Welcome! You've built a fully functional web application that uses a powerful AI to analyze and improve text. This guide will walk you through exactly what this project is, how it works, and how each part contributes to the final product.

**What is the "Prompt Analyzer"?**

At its core, the **Prompt Analyzer is a tool that helps you write better instructions for an AI**. Think of it like a spell-checker and grammar coach, but instead of correcting your English, it coaches you on how to ask an AI for things more effectively.

You give it a prompt (an instruction for an AI), and it gives you a score and feedback on how clear, specific, and well-structured your prompt is.

**Part 1: The Big Picture - How It Works**

Our project isn't one single thing; it's a system with two main parts that talk to each other: a **Front-end** and a **Backend**.

* **The Front-end (The Storefront)**: This is the part you see and interact with in your web browser. It's the visual part of the application—the text box, the button, and the results display. Its only job is to look good and communicate with the backend.
* **The Backend (The Engine Room)**: This is a hidden server that runs on your computer. It does all the heavy lifting and contains the project's "brain." You don't see it, but it's where the most important work happens.

**The Flow of Information**

When you click "Analyze," a 6-step journey begins:

1. **You Click Analyze**: The **Front-end** (script.js) grabs the prompt you wrote.
2. **A Call is Made**: The Front-end sends the prompt over the internet to our **Backend** server (server.js).
3. **The Secret Handshake**: The Backend receives the prompt. It then makes a secure, private call to the **Google AI (Gemini)**, using a secret **API Key** to prove it has permission.
4. **AI Analysis**: Google's AI analyzes the prompt based on a special set of instructions we gave it.
5. **Results are Returned**: Google AI sends the structured results (scores, feedback, etc.) back to our Backend.
6. **Displaying the Results**: Our Backend relays these results to the Front-end, which then displays them neatly on your screen.

**The Security Guard: Why the Backend is Crucial**

The **API Key** is a secret password that gives us access to Google's powerful AI. If anyone else got this key, they could use it and you'd get the bill!

* Code on the **Front-end** is **public**; anyone can see it.
* Code on the **Backend** is **private** and runs only on your machine.

By keeping the API key on the backend, we ensure it stays secret and safe. The backend acts as a secure guard, protecting our valuable key while still letting us talk to the AI.

**Part 2: The Project Files Explained**

Your project is organized into two folders, frontend and backend. Here’s what each file does.

**The frontend Folder (The Storefront)**

* **index.html**: The **skeleton** of the webpage. It's a text file that defines the structure, telling the browser what to put on the page (a title, a text box, a button, a place for results).
* **style.css**: The **interior designer**. This file contains all the rules for colors, fonts, spacing, and layout. It makes the raw HTML skeleton look polished and professional.
* **script.js**: The **store manager**. This file contains the interactive logic. It listens for the "Analyze" button click, sends the prompt to the backend, and displays the results when they arrive.

**The backend Folder (The Engine Room)**

* **server.js**: The **engine** itself. This is the most important file in the backend. It's a program that's always running, listening for requests from the front-end. It contains the logic to talk to the Google AI and process the results.
* **.env**: The **digital safe**. This is a simple text file that holds our secret GOOGLE\_API\_KEY. It's kept private and is never shared.
* **.gitignore**: A **"do not share" list**. This file tells source control tools like Git to ignore certain files. We've told it to ignore .env so we never accidentally share our secrets.
* **package.json**: The **list of ingredients** for our backend. It lists all the external tools (called "packages") that server.js needs to function, like express (the server framework) and @google/generative-ai (the tool to talk to Google).
* **package-lock.json**: A **detailed recipe**. It locks in the exact versions of the ingredients from package.json to ensure the project works consistently everywhere. This file is managed automatically by npm.

**Part 3: The Code Explained (In Simple Terms)**

You don't need to understand every line, but let's look at the most important parts.

**frontend/script.js - Making the Call**

JavaScript

function callBackendAPI(prompt) {

// We tell the browser where our backend is running.

const backendUrl = 'http://localhost:3000/analyze';

// 'fetch' is the browser's way of making a "phone call" to a server.

return fetch(backendUrl, {

method: 'POST', // We are sending (POSTing) data.

headers: { 'Content-Type': 'application/json' }, // We're sending it in JSON format.

body: JSON.stringify({ prompt: prompt }) // We package our prompt text into a neat JSON bundle.

})

// ...

}

**backend/server.js - The Magic Instruction**

This is the most clever part of our project. We don't just ask the AI to analyze the prompt; we give it a very strict set of instructions on *how* to do it.

JavaScript

const getAnalysisPrompt = (promptToAnalyze) => {

return `You are The Perfect Prompt Analyst... You must respond with only a valid JSON object...

The JSON object must have this exact structure:

{

"overallPercentage": <a number 0-100>,

"rating": "<a string like 'Good' or 'Excellent'>",

"categoryScores": { ... },

"feedback": { ... }

}

Here is the user's prompt to analyze:

'${promptToAnalyze}'`;

};

This is called a **meta-prompt**. We are instructing the AI on its personality ("You are The Perfect Prompt Analyst") and, most importantly, forcing it to reply in a clean, structured **JSON** format.

**JSON (JavaScript Object Notation)** is just a simple, organized way to structure data with labels (e.g., "rating": "Excellent"). By forcing the AI to reply in this format, we make it easy for our script.js to read and display the results.

**Part 4: How to Set Up and Run the Project**

Here is a step-by-step guide to run this project on any computer.

**Prerequisites**

1. **Node.js**: A program that lets you run JavaScript outside of a web browser. Installing it also gives you npm (Node Package Manager), a tool for installing the "ingredients" listed in package.json. Download it from [nodejs.org](https://nodejs.org/).
2. **A Code Editor**: A program for editing code files. **Visual Studio Code** is highly recommended and free.

**Setup Instructions**

1. **Get the Code**: Place the aiv2-prompt-analyzer folder somewhere on your computer. It should contain the frontend and backend subfolders.
2. **Get Your API Key**:
   * Go to **Google AI Studio**.
   * Click "Get API key" and create a new key.
   * Copy the key.
3. **Set Up the Backend**:
   * Open a terminal (like PowerShell or Command Prompt).
   * Navigate into the backend folder: cd path/to/your/project/aiv2-prompt-analyzer/backend
   * In that folder, create a new file named .env.
   * Open .env and put your API key in it like this: GOOGLE\_API\_KEY=YOUR\_KEY\_HERE
   * Back in the terminal, run the command npm install. This will read package.json and download all the necessary tools into a new node\_modules folder.
4. **Run the Project**: You will need two terminals open.
   * **Terminal 1 (Start the Backend)**:
     + Make sure you are in the backend folder.
     + Run the command: node server.js
     + You should see the message: ✅ Server is running on http://localhost:3000
     + Leave this terminal running.
   * **Terminal 2 (Start the Front-end)**:
     + If you are using Visual Studio Code, install the **"Live Server"** extension.
     + Right-click on the frontend/index.html file and choose "Open with Live Server".
     + This will automatically open the project in your web browser, and you're ready to go!