Assemble's
Fullstack Architect Development Test

Landing Page Implementation

Overview

This test evaluates your ability to build a modern, responsive landing page using Next.js and headless WordPress as a CMS. You will implement a design from Figma using data from a provided API endpoint.

Time Allocation

- **Expected completion time**: 4-6 hours
- **Submission deadline**: 3 business days from receipt
- If you need additional time, please complete what you can within the estimated timeframe and provide a detailed explanation of any challenges encountered.

Resources Provided

1. API Endpoint

http://webdevtest.assemblestaging.com/wp-json/wp/v2/pages?slug=home&_fields=id,acf&acf_format=standard

This endpoint provides all necessary content including:

- Header data (company name, navigation menu)
- Hero images (desktop and mobile versions)
- Content sections with titles, descriptions, and images
- Footer information

2. Figma Design

https://www.figma.com/site/8tX7DUnFmOfgaGzyHCGGF6/Web-Dev-Test---Assemble?node-id=0-1&t=uWMrbBdadoobBRGf-1

Technical Requirements

Core Stack

- **Framework**: Next.js (latest stable version)
- **Styling**: Tailwind CSS
- **Deployment**: Netlify or Vercel

Implementation Requirements

1. Component Architecture

- Create **modular components** for header and footer
- Components should be reusable and properly organized in the project structure
- Follow React best practices for component composition

2. Dynamic Content

- **No hardcoded text or images** - all content must come from the API response
- Properly handle the nested ACF (Advanced Custom Fields) data structure
- Implement error handling for API calls

3. Navigation

- All links must use Next.js `Link` component
- Use the exact paths/URLs provided in the API response
- Links should navigate to a 404 page (Next.js default is acceptable)

4. Image Implementation

- Use Next.js `Image` component for all images
- **Required attributes**:
  - Explicit `width` and `height` declarations
  - Meaningful `alt` tags (use alt text from API or generate descriptive text)
- Implement responsive images (desktop/mobile) where provided

5. Styling Guidelines

- Use Tailwind CSS for all styling
- Ensure responsive design matches Figma mockups
- Pay attention to spacing, typography, and color accuracy

Bonus Points

Advanced Styling (Recommended)

- Implement SCSS modules to organize styles in separate files
- Use Tailwind's `@apply` directive to create reusable utility classes
  (v3.x might be required, v4.x does not support @apply)

Accessibility Features

- Implement semantic HTML for screen reader compatibility
- Ensure proper heading hierarchy (h1, h2, h3, etc.)
- Test and verify keyboard/tab navigation works correctly

Deliverables

1. **Public GitHub Repository**
    - Clean, well-commented code
2. **Deployed Website**
    - Live URL (Netlify or Vercel)

Evaluation Criteria

1. **Code Quality** (30%)
    - Clean code according to instructions
    - Proper use of Next.js features
    - Component reusability
2. **Pixel-Perfect Implementation** (25%)
    - Accuracy to Figma design
    - Responsive behavior
3. **Technical Implementation** (25%)
    - Proper API integration
4. **Best Practices** (20%)
    - Accessibility compliance
    - Code organization

Submission Instructions

1. Email the following to **dev@assemblestudio.com**, **recruiter@assemblestudio.com**:
    - GitHub repository link
    - Deployed website URL

Additional Notes

- Focus on demonstrating your technical skills and attention to detail
- If you encounter any blockers with the API or Figma file, document them and implement a reasonable workaround
- Quality over quantity - it's better to have a polished partial implementation than a rushed complete one

---

Good luck! We look forward to reviewing your implementation.