

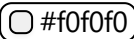




LLM Development Assistant GUI - Visual Layout

Window Structure (1200x800)

[Controls]	[Output]
Project Path:	
[Browse] {	
[Initialize Project] "project_path": "/home/user/project",	
"context": "# Game Maker Studio 2..."	
"code_structure": {	
"files": [
Task Description: "llm_dev_assistant/main.py",	
"llm_dev_assistant/parser.py"	
Write a function to],	
parse JSON configuration "functions": {...},	
files and validate them "classes": {...}	
},	
"status": "initialized"	
}	
File Path (optional):	
[Browse]	
[Request Code] [Verify...]	
[Implement...] [Verify T.]	
[Run Tests] [Plan Next]	
[Logs]	
Workflow Management: Filter: [ALL ▼] [Clear Logs]	
[Save Workflow][Load W...]	
[14:23:15] [INFO] GUI started	
[14:23:16] [INFO] Initializing proj..	
[14:23:16] [DEBUG] Parsing directory	
[14:23:17] [INFO] Found 12 scripts	
[14:23:17] [WARNING] Missing descrip.	
[14:23:18] [INFO] Project initialized	
[14:23:45] [INFO] Requesting code...	
[14:23:46] [DEBUG] Sending to LLM	
[14:23:52] [INFO] Code generated	

Visual Style Guide

Colors

- **Background:**  (#f0f0f0) (Light gray)
- **Panels:** White with subtle border
- **Text:** Black on white
- **Log Levels:**
 - **DEBUG:** Gray ( #808080)
 - **INFO:** Black ( #000000)
 - **WARNING:** Orange ( #FFA500)
 - **ERROR:** Red ( #FF0000)

Fonts

- **Default:** System default (Segoe UI on Windows, SF Pro on Mac, Ubuntu on Linux)
- **Monospace:** For code/output display (Consolas, Monaco, or system monospace)

Layout Characteristics

- **Left Panel (Controls):** Fixed width ~350px
- **Right Panels:** Flexible, split 60/40 between Output and Logs
- **Minimal styling:** Focus on functionality over aesthetics
- **Clear sections:** Separated by borders and labels
- **Raw look:** Simple tkinter widgets without heavy theming

Interactive Elements

- **Buttons:** Standard tkinter buttons, rectangular, no rounded corners
- **Text Fields:** Simple white boxes with thin borders
- **Dropdowns:** Native OS style
- **Progress Bar:** Indeterminate style during operations

Dialog Windows

Code Verification Dialog (800x600)

[Verify Code Implementation]

Original Code:

def parse_json(file_path):
 with open(file_path, 'r') as f:
 return json.load(f)

New Code:

def parse_json(file_path):
 try:
 with open(file_path, 'r') as f:
 data = json.load(f)
 validate_json(data)
 return data
 except Exception as e:
 raise ValueError(f"Error: {e}")

Requirements:

Add error handling and validation

[Verify] [Cancel]

Status Messages

- **Ready:** Default state
- **Initializing project...:** With progress bar
- **Requesting code implementation...:** With progress bar
- **Operation completed successfully:** Green text
- **Error: [message]:** Red text

Key Features Highlighted

1. **Simple, functional design** - No unnecessary decorations
2. **Clear information hierarchy** - Important info is prominent
3. **Responsive feedback** - Progress bars and status updates
4. **Accessible** - Standard controls, keyboard navigation

5. **Efficient layout** - Everything accessible without scrolling