.app {

display: grid;

grid-template-columns: 280px 1fr;

grid-template-rows: 64px 1fr;

grid-template-areas:

"topbar topbar"

"sidebar main";

min-height: 100vh;

}

1. display: grid;

Turns .app into a grid container.

All direct children of .app will be placed on this grid.

2. grid-template-columns: 280px 1fr;

Defines 2 columns:

First column = 280px wide (fixed).

Second column = 1fr (1 fraction unit = takes up remaining space).

👉 Typically: left column = sidebar, right column = main content.

3. grid-template-rows: 64px 1fr;

Defines 2 rows:

First row = 64px tall (fixed, probably a topbar/header).

Second row = 1fr (fills the rest of the page).

4. grid-template-areas:

css

Copy code

"topbar topbar"

"sidebar main";

Names regions of the grid for easier placement.

First row: "topbar topbar" → the topbar spans both columns.

Second row: "sidebar main" → sidebar on the left, main content on the right.

So the grid looks like this:

diff

Copy code

+-------------------------------+

| topbar | 64px tall

+----------------+--------------+

| sidebar | main | fills rest

| (280px) | (1fr) |

+----------------+--------------+

5. min-height: 100vh;

Ensures .app is at least the full height of the viewport (100vh = 100% of screen height).

This makes the layout stretch to the full page, even if there’s little content.

🎯 How It Works in Practice

If you had HTML like this:

html

Copy code

<div class="app">

<header class="topbar">Topbar</header>

<nav class="sidebar">Sidebar</nav>

<main class="main">Main content</main>

</div>

And styled:

css

Copy code

.topbar { grid-area: topbar; background: #13315C; color: white; }

.sidebar { grid-area: sidebar; background: #F2F4F8; }

.main { grid-area: main; background: #FFFFFF; }

👉 You’d get a dashboard layout:

Header across the top,

Sidebar on the left,

Main content on the right.

So in short: this .app rule sets up a classic app shell layout with topbar + sidebar + main using CSS Grid.