**Progressive Web App (PWA) - README**

**Overview**

This Progressive Web App (PWA) is designed to provide a secure, fast, and reliable user experience. This README outlines the vulnerabilities fixed within the application and provides instructions on how to set up and run the PWA.

**Fixed Vulnerabilities**

1. **Vulnerability 1 – SQL Injection**
   * **Fix**: Used prepared statements and parameterised queries in SQL to prevent injection attacks. All database queries are sanitised to ensure no malicious input is executed.
2. **Vulnerability 2 – Reflective Cross Site Scripting (Self XSS)**
   * **Fix**: Implemented input sanitisation and output encoding across the app. All user-generated content is properly encoded to prevent script execution.
3. **Vulnerability 3 – Command Execution**
   * **Fix**: Secured shell commands and system calls by sanitising inputs and using safe execution functions. Avoided direct execution of user inputs.
4. **Vulnerability 4 – Clickjacking**
   * **Fix**: Implemented X-Frame-Options and Content-Security-Policy headers to protect the app from clickjacking attacks by blocking it from being embedded in iframes.
5. **Vulnerability 5 – Cross Site Request Forgery (CSRF)**
   * **Fix**: Introduced CSRF tokens for every form submission to ensure that the requests are legitimate and come from the app itself. The app checks for the presence and validity of CSRF tokens.
6. **Vulnerability 6 – Directory Traversal**
   * **Fix**: Ensured that any user-provided file paths are validated, and restricted file access to specific directories to prevent traversal outside the application’s designated folder structure.
7. **Vulnerability 7 – Stored XSS**
   * **Fix**: Used sanitisation libraries (like OWASP Java HTML Sanitizer) to ensure that any stored data containing user input is stripped of harmful scripts before being rendered.
8. **Vulnerability 8 – File Upload**
   * **Fix**: Applied strict file validation checks for file type and size. Disabled the execution of scripts in uploaded files and used secure storage practices for uploaded files.
9. **Vulnerability 9 – Broken Access Control / IDOR (Insecure Direct Object References)**
   * **Fix**: Implemented proper access control checks to ensure that users can only access objects and resources that they are authorised to view.
10. **Vulnerability 10 – Unencrypted Communication – Man in the Middle**
    * **Fix**: Forced HTTPS across the application using HTTP Strict Transport Security (HSTS). All sensitive data in transit is encrypted using SSL/TLS.
11. **Vulnerability 11 – User Enumeration**
    * **Fix**: Standardised error messages during authentication and login attempts to prevent attackers from identifying whether a user exists in the system.
12. **Vulnerability 12 – Information Leakage – Server**
    * **Fix**: Removed detailed error messages and stack traces. Configured the server to only return generic error messages to avoid disclosing information about the server or application logic.
13. **Vulnerability 13 – Password Management – Plain Text Storage**
    * **Fix**: Implemented secure password storage using hashing algorithms like bcrypt or Argon2. Passwords are never stored in plain text and are properly salted.
14. **Vulnerability 14 – Privilege Escalation**
    * **Fix**: Reviewed and strengthened role-based access control (RBAC) to prevent users from gaining access to higher privilege levels through misconfigurations or bypass techniques.
15. **Vulnerability 15 – Denial of Service (DoS) Attacks**
    * **Fix**: Implemented rate limiting and IP blocking to prevent abuse of resources. The application also checks for resource consumption patterns that could indicate a DoS attack and limits access accordingly.