PROJECT ABSTRACT

Window Layout Snap Manager for Multitasking

A window-manager utility that snaps windows into halves, thirds and quadrants with drag zones and animation, applies one-click multi-window presets, auto-layout rules per application, global hotkeys and saved task layouts. Built to order with source, design documentation, web demo and complete viva kit.

01. Title

Window Layout Snap Manager for Multitasking

02. Introduction / Background

Multitasking on a single screen usually means alt-tabbing between overlapping windows or manually resizing each one into place — a small but constant tax on focus, especially during coding, research or viva preparation. Operating systems ship basic snapping, but it is limited: no named multi-window presets, no per-application rules, no saved layouts for different tasks. This utility builds the missing layer: drag-to-snap zones at edges, corners and thirds; named presets (Coding, Research, Triple-column, Focus, Review) that arrange every open window at once; auto-layout rules pinning application types to zones; hotkeys for everything; and saved layouts per task.

03. Problem Statement

Built-in OS snapping handles one window at a time with fixed halves, leaving students with repetitive manual window arrangement. This project provides the missing organization layer — multi-window presets, per-app auto-layout rules and saved task layouts — demonstrating OS-level programming, event handling, geometry and UX design through a tool the student will genuinely use daily.

04. Objectives

- Implement drag-to-snap zones for halves, thirds, quadrants and maximize with live zone highlighting and snap animation.
- Provide 5 built-in multi-window presets (Coding, Research, Triple-column, Focus, Review) bound to hotkeys.
- Build an auto-layout rule engine matching applications by name/class to zones on launch.
- Support full keyboard operation: snap, maximize, apply presets, save layouts.
- Persist named task layouts to disk (JSON) and restore them on demand.
- Deliver the design document, working web demo and full viva kit (report, PPT, Q&A;).

05. Proposed Methodology

The utility hooks window-drag events from the OS window manager. While dragging, the cursor position is tested against a snap-zone map (halves, thirds, quadrants, maximize strip); the matching zone highlights, and on release the window's geometry is animated to the zone rectangle and tagged as snapped. Presets are stored as zone assignments per window role; applying one iterates over open windows and snaps each to its zone. Auto-layout rules match new windows by application name or window class against a rule table. Saved layouts serialize window identifiers and zone assignments to JSON. Global hotkeys are registered so snapping works regardless of focus.

06. System Architecture / Working

An event layer listens for drag, focus and new-window events from the OS. A geometry engine maps cursor position to snap zones and animates snapped geometry. The preset engine assigns zones per window role and applies them in one action. The rule engine auto-snaps new windows on launch. The layout store persists named arrangements, and a global hotkey layer (Win+arrows, Win+1…5, Win+Shift+S) drives everything mouse-free. A toggleable zone overlay shows every snap zone and label so the layout language stays discoverable. The web demo replicates this desktop experience in the browser.

07. Hardware / Software Requirements

- Desktop OS: Windows or X11 Linux in the shipped build (Wayland/macOS are future scope)
- Python 3 with OS window-manager API access (Win32 / X11)
- Global hotkey registration support
- JSON for layout serialization
- Modern web browser for the single-file demo (simulated desktop)
- Git

08. Expected Outcomes

- A working utility that snaps, presets, rules and restores window layouts.
- Animated drag-to-snap with live zone highlighting and a discoverable zone overlay.
- Per-application auto-layout rules and named, restorable task layouts.
- A design document covering the zone map, rule engine, hotkey table and layout format.
- A full viva kit: report on window-manager concepts, PPT and Q&A; on event hooks, geometry and UX.

09. Applications

- Coding, research and study workflows on single screens.
- Teaching OS-level programming: events, geometry, global hotkeys.
- Demonstrating UX design for productivity tools.

10. Future Scope

- Wayland and macOS platform support.
- Validated multi-monitor zone mapping.
- Layout sharing and team preset packs.
- Window session restore after reboot with app relaunch.