**Full Stack Web Development Tasks with Explanation**

**Task 1: How to access mic using JavaScript and convert speech to text?**

**Objective:** Capture user's voice through mic and convert it into text using JavaScript.

**Tools/Technologies Used:**

* JavaScript
* Web Speech API (SpeechRecognition)

**Code Flow Explanation:**

1. Use window.SpeechRecognition or webkitSpeechRecognition.
2. Start recognition using .start().
3. Capture result in onresult event.
4. Display transcribed text on the UI.

**Sample Code:**

const recognition = new (window.SpeechRecognition || window.webkitSpeechRecognition)();

recognition.lang = 'en-US';

recognition.onresult = function(event) {

document.getElementById('output').innerText = event.results[0][0].transcript;

};

recognition.start();

**Output/Results:**

* User speaks and the voice is shown as text on the webpage.

**Task 2: How to access camera using JavaScript and click photo?**

**Objective:** Use device camera and capture a photo.

**Tools/Technologies Used:**

* JavaScript
* navigator.mediaDevices.getUserMedia
* HTML5 Canvas

**Code Flow Explanation:**

1. Use getUserMedia to access video stream.
2. Display it on a video tag.
3. On button click, draw the current frame to a canvas and save it as image.

**Sample Code:**

navigator.mediaDevices.getUserMedia({ video: true })

.then(stream => {

document.getElementById('video').srcObject = stream;

});

function capturePhoto() {

const canvas = document.getElementById('canvas');

const ctx = canvas.getContext('2d');

ctx.drawImage(document.getElementById('video'), 0, 0, canvas.width, canvas.height);

}

**Output/Results:**

* Live video feed and photo capture with canvas rendering.

**Task 3: How to do live stream of camera using JavaScript?**

**Objective:** Stream camera input live on a webpage.

**Tools/Technologies Used:**

* JavaScript
* HTML5

**Code Flow Explanation:**

1. Use getUserMedia to fetch video.
2. Assign the stream to a <video> element.
3. The user sees live video on screen.

**Sample Code:**

navigator.mediaDevices.getUserMedia({ video: true })

.then(stream => {

const video = document.querySelector("video");

video.srcObject = stream;

video.play();

});

**Output/Results:**

* A live camera feed appears on the page.

**Task 4: Record a video using JavaScript and post that recording automatically to Instagram using JavaScript**

**Objective:** Record video and share it programmatically (though direct Instagram API doesn't allow this yet without user interaction).

**Tools/Technologies Used:**

* MediaRecorder API
* Instagram Graph API (requires server & access token)

**Code Flow Explanation:**

1. Use MediaRecorder to capture video.
2. Convert blob to file.
3. Send video to backend which uploads to Instagram using access token.

**Limitations:** Instagram doesn't allow direct posting via frontend JavaScript due to CORS, access token, and user permissions.

**Output/Results:**

* Recorded video is saved and backend can be used to post.

**Task 5: Search any name in Google, take all the links related to this name and create a search engine**

**Objective:** Create a basic search engine-like UI by crawling links.

**Tools/Technologies Used:**

* Node.js backend
* Puppeteer or Cheerio for scraping
* JavaScript frontend

**Code Flow Explanation:**

1. User enters a name.
2. Backend scrapes Google search result page (with Puppeteer).
3. Extracts and sends top 10 URLs to frontend.
4. Displayed as search results.

**Output/Results:**

* Custom search engine with Google-like result links.

**Task 6: Practice the Google App Script code and try to implement something new**

**Objective:** Automate workflows using Google Apps Script.

**Tools/Technologies Used:**

* Google Apps Script
* Google Sheets, Docs, Gmail APIs

**Sample Ideas:**

* Auto email sender from Sheets
* Form submission to Sheets

**Code Example:**

function sendEmails() {

var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();

var data = sheet.getDataRange().getValues();

for (let i = 1; i < data.length; i++) {

MailApp.sendEmail(data[i][1], 'Subject', data[i][2]);

}

}

**Output/Results:**

* Emails sent to addresses listed in sheet.

**Task 7: Write a JavaScript code to drag and drop a division using mouse**

**Objective:** Make a div draggable by mouse.

**Code Flow Explanation:**

1. Add mouse event listeners.
2. Update position of div on mousemove.

**Sample Code:**

const box = document.getElementById("box");

box.onmousedown = function(event) {

function move(e) {

box.style.left = e.pageX + 'px';

box.style.top = e.pageY + 'px';

}

document.addEventListener('mousemove', move);

document.onmouseup = () => document.removeEventListener('mousemove', move);

};

**Output/Results:**

* User can drag the div around the screen.

**Task 8: Connect JavaScript with the ChatGPT and generate a response**

**Objective:** Send prompt to OpenAI API and receive a response.

**Tools/Technologies Used:**

* JavaScript
* OpenAI API

**Code Flow Explanation:**

1. Create input box.
2. Send fetch request to OpenAI API with input prompt.
3. Display response on page.

**Sample Code:**

fetch('https://api.openai.com/v1/chat/completions', {

method: 'POST',

headers: {

'Authorization': `Bearer YOUR\_API\_KEY`,

'Content-Type': 'application/json'

},

body: JSON.stringify({

model: "gpt-3.5-turbo",

messages: [{ role: "user", content: "Hello" }]

})

})

.then(res => res.json())

.then(data => console.log(data.choices[0].message.content));

**Output/Results:**

* ChatGPT response printed to browser.

**Task 9: Write a JavaScript code and use your mic to give input prompt to ChatGPT**

**Objective:** Combine speech recognition with ChatGPT API.

**Code Flow Explanation:**

1. Capture voice input using SpeechRecognition.
2. Send text to OpenAI API.
3. Display ChatGPT response.

**Output/Results:**

* Voice input triggers ChatGPT response generation.

**Task 10: Write a JavaScript code to generate an image using ChatGPT prompt**

**Objective:** Generate an image using DALL-E API via OpenAI.

**Tools/Technologies Used:**

* JavaScript
* OpenAI API (DALL·E model)

**Sample Code:**

fetch('https://api.openai.com/v1/images/generations', {

method: 'POST',

headers: {

'Authorization': `Bearer YOUR\_API\_KEY`,

'Content-Type': 'application/json'

},

body: JSON.stringify({ prompt: "A cat playing guitar", n: 1, size: "512x512" })

})

.then(res => res.json())

.then(data => {

document.getElementById('image').src = data.data[0].url;

});

**Output/Results:**

* Image is generated from prompt and displayed on page.