🔐 Security Analysis & Testing Report

✅ Section 1: Code Analysis and Security Testing

1.1 Static Code Analysis Tool Usage

Tool Used: SonarLint (IDE Plugin)

Findings:

- ✅ **Secure route** uses parameterized queries and bcrypt for password verification.
- ❌ **Insecure route** directly concatenates user input in SQL query — vulnerable to SQL Injection (OWASP A01).
- ✅ Input validation function checks for SQL control characters and enforces character limits.
- ❗ **Potential Issue**: Debug logs print raw SQL queries (line: `print(f"Executing Query: {query}")`) — may leak sensitive data in logs.

Evidence:

- SonarLint flagged string concatenation in SQL query and log statement as critical security issues.

1.2 Manual Code Review (Security Checklist)

Checklist Item	Secure Route	Insecure Route	Notes
Uses parameterized queries	✅	❌	Secure route is safe, insecure route is vulnerable
Validates user input	✅	❌	Only secure route uses `is_valid_input()`
Passwords hashed & checked securely	✅ (bcrypt)	❌ (plaintext)	Secure route follows best practices
Sensitive data in logs	❌	❌	Both routes may log sensitive queries
Catches and handles exceptions	✅	❌	Secure route uses `try-except`

1.3 Security Test Cases

Test Case Type	Description	Expected Result	Actual Result	Pass/ Fail
✅	Valid credentials on secure route	Login	Success page	✅
❌ Negative	SQL injection on insecure route (e.g. `' OR '1'='1`)	Login fails	Login successful (vulnerable)	❌
⚠️ Edge	Long input string (> 50 chars)	Input	Error message	✅

Logs and Screenshots: Available upon request (e.g., terminal output, SonarLint IDE results).

🚨 Section 2: Basic Vulnerability Assessment

2.1 Identified Vulnerabilities

1. SQL Injection (OWASP A01)

- **Location**: `login_insecure()` route
- **Description**: Directly includes user input in SQL query string.
- **Reproduction**: Enter username as `' OR '1'='1` and any password.
- **Evidence**: Successful login bypass despite invalid credentials.
- **Recommendation**: Use parameterized queries (`?`) and input validation.

2. Sensitive Data Exposure (OWASP A02)

- **Location**: `print(f"Executing Query: {query}")`
- **Description**: Query with user input printed to logs.
- **Impact**: Could expose credentials if logs are accessed.
- **Recommendation**: Remove debug logging or sanitize log output.

🧾 Section 3: Security Documentation and Reporting

3.1 Test Cases Documentation

Test ID	Type	Input	Expected Output	Actual Output	Pass/ Fail
TC001	Positive	Valid username/ password	Success Page	Success Page	✅
TC002	Negative	SQL Injection input	Rejection/Error	Success (bypass)	❌
TC003	Edge	Long string (> 50 chars)	Input Rejected	Error Message	✅

3.2 Vulnerability Report Summary

Vulnerability	Affected Route	Risk Level	Evidence	Fix Recommendation
SQL Injection	`/login_insecu`	High	Query log, successful bypass	Use parameterized queries
Log Exposure	All routes	Medium	Console prints sensitive queries	Avoid logging raw SQL

3.3 Evidence Collected

- SonarLint screenshots with flagged vulnerabilities.
- Console logs showing SQL injection query and results.
- Screenshots of test inputs and outputs (can be added as needed).