**Project Title:** Modular DSP Toolkit (App + REAPER Plugin)

**Overview:** The Modular DSP Toolkit is a two-part system consisting of:

1. A standalone JUCE-based desktop application for creating and auditioning DSP effect chains.
2. A lightweight REAPER plugin that loads and applies preconfigured DSP chains with minimal controls.

**Part 1: DSP Designer Application (JUCE App)**

**Purpose:** To enable sound designers and musicians to build, test, and refine custom DSP effect chains by importing audio (e.g., WAV files) and applying stacked DSP modules.

**Key Features:**

* **Toolbox Panel:** List of available DSP effect modules (icons/buttons) on the left.
* **Signal Flow Grid:** Columns representing sequential signal stages from left to right.
  + Each column is a stack of DSP modules (top-down processing order).
  + Signal flows through columns left-to-right.
* **Drag-and-Drop UI:**
  + Users can drag modules from the toolbox into columns.
  + Modules can be rearranged or duplicated.
* **Effect Inspector Panel (Bottom):**
  + Contextual control panel showing parameters of the currently selected DSP module.
  + Includes bypass, solo, and remove buttons.
* **Audio Import:** Load WAV files for testing effect chains.
* **Playback Engine:** Basic transport controls (Play, Pause, Stop) and waveform view.
* **Save Configuration:** Export chain as JSON (or XML) config file.
* **Organized Preset System:** Named chains saved in a known user-accessible folder.
* **Config Metadata:** Each config may contain:
  + Unique name
  + Description
  + List of DSP modules in signal order, with parameters
  + Optional macro mappings

**Optional Future Features:**

* Modulation routing system (LFOs, envelope followers)
* Real-time waveform or spectrum view per module
* Randomization or intelligent generator for effect chains

**Part 2: REAPER Plugin (JUCE-based VST/AU)**

**Purpose:** To allow use of preconfigured DSP chains in a DAW environment, without exposing full design complexity.

**Key Features:**

* **Load Configuration File:**
  + User browses or selects from a list of saved config files (from known folder).
  + Plugin immediately loads and applies the defined DSP chain.
* **Minimal Controls:**
  + Wet/Dry Mix
  + Input Volume
  + Output Volume
  + Optional: 1–2 Macro Controls (if defined in config)
* **UI Design:**
  + Dropdown menu or file browser for config selection
  + Basic visual indicator of chain (e.g., mini diagram or module icons)
  + Status area showing chain name, description, and loading confirmation
* **Lightweight Runtime:**
  + Efficient DSP pipeline matching the config file
  + Only load what’s defined; ignore unsupported modules gracefully

**Bonus Features:**

* Remember last used config per instance
* “Recent Files” quick access menu
* “Open in DSP App” button to launch the full designer with the selected config (optional cross-launching)

**Example DSP Modules (Initial Set):**

* Transient Shaper
* Pitch Shifter
* Bitcrusher
* Granular Delay
* Multi-tap Delay
* Spectral Blur / Freeze
* ADSR Envelope Shaper
* Formant Shifter
* Tape Saturation
* Comb Filter / Modal Resonator

**Configuration File Format (Conceptual):**

* Name: "WarmCrunchChain.json"
* Description: "Punchy transients + tape warmth + light delay"
* Modules (in signal order):
  + { type: "TransientShaper", params: { attack: 0.8, sustain: 0.4 } }
  + { type: "TapeSaturation", params: { drive: 0.5 } }
  + { type: "Delay", params: { time: 0.3, feedback: 0.2 } }
* Macros:
  + { label: "Warmth", targets: [ { module: 2, param: "drive" } ] }

**Design Goals:**

* User-friendly, modular, intuitive interface
* Clear visual signal flow
* Balance between creative freedom (App) and production efficiency (Plugin)
* Reusability and configurability across projects and tracks

**Next Steps:**

* Define exact DSP module specs and parameter ranges
* Build base UI framework for app and plugin
* Implement JSON schema for config files
* Create prototype with 2–3 working modules