Frontend Developer Hiring Assignment

Kanban Board View - Interactive UI Component Development

Overview

Welcome to the **Design System Component Library** frontend developer hiring challenge. This assignment evaluates your ability to build **production-grade, complex interactive components** that align with our design system architecture.

You will implement a sophisticated **Kanban Board View** component from scratch using modern web technologies. Your component will be showcased via **Storybook** stories demonstrating all features and interactions.

Timeline

Estimated Time: 8-12 hours
Submission Deadline: As specified in your Internshala application

Submission Method

Storybook Required: Your component must be documented and demonstrated through Storybook stories showing all states, interactions, and variants.

Objective

Build a fully functional **Kanban Board View** - a drag-and-drop task management system.

Your implementation should demonstrate:

- **Production-quality code architecture**
- **Enterprise-grade UI/UX patterns**
- **Performance optimization techniques**
- **Accessibility-first approach**
- **Scalable component design**

[illegible]

Technology Stack

Required Technologies

Technology	Purpose	Version
TypeScript	Type-safe development	^5.0.0
React	Component framework	^18.0.0
Tailwind CSS	Utility-first styling	^3.0.0
Vite or **Next.js**	Build tooling	Latest stable

[illegible]

[illegible]

Explicitly Forbidden

Component Libraries: Radix UI, Shadcn, Headless UI, MUI, Ant Design, Chakra UI, Mantine
CSS-in-JS: styled-components, emotion, vanilla-extract, stitches
UI Generators: Lovable, Locofy, TeleportHQ, Uizard, Builder.io
Drag Libraries: react-beautiful-dnd, dnd-kit, react-dnd (see exceptions below)
Pre-built Kanban Components: Any library that provides ready-made kanban boards

Allowed Utilities

date-fns or **dayjs** (date manipulation only)
clsx or **classnames** (conditional class management)
zustand or **jotai** (lightweight state management)
framer-motion (animations - bonus only)
@dnd-kit/core (low-level drag primitives only, not pre-built components)
Storybook (required for component documentation)

> ***Important:*** *If you use `@dnd-kit`, you must implement your own drag logic and visual feedback. Simply wrapping pre-built hooks without custom implementation will be considered non-compliant.*

Storybook Requirements

Your Storybook stories must include:

- **Default** - Basic kanban board with sample data
- **Empty State** - Board with no tasks
- **With Many Tasks** - Board with 20+ tasks to test performance
- **Different Priorities** - Showcase priority levels
- **Interactive Demo** - Fully functional drag-and-drop
- **Mobile View** - Responsive layout demonstration
- **Accessibility** - Keyboard navigation demonstration

Required Project Structure

```
kanban-component/
│
├── README.md                      # Documentation
├── package.json                   # Dependencies
├── tsconfig.json                  # TypeScript config
├── tailwind.config.js             # Tailwind customization
├── .storybook/                    # Storybook configuration
│   ├── main.ts
│   └── preview.ts
│
└── src/
    ├── components/
    │   ├── KanbanBoard/
    │   │   ├── KanbanBoard.tsx        # Main component
    │   │   ├── KanbanBoard.stories.tsx # Storybook stories
    │   │   ├── KanbanBoard.types.ts
    │   │   ├── KanbanColumn.tsx
    │   │   ├── KanbanCard.tsx
    │   │   └── TaskModal.tsx
```

[illegible]

[illegible]

```
    │   │
    │   └── primitives/                 # Reusable UI elements
    │       ├── Button.tsx
    │       ├── Modal.tsx
    │       └── Avatar.tsx
    │
    ├── hooks/
    │   ├── useDragAndDrop.ts
    │   └── useKanbanBoard.ts
    │
    ├── utils/
    │   ├── task.utils.ts
    │   └── column.utils.ts
    │
    └── styles/
        └── globals.css
```

Design Requirements

Visual Design Principles

Your implementation should follow **modern SaaS product design patterns**:

1. **Clean & Minimal** - Remove visual noise, focus on content
2. **Consistent Spacing** - Use Tailwind's spacing scale (4px base unit)
3. **Clear Hierarchy** - Typography and color establish importance
4. **Subtle Interactions** - Micro-animations provide feedback
5. **Purposeful Color** - Use color to communicate state and action

Tailwind Configuration

Extend Tailwind with design tokens that align with our system:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: {
          50: '#f0f9ff',
          100: '#e0f2fe',
          500: '#0ea5e9',
          600: '#0284c7',
          700: '#0369a1',
        },
        neutral: {
          50: '#fafafa',
          100: '#f4f4f5',
          200: '#e4e4e7',
          300: '#d4d4d8',
          700: '#3f3f46',
          900: '#18181b',
        },
      },
      spacing: {
        18: '4.5rem',
        88: '22rem',
      },
      borderRadius: {
        'xl': '0.75rem',
      },
    },
  },
}
```

Responsive Breakpoints

Breakpoint	Width	Target Device	Layout Behavior
`sm`	640px+	Large mobile	Stack columns, expand cards
`md`	768px+	Tablet	2-column layouts, side panels
`lg`	1024px+	Desktop	Full multi-column, split views
`xl`	1280px+	Large desktop	Max width containers, sidebars

Kanban Board View Detailed Requirements

Core Features

1. Data Structure

```
interface KanbanTask {
  id: string;
  title: string;
  description?: string;
  status: string; // column identifier
  priority?: 'low' | 'medium' | 'high' | 'urgent';
  assignee?: string;
  tags?: string[];
  createdAt: Date;
  dueDate?: Date;
}

interface KanbanColumn {
  id: string;
  title: string;
  color: string;
  taskIds: string[]; // ordered list of task IDs
  maxTasks?: number; // WIP limit (optional)
}

interface KanbanViewProps {
  columns: KanbanColumn[];
  tasks: Record<string, KanbanTask>;
  onTaskMove: (taskId: string, fromColumn: string, toColumn: string, newIndex: number)
=> void;
  onTaskCreate: (columnId: string, task: KanbanTask) => void;
  onTaskUpdate: (taskId: string, updates: Partial<KanbanTask>) => void;
  onTaskDelete: (taskId: string) => void;
}
```

2. Board Layout

- Minimum 3 columns, support up to 6 columns
- Each column has fixed width (280-320px) on desktop
- Horizontal scroll with smooth snap behavior
- Column headers are sticky during vertical scroll
- Empty state message when column has no tasks

3. Task Card Requirements

Must display:

- Task title (bold, truncated to 2 lines)
- Priority indicator (colored left border)
- Assignee avatar or initials
- Tag badges (max 3 visible)
- Due date badge (red if overdue)
- Comments/attachments count icons (optional)

```
// Example Task Card
<div className="bg-white border border-neutral-200 rounded-lg p-3 shadow-sm
hover:shadow-md transition-shadow cursor-grab active:cursor-grabbing">
  <div className="flex items-start justify-between mb-2">
    <h4 className="font-medium text-sm text-neutral-900 line-clamp-2">
      {task.title}
    </h4>
    {task.priority && (
      <span className={`text-xs px-2 py-0.5 rounded
${priorityColors[task.priority]}`}>
        {task.priority}
      </span>
    )}
  </div>

  {task.description && (
    <p className="text-xs text-neutral-600 mb-2 line-clamp-2">
      {task.description}
    </p>
  )}

  <div className="flex items-center justify-between">
    <div className="flex gap-1">
      {task.tags?.slice(0, 3).map(tag => (
        <span key={tag} className="text-xs bg-neutral-100 px-2 py-0.5 rounded">
          {tag}
        </span>
      ))}
    </div>

    {task.assignee && (
      <div className="w-6 h-6 bg-primary-500 rounded-full text-white text-xs flex items-center justify-center">
        {getInitials(task.assignee)}
      </div>
    )}
  </div>

  {task.dueDate && (
    <div className={`text-xs mt-2 ${isOverdue(task.dueDate) ? 'text-red-600' : 'text-neutral-500'}`}>
      Due: {formatDate(task.dueDate)}
    </div>
  )}
</div>
```

4. Drag-and-Drop Interactions

Scenario	Behavior
Start drag	Card lifts with shadow, cursor changes to grab
Dragging	Ghost/placeholder shows drop position
Hover column	Column highlights as valid drop target
Drop in column	Animate card into position, update state
Drop between tasks	Insert at exact position with reordering
Invalid drop	Card animates back to original position
Keyboard drag	Space to pick up, arrows to move, Enter to drop

5. Column Management

- Header shows: Title, task count, and WIP limit indicator
- "Add Task" button at bottom of column
- Column options menu (rename, set WIP limit, delete)
- Collapse/expand column functionality
- Drag column header to reorder columns (bonus)

6. Task Detail Modal

When clicking a task card, open modal with:

- Editable title and description (rich text optional)
- Priority selector
- Status/column dropdown
- Assignee search/select
- Tag management (add/remove)
- Due date picker
- Activity log (bonus - show move history)
- Delete task button

- Comments section (bonus)

7. Advanced Features

- **Search/Filter:** Filter tasks by assignee, tag, or priority
- **Bulk Actions:** Select multiple cards with checkboxes
- **Quick Actions:** Hover card to show quick edit, delete, duplicate icons
- **Column Limits:** Visual warning when approaching WIP limit

8. Responsive Behavior

- **Desktop:** Horizontal columns with independent scrolling
- **Tablet:** 2 columns visible, horizontal scroll
- **Mobile:** Vertical stack, swipe between columns, tab navigation

---

Accessibility Requirements

All implementations **must** meet WCAG 2.1 AA standards:

Keyboard Navigation

Key	Action
`Tab`	Move focus between interactive elements
`Shift + Tab`	Move focus backwards
`Enter` / `Space`	Activate focused element or pick up card
`Escape`	Close modal or cancel action
`Arrow Keys`	Navigate between cards or columns
`Home` / `End`	Jump to first/last card in column

ARIA Implementation

Required ARIA attributes:

```
// Example Draggable Card
<div
  role="button"
  tabIndex={0}
  aria-label={`${task.title}. Status: ${status}. Priority: ${priority}. Press space to grab.`}
  aria-grabbed={isDragging}
  onKeyDown={handleDragKeyboard}
>
  {/* card content */}
</div>

// Example Column
<div
  role="region"
  aria-label={`${column.title} column. ${taskCount} tasks.`}
>
```

```
  {/* column content */}
</div>

// Example Modal
<div
  role="dialog"
  aria-modal="true"
  aria-labelledby="modal-title"
  aria-describedby="modal-description"
>
  <h2 id="modal-title">Edit Task</h2>
  <div id="modal-description">Update task details below</div>
  {/* modal content */}
</div>
```

Visual Accessibility

- All interactive elements must have `:focus-visible` styles
- Color contrast ratio minimum 4.5:1 for text
- Focus indicators must be clearly visible (not `outline: none` without replacement)
- Text must be resizable up to 200% without loss of functionality

---

Performance Requirements

Your implementation will be tested for performance under stress conditions.

Performance Benchmarks

Metric	Target	Measurement
Initial Render	< 300ms	Time to interactive
Drag Response	< 16ms	Frame time during drag
Search/Filter	< 100ms	Results update latency
Large Dataset	Handle 500+ tasks	No visible lag
Bundle Size	< 200kb (gzipped)	Production build

Optimization Techniques

Required:

1. Use `React.memo()` for expensive components
2. Implement virtualization for long lists (>50 items per column)
3. Debounce search and filter inputs
4. Lazy load modals and detail views
5. Use `useCallback` and `useMemo` appropriately

Example Virtualization:

```
// Simplified virtual scrolling for Kanban column
const [visibleRange, setVisibleRange] = useState({ start: 0, end: 20 });
```

```tsx
const handleScroll = (e: React.UIEvent<HTMLDivElement>) => {
  const scrollTop = e.currentTarget.scrollTop;
  const itemHeight = 120; // average card height
  const start = Math.floor(scrollTop / itemHeight);
  const end = start + 20; // visible items + buffer

  setVisibleRange({ start, end });
};

return (
  <div className="overflow-y-auto" onScroll={handleScroll} style={{ height: '70vh' }}>
    <div style={{ height: tasks.length * 120 }}>
      {tasks.slice(visibleRange.start, visibleRange.end).map(task => (
        <TaskCard key={task.id} task={task} />
      ))}
    </div>
  </div>
);
```

Code Quality Standards

TypeScript Standards

1. Strict Mode Enabled

```json
// tsconfig.json
{
  "compilerOptions": {
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true
  }
}
```

2. Comprehensive Type Definitions

- No `any` types (use `unknown` if needed)
- Interface over type aliases for object shapes
- Proper generic constraints
- Discriminated unions for complex states

3. Example Type Safety

```typescript
// Good
interface TaskFormData {
  title: string;
  description?: string;
  priority: 'low' | 'medium' | 'high' | 'urgent';
  assignee?: string;
  tags: string[];
}

type FormErrors = Partial<Record<keyof TaskFormData, string>>;

// Bad ▯
interface TaskFormData {
  title: any;
  description: any;
  priority: any;
}
```

Code Organization

1. Component Structure

```
// KanbanCard.tsx

import React from 'react';
import { KanbanTask } from '@/types/kanban.types';
import { formatDate, isOverdue } from '@/utils/task.utils';

interface KanbanCardProps {
  task: KanbanTask;
  isDragging: boolean;
  onEdit: (task: KanbanTask) => void;
  onDelete: (taskId: string) => void;
}

export const KanbanCard: React.FC<KanbanCardProps> = ({
  task,
  isDragging,
  onEdit,
  onDelete,
}) => {
  // Component logic

  return (
    // JSX
  );
};
```

2. Custom Hooks Pattern

```
// useDragAndDrop.ts

import { useState, useCallback } from 'react';

interface DragState {
  isDragging: boolean;
  draggedId: string | null;
  dropTargetId: string | null;
  dragOverIndex: number | null;
}
```

```
export const useDragAndDrop = () => {
  const [state, setState] = useState<DragState>({
    isDragging: false,
    draggedId: null,
    dropTargetId: null,
    dragOverIndex: null,
  });

  const handleDragStart = useCallback((id: string) => {
    setState(prev => ({
      ...prev,
      isDragging: true,
      draggedId: id,
    }));
  }, []);

  const handleDragOver = useCallback((targetId: string, index: number) => {
    setState(prev => ({
      ...prev,
      dropTargetId: targetId,
      dragOverIndex: index,
    }));
  }, []);

  const handleDragEnd = useCallback(() => {
    setState({
      isDragging: false,
      draggedId: null,
      dropTargetId: null,
      dragOverIndex: null,
    });
  }, []);

  return {
    ...state,
    handleDragStart,
    handleDragOver,
    handleDragEnd,
  };
};
```

3. Utility Function Pattern

```
// task.utils.ts

/**
 * Checks if a task is overdue
 */
export const isOverdue = (dueDate: Date): boolean => {
  return new Date() > dueDate;
};
```

```
/**
 * Gets initials from a name
 */
export const getInitials = (name: string): string => {
  return name
    .split(' ')
    .map(part => part[0])
    .join('')
    .toUpperCase()
    .slice(0, 2);
};

/**
 * Calculates priority color classes
 */
export const getPriorityColor = (priority: string): string => {
  const colors = {
    low: 'bg-blue-100 text-blue-700 border-l-4 border-blue-500',
    medium: 'bg-yellow-100 text-yellow-700 border-l-4 border-yellow-500',
    high: 'bg-orange-100 text-orange-700 border-l-4 border-orange-500',
    urgent: 'bg-red-100 text-red-700 border-l-4 border-red-500',
  };
  return colors[priority as keyof typeof colors] || colors.medium;
};

/**
 * Reorders tasks after drag and drop
 */
export const reorderTasks = (
  tasks: string[],
  startIndex: number,
  endIndex: number
): string[] => {
  const result = Array.from(tasks);
  const [removed] = result.splice(startIndex, 1);
  result.splice(endIndex, 0, removed);
  return result;
};

/**
 * Moves task between columns
 */
export const moveTaskBetweenColumns = (
  sourceColumn: string[],
  destColumn: string[],
  sourceIndex: number,
  destIndex: number
): { source: string[]; destination: string[] } => {
  const sourceClone = Array.from(sourceColumn);
  const destClone = Array.from(destColumn);
  const [removed] = sourceClone.splice(sourceIndex, 1);
  destClone.splice(destIndex, 0, removed);

  return {
    source: sourceClone,
    destination: destClone,
  };
};
```

□ Submission Requirements

1. Repository Setup

Your GitHub repository must include:

```
□ README.md with complete documentation
□ package.json with all dependencies (including Storybook)
□ .gitignore (exclude node_modules, storybook-static)
□ Source code in /src following required structure
□ Storybook configuration in .storybook/
□ Component stories (.stories.tsx files)
□ At least 5 meaningful commits showing development progress
□ Deployed Storybook (Chromatic/Vercel/Netlify)
□ NO node_modules folder
□ NO build artifacts
□ NO API keys or secrets
```

2. Storybook Documentation

Your Storybook must include:

Required Stories:

1. **Default** - Standard board with 4 columns and sample tasks
2. **Empty** - Empty board state
3. **Large Dataset** - Board with 30+ tasks across columns
4. **Mobile Responsive** - Mobile viewport demonstration
5. **Interactive Playground** - Fully functional with controls

Story Controls:

- Toggle dark mode (bonus)
- Adjust column count
- Change task priorities
- Add/remove sample tasks

3. README.md Format

~~~~
# Kanban Board Component

## □ Live Storybook
[Your Deployed Storybook URL]

## □ Installation
\`\`\`bash
~~~~

```
npm install
npm run storybook
\`\`\`

## ▯ Architecture
[Brief explanation of your approach]

## ▯ Features
- [x] Drag-and-drop tasks
- [x] Task creation/editing
- [x] Responsive design
- [x] Keyboard accessibility

## ▯ Storybook Stories
- Default board
- Empty state
- Large dataset
- Mobile view
- Interactive playground

## ▯ Technologies
- React + TypeScript
- Tailwind CSS
- Storybook
- [Other libraries]

## ▯ Contact
[Your email]
```

4. Git Commit Guidelines

4. Git Commit Guidelines

Follow conventional commit format:

```
feat: add drag and drop for kanban cards
feat: implement task creation modal
fix: resolve card positioning bug during drag
style: improve mobile responsiveness for columns
refactor: extract task modal into separate component
docs: update README with installation instructions
perf: implement virtualization for large task lists
```

5. Storybook Deployment

Required: Deploy your Storybook to one of these platforms:

- **Chromatic** (recommended - free for open source)
- **Vercel**
- **Netlify**
- **GitHub Pages**

Include the deployed Storybook link in your README and submission.

Evaluation Rubric

Your submission will be scored across these dimensions:

1. Functionality (30 points)

Criteria	Points	Description
Core features work correctly	15	All required interactions function without errors
Edge cases handled	8	Validates inputs, handles empty states, prevents crashes
Data persistence works	7	State updates correctly, can add/edit/delete/move tasks

2. Code Quality (25 points)

Criteria	Points	Description
TypeScript usage	8	Proper types, no `any`, strict mode enabled
Component architecture	8	Clean separation, reusable components, single responsibility
Code organization	5	Logical folder structure, proper imports
Comments & docs	4	Code is self-documenting with strategic comments

3. UI/UX Design (20 points)

Criteria	Points	Description
Visual polish	8	Professional appearance, consistent styling
Interaction feedback	6	Hover states, drag feedback, smooth transitions
Responsive design	6	Works seamlessly on mobile, tablet, desktop

4. Accessibility (10 points)

Criteria	Points	Description
Keyboard navigation	4	All features accessible via keyboard
ARIA implementation	3	Proper labels, roles, live regions
Focus management	3	Logical focus order, visible focus indicators

5. Performance (10 points)

Criteria	Points	Description
Optimized rendering	5	No unnecessary re-renders, uses memoization
Handles large datasets	3	Virtualization or pagination for 100+ tasks

Bundle size	2	Production build under 200kb gzipped

6. Documentation (5 points)

Criteria	Points	Description
Storybook stories completeness	3	All required stories implemented
README quality	2	Clear setup and architecture explanation

Bonus Points (up to +15)

- Interactive story controls (+3)
- Dark mode implementation (+3)
- Additional stories beyond requirements (+3)
- Accessibility story with keyboard demo (+3)
- Performance optimization documentation (+3)

Total Possible: 100 points (115 with bonus)

Passing Score: 70 points

Disqualification Criteria

Your submission will be **immediately rejected** if any of these violations are found:

1. **Use of forbidden libraries:**
    - Component libraries (Radix, Shadcn, MUI, Ant Design, etc.)
    - Pre-built kanban/drag-drop components
    - CSS-in-JS solutions (styled-components, emotion)
2. **AI-generated UI:**
    - Code generated by Lovable, Bolt, v0, Locofy, etc.
    - Entire components copy-pasted from ChatGPT/Claude/Copilot without understanding
    - (Note: Using AI for debugging or learning is acceptable, but the final code must be your own)
3. **Plagiarism:**
    - Code copied from tutorials, Stack Overflow, or GitHub without attribution
    - Using paid templates or starter kits
4. **Non-functional submission:**
    - Cannot be run locally
    - Missing core required features
    - Critical bugs that crash the application
    - No deployed Storybook link provided
    - Storybook doesn't build or has critical errors
5. **Incomplete submission:**

  - No README
  - No source code
  - Repository is private and access not granted

Tips for Success

Before You Start

1. **Set up Storybook first:**

   ```
   npx storybook@latest init
   ```

   Configure it before building components

2. **Study reference implementations:**

   - Trello, Linear, Asana
   - Check their Storybook/design systems if available

3. **Plan your stories:**

   - List all variants you'll need to showcase
   - Think about interactive controls

During Development

1. **Build component + story together:**

   - Create component → Create story → Test → Refine
   - Don't wait until the end to add stories

2. **Implement features incrementally:**

   - Days 1-2: Basic layout + Default story
   - Days 3-4: Drag-and-drop + Interactive story
   - Days 5-6: Task management + Edge case stories
   - Days 7-8: Polish, accessibility, deployment

3. **Use Storybook for testing:**

   - Test all edge cases through stories
   - Verify responsive behavior in Storybook
   - Check accessibility with Storybook a11y addon

Common Pitfalls to Avoid

Not using Storybook properly: Stories should be interactive and demonstrate all features
Building full app instead of component: Focus on the reusable component, not a complete application
Ignoring story variants: Create stories for all important states and edge cases
Poor Storybook organization: Use clear story names and descriptions
Accessibility as afterthought: Build it in from the start, showcase it in stories

---

Learning Resources

Storybook

- [Storybook Documentation]
- [Component Story Format]
- [Storybook Addons]

Drag and Drop

- [MDN - HTML Drag and Drop API]
- [@dnd-kit documentation] (if using)

Accessibility

- [WCAG 2.1 Guidelines]
- [Storybook Accessibility Addon]

FAQ

Q: Do I need to build a full application or just the component?
A: Just the component! Build it as a reusable component library item, demonstrated through Storybook.

Q: Can I use Storybook addons?
A: Yes! Use addons like @storybook/addon-a11y, @storybook/addon-controls, etc.

Q: How many stories do I need?
A: Minimum 5 required stories, but more is better to showcase all features and states.

Q: Can I deploy my Storybook to Chromatic?
A: Yes, Chromatic is the recommended platform for Storybook deployment.

Q: Should I focus more on the component or the stories?
A: Both are equally important. A great component with poor documentation fails the assignment.

Final Checklist Before Submission

Functionality

- [ ] Drag and drop works smoothly
- [ ] Task creation/editing functional
- [ ] All interactive features work
- [ ] No console errors

Storybook

- [ ] All 5+ required stories implemented
- [ ] Stories are interactive and demonstrate features
- [ ] Storybook builds without errors
- [ ] Deployed and accessible online

Code Quality

- [ ] TypeScript strict mode enabled

- [ ] No `any` types used
- [ ] Components properly structured
- [ ] Follows project structure requirements

Accessibility

- [ ] Keyboard navigation works
- [ ] ARIA labels present
- [ ] Focus indicators visible

Documentation

- [ ] README complete with Storybook link
- [ ] Clear setup instructions
- [ ] Architecture explained

Repository

- [ ] Repository is public
- [ ] 5+ meaningful commits
- [ ] No node_modules committed

Submission Process

Step 1: Complete Your Component

Ensure all features work and Storybook stories are comprehensive.

Step 2: Deploy Storybook

Deploy to Chromatic, Vercel, or Netlify and verify all stories work.

Step 3: Prepare Repository

- Repository name: `kanban-component-[yourname]`
- Make repository **public**
- Ensure README has deployed Storybook link

Step 4: Test Locally

```
git clone [your-repo-url]
cd kanban-component-[yourname]
npm install
npm run storybook
```

Step 5: Submit via Internshala

Submit through Internshala portal:

- GitHub repository link
- Deployed Storybook link
- Brief description (2-3 paragraphs) of your implementation approach

---

🍀 Good Luck!

This assignment tests your ability to build production-ready components with excellent documentation. Focus on:

- **Component quality** over feature quantity
- **Storybook documentation** that makes your component easy to understand
- **Clean code** that others can read and maintain

Remember: A well-documented, polished component with great stories beats a feature-complete component with poor documentation.

We're rooting for you! 🚀

Appendix A: Sample Data Structure

Kanban View Sample Data

```
const sampleColumns: KanbanColumn[] = [
  { id: 'todo', title: 'To Do', color: '#6b7280', taskIds: ['task-1', 'task-2'],
maxTasks: 10 },
  { id: 'in-progress', title: 'In Progress', color: '#3b82f6', taskIds: ['task-3'],
maxTasks: 5 },
  { id: 'review', title: 'Review', color: '#f59e0b', taskIds: [], maxTasks: 3 },
  { id: 'done', title: 'Done', color: '#10b981', taskIds: ['task-4', 'task-5'] },
];

const sampleTasks: Record<string, KanbanTask> = {
  'task-1': {
    id: 'task-1',
    title: 'Implement drag and drop',
    description: 'Add D&D functionality to kanban cards',
    status: 'todo',
    priority: 'high',
    assignee: 'John Doe',
    tags: ['frontend', 'feature'],
    createdAt: new Date(2024, 0, 10),
    dueDate: new Date(2024, 0, 20),
  },
  'task-2': {
    id: 'task-2',
    title: 'Design task modal',
    description: 'Create modal for editing task details',
    status: 'todo',
    priority: 'medium',
    assignee: 'Jane Smith',
    tags: ['design', 'ui'],
    createdAt: new Date(2024, 0, 11),
    dueDate: new Date(2024, 0, 18),
  },
  'task-3': {
    id: 'task-3',
    title: 'Setup TypeScript',
    status: 'in-progress',
    priority: 'urgent',
    assignee: 'John Doe',
    tags: ['setup', 'typescript'],
    createdAt: new Date(2024, 0, 9),
  },
  'task-4': {
    id: 'task-4',
    title: 'Create project structure',
    description: 'Setup folder structure and initial files',
    status: 'done',
    priority: 'low',
    assignee: 'Jane Smith',
    tags: ['setup'],
    createdAt: new Date(2024, 0, 8),
    dueDate: new Date(2024, 0, 9),
  },
  'task-5': {
    id: 'task-5',
    title: 'Install dependencies',
    status: 'done',
    priority: 'low',
    assignee: 'John Doe',
    tags: ['setup'],
    createdAt: new Date(2024, 0, 8),
  },
};
```

Appendix B: Tailwind Configuration Template

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        primary: {
          50: '#f0f9ff',
          100: '#e0f2fe',
          200: '#bae6fd',
          300: '#7dd3fc',
          400: '#38bdf8',
          500: '#0ea5e9',
          600: '#0284c7',
          700: '#0369a1',
          800: '#075985',
          900: '#0c4a6e',
        },
        neutral: {
          50: '#fafafa',
          100: '#f4f4f5',
          200: '#e4e4e7',
          300: '#d4d4d8',
          400: '#a1a1aa',
          500: '#71717a',
          600: '#52525b',
          700: '#3f3f46',
          800: '#27272a',
          900: '#18181b',
        },
        success: {
          50: '#f0fdf4',
          500: '#10b981',
          700: '#047857',
        },
        warning: {
          50: '#fffbeb',
          500: '#f59e0b',
          700: '#b45309',
        },
        error: {
          50: '#fef2f2',
          500: '#ef4444',
          700: '#b91c1c',
        },
      },
      fontFamily: {
        sans: ['Inter', 'system-ui', 'sans-serif'],
        mono: ['Fira Code', 'monospace'],
      },
      spacing: {
        18: '4.5rem',
        88: '22rem',
        112: '28rem',
        128: '32rem',
      },
      borderRadius: {
        '4xl': '2rem',
      },
      boxShadow: {
        'card': '0 1px 3px 0 rgb(0 0 0 / 0.1), 0 1px 2px -1px rgb(0 0 0 / 0.1)',
        'card-hover': '0 10px 15px -3px rgb(0 0 0 / 0.1), 0 4px 6px -4px rgb(0 0 0 /
0.1)',
        'modal': '0 20px 25px -5px rgb(0 0 0 / 0.1), 0 8px 10px -6px rgb(0 0 0 /
0.1)',
      },
      animation: {
        'fade-in': 'fadeIn 0.2s ease-in-out',
        'slide-up': 'slideUp 0.3s ease-out',
        'slide-down': 'slideDown 0.3s ease-out',
      },
      keyframes: {
        fadeIn: {
          '0%': { opacity: '0' },
          '100%': { opacity: '1' },
        },
        slideUp: {
          '0%': { transform: 'translateY(10px)', opacity: '0' },
          '100%': { transform: 'translateY(0)', opacity: '1' },
        },
        slideDown: {
          '0%': { transform: 'translateY(-10px)', opacity: '0' },
          '100%': { transform: 'translateY(0)', opacity: '1' },
        },
      },
    },
  },
  plugins: [],
}
```

End of Kanban Board Assignment Document

This assignment is part of Design System Component Library's hiring process. All submitted code remains your intellectual property.