Codebase Issues Report

This document lists the problems in the provided Flask codebase. The list is sorted by priority, with the most critical issues first.

# Critical Problems

# High Priority Problems

## SQL Injection Vulnerabilities

Raw string formatting is used in SQL queries, which allows arbitrary SQL execution. Should use parameterized queries with placeholders.

## Weak Password Hashing

Passwords are hashed using MD5, which is insecure.

## Sensitive Information Exposure

Database errors and debug info may expose sensitive info. Should avoid printing sensitive info and handle errors properly.

print("Database not found") is too risky, and the logs can be used to leak details about the file structure in the system, you wouldn’t want to expose.

sys.exit(1) stops the whole application which could lead a HTTP 500 (Server error) and if DEBUG=true is set, Flask would expose the error with intern information about variables and so on which could expose system details that could be attacked.

## Session Security

Session cookies are not marked as secure or HttpOnly. Should do “Set SESSION\_COOKIE\_SECURE=True and SESSION\_COOKIE\_HTTPONLY=True.”.

## Duplicate Redirect Handling in API

API endpoints return redirects meant for HTML, confusing API clients. Should separate API responses from web form redirects.

## Potential Database Locking / Resource Leaks

DB connections may remain open if errors occur.

## Hardcoded Secret Key

SECRET\_KEY is hardcoded and insecure. Environment variables should be used to load secret keys so they do not get used without authorization.

# Medium Priority Problems

## Legacy Python Compatibility

from \_\_future\_\_ import with\_statement is unnecessary in Python 3. (We are using Python 3)

## Inconsistent API / Web Form Separation

API and web routes are mixed, leading to inconsistent behavior.

## Test Setup Issues

Test code references app.DATABASE but main code uses DATABASE\_PATH.

## Improper use of Flash Messages in API

Flash messages are for web, not API clients.

## No Input Validation / Sanitization

User inputs are used directly in SQL and templates, risking SQL injection and XSS.

# Low Priority Problems

## Redundant Imports

Some imports, like werkzeug.redirect, are repeated. (We have done this to make it work with Python3)

## Unimplemented Search Test

The test\_search() function is empty.

## UI / UX

Search reloads the page (inline JS could be separated).

## Database Path Hardcoding

DATABASE\_PATH is fixed (better to use env variables).

## MD5 Documentation Comment

Comment says MD5 encryption (MD5 is a hash, not encryption).

## Unnecessary check\_db\_exists in connect\_db

Exits program if DB missing (could handle more gracefully).

## Static HTML references

Images assume static paths exist (no fallback if missing).