**Comprehensive Vue 3 Component Creation Prompt for Existing Library**

I need you to create a new component for my Vue 3 component library that follows the same architecture, patterns, and code style already established. First, analyze my request to determine which type of component is needed:

**Component Type Decision Logic:**

* **Core Component**: If my request describes a simple UI element with no business logic, data fetching, or composition of other components (e.g., button, input, icon, badge).
* **Composite Component**: If my request describes a component that combines multiple core components with local state management but doesn't connect to external data sources (e.g., form, tabs, accordion, button group).
* **Smart Component**: If my request describes a component that requires data fetching, uses adapters, or integrates with APIs/services (e.g., data table, search, auto-complete, file uploader).

**Library Structure Overview:**

My component library follows a three-tier architecture with this structure:

vue-component-library/

├── src/

│ ├── adapters/ # API abstraction layer with adapter pattern

│ │ ├── httpAdapter.ts

│ │ ├── graphQLAdapter.ts

│ │ ├── mockAdapter.ts

│ │ └── dataTableAdapter.ts

│ ├── components/

│ │ ├── core/ # Simple UI components with no business logic

│ │ ├── composite/ # Combinations of core components

│ │ └── smart/ # Complex components using adapters

│ ├── composables/ # Reusable composition functions

│ │ ├── useTheme.ts

│ │ └── useI18n.ts

│ ├── styles/

│ │ ├── themes/ # Theme configuration

│ │ ├── tokens/ # Design variables (colors, spacing, typography)

│ │ └── utils/ # Mixins and utilities

│ ├── types/ # TypeScript definitions

│ └── index.ts # Main library entry point

├── tests/

│ └── unit/ # Unit tests for components

└── examples/ # Example implementations

**Implementation Requirements:**

1. **TypeScript Types**: Create in src/types/[componentName].ts
   * Define detailed interfaces for props, events, models
   * Follow existing patterns for type organization
   * Export types for library users
2. **Component Implementation**:
   * Use the appropriate folder based on component type
   * Follow Vue 3 Composition API with <script setup lang="ts"> syntax
   * Use existing composables like useTheme and useI18n where appropriate
   * Ensure components are accessible with proper ARIA attributes
   * Follow the established design system with CSS variables
3. **Adapter Pattern** (for smart components):
   * Create/extend adapters in the appropriate adapter file
   * Implement factory functions for creating adapters
   * Make components data-source agnostic
4. **Testing**:
   * Create tests/unit/[ComponentName].test.ts
   * Use Vitest and Vue Test Utils as in existing tests
   * Test all props, events, and user interactions
   * Test various states (loading, error, success, etc.)
5. **Documentation**:
   * Add detailed JSDoc comments
   * Document props, events, slots, and methods
   * Include usage examples and edge cases
6. **Library Integration**:
   * Update src/index.ts to export the new component
   * Add component to the Vue plugin installation function

**Coding Style Guidelines:**

* Use TypeScript throughout with proper typing
* Follow composition API best practices
* Prefer ref and computed over reactive
* Use destructuring and modern JS features
* Organize imports consistently
* Follow the existing naming conventions

**Example Component Creation Process:**

Please follow the example of existing components like VButton, VButtonGroup, and VDataTable, which showcase the implementation pattern for each tier. The smart components use the adapter pattern to abstract data fetching, while composite components combine and manage multiple core components.

**My Component Request:**

[I will describe the component I need here]

Based on my description, determine the appropriate tier, implement the complete component with all necessary files, and provide example usage showing how it integrates with the existing library.

**Universal Debugging Prompt for Vue Component Library**

I'm developing a Vue 3 component library with a three-tier architecture (core, composite, and smart components). Here's the context needed to help me solve the issue I'm facing:

**Project Structure**

My library is organized as follows:

vue-component-library/

├── src/

│ ├── adapters/ # API abstraction with adapter pattern

│ ├── components/

│ │ ├── core/ # Simple UI components

│ │ ├── composite/ # Combined components with local state

│ │ └── smart/ # Complex components with data fetching

│ ├── composables/ # Shared functionality (theme, i18n)

│ ├── styles/ # Design system and theme

│ ├── types/ # TypeScript definitions

│ └── index.ts # Main entry point

├── tests/ # Component tests

└── examples/ # Usage examples

**Technology Stack**

* Vue 3 with Composition API and <script setup> syntax
* TypeScript for type safety
* Vite for building
* Vitest for testing
* SCSS for styling with CSS variables for theming
* Adapter pattern for API abstraction

**Common Issues I Might Face**

* TypeScript type errors or missing type definitions
* Component communication problems (props/events)
* CSS scoping or styling inconsistencies
* Adapter implementation issues
* Testing component interactions
* Build configuration problems
* Dependency issues

**Current Problem**

[I will describe my specific issue here]

Please provide:

1. A diagnosis of what might be causing the problem
2. A detailed solution with code examples based on my project structure
3. Any relevant explanations about Vue 3, TypeScript, or the architecture pattern
4. Preventative measures to avoid similar issues in the future

If more information is needed, please let me know what would help diagnose the issue more effectively.

**1. Component Enhancement Prompt**

This prompt would help when you want to enhance existing components rather than create new ones:

# Vue Component Enhancement Prompt

I need to enhance an existing component in my Vue 3 component library. Here's the context:

## Current Component:

[Component name and brief description]

## Current Implementation:

[Key aspects of current implementation - props, events, functionality]

## Desired Enhancements:

[What new features, props, or behavior you want to add]

Please provide:

1. A detailed implementation plan for the enhancements

2. Any TypeScript type additions/modifications needed

3. Updated test cases to cover the new functionality

4. Example usage showing the enhanced capabilities

5. Any refactoring suggestions to maintain code quality

All code should follow my existing architecture and patterns described below:

[Brief architecture reminder]

**2. Documentation Generator Prompt**

For creating comprehensive documentation for your components:

# Vue Component Documentation Generator

I need comprehensive documentation for this component in my Vue 3 component library:

## Component: [Name]

Please generate complete documentation including:

1. Component overview and purpose

2. Installation and import information

3. Props API table with types, defaults, and descriptions

4. Events API table with payload types and trigger conditions

5. Slots information and usage

6. Accessibility considerations

7. Browser compatibility notes

8. Basic usage example

9. Advanced usage examples showing different configurations

10. Common pitfalls and best practices

11. Related components

The documentation should be formatted in Markdown and follow the style of my existing documentation.

**3. Performance Optimization Prompt**

For when you need help optimizing components:

# Vue Component Performance Optimization

I need to optimize this component in my Vue 3 component library:

## Component: [Name]

## Current Performance Issues:

[Describe rendering, reactivity, or other performance concerns]

## Implementation Details:

[Key aspects of the implementation that might be relevant]

Please provide:

1. Analysis of potential performance bottlenecks

2. Specific optimization techniques for Vue 3 components

3. Code changes to implement these optimizations

4. Before/after comparisons of expected performance impact

5. Testing strategies to verify improvements

All optimizations should maintain the existing API and functionality of the component.

**4. Project Migration/Upgrade Prompt**

For when you need to upgrade or migrate parts of your library:

# Vue Library Migration/Upgrade Plan

I need to upgrade/migrate the following aspect of my Vue 3 component library:

## Target for Upgrade:

[Specific part of library - dependency, pattern, API, etc.]

## Current Implementation:

[Brief description of current approach]

## Desired End State:

[What you want to achieve with the upgrade]

Please provide:

1. A step-by-step migration plan

2. Code changes required for each step

3. Potential breaking changes and how to handle them

4. Testing strategy to ensure nothing breaks

5. Rollback plan if issues arise

Consider the impact on the entire component ecosystem and minimize disruption to library users.

**5. Integration Testing Prompt**

For creating more comprehensive integration tests between components:

# Vue Component Integration Testing Plan

I need to create integration tests for these components in my Vue 3 component library:

## Components:

[List of components that interact with each other]

## Interaction Scenarios:

[How these components work together]

Please provide:

1. A comprehensive test plan covering all interaction scenarios

2. Test cases written for Vitest with Vue Test Utils

3. Mock implementations needed for testing

4. Edge cases that should be covered

5. Performance considerations in testing

The tests should verify both the individual component behavior and their interaction patterns.