**Q3: Social Network Friend Recommendation (Graph)**: Use a **graph** to represent connections between users in a social network. Implement a **BFS algorithm** to suggest friend recommendations based on mutual connections.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Social Network Friend Recommendation</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

background-color: #f0f0f0;

}

h1 {

color: #333;

}

.container {

max-width: 600px;

margin: 0 auto;

background: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.user-input {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

input, button {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #007BFF;

color: white;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.results {

margin-top: 20px;

}

.results p {

background: #e0f7fa;

padding: 10px;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="container">

<h1>Friend Recommendation</h1>

<div class="user-input">

<label for="username">Enter a username:</label>

<input type="text" id="username" placeholder="e.g., Alice">

<button id="recommendBtn">Get Recommendations</button>

</div>

<div class="results" id="results"></div>

</div>

<script>

// Social network graph (adjacency list representation)

const socialGraph = {

Alice: ["Bob", "Charlie"],

Bob: ["Alice", "David", "Eve"],

Charlie: ["Alice", "Eve"],

David: ["Bob"],

Eve: ["Bob", "Charlie", "Frank"],

Frank: ["Eve"]

};

// Breadth-First Search for friend recommendation

function recommendFriends(graph, user) {

if (!graph[user]) {

return [`User ${user} not found in the network.`];

}

const visited = new Set();

const queue = [user];

visited.add(user);

const recommendations = new Set();

while (queue.length > 0) {

const current = queue.shift();

for (const friend of graph[current]) {

if (!visited.has(friend)) {

visited.add(friend);

queue.push(friend);

// Suggest friends of friends

for (const friendsFriend of graph[friend]) {

if (friendsFriend !== user && !graph[user].includes(friendsFriend)) {

recommendations.add(friendsFriend);

}

}

}

}

}

return recommendations.size > 0

? Array.from(recommendations)

: ["No recommendations found."];

}

// Event listener for recommendation button

document.getElementById("recommendBtn").addEventListener("click", () => {

const username = document.getElementById("username").value.trim();

const resultsDiv = document.getElementById("results");

if (username === "") {

resultsDiv.innerHTML = `<p>Please enter a username.</p>`;

return;

}

const recommendations = recommendFriends(socialGraph, username);

resultsDiv.innerHTML = recommendations

.map(rec => `<p>${rec}</p>`)

.join("");

});

</script>

</body>

</html>