Secure Web App

This is a self learning project where I built a simple website, and use various cybersecurity tools and techniques to understand the workings of web applications.

What I will be doing in this project:

1. Building a simple To-do-list web page
2. Identifying and testing vulnerabilities in the web page

The attacks I will be performing:

1. XSS attacks
2. localStorage manipulation

What will I learn from this project:

1. Front end development - HTML, CSS and Javascript
2. Web Application Vulnerabilities
3. How to patch vulnerabilities in web applications

Building the web page (To-do-list App)

1. Creating a HTML foundation
    a. Title
    b. Header
    c. Form
    d. Working Buttons
    e. Display field
    f. localStorage

Title

The title will be the name of the webpage. As you can see in the tab section of the browser.

Header

Header will be the main heading of the webpage. The name of the App

Form

Form will be where the user will be able to input and submit tasks. To add in their to-do-list.
It will contain 2 elements

a. Input field - “Enter a task”
b. Submit button - “Add”

Working Buttons

These buttons will be used to customize the information within the added task.
It will contain 2 buttons:

a. Edit - To edit selected task
b. Delete - To delete selected task

Display Field

Display field is where all the added, edited and deleted tasks will be shown. It will contain an unordered list, which will add tasks to it as per the user’s needs.
The tasks will be shown in vertical order with checkboxes to its left side. Selected items will be used to edit or delete the information within them.

Local Storage

Local Storage is where all the data is saved in the local browser. It is done so that in case the browser restarts or even closes, the data will still remain saved.

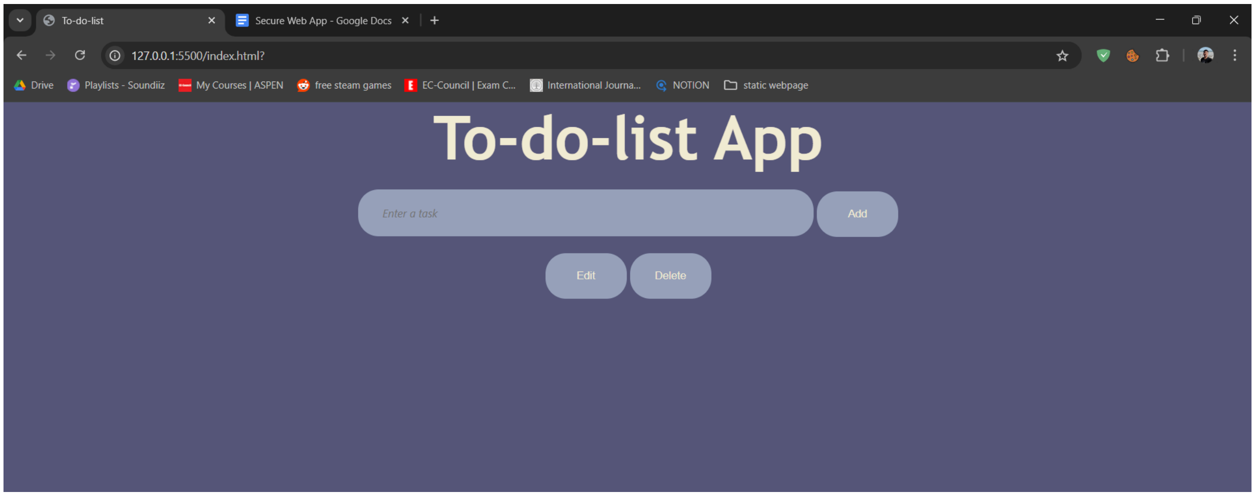
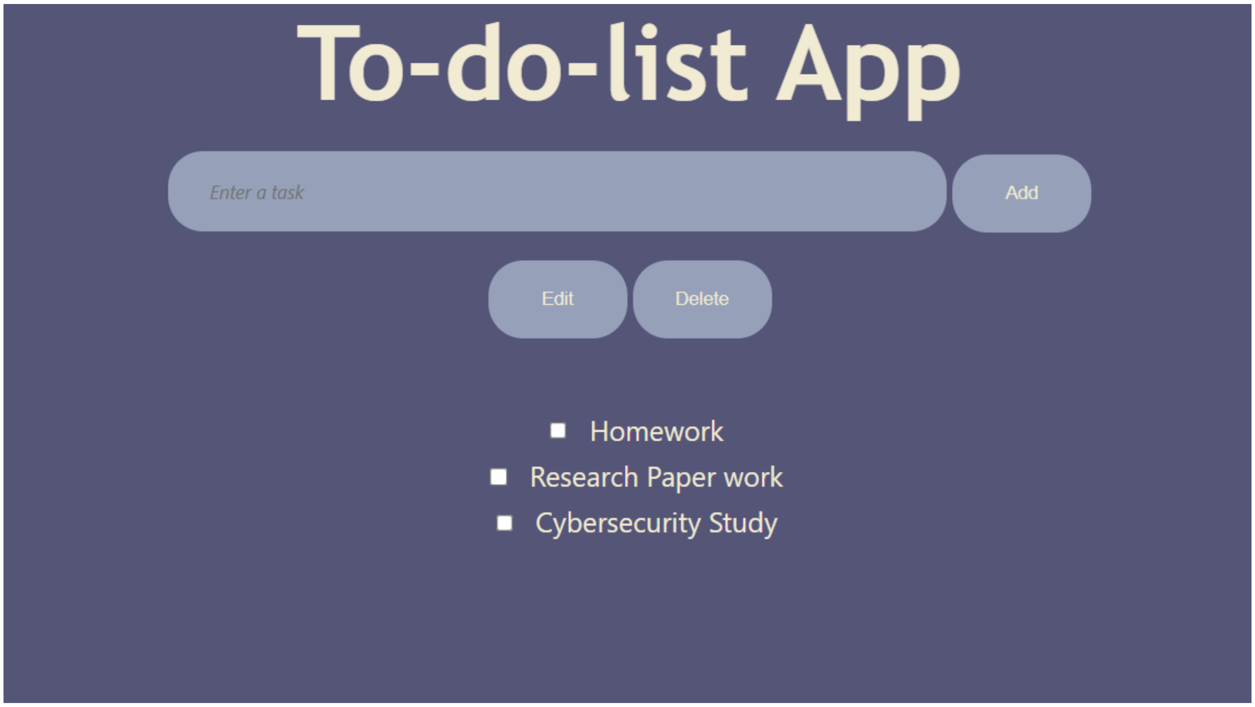
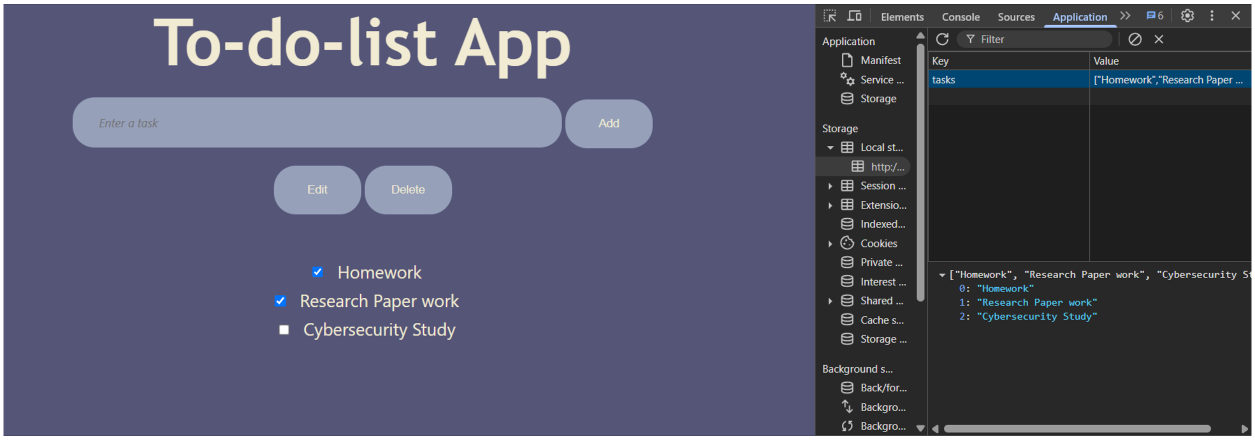


Fig. 1.1. To-do-list Web App



- ☐ Homework
- ☐ Research Paper work
- ☐ Cybersecurity Study

Fig. 1. 2. To-do-list Display Field



☑ Homework
☑ Research Paper work
□ Cybersecurity Study

Key	Value
tasks	["Homework","Research Paper ...

```
▾ ["Homework", "Research Paper work", "Cybersecurity St
    0: "Homework"
    1: "Research Paper work"
    2: "Cybersecurity Study"
```

Fig. 1. 3. Local storage of browser

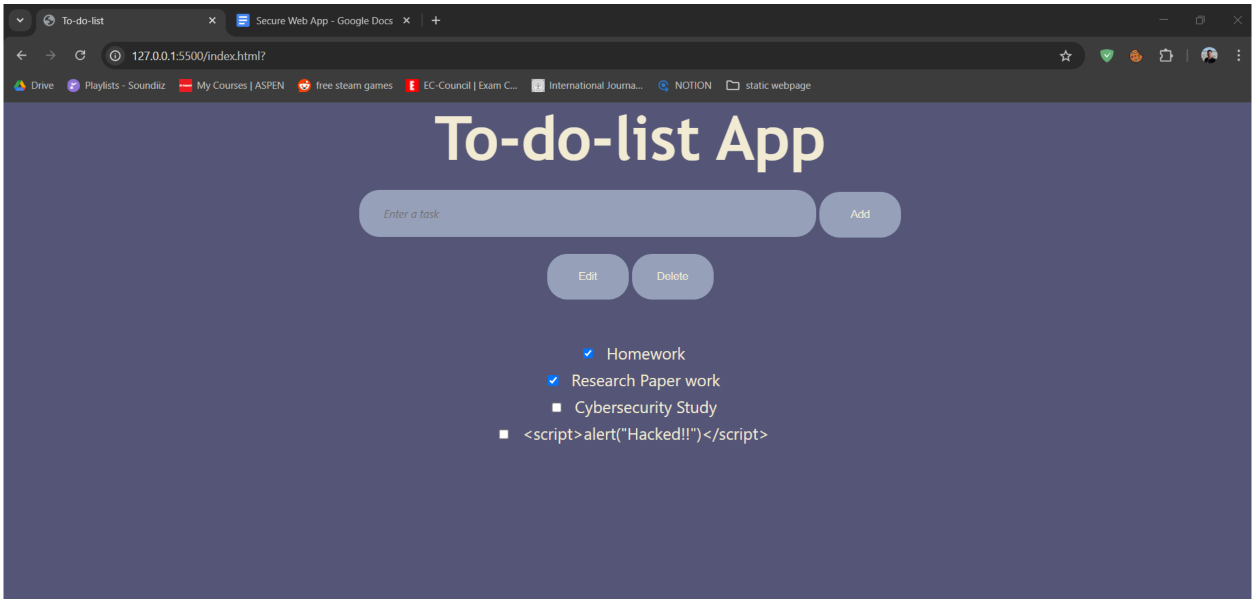
Finding Vulnerabilities of the Web app

1. XSS (Cross Site Scripting) attack

In this attack I will be running a javascript script in the input field of the web app.
Script:

<script>alert("Hacked!!")</script>

If the webpage is vulnerable to XSS attacks then the webpage will show an alert, with the message “Hacked!!”.



☑ Homework
☑ Research Paper work
□ Cybersecurity Study
□ <script>alert("Hacked!!")</script>

Fig. 2.1 Performing XSS attack on Web page

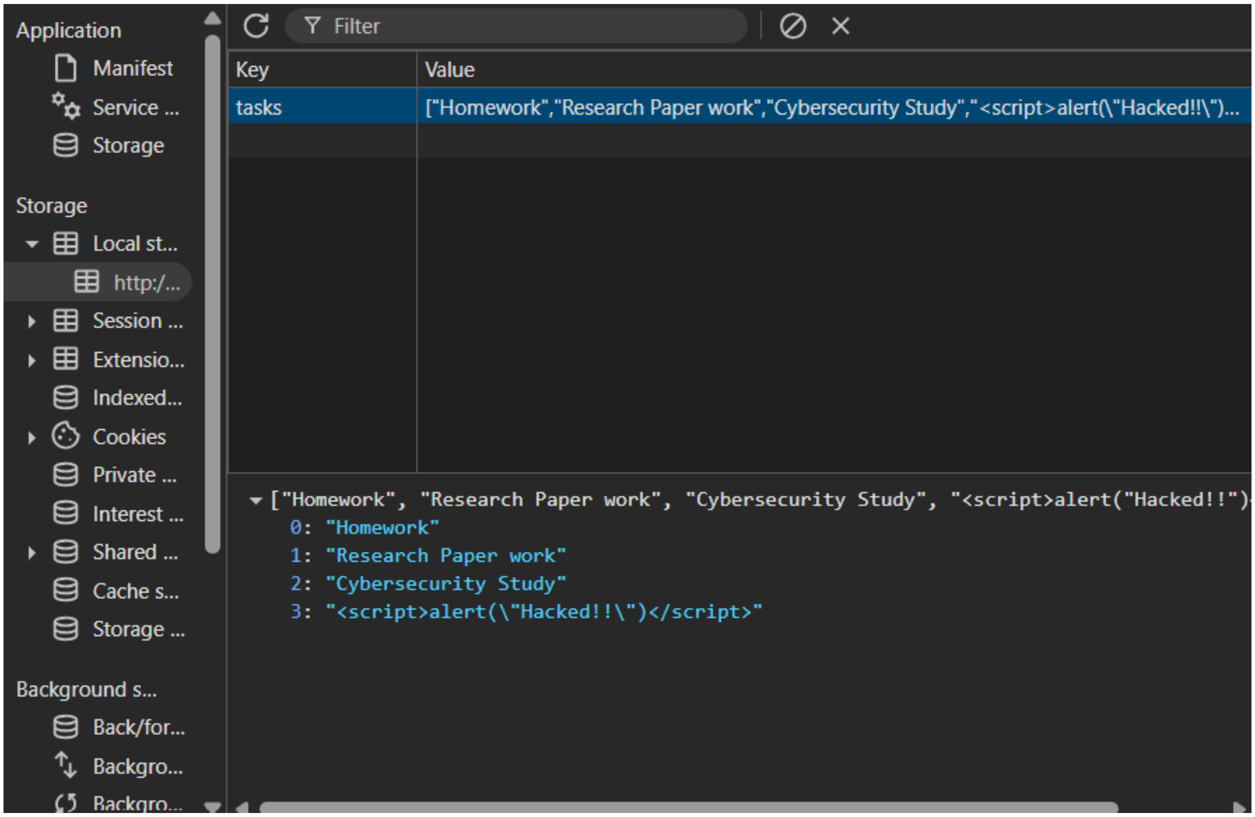
As you can see there was no alert message on the web page. Instead it took the script as a string and added it as a task.

Hence the webpage is not vulnerable to XSS attacks. Doing this the webpage does not allow DOM-based manipulation

Why XSS does not work:

1. Because during my javascript learning path. I learned another way of replacing the value of a variable through “textContent()”.
2. textContent automatically escapes HTML queries.
3. If I had used innerHTML instead of textContent, the attack would have worked. That is because innerHTML would have interpreted that command as a script and not as a string.

2. Local Storage Manipulation



Key	Value
tasks	["Homework","Research Paper work","Cybersecurity Study","<script>alert(\"Hacked!!\")...

```
▾ ["Homework", "Research Paper work", "Cybersecurity Study", "<script>alert("Hacked!!")
    0: "Homework"
    1: "Research Paper work"
    2: "Cybersecurity Study"
    3: "<script>alert(\"Hacked!!\")</script>"
```

Fig. 3.1. Local storage

Local Storage manipulation is when the attacker manipulates the values within the local storage to run malicious scripts.

For example,
Running Javascript scripts:
<script><h1>Hello</h1></script>

#which would add a header “Hello”.

As a result the script will not be executed because the rendering uses “textContent”, which does not interpret HTML or scripts.