Secure Coding Review

✅ Application Reviewed:

Flask Login App

🔍 Method Used:

Manual + Static Analysis (bandit)

🧱 Vulnerabilities Found:

1. SQL Injection
2. Plaintext Passwords
3. Reflected XSS
4. Lack of Input Validation
5. Debug Mode Enabled

🔧 Secure Coding Recommendations:

- Always **use parameterized queries** to prevent SQL injection.
- **Hash passwords** using bcrypt, never store them as plain text.
- Use render_template() or escape all user input to avoid XSS.
- **Sanitize inputs** before using them in logic or output.
- Disable **debug mode** in production.

🗂️ Structure:

- app.py
- setup_db.py

💻 Code:

app.py:

```
# secure_app.py
from flask import Flask, request, render_template, redirect
import sqlite3
import bcrypt
import os

app = Flask(__name__)
```

```
@app.route('/')
def index():
    return '''
        <h2>Login</h2>
        <form method="POST" action="/login">
            Username: <input type="text" name="username"/><br>
            Password: <input type="password" name="password"/><br>
            <input type="submit" value="Login"/>
        </form>
    '''

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password'].encode('utf-8')

    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()

    cursor.execute("SELECT password FROM users WHERE username = ?", (username,))
    row = cursor.fetchone()

    if row and bcrypt.checkpw(password, row[0].encode('utf-8')):
        return f"<h3>Welcome, {username}!</h3>"
    else:
        return "<h3>Login Failed!</h3>"

if __name__ == '__main__':
    app.run(debug=False)  # ✅ Debug mode off
```

setup_db.py:

```
import sqlite3
import bcrypt

# Connect to the SQLite database (it will create it if not exists)
conn = sqlite3.connect('users.db')
cursor = conn.cursor()
```

```
# Create users table
cursor.execute('''
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL
)
''')

# Insert a test user
username = 'admin'
plain_password = 'password123'
hashed_password = bcrypt.hashpw(plain_password.encode('utf-8'),
bcrypt.gensalt()).decode()


cursor.execute('INSERT OR IGNORE INTO users (username, password) VALUES (?, ?)',
(username, hashed_password))

# Save and close
conn.commit()
conn.close()

print("✅Database and user created.")
```

📟 Commands:

```
python setup_db.py
python app.py
```

Then Visit: http://127.0.0.1:5000