📄 PRD: One-Screen App – "Widget Feed"

Product Vision

Create a single-screen mobile app that renders a **dynamic feed of UI widgets**, defined entirely by data from a REST API. The app demonstrates how dynamic and server-driven experiences can be architected flexibly.

---

1. **Goals**

- Display both **static** and **dynamic** widgets driven from API config.
- Implement backend API that serves widget data and supports mock dynamic content.
- Use this exercise to learn **GenAI-assisted development, clean code structure, and scalable UI patterns**.

---

2. **Widget Behavior**

🔹 Types of Widgets

Each widget is described by the backend and rendered accordingly on the frontend.

- **Static Widgets** All necessary data is included in the `/widgets` response.
- **Dynamic Widgets** The `/widgets` response includes metadata and a `dataUrl`. The frontend fetches widget-specific data and handles:
    - `loadingState`:
        - `skeleton`: show loading placeholder
        - `hidden`: don't render until data is ready
    - `errorState`:
        - `hide`: skip rendering
        - `show`: render a generic fallback/error view

---

3. **Widget Types**

`widgetType`	**Description**
`expandable_list`	List inside collapsible container with animated expand/collapse.
`horizontal_cards`	Carousel-like horizontal list of cards.
`image_list`	Vertical list: image on left, text in center, trailing badge or action on right.
`text_block`	Simple text/markdown block.
`highlight_banner`	Bold banner with image, title, and optional CTA.
`quick_actions`	Grid of tappable icons (e.g., shortcuts or tools).

---

4. **Backend API**

- Endpoint: `GET /widgets`
- Returns: list of ordered widget descriptors, including:

```json
{
  "id": "widget-123",
  "type": "dynamic",
  "widgetType": "horizontal_cards",
  "config": {
    "dataUrl": "https://.../widget-data",
    "loadingState": "skeleton",
    "errorState": "hide"
  }
}
```

- Dynamic widget data responses will vary by type (simple JSON content).

---

5. **AI Integration (Strongly Encouraged)**

Participants are expected to:

- Use GenAI for **code generation, API scaffolding, animations, tests, documentation, naming, and debugging**.
- Experiment with **multi-modal prompting**, such as code, UI descriptions, and error traces.
- Reflect on how GenAI influenced the speed, quality, and DX of their build.

---

6. Success Criteria

- Fully functional one-screen app.
- Backend API implemented and consumed correctly.
- Thoughtful handling of dynamic vs static data.
- **Minimum hand-written code** using AI where possible.
- Reflection during demo: what worked, what didn't, what GenAI enabled.

---

7. UI Layout Mock

One-Screen Feed Layout

None

```
+-------------------------------------------------+
| Header: "Your Feed"                             |
+-------------------------------------------------+
| [ Expandable List Widget ▸ ]                    |
|     • Line item 1                               |
|     • Line item 2                               |
|                                                 |
| [ Horizontal Cards →→ ]                         |
|   [ Card 1 ][ Card 2 ][ Card 3 ]                |
|                                                 |
| [ Text Block ]                                  |
|   "Some paragraph of text or markdown."         |
|                                                 |
| [ Image List ]                                  |
|   [🖼️] Label          Subtitle      [⏺️]        |
|   [🖼️] Label          Subtitle      [⏺️]        |
|                                                 |
| [ Banner ]                                      |
|   "Title and CTA" with background image         |
|                                                 |
| [ Quick Actions ]                               |
|   [📅] [📊] [💰] [⚙️]                            |
+-------------------------------------------------+
```