Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **Gridify - CSS Grid Visualizer**, an interactive React and Tailwind CSS-based tool for designing and previewing CSS Grid layouts. The document details system components, interfaces, data structures, core logic, and error handling to guide development.

---

1. System Components

Component	Description	Key Responsibilities
GridEditor	Main UI for grid configuration	Render grid, handle user input
GridPreview	Visualizes the current grid layout	Display grid as per config
GridControls	UI controls for grid properties	Update grid rows/columns/gaps
GridItemManager	Manages grid items (add/remove/move/resize)	CRUD operations on grid items
CSSCodeGenerator	Generates CSS code from current grid state	Output CSS for user copy/export

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`GridConfig`	Grid configuration model	`rows` , `columns` , `gap` , `items`
`GridItem`	Represents a grid item	`id` , `rowStart` , `colStart` , `rowSpan` , `colSpan`
`GridEditorProps`	Props for GridEditor	`config: GridConfig` , `onChange`

Relationships:

- `GridEditor` holds `GridConfig` state and passes to `GridPreview` , `GridControls` , and `GridItemManager` .
- `GridItemManager` updates `GridConfig.items` .

Key Methods (Sample):

```
// GridEditor
function handleConfigChange(newConfig: GridConfig): void

// GridItemManager
function addItem(): void
function updateItem(id: string, updates: Partial<GridItem>): void
function removeItem(id: string): void
```

```
// CSSCodeGenerator
function generateCSS(config: GridConfig): string
```

---

3. Data Structure Overview

Model	Fields
GridConfig	`rows: number`, `columns: number`, `gap: number`, `items: GridItem[]`
GridItem	`id: string`, `rowStart: number`, `colStart: number`, `rowSpan: number`, `colSpan: number`

---

4. Algorithms/Logic

Grid Update Flow (Pseudocode):

```
onUserChangeGridProperty(property, value):
    config[property] = value
    updateGridPreview(config)
    updateCSSCode(generateCSS(config))
```

Grid Item Manipulation:

```
addItem():
    newItem = defaultGridItem()
    config.items.push(newItem)
    updateGridPreview(config)
```

CSS Generation:

```
generateCSS(config):
    return `
      display: grid;
      grid-template-rows: repeat(${config.rows}, 1fr);
      grid-template-columns: repeat(${config.columns}, 1fr);
      gap: ${config.gap}px;
    `
```

---

5. Error Handling

Scenario	Handling Approach
Invalid grid size (rows/columns)	Validate input, show error message
Overlapping grid items	Prevent placement, highlight conflict

Out-of-bounds item placement	Restrict movement, notify user
CSS generation failure	Fallback to default CSS, log error
UI input errors	Inline validation, disable actions

End of Document