Cybersecurity Considerations

1. Outdated Dependencies (Django, Node, Python)

Many vulnerabilities come from running outdated versions of Django, Node.js, and Python. Older releases may contain publicly known exploits.

Risks:

- Remote code execution

- SQL injection

- Cross-site scripting (XSS)

- Authentication/authorization flaws

Mitigation:

- Review release notes for all major dependencies.

- Perform upgrades in a separate feature branch.

- Test thoroughly before merging, as updates can break existing features.

- Avoid skipping major versions without compatibility checks.

2. verify=False in settings.py

Setting verify=False disables SSL certificate validation for HTTPS requests.

Risks:

- Man-in-the-middle (MITM) attacks

- Intercepted or altered data

- Spoofed endpoints

Mitigation:

- Remove or avoid using verify=False entirely.

- Ensure all requests use HTTPS with valid certificates.

- Use proper dev/staging/production SSL configurations.

3. Improper Secret/Token Management

Leaving secrets directly in code, Git, or public locations can lead to full compromise.

Risks:

- Unauthorized access to user accounts

- Stolen tokens and API keys

- Privilege escalation

Mitigation:

- Store all secrets in environment variables or a secret manager.

- Rotate keys frequently.

- Never push credentials to GitHub.

To-Do: Fix Web Certificate and verify=False Usage

- Update or replace the current SSL certificate.

- Ensure the certificate matches the domain and is trusted.

- Confirm HTTPS is enforced end-to-end (load balancer → server → application).

- Remove verify=False from all requests and confirm proper certificate validation.