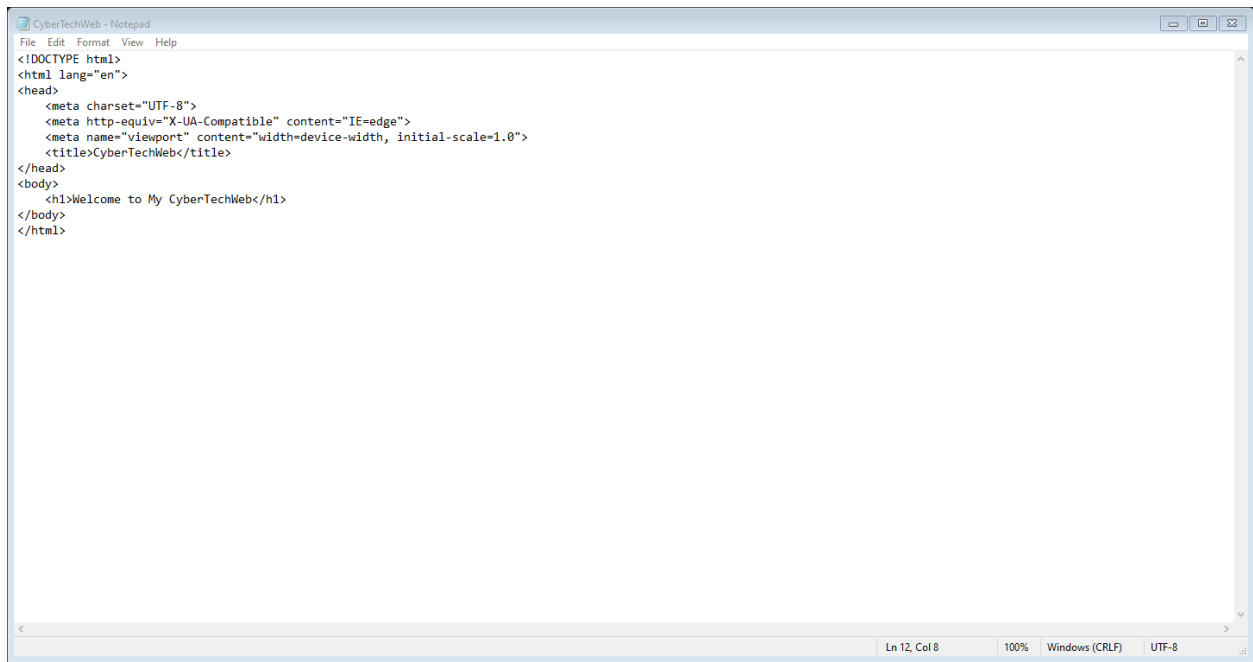


Secure a Web Application

Task 3:

Creating a simple Website of CyberTechWeb:



```
File Edit Format View Help
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CyberTechWeb</title>
</head>
<body>
  <h1>Welcome to My CyberTechWeb</h1>
</body>
</html>
```

Step 2: Apply Security Headers

Content Security Policy (CSP)

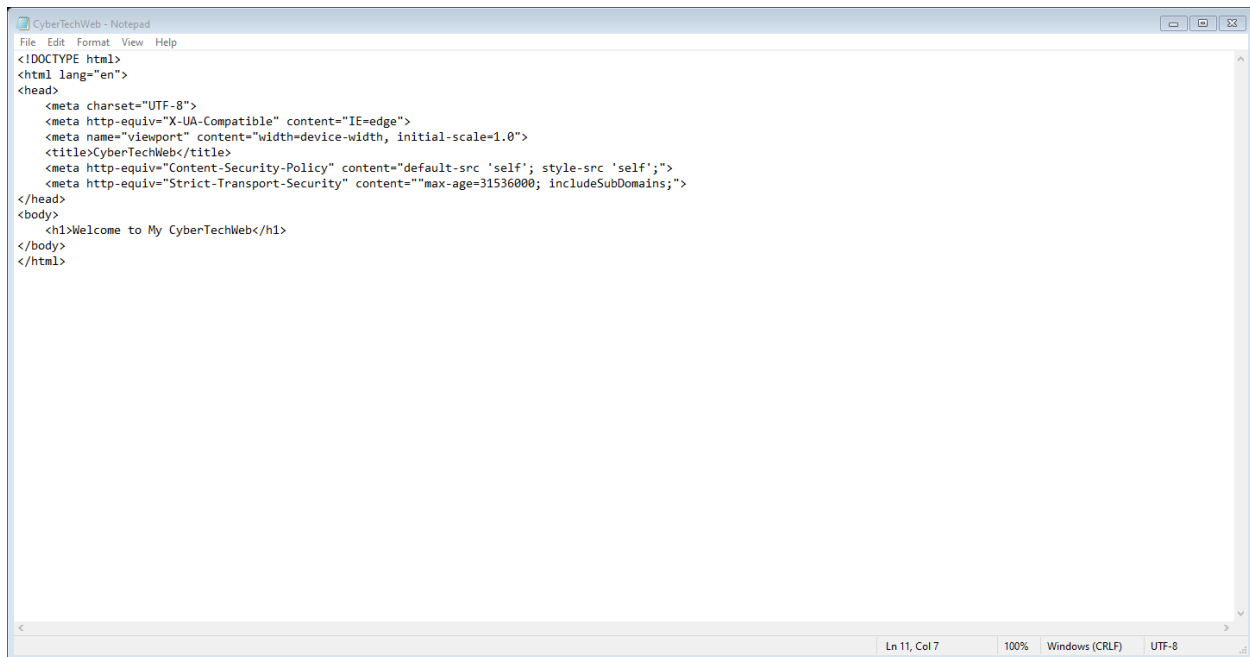
- **Overview:** CSP is a security feature that helps prevent various attacks, including Cross-Site Scripting (XSS) and data injection attacks.
- **Implementation:** To apply CSP, add the following HTTP header to your web server configuration:

Content-Security-Policy: default-src 'self' style-src 'self';

HTTP Strict Transport Security (HSTS)

- **Overview:** HSTS is a web security policy mechanism that helps protect websites against man-in-the-middle attacks such as protocol downgrades.
- **Implementation:** To enable HSTS, add the following header:

Strict-Transport-Security: max-age=31536000; includeSubDomains



```
File Edit Format View Help
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CyberTechWeb</title>
  <meta http-equiv="Content-Security-Policy" content="default-src 'self'; style-src 'self';">
  <meta http-equiv="Strict-Transport-Security" content="max-age=31536000; includeSubDomains;">
</head>
<body>
  <h1>Welcome to My CyberTechWeb</h1>
</body>
</html>
```

Review of created Simple Web Application for Common Vulnerabilities

SQL Injection Protection

SQL Injection typically occurs in backend code that interacts with databases. Since your HTML file doesn't currently include any form fields or server-side code, you're not vulnerable to SQL Injection at this stage. However, if you plan to add features like login or search forms, ensure you:

- Use prepared statements with parameterized queries.

- Always validate and sanitize user inputs before querying the database.

2. Cross-Site Scripting (XSS) Protection

XSS vulnerabilities occur when malicious scripts are injected into web pages that accept user input.

Steps to protect against XSS in your current code:

- Ensure any future user input (like forms or query parameters) is properly escaped before being output.

- Add CSP: You've already applied a Content Security Policy (CSP) to restrict loading resources from external sources. Your existing CSP in the file looks good:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; style-src 'self';">
```

This policy only allows scripts and styles from your own domain ('self'), helping to prevent XSS attacks.

3. Security Headers Review

Your **HSTS header** is applied correctly, but there's a small syntax issue. The meta tag should not have double quotes around the max-age attribute. It should look like this:

<meta http-equiv="Strict-Transport-Security" content="max-age=31536000; includeSubDomains;">