# Software Design Document (SDD)

## Project: Agentic Qtile

### 1. System Architecture

Agentic Qtile extends the Python-based Qtile core with three primary modules:

1. **The Agent Bridge (IPC):** A socket-based server listening for GAD-formatted instructions.
2. **The Cutout Engine:** A modification to libqtile.layout that supports "Floating Gaps" and "Semantic Slots."
3. **The Decorator Overlay:** A custom implementation of the Window class that uses Cairo to draw agent metadata.

### 2. Implementation Details: Tile Border Inlays

We will subclass libqtile.backend.base.Window to intercept the paint method.

* **Data Structure:** A window-specific metadata dictionary: {"agent\_id": str, "confidence": float, "status": str}.
* **Rendering Logic:**
  + Calculate border rectangle.
  + Initialize Cairo context.
  + Draw background color based on status (e.g., Amber for "Running", Green for "Success").
  + Overlay text: f"Conf: {confidence:.2f}".

### 3. Ralph Wiggin Loop Integration

The WM acts as the "Stop Hook" for the Ralph Wiggin Protocol.

* When a sandboxed window attempts to close, the WM checks the verifyCompletion status via the agent.
* If incomplete, the WM re-initializes the window context or keeps the "Cutout" open for the next iteration.

### 4. Component Diagram

[ User Intent ] -> [ GAD Agent ] <-> [ Ralph Wiggin Loop ]  
 |  
 [ Agentic Qtile IPC ]  
 / | \  
 [Layout Engine] [Cairo Overlay] [Sandbox Manager]