# Web Shop

Tarun Mangla (930676), Syed Taha Bin Zafar (928486), Raed Bin Shahid (928449)


## ABSTRACT

Modern e-commerce is a billion-dollar industry which relies on security to combat attackers. Security is needed in e-commerce websites to secure customer data and to prevent misuse of website functions. Unsecure e-commerce websites without its needed security will fail to provide its basic shopping function. This project aims to build a build a secure web application which deals with the most common vulnerabilities found on the web. According to the Open Web Application Security Project (OWASP), SQL injection, cross site scripting (XSS), broken authentication, cross site request forgery (CSRF) and sensitive data exposure are some of the common vulnerabilities found on the web[1]. In this project we have built a shopping website from ground up and have manually handled the security features without the use of a security framework.


**Keywords:** web security, web shop, php, mysql ,csrf, XSS. SQL injection, authentication

## 1. INTRODUCTION

This project aims to implement the basic features which are needed for the function of a normal e-commerce shopping website and then reinforce it with some security features. Firstly, the developer of any website needs to understand is that how the attackers attack the website and what are some of the security risks in the application and how they can be patched up and handled. The diagram below shows the different paths the attacker takes to attack the web application.

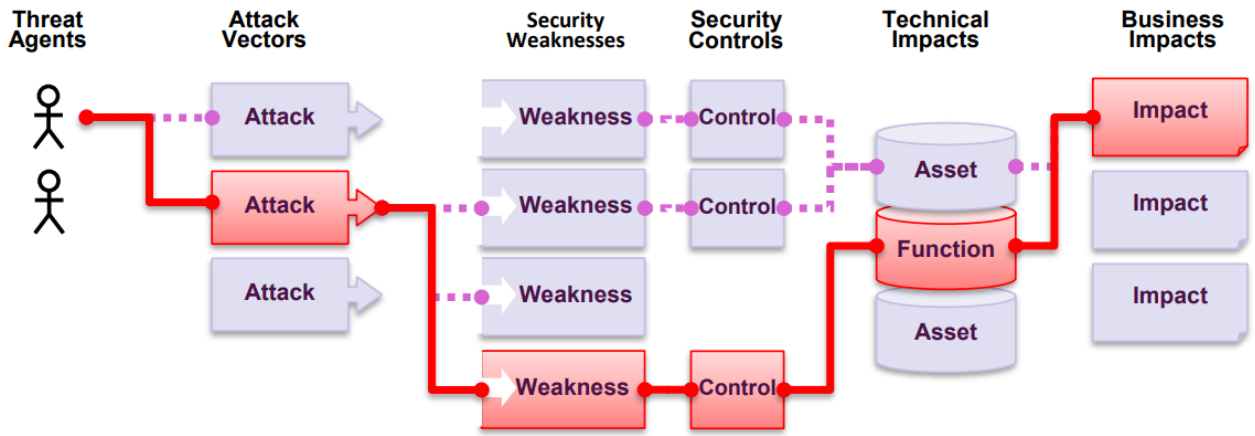


Figure 1: Application Security Risks[1]

The attackers use different attack vectors which target the security weaknesses of the websites and bypass the placed security controls to either manipulate the assets or the function of the website. These security weaknesses need to be removed and the security controls hardened during the development of the website.

### 1.1 Features and Deliverables

There are some basic features which a shopping website needs to have to be able to carry out e-commerce function. Given below are some of these features which are implemented in the project:

- User management (Signup / Login)
- Display of products
- Shopping cart management (Through cookies and session)
- Order management (Details and confirmation)

These e-commerce features need to be secured so that they're not misused by the attackers. Given below are some of attack vectors and security functions which are implemented in the project.

- Secure session and cookies handling

- Secure password storage (encryption)
- SQL injection handling
- XSS handling
- CSRF handling
- Secure logic design

# 2. PROJECT DEVELOPMENT AND WORKING METHODOLOGY

## 2.1 Development Approach

Various application development methodologies are provided by Software Development Life Cycle (SDLC) concepts. In our approach we selected the “Waterfall” Model. The reason for choosing this model is that it defines deliverables clearly in every phase, incremental resource commitment and isolation of the problem early in the process.

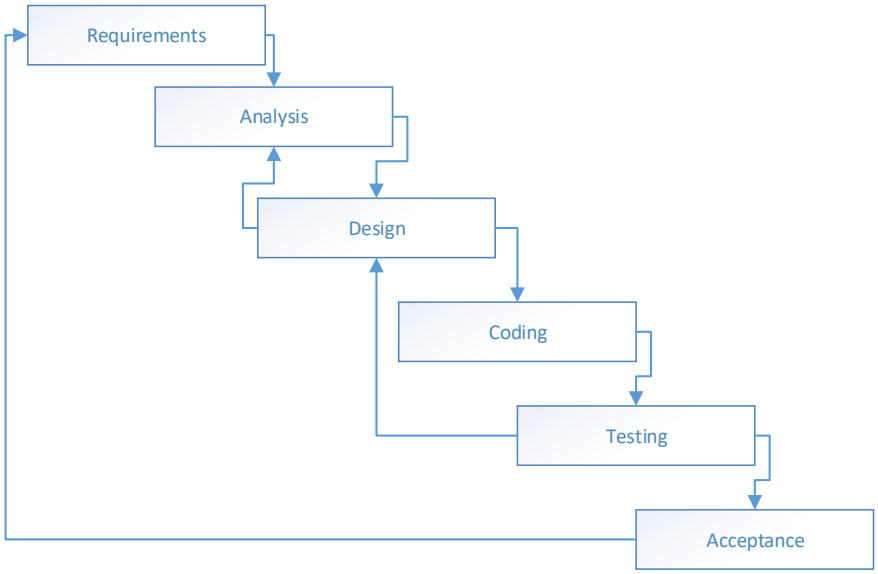


Figure 2: Waterfall Model

## 2.2 Development Environment

To accomplish this project, we have used various platforms and tools which are described as follows:

- XAMPP: It is an open source web solution server stack which combines PHP, MYSQL and an Apache web server in a single installation.
- PHP: It is one of the most reputed server-side scripting language of the web. Most of our web development is done on PHP.
- MYSQL: It is a RDBMS. We have stored our backend data on MYSQL server.
- Apache HTTP Server: It is a web server developed by Apache Software Foundation. This server handles the incoming and outgoing requests from the website.
- Bootstrap: It is an open source frontend framework which is used to design web applications and websites. We are using this framework to design our UI/UX
- jQuery: It is a JavaScript library which is used to simplify client-side scripting of HTML. Some of the Bootstrap functions make use of this library.

## 2.3 Database Design

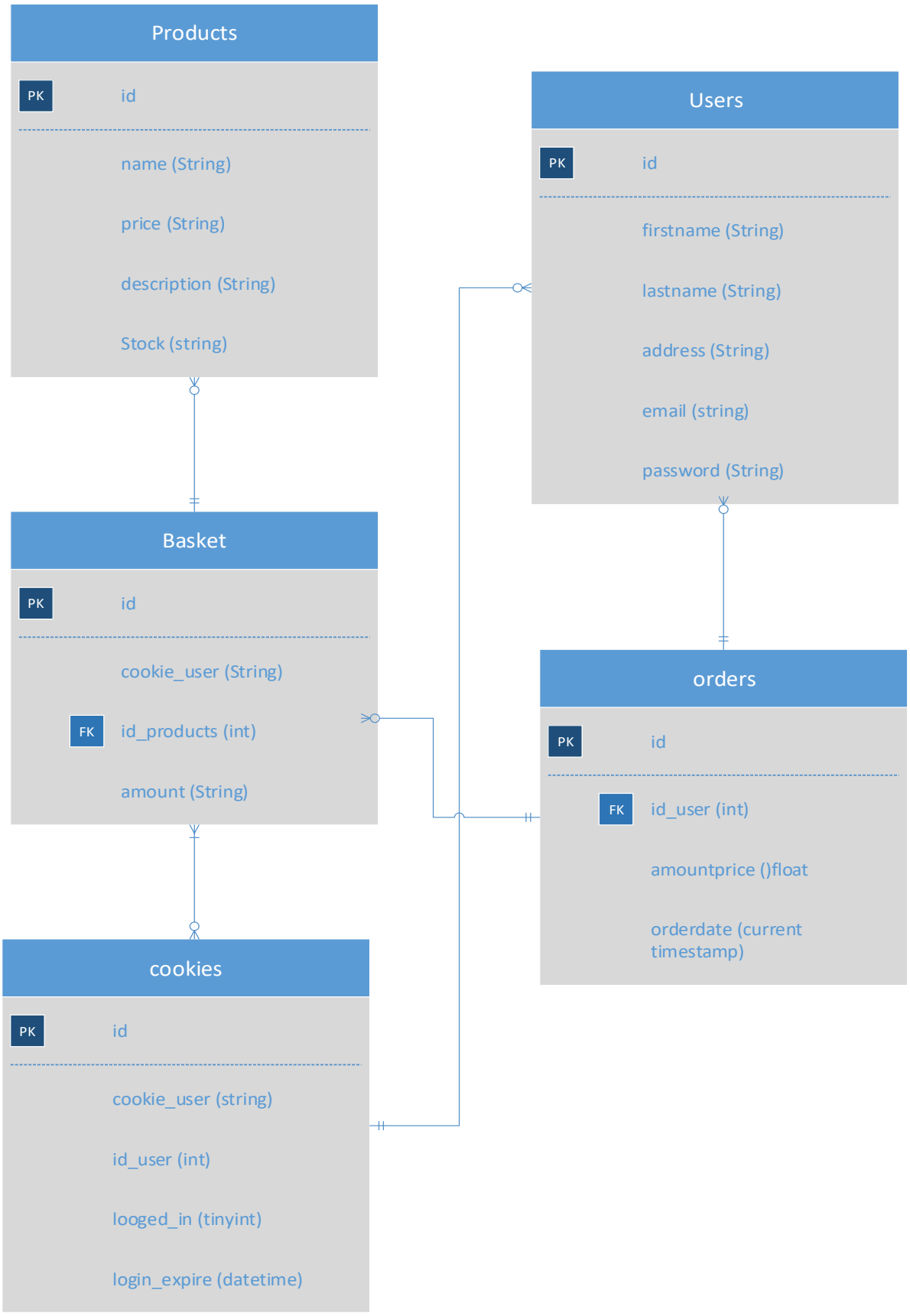


Figure 3: Entity Relationship Model

### 2.4 Website Flow Chart

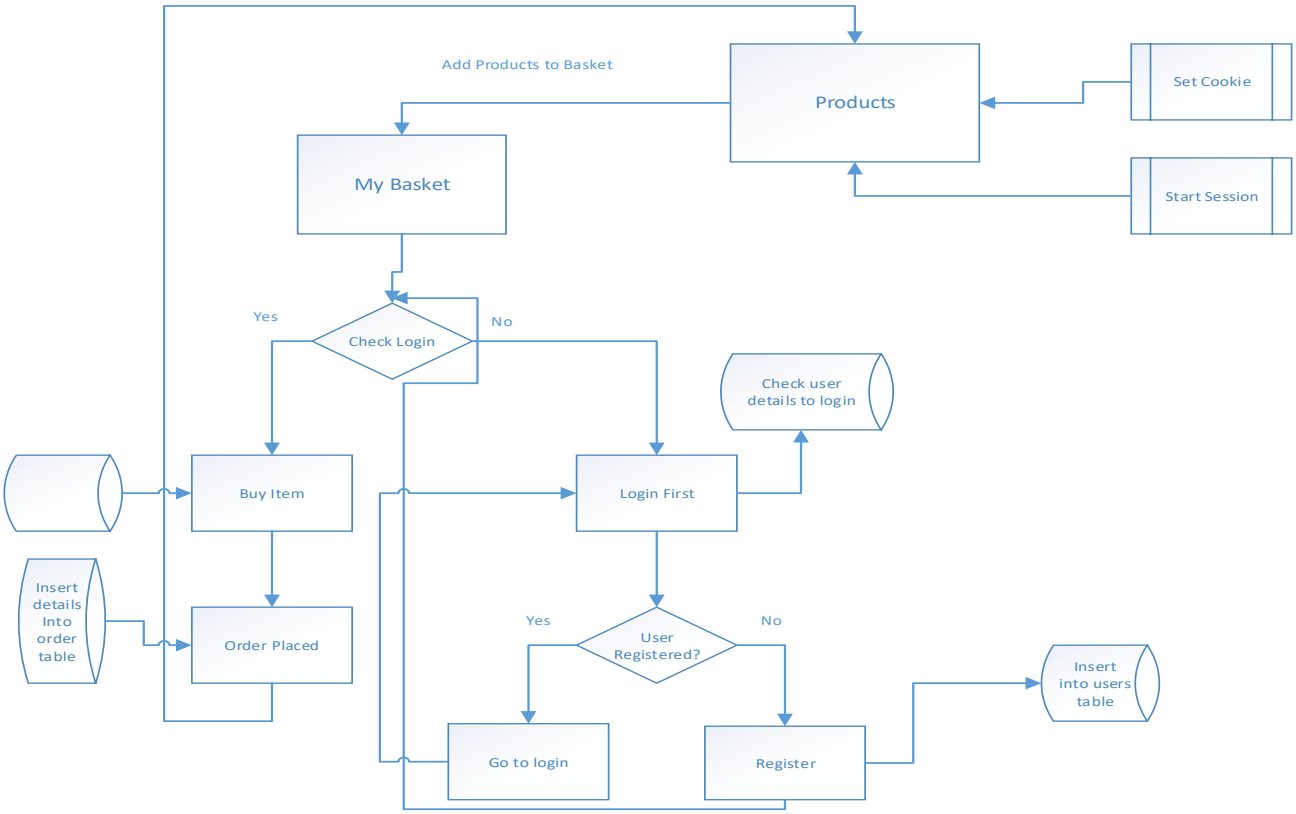


Figure 4. Website Work Flow Model

## 3. PROJECT RESULT

At the end we were able to build a working website with all the mentioned features in it. Below are some of the screenshots of the working pages.

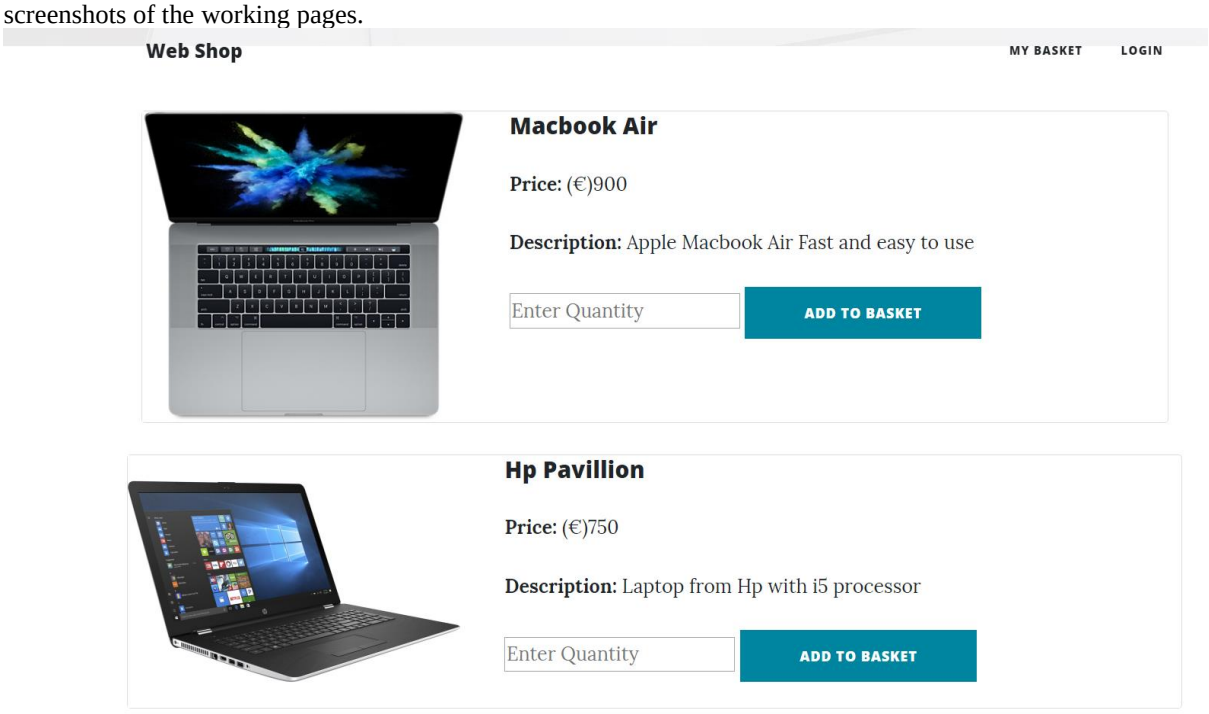


Figure 5. Products page

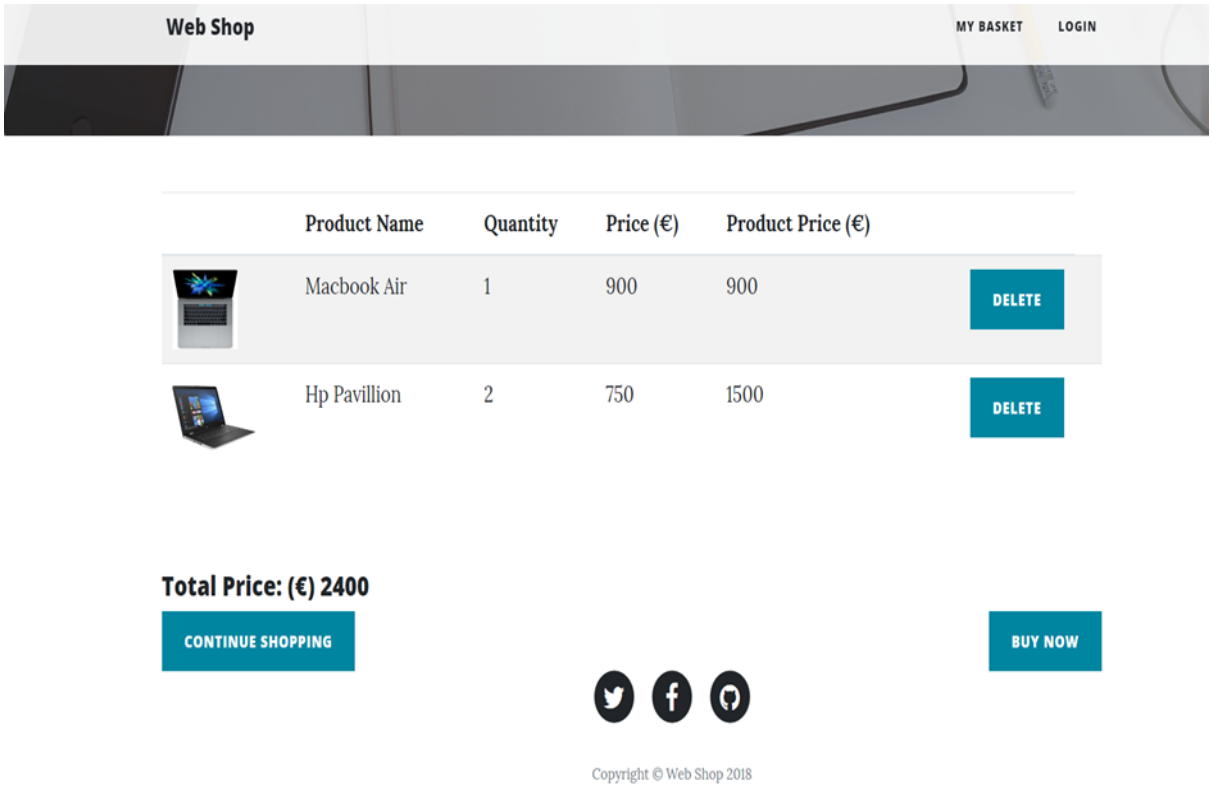


Figure 6: Basket page

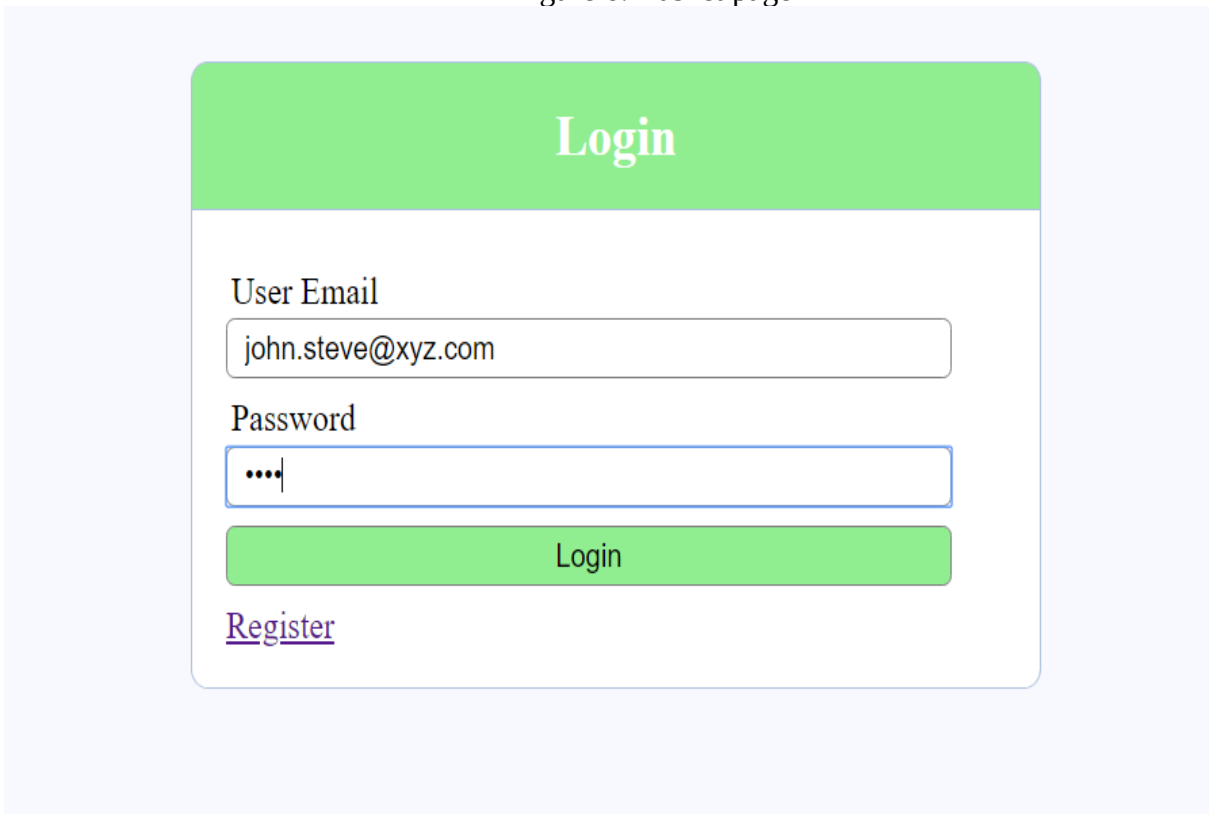


Figure 7: User login

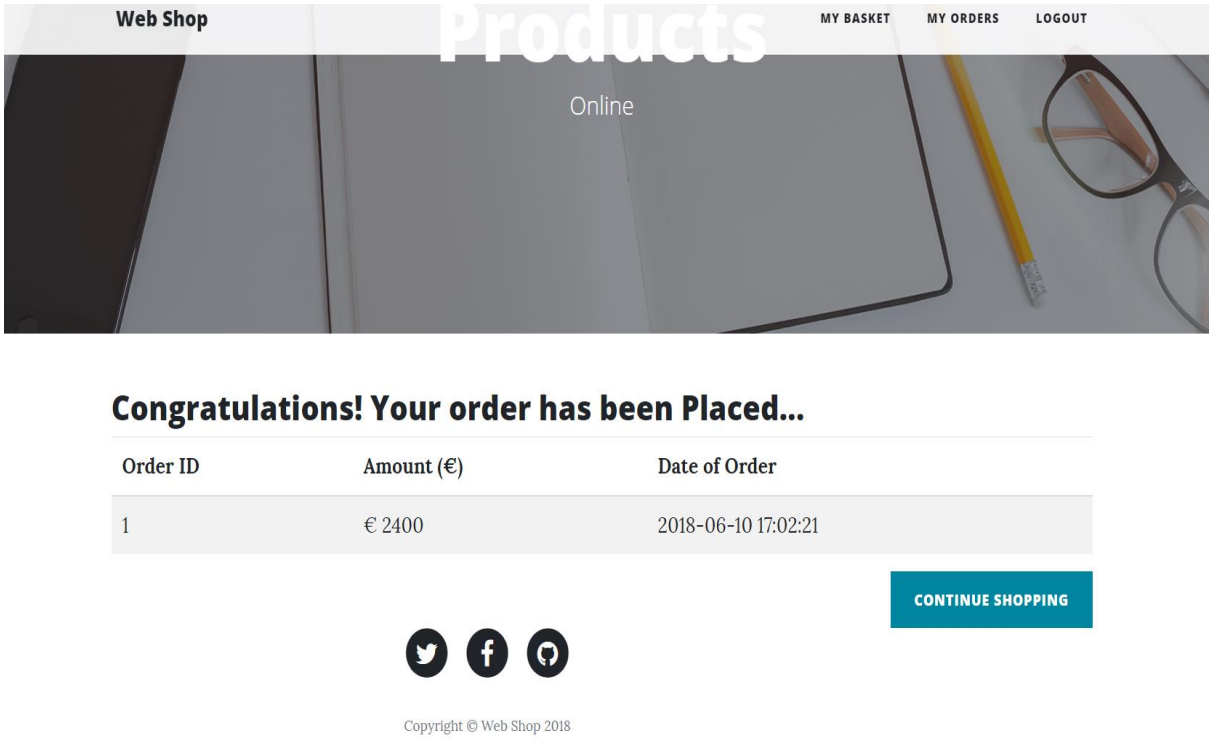


Figure 8: Order page

# 4. SECURITY FEATURES

## 4.1 Secure Session and Cookie Handling

To develop a functioning e-commerce website, we need to keep track of several actions as the user browses through the website. When they log in, when they go to different pages, access control, shopping cart management etc. are a few examples. Thus sessions are used to keep track of user identification and their state. Cookies are also used to enhance this functionality. We have used sessions and cookies securely with timeout functionality along with storing its relations in database. Given below is some of the code of these features.

```php
if(isset($_SESSION['LAST_ACTIVE']) && $_SESSION['LAST_ACTIVE']+120 <= time() ){
  session_destroy();
}
```

```php
public function setCookie(){
    $time = time() + (86400 * 200 );
    if (!isset($_COOKIE['cookie_user'])){
        $cookie_user = md5(openssl_random_pseudo_bytes( length: 32));
        setcookie( name: "cookie_user", $cookie_user, $time);
    }
    if(isset($_SESSION['userid'])){
      $user_id=$_SESSION['userid'];
      $logged_in=1;
    }
    if(!isset($_SESSION['userid'])){
      $user_id=0;
      $logged_in=0;
    }
    $date=date( format: 'Y-m-d H:i:s', $time);
      $this->addCookies($_COOKIE['cookie_user'],$date,$user_id,$logged_in);
}
```

Refer to the all.php class file to see all the code related to session and cookies.

### 4.2 Secure Password Storage (Encryption)

Password is a personal and sensitive data which must be stored in the database with proper security. Any leakage of this information can affect the company and its customers in a negative way. One way to store the password is through hashing and salting. Hash is the result of a one-way function which maps a data of arbitrary size (password) to a data of fixed size (hash). Salt which is a random data is added to the hashed password. Using built-in PHP methods this is carried within our project.

```
$pass_hash = password_hash($password,  algo: PASSWORD_DEFAULT);
```

By using the password_hash() method, the result of pass_hash can be stored in the database. The password can then be retrieved using the password_verify() method which returns a Boolean value.

```
password_verify($pass,$fetchedPass)
```

### 4.3 SQL Injection Handling

SQL injection is one of the most used attack technique according to OWASP. It is the scenario where the attacker inserts/injects a custom SQL query in the web application to exploit database functionality. One way to handle SQL injection attacks in PHP is by using prepared statements through PHP Data Object (PDO). It is more secure because SQL queries are not built on the fly but instead they are prewritten and precompiled and they only require inputs to execute.

```
$this->_conn = new PDO( dsn: 'mysql:host=localhost;dbname=webshop',  username: 'root',  passwd: '');
```

```
public function getuserdetails(){
 $sql = "select * from users where id=?";
try{
    $statement = $this->_conn->prepare($sql);
    $statement->execute(array($_SESSION['userid']));
    $fetchData = $statement->fetchAll();
    return $fetchData;
  }
  catch (PDOException $e){
  echo $e->getMessage();
  }
}
```

### 4.4 XSS Handling

XSS or cross site scripting is another popular attack technique which allows the attackers to insert malicious client-side JavaScript code into web pages which is then viewed by other users. This can be used by attackers to bypass same-origin policy access control. This attack can be used to access any session tokens, cookies or other sensitive information which is used with that website or retained by the browser. There are various ways through which XSS attacks can be handled, in our project we are using the Apache configuration of XSS protection to handle these attacks.

```
<IfModule mod_headers.c>
    Header set X-XSS-Protection "1; mode=block"
    Header always append X-Frame-Options SAMEORIGIN
    Header set X-Content-Type-Options nosniff
    Header set Content-Security-Policy "default-src 'self' 'unsafe-inline';"
</IfModule>
```

### 4.5 CSRF Handling

Cross-Site Request Forgery (CSRF) is in attack scenario where the end user is tricked into submitting malicious request. The user is forced to execute unwanted actions for which they are authenticated on a web application. These attacks do not perform theft of data but instead target state changing requests. To prevent these attacks, we are using secure tokens which only the server and the browser knows. This token is added as a hidden field in the forms. When these forms are submitted a check is made if the submitted token matches the stored token and the request fails if the tokens do not match.

```
$csrf = md5(openssl_random_pseudo_bytes( length: 32));
if (!isset($_SESSION['csrf'])){
    $_SESSION['csrf'] = $csrf;
}
```

```
<input type="hidden" name="csrf" value="<?=$_SESSION['csrf'];?>">
```

```
if(isset($_POST["submit"]) && ($_POST['csrf'] == $_SESSION['csrf'])){
$all->Userregister();
}
```

# 5. FINAL WORDS

## 5.1 Difficulties Faced

In every type of project undertaking, the members of the project face both expected and unexpected difficulties as the project progress moves forward. These problems are needed to be addressed in due time. One of the first problem we faced was learning the PHP programming language. All team members had experience with multiple languages but we weren't exposed to PHP before doing this project. Thus, we had to learn the language from scratch in a short amount of time which was a challenge. Having exposure to other programming languages helped us learn PHP in reasonable time. Collaboration and teamwork is another major challenge which we experienced while working on the project. It is quite a herculean task to make the members of a team come on the same page. Everything we say and do is quite subjective and because of that we have to make sure that all of us understand the matter at hand the same way. Arranging meetups and completing the milestones was another challenge as we were falling behind the planned schedule. But somehow we managed to complete the project in due time.

## 5.2 What We Learned

Working on this project was quite a journey. It broadened our horizons in multiple disciplines. Learning a new programming language is always quite a fun and interesting experience which we had to do. We got to learn about web technologies, how the web pages send requests to the servers and how the servers handle the requests. Learning about sessions, cookies and how everything communicates with the database was somewhat new too. The most fun part about the project was learning about the security vulnerabilities of web pages and how attackers try to exploit this. Learning how to make web pages secure against these vulnerabilities was another enjoyable process. Learning the security aspect of the web allowed us to think about the internet in ways which we hadn't thought before. Working in a team allowed us to be self-responsible and self-manage our tasks. We think this is a good way to get project work exposure and give us an idea of how bigger projects work in the industry itself.

## 5.3 Conclusion

All in all, we managed to complete the project in time. As far as we know, we completed all the required deliverables related to a working web shop and its security. Web security is an ever-evolving landscape where we get to learn about new vulnerabilities every other day and then these vulnerabilities are patched and secured by security experts. This course gave us a good jumpstart into the field of web security and we hope that it helps us in our careers in the future.

# REFERENCES


[1] OWASP, "OWASP Top 10 - 2017", https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf (10 June 2018)

[2] PHP, "PHP: Sessions - Manual", http://php.net/manual/en/book.session.php (10 June 2018)

[3] PHP, "PHP: Encrypted Storage Model - Manual", http://php.net/manual/en/security.database.storage.php (10 June 2018)

[4] PHP, "PHP: SQL Injection - Manual", http://php.net/manual/en/security.database.sql-injection.php (10 June 2018)

[5] OWASP, "PHP Security Cheat Sheet - OWASP", https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet (10 June 2018)

[6] Shiflett Chris CS., Essential PHP Security, O'Reilly Media, February 2009