**Iteration 2**

**1. Plaintext Password Storage**

* **Vulnerability**: Passwords are stored as plaintext in the SQLite database.
* **Impact**: If the database is compromised, all user passwords are easily accessible.
* **Mitigation**: Use a hashing algorithm like bcrypt or Argon2 to store password hashes instead of plaintext.

**2. SQL Injection**

* **Vulnerability**: While the code uses parameterized queries (which reduces risk), improper sanitization elsewhere or poorly written additions could inadvertently introduce SQL injection vulnerabilities.
* **Mitigation**: Continue using parameterized queries and ensure no raw SQL execution is used.

**3. Hardcoded Admin Credentials**

* **Vulnerability**: The admin username and password are hardcoded and predictable.
* **Impact**: Attackers could use this to gain unauthorized access if the system is deployed as is.
* **Mitigation**: Generate the admin credentials dynamically or allow configuration through environment variables.

**4. Insecure Password Policy**

* **Vulnerability**: The password policy is non-existent (no minimum length, complexity, or entropy requirements).
* **Impact**: Users can create weak passwords, making brute-force attacks easier.
* **Mitigation**: Enforce a robust password policy, such as requiring a mix of uppercase, lowercase, numbers, and special characters.

**5. Lack of Rate Limiting**

* **Vulnerability**: There is no mechanism to limit login attempts.
* **Impact**: Attackers can attempt brute-force or credential stuffing attacks without restriction.
* **Mitigation**: Implement rate limiting or account lockouts after a certain number of failed attempts.

**6. No Secure Communication**

* **Vulnerability**: The application does not specify how communication occurs (e.g., over HTTP/HTTPS).
* **Impact**: If this were deployed without TLS (HTTPS), credentials could be intercepted in transit.
* **Mitigation**: Ensure that the application is deployed over a secure channel using HTTPS.

**7. No Session Management**

* **Vulnerability**: The application does not implement proper session management after login.
* **Impact**: If the application were extended for multi-page functionality, lack of session handling could expose users to session hijacking.
* **Mitigation**: Use secure session tokens stored in memory or cookies with flags like HttpOnly and Secure.

**8. No Input Validation for Registration**

* **Vulnerability**: The username and password fields are not validated for length, special characters, or injection risks.
* **Impact**: Malicious inputs could crash the application or lead to unexpected behavior.
* **Mitigation**: Add input validation for both username and password fields.

**9. No Audit or Logging Mechanism**

* **Vulnerability**: The application does not log user activities (e.g., failed login attempts, account deletions).
* **Impact**: Malicious activities may go undetected.
* **Mitigation**: Implement logging for critical events while ensuring sensitive information (e.g., passwords) is not logged.

**10. Potential Denial-of-Service (DoS)**

* **Vulnerability**: The application does not protect against resource exhaustion attacks (e.g., creating many accounts in a short time).
* **Impact**: An attacker could overload the database or application resources.

**Mitigation**: Add safeguards such as rate-limiting and CAPTCHAs for account creation.