# Senior Theme Developer Take-Home Task

Implement a **Product Detail Page (PDP)** from a provided Figma design using **Jinja** templates. Your solution will be assessed on code quality, design fidelity, accessibility, performance, and maintainability.

---

## 1) Objectives

- Translate a Figma PDP design into semantically correct, responsive HTML/CSS powered by **Jinja** templates. Should match the figma design to a near-perfect degree of accuracy.
- Demonstrate **front-end engineering best practices**: structure, naming, performance, responsiveness, accessibility, and componentization.
- Show that you can make pragmatic decisions when the design is ambiguous.

---

## 2) What We Provide

- **Figma design** Link to the PDP design (desktop + mobile frames, spec, and exportable assets). https://www.figma.com/design/LZoNbBKIQ7dIrsIEdkHoUV/Untitled?node-id=0-1&p=f&t=k90ksB1Mig8nbTgp-0to
- **Starter repo (zip file)** Minimal Jinja set-up with a simple dev server. Instructions to run can be found below.
- **Sample data**: JSON sample for one product (you can find in ./data/product.json)

> If you prefer not to use the starter repo, you may set up your own. Please keep the tech choices lightweight and document any tooling you add.

---

## 3) What You Deliver

- A working **Jinja-rendered PDP** that matches the design to a near-perfect degree of accuracy.
- All **templates**, **partials**, **styles**, **assets**, and **scripts** in a single repository.
- A **README.md** explaining:
  - How to run the project locally
  - Your assumptions and notable decisions
  - Any deviations from the design and why
  - Anything you would add with more time

> Please do **not** include compiled/minified artifacts in version control unless necessary. Provide instructions on how to build them instead.

---

## 4) Technical Requirements

### 4.1 Templating & Structure

- Use **Jinja** with a clear layout hierarchy:
  - `layout.jinja` (global layout)
  - `header.jinja` (global header)
  - `footer.jinja` (global footer)
  - `product.jinja` (page template)
  - `sections/` (e.g., gallery, price, variant selector, add-to-cart, reviews summary, badges)
- Leverage **Jinja features**: template inheritance, blocks, macros/`include`, filters, and control flow for conditional UI.

```
/src/
    /templates/
        product.jinja
    /sections/
        gallery.jinja
        ... more
/static/
    /css/
    /js/
    /images/
/data/
  product.json
```

Feel free to adjust the structure or add new folders where you deem necessary.

### 4.2 Styling

- You may use **vanilla CSS** or a utility framework (e.g., Tailwind). If using a framework, configure it thoughtfully.
- Organize styles by component.
- Provide **responsive** styles at least for the breakpoints shown in Figma (desktop & mobile). Tablet is a plus.

### 4.3 Performance

- Avoid layout shift (reserve space for media, stable typography).
- Keep CSS/JS minimal and defer non-critical assets.
- Lighthouse/Web Vitals awareness (you don't need to submit scores, but code like you care).

### 4.4 Engineering Quality

- Clear naming, small components, and DRY templates.
- Handle edge cases: long product names, missing secondary images, out-of-stock variants, sale price vs. regular price, etc.

---

## 5) Functional Scope (Minimum)

Implement the following sections/components to match Figma:

1. **Product media gallery** (primary image + thumbnails; support at least 1–6 images)
2. **Title & vendor/brand**
3. **Pricing** (regular vs. sale price; show savings where applicable)
4. **Variant selection** (e.g., size, color) with disabled/out-of-stock states
5. **Quantity selector**
6. **Add to Cart** (no backend required; submit can be a no-op or simple JS toast)
7. **Description** (rich text)
8. **Specs / Details** (accordion or tabs)
9. **Reviews summary** (average rating + count; stars UI)
10. **Related products** (3–4 items; can reuse sample data)

---

## 6) Data Schema (Sample)

You can find a sample data schema in `/data/product.json`.


```
{
    "id": "d8f04529-2844-4314-88ff-e8f7099788c1",
    "product_class": null,
    "sku": "Z.18181.1754137942847106",
    "barcode": "",
    "name": "طقم مناشف موردة - 3 قطع\t\t\t\t\t\t\t\t",
    "slug": "طقم-مناشف-موردة-3-قطع",
    "price": 227.0,
    "sale_price": 159.0,
    "formatted_price": "227.00  ر.س ",
    "formatted_sale_price": "159.00  ر.س ",
    "currency": "SAR",
    "currency_symbol": " ر.س ",
    "display_order": 32766,
    "images": [ ... ],
    ...
}
```


---

## 7) Constraints & Assumptions

- **No backend** required; render with local JSON.
- If you add a small build step (e.g., Tailwind or PostCSS), document it.
- Use only open fonts/assets or those exported from Figma.

---

## 8) How to Run (Example)

- `npm install` (if you add a build step)
- `python -m http.server` or lightweight Flask app to render Jinja templates
- `python app.py` (if using Flask) → visit `http://localhost:5002/product/sku-123`

> If you use Flask for convenience, keep routing minimal and avoid framework-specific features beyond rendering Jinja.

---

## 9) Submission

- Share a **Git repository** link with instructions to run locally.
- Include screenshots for mobile & desktop.
- Provide a short note on what you'd tackle next with more time.

---

## 10) Questions

If anything is unclear, add reasonable assumptions to your README and proceed. We're evaluating judgment as much as execution.