# SQL Injection Playground – Project Report

Author: Aslam Mohamed

Date: 2025-09-10

## 1. Introduction
SQL Injection (SQLi) remains one of the most critical web vulnerabilities in modern web applications. It enables attackers to manipulate backend SQL queries by inserting malicious input, potentially exposing sensitive data or damaging the database. This project, **SQLi Playground**, is designed as an educational tool to simulate how SQLi works, demonstrate safe practices, and showcase a basic detection mechanism.

## 2. Abstract
The main goal of this project is to create a controlled environment for learners to understand the mechanics of SQL Injection vulnerabilities and defenses. The web app contains both a vulnerable login form and a safe login form using parameterized queries. In addition, it features a basic Python-based detection engine that scans inputs for common SQLi patterns and logs suspicious activities in real-time. This holistic approach aids in learning about both attack methods and security defenses.

## 3. Tools Used
- **Flask**: Lightweight web framework for handling routes and rendering pages.
- **SQLite**: Lightweight database to store user credentials and demonstrate SQL interactions.
- **Python**: Used to implement backend logic, the detection engine, and manage database interactions.
- **Bootstrap**: CSS framework used to design responsive and clean user interfaces.
- **Requests Library**: Used in the detection engine to perform automated payload injection tests.

## 4. Steps Involved in Building the Project
### Step 1: Web Application Design
- Built separate login pages: one intentionally vulnerable and one secured with parameterized queries.
- Created informative About and Dashboard pages using Bootstrap for better user experience.

### Step 2: Database Setup
- Implemented SQLite database with a simple 'users' table.
- Added sample users (e.g., alice, bob) for demonstration purposes.

### Step 3: SQL Injection Detector Implementation
- Developed a Python-based detection system that compares user input against a

curated list of SQLi payload patterns.
- Logs detected payloads along with timestamps and response information into a log file.

**Step 4: Integration with Flask**
- Integrated the detector into the vulnerable login flow.
- Displayed warnings when malicious patterns were detected.

**Step 5: Documentation and Educational Support**
- Created an About page explaining vulnerabilities, safe coding practices, and the purpose of the detection engine.

**5. Conclusion**
This project serves as a practical and interactive educational tool to understand SQL Injection. It highlights the importance of secure coding practices, particularly the use of parameterized queries. While the simple detection engine demonstrates basic monitoring capability, real-world systems employ more advanced machine learning models and heuristic techniques. Overall, the project effectively bridges theoretical learning with hands-on practice in a safe environment.