Software Security Concerns

1) Authentication

- **Current Implementation**
  - The backend uses **JWT (JSON Web Tokens)** for authentication.
  - On login, the user gets a signed JWT with their ID and role.
  - The frontend stores the token in **localStorage + cookie** for API access.
  - Protected endpoints require a valid token using @jwt_required in backend routes.
- **Strengths**
  - Tokens prevent the need to store sessions server-side.
  - Backend rejects requests without valid tokens.
- **Weaknesses**
  - Currently no token **expiration/refresh** mechanism.
  - Token stored in **localStorage** is vulnerable to XSS attacks.
- **Improvements**
  - Add **short-lived access tokens** + **refresh tokens**.
  - Store tokens in **HttpOnly cookies** instead of localStorage.
  - Enforce **strong password policy** and **rate limiting** on login attempts.

2) Role Management

- **Current Implementation**
  - Users have a role column (user or admin).
  - Role enum is defined in models.py.
  - At the moment, the frontend does not expose admin-only actions, but role support exists for future admin dashboards.
- **Improvements**
  - Add **role-based route protection** (e.g., only admin can access /admin/*).
  - Validate role in backend endpoints with a decorator like:

```
from fastapi import HTTPException, status

def require_admin(user):
    if user.role != "admin":
        raise HTTPException(status.HTTP_403_FORBIDDEN, "Admin access required")
```

3) Threats & Countermeasures

- **Threat: SQL Injection**
    - Counter: SQLAlchemy ORM + parameterized queries already mitigate this.
- **Threat: Cross-Site Scripting (XSS)**
    - Counter: Escape all user-provided text before rendering.
    - Move JWT from localStorage to HttpOnly cookies.
- **Threat: Data Leakage**
    - Counter: Encrypt sensitive fields (analysis text) at rest.
    - Don’t expose raw results in public APIs.
- **Threat: Brute Force Attacks**
    - Counter: Add **rate limiting** and **account lockout** after multiple failed login attempts.
- **Threat: Weak HTTPS**
    - Counter: Always deploy with **HTTPS** in production.

Conclusion for F

- **Authentication :** Implemented with JWT, improvements proposed (refresh tokens, HttpOnly).
- **Role Management :** Role enum exists, but should be enforced at route level.
- **Security Threats & Countermeasures :** Identified common threats (SQLi, XSS, brute force, data leakage) and solutions.