Vinsly Desktop - Agent Studio

A Tauri-powered desktop application for designing, organising, and analysing Claude AI agents. This is the desktop version of the Vinsly Agent Studio, with native integration for file system operations and desktop-first workflows.

Features

- 🖥️**Native Desktop Experience** – Built with Tauri 2.0 for a fast, secure, and lightweight desktop application.
- 📁**File System Integration** – Direct access to `~/.claude/agents/` and project-level `.claude/agents/` directories.
- 🔧**Agent Management** – Create, edit, duplicate, favourite, and organise your Claude agents with a focused editor.
- 🖨️**Map View** – Visualise your "agent organisation" as a graph, grouped by scope, with exportable diagrams.
- 📊**Analytics Dashboard** – Visualise agent complexity, model distribution, tool usage, and get data-driven recommendations.
- 🔍**Scan & Watch** – Scan global and project directories for agents, and configure watched folders for auto-discovery.
- 💾**Persistent Storage** – Settings, layout preferences, and local account details are stored using Tauri Store.
- 🕚**Light/Dark Themes** – System-aware theming with a dedicated appearance section in Settings.
- 📥**Import/Export** – Import agents from `.md` and `.zip` bundles, export individual agents or curated sets.
- 🪠**Guided Tours** – Inline guided tours to explain List, Map, Analytics, and editor flows directly in the app.
- 🔐**Licensing & Account (beta)** – In-app licence key + email activation flow (backed by Lemon Squeezy), plus a local display name used across the UI (e.g. "[Name] Organisation").

UI/UX Design System

Color Theme

Dark Mode (default): - Background: `#15171c` (deep charcoal) - Surface: `#1f2229` / `#2c313a` (layered grays) - Text: `#f2f4f8` (soft white) / `#b7bdc8` (muted gray for secondary) - Border: `#3d4450` (subtle division) - Accent: `#C17356` (warm terracotta/rust) with hover state `#B06A4E`

Light Mode: - Background: `#FBFBFA` (warm off-white) - Surface: `#F7F7F5` / `#F0F0EE` (layered neutrals) - Text: `#111111` (near black) / `#555555` (medium gray for secondary) - Border: `#E5E5E3` (soft beige-gray) - Accent: `#C17356` (same terracotta across both themes)

Semantic Colors: - Success: `#00C851` - Warning: `#ffbb33` - Danger: `#ff4444`

Aesthetic & Vibe

- **Typography**: Inter (UI) and Fira Code (monospace) – clean, modern, developer-focused

- **Border Radii**: Crisp and precise (max 0.625rem / 10px) – following professional desktop tool conventions like VS Code, Linear, and Figma rather than consumer app trends. Buttons, cards, modals, and inputs use subtle rounding that conveys focus and precision.
- **Animations**: Smooth Framer Motion transitions for view changes and interactions
- **Layout**: Spacious, breathable interfaces with clear hierarchy and generous whitespace
- **Tone**: Focused desktop tool for professionals – not playful or casual, but approachable and refined

When designing the landing page or related marketing materials, match this warm-neutral, professional aesthetic with the terracotta accent as the signature brand color.

Prerequisites

Before running the Vinsly Desktop application, ensure you have:

- **Node.js** (v18 or higher)
- **Rust** (latest stable version) - Install from rustup.rs
- **Claude CLI** (optional, for future test console features) - Install following Claude Code documentation

Installation

1. **Clone or navigate to the project**:

```
cd Vinsly-Desktop
```

2. **Install dependencies**:

```
npm install
```

3. **Run in development mode**:

```
npm run tauri dev
```

4. **Build for production**:

```
npm run tauri build
```

Project Structure

```
Vinsly-Desktop/
├── src/                        # React TypeScript frontend
│   ├── components/             # React components
│   │   ├── screens/            # Main screen components
│   │   ├── analytics/          # Analytics visualizations
│   │   ├── form/               # Form components
│   │   ├── icons/              # Icon components
│   │   ├── tools/              # Tool selector components
│   │   └── wizard/             # Wizard step components
│   ├── utils/                  # Utility functions
│   │   ├── storage.ts          # Tauri Store wrapper (settings, licence,
display name)
│   │   ├── tauriCommands.ts    # Rust command wrappers (agent filesystem)
│   │   ├── agentImport.ts      # Agent import from files / zip
│   │   ├── agentExport.ts      # Agent export to zip
│   │   ├── analytics.ts        # Analytics calculations & recommendations
│   │   └── fuzzyMatch.ts       # Search functionality
│   ├── types.ts                # Core TypeScript type definitions
│   ├── types/licensing.ts      # Licence-related types
│   ├── constants.ts            # Application constants
│   ├── animations.ts           # Framer Motion animations
│   └── App.tsx                 # Main application shell (routing, tours,
activation)
├── src-tauri/                  # Rust backend
│   ├── src/
│   │   └── lib.rs             # Tauri commands and plugins
│   ├── capabilities/          # Permission configurations
│   │   └── default.json       # Default capability set
│   ├── Cargo.toml             # Rust dependencies
│   └── tauri.conf.json        # Tauri configuration
├── tailwind.config.js         # Tailwind CSS configuration
├── postcss.config.js          # PostCSS configuration
└── package.json               # Node.js dependencies
```

Key Technologies

Frontend

- **React 19** - UI framework
- **TypeScript** - Type-safe development
- **Vite** - Fast build tool
- **Tailwind CSS 4** - Utility-first styling
- **Framer Motion** - Smooth animations
- **Recharts** - Data visualization
- **JSZip** - ZIP file handling
- **@fontsource** - Self-hosted fonts (Inter, Fira Code)

Backend (Tauri)

- **Tauri 2.0** - Native desktop framework
- **Rust** - Backend language
- **Tauri Plugins**:
- `tauri-plugin-store` - Persistent key-value storage

- `tauri-plugin-dialog` - Native file dialogs
- `tauri-plugin-fs` - File system access
- `tauri-plugin-shell` - Process spawning

Available Scripts

- `npm run dev` - Start Vite development server
- `npm run build` - Build frontend for production
- `npm run tauri dev` - Run Tauri app in development mode
- `npm run tauri build` - Build Tauri app for production

Tauri Commands

The application exposes several Rust commands for native functionality:

File Operations

- `list_agents(scope, projectPath?)` – List all agent files from project or global scope.
- `read_agent(path)` – Read a single agent file.
- `write_agent(scope, name, content, projectPath?)` – Write an agent file (creating directories if needed).
- `delete_agent(path)` – Delete an agent file.
- `list_agents_from_directory(directory)` – Scan an arbitrary directory for `.claude/agents` content.

System

- `get_home_dir()` – Get cross-platform home directory path.

Development Notes

Debugging

- Enable Tauri DevTools in development mode
- Rust logs available in terminal
- Frontend console available via browser DevTools

Building for Distribution

```
npm run tauri build
```

This creates platform-specific installers in `src-tauri/target/release/bundle/`

Permissions

All permissions are configured in `src-tauri/capabilities/default.json`. The application has access to: - File system read/write - Dialog (open/save) - Store (persistent storage) - Shell (process spawning) - Event system (for streaming CLI output)

Troubleshooting

Build Issues

- Ensure Rust is installed and up to date: `rustup update`
- Clear node_modules and reinstall: `rm -rf node_modules && npm install`
- Clear Rust build cache: `cd src-tauri && cargo clean`

Runtime Issues

- Check that Claude CLI is in your PATH for Test Console features
- Verify permissions in capabilities configuration
- Check Tauri console for Rust backend errors

Pricing & Licensing (Planned)

Vinsly Desktop is intended to be sold as a **pay-to-own** product using Lemon Squeezy for billing and licence key distribution.

- **Launch pricing (planned)** – Initial target is **USD 49–59** one-time, including at least 12 months of updates.
- **Future pricing** – As deeper Claude Skills / automation features are added, the standard one-time price may move towards **USD 79–89**, with optional “supporter” or “Pro” tiers for users who want to support development more directly.
- **No subscription required** – The goal is a simple, developer-friendly one-off purchase rather than a recurring subscription for the core desktop app.

Details may evolve as the product and licensing integration mature, but the intent is to keep activation straightforward: buy on the landing page, receive a Lemon Squeezy licence key, and activate inside the app using the built-in licence + email flow.