# Online Shopping System Test

Contents

[Online Shopping System Test](#_Toc172575676)

[1. Introduction](#_Toc172575677)

[2. The User Experience](#_Toc172575678)

[i. Registering a User](#_Toc172575679)

[ii. Profile Page](#_Toc172575680)

[iii. Logout](#_Toc172575681)

[3. Security](#_Toc172575682)

[i. Hashing Passwords](#_Toc172575683)

[ii. SQL Injection](#_Toc172575684)

[iii. Brute Forcing](#_Toc172575685)

[4. Miscellaneous Tests](#_Toc172575686)

[i. Browsers](#_Toc172575687)

[5. Conclusion](#_Toc172575688)

[6. References](#_Toc172575689)

## Introduction

This document aims to thoroughly test the Colchester Groceries Online Shopping System, hosted at <https://jamesedney.eu.pythonanywhere.com/colchestergroceries/> . It will test the user experience, as well as the proposed security measures.

## The User Experience

### Registering a User

A screenshot of a computer

Description automatically generatedA white background with black and white clouds

Description automatically generated with medium confidenceThe first test to carry out will be to register as a genuine user. Selecting register at the log on page and filling out the form takes the user to the 2FA setup, as expected. Scanning the QR code with the 2FA application provides the user with a 30 second TOTP, and using that to login takes the user to the homepage . This is demonstrated in the flow chart below. Due to Google’s security policy, it is not possible to provide a screenshot of the TOTP.

*Above: A flowchart showing a successful login*

### Profile Page

The profile page has been implemented to enhance the user experience. The screenshot shows that the profile page has been successfully populated from the SQL database.

A white background with black lines

Description automatically generated

*Above: The profile page populated from the details stored in the SQL database.*

### Logout

A screenshot of a computer

Description automatically generatedAnother important for the user is to ensure their session information has been removed once they logout. The below screenshot shows the user information being stored in a base64 encoded format.

*Above: The session information being stored in base64 encoded format, as can be seen in the bottom right.*

Pressing logout and checking the developer tools shows the session storage is no longer stored, as seen below.

A screenshot of a computer

Description automatically generated

*Above: All session information being removed when the user logs out.*

**Test Result:** These basic tests verify a user is able to register to the online shopping system. They also check that the profile page is correctly populated, and that the user can log out without their session information being stored.

**Recommendations:** In future iterations of the system, SMS or emailed based 2FA will improve the user experience, negating the need for a separate application. Enforcing more complex passwords is also recommended. Another feature that will enhance the user experience will be the ability to amend their details.

## Security

### Hashing Passwords

A black background with white text

Description automatically generatedThe proposal for the Online Shopping System suggested to hash passwords as a way to enhance security and prevent the risk of information disclosure. To check the above password is correctly hashed, it is necessary to check the SQL databases. Checking the database shows that the password is hashed using PBKDF2, as shown in the screenshot below.

*Above: Passwords hashed in the SQL database.*

### SQL Injection

The proposal for the system also suggested measures to prevent SQL injection attacks. The Online Shopping System checks to make sure the user can’t input any characters that would cause this type of attack. The screenshot below shows this. By using a well known SQL injection statement (Portswigger, N.D.) the system returns warning message.

**A screenshot of a phone

Description automatically generated**

*Left: An SQL attack returning a warning message*

### Brute Forcing

The Online Shopping System also uses a lockout mechanism to ensure a malicious user cannot brute force the system and gain access. The below screenshot shows the outcome of inputting incorrect or non existent credentials multiple times.

*Left: Fake credentials being input, causing an ambiguous error message*

A screenshot of a login form

Description automatically generated

*A green and white text

Description automatically generated*

*Above: The lockout screen*

**Test Result:** These tests check the proposed security measures. SQL injection attacks have been prevented through the use of regular expressions, and brute forcing has been prevented through the use of a lockout mechanism. Passwords are hashed using the NIST recommended PBKDF2 with SHA256 (OWASP, N.D.).

**Recommendations:** The security mechanisms could be further enhanced with the use of secure cookies (Singh, 2020) as opposed to the fairly easy to decrypt base64 session cookie .

## Miscellaneous Tests

### Browsers

The Online Shopping System was tested on three different browsers to ensure the experience remains the same. No differences were noted.

A screenshot of a computer

Description automatically generated

*Above: The Online Shopping System on Microsoft Edge.*

*A screenshot of a computer

Description automatically generated*

*Above: The Online Shopping System tested on Google Chrome*

*A screenshot of a computer

Description automatically generated*

*Above: The Online Shopping System tested on DuckDuckGo (Mobile)*

## Conclusion

This document tested the Colchester Groceries Online Shopping System. The system proved intuitive to use and the security measures were successfully implemented. To improve the application, it is recommended to implement SMS or email based two factor authentication to improve the user experience and negate the need for an extra application.

## References

* OWASP. (N.D.) Password Storage Cheat Sheet. Available from: <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html> [Accessed 20 July 2024].
* Portswigger. (N.D.) SQL Injection. Available from: <https://portswigger.net/web-security/sql-injection> [Accessed 22 July 2024].
* Singh, T. & Meenakshi (2020) Prevention of session hijacking using token and session id reset approach. *International journal of information technology* 12 (3): 781–788. DOI: 10.1007/s41870-020-00486-w