Trial Day Project: **Interactive Template Builder**

Overview

Build an interactive popup template builder that allows users to create, customize, and preview marketing popup templates using drag-and-drop functionality. This tool simulates a core feature of SaaS marketing platforms where non-technical users design popups without coding.

User Story

As a **marketing manager**, I want to **visually design popup templates by dragging and arranging elements** so that **I can quickly create professional-looking popups without developer assistance**.

Requirements

Core Features

1. **Canvas Area**
    a. Fixed-size canvas (e.g., 400x500px) representing the popup
    b. Visual boundary indicating the editable area

2. **Element Palette**
    a. Draggable elements: Heading, Text, Button, Image placeholder
    b. Elements can be dragged from palette onto canvas
    c. Visual feedback during drag operation

3. **Element Manipulation**
    a. Select elements on canvas by clicking
    b. Move elements by dragging within canvas
    c. Resize elements using corner handles
    d. Delete selected elements (keyboard or button)

4. **Properties Panel**
    a. When an element is selected, show editable properties:
        i. **Heading/Text:** content, font size, color, alignment
        ii. **Button:** text, background color, text color, border radius
        iii. **Image:** URL input, alt text
    b. Changes reflect immediately on canvas (live preview)
5. **Template Management**

a. Save template to backend (**POST** endpoint)
b. Load existing templates (**GET** endpoint)
c. List saved templates with ability to load/delete
d. Export template as **JSON**

Technical Requirements

Requirement	***Specification***
Frontend Framework	Vue 3 (Composition API)
Language	TypeScript (strict mode)
Backend	Node.js with Express
Testing	Jest - minimum 2 meaningful tests
Database	In-memory or JSON file (no external DB required)
Styling	CSS/SCSS (no UI framework required)
Build Tool	Vite

Allowed Libraries

1. Vue 3, Vue Router (if needed)
2. Pinia or Vuex for state management
3. Any drag-and-drop library (vue-draggable, @vueuse/core, or native HTML5 DnD)
4. uuid or nanoid for ID generation
5. Express for backend
6. Jest + Vue Test Utils for testing

API Endpoints

Method	Endpoint	Description
GET	/api/templates	List all saved templates
POST	/api/templates	Create/update template
DELETE	/api/templates/:id	Delete template

Expected Data Structure

TypeScript

```typescript
interface Position {
  x: number;
  y: number;
}

interface Size {
  width: number;
  height: number;
}

interface BaseElement {
  id: string;
  type: ElementType;
  position: Position;
  size: Size;
}

interface HeadingElement extends BaseElement {
  type: 'heading';
  content: string;
  fontSize: number;
  color: string;
  alignment: 'left' | 'center' | 'right';
}

interface TextElement extends BaseElement {
  type: 'text';
  content: string;
  fontSize: number;
  color: string;
  alignment: 'left' | 'center' | 'right';
}

interface ButtonElement extends BaseElement {
  type: 'button';
  text: string;
  backgroundColor: string;
```

```
  textColor: string;
  borderRadius: number;
}

interface ImageElement extends BaseElement {
  type: 'image';
  url: string;
  altText: string;
}

type ElementType = 'heading' | 'text' | 'button' | 'image';
type TemplateElement = HeadingElement | TextElement |
ButtonElement | ImageElement;

interface Template {
  id: string;
  name: string;
  elements: TemplateElement[];
  canvasSize: Size;
  backgroundColor: string;
  createdAt: string;
  updatedAt: string;
}
```

Bonus Features

If time permits, consider implementing:

- **Undo/Redo** (Ctrl+Z/Y with history states)
- **Z-index management** (bring forward/send backward)
- **Grid snap** functionality for precise element placement
- **Keyboard shortcuts** (Delete, arrow keys for nudging)
- **Copy/paste elements** (Ctrl+C/V)
- **Template preview modal** (see popup as end-user would)
- **Import template from JSON**

Evaluation Criteria

Criteria	***Weight***
Code Architecture	20%
TypeScript Usage	20%
Functionality	25%
Code Quality	15%
Testing	10%
UI/UX Quality	10%

Time Limit

You have **4 days** to complete this project. We expect **5-8 hours** of actual development time.

Focus on:

1. **Core functionality working correctly**
2. **Clean, maintainable code**
3. **Good TypeScript practices**
4. **At least 2 meaningful tests**

It's better to have solid core features than incomplete bonus features.

Submission Instructions

1. Create a **private GitHub repository**
2. Include a **README.md** with:
    a. Setup instructions (how to install and run)
    b. Brief description of your architecture decisions
    c. Any assumptions you made
    d. What you would improve with more time
3. Ensure the project runs with:

```shell
npm install
npm run dev    # Start the application
npm run server # Start backend (can be combined)
```

```
npm run test   # Run tests
```

4. **(Optional)** Deploy to Vercel, Netlify, or similar
5. Submit the GitHub repository URL when complete.