**The Two Core Concepts Explained**

**1. Live Settings**

* **Current Situation:** This is effectively already in place in the current project. In the current profiles tab it is possible to change live but I want to remove the current profiles system and build a better one!
* **Purpose:** The "Live Settings" represent the configuration that is *currently active* in the plugin for the car you are driving. Their main purpose is to allow for immediate, in-the-moment adjustments during a race or practice session against the profile for the active car.
* **UI Location:** The existing tabs—**"Launch Settings," "Fuel,"** and **"Dash Control"**—will be the exclusive home for editing these Live Settings. This is already implemented in the project.
* **Behavior:** Changes made on these tabs are for the active car for the current session. When a new car is loaded in a new session, the Live Settings are completely overwritten by that car's saved profile, ensuring a fresh, predictable start every time. Changes made during live session should be retained after in that car’s profile.

**2. Car Profiles**

* **Current Situation:** This is effectively already in place in the current project. In the current profiles tab it is possible to change live but I want to remove the current profiles system and build a better one!
* **Purpose:** "Car Profiles" are the permanent, saved library of detailed setups for each of your cars. This is where you will build and store your baseline configurations.
* **UI Location:** All management and editing of these permanent profiles will happen **exclusively on the new, reworked "Profiles" tab**.
* **Data Storage:** All Car Profiles will be stored together in a single collection and saved to a dedicated, human-readable JSON file (e.g., LalaLaunch\_CarProfiles.json) in the SimHub PluginData folder.

**The Unified CarProfile Data Class**

To make this system work, we need a single, authoritative data structure to hold all settings that can be different for each car.

* **Purpose:** This class will be the blueprint for a car profile, containing every possible car-specific setting from all modules. Maybe in it’s own file?
* **Contents:**
  + **General Properties:**
    - ProfileName: A unique name for the profile (e.g., "Porsche 911 GT3 R").
  + **Launch Control Properties:**
    - All 12 settings currently on the "Launch Settings" tab, including TargetRpm, SlipThreshold, RpmTolerance, MaxSpeed, MinSpeed, PreClutchBite, etc.
  + **Fuel & Pit Properties:**
    - TankSize: The car's maximum fuel capacity.
    - RefuelRate: A user-configurable override for the car's refueling speed.
  + **Dash Display Properties:**
    - All car-specific settings from the "Dash Control" tab, such as RejoinWarningLingerTime, SpinYawRateThreshold, etc.

**Detailed Design of the New "Profiles" Tab**

This tab will be transformed into an all-in-one **"Car Profile Editor"** with a two-panel layout.

**Left Panel: Profile Management**

This panel is for managing your library of profiles.

* **Profile List:** A list box showing the names of all saved car profiles. Selecting a profile here will load its data into the editor on the right.
* **Management Buttons:**
  + **New:** Prompts for a name, then creates a new, blank profile with default values and adds it to the list, ready for editing.
  + **Clone:** Makes an exact copy of the currently selected profile, prompts for a new name or existing name selection, and adds it to the list.
  + **Delete:** Asks for confirmation, then permanently removes the selected profile from the library.

**Right Panel: Profile Editor**

This panel is where you edit the settings for the profile you've selected on the left. It will be disabled if no profile is selected.

* **Header:** Displays the name of the profile currently being edited.
* **Settings TabControl:** To keep the many settings organized, this panel will contain a set of sub-tabs:
  + **"Launch Control" Tab:** Will contain all 12 input controls (sliders, toggles) for the launch settings.
  + **"Fuel & Pit" Tab:** Will contain the input controls for TankSize and RefuelRate.
  + **"Dash Display" Tab:** Will contain all the toggles and sliders for the car-specific dash preferences.
* **Action Buttons:**
  + **Save Changes:** Saves any modifications made in the editor to the master library of profiles and writes them to the JSON file.
  + **Cancel/Revert:** Discards any unsaved changes and reloads the selected profile's original data.
  + **Apply to Live Session:** This is the critical "bridge" between the editor and the live session. Clicking this button will instantly copy all the settings from the profile you are editing into the "Live Settings," overriding any temporary tweaks you may have made on the other tabs.

**Data Flow Summary**

1. **On Plugin Startup:** The plugin loads both the last-used "Live Settings" and the entire library of "Car Profiles" from their respective files.
2. **When a Car is Loaded in the Sim:** The plugin gets the car name, finds the matching profile in its library, and **copies** that profile's data into the "Live Settings" object. If no specific profile is found, it uses a "Default Profile."
3. **During a Session:** The "Launch Settings" and other tabs modify the "Live Settings" object for tweaks. The "Profiles" tab modifies the permanent, saved profiles completely independently.
4. **On Plugin Shutdown:** The plugin saves both the "Live Settings" and the entire library of "Car Profiles" to their files.

This design provides the clear separation and powerful functionality you envisioned. It eliminates the current confusion and creates a robust, scalable foundation for the future.

**DONE SO FAR**

A start had been made before the AI got a serious error. The following files for the new system have been created and added to the project already but need integrating with other parts:

CarProfiles.cs

ProfilesManagerViewModel.cs

ProfilesManagerView.xaml – Original that was already updated.

**CLEAN UP**

The project already has an attempt at building the profiles system but it was not great. Therefore, there will have to be code removal and careful clean out of the old system before starting. Care must be taken not to disturb other parts of code, shared variables etc in the process. This is why the new code should be housed in it’s own file. Below is the current construct of the project:

A screenshot of a computer program

AI-generated content may be incorrect.