STEP 1: Project Setup

- Create a folder: `ai-website-helper`

STEP 2: Create the Required Files

- `server.js` — Node.js backend
- `.env` — to safely store your OpenRouter API key
- `frontend.html` — browser frontend interface

STEP 3: Get an API Key

- Sign up and get an API key from https://openrouter.ai

Save it inside `.env` like:

bash
CopyEdit
```
OPENROUTER_API_KEY=your_key_here
```

-

✅ STEP 4: Write Backend Code (server.js)

Make sure your `server.js`:

- Uses `dotenv` to load `.env`
- Has CORS enabled
- Exposes an endpoint like `/generate` that:
  - Receives user input
  - Sends it to OpenRouter API using `axios`

- Writes the response to `generated-site/index.html`
- Executes `firebase deploy` using `child_process` (optional)

Basic Structure:

```
require("dotenv").config();
const express = require("express");
const axios = require("axios");
const fs = require("fs");
const cors = require("cors");
const { exec } = require("child_process");

const app = express();
app.use(cors());
app.use(express.json());

app.post("/generate", async (req, res) => {
  const { prompt } = req.body;

  try {
    const response = await axios.post(
      "https://openrouter.ai/api/v1/chat/completions",
      {
        model: "gpt-3.5-turbo",
        messages: [
          { role: "system", content: "You generate complete HTML/CSS 
websites" },
          { role: "user", content: prompt },
        ],
      },
      {
        headers: {
          Authorization: `Bearer ${process.env.OPENROUTER_API_KEY}`,
          "Content-Type": "application/json",
        },
      }
    );

    const htmlCode = response.data.choices[0].message.content;
    fs.writeFileSync("./generated-site/index.html", htmlCode);

    // Optional: Deploy to Firebase automatically
    exec("firebase deploy", { cwd: "./generated-site" }, (error,
stdout, stderr) => {
      if (error) {
        return res.status(500).send("Deploy failed.");
      }
      res.send("Website generated and deployed!");
    });

  } catch (err) {
    console.error(err);
    res.status(500).send("Something went wrong.");
  }
});

app.listen(3000, () => console.log("Server started on
http://localhost:3000"));
```

STEP 5: Firebase Hosting Setup

Inside your `generated-site` folder (for deployment):

```
firebase login
firebase init hosting
```

Choose:

- “Use an existing project” or “Create a new project”
- Public directory: `.` (if deploying root `index.html`)

- Configure as SPA: No
- Overwrite `index.html`: No

Add a `firebase.json` inside `generated-site`:

json
CopyEdit
```
{
  "hosting": {
    "public": ".",
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**"]
  }
}
```

STEP 6: Running & Testing

- Run backend:

```
node server.js
```

- Open `frontend.html` using Live Server or browser
- When you click **Generate**, it will:
  - Send input to `server.js`
  - Generate site with OpenRouter
  - Save to `generated-site/index.html`
  - Deploy to Firebase
  - Return live URL (optional: parse `stdout` to extract it)

Final Approach

voice/text prompt-driven website generator, where the **input is a theme or title** (like "Photography Portfolio", "College Resume", "Tech Blog"), and based on that:

- A dynamic, responsive website is generated (HTML/CSS/JS).
- It's automatically hosted on Firebase.
- You get a **live Firebase URL** in return.

Let's break this down into a **refined step-by-step plan**, followed by the **code upgrades** you need to handle **voice/text input + theme-to-full-site generation + Firebase deployment + return URL**.

Updated End-to-End Flow

Step-by-Step Workflow

Step	Description
1	User opens `frontend.html`, speaks or types prompt
2	Prompt is sent to backend (`server.js`)
3	Backend sends prompt to **OpenRouter** (ChatGPT-like API)
4	OpenRouter generates a complete HTML page
5	Backend saves this as `index.html` in a new project folder
6	Firebase deploys this folder under a unique site (ex: `portfolio-1234.web.app`)
7	Backend returns the **live Firebase URL** to the frontend
8	Frontend displays link to user

Frontend (Voice + Text UI) – `frontend.html`

```
<!DOCTYPE html>
<html>
<head>
  <title>AI Website Generator</title>
</head>
<body>
  <h2>Describe your website (Theme or Title)</h2>
  <input type="text" id="promptInput" placeholder="e.g. A personal
blog for a student">
  <button onclick="generateWebsite()">Generate</button>
  <button onclick="startVoiceInput()">🎙️ Voice Input</button>
  <p id="status"></p>
  <a id="liveLink" href="#" target="_blank"></a>

  <script>
    function generateWebsite() {
      const prompt = document.getElementById("promptInput").value;
      document.getElementById("status").innerText = "Generating and
deploying...";
      fetch("http://localhost:3000/generate", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ prompt }),
      })
      .then(res => res.json())
      .then(data => {
        document.getElementById("status").innerText = "Website is
live!";
        const link = document.getElementById("liveLink");
        link.href = data.url;
        link.innerText = data.url;
      })
      .catch(() => document.getElementById("status").innerText =
"Error occurred.");
    }

    function startVoiceInput() {
      const recognition = new (window.SpeechRecognition ||
window.webkitSpeechRecognition)();
      recognition.lang = 'en-US';
      recognition.start();
      recognition.onresult = (event) => {
        document.getElementById("promptInput").value =
event.results[0][0].transcript;
      };
    }
  </script>
</body>
</html>
```

Backend (Node.js + OpenRouter + Firebase CLI) – `server.js`

js
CopyEdit

```
require("dotenv").config();
const express = require("express");
const axios = require("axios");
const fs = require("fs");
const path = require("path");
const cors = require("cors");
const { exec } = require("child_process");
const { v4: uuidv4 } = require("uuid");

const app = express();
app.use(cors());
app.use(express.json());

app.post("/generate", async (req, res) => {
  const prompt = req.body.prompt;
  const siteId = `site-${uuidv4().split("-")[0]}`;
  const projectDir = path.join(__dirname, siteId);

  try {
    const response = await axios.post(
      "https://openrouter.ai/api/v1/chat/completions",
      {
        model: "gpt-3.5-turbo",
        messages: [
          { role: "system", content: "You are a web designer that
builds complete HTML websites using a given theme/title. Output must
be a complete index.html with styling and responsive layout." },
          { role: "user", content: `Build a responsive HTML/CSS
website for: ${prompt}` },
        ],
      },
      {
        headers: {
          Authorization: `Bearer ${process.env.OPENROUTER_API_KEY}`,
          "Content-Type": "application/json",
        },
      }
    );

    const html = response.data.choices[0].message.content;

    fs.mkdirSync(projectDir);
    fs.writeFileSync(`${projectDir}/index.html`, html);
    fs.writeFileSync(`${projectDir}/firebase.json`, JSON.stringify({
      hosting: { public: ".", ignore: ["firebase.json", "**/.*",
"**/node_modules/**"] }
    }));

    exec(`firebase deploy --project ${process.env.FIREBASE_PROJECT_ID}
--only hosting --config firebase.json --cwd ${projectDir}`, (err,
stdout) => {
      if (err) {
        console.error(err);
        return res.status(500).send({ error: "Deployment failed" });
      }

      const match = stdout.match(/https:\/\/[^\s]+\.web\.app/);
      const liveUrl = match ? match[0] : null;
      res.send({ url: liveUrl });
    });

  } catch (err) {
    console.error(err);
    res.status(500).send({ error: "Generation or deployment failed"
});
  }
});

app.listen(3000, () => console.log("Server running at
http://localhost:3000"));
```

`.env` File (Example)

```
OPENROUTER_API_KEY=your_openrouter_key
FIREBASE_PROJECT_ID=your_firebase_project_id
```

Make sure your Firebase project ID is linked to hosting.

One-Time Firebase Setup (CLI)

```
npm install -g firebase-tools
firebase login
firebase init hosting  # Do this in a dummy folder and copy
firebase.json format
```
