Module 1 – SE -Overview of IT Industry(LAB)

**1) Write a simple "Hello World" program in two different programming languages of your choice. Compare the structure and syntax.**

Ans:

PYTHON:

* print(“Hello World”)

JAVA:

* public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

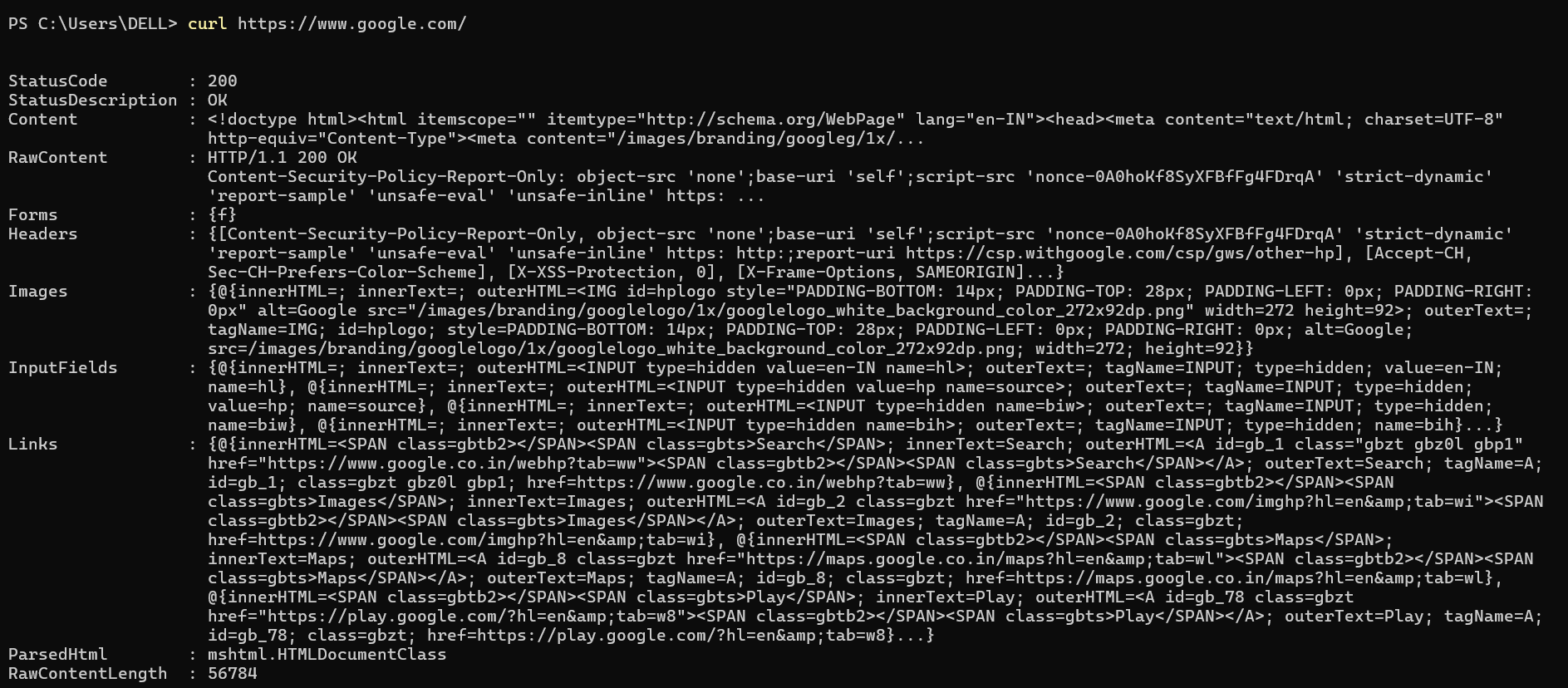
}

| **Feature** | **Python** | **Java** |
| --- | --- | --- |
|  |  |  |
| **Code Length** | Very short (1 line) | More detailed (5+ lines) |
| **Syntax Simplicity** | Simple and easy to read | Verbose and structured |
| **Entry Point** | No special entry point | Requires main() method |
| **Semicolon** | Not required | Required at the end of statements |
| **Curly Braces** | Not used | Required to define code blocks |
| **Typing** | Dynamically typed | Statically typed |

**2) Simulate HTTP and FTP requests using command line tools (e.g., curl).**

**Ans.**

* HTTP Request Simulation:
* curl <https://www.google.com/>
* This fetches the website's HTML and shows it in your terminal.



**3) Identify and explain three common application security vulnerabilities. Suggest possible solutions.**

**1. SQL Injection: Unsafe vs Safe**

**❌ Vulnerable Code (Unsafe)**

python

CopyEdit

# vulnerable\_sql.py

import sqlite3

conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute("CREATE TABLE users (username TEXT, password TEXT)")

c.execute("INSERT INTO users VALUES ('admin', 'admin123')")

username = input("Enter username: ") # Attacker types: admin' OR '1'='1

query = f"SELECT \* FROM users WHERE username = '{username}'"

print("Query:", query)

c.execute(query)

if c.fetchone():

print("Logged in")

else:

print("Login failed")

**💥 Output (Injection):**

Enter username: admin' OR '1'='1

Query: SELECT \* FROM users WHERE username = 'admin' OR '1'='1'

Logged in ✅ (Hacker bypassed login!)

**✅ Secure Version (Safe)**

python

CopyEdit

# secure\_sql.py

username = input("Enter username: ")

query = "SELECT \* FROM users WHERE username = ?"

c.execute(query, (username,))

if c.fetchone():

print("Logged in")

else:

print("Login failed")

**✅ Output:**

pgsql

CopyEdit

Enter username: admin' OR '1'='1

Login failed ❌

**✅ 2. XSS Attack: Unsafe vs Safe**

**❌ Vulnerable HTML (Unsafe)**

html

CopyEdit

<!-- vulnerable\_xss.html -->

<html>

<body>

<h2>Comment:</h2>

<div id="comment"></div>

<script>

// Simulate user input (dangerous!)

var userInput = "<script>alert('Hacked!')</script>";

document.getElementById("comment").innerHTML = userInput;

</script>

</body>

</html>

**💥 Output:**

* Browser popup: Hacked! 😱

**✅ Secure HTML (Safe)**

html

CopyEdit

<!-- safe\_xss.html -->

<html>

<body>

<h2>Comment:</h2>

<div id="comment"></div>

<script>

var userInput = "<script>alert('Hacked!')</script>";

// Escape input to prevent script execution

var div = document.createElement("div");

div.innerText = userInput;

document.getElementById("comment").appendChild(div);

</script>

</body>

</html>

**✅ Output:**

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

* No alert pops up ✅

**✅ 3. Broken Authentication (Example only)**

**❌ Unsafe:**

python

CopyEdit

password = input("Enter password: ")

stored\_password = "admin123" # Plaintext (bad)

if password == stored\_password:

print("Access granted")

**✅ Safe Version:**

python

CopyEdit

import bcrypt

# Register

password = b"admin123"

hashed = bcrypt.hashpw(password, bcrypt.gensalt())

# Login

input\_pw = input("Enter password: ").encode()

if bcrypt.checkpw(input\_pw, hashed):

print("Access granted ✅")

else:

print("Access denied ❌")

**🧪 Output:**

Enter password: admin123

Access granted ✅