Alles klar – hier kommt **eine vollständige, aufbereitete, technisch saubere und kopierfertige Entwickler-Dokumentation**, basierend auf **deinem Code (index.html, styles.css, sx-core.css, app.js)** und den Inhalten der hochgeladenen **Dev-Doku.pdf**.

Diese Dokumentation ist so geschrieben, dass **jeder Entwickler – auch du selbst nach Monaten Pause – das gesamte System (Frontend-MVP) sofort versteht, erweitern kann und alle Zusammenhänge checkt**.

🟦 SignalOne.cloud – Technische Entwickler-Dokumentation (Frontend-MVP)

Stand: Dezember 2025
Basierend auf **Dev-Doku.pdf** und dem vorliegenden Live-Code.

1) SYSTEMÜBERBLICK – ARCHITEKTUR

SignalOne ist eine **No-Framework SPA** (Single-Page-Application), gebaut für maximale Geschwindigkeit, Kontrolle und Erweiterbarkeit.

Technologie-Stack

Layer	Technologie	Zweck
Struktur	HTML5	Statischer Rahmen: Sidebar, Topbar, Views
Styles	CSS3	UI-Komponenten, Design System, SX-Core (Animationen, Loader)
Logic	Vanilla JavaScript ES6 Modules	State, Routing, Module Loader, Demo/Live-Modus, MetaMock, UI-Framework

Keine Frameworks (React, Vue), kein jQuery, keine Build-Tools → **100% nativ**.

SPA-Prinzip

- Die App lädt **nur eine HTML-Datei**.
- Alle inhaltlichen Seiten sind <section class="view">.
- JS blendet Views sichtbar/unsichtbar → kein Reload.

2) INDEX.HTML – STRUKTUR & FUNKTIONEN

Die HTML-Datei definiert **das vollständige, unveränderbare Grundgerüst**, auf das alle Module aufsetzen.

✔ Sidebar (<nav class="sidebar">)

- Dynamische Navigation (<ul id="navbar">)
- Jeder Button wird **per JS generiert** (renderNav()).
- Status-Bereiche:
  - #sidebarSystemDot, #sidebarSystemLabel
  - #sidebarCampaignDot, #sidebarCampaignLabel
  - #sidebarMetaDot, #sidebarMetaLabel
- Settings-Button (#settingsButton) → öffnet Settings-Modul.

✔ Topbar

- Live-Time + Date
- Begrüßung: #topbarGreeting
- Brand & Campaign Selects
- Meta Connect Button
- Notifications / Profil / Logout Buttons

✔ View-System

Jede App-Seite ist ein eigener <section>:

<section id="dashboardView" class="view"></section>

<section id="creativeLibraryView" class="view hidden"></section>

…

Die Zuweisung erfolgt über:

viewIdMap = { dashboard: "dashboardView", … }

✔ Modal & Toast Container

- Modal Overlay (#modalOverlay)
- Toast Container (#toastContainer)
- Global Loader (#globalLoader)

3) CSS – DESIGN SYSTEM & GUI FRAMEWORK

3.1 styles.css

Das zentrale Design-System:

🎨 Design Tokens (root-Variablen)

- Primärfarben
- Neutral Background Stack
- Textfarben
- Statusfarben
- Sidebar Icon Duotone Gold palette
- Shadows, Radius, Inputs

🧱 Komponenten

- Sidebar (komplett animiert, duotone icons)
- Topbar
- Views (Glassmorphism + Blur + Gold-Header)
- Cards
- KPI-Gitter
- Tabellen
- Creative Library UI
- Buttons (Meta-Button, Icon-Buttons)
- Modals
- Toasts
- Utility Klassen

✨ Interaktionen

- Hover-Lift
- Press Feedback
- Fadeln-Up Animations

3.2 sx-core.css

Enthält „Core Experience Layer“:

🌫️ Global Loader

- Blur
- Drei pulsierende Punkte (loader-dots)
- Smooth transitions

🦴 Skeleton Loader

- Shimmer-Effekt
- Wird in JS via applySectionSkeleton() injiziert

🔄 View Transitions

- Smooth Fade / translateY
- Aktiv durch .view.is-active

4) app.js – DAS JAVASCRIPT-BACKBONE

Die 1000+ Zeilen Code ergeben folgendes Subsystem:

4.1 AppState – zentrales State-Management

Alles wird hier gespeichert:

```
AppState = {
  currentModule: "dashboard",
  metaConnected: false,
  meta: { accessToken, user },
  selectedBrandId,
  selectedCampaignId,
  licenseLevel,
  systemHealthy,
  notifications: [],
  settings: { demoMode, dataMode, theme, currency… }
}
```

Alle UI-Komponenten lesen und schreiben diesen globalen State.

4.2 MetaAuthMock – simulierte Meta Ads Connect

Features:

- Persistenz via localStorage
- Simulierter OAuth-Flow
- Übergibt:
  - accessToken
  - user
- Aktualisiert:
  - Sidebar Status
  - Topbar Greeting
  - Campaign Health
  - Subheader

Mock-Schlüssel:
signalone_meta_mock_v1

4.3 Module Registry (Dynamic Imports)

Jedes Modul hat:

export function render(root, AppState, { useDemoMode })

In app.js:

modules = {

 dashboard: () => import("./packages/dashboard/index.js"),

 creativeLibrary: …

}

Neue Module → einfach ergänzen.

4.4 Navigation / Routing

navigateTo(key) macht:

1. currentModule setzen
2. Sidebar hervorheben
3. View via setActiveView() einblenden
4. loadModule(key) ausführen

View-Wechsel ist instant, Inhalt kommt async.

4.5 loadModule(key)

1. Loader anzeigen
2. Skeleton einsetzen
3. Modul dynamisch importieren
4. render() der Moduldatei ausführen
5. View animiert einblenden

Falls Modul fehlt → automatisch generierte „Noch nicht implementiert“-Seite.

4.6 Brand & Campaign Kontext

- BrandSelect & CampaignSelect reagieren live
- Jeder View bekommt Subheader (Owner, Kampagnenzahl, Vertical)
- Subheader wird dynamisch generiert (Icon + Text)

4.7 Status-System

✓ System Health

✓ Campaign Health

✓ Meta Connect Status

UI wird immer automatisch aktualisiert.

4.8 Global UI Components

Toasts

showToast(msg, type)

Modal

openSystemModal(title, bodyHtml)

Global Loader

showGlobalLoader() / hideGlobalLoader()

4.9 TestingLog API

Persistenz:
signalone_testing_log_v1

Funktionen:

- add(entry)
- clear()
- seedDemo()
- openTestSlot(A, B)

Modules können alles direkt nutzen über:

window.SignalOne.TestingLog

4.10 Bootstrap Ablauf

Beim App-Start wird ausgeführt:

1. MetaAuthMock.init()
2. Sidebar erzeugen
3. BrandSelect / CampaignSelect laden
4. Status & Topbar aktualisieren
5. View aktivieren
6. Modul laden (Dashboard)
7. Interval für Uhrzeit beginnen

5) AKTUELLER ENTWICKLUNGSSTAND (DEZ 2025)

✓ Fertig:

- HTML Grundgerüst
- Styles (Vision-Pro Titanium UI)
- SX-Core Loader & Skeleton
- Navigation & Routing
- Demo Meta Auth
- Brand/Campaign Context Engine
- Dashboard-Logik (Gerüst)
- Creative Library (Basis UI)
- Testing Log (global API + Persistenz)
- Modal & Toast System
- Status Wiring (System, Campaign, Meta)

✓ Codequalität:

- Voll modularisiert
- State-Driven UI
- Erweiterbar durch Packages
- Zero dependencies

6) ROADMAP / WEITERER PLAN

Phase P1 – Fundament (praktisch abgeschlossen)

Nur noch Kleinigkeiten und polieren.

Phase P2 – Creative Library

- Grid + List Mode
- Filter (Tag, Format, Metric)
- Creative Insights KPIs
- A/B Testing Hook (TestingLog API nutzen)

Phase P3 – Campaigns Module

- Kampagnentabelle
- Health Indicators
- Status Filter
- Modale für Kampagnendetails

Phase P4 – Sensei

- Prompt-basiertes System
- Strategische Empfehlungen
- Generatoren
- Playbooks

Phase P5 – Meta Connect

- Echte API-Anbindung (später)
- Token-Flow ersetzen
- Permissions-Scopes

Phase P6 – Reports & Logs

- CSV/Excel Export
- Activity / Event Log
- Diagnostics Panel

7) WAS DU TUN MUSST, WENN DU WIEDER EINSTEIGST

Cheat Sheet

1. **index.html prüfen** → stimmt ViewIdMap?
2. **Neues Modul hinzufügen?** → Registry + Section + Icon.
3. **Brands/Kampagnen erweitern?** → DemoData modifizieren.
4. **UI kaputt?**
   - data-view-title gesetzt?
   - ViewIdMap stimmt?
   - Klassennamen korrekt?
5. **MetaConnect kaputt?** → localStorage Key löschen.
6. **TestingLog spinnt?** → seedDemo() callen.

8) EXPORT – KOPIERFREUNDLICH

Wenn du willst, kann ich daraus ein **Markdown-PDF**, ein **Confluence-Dokument**, oder eine **README.md für GitHub** generieren.

Sag nur *"Erstelle README"*.

Wenn du möchtest, erweitere ich die Dokumentation jetzt auch noch um:

👉 **Backend-Architektur**
👉 **Datenflüsse**

👉 **Complete Module Development Guide**
👉 **UI-Patterns & Komponentenbibliothek**

Sag mir einfach was du brauchst.