AETHER

Technical & Functional Specifications

High-performance memory-safe web browser

Contents

PART 1: SYSTEM ARCHITECTURE **2**
- 1.1 C4 Model - System Context 2
- 1.2 C4 Model - Containers 2
- 1.3 Technology Stack Justification 2
- 1.4 Architecture Decision Records (ADRs) 3

PART 2: DATA & API MODELING **3**
- 2.1 Database Schema 3
- 2.2 Extension API Definition 4
- 2.3 Internal IPC (Inter-Process Communication) 4

PART 3: UI/UX DESIGN **5**
- 3.1 Design Philosophy 5
- 3.2 Key User Flows 5
- 3.3 UI Components Specification 6
- 3.4 Wireframes & Mockups 6
- 3.5 Color Palette & Typography 6

PART 1: SYSTEM ARCHITECTURE

1.1 C4 Model - System Context

High-level view of Aether Browser in its ecosystem:

External Systems:

- Web Servers (HTTP/HTTPS): Target websites and web applications
- DNS Servers: Domain name resolution
- Extension Repositories: Future extension marketplace
- Update Servers: Browser update distribution

User Types:

- End Users: Privacy-conscious web browsers
- Developers: Extension creators and contributors

1.2 C4 Model - Containers

Technical components of Aether Browser:

Container	Technology	Responsibility	Justification
Browser Engine	Rust + WebKit/Servo	HTML/CSS rendering, JS execution	Memory safety, performance
UI Layer	Native (Qt/GTK)	User interface, tabs, menus	Platform integration
Network Stack	Rust (reqwest/hyper)	HTTP/HTTPS requests	Security, async I/O
Storage Layer	SQLite + LevelDB	Bookmarks, history, cache	Embedded, fast, reliable
Extension API	Rust + WebAssembly	Extension runtime	Security sandboxing
Update Service	Rust client	Auto-update mechanism	Secure binary updates

1.3 Technology Stack Justification

Core Language: Rust

Justification: Memory safety without garbage collection, excellent performance, strong concurrency model, and growing ecosystem for systems programming. Critical for browser security (prevents buffer overflows, use-after-free vulnerabilities).

Rendering Engine: WebKit/Servo Hybrid

Justification: WebKit provides proven web standards compliance. Servo components (written in Rust) offer modern parallelism for layout and rendering. Hybrid approach balances compatibility with innovation.

UI Framework: Qt (C++) / GTK (Linux)

Justification: Qt provides cross-platform native UI with minimal overhead. GTK for Linux ensures proper desktop integration. Both offer mature, performance-focused UI toolkits.

Network: Hyper + Rustls

Justification: Hyper is a fast HTTP implementation in Rust. Rustls provides TLS 1.3 without OpenSSL dependencies, reducing attack surface and improving security audit capability.

Storage: SQLite + LevelDB

Justification: SQLite for structured data (bookmarks, history) - battle-tested, zero-config. LevelDB for cache and key-value storage - fast writes, proven reliability (used by Chrome).

1.4 Architecture Decision Records (ADRs)

ADR-001: Use Rust for Core Browser

- Decision: Core browser logic written in Rust
- Rationale: Memory safety, performance, modern tooling
- Alternatives considered: C++ (security concerns), Go (GC overhead)
- Consequences: Team needs Rust training, smaller talent pool

ADR-002: Hybrid WebKit/Servo Engine

- Decision: Use WebKit with Servo components
- Rationale: Web compatibility + modern performance
- Alternatives: Pure Servo (incomplete), Pure WebKit (no Rust)
- Consequences: Complex integration, ongoing maintenance

ADR-003: No Chromium Fork

- Decision: Build from scratch vs forking Chromium
- Rationale: Full control, no Google dependencies, cleaner codebase
- Alternatives: Chromium fork (like Brave/Edge)
- Consequences: Longer development, compatibility challenges

PART 2: DATA & API MODELING

2.1 Database Schema

Local Storage (SQLite)

Aether stores user data locally with no cloud sync (privacy-first):

Table	Columns	Purpose	Indexes
bookmarks	id, title, url, folder_id, created_at	User bookmarks	url, folder_id
history	id, url, title, visit_count, last_visit	Browsing history	url, last_visit
tabs	id, session_id, url, title, position	Tab restoration	session_id
settings	key, value, updated_at	User preferences	key
cookies	domain, name, value, expires, secure	Cookie storage	domain

Entity Relationships:

- bookmarks → folder_id (self-referential for hierarchy)
- tabs → session_id (groups tabs per browser session)
- All tables isolated (no foreign keys to reduce complexity)

2.2 Extension API Definition

Future extension API (WebExtension-compatible subset):

API Namespace	Methods	Purpose	Security
tabs	create(), update(), close()	Tab management	Manifest permission
bookmarks	create(), search(), remove()	Bookmark manipulation	Manifest permission
storage.local	get(), set(), remove()	Extension data storage	Sandboxed per extension
webRequest	onBeforeRequest, onHeadersReceived	Network interception	Restricted to patterns
contextMenus	create(), update()	Right-click menu items	Basic permission

API Security Model:

- All extensions run in isolated WebAssembly sandbox
- Manifest declares required permissions (principle of least privilege)
- No direct DOM access - must use message passing

2.3 Internal IPC (Inter-Process Communication)

Aether uses multi-process architecture for security and stability:

Process Type	Responsibility	Communication Method	Crash Impact

Browser Process	Main UI, tab management	Master - spawns all others	Total failure
Renderer Process	Web content rendering (per tab)	IPC to Browser Process	Single tab crashes
Network Process	All HTTP/HTTPS requests	IPC to Browser + Renderers	Network failure only
GPU Process	Hardware-accelerated rendering	Shared memory + IPC	Rendering degradation
Extension Process	Extension runtime (per extension)	Restricted IPC	Single extension fails

IPC Protocol: JSON-based messages over Unix domain sockets (Linux/Mac) or named pipes (Windows)

PART 3: UI/UX DESIGN

3.1 Design Philosophy

Minimalist, Distraction-Free Browsing: Aether prioritizes content over chrome (UI elements). Every UI element must justify its existence.

Core Design Principles:

- Privacy-First: No telemetry, no analytics, no phone-home
- Speed: <1 second startup time, instant tab switching
- Simplicity: Clean UI with no clutter or unnecessary features
- Accessibility: WCAG 2.1 AA compliance from day one

3.2 Key User Flows

Primary User Stories:

User Story 1: Browse a Website

- User types URL in address bar → Press Enter
- Page loads with progress indicator
- Tracker blocking status shown in address bar icon

User Story 2: Manage Tabs

- User clicks '+' button to open new tab
- Tabs displayed horizontally with title and favicon
- Middle-click or 'X' button closes tab

User Story 3: Save Bookmark

- User clicks star icon in address bar
- Dialog appears to edit title and choose folder
- Bookmark saved locally (no sync)

3.3 UI Components Specification

Component	Location	Functionality	Keyboard Shortcut
Address Bar	Top center	URL input, search, suggestions	Ctrl+L
Tab Bar	Below address bar	Tab management, reordering	Ctrl+T (new), Ctrl+W (close)
Tracker Shield	Address bar (right)	Show blocked trackers count	Click to expand
Bookmark Star	Address bar (right)	Add/remove bookmark	Ctrl+D
Menu Button	Top right (3 dots)	Settings, history, downloads	Alt+F
Back/Forward	Address bar (left)	Navigation history	Alt+Left/Right
Reload Button	Address bar (right)	Refresh current page	Ctrl+R / F5

3.4 Wireframes & Mockups

Design Tool: Figma (recommended) or Sketch

Key Screens to Design:

- Main Browser Window (with single tab)
- Multi-Tab View (6+ tabs)
- Settings Panel
- Bookmark Manager
- History View
- Tracker Blocking Dialog

Responsive Breakpoints:

- Desktop: 1920x1080, 1366x768, 2560x1440
- Small windows: 800x600 minimum (graceful degradation)

3.5 Color Palette & Typography

Dark Mode (Default):

- Background: #1E1E1E (Dark Gray)
- Text: #E0E0E0 (Light Gray)
- Accent: #4A9EFF (Blue) for links and actions
- Warning: #FF6B6B (Red) for security alerts

Light Mode (Optional):

- Background: #FFFFFF (White)
- Text: #2C2C2C (Dark Gray)
- Accent: #0066CC (Darker Blue)

Typography:

- UI Font: System default (San Francisco on Mac, Segoe UI on Windows, Roboto on Linux)
- Monospace: JetBrains Mono (for developer tools)
- Base Size: 14px (scalable via accessibility settings)

Document Date: Mars 01, 2026

Modify Date: November 13, 2026