**SQL Injection Attack**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>

Access Wifi

</title>

</head>

<body>

<br>

<label for="code">Code:</label>

<input type="text" id="code"><br><br>

<button id="submit">Submit</button>

<h5 id="text"></h5>

<script type="text/javascript" src="submit.js"></script>

</body>

</html>

**Submit.js**

function makeHttpObject() {

try {

return new XMLHttpRequest();

} catch (error) {

}

try {

return new ActiveXObject("Msxml2.XMLHTTP");

} catch (error) {

}

try {

return new ActiveXObject("Microsoft.XMLHTTP");

} catch (error) {

}

throw new Error("Could not create HTTP request object.");

}

function welcome(){

alert("welcome")

}

const button = document.getElementById("submit");

//button.addEventListener("click",welcome);

button.addEventListener("click", e => {

const code = document.getElementById("code").value;

console.log(code)

const url = "http://localhost:5000/search?code=" + code;

let request = makeHttpObject();

console.log(url);

request.open("GET", url, true);

request.send(null);

request.onreadystatechange = function () {

if (request.readyState == 4)

var text = request.responseText;

document.getElementById("text").innerHTML = text;

};

});

**Api.py**

from flask import Flask, request, render\_template

from flask\_cors import CORS

import sqlite3

app = Flask(\_\_name\_\_)

CORS(app)

@app.route("/get")

def get():

conn = sqlite3.connect("data.db")

c = conn.cursor()

c.execute("select \* from data")

data = c.fetchall()

return "<br>".join([i[0] for i in data])

@app.route("/register")

def register():

code = request.args.get('code')

conn = sqlite3.connect("data.db")

c = conn.cursor()

try:

c.execute("INSERT INTO data VALUES (?)", (code,))

conn.commit()

return f"Successfully added {code}"

except sqlite3.Error as e:

return str(e)

@app.route("/search")

def search():

code = request.args.get('code')

conn = sqlite3.connect("data.db")

c = conn.cursor()

try:

statement = "select \* from data where data='" + code + "'"

c.execute(statement)

found = c.fetchall()

if found == []:

return f"Invalid Code<br>{statement}"

else:

return f"Wifi Connection Established<br>{statement}"

except sqlite3.Error as e:

return str(e) + f"<br>{statement}"

@app.route("/login")

def login():

return open("login.html").read()

@app.route("/")

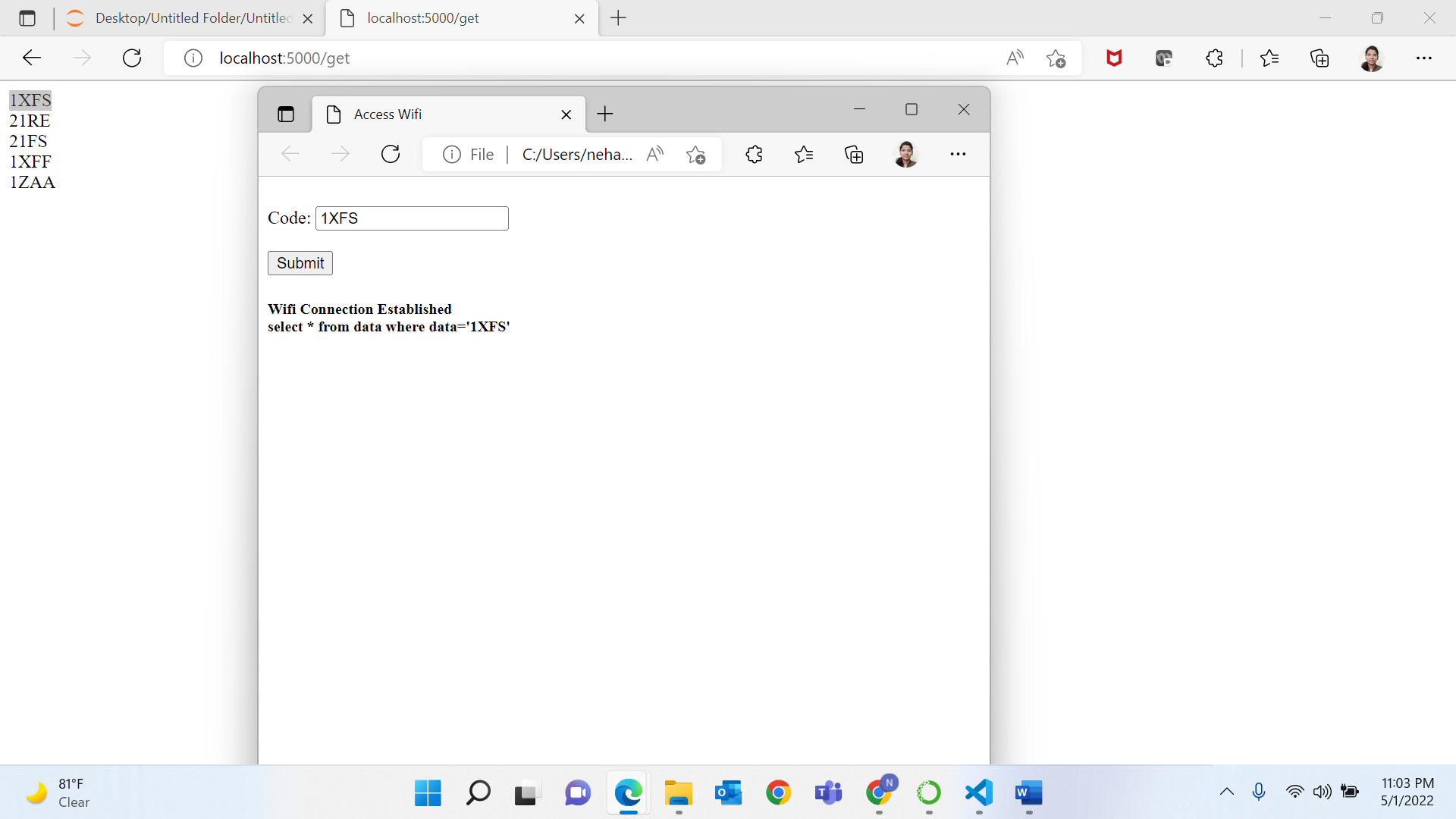
def main():

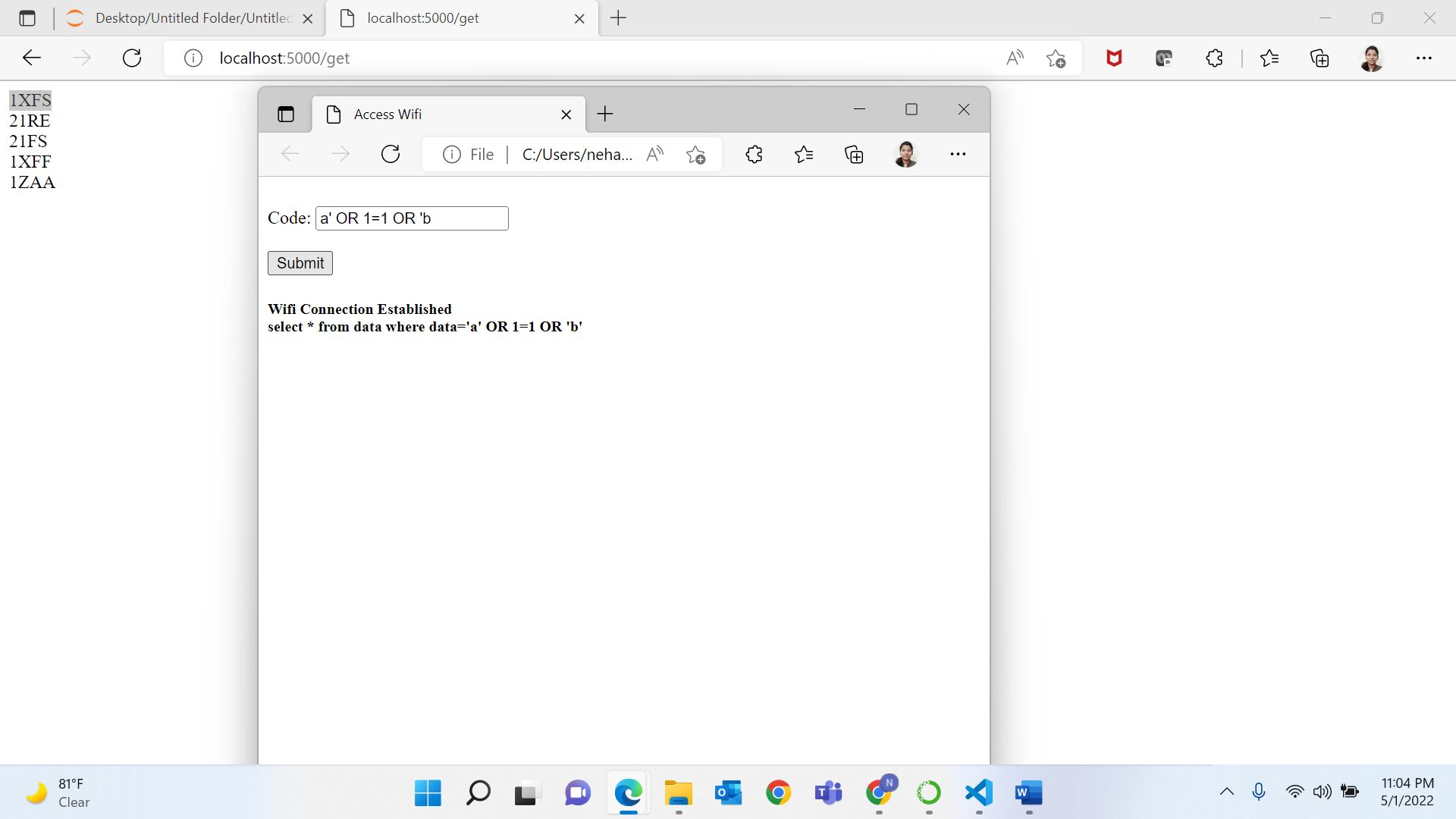
return open("403.html").read()

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0")

**Output**





**XSS Attack**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Cross-Site Scripting (XSS) Attacks</title>

<link rel="stylesheet" href="style.css" />

<script src="app.js" defer></script>

</head>

<body>

<section id="user-input">

<form>

<div class="form-control">

<label for="user-message">Your Message</label>

<textarea id="user-message" name="user-message"></textarea>

</div>

<div class="form-control">

<label for="message-image">Message Image</label>

<input type="text" id="message-image" name="message-image" />

</div>

<button type="submit">Send Message</button>

</form>

</section>

<section id="user-messages">

<ul></ul>

</section>

</body>

</html>

**App.js**

const userMessages = [];

const userMessageForm = document.querySelector('form');

const userMessagesList = document.querySelector('ul');

function renderMessages() {

let messageItems = '';

for (const message of userMessages) {

messageItems = `

${messageItems}

<li class="message-item">

<div class="message-image">

<img src="${message.image}" alt="${message.text}">

</div>

<p>${message.text}</p>

</li>

`;

}

userMessagesList.innerHTML = messageItems;

}

function formSubmitHandler(event) {

event.preventDefault();

const userMessageInput = event.target.querySelector('textarea');

const messageImageInput = event.target.querySelector('input');

const userMessage = userMessageInput.value;

const imageUrl = messageImageInput.value;

if (

!userMessage ||

!imageUrl ||

userMessage.trim().length === 0 ||

imageUrl.trim().length === 0

) {

alert('Please insert a valid message and image.');

return;

}

userMessages.push({

text: userMessage,

image: imageUrl,

});

userMessageInput.value = '';

messageImageInput.value = '';

renderMessages();

}

userMessageForm.addEventListener('submit', formSubmitHandler);

**Output:**

