FocusLock for Windows — Rust Design Plan (Rust)

0) Modern UI Recommendation

Primary: **Tauri + React + Fluent 2 (Microsoft)** - **Why**: Native-looking Windows 11 visuals (Fluent 2), fast iteration with React, tiny Rust backend via Tauri, low memory (compared to Electron), easy theming and accessibility. - **Where it fits**: All app surfaces except the background enforcement service. Tray menu via Tauri plugin, Windows Toasts via Rust service.

Alternatives - **egui + winit** (pure Rust): fastest to ship, lightweight; look is less “Windows 11 native,” but good for dev tools. - **WinUI 3 (Windows App SDK) via Rust bindings**: most native, but steeper setup and slower iteration vs React.

---

1) Project Overview

Goal: Block selected apps for a user-chosen time window; provide a favorites bar, fast app search, and remaining-time notifications.
Target OS: Windows 10/11 (x64).
Primary language: Rust (stable).
UX perf targets: Instant search (<50ms common queries), idle CPU <1%, RAM <60MB (UI+service).

---

2) High-Level Architecture

- **UI App (Tauri + React + Fluent 2):** Configuration/control (start/stop session, search, favorites, status).
- **Core Service (Rust, background):** Process watcher, rule evaluation, enforcement, timers, notifications.
- **IPC:** Secure named pipes for UI ↔ service (request/response + event stream).
- **Persistence:** SQLite (settings, rules, favorites, logs).
- **Strict/Tamper mode:** Optional passphrase; restricts quitting during active session.

```
[UI App (Tauri)]  <—named pipe—>  [Core Service]
   |                                  |
   |                                  +— Process watcher (WMI/ETW)
   |                                  +— Block evaluator & enforcer
   |                                  +— Scheduler & countdown timer
   |                                  +— Toast notifier & tray updater
   |
   +— SQLite (settings DB)
```

---

3) Tech Stack

Rust crates
- Windows API: `windows`
- Async: `tokio`
- IPC: `interprocess` or `tokio::net::windows::named_pipe`
- DB: `rusqlite` (bundled) or `sqlx` (offline)
- Serde: `serde`, `serde_json`
- Fuzzy search: `fuzzy_matcher`
- Tray/Toasts: Tray via `tauri-plugin` or Win32; Toasts via WinRT (`windows::UI::Notifications`)
- UI (web): React + **Fluent UI (v9)** components; Tauri for shell

Process monitoring: WMI `Win32_ProcessStartTrace` or ETW (preferred); fallback polling (Toolhelp32Snapshot).

Enforcement: `OpenProcess`, `TerminateProcess`; soft mode uses minimize/overlay/refocus.

---

4) Core Service Design

4.1 Modules

- `scheduler`: sessions & timers
- `watcher`: process start events + fallback polling
- `evaluator`: rule matching
- `enforcer`: hard/soft actions
- `notifier`: toasts + tray
- `repo`: SQLite persistence
- `ipc_server`: named pipe server (UI ↔ service)
- `catalog`: indexing & search
- `appid`: identity resolution

4.2 Data Model (SQLite)

```sql
CREATE TABLE settings (
  key TEXT PRIMARY KEY,
  value TEXT NOT NULL
);

CREATE TABLE favorites (
  id INTEGER PRIMARY KEY,
  app_id TEXT NOT NULL,
  display_name TEXT NOT NULL,
  pinned_order INTEGER
);
```

```
CREATE TABLE block_rules (
  id INTEGER PRIMARY KEY,
  app_id TEXT NOT NULL,
  match_kind TEXT NOT NULL,   -- 'exe'|'package'|'lnk'|'path'|'regex'
  mode TEXT NOT NULL          -- 'hard'|'soft'
);

CREATE TABLE sessions (
  id INTEGER PRIMARY KEY,
  start_utc INTEGER NOT NULL,
  end_utc INTEGER NOT NULL,
  status TEXT NOT NULL        -- 'scheduled'|'running'|'completed'|'canceled'
);

CREATE TABLE usage_log (
  ts_utc INTEGER NOT NULL,
  event TEXT NOT NULL,        --
'blocked_launch'|'terminated'|'soft_block'|'session_start'|'session_end'
  app_id TEXT,
  details TEXT
);

CREATE INDEX idx_block_rules_app ON block_rules(app_id);
CREATE INDEX idx_favorites_order ON favorites(pinned_order);
```

Canonical App Identity (`app_id`)

```
enum AppIdentity {
  ExeName(String),        // "discord.exe"
  FullPath(String),       // "C:\\Program Files\\Discord\\Discord.exe"
  PackageFamily(String), // MSIX/Store PFN
  LnkTarget(String),      // resolved .lnk target
}
```

4.3 Matching & Enforcement

1) Resolve identity from PID (path, basename, PFN if Appx).
2) Match against `block_rules` (exe, path prefix, package, regex).
3) If session active & rule matches:
- `hard` → terminate;
- `soft` → minimize, show overlay/toast, refocus UI.
4) Log every action.

4.4 Scheduler & Countdown

- Single active session; immediate or scheduled start; duration in minutes.
- 1s tick; emits `RemainingTime { seconds }` events to UI & tray (IPC pub/sub).

4.5 Notifications

- Toasts: session start; 15/5/1-minute checkpoints; on block attempts ("Discord blocked: 32:14 remaining").
- Tray: tooltip shows mm:ss; menu for Start/Stop, Quick-add last blocked, Strict mode toggle.

---

5) UI App (Tauri + React + Fluent 2)

5.1 Screens & Features

- **Home / Session**: duration presets (25/45/60), custom input, Start button; big remaining timer; recent blocked list with "Add Rule".
- **Favorites bar** (left): pinned apps (icons, drag-reorder); toggle block mode (soft/hard) per app.
- **Search**: instant results from catalog (Start Menu, Registry, Appx, common Program Files paths); actions: Add to block list, Reveal in Explorer.
- **Rules**: table of rules (app, match kind, mode) with edit/delete.
- **Settings**: strict mode/passphrase, hotkeys, autostart service, notification cadence.

5.2 UX Details

- Fluent 2 components for Windows 11 look; dark/light mode sync with system.
- Icon extraction via Rust (`SHGetFileInfo`) + cache; surfaced to UI via Tauri command.
- Keyboard: `Ctrl+F` focuses search; `Enter` adds selected to rules.
- Global hotkeys: Register via service; UI reflects status.

---

6) Indexing & Search

- **Sources**: Start Menu shortcuts (system & user), Registry Uninstall keys, Appx packages, shallow scan of Program Files and `%LOCALAPPDATA%\\Programs`.
- **Catalog** (SQLite `apps` table): `app_id`, `display_name`, `exe_or_target`, `package_family`, `lnk_path`, `icon_cache_key`.
- **Search**: in-memory index on launch; fuzzy match top-N (<50ms target); return top 30.

---

7) IPC Contract (Rust Types)

```
// Requests UI -> Service
enum Request {
  Ping,
  StartSession { duration_secs: u32 },
  StopSession,
  AddRule { rule: BlockRule },
  RemoveRule { id: i64 },
  ListRules,
  AddFavorite { app: AppSummary },
  RemoveFavorite { id: i64 },
  ListFavorites,
  Search { query: String, limit: u32 },
  GetStatus,
  SetStrictMode { enabled: bool, passphrase_opt: Option<String> },
}

// Events Service -> UI
enum Event {
  Heartbeat,
  SessionStarted { end_utc: i64 },
  SessionStopped,
  RemainingTime { seconds: u32 },
  BlockedLaunch { app: AppSummary, mode: BlockMode },
  Warning { message: String },
}

#[derive(Clone)]
struct BlockRule {
  id: Option<i64>,
  app_id: String,
  match_kind: MatchKind,
  mode: BlockMode,
}

enum MatchKind { Exe, Path, Package, Lnk, Regex }
enum BlockMode { Hard, Soft }

#[derive(Clone)]
struct AppSummary {
  app_id: String,
  display_name: String,
  exe_or_target: Option<String>,
  package_family: Option<String>,
  icon_hint: Option<String>,
}
```

8) Safety & Tamper Controls

- System whitelist (e.g., `explorer.exe`, `ctfmon.exe`, AV processes).
- Never terminate elevated/system-protected processes.
- Strict mode: hide Quit; stopping requires passphrase.
- Optional watchdog: service restarts UI if closed during session.

---

9) Settings & Defaults

- Default block mode: **soft**.
- Default toasts: 15m, 5m, 1m remaining.
- Autostart service at login via Run key (toggleable).

---

10) Build, Packaging, Updates

- Installer: WiX Toolset or NSIS; install UI + service; create Run key for service UI.
- Code signing recommended (smoother SmartScreen, toast activation).
- Optional auto-update for UI (Tauri), keep offline installer SKU.

---

11) Telemetry (Optional, Off by Default)

- Local-only usage stats (sessions, minutes focused, blocks).
- Export JSON/CSV; no network by default.

---

12) Testing Plan

- **Unit**: matcher (exe/path/package/regex), scheduler edge cases, search scorer.
- **Integration**: fake process spawn, enforcement, IPC timeouts.
- **Perf**: 10k app catalog, search latency, watcher latency (<200ms event→action).
- **Manual**: block `notepad.exe` soft/hard; strict mode stop/passphrase; toast checkpoints.

---

13) Implementation Outline (Modules & Key Files)

```
/focuslock
  /service (Rust)
    main.rs
    ipc_server.rs
    scheduler.rs
    watcher.rs
    evaluator.rs
    enforcer.rs
    notifier.rs
    repo.rs
    catalog.rs
    appid.rs
  /ui (Tauri + React + Fluent)
    src/
      main.ts
      App.tsx
      components/
        FavoritesBar.tsx
        SessionPanel.tsx
        SearchBox.tsx
        RulesTable.tsx
        SettingsPane.tsx
      tauri.conf.json
  /shared
    protocol.rs (Request/Event types)
  Cargo.toml (workspace)
```

---

14) Backlog / Future Enhancements

- Website blocking via Windows Filtering Platform.
- Calendar integration (auto-start sessions).
- Per-app quotas (e.g., “Discord ≤ 30 min/day”).
- Profiles (workday vs weekend).
- Multi-monitor full-screen focus shield.