] = email

                if is_admin is not None:
                    u["is_admin"] = "true" if is_admin else "false"

                updated_user = u
                break

        if updated_user is None:
            raise ValueError("user_not_found")

        with open(self.path, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
            writer.writeheader()
            writer.writerows(users)

        updated_user["id"] = int(updated_user["id"])
        updated_user["is_admin"] = str(updated_user.get("is_admin", "false")).lower() in {"true", "1", "yes"}
        updated_user["is_suspended"] = str(updated_user.get("is_suspended", "false")).lower() in {"true", "1", "yes"}

        return updated_user


    def suspend_user(self, admin_id: int, target_id: int, duration_minutes: int):
        if not self._is_admin(admin_id):
            raise PermissionError("Admin privileges required")

        rows = self.repo.read_all(self.path)
        target = None

        suspended_until_dt = datetime.now() + timedelta(minutes=duration_minutes)
        suspended_until_str = suspended_until_dt.isoformat()

        for u in rows:
            if int(u["id"]) == int(target_id):
```

```
                u["is_suspended"] = "true"
                u["suspended_until"] = suspended_until_str
                target = u
                break

        if target is None:
            raise ValueError("user_not_found")

        with open(self.path, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
            writer.writeheader()
            writer.writerows(rows)

        return target

    def unsuspend_user(self, admin_id: int, target_id: int) -> Dict:
        if not self._is_admin(admin_id):
            raise PermissionError("Admin privileges required")

        users = self.repo.read_all(self.path)
        unsuspended_user = None

        for u in users:
            if int(u["id"]) == int(target_id):
                u["is_suspended"] = "false"
                u["suspended_until"] = ""
                unsuspended_user = u
                break

        if unsuspended_user is None:
            raise ValueError("user_not_found")

        with open(self.path, "w", newline="", encoding="utf-8") as f:
            writer = csv.DictWriter(f, fieldnames=FIELDNAMES)
            writer.writeheader()
            writer.writerows(users)

```

```
217         return unsuspended_user
```