Frontend Technical Test – Vue.js, Tailwind CSS, Pinia

AlienSoft Technologies is a bespoke software development company based in Nairobi Kenya. We are hiring interns for frontend Developers (VueJS). Interns will get a 6 months' contract with **KES 30,000 per month stipend**. After completion, we will on board candidates to full time employment based on performance.

Read through this exercise and complete it within the given time for evaluation.

1. Objective

Build a **mini Product Management App** in **Vue 3** that communicates with the public API at **dummyjson.com**.

The app must support:

- **User authentication** (login + token handling)
- **Product listing with search and filter**
- **View Product**
- **Add Product**
- **State management with Pinia**

Here are the mock-ups: https://drive.google.com/drive/folders/1XdcL0OTWzd-4hOoRbkY0mSnHBwq1HjTu?usp=sharing . **Match the layouts as closely as reasonably possible.** Primary colour: **#000080**.

Use of AI coding agents (e.g. ChatGPT, GitHub Copilot, etc.) is encouraged.
We will still evaluate how you structure, understand, and integrate the solution.

2. Tech Stack Requirements

- **Vue 3** (Composition API preferred)
- **Vue Router**
- **Pinia** (for auth & product state)
- **Tailwind CSS**
- You may use **any Tailwind UI library**
  - For engaged candidates, a **login for TailwindCSS Plus** will be provided so you can use their components when working. For now, you may use free components.

HTTP calls may use fetch or axios.

3. APIs

Use **DummyJSON**:

- **Auth API docs:** https://dummyjson.com/docs/auth

- **Products API docs:** https://dummyjson.com/docs/products

Use these APIs to:

- Authenticate (login)
- List products
- Get single product
- Add product
- (Nice to have) Update and delete product

Auth token from login must be used in subsequent product-related requests where required.

4. Required Pages

You must implement **4 pages**:

4.1 Login Page (/login)

Purpose: Authenticate user and store token in Pinia.

Requirements:

- Centered login form with:
  - Username
  - Password
  - Login button
- On submit:
  - Call DummyJSON **login endpoint**.
  - On success:
    - Save **token** (and optionally user info) in a **Pinia auth store**.
    - Persist auth (e.g. localStorage) so refresh keeps the session.
    - Redirect to **Home (/) / Products (/products)**.
  - On failure:
    - Show clear error message.

Route protection:

- If **not logged in**, users should only access /login.
- If **logged in**, prevent visiting /login (redirect to Home/Products).

4.2 Product List (/products)

Purpose: Display products in a tidy table with filters, using Pinia.

Top section (above table):

- Search input (at least by product title).
- Optional filters (e.g. category, price range) – bonus.
- **“Add New Product”** button on the right → navigates to Add Product page.

Table layout:

Each row should display:

- **Circular thumbnail** (from product thumbnail URL).
- **Title**
    - Below the title: a **truncated description** (e.g. first 80–100 characters + …) in a **faded/grey font**.
- **Category**
- **Price**
- **Stock**

Each row should be clickable or include a **“View”** action that goes to the **View Product** page.

Sample product fields (from API):

```
{
  "id": 1,
  "title": "Essence Mascara Lash Princess",
  "description": "The Essence Mascara Lash Princess is a popular mascara known for its volumizing and lengthening effects. Achieve dramatic lashes with this long-lasting and cruelty-free formula.",
  "category": "beauty",
  "price": 9.99,
  "discountPercentage": 10.48,
  "rating": 2.56,
  "stock": 99,
  "thumbnail": "https://cdn.dummyjson.com/product-images/beauty/essence-mascara-lash-princess/thumbnail.webp"
}
```

Product store (Pinia):

- State: products, isLoading, error, etc.
- Actions:

    - fetchProducts()
    - addProduct()
    - updateProduct()
    - deleteProduct()
- Home page should:
    - Load products via store.
    - Reactively filter results based on search/filter inputs.

Styling:

- Use **#000080** as main accent color (buttons, links, headings).
- Clean table, hover states, good spacing, rounded corners, subtle shadows.

4.3 View Product Page (/products/:id)

Purpose: Show full product details.

Requirements:

- Fetch product by id using DummyJSON (or from Pinia if already loaded).
- Show:
    - Large image / thumbnail
    - Title
    - Full description
    - Category
    - Price
    - Stock
    - (Nice to have) Rating and discount
- Actions:
    - **Back** to Home.
    - **Delete** button:
        - Confirm before deleting.
        - Call delete endpoint.
        - On success: update Pinia store and redirect to Home.
    - (Optional) **Edit**:
        - Allow updating product details via update endpoint.

        - Sync changes with store.

Design as a neat product detail card / layout.

4.4 Add Product Page (/products/new)

Purpose: Create a new product using DummyJSON.

Form fields:

- Title (required)
- Description
- Category
- Price
- Stock
- Thumbnail URL

Behavior:

- On submit:
    - Call add product endpoint.
    - On success:
        - Add the created product to Pinia store.
        - Redirect to Home or product details page.
    - Show basic validation and error messages.

5. State Management Details

Auth Store (Pinia)

- State:
    - token
    - user (optional)
- Getters / computed:
    - isAuthenticated
- Actions:
    - login(credentials)
    - logout()
    - restoreSession() – initialize state from localStorage on app load.

Use this auth state in **route guards** to protect all non-login pages.

Product Store (Pinia)

- State:
  - products
  - isLoading
  - error
- Actions:
  - fetchProducts()
  - fetchProductById(id)
  - addProduct(payload)
  - updateProduct(id, payload)
  - deleteProduct(id)
- Ensure API calls that require auth **use the saved token**.

6. UX & UI Expectations

- Use Tailwind classes (utility-first).
- Align with the provided mockups as much as possible.
- Primary color: **#000080**.
- Reasonable responsiveness (works nicely on laptop + tablet widths).
- Show:
  - Loading states (spinner/skeletons).
  - Error messages where applicable.

7. Deliverables

Please submit:

1. **Git repository link** (GitHub (public)) with:
   - Source code.
   - README.md including:
     - Setup steps (npm install, npm run dev, etc.).
     - Any environment/config needed.
     - Short explanation of your structure and any assumptions.

2. Host it on Github Pages

8. Rules, Tools & Deadline

- **You may use AI coding agents** (e.g. ChatGPT, Copilot).
  We will assess how well the final solution is structured, understandable, and working.
- You may use any **Tailwind UI library**.
  TailwindCSS Plus access will be provided to selected candidates so you can reuse component patterns from there. Use the free components for fast adoption

Deadline & Submission

- You have **23 hours** to complete the exercise.
- **Submission deadline:**
  25th November 2025, 2:00 PM (Africa/Nairobi time).
- **Submit to:** hr@aliensoft.co.ke

Late submissions may not be considered.