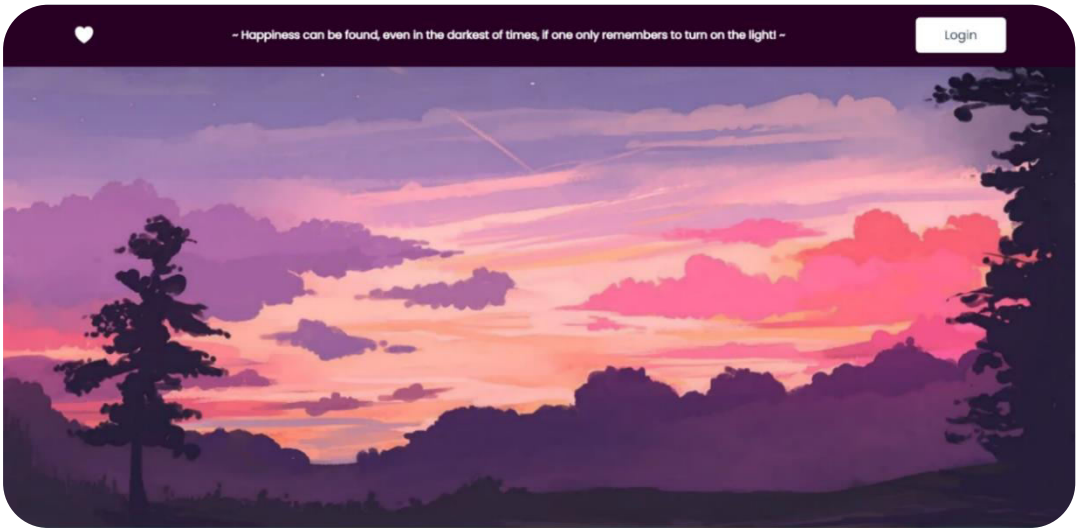
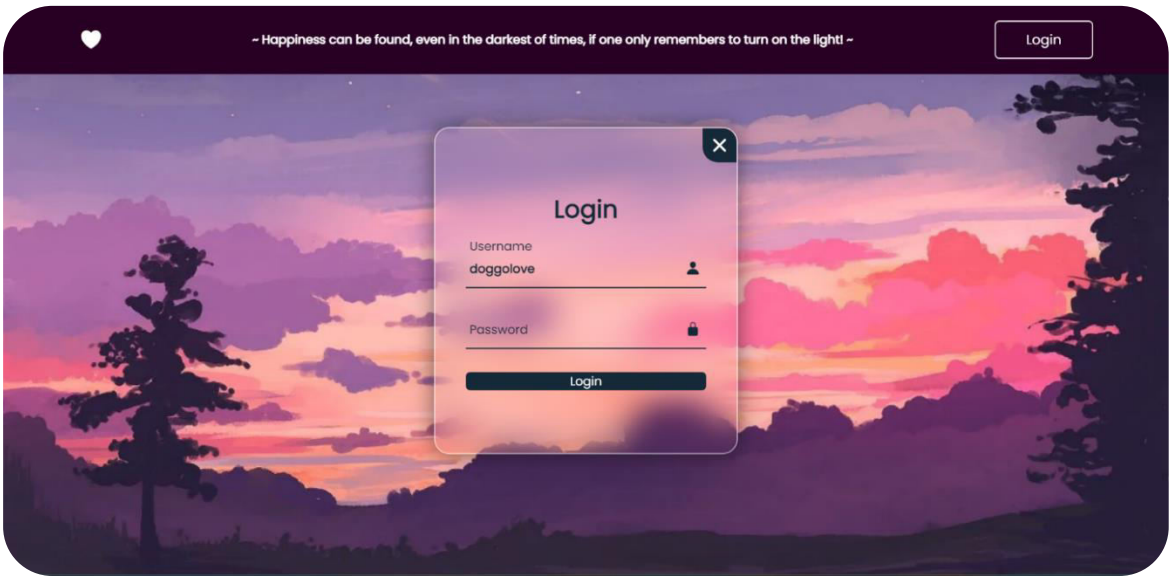
Project Documentation

The site is made using xampp to help make the connection between backend and frontend. For it, the modules used are Apache and MySQL. A database is used to store the valid username and password combination that will allow the user to access the site content. To access the site, one must copy all the site files into the htdocs folder of xampp. After that, accessing http://localhost will allow one to access the site. Upon entering the site, the user is presented with a simple page with a Login button in the top right corner, that lights up when one hovers with the cursor over it. Keep in mind this button, as it might be a tool that could hint to a secret later on.



After pressing the Login button, a Login form will appear (that can once again be closed if we press on the X button in the top right corner). For the purposes of this project, we will be using the user 'doggolove', a user that is already registered. However, the password is unknown and it isn't an extremely common one so brute-forcing would not do the trick in this case.



Hint: Check if the site has any vulnerabilities to SQL injection and try to login. Trying to find the password is not going to get you anywhere.

SQL injection

Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '"' at line 1 in E:\xampp\htdocs\authentication.php:13 Stack trace: #0 E:\xampp\htdocs\authentication.php(13): mysqli_query(Object(mysqli), 'select *from lo...') #1 {main} thrown in **E:\xampp\htdocs\authentication.php** on line **13**

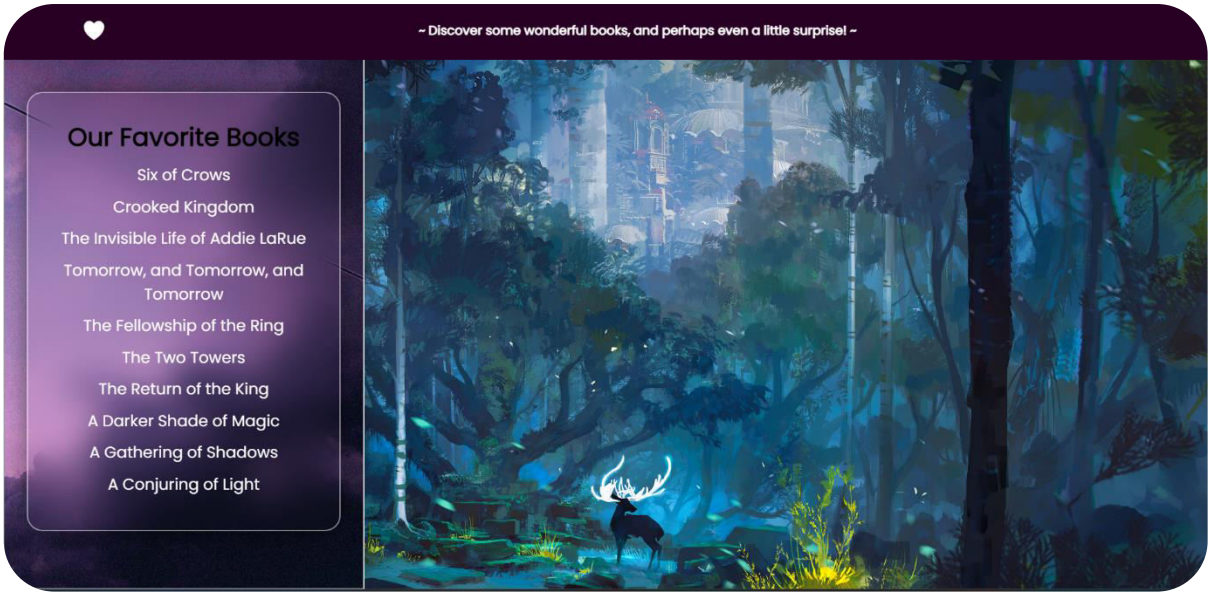
The vulnerability that can be exploited is SQL injection, allowing us to inject a query into the site and bypass the login process even though we did not enter the correct user credentials. To check if the site is vulnerable, we can enter ' and we will be getting the message shown above. That error will indicate that the site is vulnerable.

Upon inserting ' OR '1'='1 query in the password field, we get access to the content of the site, since the query we entered is always true.

Prevention

In order to prevent from SQL injection, we have to make sure that the user is not allowed to enter characters that will be interpreted as SQL syntax. Therefore, we can use a function that will ensure that any special characters in the input are properly escaped before being used in the query. More details on this will be provided at the end.

After successfully accessing the account, we are sent to a page that presents some magnificent books.

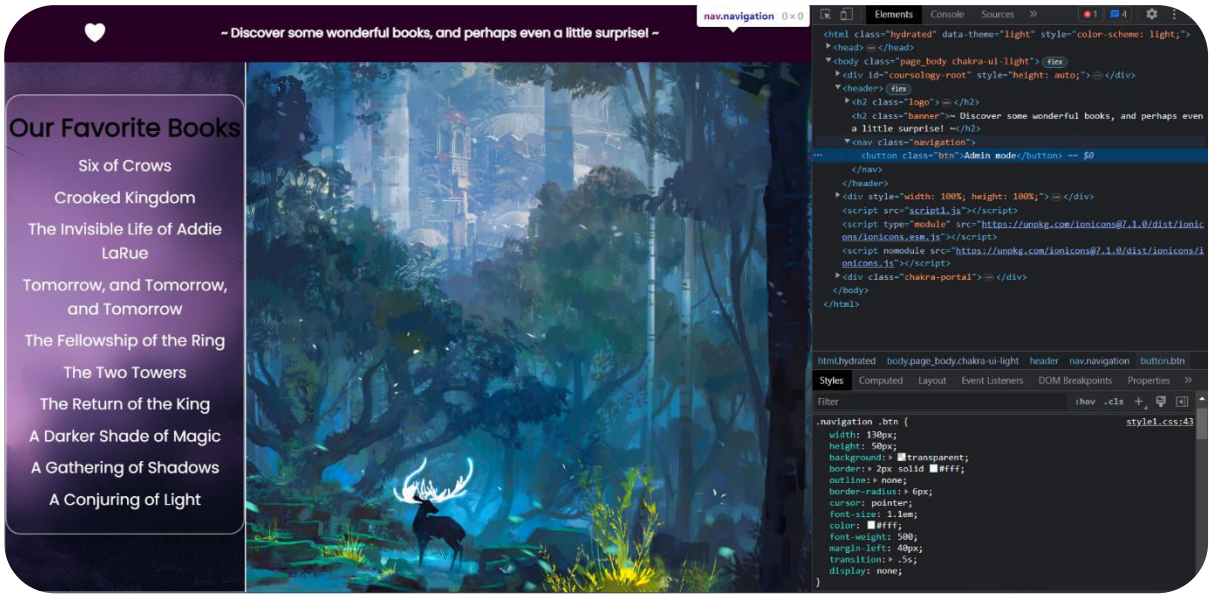


If you want to see more of the page, try to gain Admin access in order to reveal the secret.

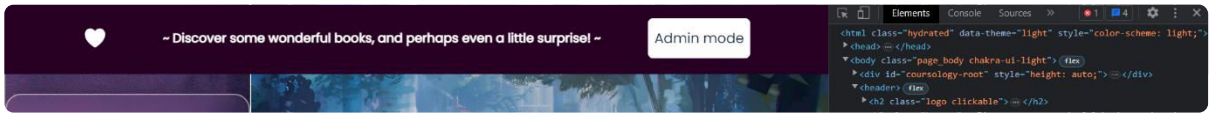
Hint: Remember that Login button from the previous page? Notice something different about this page? Try to study the code.

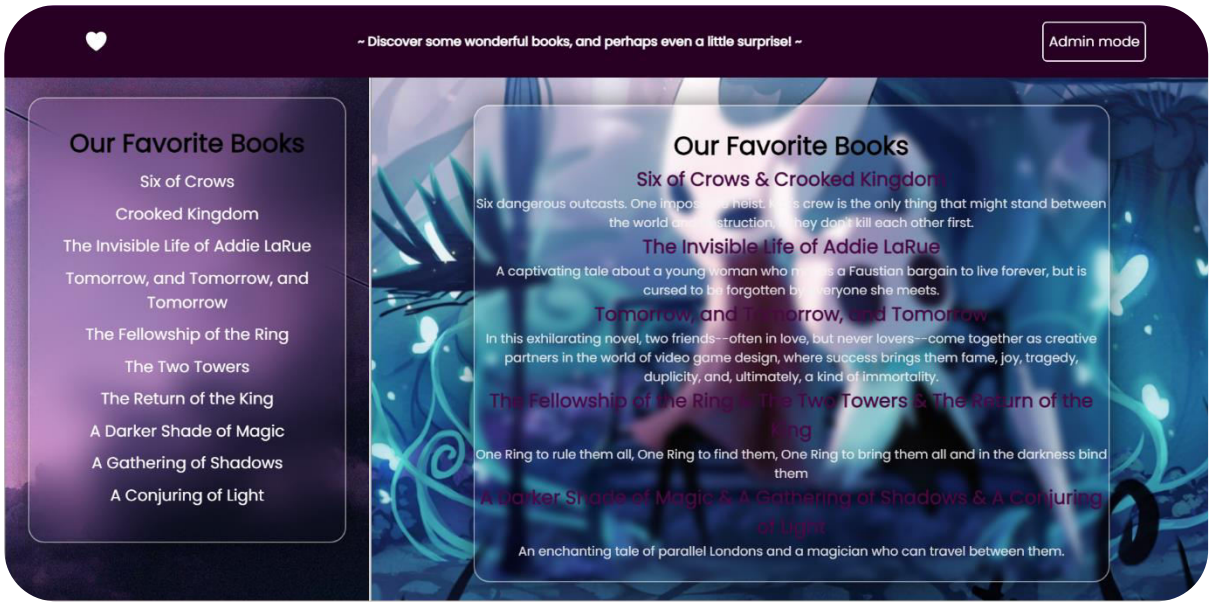
Secret revealed – Bypassing Client-Side Controls

After inspecting the source code, we can see that there is a button that does not seem to appear anywhere on the site. After looking in the css part, we notice the property “display: none”. If we remove that, we are able to access the Admin mode button.



Pressing on it causes the heart logo to become clickable, which then reveals the secret page upon being clicked.





Thank you for reading, I hope you enjoyed this little site, we sure had fun with it!

Code explained

SQL injection prevention

```php
<?php
    include('connection.php');
    $username = $_POST['user'];
    $password = $_POST['pass'];

        // $username = stripcslashes($username);
        // $password = stripcslashes($password);
        // $username = mysqli_real_escape_string($con, $username);
        // $password = mysqli_real_escape_string($con, $password);


        $sql = "select *from login where username = '$username' and password = '$password'";
        $result = mysqli_query($con, $sql);
        $row = mysqli_fetch_array($result, MYSQLI_ASSOC);
        $count = mysqli_num_rows($result);

        if($count >= 1){
           header("Location: index1.html");
        }
        else{
            echo "<h1> Login failed. Invalid username or password.</h1>";
        }
?>
```

This is the php code used for the authentication process. The code for preventing SQL injection is commented out, which leaves the app vulnerable, but if we remove the comments the special characters will be escaped and the SQL query the attacker might try to enter will not work, only showing that the login failed because of invalid username or password (even though we enter the same query that let us login before: ' OR '1'='1).

Login failed. Invalid username or password.

Secret explained – Bypassing Client-Side Controls

```
const btnPopup = document.querySelector('.btn');
const icon = document.querySelector('.logo');
const secret_page = document.querySelector('.secret');

btnPopup.addEventListener('click', () => {
    icon.classList.add('clickable');
});

icon.addEventListener('click', ()=> {
    secret_page.classList.add('show');
});
```

To implement this, there were multiple steps. First of all, the button has the "display: none" property that, when removed, makes the button once again visible. After clicking on it, the script shown above adds the clickable class to our icon. Initially, the icon can not be clicked, but when it gains the "clickable" class, it also gets the "pointer-events: auto" property, which allows the user to click on it.

```
.navigation .btn {
    width: 130px;
    height: 50px;
    background: transparent;
    border: 2px solid #fff;
    outline: none;
    border-radius: 6px;
    cursor: pointer;
    font-size: 1.1em;
    color: #fff;
    font-weight: 500;
    margin-left: 40px;
    transition: .5s;
    display: none;
}
```

```
.logo {
    pointer-events: none;
    font-size: 2em;
    color: #fff;
    user-select: none;
}

.logo.clickable {
    pointer-events: auto;
    cursor: pointer;
}
```

```
.secret {
   transform: scale(0);
}

.secret.show {
    transform: scale(1);
}
```

Lastly, the final of the script shown above adds the "show" class to our secret_page. This is rather simple, since the page is hidden without the "show" class, and gets revealed when it gains that class.

Running example of the project with video:

In the following video you have an example of how the site works and how to check and exploit its vulnerability.

https://youtu.be/9Xl0Ee436WA