Student ID: CA/JA1/4182

REPORT OF TASK 3

Aim: Secure Coding Review

Choose a programming language and application. Review the code for security vulnerabilities and provide recommendations for secure coding practices. Use tools like static code analyzers or manual code review.

Objective:

The objective of a Secure Coding Review is to identify and mitigate security vulnerabilities within an application's source code by analyzing it against established secure coding practices and standards. This ensures the application is robust, secure, and resistant to potential attacks or exploits.

Programming Language: Python

Sample Code:

CODES > CodeAlpha > app.py > get_db_connection

```python
from flask import Flask, request, jsonify
import sqlite3
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)

# Secure connection to the database
Tabnine | Edit | Test | Explain | Document | Ask
def get_db_connection():
    conn = sqlite3.connect('example.db')
    conn.row_factory = sqlite3.Row
    return conn

# Secure user data retrieval
Tabnine | Edit | Test | Explain | Document | Ask
@app.route('/user/<username>', methods=['GET'])
def get_user(username):
    conn = get_db_connection()
    user = conn.execute("SELECT * FROM users WHERE username = ?", (username,)).fetchone()
    conn.close()
    if user is None:
        return jsonify({"error": "User not found"}), 404
    return jsonify({"username": user["username"]})  # Expose only necessary fields

# Secure user addition
Tabnine | Edit | Test | Explain | Document | Ask
@app.route('/user', methods=['POST'])
def add_user():
    data = request.json
    username = data['username']
    password = data['password']
    hashed_password = generate_password_hash(password)  # Hash the password
    conn = get_db_connection()
    conn.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username, hashed_password))
    conn.commit()
    conn.close()
    return jsonify({"message": "User added"}), 201

if __name__ == "__main__":
    app.run()  # Remove debug=True for production
```

- This is the sample code which was taken for the task code analysis
- The sample vulnerable code is present over the open source on internet so we can find the code easily.
- For the analysis we are using **Bandit tool** for the code reviewing
- The use of Bandit Tool [bandit -r file/path]

Output: The code review by using of Bandit Tool

```
PS F:\CODES\CodeAlpha> bandit app.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.12.1
Run started:2025-01-08 07:36:36.109689

Test results:
>> Issue: [B608:hardcoded_sql_expressions] Possible SQL injection vector through string-based query construction.
   Severity: Medium   Confidence: Medium
   CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b608_hardcoded_sql_expressions.html
   Location: .\app.py:16:26
15          conn = get_db_connection()
16          user = conn.execute(f"SELECT * FROM users WHERE username = '{username}'").fetchone()
17          conn.close()

--------------------------------------------------
>> Issue: [B608:hardcoded_sql_expressions] Possible SQL injection vector through string-based query construction.
   Severity: Medium   Confidence: Medium
   CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b608_hardcoded_sql_expressions.html
   Location: .\app.py:29:19
28          conn = get_db_connection()
29          conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30          conn.commit()

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
   CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b608_hardcoded_sql_expressions.html
   Location: .\app.py:29:19
28          conn = get_db_connection()
29          conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30          conn.commit()
```

```
28          conn = get_db_connection()
29          conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30          conn.commit()

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
29          conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30          conn.commit()

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
34      if __name__ == "__main__":
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
```

```
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
28         conn = get_db_connection()
29         conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30         conn.commit()

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
29         conn.execute(f"INSERT INTO users (username, password) VALUES ('{username}', '{password}')")
30         conn.commit()

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4

--------------------------------------------------
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
34     if __name__ == "__main__":
>> Issue: [B201:flask_debug_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
   Severity: High   Confidence: Medium
   CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)
   More Info: https://bandit.readthedocs.io/en/1.8.0/plugins/b201_flask_debug_true.html
   Location: .\app.py:35:4
```

```
--------------------------------------------------

Code scanned:
        Total lines of code: 27
        Total lines skipped (#nosec): 0

Code scanned:
        Total lines of code: 27
        Total lines skipped (#nosec): 0

        Total lines of code: 27
        Total lines skipped (#nosec): 0

Run metrics:
        Total lines skipped (#nosec): 0

Run metrics:

Run metrics:
Run metrics:
        Total issues (by severity):
                Undefined: 0
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 2
                High: 1
                Undefined: 0
                Low: 0
                Medium: 2
                High: 1
                Low: 0
                Medium: 2
                High: 1
                Medium: 2
                High: 1
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 3
                High: 0
```

Overview:

The provided code implements a Flask web application that interacts with an SQLite database to handle user data. It features endpoints for retrieving user information (GET /user/<username>) and adding new users (POST /user) while hashing passwords for secure storage. However, there are several security concerns. The database connection is established for every request without a connection pool, potentially leading to performance issues. Input validation is missing for username and password fields, increasing the risk of processing invalid or malicious data. While parameterized queries effectively prevent SQL injection, sensitive user information might still be exposed if the returned data contains unnecessary fields**.**

Vulnerability Scanning Result:

1. Input Validation Vulnerability

Issue: Input data for username and password in the POST /user endpoint is not validated.

```
data = request.json
username = data['username']
password = data['password']
```

2. Missing Security Headers

Issue: No security headers are implemented.

3. Debug Mode in Production

Issue: Debug mode could be enabled in production.

```
if __name__ == "__main__":
    app.run(debug=True)  # Risky in production
```

Mitigation Technique:

1. **Input Validation Vulnerability**

Fixed Code: Use input validation to ensure both fields are present and meet criteria (e.g., length, allowed characters).

```
from flask import abort
data = request.json
username = data.get('username', '').strip()
password = data.get('password', '').strip()
if not username or not password:
    abort(400, description="Username and password are required.")
if len(username) < 3 or len(password) < 8:
    abort(400, description="Username must be at least 3 characters and password at least 8.")
if not username.isalnum():
    abort(400, description="Username must contain only alphanumeric characters.")
```

2. **Missing Security Headers**

Fixed Code: Use Flask-Talisman to add security headers.

```
from flask_talisman import Talisman
talisman = Talisman(app)
```

3. Debug Mode in Production

Fixed Code: Disable debug mode and use environment variables to differentiate environments.

```
import os

if __name__ == "__main__":

  app.run(debug=os.getenv('FLASK_DEBUG', 'False') == 'True')
```

- By using above mitigation technique we can fix all the vulnerabilities in the present code.