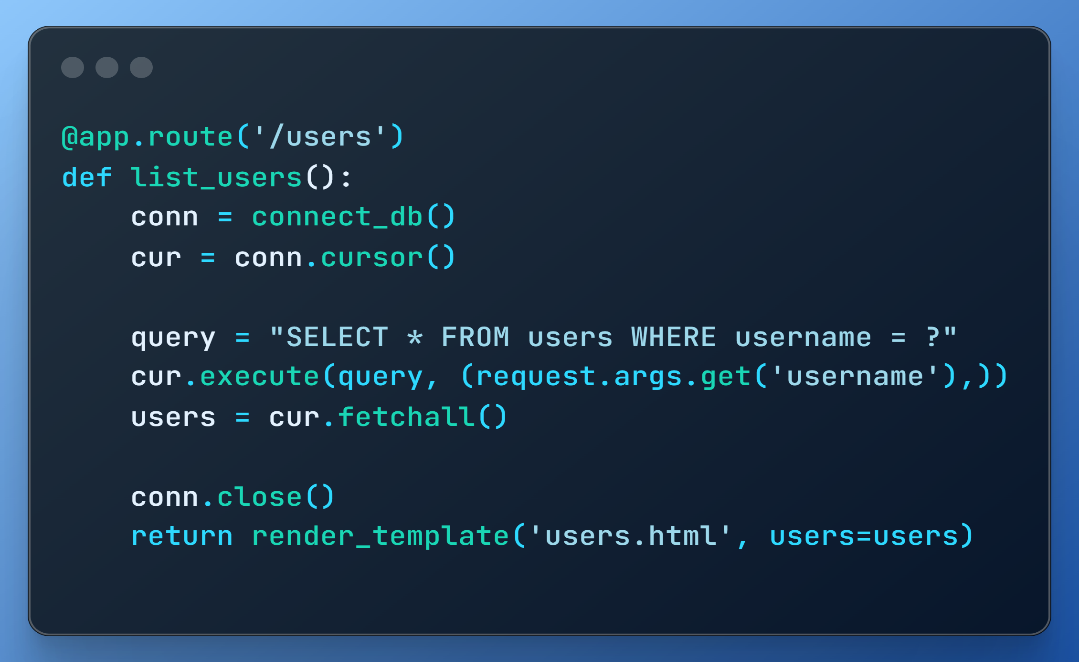
**We Will Start with Flask App that Contain vulnerabilities:**

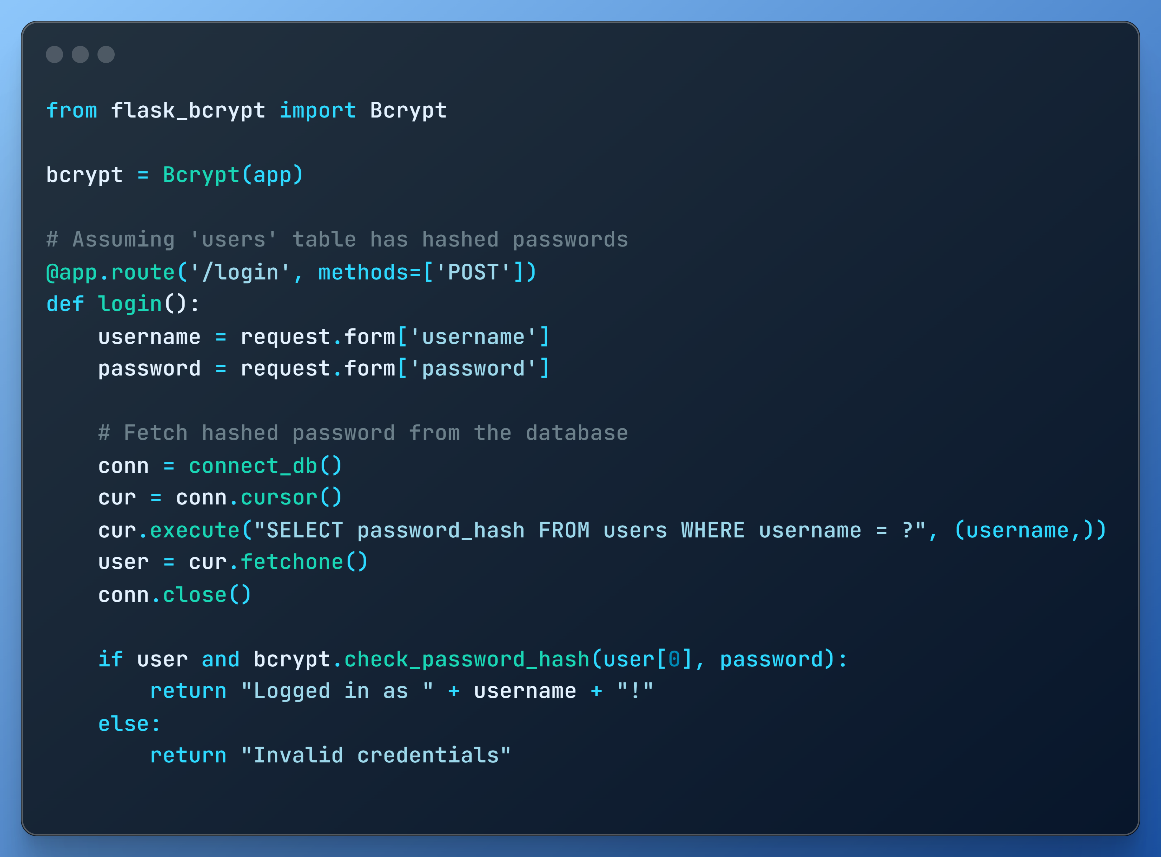


**Potential Vulnerabilities:**

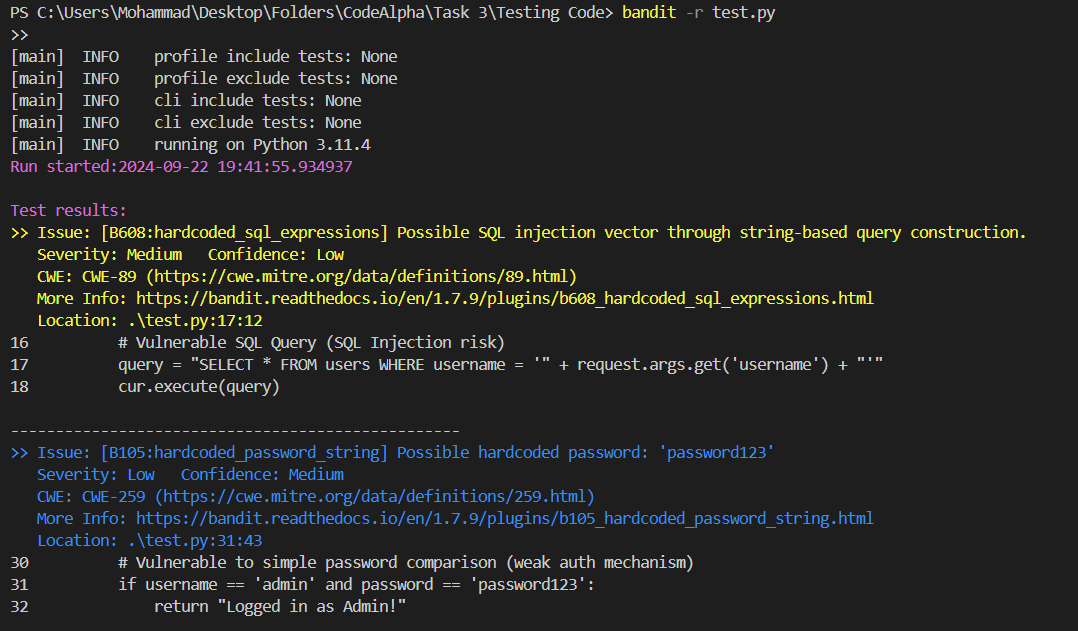
1. **SQL Injection:**

* The list\_users function constructs a SQL query by concatenating user input (username) directly into the query. This makes it vulnerable to SQL Injection.
* **How to fix it?**
  + The query in list\_users is susceptible to SQL Injection. An attacker could pass malicious input (e.g., admin' OR '1'='1) to bypass the query and retrieve all users. **Solution**: Use **parameterized queries** (also called **prepared statements**).

1. **Weak Authentication**:

* In the login route, the authentication is based on a hardcoded password, which is insecure and doesn’t use proper hashing or salting.
* **How to fix it?**
  + The login function uses hardcoded credentials without hashing the password, making it highly insecure. **Solution: Use hashed passwords and a proper authentication flow.**

**Using Bandit Code Analyzer:**



**Summary:**

* **Use Parameterized Queries**: Avoid building SQL queries by concatenating user input. Always use parameterized queries or ORM (e.g., SQLAlchemy).
* **Hash Passwords**: Never store or use plaintext passwords. Use strong hashing algorithms (e.g., bcrypt) with salting.
* **Input Validation**: Always validate and sanitize user input to prevent XSS and other attacks.
* **Use Secure Communication**: Implement HTTPS to secure data transmission.