Text-to-Website Generation Using CampEdUI - Implementation Guide

1. Overview
- Build a pipeline to convert structured JSON input into React JSX using CampEdUI components via GPT-4
- Focus on generating clean, semantic, and maintainable code without requiring local GPU.

2. Tasks to Cover
  a. Model Training (via API):
    • Use prompt engineering with GPT-4 to translate JSON → JSX (HTML/CSS).
    • Ensure semantic structure, accessibility, and responsiveness.
  b. Website Generation:
    • Generate webpage layouts handling headers, paragraphs, images, buttons.
    • Output valid CampEdUI-based code.
  c. Component Integration:
    • Map JSON component types to CampEdUI equivalents (Button, Card, Typography, etc.).
    • Follow CampEdUI styling conventions (variants, props, imported components).

3. Assessment Criteria
  1. Model Accuracy:
    • Correct structure and usable webpage output.
    • Well-formed HTML/CSS/JSX code.
  2. Code Quality:
    • Clean, readable, semantic code.
    • Adherence to accessibility and responsive best practices.
  3. CampEdUI Integration:
    • All UI elements sourced from CampEdUI.
    • Compliance with design system’s conventions.
  4. Presentation & Documentation:
    • Clear explanation of approach, model selection, training, prompting.
    • Sample inputs, outputs, and reasoning documented.

4. Expected Deliverables
  1. Model & Training Artifacts:
    • Prompt templates and instructions to reproduce results with GPT-4.
  2. Generation Script:
    • A Python script (generate.py) that accepts JSON and outputs CampEdUI JSX.
  3. Documentation & Demo:
    • A short write-up describing approach, prompt design, architecture.
    • Example demonstrating JSON → generated webpage.

5. Implementation Instructions
  1. Prerequisites:
    • Python 3.8+, openai, python-dotenv libraries.
    • Node.js + npm for React demo, camped-ui npm package.
    • OpenAI API key (set as environment variable).

  2. JSON Schema (IR):
    • Define a consistent structure:
    {
      "type": "Page",
      "components": [

```
      { "type": "HeroSection", "props": { "title": "...", "subtitle": "...", "cta": { "label": "...", "variant": "..." } } },
      { "type": "ImageBlock", "props": { "src": "...", "alt": "...", "width": 800, "height": 400 } },
      { "type": "CardGrid", "props": { "cards": [ { "title": "...", "description": "...", "icon": "IconName" }, ... ] } }
    ]
  }
```

3. Prompt Template:
   • System instruction: Outline requirements (CampEdUI imports, valid JSX, no extra prose).
   • Include 2-3 few-shot examples mapping JSON → desired JSX output.
   • Append new input JSON as final user message.

4. generate.py Script Outline:
   • Load JSON file from CLI argument.
   • Construct chat messages: system instruction, few-shot pairs, new JSON.
   • Call openai.ChatCompletion.create(model="gpt-4", ...).
   • Extract and save returned content as App.generated.jsx.

5. React Demo Setup:
   • Scaffold new project: npx create-react-app demo-site && cd demo-site && npm install camped-ui.
   • Copy App.generated.jsx into src/, create a wrapper App.js that imports it.
   • npm start to preview the generated page.

6. Code Structure Example
- Directory:
  text2website/
  ■■■ data/
  ■■■ scripts/
  ■ ■■■ generate.py
  ■■■ samples/
  ■ ■■■ sample_page.json
  ■ ■■■ sample_output.jsx
  ■■■ README.md (overview, instructions)
  ■■■ demo-site/ (React project with App.generated.jsx)

7. Visual Output Expectations
- Generated JSX should import:
  import { Button, Card, CardGrid, HeroSection, Image, Typography } from "camped-ui";
- Example JSX snippet:

```
export default function App() {
  return (
    <div>
      <HeroSection>
        <Typography variant="h1">Welcome</Typography>
        <Typography variant="subtitle1">This is a demo</Typography>
        <Button variant="primary">Get Started</Button>
      </HeroSection>
      <CardGrid>
        <Card>
          <Typography variant="h2">Feature A</Typography>
          <Typography variant="body1">Detail</Typography>
          <RocketIcon />
```

```
          </Card>
          ...
        </CardGrid>
      </div>
    );
  }
```

8. Tips & Next Steps
  - Refine prompts if component names/props mismatch CampEdUI.
  - Validate JSX by running lint/build in React demo.
  - Expand schema for multi-page sites or routing.
  - Document any limitations (e.g., nested layouts, custom CSS fallback).

This guide outlines the full workflow to satisfy all assessment requirements without a local GPU.