Week 1 – Security Assessment Documentation

This document outlines the activities, testing methodology, findings, and learning outcomes from **Week 1: Security Assessment** conducted as part of my cybersecurity internship and training tasks.

The objective of this week was to gain hands-on experience in **basic web application security assessment**, identify common vulnerabilities, and document findings in a structured and professional manner.

1. Understanding the Application

1.1 Application Setup

A mock web-based application was selected from GitHub for cybersecurity testing purposes.

Setup Steps:

- Installed project dependencies using Node Package Manager (npm)
- Started the application using the following commands:

```
npm install
npm start
```

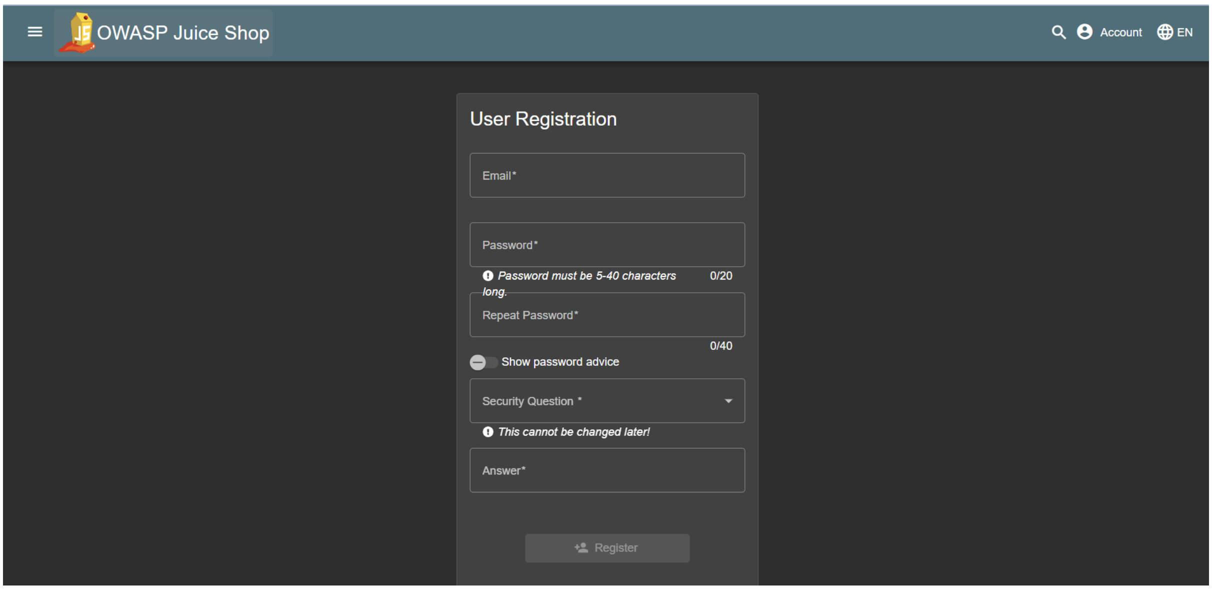
- Accessed the application locally at:

```
http://localhost:3000
```

1.2 Application Exploration

The following core functionalities were explored to understand application behavior and user interaction:

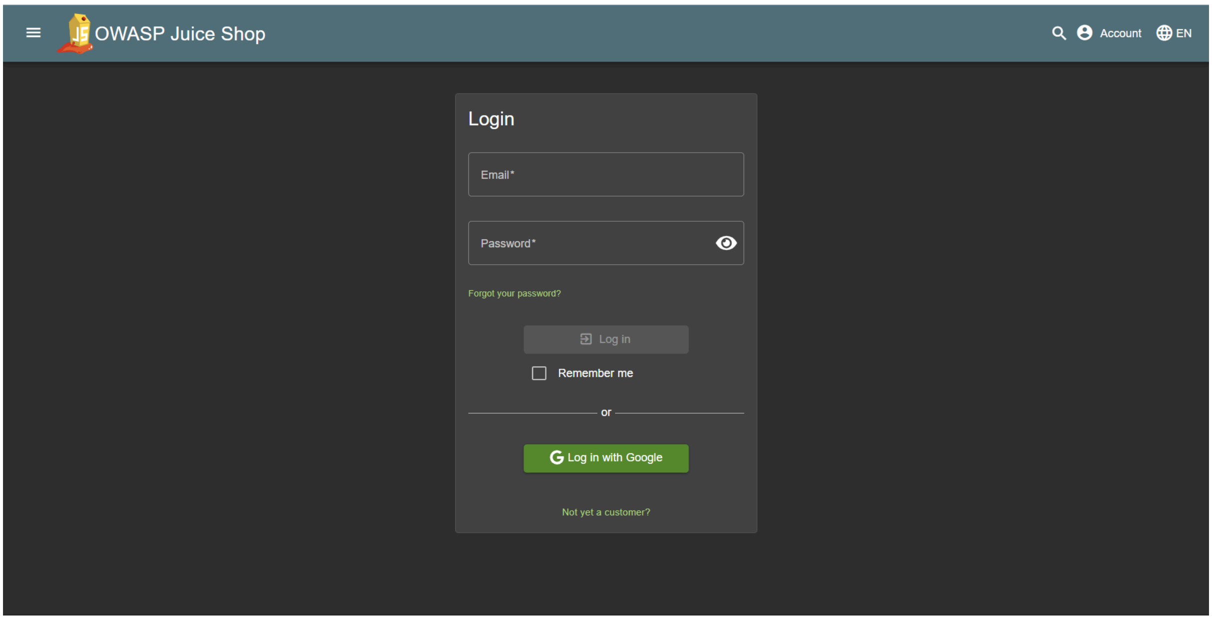
- User Signup Page



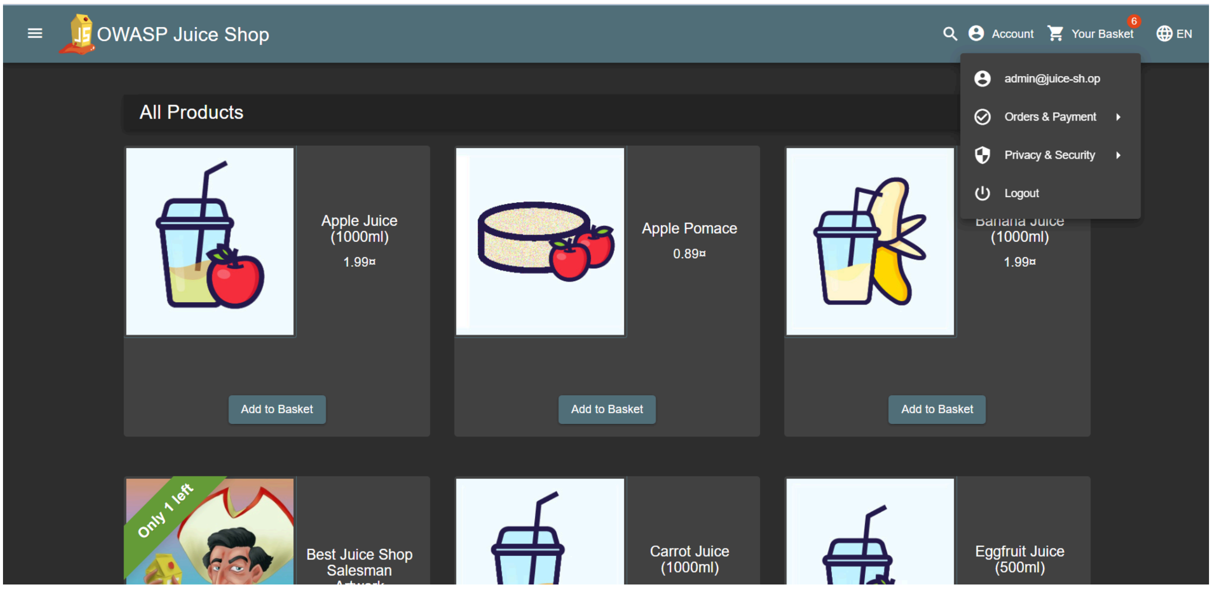
Password must be 5-40 characters long.

This cannot be changed later!

- User Login Page



- User Profile Page



This initial exploration helped identify **input fields**, authentication logic, and areas where user-supplied data is processed.

2. Basic Vulnerability Assessment

A basic security assessment was performed using both **automated tools** and **manual testing techniques** to identify common web application vulnerabilities.

2.1 Tools Used

The following tools were used during the assessment:

- **OWASP ZAP** – Automated web application vulnerability scanner
- **Browser Developer Tools** – Manual inspection and client-side testing
- **Web Browser (Chrome/Firefox)** – Application interaction and testing

2.2 Cross-Site Scripting (XSS) Testing

Objective:

To determine whether user input is properly validated and encoded before being rendered in the browser.

Methodology:

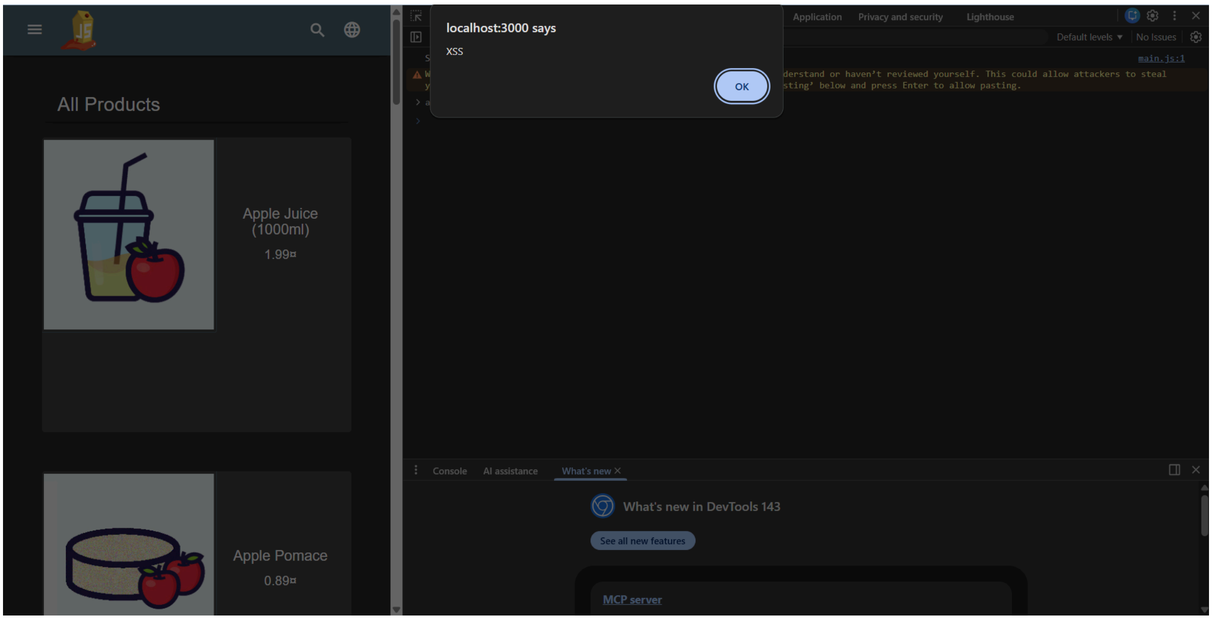
- Identified text input fields across the application
- Injected the following test payload:

```
<script>alert('XSS');</script>
```

- Observed browser behavior to check if JavaScript execution occurred

Result:

- Successful execution of JavaScript indicated the presence of **Cross-Site Scripting (XSS)** vulnerabilities



localhost:3000 says

XSS

```
derstand or haven't reviewed yourself. This could allow attackers to steal
sting' below and press Enter to allow pasting.
```

2.3 Basic SQL Injection Testing

Objective:

To test whether authentication mechanisms are protected against SQL Injection attacks.

Methodology:

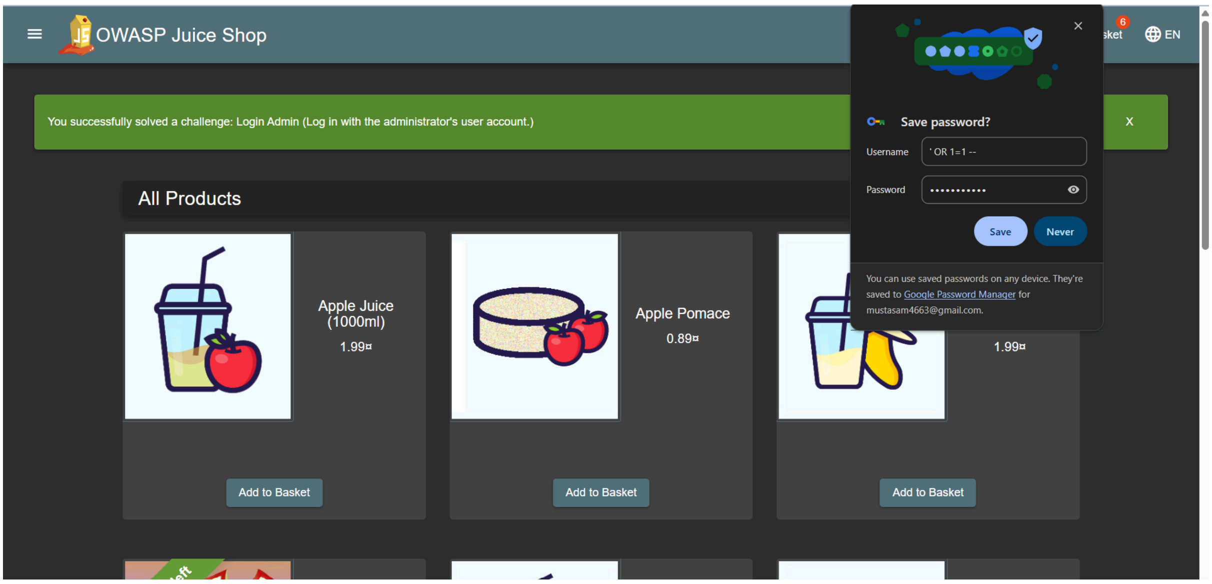
- Navigated to the login page
- Entered the following payload in both username and password fields:

```
admin' OR '1'='1
```

- Observed login behavior

Result:

- Successful login or abnormal behavior indicated potential **SQL Injection (SQLi)** vulnerability



You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

Save password?

Username: ' OR 1=1 --

Password: •••••••••••

You can use saved passwords on any device. They're saved to Google Password Manager for mustasam4663@gmail.com.

2.4 Additional Focus Areas

During testing, special attention was given to:

- **Weak password storage practices** (plain text or weak hashing indicators)
- **Security misconfigurations**, such as:
  - Verbose error messages
  - Lack of input restrictions
  - Missing validation mechanisms

3. Findings Documentation

3.1 Vulnerabilities Found

1. **Cross-Site Scripting (XSS)**
   - User input executed as JavaScript in the browser
   - Indicates lack of proper input sanitization and output encoding
2. **SQL Injection (SQLi)**
   - Authentication bypass possible using crafted input

    - Indicates insecure query handling

3. **Security Misconfigurations**
    - Insufficient input validation
    - Inadequate error handling

3.2 Areas of Improvement

Based on the identified vulnerabilities, the following improvements are recommended:

- Implement strict **input validation and sanitization**
- Apply proper **output encoding** to prevent XSS
- Use **parameterized queries / prepared statements** to prevent SQL Injection
- Improve **password storage** using strong hashing algorithms
- Disable detailed error messages in production environments
- Conduct regular security testing using automated and manual techniques

4. Learning Outcomes

By completing Week 1 tasks, the following skills and knowledge were developed:

- Understanding of basic web application architecture
- Hands-on experience with **OWASP ZAP**
- Practical understanding of **XSS and SQL Injection**
- Familiarity with browser-based security testing
- Ability to document security findings professionally

5. Ethical Considerations

All testing activities were performed:

- On **mock or intentionally vulnerable applications**
- In a **local and authorized testing environment**
- Strictly for **educational and learning purposes**

No real-world or production systems were targeted.

6. Conclusion

Week 1 successfully introduced foundational concepts of web application security assessment. The activities helped build a strong understanding of how common vulnerabilities arise and how they can be identified using basic tools and techniques.

This documentation serves as a record of learning and practical work completed during **Week 1: Security Assessment**.

📌 *This document is part of my cybersecurity internship learning portfolio.*