PROJECT_ALPHA — System Design

Purpose: Network-level parental/school filtering and monitoring system that blocks listed websites for selected devices and provides admins with per-device time-on-site and visited-content visibility (within legal/ethical boundaries).

---

1. Goals & Constraints

Primary goals - Block domains/websites (porn, undesirable social apps, etc.) for selected devices (IP/MAC/groups). - Provide per-device reports: time spent per domain, top visited domains, blocked-attempt log. - Make enforcement robust against simple bypasses (DNS change, casual VPNs) while preserving privacy and complying with consent/regulations.

Constraints & non-goals - Aim to avoid full content recording for minors unless explicit consent & legal compliance exist. Default to domain-level logging; optional agent for URL-level tracking. - Support mixed environments: managed devices (school-owned) and unmanaged (BYOD). - Start with a simple, deployable MVP — extensible to a cluster later.

---

2. High-Level Architecture

```
[Devices] --(DNS + HTTP/HTTPS)-->
[Edge Router/Firewall] --> [Traffic Redirect]
                        --> [DNS Filter (Pi-hole / custom)]
                        --> [Transparent Proxy (optional)]
                        --> [Packet/Flow Collector (NetFlow/sFlow)]

Logs --> [Log Aggregator] --> [Analytics] --> [TimeSessionizer]
                                            --> [Database (Postgres +
ClickHouse/Timescale)]

Admin UI <--> [Backend API (Auth, Policy Engine, Rules Manager)]

Optional: Agent on managed devices --> Reports events to Backend (URL, active
time)
```

Primary components - Edge Router / Firewall (pfSense, iptables/nftables) — enforces network-level blocks, redirects DNS. - DNS Filtering Service — Pi-hole or custom DNS with blocklist management and per-IP logging. - Transparent Proxy / TLS Interception (optional, with consent) — Squid or mitmproxy for deeper inspection. - Agent (optional) — lightweight cross-platform service for managed devices to report active URL and focus time. - Backend (API) — Policy engine, device registry, jobs to process logs, web UI backend. - Data Storage & Analytics — Postgres for relational data; ClickHouse/Timescale for high-volume logs & fast analytics. - Admin Dashboard — React + Tailwind UI for rule management, reports, and alerts.

---

3. Data Flow & Sequence (Domain-level MVP)

1. Device issues DNS query for `instagram.com`.
2. Edge firewall redirects DNS (port 53) to DNS Filter.
3. DNS Filter checks: device IP -> device group -> policy -> blocklist.
4. If blocked: return NXDOMAIN (or redirect to block notice page) and log the attempt.
5. If allowed: resolve and forward.
6. DNS logs (domain, device_id, timestamp) are batched to Log Aggregator.
7. TimeSessionizer jobs run every minute to convert DNS/proxy logs into domain sessions using heuristics (5 minute idle timeout, session start/stop).
8. Admin UI queries API for per-device sessions, top domains, blocked attempts.

Optional agent flow (for URL-level accuracy): - Agent logs active URL, window focus, timestamps and POSTs events to Backend over TLS. - Backend merges agent events with network logs for reconciliation.

---

4. Core Modules & Responsibilities

4.1 Edge Router / Firewall

- Enforce NAT and redirect DNS to internal resolvers.
- Block outbound traffic to known VPN endpoints and suspicious ports.
- Apply network segmentation (student devices vs admin devices).
- Tools: pfSense, MikroTik, iptables/nftables.

4.2 DNS Filtering Service

- Maintain categorized blocklists, allowlists, custom lists.
- Per-query logging containing: `timestamp`, `device_ip`, `device_mac` (if available), `domain`, `qtype`, `response`.
- Provide API to check if domain is blocked and to add/remove domains.
- Tools: Pi-hole (MVP) or PowerDNS with middleware (scale).

4.3 Policy & Rule Engine (Backend)

- Manage device registry: (id, mac, ip, name, group_id, metadata).
- Store policies mapped to groups. Policy contains blocklists and exceptions.
- Evaluate queries from DNS Filter (fast lookup cache e.g., Redis).

4.4 Log Aggregator & Sessionizer

- Consume DNS logs and proxy logs (via filebeat/rsyslog or direct POST).
- Normalize logs and push to analytics DB.
- Sessionize domains per device using heuristics (idle timeout = 5 minutes configurable).

4.5 Analytics DB

- Hot storage: ClickHouse / Timescale for high write throughput and analytical queries.
- Relational: Postgres for devices, policies, users, audit logs.

4.6 Admin Dashboard

- Device list, Device detail (sessions timeline, top domains, blocked attempts), Policy editor, Blocklist manager, Reports & Exports.
- RBAC: roles like `superadmin`, `school_admin`, `teacher`, `parent`.

4.7 Agent (Optional)

- Cross-platform lightweight service.
- Reads active URLs from browser history/API (respect platform limitations); monitors foreground app/window to compute active time.
- Secure enrollment and per-device key. Data minimization by default (send domain + active_time; full URL optional).

---

5. Data Model (Core Tables)

```
-- devices
CREATE TABLE devices (
  id uuid PRIMARY KEY,
  mac varchar,
  ip inet,
  name varchar,
  group_id uuid,
  registered_at timestamptz
);

-- groups
CREATE TABLE groups (
  id uuid PRIMARY KEY,
  name varchar
);

-- policies
CREATE TABLE policies (
  id uuid PRIMARY KEY,
  name varchar,
  description text
);

-- blocklists
CREATE TABLE blocklists (
  id uuid PRIMARY KEY,
  name varchar,
  source varchar,
  created_at timestamptz
);

-- dns_logs (ingest table, high volume)
CREATE TABLE dns_logs (
```

```
  id uuid,
  device_id uuid,
  device_ip inet,
  domain varchar,
  qtype smallint,
  timestamp timestamptz
);

-- domain_sessions (computed)
CREATE TABLE domain_sessions (
  id uuid,
  device_id uuid,
  domain varchar,
  start_ts timestamptz,
  end_ts timestamptz,
  duration_seconds int
);
```

Notes: use ClickHouse or Timescale for `dns_logs` for scale; Postgres for metadata.

---

6. API Contract (sample endpoints)

```
GET /api/v1/devices
GET /api/v1/devices/{id}
POST /api/v1/devices -- register device
PUT /api/v1/policies/{id}
GET /api/v1/reports/device/{id}?from=...&to=...
POST /api/v1/blocklist -- add domain
POST /api/v1/dns-lookup (internal) -- check domain for device
POST /api/v1/agent/events -- agent uploads
```

Internal fast-check endpoint (used by DNS filter) - `POST /api/v1/dns-lookup` payload: `{ device_ip, device_mac, domain }` - Response: `{ action: 'ALLOW' | 'BLOCK' | 'REDIRECT', reason }`

Authentication: mTLS for internal calls or HMAC-signed requests (low-latency). Cache results in Redis for 60s.

---

7. Sessionization Heuristics (how we convert logs to "time spent")

- Use DNS + SNI + proxy logs to detect domain activity.
- Group consecutive events to the same domain into sessions if gap < `idle_timeout` (default 300s).
- Session start = first request timestamp; end = last request timestamp + `post_activity` buffer (e.g., 10s).

- Device active-time approximated as sum(session durations). If agent present, use agent active-time for accuracy.

Edge cases: background auto-refresh or video streaming (large continuous requests) — use heuristics to avoid overcounting (e.g., detect high throughput but no user activity -> flag as passive consumption).

---

8. Enforcement Strategies

DNS-level (MVP)

- Return `NXDOMAIN` for blocked domains.
- Or return local HTTP redirect to block notice for HTTP(S) (use captive portal-like behavior for UX).

Proxy-level (optional)

- Transparent proxy handles HTTP; for HTTPS optionally perform TLS interception only on managed devices.
- If TLS interception not possible, log SNI where available.

Agent-level

- Agent enforces OS-level blocking via hosts file or local firewall for unmanaged networks when installed.

---

9. Anti-bypass Measures (VPN, DoH, DNS change)

- Force DNS by redirecting port 53 traffic to internal resolver (iptables/nftables). Also handle TCP/53.
- Block common DoH providers (cloudflare-dns.com, dns.google) at DNS & HTTP level.
- Block common VPN ports and known VPN provider IP ranges.
- Heuristic detection: long-lived encrypted connection to exotic IP -> flag in dashboard.
- Agent: detect running VPN clients/processes and alert admin.

---

10. Security & Privacy

- Default to domain-only logging. Allow opt-in for full URL capture only with consent & explicit policy.
- RBAC, audit logs, encryption at rest (AES-256) and TLS in transit.
- Data retention policy: configurable; default 30 days for detailed logs, 12 months aggregated.
- Provide export & forget functions for GDPR/COPPA compliance.

---

11. Admin UX / Wireframes (text)

Dashboard landing: summary widgets: online devices, blocked attempts today, suspicious VPN flags.

Devices page: table with columns (name, IP, MAC, group, last_seen, top_domain_24h, actions).

Device detail page: timeline (last 24h) with sessions; top domains with time bars; blocked attempts list with timestamps; button to request unblock.

Policy editor: drop zones for categories (Porn, Social, Streaming). Save/assign to groups.

Blocklist manager: search/add/remove domains, import from file, schedule updates.

Reports: generate CSV/PDF per device or per group, schedule email reports.

---

12. MVP Roadmap (milestones)

- **Week 0:** Project kick-off, choose stack, infra readiness.
- **Week 1–2:** Implement DNS filter + redirect (Pi-hole), device registry, backend `dns-lookup` endpoint and simple UI (devices list).
- **Week 3–4:** Log ingestion, basic sessionizer, device detail page + blocked attempts view.
- **Week 5–6:** Policy editor, blocklist import, report export.
- **Week 7+:** Optional agent prototype, VPN-detection heuristics, scaling to ClickHouse.

(If you want I can convert this into a more detailed sprint plan.)

---

13. Testing & QA

- Unit tests for policy engine and sessionizer logic.
- Integration tests: simulate device traffic, DNS queries, VPN attempts.
- Performance tests: DNS QPS, log ingestion rates.
- User testing with a small class of devices before wider roll-out.

---

14. Example Config Snippets (MVP)

iptables - redirect DNS to internal resolver (192.168.1.2):

```
iptables -t nat -A PREROUTING -p udp --dport 53 -j DNAT --to-destination
192.168.1.2:53
iptables -t nat -A PREROUTING -p tcp --dport 53 -j DNAT --to-destination
192.168.1.2:53
```

Pi-hole custom blacklist entry (example):

```
# /etc/pihole/custom.list
0.0.0.0 badpornsite.example
0.0.0.0 socialtimewaster.example
```

Fast-check cache (Redis) TTL: 60s for dns-lookup responses.

---

15. Compliance & Legal Checklist

- Privacy policy tailored for minors/educational settings.
- Consent/opt-in flows for parents and staff.
- Data retention & deletion policy; export/right-to-be-forgotten.
- If using TLS interception: documented process, secured CA, and formal consent.

---

16. Next concrete deliverables I can produce now (pick one)

- A. Docker Compose prototype (Pi-hole + backend + simple UI skeleton).
- B. OpenAPI spec for the backend (full endpoints documented).
- C. Agent prototype (Python script for Linux/Windows to capture active domains).
- D. Detailed 8-week sprint plan with tasks and acceptance criteria.

Tell me which one and I will produce it immediately.