System Design Document: Dallal

Project Name: Dallal (Open Home Lab Discovery & Dashboard)

Version: 1.2.0 (Application Registry Update)

Document Date: December 8, 2025

Target Environment: Self-Hosted Home Network / Air-Gapped LAN (Docker)

1. Executive Summary

Dallal is a self-hosted, web-based "Single Pane of Glass" application designed to simplify the management of home server and offline lab environments. Unlike static dashboards, Dallal features an **Active Discovery Engine** that autonomously scans the local network to identify running web services.

Core Philosophy: Local Sovereignty. Dallal is architected to function 100% offline. It has **zero dependencies on the public internet**. All assets, including fonts, icons, JavaScript libraries, and documentation, are bundled directly into the container. It is fully functional in air-gapped environments where no internet connection is available.

2. Functional Requirements

2.1. Offline Network Auto-Discovery Engine

The system proactively probes the Local Area Network (LAN) to identify services without calling home or querying external databases.

- **Subnet Scanning (ARP & ICMP):**
  - Accepts a CIDR range (e.g., `192.168.1.0/24`).
  - Utilizes **ARP scans** (Address Resolution Protocol) as the primary method for discovering live hosts, ensuring detection even if firewalls block ICMP (Ping).
  - Falls back to ICMP Echo Requests for routed subnets.
- **Port & Service Detection (TCP Connect):**
  - Scans active hosts against a customizable local configuration file of common web ports (e.g., 80, 443, 8080, 8123, 9000, 9443).
  - Uses rapid TCP Connect scanning (non-blocking) to minimize scan duration.
- **Local Fingerprinting:**
  - **HTML Analysis:** Fetches the root page of discovered ports and parses the `<title>` tag to name the service.
  - **Favicon Scraping:** Direct HTTP requests to the local IP (e.g., `http://192.168.1.50:8123/favicon.ico`) to cache icons locally. **No external icon APIs (like Google Favicon) are used.**
  - **Header Analysis:** Inspects `Server` and `X-Powered-By` HTTP headers to identify underlying technologies (e.g., identifying "Jetty" implies a Java app, "TwistedWeb" often implies Python/Media tools).

- **Passive Monitoring:**
  - Optional integration with local DNS servers (e.g., Bind9, Unbound) to detect new hostnames without active scanning.

2.2. The "Wrapper" Interface (UI/UX)

The user experience is designed for complete isolation from the public web.

- **Sidebar Navigation:** A persistent, collapsible left-hand menu.
- **Asset Bundling:** All UI elements (fonts like Inter/Roboto, icon sets like FontAwesome/Lucide) are served from the local Dallal container. No requests are made to `fonts.googleapis.com` or CDNs.
- **Dual View Modes:**
  - **Embedded (IFrame):** Loads the application directly inside the dashboard.
  - **External (New Tab):** Opens the local IP in a new tab.
- **Smart Fallbacks:** If a service has no icon, a generated SVG based on the service initials is created locally by the frontend.

2.3. Manual Management

- **Custom Entries:** Form input for specialized internal tools.
- **Local Image Upload:** Users upload custom PNG/JPG icons directly to the Dallal server storage (persisted in the `/data` volume), ensuring no reliance on hotlinking external images.
- **Grouping:** Logical grouping (Infrastructure, Media, Dev) stored in the local SQLite database.

2.4. Local API Smart Widgets

Dallal bypasses the web UI to fetch raw JSON data from local APIs for "High Value" applications identified in the user's stack.

- **Virtualization (Proxmox/ESXi):** Connects via local IP to fetch cluster health, CPU, RAM, and VM count.
- **Containerization (Docker/Portainer):** Mounts `/var/run/docker.sock` or queries the Portainer API to show Container Count (Running/Stopped).
- **Network Blocking (Pi-hole/AdGuard):** Fetches `queries_blocked_today` and `dns_queries_today` .
- **Media Management (The Arrs):*
  - **Sonarr/Radarr/Lidarr:** Queries the `/api/queue` endpoint to display "Currently Downloading" counts or "Missing Media" counts.
  - **Plex/Jellyfin:** Queries `/status/sessions` to display current active stream count.
- **Downloaders (Sabnzbd/qBittorrent):** Fetches current download speed and "Time Remaining" for active queues.
- **Storage (TrueNAS/Unraid):** Fetches ZFS Pool health status and available disk space.

3. System Architecture

3.1. Technology Stack (Strictly Local)

- **Frontend:** React.js (Vite build).
  - *Build Configuration:* Configured to bundle all assets. `npm build` produces a static folder where `index.html` references relative paths only.
  - *Icons:* `lucide-react` (bundled).
- **Backend:** Python (FastAPI).
  - *Networking:* `scapy` (for ARP/Layer 2 access), `socket`, `asyncio`, `requests`.
  - *Server:* `Uvicorn` (ASGI).
- **Database:** SQLite.
  - *Storage:* A single `dallal.db` file located in the persistent volume.
- **Container OS:** Alpine Linux.
  - *Rationale:* Minimal footprint, reduced attack surface.

3.2. Data Flow (Offline Context)

1. **Scan Initiated:** User clicks "Scan LAN".
2. **Layer 2 Probe:** Backend sends ARP "Who-Has" broadcast packets.
3. **Response:** Local devices reply with MAC addresses.
4. **Layer 3 Probe:** Backend attempts TCP handshake on defined ports for identified MACs.
5. **Metadata Fetch:** Backend performs HTTP GET on open ports.
6. **Persistence:** Results saved to SQLite. Use of `MAC Address` as the unique identifier allows devices to change IPs (DHCP) while retaining their dashboard config.
7. **Rendering:** Frontend fetches JSON from Backend API.

4. Database Schema

Table: `services`

Field	Type	Description
`id`	UUID	Primary Key
`mac_address`	String	**Critical for Local ID.** Used to track devices even if IP changes.
`display_name`	String	e.g., "Home Assistant"
`internal_ip`	String	e.g., "192.168.1.50"
`port`	Integer	e.g., 8123
`category`	String	e.g., "Media", "Networking", "Security"
`local_icon_path`	String	Path to locally stored/cached icon (e.g., `/static/icons/uuid.png`)
`view_mode`	Enum	`IFRAME` or `EXTERNAL_TAB`

`is_online`	Boolean	Result of last health check ping.
`widget_type`	String	e.g., "sonarr_queue", "proxmox_stats", "generic_ping"
`api_key`	String	Encrypted locally. Optional for public dashboards.

5. Service Compatibility & Handling Strategy

This section defines how Dallal handles specific classes of applications from the User's Service Registry.

5.1. Category Mapping

User Category	Dallal Group	Default Icon
Virtualization & Containers	`Infrastructure`	Server
Networking & DNS	`Network`	Wifi
Storage, Backup & Sync	`Storage`	HardDrive
Media Servers	`Media`	PlayCircle
Downloaders & Automation	`Downloads`	DownloadCloud
Home Automation & IoT	`Smart Home`	Home
Development, Git & CI/CD	`Development`	Code
Monitoring, Logging & Security	`Monitoring`	Activity
Identity, Auth & Directory	`Security`	Shield
Databases	`Data`	Database
Communication	`Social`	MessageCircle
Web Apps & Tools	`Tools`	Grid

5.2. View Mode Enforcement (Security & Headers)

Certain applications in the user's stack enforce strict security headers (`X-Frame-Options: DENY` or `SAMEORIGIN`) or Content Security Policies (CSP) that prevent them from being rendered inside an IFrame. Dallal auto-detects these, but defaults are defined below:

- **Strict `EXTERNAL_TAB` Enforcement:**
  - **Firewalls/Routers:** *pfSense, OPNsense, OpenWRT* (Often block embedding to prevent clickjacking).

    - **Identity/Auth:** *Keycloak, Authentik, Vaultwarden, Authelia* (Security best practice prevents embedding login screens).
    - **Banking/Finance:** *Firefly III* (Often has strict CSP).
    - **Virtualization Consoles:** *Proxmox (Console View), VMware ESXi* (VNC/Spice sockets often fail inside nested IFrames).
- `IFRAME` **Compatible (Usually):**
    - **Media:** *Radarr, Sonarr, Jellyfin* (May require "Allow iFraming" checkbox in app settings).
    - **Dashboards:** *Grafana* (Requires `allow_embedding = true` in `grafana.ini`).
    - **Docs/Wiki:** *BookStack, Wiki.js*.

5.3. Offline Asset Risks & Mitigation

Many "self-hosted" apps still rely on public CDNs (Google Fonts, cdnjs) which break in air-gapped labs.

- **Identified Risks:**
    - *Plex:* Often requires internet for auth and metadata. **Mitigation:** Suggest *Jellyfin* in UI or link to Plex "List of IP addresses... allowed without auth" guide.
    - *Heimdall / Dashy:* These dashboards often fetch icons from GitHub/internet. **Mitigation:** Dallal replaces these dashboards entirely, removing the dependency.
    - *Docs:* Some documentation tools load fonts from Google. **Mitigation:** Dallal's "Wrapper" mode does not fix the app's internal missing fonts, but ensures the *Dashboard itself* remains 100% usable.

6. Technical Risks & Mitigations (Offline Focused)

Risk	Description	Mitigation Strategy
Clock Drift	Without internet (NTP), logs and charts may de-sync.	Allow backend to sync time from a local router or hypervisor via guest tools.
HTTPS/SSL Errors	Local services often use Self-Signed Certificates, causing browser warnings in IFrames.	Backend includes a "Certificate Authority" generator to create a root CA users can install in their browser to trust local HTTPS services.
Lack of Updates	No internet means no automatic container updates.	Implement a visible "Version Hash" in the footer so users can manually verify against releases when they do have access.

7. Future Roadmap (Post-V1.0)

7.1. Local Authentication (No Cloud SSO)

- **Local LDAP/AD Integration:** Allow users to authenticate against a local Active Directory or OpenLDAP server.
- **Basic Auth:** Simple username/password stored (hashed) in the local SQLite DB.

7.2. Self-Hosted Remote Access

- **WireGuard Configuration Generator:** Instead of cloud tunnels, the UI provides a configuration page to help set up a local WireGuard container.
- **Reverse Proxy Manager:** A visual interface to manage a local Nginx instance (handling `plex.lab.local` etc.).

7.3. Hardware Power Management (Wake-on-LAN)

- **Magic Packet:** Backend integration to send WoL packets to specific MAC addresses.
- **Shutdown Command:** Integration with QEMU-Guest-Agent or SSH to gracefully shut down local servers from the dashboard.

7.4. "Sneakernet" Backup

- **USB Export:** Ability to detect a plugged-in USB drive (mapped to the container) and automatically dump the database and config for physical backup