I’d like to add a feature that integrates the Google Web Fonts API into our Node.js app. The goal is to let users customize their “Share Pages” by selecting a font for the headline and description text. Here’s the intended functionality and structure:

### **Feature Overview**

* **Purpose:**Allow users to browse or search the Google Web Fonts library and select a font for their custom share page. Once selected, the app will generate the necessary code (a <link> tag and CSS) to load and apply the font to the share page.
* **Workflow:**
  1. **Fetch Font Data:**
     + Create a backend API endpoint that retrieves the list of fonts from the Google Web Fonts API.
  2. **User Interface:**
     + Build a client-side UI that displays the fonts. This UI should include a search input to filter fonts by name.
     + Display each font (with a preview using its own font family) along with a “Select” button.
  3. **Font Selection:**
     + When a user clicks on a font, store the selection (for example, in a session or database) and generate the code snippet needed to include the font in the share page.
     + The generated snippet should include a <link> tag (e.g., <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">) and CSS instructions (e.g., font-family: 'Roboto', sans-serif;).

### **Backend Example (Node.js/Express)**

Create an endpoint (e.g., /api/fonts) that fetches data from the Google Web Fonts API. Make sure you have your API key stored in an environment variable (e.g., GOOGLE\_FONTS\_API\_KEY).

// fonts.js (Express router)

const express = require('express');

const fetch = require('node-fetch'); // or use built-in fetch if available

const router = express.Router();

const API\_KEY = process.env.GOOGLE\_FONTS\_API\_KEY;

router.get('/api/fonts', async (req, res) => {

try {

const response = await fetch(`https://www.googleapis.com/webfonts/v1/webfonts?key=${API\_KEY}`);

const data = await response.json();

// Optionally, you could sort or cache the fonts list here

res.json(data.items);

} catch (error) {

console.error('Error fetching fonts:', error);

res.status(500).json({ error: 'Failed to fetch fonts' });

}

});

module.exports = router;

Then include this router in your main server file.

### **Frontend Example (HTML/JavaScript)**

Below is a sample HTML file with inline JavaScript. This UI allows users to search and browse fonts, and when they select one, it prints the code snippet (you can adapt this to store the selection or update the share page directly):

<!DOCTYPE html>

<html>

<head>

<title>Select a Google Font</title>

<style>

body {

font-family: sans-serif;

margin: 20px;

}

#fontList li {

cursor: pointer;

padding: 5px;

margin: 2px 0;

border: 1px solid #ddd;

}

#fontList li:hover {

background-color: #f0f0f0;

}

</style>

</head>

<body>

<h1>Choose a Font for Your Share Page</h1>

<input type="text" id="fontSearch" placeholder="Search fonts..." style="width: 100%; padding: 8px;">

<ul id="fontList"></ul>

<script>

async function fetchFonts() {

const response = await fetch('/api/fonts');

const fonts = await response.json();

return fonts;

}

function displayFonts(fonts) {

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

fontList.innerHTML = '';

fonts.forEach(font => {

const li = document.createElement('li');

li.textContent = font.family;

li.style.fontFamily = font.family; // preview font style

li.addEventListener('click', () => selectFont(font));

fontList.appendChild(li);

});

}

function selectFont(font) {

// For demonstration, output the code snippet in the console

// In production, store this selection and update the share page accordingly

const fontFamily = font.family.replace(/ /g, '+'); // format font family for URL

const linkTag = `<link href="https://fonts.googleapis.com/css2?family=${fontFamily}:wght@400;700&display=swap" rel="stylesheet">`;

const cssSnippet = `font-family: '${font.family}', sans-serif;`;

console.log('Include the following in your Share Page:');

console.log(linkTag);

console.log(cssSnippet);

// Optionally, display the snippet to the user or store the choice

alert(`Selected font: ${font.family}\n\nInclude this in your HTML head:\n${linkTag}\n\nAnd use this CSS:\n${cssSnippet}`);

}

document.getElementById('fontSearch').addEventListener('input', async function() {

const searchTerm = this.value.toLowerCase();

const fonts = await fetchFonts();

const filteredFonts = fonts.filter(font => font.family.toLowerCase().includes(searchTerm));

displayFonts(filteredFonts);

});

// Initial load: fetch and display all fonts

fetchFonts().then(displayFonts);

</script>

</body>

</html>

### **Instructions for the AI Agent**

1. **Backend Tasks:**
   * Create an Express route /api/fonts that fetches the list of Google fonts using the Google Web Fonts API and returns the JSON data.
   * Ensure that the API key is stored securely in an environment variable.
   * (Optionally) Implement caching for the fonts list to reduce API calls.
2. **Frontend Tasks:**
   * Build a simple UI that displays a search box and a list of fonts.
   * Allow users to filter fonts by name using the search input.
   * When a user selects a font, generate a code snippet that includes:
     + A <link> tag to load the font from Google.
     + A CSS rule for applying the font (e.g., font-family: 'SelectedFont', sans-serif;).
   * Display the generated code snippet (e.g., via an alert or by updating the DOM) and/or store the selected font for later use on the user’s share page.
3. **Integration with Share Pages:**
   * When a user finalizes their selection, save the font choice (via session, database, or state management).
   * Update the share page generation logic to include the <link> tag in the HTML head and apply the font to the headline and description.

Please proceed with these steps. Let me know if you need further details or if there are any questions regarding this feature.