# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9-21-2025** | **Justine kummer** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Justine kummer

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

The values of secure communications of the company are integrity and confidentiality. Sensitive data must be protected and encrypted. TLS should be used for all moving data as a security measure. Artemis financial may work internationally and across states so it has to follow foreign laws like GDPR. This is used to show the regulations for EU personal data. Governmental restrictions should be considered. External threats can be unauthorized access, input-based attacks, data leaks, exploiting outdated libraries and brute force attacks. DevSecOps can be used. Its updated system management makes more secure applications by integrating security measures into each step. Moving to more secure frameworks and token-based authentication can also be helpful like 0Auth2. Artemis financial would do well with a secure modernization path that uses TLS, secures secrets safely using authorization and scans for patches/dependencies. Using open-source libraries is also beneficial but will need routine CVE scanning and dependency checks.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Finacial's software application. Justify your reasoning for why each area is relevant to the software application.

The most important parts for Artemis Financial are APIs, input validation, cryptography, client/server security, code quality, and dependency management, because those affect the confidentiality and integrity of sensitive financial data. The API uses rest endpoints, leaving it unsecured. A way to fix this is to make all API require a valid token and role-based permissions. This is the main way to attack. Input validation is important to prevent SQL injection attacks and other injection attacks. User inputs should be validated and sanitized to keep this from happening. Cryptography is important because data should be encrypted to protect vulnerable data. Client/server security should be secure to protect deployment configurations like firewalls and ports. Code quality and encapsulation is next. Code security practices help reduce logic level vulnerabilities. To prevent supply chain risk dependence management is necessary. Since the API is server based these are all important to protect confidentiality and data integrity.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

1. hard coded database credentials, this exposes secrets in the source and increases the risk of DB unauthorized access. To fix this you can remove the credentials from the code or use a secrets manager. Located in DocData.java. 2. missing the authorization and authentication. The rest endpoints are accessible with no authentication leaving it vulnerable. Add a token-based authentication process like 0Auth2. 3.lack of validation of input. Inputs are freely accepted without sanitation or authentication. This leaves it very easy for injection-based attacks. A solution is to implement server-side validation and parameterized queries. 4. stack traces are leaking information. The stack traces are printed, and sensitive details can leak into the logs. Replacing it with structured logging and sensitive details should be logged to appropriate protected stores. Another is outdated dependencies. These leave the application open to attack and software mismatch. Upgrading any out-of-date software should fix this issue. Poor coding practices like poor naming skills and encapsulation are also prevalent. This increases the chance of logic misuse and decreases the validation boundaries. Incomplete place holder code should be completed and reviewed. This can cause logic issues and insecure faults. There are lots of high security issues, these must be fixed before completion.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions are provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

[Include your findings here.]

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

Many issues need to be addressed. The first is outdated software. To keep up with the newest security trends and prevent outdated security flaws. Next is to remove hard-coded credentials then rotate the credentials after. Set up authentication and authorization using a token-based system. Use TLS and upgrade dependencies. Complete place holder code to test and review. After deployment set up a user policy and Maintenace. These will all reduce future risk to the application in the future.