Senior Frontend Developer Assessment

🎯 Overview

Build a simplified **wishlist widget** that can be embedded on any website. The widget should allow users to organize content items (products, articles, etc.) into collections.

Figma Design Reference: https://www.figma.com/design/bET0YSkNKNtlWODBVcy3cT/plugilo-Dock?node-id=0-1&t=cTqMXuXQ78JYqsjf-1

Note: You can comment on the Figma design if you have questions or need clarification on any aspect of the design.

Core Components:

- **Dock**: Floating widget that opens/closes
- **Stacks**: Collections to organize items
- **Cards**: Individual content items

📌 Reference Example

Visit **https://www.it-daily.net/** to see a live example:

1. Accept all cookies
2. Click the **"Merken & Teilen"** button on any article
3. Try it in **Anonymous mode** (without login) to see the widget behavior
4. Note the interactions, animations, and user flow

Important: This is for understanding the concept and behavior only.

- **You don't need to copy the UI design** - create your own style
- Use your preferred CSS framework/library (Tailwind, Styled Components, MUI, ShadCN, etc.)
- Focus on the functionality and user experience patterns, but nice visual design is welcome

---

📋 What to Build

1. Dock Widget (Embeddable)

A floating button that expands into a panel showing the user's stacks.

Must Have:

- Minimize/expand states with smooth animations

- Can be embedded into third-party websites without conflicts
- Responsive (mobile + desktop)

Nice To Have:

- Use Web Component (Shadow DOM for style isolation)
- Theme support (light/dark)

Example Usage:

```
<wishlist-dock data-theme="dark"></wishlist-dock>
```

2. Stack Management

Users can create and organize collections.

Stack Fields:

- **Cover**: Image/color (can be randomly generated - random color, gradient, or placeholder image)
- **Name**: Stack title

Must Have:

- Create/delete stacks
- View list of stacks with card counts
- Click to view stack contents

3. Card Management

Display and manage items within stacks.

Card Fields:

- **Cover**: Image (required)
- **Name**: Card title (required)
- **Description**: Brief description (optional)
- **Selected Stack**: Which stack this card belongs to (required)

Must Have:

- Add/remove cards
- Display all card fields (cover, name, description)
- Move cards between stacks (drag & drop or select different stack)
- **Swipe Mode**: Cards displayed in swipeable stack (like Tinder/card deck)
  - Use a library to support swiping cards
  - Swipe left/right to navigate through cards

    - Opens when clicking on a stack
    - Show card counter

4. State & Data

Must Have:

- State management solution (your choice)
- **Optimistic UI updates**: Update UI immediately, sync with API later
- API layer for CRUD operations (use fake/mock API service if needed)

Nice To Have:

- Data persistence (localStorage, IndexedDB, or similar)
- Loading and error states

Important - Optimistic UI:

- When user adds/removes cards → Update UI instantly
- When user creates/deletes stacks → Update UI instantly
- Sync with backend in the background
- Handle API failures gracefully (rollback with error message)
- Show sync indicators when needed

🛠️ Technical Requirements

Required:

- React 18+
- Embeddable component (iframe, Web Component, or similar approach)
- Modern CSS (your choice of approach)
- Clean, maintainable code structure

You Decide:

- State management approach
- Styling solution
- Libraries/tools to use
- How to split and organize features
- Code splitting strategy

📊 Evaluation Focus

We'll assess your skills in:

1. **React Architecture**
2. **Embeddable Widget Integration**
3. **Code Quality**
4. **State Management**
5. **Styling & UX**
6. **Feature Implementation**

📦 Deliverables

1. **GitHub Repository**
    - Clean commit history
    - Source code
2. **README.md** with:
    - Setup instructions
    - How to embed the widget
    - Architecture decisions
    - Trade-offs you made
    - What you'd improve with more time
3. **Demo**
    - Live URL

💡 Optional Enhancements

Pick any that showcase your strengths:

- Unit/integration tests
- Code splitting & lazy loading
- Advanced animations
- Accessibility features
- Offline support
- Search/filter functionality
- Keyboard shortcuts

📝 Notes

- **Quality over quantity** - Focus on core features done well
- **Document decisions** - Explain your choices in README
- **Use your judgment** - Choose tools and patterns you're comfortable with
- **Be pragmatic** - Mock what's needed, skip what's not critical

---

Good luck! We're excited to see your approach. 🚀