**GoalFlow – API Layer Documentation (v1)**

**1) Purpose & Scope**

The API layer exposes GoalFlow’s application capabilities (auth, goals, progress logs, reminders, real-time updates) via **ASP.NET Core Minimal APIs**. It wires up cross-cutting services (JWT auth, authorization policies, CORS, rate limiting, ProblemDetails, Swagger, Serilog, Hangfire, SignalR) and delegates business logic to **MediatR handlers** in the Application layer.

**Principles**

* Keep endpoints thin; delegate logic to Application.
* Secure-by-default: short-lived access tokens, ownership policy, rate limits.
* Consistent error responses via ProblemDetails.
* Real-time UX via SignalR.

**2) Key Components**

**2.1 Authentication & Tokens**

* **JWT Bearer** with HMAC-SHA256.
* **TokenService** issues:
  + **Access token** (15 min).
  + **Refresh token** (random 64B, returned raw; only **SHA-256 hash** stored in DB).
* **Rotation**: on refresh, old token revoked; new pair issued.

**2.2 Authorization (Ownership)**

* **Requirement**: MustOwnGoal.
* **Handler**: MustOwnGoalHandler checks route {id} vs current user’s uid in DB.
* **Policy**: "OwnerGoal" applied on GET/PUT/DELETE /api/goals/{id}.

**2.3 Contracts (DTOs)**

* UserDto, RefreshDto, CreateGoalBody, CreateProgressBody, UpdateGoalBody.
* Purpose: strict, explicit API inputs; Swagger shows shapes.

**2.4 ProblemDetails (Errors)**

* Uniform error payloads:
  + 400 validation (FluentValidation),
  + 401 unauthorized,
  + 404 not found,
  + 500 default.
* Human-readable Title/Detail.

**2.5 Rate Limiting**

* **fixed**: 100 req/min per IP (general).
* **login**: 5 req/min per IP (auth endpoints).
* Returns 429 with no queue.

**2.6 CORS**

* Policy **client**: allows local dev origins and credentials for SignalR.

**2.7 Swagger**

* Global **Bearer** security requirement.
* Dev-only UI enabled.

**2.8 SignalR (Real-Time)**

* **Hub**: ProgressHub (/hubs/progress).
* **Client contract**: IProgressClient → ProgressLogged, JoinedGoal.
* **Join flow**: JoinGoal(goalId) adds connection to goal:{id} group.

**2.9 Domain → Real-Time Bridge**

* **Notification handler**: ProgressLogCreatedHandler (MediatR INotificationHandler).
* On ProgressLogCreated: broadcasts ProgressLogged(dto) to goal:{id}.

**2.10 Background Jobs**

* **Hangfire** dashboard /hangfire.
* Sample recurring job: ReminderProcessor.ProcessDueReminders() every minute.

**3) Request Lifecycle (Typical)**

Client → HTTPS → Minimal API Endpoint → [AuthN: JWT] → [AuthZ: Policy?]

→ MediatR Command/Query → Application Handler → Infrastructure/DB

→ Result → ProblemDetails (on error) or 2xx/201 with payload

**Progress log flow with real-time:**

POST /api/goals/{id}/progress-logs

→ CreateProgressLogCommand → DB insert → Publish ProgressLogCreated

→ ProgressLogCreatedHandler → SignalR: Clients.Group("goal:{id}").ProgressLogged(dto)

→ Connected clients receive update immediately

**4) Security Model**

* **Authentication**: JWT; strict validation; ClockSkew = 1m.
* **Authorization**: "OwnerGoal" ensures per-resource ownership.
* **Tokens**:
  + Access: short-lived; carry sub, email, uid.
  + Refresh: **hash-at-rest**, rotation on refresh.
* **Rate limiting**: blunt-force protection on hot paths (login, CRUD).
* **CORS**: explicitly allow known dev origins.
* **Operational**: Hangfire dashboard should be protected in non-dev.

**5) Configuration & Environment**

* **appsettings.\*.json → Jwt section**: Issuer, Audience, Key.
* **Origins** in CORS should align with FE envs.
* Serilog console sink; can switch to JSON format and add sinks (Seq, Loki, etc.).
* Infrastructure registration via AddInfrastructure(...) (EF Core, Redis, etc.).

**6) Error Handling & Observability**

* **ProblemDetails** assures consistent client errors.
* **Serilog + Span enricher** for trace correlation (future OTel-ready).
* Health endpoints: /health/live, /health/ready.

**7) Extensibility Guidelines**

**Add a new secure endpoint**

1. Define request DTO (Contracts).
2. Add Minimal API route; **authorize** ([Authorize] + policy if resource-scoped).
3. Send a **MediatR** command/query; return standardized results.
4. If event-driven, publish a domain notification; handle it with INotificationHandler.
5. Document example requests/responses in Swagger (optional examples).

**Add another ownership-scoped resource**

* Create MustOwnX requirement/handler or reuse pattern with route key lookup.
* Register policy and apply .RequireAuthorization("OwnerX") on routes.

**Add a real-time event**

* Extend IProgressClient or create a new Hub/contract.
* From the notification handler, broadcast to a suitable group/topic.

**8) Testing Checklist**

**Unit**

* TokenService:
  + Issues access/refresh; stores **hash**; rotates; rejects inactive/invalid.
* MustOwnGoalHandler:
  + Succeeds for owner; fails otherwise; handles missing/invalid route IDs.

**Integration**

* Auth endpoints:
  + register/login/refresh happy/edge paths; rate limit behaviour (429).
* Goals CRUD:
  + Ownership enforced on GET/PUT/DELETE by policy.
  + Validation errors map to 400 with details.
* Progress:
  + Create log returns 201; SignalR broadcast fired (verify via test client or hub spy).
* ProblemDetails shape for 400/401/404/500.

**E2E (dev)**

* FE client can:
  + Login, create goal, update, list, delete.
  + Open SignalR connection, join goal, see real-time ProgressLogged.

**9) API Examples (Copy-ready)**

**Login**

POST /api/auth/login

Content-Type: application/json

{ "email": "user@example.com", "password": "P@ssw0rd!" }

**Refresh**

POST /api/auth/refresh

Content-Type: application/json

{ "userId": "<guid>", "refreshToken": "<raw-base64>" }

**Create Goal**

POST /api/goals

Authorization: Bearer <jwt>

Content-Type: application/json

{

"title": "Learn React",

"specific": "...",

"measurable": "...",

"achievable": "...",

"relevant": "...",

"timeBound": "2025-12-31T23:59:59Z",

"description": "Optional",

"priority": "High"

}

**SignalR (client pseudo)**

const conn = new signalR.HubConnectionBuilder()

.withUrl("/hubs/progress", { accessTokenFactory: () => jwt })

.build();

conn.on("ProgressLogged", payload => console.log(payload));

await conn.start();

await conn.invoke("JoinGoal", goalId);

**10) “Done & Dusted” – What’s Complete**

* Minimal APIs wired with **JWT / Owner policy / rate limiting / CORS / Swagger / ProblemDetails**.
* Endpoints for **Auth, Goals, Progress Logs, Reminders**.
* **SignalR hub** + **MediatR notification handler** for real-time progress.
* **Hangfire** integration with a sample recurring job.
* **Comments & XML docs** across helpers/services for interview readability.