CitizenshipApp

Technical Blueprint (Developer-Follow)

Professional Word export generated from the project blueprint.

Cập nhật theo code + docs hiện có trong repo (mốc tài liệu: **2026-01-19**).

Stack: **.NET 9** • Clean Architecture • ASP.NET Core API • Blazor WASM UI • EF Core + SQL Server • WorkerService • Tests.

1) Mục tiêu sản phẩm

App ôn thi quốc tịch Mỹ cho người lớn tuổi (elderly-first): thao tác ít, chữ lớn, nút lớn, phản hồi rõ ràng, lỗi dễ hiểu.

Yêu cầu phi chức năng:

- Tính nhất quán API contract (Contracts ở Shared) và lỗi theo chuẩn RFC7807 (ProblemDetails).
- Bảo mật tối thiểu: JWT + rate limit cho auth + CORS an toàn.
- Quan sát: correlation id cho mọi request/response.

2) Cấu trúc repo và trách nhiệm từng project

Solution: CitizenshipApp.sln

Projects:

- Api/ — ASP.NET Core Web API (composition root: DI + middleware + controllers)
- Application/ — định nghĩa interface/use-case contract (ports) để Api không phụ thuộc EF trực tiếp
- Domain/ — entity + enum thuần (không phụ thuộc ASP.NET/EF/Identity)
- Infrastructure/ — EF Core, Identity, seed data, và implementation của Application interfaces
- Shared/ — Contracts dùng chung cho API + UI (Không dùng DTO mà sẽ dùng Contracts cho toàn bộ dự án như API, Ui.Blazor, WorkerSerivce, …)
- Ui.Blazor/ — Blazor WebAssembly frontend
- WorkerService/ — background service skeleton (chưa có job nghiệp vụ)
- Tests/ — unit + integration tests

Docs (source of truth): docs/* (PROJECT-STATE.md, DECISIONS.md, UX-RULES.md, LOCAL-CONFIG.md)

3) Luật kiến trúc (Clean Architecture)

3.1 Quy tắc phụ thuộc

- **Domain**: không tham chiếu ASP.NET, EF Core, Identity.
- **Application**: chỉ định nghĩa interface/contract; không query DB.
- **Infrastructure**: implement Application, chứa EF Core/Identity.
- **Api**: gọi Application, cấu hình pipeline; controller mỏng.
- **Ui.Blazor**: gọi API qua ApiClient; không logic DB.
- **Shared**: Contracts duy nhất cho payload public.

3.2 Nguyên tắc controller mỏng

Controller chỉ:

- nhận input, validate (ModelState)
- gọi service (Application interface)
- trả Contracts (Không dùng DTO mà sẽ dùng Contracts cho toàn bộ dự án như API, Ui.Blazor, WorkerSerivce, …)

Không nhét business rule/EF query dài trong controller

4) Local run (theo `docs/LOCAL-CONFIG.md`)

Ports (launchSettings):

- API: https://localhost:7070 (HTTP: http://localhost:5294)
- UI: https://localhost:7031 (HTTP: http://localhost:5215)

4.1 DB bằng Docker (khuyến nghị)

- docker-compose.sqlserver.yml + .env.example → copy thành .env và set MSSQL_SA_PASSWORD.

4.2 Secrets (Development)

API load user-secrets trong Development (Api/Program.cs).

Chạy trong folder Api/:

```
dotnet user-secrets set "ConnectionStrings:DefaultConnection"
"Server=localhost,1433;Database=CitizenshipApp;User
Id=sa;Password=<YOUR_PASSWORD>;TrustServerCertificate=True"

dotnet user-secrets set "Jwt:Key" "<YOUR_LONG_RANDOM_KEY_32+>"

# optional seed admin

dotnet user-secrets set "Seed:AdminEmail" "admin@local"

dotnet user-secrets set "Seed:AdminPassword" "ChangeMe123!"
```

4.3 Run order

```
# 1) DB

docker compose -f docker-compose.sqlserver.yml --env-file .env up -d


# 2) API

dotnet run --project Api


# 3) UI

dotnet run --project Ui.Blazor


# 4) Worker (optional)

dotnet run --project WorkerService
```

5) API blueprint

5.1 Cross-cutting behaviors (Api/Program.cs)

ValidationProblemDetails (ModelState)

- Khi invalid model → trả application/problem+json (RFC7807) + traceId + correlationId + instance.

Rate limiting (BL-015)

- Policy auth-login: 10 requests / 1 phút / IP
- Policy auth-register: 5 requests / 5 phút / IP
- Policy auth-general: sliding window 30 / 1 phút / IP
- Reject → 429 ProblemDetails.

Global exception handler → ProblemDetails

- Map exception → status:
  - UnauthorizedAccessException → 401
  - ArgumentException/InvalidOperationException → 400
  - KeyNotFoundException → 404
  - others → 500
- Dev environment mới đính kèm message/type chi tiết.

CorrelationIdMiddleware (BL-014)

- Header: X-Correlation-ID
- Nếu client gửi thì reuse; không thì generate.
- Response luôn trả header này.
- Log request tối giản; bỏ qua /health, /swagger.

CORS

- Dev: allow any.
- Non-dev: đọc Cors:AllowedOrigins; nếu không cấu hình thì deny-by-default (an toàn).

Reverse proxy

- Chỉ bật forwarded headers khi Proxy:Enabled=true (ADR-014).

Health checks (ADR-015)

- GET /health/live (self)
- GET /health/ready (self + DB)

5.2 Auth (Api/Controllers/AuthController.cs)

Endpoints:

- POST /api/auth/register (rate limit: auth-register)
  - Create AppUser (Identity)
  - Create UserProfile(IsOnboarded=false)
  - Create UserSettings default (elderly-first)
  - Transaction để tránh orphan
  - Return AuthResponse (JWT + UserId + IsOnboarded)

- POST /api/auth/login (rate limit: auth-login)
  - Check password
  - Read UserProfile.IsOnboarded
  - Return JWT

JWT claim rule (ADR-012): token luôn có cả sub và ClaimTypes.NameIdentifier để extract userId ổn định.

5.3 Me (Api/Controllers/MeController.cs) — require `[Authorize]`

- GET /api/me/profile
- PUT /api/me/onboarding/complete
- GET /api/me/settings/full
- PUT /api/me/settings/full

Settings contract:

- Contract: Shared.Contracts.Me.UserSettingContracts (property-based record + DataAnnotations) (ADR-016).
- Legacy MVP endpoint đã removed; dùng /api/me/settings/full làm nguồn duy nhất (ADR-023).

5.4 Deck/Questions/Study

Application/Decks/IDeckQueryService (read-only) + Infrastructure/Decks/DeckQueryService:

- GET /api/decks trả danh sách deck.

Application/Study/IStudyService + Infrastructure/Study/StudyService:

- GET /api/study/next?deckId=...
- POST /api/study/answer
- GET /api/study/today

Chiến lược chọn câu “random” (BL-018):

- Tránh Count + Skip.
- Dùng pivot Guid + query lấy QuestionId gần pivot (keyset-like), wrap-around nếu hết.
- Sau đó load question + options.

6) UI blueprint (Blazor WASM)

6.1 Program + DI (Ui.Blazor/Program.cs)

- Set HttpClient.BaseAddress theo Api:BaseUrl (prod) hoặc mặc định dev.
- Typed client: ApiClient + AuthHeaderHandler (attach Bearer token).

6.2 Route guards (Ui.Blazor/Shared/RouteGuard.razor)

- Chưa login: chỉ cho /login, /register, /.
- Login nhưng chưa onboarded: ép về /onboarding.
- Đã onboarded: chặn quay lại /login//register.

6.3 ApiClient error handling

Ui.Blazor/Services/ApiClient.cs:

- Không dùng EnsureSuccessStatusCode().
- Với non-success: parse application/problem+json thành ApiException (bao gồm validation errors).

6.4 User settings client cache

Ui.Blazor/Services/UserSettingsState.cs:

- Cache settings cho user đã login.
- Apply global effects (hiện tại: FontScale) qua JS interop (UiDomInterop).

7) Data model blueprint (Domain + EF Core)

7.1 User data

- Identity user: Infrastructure/Identity/AppUser.
- Domain:
  - UserProfile (IsOnboarded + timestamps)
  - UserSettings (Language/FontScale/AudioSpeed/DailyGoalMinutes/Focus/SilentMode + timestamps)

Pattern quan trọng: UserSettings.Id == UserProfile.Id (shared PK).

7.2 Question bank + study log

Domain entities:

- Deck (DeckId, Code, Name, IsActive)
- Question (QuestionId, DeckId, Type, prompts bilingual, explains, CorrectOptionKey)
- QuestionOption (key/text/sortOrder)
- StudyEvent (UserId, DeckId, QuestionId, SelectedKey, IsCorrect, CreatedUtc)

Migrations: Infrastructure/Persistence/Migrations/*.

8) Testing blueprint

Projects:

- Tests/Api.IntegrationTests dùng WebApplicationFactory<Program>.
- DB cho integration test: SQLite in-memory (Data Source=:memory:) (ADR-010).
- Set JWT config test cố định.
- Init schema bằng EnsureCreated().

Hiện có test cho validation (auth + settings).

9) Quy trình thêm feature (developer cookbook)

9.1 Thêm endpoint mới

1) Tạo/điều chỉnh Contract trong Shared/Contracts/...

2) Thêm interface method trong Application/...

3) Implement trong Infrastructure/...

4) Controller ở Api/... gọi interface (controller mỏng)

5) Update Ui.Blazor/Services/ApiClient.cs + page/state

6) Viết integration test trong Tests/Api.IntegrationTests

7) Update docs nếu thay đổi lớn

9.2 Thêm bảng/đổi schema

1) Thêm entity trong Domain

2) Cấu hình EF trong Infrastructure/Persistence/Configurations

3) Update AppDbContext

4) Tạo migration:

```
dotnet ef migrations add <Name> --project Infrastructure --startup-project Api
```

5) Update seed (nếu cần) Infrastructure/Persistence/SeedData.cs

6) Add/adjust tests

10) Quy ước chất lượng code (áp dụng cho contributor)

Yêu cầu của project: mọi code mới/chỉnh sửa phải có **comment chi tiết bằng tiếng Anh**, tập trung vào "why", assumption, edge case, và safety/security.

Chuẩn clean code:

- Tên biến/class/method bằng tiếng Anh, nhất quán.
- Read-only query dùng AsNoTracking().
- Ưu tiên cancellation token cho I/O.
- Không trả lỗi dạng string ad-hoc cho endpoint mới; dùng ProblemDetails.
- Tránh Count + Skip khi dataset lớn; dùng keyset/paging ổn định.

11) Roadmap anchors (từ docs)

Các mốc/backlog thường gặp trong docs:

- Paging cho question list lớn
- Study flow integration tests
- JWT unit tests
- Worker background jobs
- CI analyzers/formatting gate
- Audio/TTS integration

Appendix — File nên đọc trước khi onboard dev mới

1) docs/PROJECT-STATE.md

2) docs/DECISIONS.md

3) docs/LOCAL-CONFIG.md

4) Api/Program.cs

5) Ui.Blazor/Program.cs

6) Infrastructure/Persistence/AppDbContext.cs + migrations

7) Infrastructure/Study/StudyService.cs

8) Ui.Blazor/Services/ApiClient.cs + Shared/RouteGuard.razor