Low Level Design Document

This Low Level Design (LLD) document outlines the implementation details for **FlexGrid - Responsive Layout Playground**, an interactive React tool for visualizing and experimenting with CSS Flexbox and Grid layouts using Tailwind CSS.

---

1. System Components

Component	Description	Key Responsibilities
App	Root component	Routing, state management
LayoutPlayground	Main UI for layout experimentation	Renders controls and preview area
ControlPanel	UI controls for layout properties	Updates layout state
PreviewArea	Visualizes the layout with current settings	Applies dynamic Tailwind classes
LayoutItem	Represents a single item in the layout	Renders draggable/resizable item
Utils	Helper functions	Class generation, validation

---

2. Class/Interface Overview

Name	Type	Relationships	Key Methods/Attributes
`LayoutConfig`	Interface	Used by LayoutPlayground	`type`, `direction`, `gap`, `items`
`App`	Function	Parent of all components	-
`LayoutPlayground`	Function	Uses ControlPanel, PreviewArea	`handleConfigChange`, `config`
`ControlPanel`	Function	Child of LayoutPlayground	`onChange`, `config`
`PreviewArea`	Function	Child of LayoutPlayground	`config`, `items`
`LayoutItem`	Function	Child of PreviewArea	`itemConfig`

Key Interface Example:

```
interface LayoutConfig {
  type: 'flex' | 'grid';
  direction?: 'row' | 'column';
  gap?: string;
  items: LayoutItemConfig[];
}
interface LayoutItemConfig {
  id: string;
  content: string;
```

```
  style?: object;
}
```

---

3. Data Structure Overview

Model	Fields	Purpose
LayoutConfig	type, direction, gap, items	Stores current layout settings
LayoutItemConfig	id, content, style	Stores item-specific settings

---

4. Algorithms / Logic

Dynamic Tailwind Class Generation:

```
function getContainerClasses(config) {
  if (config.type === 'flex') {
    return `flex flex-${config.direction} gap-${config.gap}`;
  } else {
    return `grid grid-cols-${config.items.length} gap-${config.gap}`;
  }
}
```

Layout Update Flow:

1. User changes a control (e.g., direction, gap).
2. `ControlPanel` calls `onChange`.
3. `LayoutPlayground` updates `config` state.
4. `PreviewArea` re-renders with new Tailwind classes.

---

5. Error Handling

Scenario	Handling Approach
Invalid config values	Fallback to defaults, show warning
Unsupported Tailwind class	Ignore or display error message
Empty items array	Show placeholder in PreviewArea
JS runtime errors	Error boundary at App level

End of Document