

**Products Review Tab – “Rate & read products”**

Here you see a grid of cosmetics and perfumes pulled from **/products**.

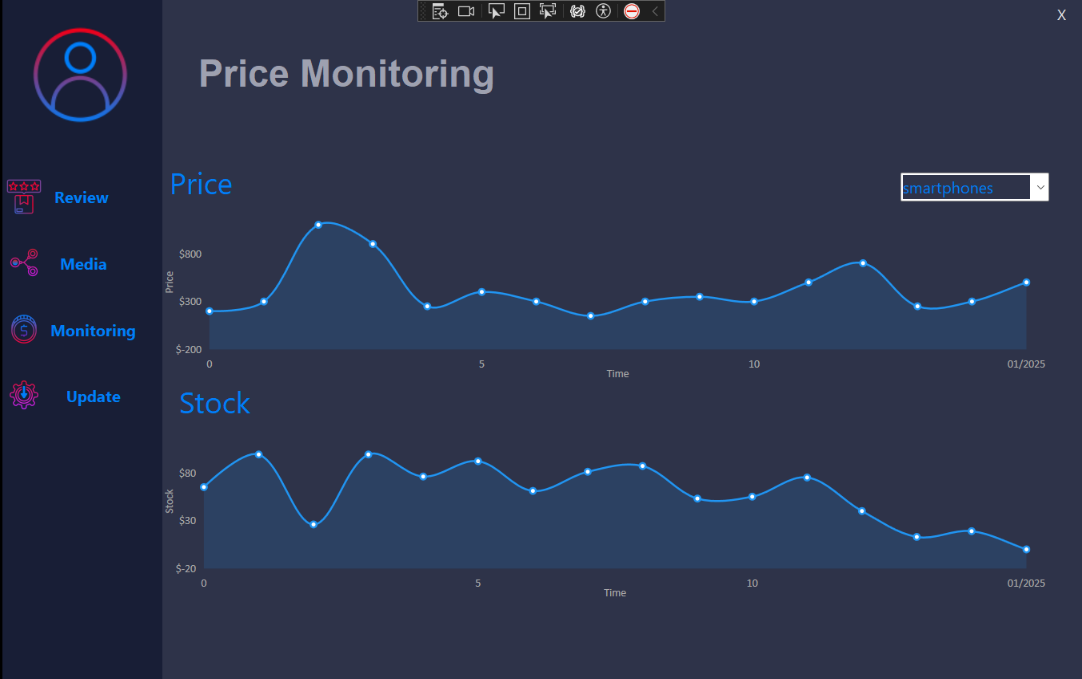
* **Tiles:** Each tile shows a product image, brand name, price in blue, and its average star-rating.
* **Tiny chat-bubble icon:** Click it to open or leave a review—reviews are stored via **/comments** and linked to the product ID.
* **Use case:** Mimics a light e-commerce front-end where shoppers skim prices, glance at ratings, and drill into reviews.



**Social Media Tab – “Mini-feed of user posts”**

This dashboard combines **/users**, **/posts**, and **/comments**:

* **Left-side avatar + username:** Pulled from /users.
* **Headline & topic tags:** Come from the post title plus its tags.
* **Story snippet:** The first ~3 sentences of each post’s body.
* **Chat-bubble:** Opens the comment thread for that post (data from /comments).
* **Why it matters:** Demonstrates how several endpoints can be stitched together to look like a social feed, complete with basic engagement.



**Price Monitoring Tab – “Follow price & stock trends”**

Two synchronized line charts visualize data from /products:

* **Top chart – Price:** Tracks historical selling price of the selected category (here, *smartphones*).
* **Bottom chart – Stock:** Tracks the number of units left in inventory over the same dates.
* **Dropdown (upper right):** Lets you switch categories; the graph redraws instantly with cached data or fetches fresh data if needed.
* **Take-away:** Shows how simple analytics can be layered on API data to spot spikes, troughs, or supply issues at a glance



**Update Tab – “Edit a user profile”**

This screen lets you test a **PATCH** request.

* **What you do:**
  1. Type the ID of the user you want to change in the gray field at the top.
  2. Edit any of the fields inside the card (first name, last name, age, or gender).
  3. Press **Save Changes**.
* **What happens behind the scenes:**
  1. The app sends a single PATCH call to **/users/{id}** on DummyJSON, containing only the fields you changed.
  2. Changes are first cached locally in memory so the UI updates instantly and then synced to the API.