Mock-to-Real Migration Plan

Mock API to Real Backend — Migration Plan

Generated: 2026-03-26 **Author:** FlowForge (ALAI DevOps) **Project:** LumisCare Enterprise Healthcare Platform **Scope:** 222 mock endpoints → 11 Spring Boot microservices via Web BFF

---

1. Current State Summary

Component	Status
Mock API (server.js)	Running — 5,280 lines, 222 endpoints, frontend pointed here
Spring Boot services	11 deployed on Azure Container Apps — all Running
PostgreSQL (10 databases)	Flyway migrated, schemas applied, zero seed data
Web BFF	Deployed — assessment, careplan, incidents, hr, finance clients wired; visits + scheduling + notification + identity NOT fully wired
Frontend	Points to mock API

---

2. Coverage Analysis — 222 Mock Endpoints vs Real Services

Method

Mock endpoints extracted from `mock-api/server.js` via route registration. Real coverage determined from OpenAPI specs in `openapi-specs/` and Web BFF controller inventory in `backend/bff/web-bff/src/main/java/.../controller/`.

Total endpoint count breakdown:

- 222 total mock registrations
- 18 are catch-all wildcards (`/api/*`, `/employees/:id`, etc.) — not real routes
- 204 are distinct addressable mock endpoints

Coverage By Service Domain

VISITS — 14 mock endpoints

Mock routes: `GET/POST /api/v1/visits`, `GET/PATCH/DELETE /api/v1/visits/:id`, check-in, check-out, eMAR, tasks, clinical-observations, SOS, travel-record, publish, running-late, pre-visit-brief

Real service (`visits-service`): OpenAPI defines `/visits`, `/visits/{visitId}`, check-in, check-out, tasks, observations, status, notes, eMAR.

Web BFF: No `VisitsController` found in BFF controller list. `VISITS_SERVICE_BASE_URL` is configured in `application.yml` (port 8085) but no BFF controller proxies it yet.

Coverage status: Service READY, BFF wiring MISSING. Estimated real coverage: 9 of 14 endpoints have a backend implementation. 5 (clinical-observations format, SOS, pre-visit-brief, travel-record, eMAR administer/refuse detail) need verification.

SCHEDULING / ROSTERING — 8 mock endpoints

Mock routes: `GET /api/v1/schedule-blocks`, `GET/POST/PATCH/DELETE /api/v1/schedule-blocks/:id`, `GET /api/v1/completed-events`, bulk-confirm, publish

Real service (`scheduling-service`): OpenAPI defines schedule-events, appointments, rostered-hours, recurring-schedules, availability. Maps partially — `schedule-blocks` naming differs from `schedule-events`.

Web BFF: `CalendarController.java` exists. `SCHEDULING_SERVICE_BASE_URL` configured (port 8086).

Coverage status: PARTIAL — naming mismatch between mock and real service paths needs alignment. Estimated real coverage: 5 of 8.

ASSESSMENT — 12 mock endpoints

Mock routes: `GET/POST/PUT/PATCH /api/v1/assessments`, body-map, scores, complete, `GET /api/v1/intake`

Real service (`assessment-service`): Full OpenAPI — assessment CRUD, sections, subsections, questions, risk sections, observations, photos, medications, status, submit, intake.

Web BFF: `BffAssessmentController.java` + `AssessmentWebDashBoardController.java` exist. `AssessmentClientConfig` wired.

Coverage status: COVERED for core paths. Scores/clinical-tools endpoints may map to risk-section in real service. Estimated real coverage: 10 of 12.

CARE PLANS — 14 mock endpoints

Mock routes: `GET/POST/PUT/DELETE /api/v1/care-plans`, versions, tasks, risk-alerts, reviews, approve, publish, submit-for-approval, consent, review-due

Real service (`careplan-service`): OpenAPI defines care-plans CRUD, publish, regenerate, generation-status, versions, master-task-types/visit-types.

Web BFF: `CarePlanController.java` exists. `CarePlanClientConfig` wired.

Coverage status: Core CRUD + publish + versions covered. Task management, risk-alerts, reviews, consent are mock-only. Estimated real coverage: 7 of 14.

HR — 18 mock endpoints

Mock routes: `GET /api/v1/hr/supervisions`, absences, dbs, bradford, care-certificate, training matrix, supervision-schedule, cqc-inspection-pack; legacy `/employees/*`

Real service (`hr-service`): Extensive OpenAPI — employees full profile, leaves, supervisions, training, appraisals, documents, equality diversity, onboarding.

Web BFF: `EmployeeController.java` + `ResumeUploadController.java` exist. `HrClientConfig` wired.

Coverage status: Employee core COVERED. HR compliance sub-paths (DBS checks, Bradford factor, care certificate, CQC pack) are mock-only abstractions — hr-service has training and supervision but not as named sub-routes. Estimated real coverage: 10 of 18.

FINANCE — 20 mock endpoints

Mock routes: invoices, timesheets, rate-cards, funders, invoice-groups, payment-groups, payroll preview/export/approve, aged-debtors, NMW compliance, bank-holidays, month-end status, dashboard

Real service (`finance-service`): OpenAPI defines invoices (full lifecycle), timesheets, invoice-groups, payment-groups, mandates, dashboard. Rate-cards and funders are mock-only concepts.

Web BFF: `FinanceBffController.java` exists. `FinanceClientConfig` wired.

Coverage status: Invoice + timesheet core COVERED. Rate-cards, funders, payroll export, NMW compliance, aged-debtors are MOCK-ONLY. Estimated real coverage: 12 of 20.

INCIDENTS — 8 mock endpoints

Mock routes: `GET/POST/PATCH/DELETE /api/v1/incidents`, `GET /api/v1/incidents/:id`

Real service (`incidents-service`): OpenAPI defines accidents, complaints, hazards, safeguarding, summary, actions log.

Web BFF: `IncidentsController.java` exists. `IncidentsClientConfig` wired.

Coverage status: COVERED — mock uses generic `/incidents` but real service has typed categories (accidents, complaints, hazards, safeguarding). Path mapping required in BFF. Estimated real coverage: 6 of 8.

NOTIFICATIONS — 10 mock endpoints

Mock routes: notifications inbox, unread-count, preferences, devices register/unregister, mark-all-read, push

Real service (`notification-service`): OpenAPI defines inbox, devices, preferences, templates, unread-count, send, mark-read.

Web BFF: No `NotificationController` found. `notification-service` NOT listed in `application.yml` service URLs.

Coverage status: Service READY, BFF wiring MISSING. Estimated real coverage: 7 of 10 (paths align), but BFF layer absent.

IDENTITY / USERS / ORGS / SERVICE USERS — 14 mock endpoints

Mock routes: `GET /api/v1/users`, `GET /api/v1/service-users`, organizations, settings, carers, skills

Real service (`identity-service`): OpenAPI defines users, organizations, service-users, roles, settings.

Web BFF: `ServiceUsersController.java` exists. `IdentityClientConfig` found in config.

Coverage status: PARTIAL — users and service-users COVERED. Carers-by-skills, skills list, rostering-settings are mock-only. Estimated real coverage: 9 of 14.

FAMILY PORTAL — 16 mock endpoints

Mock routes: family care-plan view, complaints, messages/threads, notifications, payments initiate/confirm

Real service: No dedicated family-portal-service. Family portal routes would be served by web-bff aggregating careplan-service + notification-service + identity-service.

Web BFF: No family portal controller found.

Coverage status: MOCK-ONLY. No real backend for family portal BFF layer. Estimated real coverage: 0 of 16.

MISC (dashboard, CQC, EVV, import, audit, observations, insurance, handoff) — 24 mock endpoints

Includes: `GET /api/v1/dashboard/stats`, `/api/v1/cqc-readiness`, `/api/v1/evv/*`, `/api/v1/import/*`, `/api/v1/audit-logs`, `/api/v1/insurance/*`, `/api/v1/handoff/*`, `/api/v1/observations`

Real services: EVV (electronic visit verification) maps to visits-service tracking endpoints. Dashboard stats aggregate from multiple services. Import, insurance, handoff are mock-only constructs.

Coverage status: MOSTLY MOCK-ONLY. Estimated real coverage: 4 of 24 (dashboard + some audit, EVV via visits tracking).

Coverage Summary Table

Domain	Mock Endpoints	Real Coverage	Status
Visits	14	~9	BFF wiring missing
Scheduling	8	~5	Path naming mismatch
Assessment	12	~10	BFF wired, good coverage
Care Plans	14	~7	BFF wired, tasks/risks mock-only
HR	18	~10	BFF wired, compliance routes mock-only
Finance	20	~12	BFF wired, rate-cards/payroll mock-only
Incidents	8	~6	BFF wired, path mapping needed
Notifications	10	~7	BFF wiring MISSING
Identity/Users/Orgs	14	~9	BFF partial
Family Portal	16	0	No backend at all
Misc / EVV / Dashboard	24	~4	Mostly mock-only
TOTAL	**158 (excl wildcards)**	**~79**	**~50% real coverage**

Bottom line: approximately 79 of 158 meaningful mock endpoints have a corresponding real service implementation. The remaining ~79 are either BFF-wiring gaps, path-mapping gaps, or features that do not yet exist in any real service.

3. Seed Data Blocking Issue

Problem

Azure PostgreSQL Flexible Server blocks all public inbound connections by default. The 10 service databases have schemas (Flyway applied) but no seed data. Attempts to run seed scripts from a local machine are blocked by the firewall.

Options to Load Seed Data

Option A — Azure Cloud Shell (Recommended, zero setup)

```
# Open Azure Portal → Cloud Shell (Bash)
# Download seed scripts from repo or paste inline
psql "host=<your-pg-server>.postgres.database.azure.com \
      user=<admin> \
      dbname=identity_db \
      sslmode=require" \
  -f seed-identity.sql
```

No firewall changes needed. Cloud Shell is inside Azure's network boundary.

Option B — Temporary Firewall Rule (simple but requires cleanup)

```
# Add current IP to Azure PostgreSQL firewall
MY_IP=$(curl -s https://api.ipify.org)
az postgres flexible-server firewall-rule create \
  --resource-group rg-vcc-dev-001 \
  --name <pg-server-name> \
  --rule-name temp-local-dev \
  --start-ip-address $MY_IP \
  --end-ip-address $MY_IP

# Run seeds, then REMOVE the rule
az postgres flexible-server firewall-rule delete \
  --resource-group rg-vcc-dev-001 \
  --name <pg-server-name> \
  --rule-name temp-local-dev --yes
```

Option C — Run psql from a Container App (uses private endpoint, no firewall change)

```
# Exec into any running Container App that has psql available
az containerapp exec \
  --name ca-vcc-dev-identity-api-001 \
  --resource-group rg-vcc-dev-001 \
  --command "/bin/bash"
# Inside the container:
psql $DATABASE_URL -f /tmp/seed.sql
```

Option D — Seed via Service REST API (safest, uses existing auth/validation)

Write seed data through the service's own API endpoints (POST requests). Slower but validates business logic along with data. Best for production-like seed data.

Recommended approach for dev environment: Option A (Azure Cloud Shell) for bulk seed load, then Option D for per-feature test data.

---

4. Migration Strategy

Recommended: Option C — Proxy Pattern in Web BFF

The Web BFF already exists and has partial service wiring. The safest migration path is:

1. Web BFF handles ALL frontend requests
2. For endpoints with real service backing: BFF proxies to the real microservice
3. For endpoints without real backing: BFF falls back to mock API (temporarily)
4. Mock API is never called directly by frontend — only via BFF fallback

This is superior to Option A (hard cutover) and Option B (per-service switch) because:

- Frontend changes exactly once: point to BFF URL
- Rollback is instant (mock fallback still running)
- Each endpoint can be promoted independently with zero frontend changes
- Canary testing per-route is trivial

Migration Architecture

```
Frontend (React)
     |
     v
Web BFF (Azure Container Apps, port 8080)
     |-- /api/v1/assessments        --> assessment-service (LIVE)
     |-- /api/v1/care-plans         --> careplan-service (LIVE)
     |-- /api/v1/incidents/*        --> incidents-service (LIVE)
     |-- /api/v1/hr/*               --> hr-service (LIVE)
     |-- /api/v1/finance/*          --> finance-service (LIVE)
     |-- /api/v1/visits             --> visits-service (WIRE NOW)
     |-- /api/v1/notifications/*    --> notification-service (WIRE NOW)
     |-- /api/v1/schedule-blocks/*  --> scheduling-service (WIRE NOW)
     |-- /api/v1/family/*           --> [MOCK FALLBACK] no service yet
     |-- /api/v1/evv/*              --> [MOCK FALLBACK] no service yet
     |-- /api/v1/dashboard/stats    --> [BFF aggregation from multiple services]
     |-- /api/v1/cqc-readiness      --> [MOCK FALLBACK]
     |-- everything else            --> [MOCK FALLBACK]
     |
     v (fallback only)
Mock API (Node.js, to be decommissioned incrementally)
```

---

5. Step-by-Step Migration Plan

Phase 0 — Preparation (1-2 days)

0.1 Load seed data

- Use Azure Cloud Shell + Option A above
- Minimum seed sets needed per service:
  - `identity_db`: 1 organization, 3-5 users (admin, care-manager, carer), 2-3 service users
  - `assessment_db`: 2-3 assessment templates (assessment_questions, assessment_sections)
  - `careplan_db`: master task types, master visit types
  - `visits_db`: visit statuses, task types
  - `hr_db`: leave types, training categories
  - `scheduling_db`: appointment types

- `finance_db`: invoice statuses, timesheet statuses
- `incidents_db`: incident categories (accidents, complaints, hazards, safeguarding)
- `notification_db`: notification templates

0.2 Verify all 11 Container Apps are healthy

```
az containerapp list \
  --resource-group rg-vcc-dev-001 \
  --query "[].{name:name, status:properties.runningStatus}" \
  --output table
```

0.3 Set frontend env var (do not deploy yet)

```
# In frontend/web/.env.production
VITE_API_BASE_URL=https://<web-bff-fqdn>.azurecontainerapps.io
```

Phase 1 — Wire Missing BFF Routes (3-5 days)

1.1 Wire visits-service to BFF

- Add `VisitsController.java` to `backend/bff/web-bff/src/main/java/.../controller/`
- Configure `VISITS_SERVICE_BASE_URL` env var in Container App (already in application.yml)
- Map mock routes to real service paths:
  - `GET /api/v1/visits` → `GET /visits` (visits-service)
  - `POST /api/v1/visits/:id/check-in` → `POST /visits/{visitId}/check-in`
  - `POST /api/v1/visits/:id/check-out` → `POST /visits/{visitId}/check-out`
  - `GET /api/v1/visits/:id/emar` → `GET /visits/{visitId}/emar`

1.2 Wire notification-service to BFF

- Add `NotificationController.java`
- Add `notification.service.base-url` to `application.yml`
- Set `NOTIFICATION_SERVICE_BASE_URL` env var in Container App
- Map: `/api/v1/notifications/*` → notification-service paths

1.3 Wire scheduling-service to BFF

- `CalendarController.java` exists — extend to handle `schedule-blocks` alias
- `schedule-blocks` in mock = `schedule-events` in real service
- Map: `GET/POST /api/v1/schedule-blocks` → `GET/POST /schedule-events`

1.4 Verify identity-service routing

- `ServiceUsersController.java` exists — confirm it handles `/api/v1/service-users/*`

- Add routing for `/api/v1/organizations/*` and `/api/v1/users/*`

Phase 2 — Switch Frontend to BFF (1 day)

2.1 Update frontend environment

```
# In frontend/web/ Azure Static Web App config:
VITE_API_BASE_URL=https://<web-bff-fqdn>.azurecontainerapps.io
```

2.2 Add BFF fallback proxy for unimplemented routes In Web BFF, add a fallback route that proxies unknown `/api/v1/*` requests to the mock API:

```
# application-dev.yml
mock-api:
  fallback-url: ${MOCK_API_URL:http://mock-api:3000}
  fallback-enabled: ${MOCK_API_FALLBACK_ENABLED:true}
```

This ensures zero frontend breakage during migration. As each BFF controller is added, the fallback for that route is removed.

2.3 Keep mock API running (do not shut down) Mock API remains as a fallback target for the BFF. It can be stopped when fallback-enabled is set to false.

Phase 3 — Per-Domain Cutover (1-2 weeks)

Work through each domain in priority order. For each:

1. Confirm seed data in that service's DB
2. Smoke-test the real service endpoint directly (health + basic CRUD)
3. Enable BFF routing to real service
4. Disable fallback for that domain's routes
5. Run Playwright smoke tests

Priority order:

1. Identity (login, users, orgs) — everything else depends on auth
2. Assessment (core clinical workflow)
3. Care Plans (built on assessment)
4. Visits (mobile carer workflow)
5. Incidents (safety-critical, short paths)
6. HR (staff management)
7. Scheduling (complex, validate separately)

8. Finance (invoicing, last due to rate-card gaps)
9. Notifications (async, lower urgency)
10. Family Portal (no service — keep mock until service is built)

Phase 4 — Decommission Mock (after Phase 3 complete)

4.1 Disable BFF fallback

```
MOCK_API_FALLBACK_ENABLED=false
```

4.2 Run full Playwright test suite Confirm zero regressions. Mock fallback should not be needed.

4.3 Stop mock API Container App (or LaunchAgent if local)

```
az containerapp update \
  --name ca-vcc-dev-mock-api \
  --resource-group rg-vcc-dev-001 \
  --min-replicas 0 \
  --max-replicas 0
```

4.4 Archive mock-api/server.js (do not delete — reference for missing features)

---

6. Rollback Plan

Rollback is available at every phase with zero data loss risk.

Instant rollback (any phase)

```
# Re-point frontend to mock API
# In Azure Static Web App config:
VITE_API_BASE_URL=https://mock-api-url

# OR in BFF — re-enable fallback
MOCK_API_FALLBACK_ENABLED=true
```

Because the mock API is never decommissioned during active migration (only at Phase 4), rollback is a single env var change at the frontend or BFF level.

Per-service rollback

If a specific service is misbehaving, disable only its BFF route and re-enable fallback for that domain only. All other services continue serving real data.

Database rollback

Schema rollback is possible via Flyway undo migrations. Data rollback requires a database restore from Azure Backup (automated daily snapshots on PostgreSQL Flexible Server).

---

7. Endpoints That Are Mock-Only (No Real Service Equivalent)

These require new service implementation or design decisions before migration is possible:

Mock Endpoint	Category	Effort
`GET /api/v1/finance/rate-cards`	Finance	Medium — add to finance-service
`GET /api/v1/finance/funders`	Finance	Medium — add to finance-service
`GET /api/v1/finance/payroll/*`	Finance	High — payroll is a separate domain
`GET /api/v1/finance/aged-debtors`	Finance	Low — calculated report from invoices
`GET /api/v1/finance/nmw-compliance`	Finance	Medium
`GET /api/v1/hr/bradford`	HR	Low — calculated from absences
`GET /api/v1/hr/care-certificate`	HR	Medium
`GET /api/v1/hr/cqc-inspection-pack`	HR	High
`GET /api/v1/cqc-readiness`	Compliance	Medium — aggregate
`GET /api/v1/evv/*`	EVV	Medium — visits-service has tracking
`GET /api/v1/family/*`	Family Portal	High — no BFF layer exists
`GET /api/v1/insurance/*`	Insurance	Not in architecture — park for now
`GET /api/v1/import/*`	Bulk Import	Medium — batch job, not REST
`GET /api/v1/handoff/*`	Handoff notes	Low — maps to visits-service notes
`GET /api/v1/observations`	Observations	Low — maps to assessment-service

8. Timeline Estimate

Phase	Work	Duration
Phase 0 — Seed data + health verification	1-2 days	DevOps + 1 backend dev
Phase 1 — Wire 3 missing BFF routes	3-5 days	2 backend devs
Phase 2 — Frontend switch + BFF fallback	1 day	1 frontend dev + 1 backend dev
Phase 3 — Per-domain cutover (10 domains)	8-10 days	2 backend devs
Phase 4 — Decommission mock	1 day	DevOps
Total		**~3 weeks**

9. Risk Register

Risk	Likelihood	Impact	Mitigation
Real service returns different schema than mock	High	High	Adapter layer in BFF; validate against OpenAPI spec
Empty seed data causes null pointer errors	Medium	Medium	Populate seed before Phase 2 cutover
PostgreSQL connection limits under load	Low	High	Connection pooling via PgBouncer (already in infra)
Visits-service geofence check fails on dev	Medium	Medium	Disable geofence enforcement flag in dev profile
Auth token propagation missing in new BFF controllers	Medium	High	Follow existing `WebClientConfig` JWT propagation pattern
Family portal with no service breaks the portal	High	Medium	Mock fallback covers family routes until service is built
Azure Container App cold start latency	Low	Low	Min replicas = 1 already set in dev.bicepparam

10. Key File Locations

File	Purpose
`/Users/makinja/projects/client/lumiscare/mock-api/server.js`	Source of truth for all mock endpoint behavior
`/Users/makinja/projects/client/lumiscare/backend/bff/web-bff/src/main/resources/application.yml`	BFF service URL configuration
`/Users/makinja/projects/client/lumiscare/backend/bff/web-bff/src/main/java/.../controller/`	BFF controllers — add new ones here
`/Users/makinja/projects/client/lumiscare/backend/bff/web-bff/src/main/java/.../config/`	Service client configs — add wiring here
`/Users/makinja/projects/client/lumiscare/openapi-specs/`	OpenAPI specs per service — source of truth for real paths
`/Users/makinja/projects/client/lumiscare/infrastructure/env-parameters/dev.bicepparam`	Azure infra sizing and image names

DA Corrections — 9 Critical Gaps Addressed

Added 2026-03-26 — Devils Advocate review identified these gaps in the original plan. All 9 are corrections to existing sections or new sections not previously covered.

DA-1. Endpoint Priority Matrix

Full classification of all 222 mock endpoint registrations (18 catch-all wildcards excluded = 204 addressable endpoints).

Status definitions:

- `REAL_READY` — Spring Boot service has the endpoint AND BFF wiring exists. Can migrate immediately after seed data is loaded.
- `NEEDS_WIRING` — Spring Boot service endpoint exists in OpenAPI spec, but BFF controller for this route is absent or incomplete.
- `MOCK_ONLY` — No real service endpoint. Feature does not exist in any deployed Spring Boot service.
- `SCOPE_OUT` — Not needed for MVP. Defer to post-MVP roadmap.

Domain	Mock Route	Status	Notes
Assessment	`GET /api/v1/assessments`	REAL_READY	BFF wired
	`GET /api/v1/assessments/:id`	REAL_READY	BFF wired
	`POST /api/v1/assessments`	REAL_READY	BFF wired

Domain	Mock Route	Status	Notes
	`PUT /api/v1/assessments/:id`	REAL_READY	BFF wired
	`PATCH /api/v1/assessments/:id`	REAL_READY	BFF wired
	`POST /api/v1/assessments/:id/complete`	REAL_READY	BFF wired
	`GET /api/v1/assessments/:id/body-map`	NEEDS_WIRING	Service has photos/observations endpoints; body-map is a frontend-assembled view
	`PUT /api/v1/assessments/:id/body-map`	NEEDS_WIRING	As above
	`GET /api/v1/assessments/:id/scores`	NEEDS_WIRING	Maps to risk-section in assessment-service
	`GET /api/v1/assessments/:id/scores/:tool`	NEEDS_WIRING	As above
	`POST /api/v1/assessments/:id/scores/:tool`	NEEDS_WIRING	As above
	`GET /api/v1/intake`	REAL_READY	OpenAPI `/api/v1/intake` defined
	`GET /api/v1/intake/:id`	REAL_READY	OpenAPI `/api/v1/intake/{intake_id}` defined
	`POST /api/v1/intake`	REAL_READY	OpenAPI defined
	`PATCH /api/v1/intake/:id`	NEEDS_WIRING	OpenAPI has GET/POST only; PATCH not in spec
	`DELETE /api/v1/intake/:id`	SCOPE_OUT	Not in spec; soft-delete not required for MVP
Care Plans	`GET /api/v1/care-plans`	REAL_READY	BFF wired
	`GET /api/v1/care-plans/:id`	REAL_READY	BFF wired
	`POST /api/v1/care-plans`	REAL_READY	BFF wired
	`PUT /api/v1/care-plans/:id`	REAL_READY	BFF wired
	`DELETE /api/v1/care-plans/:id`	REAL_READY	BFF wired
	`POST /api/v1/care-plans/:id/publish`	REAL_READY	BFF wired
	`GET /api/v1/care-plans/:id/versions`	REAL_READY	BFF wired

Domain	Mock Route	Status	Notes
	`GET /api/v1/care-plans/review-due`	MOCK_ONLY	No real endpoint; aggregate query needed
	`GET /api/v1/master-task-types`	REAL_READY	Service defines master-task-types
	`GET /api/v1/master-visit-types`	REAL_READY	Service defines master-visit-types
	Care plan tasks routes	MOCK_ONLY	careplan-service has task types, not task CRUD per plan
	Care plan risk-alerts routes	MOCK_ONLY	Not in any service spec
	Care plan reviews/consent routes	MOCK_ONLY	Not in any service spec
	`POST /api/v1/care-plans/:id/submit-for-approval`	MOCK_ONLY	Status lifecycle managed differently in real service
Visits	`GET /api/v1/visits`	NEEDS_WIRING	visits-service READY, no BFF VisitsController
	`GET /api/v1/visits/:id`	NEEDS_WIRING	As above
	`POST /api/v1/visits`	NEEDS_WIRING	As above
	`PATCH /api/v1/visits/:id`	NEEDS_WIRING	As above
	`DELETE /api/v1/visits/:id`	NEEDS_WIRING	As above
	`POST /api/v1/visits/publish`	MOCK_ONLY	No equivalent in visits-service spec
	`POST /api/v1/visits/:id/check-in`	NEEDS_WIRING	visits-service has `/visits/{visitId}/check-in`
	`POST /api/v1/visits/:id/check-out`	NEEDS_WIRING	visits-service has `/visits/{visitId}/check-out`
	`GET /api/v1/visits/:id/emar`	NEEDS_WIRING	visits-service has `/visits/{visitId}/emar`
	`POST /api/v1/visits/:id/emar`	NEEDS_WIRING	visits-service has `/visits/{visitId}/emar` POST
	Clinical observations (per-visit)	NEEDS_WIRING	visits-service has `/visits/{visitId}/observations`
	SOS / running-late / pre-visit-brief	MOCK_ONLY	Not defined in visits-service OpenAPI
	Travel record	MOCK_ONLY	Tracking endpoints exist but different shape

Domain	Mock Route	Status	Notes
Scheduling	`GET /api/v1/schedule-blocks`	NEEDS_WIRING	scheduling-service has `/schedule-events`; CalendarController exists but path mismatch
	`POST /api/v1/schedule-blocks`	NEEDS_WIRING	As above
	`PATCH /api/v1/schedule-blocks/:id`	NEEDS_WIRING	As above
	`DELETE /api/v1/schedule-blocks/:id`	NEEDS_WIRING	As above
	`GET /api/v1/completed-events`	MOCK_ONLY	No direct real equivalent; map to completed appointments
	`POST /api/v1/completed-events/:id/confirm`	MOCK_ONLY	scheduling-service manages this via status update
	`POST /api/v1/completed-events/bulk-confirm`	MOCK_ONLY	No bulk confirm endpoint in spec
	Schedule publish route	MOCK_ONLY	scheduling-service has `/schedules/generate` + `/schedules/{id}/approve` which differs
HR	`GET /employees/list`	REAL_READY	BFF EmployeeController wired
	`GET /employees/:id`	REAL_READY	BFF wired
	`POST /employees`	REAL_READY	BFF wired
	`PATCH /employees/:id`	REAL_READY	BFF wired
	`DELETE /employees/:id`	REAL_READY	BFF wired
	`GET /api/v1/leaves`	REAL_READY	hr-service has leaves endpoint
	`POST /api/v1/leaves`	REAL_READY	hr-service has leaves POST
	`PATCH /api/v1/leaves/:id`	REAL_READY	hr-service has leaves PATCH
	`DELETE /api/v1/leaves/:id`	REAL_READY	hr-service has leaves DELETE
	`GET /api/v1/hr/supervisions`	NEEDS_WIRING	hr-service has supervisions; BFF sub-path mapping needed
	`GET /api/v1/hr/absences`	NEEDS_WIRING	hr-service has absences/leaves

Domain	Mock Route	Status	Notes
	`GET /api/v1/hr/dbs`	MOCK_ONLY	Not in hr-service OpenAPI; compliance documents endpoint covers this partially
	`GET /api/v1/hr/bradford`	MOCK_ONLY	Calculated from absences; no real endpoint
	`GET /api/v1/hr/care-certificate`	MOCK_ONLY	Not in hr-service spec
	`GET /api/v1/training/courses`	NEEDS_WIRING	hr-service has training; BFF not wired
	`GET /api/v1/training/compliance`	NEEDS_WIRING	hr-service has training compliance; BFF not wired
	`GET /api/v1/hr/cqc-inspection-pack`	MOCK_ONLY	No real equivalent — aggregate report
Finance	`GET /api/v1/finance/invoices`	REAL_READY	BFF FinanceBffController wired
	`POST /api/v1/finance/invoices`	REAL_READY	BFF wired
	`PUT /api/v1/finance/invoices/:id`	REAL_READY	BFF wired
	`GET /api/v1/finance/invoice-groups`	REAL_READY	BFF wired
	`GET /api/v1/finance/payment-groups`	REAL_READY	BFF wired
	`GET /api/v1/finance/dashboard`	REAL_READY	BFF wired
	Finance timesheets routes	REAL_READY	finance-service defines timesheets
	`GET /api/v1/finance/rate-cards`	MOCK_ONLY	Not in finance-service spec
	`GET /api/v1/finance/funders`	MOCK_ONLY	Not in finance-service spec
	`GET /api/v1/finance/payroll/*`	MOCK_ONLY	Payroll is a separate domain
	`GET /api/v1/finance/aged-debtors`	MOCK_ONLY	Calculated report — not in spec
	`GET /api/v1/finance/nmw-compliance`	MOCK_ONLY	Not in finance-service spec

Domain	Mock Route	Status	Notes
Incidents	`GET /api/v1/incidents`	NEEDS_WIRING	incidents-service uses typed routes (`/accidents`, `/safeguarding`, `/complaints`, `/hazards`); BFF IncidentsController must aggregate or map
	`GET /api/v1/incidents/:id`	NEEDS_WIRING	Type-specific in real service
	`POST /api/v1/incidents`	NEEDS_WIRING	Type must be specified to route to correct sub-endpoint
	`PATCH /api/v1/incidents/:id`	NEEDS_WIRING	As above
	`DELETE /api/v1/incidents/:id`	NEEDS_WIRING	As above
	`GET /api/v1/incidents/summary`	REAL_READY	incidents-service has `/summary`
	Export endpoints	REAL_READY	incidents-service has CSV export per type
Notifications	`GET /api/v1/notifications/inbox`	NEEDS_WIRING	notification-service READY, BFF NOT wired
	`GET /api/v1/notifications/inbox/unread-count`	NEEDS_WIRING	As above
	`PATCH /api/v1/notifications/inbox/:id/read`	NEEDS_WIRING	As above
	`PATCH /api/v1/notifications/inbox/read-all`	NEEDS_WIRING	As above
	`POST /api/v1/notifications/mark-all-read`	NEEDS_WIRING	As above
	`GET /api/v1/notifications/preferences`	NEEDS_WIRING	notification-service has preferences
	`POST /api/v1/notifications/preferences`	NEEDS_WIRING	As above
	`PATCH /api/v1/notifications/preferences/quiet-hours`	NEEDS_WIRING	As above
	Notification device register/unregister	NEEDS_WIRING	notification-service has devices endpoint
	`GET /api/v1/notifications/analytics`	MOCK_ONLY	Not in notification-service spec

Domain	Mock Route	Status	Notes
Identity	`GET /api/v1/service-users`	REAL_READY	identity-service + BFF ServiceUsersController
	`GET /api/v1/service-users/:id`	REAL_READY	BFF wired
	`POST /api/v1/service-users`	REAL_READY	BFF wired
	`PATCH /api/v1/service-users/:id`	REAL_READY	BFF wired
	`DELETE /api/v1/service-users/:id`	REAL_READY	BFF wired
	`GET /api/v1/organizations`	NEEDS_WIRING	identity-service has organizations; BFF routing partial
	`GET /api/v1/organizations/:id`	NEEDS_WIRING	As above
	`GET /api/v1/users`	NEEDS_WIRING	identity-service has users; BFF routing partial
	`GET /users/about-me`	REAL_READY	identity-service `/users/me`
	`GET /api/v1/carers`	NEEDS_WIRING	identity-service users filtered by carer role
	`POST /api/v1/carers/by-skills`	MOCK_ONLY	No real endpoint — would require identity + hr query
	`GET /api/v1/skills`	MOCK_ONLY	Not in any service spec
	`GET /api/v1/settings`	NEEDS_WIRING	identity-service has org settings
	`PATCH /api/v1/settings`	NEEDS_WIRING	As above
	`GET /api/v1/rostering-settings`	MOCK_ONLY	Not in any service spec
Family Portal	All 16 `GET/POST /api/v1/family/*` routes	MOCK_ONLY	No FamilyPortalService; no BFF controller (see DA-8)
Misc / EVV / Dashboard	`GET /api/v1/dashboard/stats`	NEEDS_WIRING	Requires BFF aggregation across 4+ services
	`GET /api/v1/cqc-readiness`	MOCK_ONLY	Aggregate report — no real service
	`GET /api/v1/evv/compliance`	NEEDS_WIRING	visits-service tracking endpoints cover EVV data
	`GET /api/v1/evv/records`	NEEDS_WIRING	visits-service has location tracking history
	`GET /api/v1/audit-logs`	MOCK_ONLY	No audit-service in architecture
	`GET /api/v1/insurance/*`	SCOPE_OUT	Not in product architecture — park post-MVP

Domain	Mock Route	Status	Notes
	`GET /api/v1/import/*`	SCOPE_OUT	Batch job domain, not REST for MVP
	`GET /api/v1/handoff/notes`	NEEDS_WIRING	visits-service `/visits/{visitId}/notes` covers this
	`GET /api/v1/handoff/history`	NEEDS_WIRING	visits-service notes with type=handover
	`GET /api/v1/supervisor/dashboard`	MOCK_ONLY	BFF aggregation — not yet designed
	`GET /api/v1/supervisor/dashboard/carer-locations`	NEEDS_WIRING	visits-service `/tracking/locations/current`
	`GET /api/v1/observations` (global)	NEEDS_WIRING	visits-service has per-visit observations; no global endpoint

Summary counts:

Status	Count	Action
REAL_READY	~65	Wire frontend → BFF, validate schema (see DA-3)
NEEDS_WIRING	~72	Add BFF controller code (Phase 1 work)
MOCK_ONLY	~52	Keep on mock fallback; schedule for post-MVP
SCOPE_OUT	~15	Remove from BFF routing; return 404 or 501

DA-2. Seed Data Verification (Phase 0.4)

The original plan (Phase 0.1) lists what to seed but provides no way to verify seeds actually loaded. Add Phase 0.4 after all seed scripts have run.

Phase 0.4 — Verify Seed Data Loaded Correctly

Run these queries via Azure Cloud Shell after Phase 0.1 seed load:

```
-- identity_db
SELECT COUNT(*) AS orgs          FROM organizations;          -- expect >= 1
SELECT COUNT(*) AS users         FROM users;                  -- expect >= 3
SELECT COUNT(*) AS service_users FROM service_users;          -- expect >= 2
```

```
SELECT COUNT(*) AS roles         FROM roles;                 -- expect >= 12 (system roles)

-- assessment_db
SELECT COUNT(*) AS intake        FROM assessment_intake;     -- expect >= 0 (can be empty)
SELECT COUNT(*) AS sections      FROM assessment_sections;   -- expect >= 7

-- careplan_db
SELECT COUNT(*) AS task_types    FROM master_task_types;     -- expect >= 5
SELECT COUNT(*) AS visit_types   FROM master_visit_types;    -- expect >= 3

-- visits_db
SELECT COUNT(*) AS visit_statuses FROM visit_status_lookup;  -- expect >= 6 (or check enum
table)

-- hr_db
SELECT COUNT(*) AS leave_types   FROM leave_types;           -- expect >= 4
SELECT COUNT(*) AS training_cats FROM training_categories;   -- expect >= 3

-- scheduling_db
SELECT COUNT(*) AS appt_types    FROM appointment_types;     -- expect >= 4

-- finance_db
SELECT COUNT(*) AS inv_statuses  FROM invoice_status_lookup;  -- expect >= 4

-- incidents_db
SELECT COUNT(*) AS categories    FROM incident_categories;   -- expect >= 4 (acc, saf, cmp,
haz)

-- notification_db
SELECT COUNT(*) AS templates     FROM notification_templates; -- expect >= 5
```

verify-seeds.sh — script to run from Azure Cloud Shell:

```
#!/bin/bash
# verify-seeds.sh — Run from Azure Cloud Shell after Phase 0.1 seed load
# Usage: bash verify-seeds.sh <pg-server-name> <admin-user>

PG_SERVER="${1}.postgres.database.azure.com"
PG_USER="$2"
FAILED=0
```

```
run_check() {
  local DB=$1 QUERY=$2 LABEL=$3 MIN=$4
  COUNT=$(psql "host=$PG_SERVER user=$PG_USER dbname=$DB sslmode=require" \
    -tAc "$QUERY" 2>/dev/null)
  if [ -z "$COUNT" ] || [ "$COUNT" -lt "$MIN" ]; then
    echo "FAIL [$DB] $LABEL: got $COUNT (expected >= $MIN)"
    FAILED=$((FAILED+1))
  else
    echo "PASS [$DB] $LABEL: $COUNT rows"
  fi
}

run_check "identity_db"      "SELECT COUNT(*) FROM organizations;"    "organizations"     1
run_check "identity_db"      "SELECT COUNT(*) FROM users;"            "users"             3
run_check "identity_db"      "SELECT COUNT(*) FROM service_users;"    "service_users"     2
run_check "identity_db"      "SELECT COUNT(*) FROM roles;"            "roles"             12
run_check "careplan_db"      "SELECT COUNT(*) FROM master_task_types;" "master_task_types" 5
run_check "careplan_db"      "SELECT COUNT(*) FROM master_visit_types;" "master_visit_types" 3
run_check "hr_db"            "SELECT COUNT(*) FROM leave_types;"      "leave_types"       4
run_check "scheduling_db"    "SELECT COUNT(*) FROM appointment_types;" "appointment_types" 4
run_check "incidents_db"     "SELECT COUNT(*) FROM incident_categories;" "incident_categories"
4
run_check "notification_db"  "SELECT COUNT(*) FROM notification_templates;"
"notification_templates" 5

if [ "$FAILED" -gt 0 ]; then
  echo ""
  echo "SEED VERIFICATION: $FAILED check(s) FAILED. Do NOT proceed to Phase 1."
  exit 1
else
  echo ""
  echo "SEED VERIFICATION: All checks PASSED. Safe to proceed to Phase 1."
  exit 0
fi
```

Gate: Phase 1 (BFF wiring) must not begin until `verify-seeds.sh` exits 0.

DA-3. Shape Mismatch Matrix

Comparison of mock API response shapes vs real OpenAPI-defined shapes for high-traffic routes. These mismatches require BFF adapter code — the BFF cannot simply proxy; it must transform.

Visits — `GET /api/v1/visits`

Field	Mock shape	Real service shape	Action
Response wrapper	Array `[...]`	Paginated object `{ content: [...], pageInfo: { ... } }`	BFF must extract `content` array or frontend must handle pagination
ID format	`"visit-0001"` (string with prefix)	UUID `"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"`	BFF must map or real service auto-generates UUIDs
Client reference	`clientId` + `clientName` (flat)	`serviceUserId` (UUID) — no embedded name	BFF must enrich with identity-service lookup or frontend fetches separately
Status values	`"in-progress"`, `"scheduled"`, `"completed"` (kebab-case)	`in_progress`, `scheduled`, `completed` (snake_case enum)	BFF adapter or frontend normalizes
Visit type	`"personal-care"`, `"medication"` (kebab-case strings)	`AppointmentType` enum: `care_visit`, `assessment_visit`, `medication_only`, `welfare_check`	Mapping required — not 1:1
GPS check-in	`gpsCheckIn: { lat, lng }`	`checkInLocation: { latitude, longitude, accuracy, altitude, timestamp }` (GpsLocation schema)	Field rename + structure change
Duration	`duration: 60` (minutes, integer)	Derived from `scheduledStartTime` / `scheduledEndTime` — no standalone `duration` field	BFF calculates or omits
Required headers	None	`X-VCU-Organization-Id`, `X-VCU-User-Id`, `X-VCU-Roles` must be forwarded	BFF must propagate JWT-extracted headers

Incidents — `GET /api/v1/incidents`

Field	Mock shape	Real service shape	Action
Response wrapper	`{ incidents: [...], total: N }`	Each type has own paginated endpoint: `/accidents`, `/safeguarding`, `/complaints`, `/hazards`	BFF must fan-out to 4 endpoints and merge, OR frontend switches to type-specific routes
`id` format	`"inc-001"`	UUID	BFF-generated or real service UUID

Field	Mock shape	Real service shape	Action
`type` values	`"Accident"`, `"Medication_Error"`, `"Safeguarding"` (mixed case, non-typed)	Routes are typed: `/accidents` = type ACCIDENT/INCIDENT/NEAR_MISS; `/safeguarding` uses `ConcernType` enum	Routing logic required in BFF
`status` values	`"Resolved"`, `"Under_Review"`, `"Investigating"`	`LogStatus` enum: `IDENTIFIED`, `RAISED`, `IN_PROGRESS`, `UPHELD`, `NOT_UPHELD`, `PARTIALLY_UPHELD`, `WITHDRAWN`, `RESOLVED`	Mapping table needed
`incident_number`	`"INC-2024-001"` (mock format)	Auto-generated: `"ACC-2026-00001"`, `"SAF-2026-00001"` etc.	Display only; reference format changes
Duplicate fields	Both `serviceUserName` and `service_user_name` exist	Single `serviceUserId` (UUID reference)	Remove duplicates; BFF enriches name
`actionsTaken`	Embedded array of action objects	Separate endpoint: `/logs/{logType}/{logId}/actions`	BFF must fetch separately or omit on list
Header	`X-Org-Id` (mock uses no header)	`X-VCC-Organization-Id` required	BFF must add

Scheduling — `GET /api/v1/schedule-blocks`

Field	Mock shape	Real service shape	Action
Path	`/api/v1/schedule-blocks`	`/api/v1/scheduling/schedule-events`	BFF maps path
Response wrapper	Array `[...]`	Paginated `{ content: [...], ... }`	BFF unwraps or frontend adapts
`type` values	`"team-meeting"`, `"training"`, `"supervision"`, `"appraisal"`	`ScheduleEventType` enum: `appraisal`, `meeting`, `supervision`, `training`	`team-meeting` → `meeting` mapping
`id` format	`"sb-001"`	UUID	BFF maps
`carerId`	String `"3"`	UUID	BFF must translate carer numeric ID → UUID via identity-service

Assessment — `GET /api/v1/assessments/:id`

Field	Mock shape	Real service shape	Action
`completionStatus`	`"IN_PROGRESS"` / `"COMPLETED"`	`status` field in real service with different enum values	Field rename

Field	Mock shape	Real service shape	Action
`patientId`	UUID reference	`serviceUserId` — same concept, different field name	BFF renames
`assessmentId` + `id`	Both present (legacy compat)	Single `id` field	BFF normalizes to single `id`
`overview`	Nested object `{ riskLevel, allergies, dnacp, nextAssessmentDate, ... }`	Not a single nested block — data spread across sections and risk-section endpoints	BFF must assemble from multiple sub-endpoints
`assessments`	Array of section objects with nested questions	Sections are separate sub-resources fetched via `/assessments/{id}/sections`	BFF must aggregate
Header	No header in mock	`X-VCC-User-Id`, `X-VCC-Organization-Id`, `X-VCC-User-Email` required	BFF forwards

Care Plans — `GET /api/v1/care-plans/:id`

Field	Mock shape	Real service shape	Action
`clientId`	String	Real service uses `serviceUserId` (UUID)	Field rename in BFF adapter
`status` values	Custom string values	`CarePlanStatus` enum: `draft`, `ai_generated`, `under_review`, `approved`, `active`, `suspended`, `completed`, `cancelled`	Map mock statuses to enum values
Server URL	Mock uses `/api/v1/care-plans`	Real service base: `https://api.vcc2.com/v1/api/v1/care-plans` — note double prefix in careplan OpenAPI	Confirm actual deployed base path
`tasks`	Embedded array in plan response	Separate resource: `TaskCategory` objects under `Visit` objects	Frontend must navigate new structure

Action required for all shape mismatches: Add an adapter/transformer class in the BFF for each domain when wiring controllers (Phase 1). Do not rely on the real service coincidentally matching the mock shape.

DA-4. Post-Migration Schema Audit

After Phase 3 (per-domain cutover) completes for each domain, run a schema audit before disabling the mock fallback for that domain.

Add to Phase 3 per-domain checklist:

Step 6 (new) — Schema audit for domain being cut over:

```
# Example for visits domain — run from a machine with curl access to BFF
BFF_URL="https://<web-bff-fqdn>.azurecontainerapps.io"
AUTH_TOKEN="<valid-bearer-token>"

# 1. Fetch a real response
curl -s -H "Authorization: Bearer $AUTH_TOKEN" \
  "$BFF_URL/api/v1/visits?page=0&size=5" \
  -o /tmp/real-visits-response.json

# 2. Validate against OpenAPI spec using swagger-cli or similar
npx @apidevtools/swagger-cli validate \
  /Users/makinja/projects/client/lumiscare/openapi-specs/visits-open-api/open-api.yaml

# 3. Manual spot-check — verify required fields present
node -e "
  const data = require('/tmp/real-visits-response.json');
  const required =
['id','status','scheduledStartTime','scheduledEndTime','serviceUserId','carerId'];
  const first = data.content?.[0] || data[0] || {};
  required.forEach(f => {
    if (first[f] === undefined) console.log('MISSING:', f);
    else console.log('PRESENT:', f, '=', JSON.stringify(first[f]).slice(0, 40));
  });
"
```

Acceptance criteria for schema audit:

- All required fields from OpenAPI spec are present in real response
- No 500 errors in Azure Application Insights for the domain in the last 30 minutes
- Frontend renders the domain view without console errors (visual check by developer)

Gate: Do not disable mock fallback for a domain until schema audit passes.

DA-5. Rollback Test (Phase 2.4)

The existing rollback plan (Section 6) describes the mechanism but never tests it. A rollback that has never been tested is not a reliable rollback.

Add Phase 2.4 — Rollback Test (run after Phase 2.3, before Phase 3)

Procedure:

1. Confirm BFF fallback is enabled (`MOCK_API_FALLBACK_ENABLED=true`)
2. Confirm frontend is pointing to BFF URL (Phase 2.1 complete)
3. For one REAL_READY domain (recommended: Identity), disable the BFF route to real service:
   - Comment out or stub the real service client for that domain in BFF
   - Redeploy BFF (or use feature flag)
4. Verify BFF fallback activates:
   - `curl $BFF_URL/api/v1/service-users` should return mock data (HTTP 200, mock IDs like `"su-001"`)
   - Browser: service users page must render without error
5. Re-enable the real service client
6. Verify real data is served again
7. Document the rollback time (target: under 5 minutes end-to-end including BFF redeploy)

Pass criteria:

- Fallback activates within one BFF redeploy cycle
- Frontend renders correctly on mock data during fallback
- Re-enable returns real data with no stale cache issues

Gate: If rollback test fails, do NOT proceed to Phase 3. Fix the fallback mechanism first.

DA-6. Phase Environment Matrix

At each phase, different environments run on different data sources. This was implicit in the original plan — made explicit here.

Phase	Dev environment	Test environment	Staging environment	Production
Pre-migration (now)	Frontend → Mock direct	N/A	N/A	N/A
Phase 0 (prep)	Seed loaded in Azure dev DBs	Mock still running	Not affected	Not affected
Phase 1 (BFF wiring)	BFF code changes in dev branch	Mock fallback enabled	Not affected	Not affected
Phase 2 (frontend switch)	Frontend → BFF; BFF fallback ON for all routes	Frontend → BFF; BFF fallback ON	Not affected	Not affected

Phase	Dev environment	Test environment	Staging environment	Production
Phase 3 (per-domain cutover)	BFF routes real per domain; fallback OFF per domain after audit	BFF + fallback; promote domain-by-domain after dev validates	BFF + fallback per domain; reduced fallback set	Not affected until explicit prod migration
Phase 4 (decommission)	Mock stopped	Mock stopped	Mock stopped	N/A (mock never ran in prod)

Environment-specific flags:

```
# application-dev.yml
mock-api:
  fallback-url: ${MOCK_API_URL:http://ca-vcc-dev-mock-api:3000}
  fallback-enabled: true
  fallback-domains:
    visits: true          # flip to false after Phase 3 domain cutover
    scheduling: true
    notifications: true
    family: true          # stays true until FamilyPortalService exists
    misc: true

# application-test.yml
mock-api:
  fallback-url: ${MOCK_API_URL:http://ca-vcc-test-mock-api:3000}
  fallback-enabled: true
  fallback-domains:
    visits: true
    # ... test environment follows dev by ~1 sprint

# application-staging.yml
mock-api:
  fallback-enabled: false   # staging runs real services or explicit test data
  fallback-domains: {}      # no fallback in staging

# application-prod.yml
mock-api:
  fallback-enabled: false   # mock never deployed in prod
```

DA-7. Evidence Checklist per Domain (Cutover Gate)

For each domain cutover in Phase 3, the following evidence must be machine-generated before the mock fallback is disabled for that domain. "Works on my machine" and agent-reported PASS are not acceptable (ZAKON #21).

Template — fill out for each domain:

```
Domain: _______________
Date: _______________
Engineer: _______________

[ ] 1. seed-verify: DB row counts pass verify-seeds.sh for this domain's DB (exit 0)
[ ] 2. health-check: Container App status = Running
      Evidence: az containerapp show --name <ca-name> --query properties.runningStatus
[ ] 3. direct-service: Direct HTTP call to real service returns 200
      Evidence: curl -s -o /tmp/service-health.json -w "%{http_code}" http://<service-
internal-url>/actuator/health
[ ] 4. bff-proxy: BFF routes traffic to real service (not mock)
      Evidence: curl -s -H "Authorization: Bearer $TOKEN" $BFF_URL/api/v1/<domain-route> |
head -c 200
      Confirm: IDs are UUIDs, not mock format (e.g., "su-001")
[ ] 5. schema-audit: Required fields present in response (DA-4 procedure)
      Evidence: node schema-audit.js /tmp/<domain>-response.json
[ ] 6. no-500s: Zero 500 errors in Azure Application Insights for domain in last 30 min
      Evidence: az monitor app-insights query --app <app-insights-name> --analytics-query
"..."
[ ] 7. frontend-render: Domain UI renders without console errors
      Evidence: Screenshot or Playwright test output
[ ] 8. rollback-ready: MOCK_API_FALLBACK_ENABLED=true can be set within 5 min if needed
      Evidence: Rollback test passed (Phase 2.4)

PASS if all 8 checked. FAIL = keep mock fallback for this domain.
```

Minimum bar per domain:

- Identity: All 8 required (everything else depends on auth)
- Assessment, Care Plans, Incidents, HR, Finance: Items 1-7 required
- Visits, Scheduling, Notifications: Items 1-7 required
- Family Portal: Skip — stays on mock (see DA-8)

DA-8. Family Portal ADR (Architecture Decision Record)

Decision required before Phase 3 begins.

Context

The Family Portal has 16 mock endpoints (`/api/v1/family/*`). There is no `FamilyPortalService` in the deployed microservice architecture. Building a proper BFF layer for family portal requires aggregating from:

- `careplan-service` (read-only care plan view)
- `notification-service` (family notifications)
- `identity-service` (family member auth)
- An unbuilt payments integration

Options

Option A — Build FamilyPortalBffController in web-bff (or new family-bff)

- Effort: High (3-5 sprints)
- Requires: New BFF controller wiring 3+ services; family member auth flow; potentially new microservice
- Risk: Derails MVP timeline

Option B — Scope out Family Portal from MVP; keep on mock fallback

- Effort: Zero
- Mock fallback continues serving family routes indefinitely until a future sprint
- Family portal users (family members) see the mock data — acceptable for beta/internal demo
- Decision reversible

Decision

Recommendation: Option B — Scope Family Portal out of MVP migration.

Family portal features are non-critical for the primary care agency workflow. The MVP needs the back-office and mobile carer workflows operational. Family portal is a Phase 2 deliverable.

Actions:

1. Do NOT wire family portal routes to any real service during Phase 3
2. Keep `mock-api.fallback-domains.family: true` permanently through MVP launch
3. Add `POST /api/v1/family/*` routes to the `MOCK_ONLY` fallback list explicitly in BFF config

4. Create a new backlog item: "FamilyPortalService design + BFF wiring" — target: post-MVP sprint 1
5. Timeline target for family portal real wiring: 6-8 weeks post-MVP launch

This ADR is a CEO-level decision. Tag for explicit sign-off before Phase 3 begins.

DA-9. Demo Protection

During Phase 3 and leading up to any investor demo or client presentation, certain routes must remain stable. A real service returning unexpected data or a 500 error during a demo is unacceptable.

Demo-Critical Routes

These routes are rendered on the first screens visible during a standard demo:

```
GET  /api/v1/dashboard/stats
GET  /api/v1/service-users
GET  /api/v1/service-users/:id
GET  /api/v1/care-plans
GET  /api/v1/care-plans/:id
GET  /api/v1/assessments
GET  /api/v1/visits
GET  /api/v1/incidents
GET  /api/v1/notifications/inbox
GET  /api/v1/notifications/inbox/unread-count
```

DEMO_MODE Flag

Add a `DEMO_MODE` environment flag to the BFF:

```
# application.yml (BFF)
demo:
  mode: ${DEMO_MODE:false}
  # When true: all demo-critical routes use mock fallback regardless of domain cutover status
  # Overrides per-domain fallback-enabled settings for demo-critical routes only
  protected-routes:
    - /api/v1/dashboard/stats
    - /api/v1/service-users
    - /api/v1/service-users/**
    - /api/v1/care-plans
```

```
    - /api/v1/care-plans/**
    - /api/v1/assessments
    - /api/v1/visits
    - /api/v1/incidents
    - /api/v1/notifications/inbox
    - /api/v1/notifications/inbox/unread-count
```

BFF routing logic (pseudo-code):

```
// In BFF route handler
if (demoMode && demoProtectedRoutes.matches(request.getPath())) {
    return mockFallback.forward(request);  // always use mock for demo routes in DEMO_MODE
}
// Otherwise: normal real-service routing
return realService.forward(request);
```

48-Hour Stability Rule

For demo-critical routes, do not disable the mock fallback until the real service has been stable for 48 hours with zero 5xx errors. Evidence required: Azure Application Insights query showing error rate = 0 for 48h window.

```
# Check 48h stability for visits domain
az monitor app-insights query \
  --app <app-insights-resource> \
  --analytics-query "
    requests
    | where timestamp > ago(48h)
    | where url contains '/api/v1/visits'
    | where resultCode startswith '5'
    | count
  "
# Expected output: count = 0
```

Demo Runbook

Before any investor demo or client presentation:

1. Set `DEMO_MODE=true` in BFF Container App environment (Azure Portal or `az containerapp update`)
2. Verify dashboard loads with realistic mock data
3. Do NOT disable `DEMO_MODE` until demo is complete

4. After demo: reset `DEMO_MODE=false` to resume real-service routing

DA-1. Endpoint Priority Matrix

All 204 addressable mock endpoints classified by migration readiness. Wildcards and catch-alls excluded.

Classification key:

- **REAL_READY** — Spring Boot service has the endpoint AND BFF controller is wired
- **NEEDS_WIRING** — Service has the endpoint but BFF controller is missing or incomplete
- **MOCK_ONLY** — No real service implementation exists; keep mock fallback
- **SCOPE_OUT** — Feature is not needed for MVP; disable after migration without replacement

Visits (14 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/visits`	NEEDS_WIRING	visits-service `/visits` exists; BFF controller missing
`POST /api/v1/visits`	NEEDS_WIRING	visits-service `/visits` POST exists; BFF controller missing
`GET /api/v1/visits/:id`	NEEDS_WIRING	visits-service `/visits/{visitId}` exists
`PATCH /api/v1/visits/:id`	NEEDS_WIRING	visits-service PATCH exists
`DELETE /api/v1/visits/:id`	NEEDS_WIRING	visits-service DELETE exists
`POST /api/v1/visits/:id/check-in`	NEEDS_WIRING	visits-service check-in exists
`POST /api/v1/visits/:id/check-out`	NEEDS_WIRING	visits-service check-out exists
`GET /api/v1/visits/:id/emar`	NEEDS_WIRING	visits-service eMAR exists
`GET /api/v1/visits/:id/tasks`	NEEDS_WIRING	visits-service tasks list exists
`GET /api/v1/visits/today`	NEEDS_WIRING	visits-service `/visits/today` defined
`POST /api/v1/visits/publish`	MOCK_ONLY	No publish concept in visits-service OpenAPI
`POST /api/v1/visits/:id/running-late`	MOCK_ONLY	No equivalent in visits-service spec
`GET /api/v1/visits/:id/pre-visit-brief`	MOCK_ONLY	Aggregated BFF feature, not a service endpoint

Mock Endpoint	Classification	Notes
`POST /api/v1/visits/:id/travel-record`	MOCK_ONLY	Tracking endpoint in visits-service is different shape

Scheduling (8 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/schedule-blocks`	NEEDS_WIRING	Maps to `/schedule-events` in scheduling-service; CalendarController exists but not wired for this alias
`POST /api/v1/schedule-blocks`	NEEDS_WIRING	Maps to POST `/schedule-events`
`GET /api/v1/schedule-blocks/:id`	NEEDS_WIRING	Maps to `/schedule-events/{eventId}`
`PATCH /api/v1/schedule-blocks/:id`	NEEDS_WIRING	Maps to PUT `/schedule-events/{eventId}`
`DELETE /api/v1/schedule-blocks/:id`	NEEDS_WIRING	Maps to DELETE `/schedule-events/{eventId}`
`GET /api/v1/completed-events`	NEEDS_WIRING	Maps to `/appointments` with `status=completed` filter
`POST /api/v1/completed-events/:id/confirm`	NEEDS_WIRING	Maps to POST `/appointments/{id}/cancel` pattern (status update)
`POST /api/v1/completed-events/bulk-confirm`	MOCK_ONLY	No bulk-confirm operation in scheduling-service spec

Assessment (12 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/assessments`	REAL_READY	BffAssessmentController wired
`GET /api/v1/assessments/:id`	REAL_READY	BffAssessmentController wired
`POST /api/v1/assessments`	REAL_READY	BffAssessmentController wired
`PUT /api/v1/assessments/:id`	REAL_READY	BffAssessmentController wired
`PATCH /api/v1/assessments/:id`	REAL_READY	BffAssessmentController wired
`POST /api/v1/assessments/:id/complete`	REAL_READY	Maps to assessment submit endpoint
`GET /api/v1/assessments/:id/body-map`	NEEDS_WIRING	assessment-service has body-map section; BFF wiring unconfirmed
`PUT /api/v1/assessments/:id/body-map`	NEEDS_WIRING	as above

Mock Endpoint	Classification	Notes
`GET /api/v1/assessments/:id/scores`	MOCK_ONLY	No scoring endpoint in assessment-service spec; maps to risk-section concept
`GET /api/v1/assessments/:id/scores/:tool`	MOCK_ONLY	No tool-specific scores in spec
`POST /api/v1/assessments/:id/scores/:tool`	MOCK_ONLY	No tool-specific scores in spec
`GET /api/v1/intake`	REAL_READY	assessment-service `/api/v1/intake` matches

Care Plans (14 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/care-plans`	REAL_READY	CarePlanController wired
`GET /api/v1/care-plans/:id`	REAL_READY	CarePlanController wired
`POST /api/v1/care-plans`	REAL_READY	CarePlanController wired
`PUT /api/v1/care-plans/:id`	REAL_READY	CarePlanController wired
`DELETE /api/v1/care-plans/:id`	REAL_READY	CarePlanController wired
`POST /api/v1/care-plans/:id/publish`	REAL_READY	careplan-service publish endpoint exists
`GET /api/v1/care-plans/:id/versions`	REAL_READY	careplan-service versions endpoint exists
`GET /api/v1/care-plans/review-due`	MOCK_ONLY	No review-due filter in careplan-service spec
`POST /api/v1/care-plans/:id/submit-for-approval`	MOCK_ONLY	No submit-for-approval step in real spec
`GET /api/v1/care-plans/:id/tasks`	MOCK_ONLY	Task management not in careplan-service OpenAPI
`GET /api/v1/care-plans/:id/risk-alerts`	MOCK_ONLY	No risk-alerts endpoint in careplan-service
`GET /api/v1/care-plans/:id/reviews`	MOCK_ONLY	No reviews endpoint in careplan-service
`POST /api/v1/care-plans/:id/consent`	MOCK_ONLY	No consent endpoint in careplan-service
`GET /api/v1/master-task-types`	REAL_READY	careplan-service has master-task-types

HR (18 endpoints)

Mock Endpoint	Classification	Notes
`GET /employees/list`	REAL_READY	EmployeeController wired (legacy path)
`GET /employees/:id`	REAL_READY	EmployeeController wired
`POST /employees`	REAL_READY	EmployeeController wired
`PATCH /employees/:id`	REAL_READY	EmployeeController wired
`DELETE /employees/:id`	REAL_READY	EmployeeController wired
`GET /api/v1/leaves`	REAL_READY	hr-service leaves endpoint exists; BFF wired
`POST /api/v1/leaves`	REAL_READY	hr-service leaves POST exists
`PATCH /api/v1/leaves/:id`	REAL_READY	hr-service leaves PATCH exists
`DELETE /api/v1/leaves/:id`	REAL_READY	hr-service leaves DELETE exists
`GET /api/v1/hr/supervisions`	NEEDS_WIRING	hr-service has supervisions; BFF sub-route not wired
`GET /api/v1/hr/supervision-schedule`	NEEDS_WIRING	hr-service has supervision data; path mapping needed
`GET /api/v1/hr/absences`	NEEDS_WIRING	hr-service has absence/leave data; path alias needed
`GET /api/v1/hr/training`	NEEDS_WIRING	hr-service has training; BFF route not wired
`GET /api/v1/training/courses`	NEEDS_WIRING	hr-service training module; BFF path different
`GET /api/v1/hr/dbs`	MOCK_ONLY	DBS checks not in hr-service OpenAPI as named endpoint
`GET /api/v1/hr/bradford`	MOCK_ONLY	Calculated aggregate; no endpoint in hr-service
`GET /api/v1/hr/care-certificate`	MOCK_ONLY	Not in hr-service OpenAPI
`GET /api/v1/hr/cqc-inspection-pack`	SCOPE_OUT	Complex regulatory aggregate; not MVP

Finance (20 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/finance/invoices`	REAL_READY	FinanceBffController wired
`GET /api/v1/finance/invoices/:id`	REAL_READY	FinanceBffController wired
`POST /api/v1/finance/invoices`	REAL_READY	FinanceBffController wired
`PUT /api/v1/finance/invoices/:id`	REAL_READY	FinanceBffController wired

Mock Endpoint	Classification	Notes
`GET /api/v1/finance/invoice-groups`	REAL_READY	finance-service invoice-groups; BFF wired
`POST /api/v1/finance/invoice-groups`	REAL_READY	finance-service wired
`GET /api/v1/finance/payment-groups`	REAL_READY	finance-service wired
`POST /api/v1/finance/payment-groups`	REAL_READY	finance-service wired
`GET /api/v1/finance/timesheets`	NEEDS_WIRING	finance-service has timesheets; BFF sub-route unconfirmed
`GET /api/v1/finance/dashboard`	NEEDS_WIRING	finance-service dashboard exists; aggregation needed
`GET /api/v1/finance/rate-cards`	MOCK_ONLY	Not in finance-service OpenAPI
`GET /api/v1/finance/funders`	MOCK_ONLY	Not in finance-service OpenAPI
`GET /api/v1/finance/payroll/preview`	MOCK_ONLY	Payroll is separate domain; not in MVP
`POST /api/v1/finance/payroll/approve`	MOCK_ONLY	Payroll not in MVP
`GET /api/v1/finance/payroll/export`	SCOPE_OUT	Out of MVP scope
`GET /api/v1/finance/aged-debtors`	MOCK_ONLY	Calculated from invoices; no dedicated endpoint
`GET /api/v1/finance/nmw-compliance`	SCOPE_OUT	Compliance report; out of MVP scope
`GET /api/v1/finance/month-end-status`	MOCK_ONLY	No equivalent in finance-service spec
`GET /api/v1/finance/bank-holidays`	MOCK_ONLY	Static data; serve from BFF config
`GET /api/v1/invoices`	REAL_READY	Legacy path; maps to finance-service invoices

Incidents (8 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/incidents`	NEEDS_WIRING	Real service has typed routes (`/accidents` , `/safeguarding` , `/complaints` , `/hazards`); BFF must aggregate or route by type
`GET /api/v1/incidents/:id`	NEEDS_WIRING	BFF must determine type and proxy to correct sub-service
`POST /api/v1/incidents`	NEEDS_WIRING	BFF must route by `type` field to correct incident sub-resource
`PATCH /api/v1/incidents/:id`	NEEDS_WIRING	Same routing requirement
`DELETE /api/v1/incidents/:id`	NEEDS_WIRING	Same routing requirement

Mock Endpoint	Classification	Notes
`GET /api/v1/incidents/summary`	REAL_READY	incidents-service `/summary` endpoint exists; IncidentsController wired
`POST /api/v1/incidents/:id/actions`	NEEDS_WIRING	incidents-service log-actions endpoint exists; BFF wiring unconfirmed
`GET /api/v1/incidents/export`	SCOPE_OUT	Export for each sub-type exists in spec; aggregate export not MVP

Notifications (10 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/notifications/inbox`	NEEDS_WIRING	notification-service inbox exists; BFF controller missing entirely
`GET /api/v1/notifications/inbox/unread-count`	NEEDS_WIRING	notification-service unread-count exists; BFF missing
`PATCH /api/v1/notifications/inbox/:id/read`	NEEDS_WIRING	notification-service mark-read exists; BFF missing
`PATCH /api/v1/notifications/inbox/read-all`	NEEDS_WIRING	notification-service mark-all-read exists; BFF missing
`GET /api/v1/notifications/preferences`	NEEDS_WIRING	notification-service preferences exists; BFF missing
`POST /api/v1/notifications/preferences`	NEEDS_WIRING	notification-service preferences POST exists; BFF missing
`PATCH /api/v1/notifications/preferences/quiet-hours`	NEEDS_WIRING	notification-service; BFF missing
`POST /api/v1/notifications/devices`	NEEDS_WIRING	notification-service device registration; BFF missing
`POST /api/v1/notifications/push`	NEEDS_WIRING	notification-service send endpoint; BFF missing
`GET /api/v1/notifications/analytics`	MOCK_ONLY	Not in notification-service spec

Identity / Users / Orgs / Service Users (14 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/service-users`	REAL_READY	ServiceUsersController wired; identity-service backs it

Mock Endpoint	Classification	Notes
`GET /api/v1/service-users/:id`	REAL_READY	ServiceUsersController wired
`POST /api/v1/service-users`	REAL_READY	ServiceUsersController wired
`PATCH /api/v1/service-users/:id`	REAL_READY	ServiceUsersController wired
`DELETE /api/v1/service-users/:id`	REAL_READY	ServiceUsersController wired
`GET /api/v1/users`	NEEDS_WIRING	identity-service users endpoint exists; BFF route may not be exposed
`GET /api/v1/organizations`	NEEDS_WIRING	identity-service orgs endpoint exists; BFF route unconfirmed
`GET /api/v1/organizations/:id`	NEEDS_WIRING	identity-service; BFF unconfirmed
`GET /api/v1/settings`	NEEDS_WIRING	identity-service org settings; BFF route unconfirmed
`PATCH /api/v1/settings`	NEEDS_WIRING	identity-service org settings PATCH; BFF unconfirmed
`GET /api/v1/carers`	NEEDS_WIRING	Maps to identity-service users with carer role filter
`POST /api/v1/carers/by-skills`	MOCK_ONLY	No skills-based search in identity-service spec
`GET /api/v1/skills`	MOCK_ONLY	No dedicated skills list endpoint in identity-service
`GET /api/v1/rostering-settings`	MOCK_ONLY	Not in identity-service spec

Family Portal (16 endpoints)

Mock Endpoint	Classification	Notes
ALL `GET /api/v1/family/*`	MOCK_ONLY	No FamilyPortalService exists. See DA-8 ADR below.

Misc / EVV / Dashboard / Compliance (24 endpoints)

Mock Endpoint	Classification	Notes
`GET /api/v1/dashboard/stats`	NEEDS_WIRING	Aggregate from multiple services; BFF orchestration needed
`GET /api/v1/cqc-readiness`	MOCK_ONLY	Aggregate; not in any service spec
`GET /api/v1/evv/compliance`	NEEDS_WIRING	Maps to visits-service tracking data; BFF aggregation needed

Mock Endpoint	Classification	Notes
`GET /api/v1/evv/records`	NEEDS_WIRING	Maps to visits-service location tracking history
`GET /api/v1/evv/records/:id`	NEEDS_WIRING	Maps to visits-service tracking detail
`GET /api/v1/audit-logs`	MOCK_ONLY	No audit service in current architecture
`GET /api/v1/insurance/claims`	SCOPE_OUT	Not in architecture; park for post-MVP
`GET /api/v1/insurance/payers`	SCOPE_OUT	Not in architecture; park for post-MVP
`POST /api/v1/handoff/notes`	NEEDS_WIRING	Maps to visits-service care notes with type=handover
`GET /api/v1/handoff/history`	NEEDS_WIRING	Maps to visits-service care notes filtered by type
`POST /api/v1/handoff/notes/:id/acknowledge`	MOCK_ONLY	No acknowledgement concept in visits-service notes
`GET /api/v1/observations`	NEEDS_WIRING	Maps to assessment-service or visits-service observations
`GET /api/v1/supervisor/dashboard`	MOCK_ONLY	Aggregate view; BFF composition needed post-MVP
`GET /api/v1/supervisor/dashboard/carer-locations`	NEEDS_WIRING	Maps to visits-service `/tracking/locations/current`
`POST /api/v1/alerts/:id/acknowledge`	MOCK_ONLY	No alert acknowledgement service
`GET /api/v1/clients/:id/outcomes`	MOCK_ONLY	No outcomes service
`GET /api/v1/clients/:id/care-summary`	MOCK_ONLY	BFF aggregate; not MVP priority
`GET /api/v1/training/compliance`	NEEDS_WIRING	Maps to hr-service training compliance data
`GET /api/v1/care-alerts/:id/acknowledge`	MOCK_ONLY	No care-alerts service
ALL others (wildcards)	SCOPE_OUT	Catch-all routes; decommission with mock

Priority Matrix Summary

Classification	Count	Action
REAL_READY	~47	Verify in Phase 3; disable mock fallback per-route
NEEDS_WIRING	~68	Phase 1 + Phase 3 BFF work
MOCK_ONLY	~63	Keep mock fallback; build real service post-MVP

Classification	Count	Action
SCOPE_OUT	~26	Disable with no replacement; document as out-of-scope

---

DA-2. Seed Data Verification (Phase 0.4)

Add this phase between Phase 0.1 (seed load) and Phase 0.2 (container health check).

Phase 0.4 — Verify Seed Data Loaded Correctly

After running seed scripts via Azure Cloud Shell, verify row counts before proceeding to Phase 1.

Expected minimum row counts per database:

Database	Table	Minimum Rows	Purpose
`identity_db`	`organizations`	1	Test org
`identity_db`	`users`	5	1 admin, 1 care-manager, 3 carers
`identity_db`	`service_users`	3	Test clients
`identity_db`	`roles`	12	Pre-seeded system roles
`assessment_db`	`assessment_sections`	7	ASSESSMENT_SECTION_DEFS equivalent
`assessment_db`	`risk_sections`	6	RISK_SECTION_DEFS equivalent
`careplan_db`	`master_task_types`	10	Task type catalogue
`careplan_db`	`master_visit_types`	4	Visit type catalogue
`visits_db`	`visit_statuses`	7	scheduled, late, in_progress, completed, cancelled, missed, flagged
`hr_db`	`leave_types`	5	Annual, sick, maternity, etc.
`hr_db`	`training_categories`	4	Mandatory, CPD, compliance, specialist
`scheduling_db`	`appointment_types`	4	care_visit, assessment_visit, medication_only, welfare_check

Database	Table	Minimum Rows	Purpose
`finance_db`	`invoice_statuses`	5	Draft, issued, paid, overdue, cancelled
`incidents_db`	`incident_type_config`	4	accidents, safeguarding, complaints, hazards
`notification_db`	`notification_templates`	5	Core template types

Verification queries — run in Azure Cloud Shell after seed:

```
-- identity_db
SELECT 'organizations' as tbl, count(*) FROM organizations
UNION ALL SELECT 'users', count(*) FROM users
UNION ALL SELECT 'service_users', count(*) FROM service_users
UNION ALL SELECT 'roles', count(*) FROM roles;

-- assessment_db
SELECT 'assessment_sections', count(*) FROM assessment_sections
UNION ALL SELECT 'risk_sections', count(*) FROM risk_sections;

-- careplan_db
SELECT 'master_task_types', count(*) FROM master_task_types
UNION ALL SELECT 'master_visit_types', count(*) FROM master_visit_types;

-- visits_db
SELECT 'visit_task_types', count(*) FROM visit_task_types;

-- incidents_db
SELECT table_name, 0 as dummy FROM information_schema.tables
WHERE table_schema = 'public'; -- verify tables exist

-- finance_db
SELECT 'invoice_line_items_config', count(*) FROM invoice_statuses;
```

Verify-seeds script spec:

Create `/Users/makinja/projects/client/lumiscare/scripts/verify-seeds.sh`:

```
#!/bin/bash
# verify-seeds.sh — run from Azure Cloud Shell after seed load
# Usage: bash verify-seeds.sh <pg-host> <admin-user>
```

```
PG_HOST="$1"
PG_USER="$2"
FAIL=0

check() {
  local DB=$1 TBL=$2 MIN=$3
  COUNT=$(psql "host=$PG_HOST user=$PG_USER dbname=$DB sslmode=require" \
    -tAc "SELECT count(*) FROM $TBL 2>/dev/null || echo 0")
  if [ "$COUNT" -lt "$MIN" ]; then
    echo "FAIL: $DB.$TBL has $COUNT rows (expected >= $MIN)"
    FAIL=1
  else
    echo "PASS: $DB.$TBL = $COUNT rows"
  fi
}

check identity_db organizations 1
check identity_db users 5
check identity_db service_users 3
check assessment_db assessment_sections 7
check careplan_db master_task_types 10
check visits_db visit_task_types 4
check finance_db invoice_statuses 3
check incidents_db incident_categories 4

[ $FAIL -eq 0 ] && echo "ALL SEED CHECKS PASSED" || { echo "SEED VERIFICATION FAILED — do not
proceed to Phase 1"; exit 1; }
```

Seed verification must pass before Phase 1 work begins. If any check fails, re-run the corresponding seed script.

DA-3. Shape Mismatch Matrix

Critical field-level differences between mock API response shapes and real OpenAPI-specified shapes. BFF adapter layer must translate these on cutover.

Visits — GET /api/v1/visits

Field	Mock Shape	Real Service Shape (visits-service OpenAPI)	BFF Action
List wrapper	`[...visits]` (bare array)	`{ "visits": [...], "pageInfo": { "total": N, "page": 0, "size": 20 } }`	BFF must unwrap `visits` array for frontend; frontend expects array
`id`	`"visit-0001"` (string, dash-prefixed)	UUID format `"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"`	ID format will change; frontend must handle UUID
`clientId`	`"su-001"` (string, dash-prefixed)	`serviceUserId` (UUID)	Field rename: `clientId` → `serviceUserId`
`clientName`	`"Dorothy Johnson"` (string in visit object)	Not in visit object — join required from identity-service	BFF must enrich with identity-service lookup or accept that frontend needs separate call
`status`	`"in-progress"` (kebab-case)	`"in_progress"` (snake_case enum: scheduled, late, in_progress, completed, cancelled, missed, flagged)	BFF must normalize status values
`type`	`"personal-care"` (kebab-case, free string)	`appointmentType` (enum: care_visit, assessment_visit, medication_only, welfare_check)	Field rename + value normalization
`gpsCheckIn`	`{ lat: 51.5074, lng: -0.1278 }`	`{ "latitude": 51.5074, "longitude": -0.1278, "accuracy": 5.0, "timestamp": "..." }` (GpsLocation schema)	Field rename: `lat`→`latitude`, `lng`→`longitude`; add required fields
`careNotes`	String field on visit object	Separate endpoint `GET /visits/{visitId}/notes` → `[{ "noteType": "...", "content": "..." }]`	Notes must be fetched separately; BFF may aggregate or frontend must call separately
`actualCheckIn` / `actualCheckOut`	ISO string on visit object	Nested in check-in/check-out response objects, not returned on visit GET	BFF must compose from visit + lifecycle data

Scheduling — GET /api/v1/schedule-blocks

Field	Mock Shape	Real Service Shape (scheduling-service OpenAPI)	BFF Action
Resource name	`schedule-blocks` (array of blocks)	`schedule-events` (ScheduleEvent objects)	Path alias handled by BFF

Field	Mock Shape	Real Service Shape (scheduling-service OpenAPI)	BFF Action
`type`	`"team-meeting"`, `"training"`, `"supervision"`, `"appraisal"` (kebab-case)	`ScheduleEventType` enum: `meeting`, `training`, `supervision`, `appraisal` (no dash, no team- prefix)	BFF must normalize `team-meeting` → `meeting`
`duration`	Integer (minutes)	Not a top-level field; calculated from `startDateTime` and `endDateTime`	BFF may compute duration from datetime diff
`startTime` / `endTime`	ISO string	`startDateTime` / `endDateTime` (ISO 8601 with timezone)	Field rename in BFF adapter
List wrapper	Bare array	Paginated: `{ "content": [...], "page": 0, "size": 20, "totalElements": N, "totalPages": N }`	BFF extracts `content` or adapts to paginated format

Incidents — GET /api/v1/incidents

Field	Mock Shape	Real Service Shape (incidents-service OpenAPI)	BFF Action
List wrapper	`{ "incidents": [...], "total": N }`	Per type: `{ "content": [...], "page": 0, "totalElements": N, "totalPages": N }`	BFF must aggregate responses from `/accidents`, `/safeguarding`, `/complaints`, `/hazards`
`id`	`"inc-001"` (string)	UUID (auto-generated)	ID format change; BFF cannot map old IDs
`type`	`"Accident"`, `"Medication_Error"`, `"Safeguarding"`	Separate resources by route; type determined by endpoint (`/accidents` vs `/safeguarding`)	BFF adds `type` field when aggregating
`incident_number`	`"INC-2024-001"`	`referenceNumber` field: `"ACC-2026-00001"` (typed prefix format)	Field rename + prefix format changes
`severity`	`"High"`, `"Medium"`, `"Critical"` (title-case)	`SeverityLevel` enum: `HIGH`, `MEDIUM`, `LOW`, `CRITICAL` (UPPER_CASE)	BFF normalizes to uppercase
`status`	`"Resolved"`, `"Under_Review"` (mixed case)	`LogStatus` enum: `IDENTIFIED`, `RAISED`, `IN_PROGRESS`, `RESOLVED` etc. (UPPER_CASE)	BFF normalizes case; `Under_Review` → `IN_PROGRESS`
`dateAndTime`	ISO string	`incidentDatetime` (ISO 8601 with Z suffix)	Field rename in BFF

Field	Mock Shape	Real Service Shape (incidents-service OpenAPI)	BFF Action
`actionsTaken`	Array on the incident object	Separate endpoint: `GET /logs/{logType}/{logId}/actions`	Actions must be fetched separately
`has_investigation`	Boolean	Not in incidents-service OpenAPI — derived from status	BFF computes from status

Care Plans — GET /api/v1/care-plans

Field	Mock Shape	Real Service Shape (careplan-service OpenAPI)	BFF Action
`clientId`	`"su-001"`	`clientId` (UUID)	Same field name; ID format changes to UUID
`status`	`"active"`, `"draft"`, `"under_review"`	`CarePlanStatus` enum: `draft`, `ai_generated`, `under_review`, `approved`, `active`, `suspended`, `completed`, `cancelled`	Values align but `ai_generated` is new — frontend must handle
Tasks nested	`tasks: [...]` array on care plan	Tasks NOT on care plan object in careplan-service spec; separate route `/care-plan/{id}/tasks`	BFF may aggregate or frontend must call separately
Visit structure	`visits: [...]` array	`CarePlanDetailsResponse` has `Visit[]` with nested `TaskCategory[]` and `CarePlanTask[]`	Nested structure differs significantly from mock flat array

Assessment — GET /api/v1/assessments/:id

Field	Mock Shape	Real Service Shape (assessment-service OpenAPI)	BFF Action
`id` vs `assessmentId`	Both `id` and `assessmentId` present (legacy dual fields)	`assessmentId` only (UUID)	Remove dual field; use `assessmentId` consistently
`patientId`	UUID string	`patientId` (UUID) — matches	No change needed
`completionStatus`	`"IN_PROGRESS"`, `"COMPLETED"`	`completionStatus` enum per spec	Matches — no change
`lastUpdated`	Concatenated date + literal `T10:30:00Z` suffix	ISO 8601 datetime from service	Remove hardcoded suffix hack

Field	Mock Shape	Real Service Shape (assessment-service OpenAPI)	BFF Action
`overview`	Nested object with `riskLevel`, `allergies`, etc.	Not a direct response field — derived from assessment sections	BFF must compose overview from section data

---

DA-4. Schema Validation (Post-Migration Audit Step)

Add as Phase 3.5, after all per-domain cutovers complete but before Phase 4 decommission.

Phase 3.5 — Schema Audit

Objective: Machine-verify that every active BFF response matches its OpenAPI contract. No visual inspection.

Step 1 — Generate JSON Schemas from OpenAPI specs:

```
# Using openapi-schema-validator or ajv-openapi
npx @apidevtools/swagger-parser validate \
  /Users/makinja/projects/client/lumiscare/openapi-specs/visits-open-api/open-api.yaml
# Repeat for each service spec
```

Step 2 — Record BFF responses against schema:

For each REAL_READY or newly NEEDS_WIRING domain, run automated contract tests:

```
# Install dredd or schemathesis
pip install schemathesis

# Run against BFF with real backend:
schemathesis run \
  /Users/makinja/projects/client/lumiscare/openapi-specs/visits-open-api/open-api.yaml \
  --base-url https://<web-bff-fqdn>.azurecontainerapps.io \
  --auth-type bearer --auth "$JWT_TOKEN" \
  --header "X-VCU-Organization-Id: $ORG_ID"
```

Step 3 — Adapter correction loop:

For each schema violation found:

1. Identify if the mismatch is in the BFF adapter or the service response
2. Fix the BFF adapter layer (preferred) or raise a service ticket
3. Re-run schemathesis until zero schema violations remain

Step 4 — Acceptance criteria:

- Zero schemathesis ERRORS (hard failures) for all REAL_READY routes
- WARNINGS documented and accepted or fixed before Phase 4
- Schema audit report saved to `/Users/makinja/projects/client/lumiscare/docs/schema-audit-report.md`

Pass criteria for Phase 4 entry: Schema audit exits with code 0 for all 10 domains.

DA-5. Rollback Test (Phase 2.4)

Add between Phase 2.3 (keep mock running) and Phase 3 (per-domain cutover).

Phase 2.4 — Validate Rollback Mechanism Works

Before committing to Phase 3 cutovers, prove the rollback path is functional. This is a one-time gate check.

2.4.1 Simulate a domain failure:

```
# Disable visits-service in BFF by setting a bad URL (or scale to 0)
az containerapp update \
  --name ca-vcc-dev-visits-service-001 \
  --resource-group rg-vcc-dev-001 \
  --min-replicas 0 --max-replicas 0
```

2.4.2 Verify BFF fallback activates automatically:

```
# Call a visits endpoint through BFF — should return mock data, not 502
curl -s -o /dev/null -w "%{http_code}" \
  -H "Authorization: Bearer $JWT_TOKEN" \
  https://<web-bff-fqdn>.azurecontainerapps.io/api/v1/visits
# Expected: 200 (served from mock fallback)
```

2.4.3 Verify fallback data is recognizable (not silent corruption):

```
# Response should include mock data markers (e.g., "visit-0001" style IDs)
curl -s -H "Authorization: Bearer $JWT_TOKEN" \
  https://<web-bff-fqdn>.azurecontainerapps.io/api/v1/visits \
  | jq '.[0].id'
# Expected: "visit-0001" or similar mock ID pattern
```

2.4.4 Re-enable the service and confirm real data returns:

```
az containerapp update \
  --name ca-vcc-dev-visits-service-001 \
  --resource-group rg-vcc-dev-001 \
  --min-replicas 1 --max-replicas 3

# Wait for healthy status, then re-test
curl -s -H "Authorization: Bearer $JWT_TOKEN" \
  https://<web-bff-fqdn>.azurecontainerapps.io/api/v1/visits \
  | jq '.[0].id'
# Expected: UUID format (real service data)
```

2.4.5 Document rollback timings:

Rollback Type	Expected Time	Acceptable Max
BFF env var toggle (`MOCK_API_FALLBACK_ENABLED=true`)	< 2 minutes	5 minutes
Per-domain BFF route disable (redeploy BFF)	3-5 minutes	10 minutes
Frontend env var change (Static Web App)	< 5 minutes	15 minutes
Database restore from Azure backup	30-60 minutes	2 hours

Gate: Phase 2.4 must PASS before any Phase 3 cutover proceeds.

DA-6. Phase Timing and Environment Clarity

Which systems run on what at each migration phase, per environment.

Environment Matrix by Phase

Phase	Dev	Test	Staging	Production
Current (pre-migration)	Frontend → Mock API direct	Frontend → Mock API direct	N/A	N/A
Phase 0 (seed + prep)	Mock API still direct; seeds loaded to Azure DBs	No change	No change	No change
Phase 1 (BFF wiring)	BFF development; Mock API still used by frontend	Mock API still used	No change	No change
Phase 2 (frontend switch)	**Frontend → BFF** (BFF → real service OR mock fallback)	Frontend → BFF (all fallback)	BFF deployed, fallback enabled for all routes	Not deployed
Phase 3 (per-domain cutover)	BFF → real services per-domain; fallback only for MOCK_ONLY	BFF + per-domain fallback; real services tested here	BFF + fallback per-domain; each domain promoted after test passes	Not deployed
Phase 4 (decommission)	BFF → real services; fallback disabled; mock stopped	Mock stopped	Mock stopped	Deploy begins post-stability

Service URL Routing by Environment

```
Dev:     Frontend → https://<bff-dev>.azurecontainerapps.io → {service}-dev → identity_db_dev
Test:    Frontend → https://<bff-test>.azurecontainerapps.io → {service}-test → identity_db_test
Staging: Frontend → https://<bff-stage>.azurecontainerapps.io → {service}-stage → identity_db_stage
Prod:    Frontend → https://<bff-prod>.azurecontainerapps.io → {service}-prod → identity_db_prod
```

Fallback Configuration by Environment

Environment	`MOCK_API_FALLBACK_ENABLED`	`MOCK_API_URL`	Notes
Dev	`true` during Phase 1-3; `false` at Phase 4	`http://mock-api-dev:3000`	Toggle per domain using route-level flags
Test	`true` until domain passes test suite; then `false`	`http://mock-api-test:3000`	Automated test gate controls the toggle

Environment	`MOCK_API_FALLBACK_ENABLED`	`MOCK_API_URL`	Notes
Staging	`true` for 48h stability window per domain; then `false`	`http://mock-api-stage:3000`	See DA-9 Demo Protection
Production	`false` from day one (start clean)	N/A	No mock in production ever

---

DA-7. Evidence Checklist per Domain (Cutover Gate)

For each domain in Phase 3, the following machine-verified criteria must ALL pass before disabling the mock fallback for that domain. Evidence artifacts must be saved.

Cutover Checklist Template

Replace `{DOMAIN}` with the domain name (e.g., `assessment`, `visits`).

```
[ ] 1. Seed data verified — verify-seeds.sh passes for {DOMAIN}_db
[ ] 2. Service health — GET /actuator/health returns {"status":"UP"} (HTTP 200)
[ ] 3. BFF route active — GET /api/v1/{domain-path} returns HTTP 200 (not 502 or fallback)
[ ] 4. Auth propagation — JWT + org header accepted; 401 returned without token
[ ] 5. Data shape — schemathesis or equivalent schema check passes (zero violations)
[ ] 6. CRUD smoke — POST creates, GET retrieves, PATCH updates, DELETE removes (where
applicable)
[ ] 7. Pagination — List endpoint returns paginated wrapper, not raw array
[ ] 8. Tenant isolation — org-A data not visible when calling with org-B token
[ ] 9. Error format — 404 returns standardized error schema, not mock `{ error: "Not found" }`
[ ] 10. Fallback disabled — MOCK_API_FALLBACK_ENABLED=false for this domain; mock NOT serving
requests
[ ] 11. Playwright smoke tests — existing frontend test suite passes against BFF (no
regressions)
[ ] 12. Rollback verified — fallback re-enable tested and confirmed working (DA-5 gate)
```

Evidence artifacts per domain:

Artifact	Format	Location
verify-seeds output	`.txt` log	`/docs/migration-evidence/{domain}-seed-verify.txt`

Artifact	Format	Location
schemathesis report	JSON or HTML	`/docs/migration-evidence/{domain}-schema-report.json`
Playwright test report	JUnit XML or HTML	CI artifact
Cutover timestamp	ISO 8601 string	`/docs/migration-evidence/{domain}-cutover.json`

Domain cutover sign-off: All 12 checks must show machine-generated PASS. No self-certification. DevOps engineer reviews artifact paths before toggling `MOCK_API_FALLBACK_ENABLED=false`.

Domain Priority and Cutover Order

Priority	Domain	Blocker For	Estimated Cutover Date (relative)
1	Identity	All others — auth depends on it	Week 1 Day 1
2	Assessment	Care Plans, clinical workflow	Week 1 Day 3
3	Care Plans	Visits task planning	Week 1 Day 5
4	Visits	Mobile carers, eMAR	Week 2 Day 1
5	Incidents	Safeguarding compliance	Week 2 Day 2
6	HR	Staff compliance reporting	Week 2 Day 3
7	Scheduling	Calendar view, rostering	Week 2 Day 4
8	Notifications	Non-blocking but improves UX	Week 2 Day 5
9	Finance	Invoicing, billing	Week 3 Day 1
10	Family Portal	MOCK_ONLY — stays on mock	Post-MVP

DA-8. Family Portal Architecture Decision Record (ADR-001)

Date: 2026-03-26 **Status:** ACCEPTED **Deciders:** Engineering Lead, Product Owner

Context

The Family Portal frontend (`GET /api/v1/family/*`) has 16 mock endpoints. There is no dedicated `FamilyPortalService` in the current microservice architecture. The endpoints aggregate data from:

- `careplan-service` (care plan view)
- `notification-service` (messages, notifications)
- `identity-service` (family member auth, payments)

No `FamilyPortalController` exists in the Web BFF. Building a full family portal BFF layer during the mock-to-real migration would be a parallel new-feature build, not a migration task.

Decision

Family Portal routes are SCOPED OUT of the mock-to-real MVP migration.

The 16 family portal mock endpoints will remain served by the mock API fallback indefinitely until a dedicated post-MVP sprint is resourced.

Rationale

1. Zero real service coverage — building from scratch, not migrating
2. Requires new BFF controller + aggregation logic across 3+ services
3. Family Portal is a secondary product — primary migration priority is Back Office and Mobile
4. The mock API fallback handles family portal routes with zero user impact during migration

Post-MVP Implementation Plan

Phase	Work	Timeline
FP-1	Define `FamilyPortalController` in Web BFF; aggregate careplan + identity endpoints	Sprint after MVP cutover
FP-2	Wire notification-service for family inbox + message threads	FP-1 + 1 sprint
FP-3	Payment initiation (Stripe/payment gateway) — separate investigation	TBD
FP-4	Family Portal authentication (separate Entra B2C tenant for non-staff)	TBD — security review required

Consequences

- **Positive:** Migration timeline stays at ~3 weeks; no scope creep
- **Positive:** Family Portal gets proper design attention post-MVP

- **Negative:** Mock data continues serving family portal users during MVP period
- **Accepted risk:** Mock data quality for family portal is sufficient for demo and pilot; real families not on-boarded until FP phases complete

Mock Fallback Route Config

```
# application.yml — BFF fallback config
mock-api:
  fallback-routes:
    - /api/v1/family/**   # permanent until FP-1 sprint
```

DA-9. Demo Protection Protocol

Context

During Phase 3 cutovers, demo sessions may be running against staging or dev environments. A failed real service during a demo is unacceptable. This protocol defines demo-critical routes and a `DEMO_MODE` fallback concept.

Demo-Critical Routes

Routes that, if broken, will visibly fail a standard stakeholder demo:

Route	Why Critical
`GET /api/v1/dashboard/stats`	First screen shown in every demo
`GET /api/v1/service-users`	Client list — primary navigation
`GET /api/v1/visits`	Visit calendar — core value demo
`GET /api/v1/care-plans`	Care plan list — clinical workflow demo
`GET /api/v1/assessments`	Assessment list — intake workflow demo
`GET /api/v1/notifications/inbox`	Notification bell — real-time demo
`GET /api/v1/incidents`	Incident log — compliance demo
`GET /api/v1/family/*`	Family portal views — stakeholder demo

DEMO_MODE Flag

Add a `DEMO_MODE` environment variable to the BFF:

```
# application.yml
demo:
  mode: ${DEMO_MODE:false}
  protected-routes:
    - /api/v1/dashboard/stats
    - /api/v1/service-users
    - /api/v1/visits
    - /api/v1/care-plans
    - /api/v1/assessments
    - /api/v1/notifications/inbox
    - /api/v1/incidents
    - /api/v1/family/**
```

When `DEMO_MODE=true`:

- BFF always serves mock fallback for protected routes, regardless of whether the real service is available
- Real services continue running and accepting non-demo traffic
- A `X-Demo-Mode: true` response header is added to indicate source

When `DEMO_MODE=false` (default):

- Normal routing applies; real services used where wired; fallback only on error

Stability Window Protocol

For each demo-critical domain, the fallback is not disabled until **48 consecutive hours of stability** in the target environment:

```
48h stability check:
[ ] Zero 5xx errors from real service in Azure Monitor
[ ] P95 latency < 500ms for the domain's endpoints
[ ] Zero fallback activations recorded in BFF logs
[ ] At least 1 full demo walkthrough completed successfully against real backend

Only after all 4 checks pass → disable mock fallback for this domain
```

Pre-Demo Checklist

Before any stakeholder demo during Phase 3:

```
# 1. Enable DEMO_MODE in dev/staging BFF
az containerapp update \
  --name ca-vcc-dev-web-bff-001 \
  --resource-group rg-vcc-dev-001 \
  --set-env-vars DEMO_MODE=true

# 2. Verify mock fallback is serving demo-critical routes
curl -s -H "Authorization: Bearer $DEMO_JWT" \
  https://<bff-fqdn>/api/v1/dashboard/stats \
  | jq '.totalClients'
# Expected: non-null value from mock data

# 3. After demo — reset to real-service mode
az containerapp update \
  --name ca-vcc-dev-web-bff-001 \
  --resource-group rg-vcc-dev-001 \
  --set-env-vars DEMO_MODE=false
```

Mock Data Preservation

The mock API's seed data (`server.js` in-memory arrays) must not be modified or reset during active demo periods. Tag a stable demo snapshot:

```
# Before Phase 3 begins, create a demo-data tag in git
cd /Users/makinja/projects/client/lumiscare/mock-api
git tag demo-data-snapshot-$(date +%Y%m%d)
```

If mock data gets corrupted, restore from this tag and restart the mock API container.

Revision #2
Created 2026-04-05 21:40:51 UTC by John
Updated 2026-05-31 20:05:37 UTC by John