Startup Logo: Optify Replacement

Background

Replacing the default "TAB" splash graphic with the Optify logo touches both the LVGL asset pipeline and the startup animation app. This note captures the exact tooling setup, file changes, and build steps that were used so the update can be reproduced or adjusted later.

Source Assets

- **Original image**: `app/assets/images/logo_tab.c` (LVGL v8 descriptor, 180×80, RGB565, generated for the original TAB artwork).
- **New artwork**: `optify_logo_transparent.png` (external PNG provided by the client). After conversion, the descriptor `app/assets/images/optify_logo_transparent.c` reports 129×56 pixels in RGB565 format.

LVGL Image Converter Settings

1. Open https://lvgl.io/tools/imageconverter (LVGL v9 UI).
2. Upload `optify_logo_transparent.png`.
3. Set **Color format** to `True color` → `RGB565` (matches the project's existing splash assets).
4. Uncheck **Dither** and **Output alpha byte order swap**; leave **Bpp** at the default for RGB565.
5. Enable **Advanced settings** → **Use custom color format** and select **Compatible with LVGL v8** (disables the v9-specific header fields and produces `lv_image_dsc_t`).
6. Choose **Output** → **C array** and click **Convert**.
7. Save the generated file as `app/assets/images/optify_logo_transparent.c` inside the repository.

> **Tip**: If a v9-style descriptor slips through (fields like `always_zero` or `data_size = ... * 4`), adjust the header manually to mirror the layout used in `logo_tab.c`.

Code Integration

- Declare the asset so LVGL components can reference it:
  - Edit `app/assets/assets.h` and add `LV_IMG_DECLARE(optify_logo_transparent);` alongside the other image declarations.
- Point the startup animation at the new descriptor:
  - In `app/apps/app_startup_anim/app_startup_anim.cpp`, change `_logo_tab->setSrc(&logo_tab);` to `_logo_tab->setSrc(&optify_logo_transparent);`.
  - No other animation logic changes were required; the existing positioning and timing work with the new dimensions.

Build System Notes

- The ESP-IDF/CMake project uses globbing to collect asset sources. When a new `*.c` image is introduced, clear CMake's state so it notices the file:
  - From the repository root on Windows PowerShell:

    ```
    Remove-Item build/CMakeCache.txt
    ```

    (or run `idf.py fullclean` if you prefer the IDF helper.)
- Re-configure and compile through the command-line ESP-IDF workflow (per the project README, not the VS Code GUI):

  ```
  idf.py reconfigure
  idf.py build
  ```

Flash & Verification

- Flash the firmware to the device using the standard CLI steps:

  ```
  idf.py -p <COM_PORT> flash monitor
  ```

- Watch the startup animation: the Optify logo should fade/slide in where the TAB artwork used to appear.

Files Touched

- `app/assets/assets.h`
- `app/assets/images/optify_logo_transparent.c`
- `app/apps/app_startup_anim/app_startup_anim.cpp`

Keep this checklist with the source so future updates to branding can follow the same pipeline with minimal friction.