Secure Coding Review - Final Submission Draft

Internship: CodeAlpha

Task: Secure Coding Review

Name: Aisha Ibrahim Zakari

Date: 20th June 2025

1. Code Reviewed

Language: Python

Sample Function: Below is the function reviewed for security issues:

```
import sqlite3
from flask import Flask, request

app = Flask(__name__)

@app.route('/login')
def login():
username = request.args.get('username')
password = request.args.get('password')

conn = sqlite3.connect('users.db')
cursor = conn.cursor()
cursor.execute(f"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'")
result = cursor.fetchone()

if result:
```

return "Login successful"

else:

return "Invalid credentials"

2. Identified Vulnerabilities

Type	Description	Severity
SQL Injection	User input is directly inserted into SQL without validation.	Critical
Hardcoded DB File	Database name is hardcoded, making it harder to change environments securely.	Medium
No Input Validation	Username and password are not validated or sanitized.	High
Verbose Error Handling	Error messages could reveal system logic if exceptions are raised (not shown here).	Medium

3. Manual Inspection Method Used

- Followed OWASP Secure Coding Practices
- Checked for input sanitization, injection flaws, and authentication logic
- No tools were used for this review (manual-only, per task option)

4. Secure Coding Recommendations

Issue	Recommendation

| SQL Injection | Use parameterized queries (?) or ORM instead of string concatenation |

| Hardcoded DB File | Use environment variables to manage DB paths |

| No Input Validation | Sanitize inputs using validation libraries or frameworks |

| Error Handling | Add try-except blocks with generic user messages, log errors internally |

Example Fix:

```
cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?", (username, password))
```

5. Summary

A simple Flask app was reviewed manually. It contained multiple high-severity security vulnerabilities, including SQL injection and lack of input validation. Secure coding practices must be applied to avoid these risks, especially when dealing with authentication and databases.