Mediconnect PRD — V1.1 ADDENDUM

The following sections append to the previously delivered V1 PRD. No prior content is repeated.

---

1) Data Model Pack (ERD + SQL + RLS)

1.1 ERD (entities + relationships)

- user (id) —< consult (patient_id, gp_id, specialist_id?)
- consult (id) —< message (consult_id)
- consult (id) —< referral (consult_id, specialist_id, to_org_id?)
- consult (id) —< appointment (consult_id)
- consult (id) —< prescription (consult_id, patient_id, prescriber_id)
- prescription (id) —< pharmacy_claim (prescription_id, org_id)
- consult (id) —< lab_order (consult_id, patient_id, org_id)
- organization (id, type: pharmacy|lab|clinic) —< pharmacy_claim (org_id), lab_order (org_id), user (org_id nullable)
- audit_event (id) references actor_user_id, actor_org_id (nullable), subject_table, subject_id

Cardinalities
- user:consult = 1:N (patient has many consults; GP/Specialist linked per consult)
- consult:message = 1:N
- consult:referral = 1:N
- consult:appointment = 1:N
- consult:prescription = 1:N
- prescription:pharmacy_claim = 1:N
- consult:lab_order = 1:N
- organization:pharmacy_claim = 1:N
- organization:lab_order = 1:N

1.2 SQL DDL (PostgreSQL stubs)

```
-- Enable required extensions
CREATE EXTENSION IF NOT EXISTS pgcrypto; -- for gen_random_uuid()

-- Enums
DO $$ BEGIN
  CREATE TYPE user_role AS ENUM ('patient','gp','specialist','pharmacy_admin','dia-
gnostics_admin','support','ops');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE org_type AS ENUM ('clinic','pharmacy','lab');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE consult_status AS ENUM ('draft','active','completed','cancelled');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE message_type AS ENUM ('text','image','file','system');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE referral_status AS ENUM ('proposed','sent','accepted','declined','com-
pleted');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE prescription_status AS ENUM ('draft','issued','revoked');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE claim_status AS ENUM ('pending','approved','rejected','reversed');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;
DO $$ BEGIN
  CREATE TYPE lab_order_status AS ENUM ('created','sent','sample_collected','in_progre
ss','completed','cancelled');
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

-- Organizations
CREATE TABLE IF NOT EXISTS organization (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  type            org_type NOT NULL,
  name            TEXT NOT NULL,
  country_code    TEXT NOT NULL,
  address         TEXT,
  phone           TEXT,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_org_type ON organization(type);

-- Users
CREATE TABLE IF NOT EXISTS "user" (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  role            user_role NOT NULL,
  org_id          UUID REFERENCES organization(id) ON DELETE SET NULL,
  phone_e164      TEXT NOT NULL UNIQUE,
  full_name       TEXT,
  email           TEXT,
  dob             DATE,
  gender          TEXT,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  deleted_at      TIMESTAMPTZ
);
CREATE INDEX IF NOT EXISTS idx_user_role ON "user"(role);
CREATE INDEX IF NOT EXISTS idx_user_org ON "user"(org_id);
```

```
-- Consults
CREATE TABLE IF NOT EXISTS consult (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  status          consult_status NOT NULL DEFAULT 'active',
  patient_id      UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  gp_id           UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  specialist_id   UUID REFERENCES "user"(id) ON DELETE SET NULL,
  chief_complaint TEXT,
  started_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  ended_at        TIMESTAMPTZ,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_consult_patient ON consult(patient_id);
CREATE INDEX IF NOT EXISTS idx_consult_gp ON consult(gp_id);
CREATE INDEX IF NOT EXISTS idx_consult_specialist ON consult(specialist_id);

-- Messages
CREATE TABLE IF NOT EXISTS message (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  consult_id      UUID NOT NULL REFERENCES consult(id) ON DELETE CASCADE,
  sender_id       UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  type            message_type NOT NULL DEFAULT 'text',
  body_text       TEXT,
  media_url       TEXT,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_message_consult ON message(consult_id);
CREATE INDEX IF NOT EXISTS idx_message_sender ON message(sender_id);

-- Referrals
CREATE TABLE IF NOT EXISTS referral (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  consult_id      UUID NOT NULL REFERENCES consult(id) ON DELETE CASCADE,
  from_gp_id      UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  to_specialist_id UUID REFERENCES "user"(id) ON DELETE SET NULL,
  to_org_id       UUID REFERENCES organization(id) ON DELETE SET NULL,
  status          referral_status NOT NULL DEFAULT 'sent',
  reason          TEXT,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_referral_consult ON referral(consult_id);
CREATE INDEX IF NOT EXISTS idx_referral_to_org ON referral(to_org_id);

-- Appointments
CREATE TABLE IF NOT EXISTS appointment (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  consult_id      UUID NOT NULL REFERENCES consult(id) ON DELETE CASCADE,
  scheduled_start TIMESTAMPTZ NOT NULL,
  scheduled_end   TIMESTAMPTZ,
  location        TEXT,
  created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at      TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_appt_consult ON appointment(consult_id);
CREATE INDEX IF NOT EXISTS idx_appt_start ON appointment(scheduled_start);

-- Prescriptions
CREATE TABLE IF NOT EXISTS prescription (
  id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  consult_id        UUID NOT NULL REFERENCES consult(id) ON DELETE CASCADE,
  patient_id        UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
```

```
  prescriber_id     UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  status            prescription_status NOT NULL DEFAULT 'issued',
  items_json        JSONB NOT NULL, -- list of items {drug, dose, qty, instructions}
  qr_code           TEXT,           -- base64 or url
  qr_enabled        BOOLEAN NOT NULL DEFAULT TRUE,
  pdf_downloaded_at TIMESTAMPTZ,
  notes             TEXT,
  created_at        TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at        TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_rx_consult ON prescription(consult_id);
CREATE INDEX IF NOT EXISTS idx_rx_patient ON prescription(patient_id);
CREATE INDEX IF NOT EXISTS idx_rx_qr_enabled ON prescription(qr_enabled);

-- Pharmacy Claims
CREATE TABLE IF NOT EXISTS pharmacy_claim (
  id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  prescription_id   UUID NOT NULL REFERENCES prescription(id) ON DELETE CASCADE,
  org_id            UUID NOT NULL REFERENCES organization(id) ON DELETE RESTRICT,
  claimant_user_id  UUID REFERENCES "user"(id) ON DELETE SET NULL,
  status            claim_status NOT NULL DEFAULT 'pending',
  items_filled_json JSONB NOT NULL, -- subset of items_json with fill info
  qr_verified       BOOLEAN NOT NULL DEFAULT FALSE,
  verified_at       TIMESTAMPTZ,
  created_at        TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at        TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE (prescription_id, org_id)
);
CREATE INDEX IF NOT EXISTS idx_claim_rx ON pharmacy_claim(prescription_id);
CREATE INDEX IF NOT EXISTS idx_claim_org ON pharmacy_claim(org_id);

-- Lab Orders
CREATE TABLE IF NOT EXISTS lab_order (
  id                UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  consult_id        UUID NOT NULL REFERENCES consult(id) ON DELETE CASCADE,
  patient_id        UUID NOT NULL REFERENCES "user"(id) ON DELETE RESTRICT,
  org_id            UUID NOT NULL REFERENCES organization(id) ON DELETE RESTRICT,
  tests_json        JSONB NOT NULL, -- list of tests
  status            lab_order_status NOT NULL DEFAULT 'created',
  results_url       TEXT,
  created_at        TIMESTAMPTZ NOT NULL DEFAULT now(),
  updated_at        TIMESTAMPTZ NOT NULL DEFAULT now()
);
CREATE INDEX IF NOT EXISTS idx_lab_order_consult ON lab_order(consult_id);
CREATE INDEX IF NOT EXISTS idx_lab_order_org ON lab_order(org_id);

-- Audit Events
CREATE TABLE IF NOT EXISTS audit_event (
  id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  occurred_at     TIMESTAMPTZ NOT NULL DEFAULT now(),
  actor_user_id   UUID REFERENCES "user"(id) ON DELETE SET NULL,
  actor_org_id    UUID REFERENCES organization(id) ON DELETE SET NULL,
  actor_role      user_role,
  action          TEXT NOT NULL, -- canonical event name
  subject_table   TEXT NOT NULL,
  subject_id      UUID,
  reason          TEXT,           -- free-form justification or policy reference
  request_id      TEXT,           -- trace
  ip              INET,
  user_agent      TEXT,
  context         JSONB           -- any extra structured details
);
CREATE INDEX IF NOT EXISTS idx_audit_when ON audit_event(occurred_at);
```

```
CREATE INDEX IF NOT EXISTS idx_audit_actor ON audit_event(actor_user_id);
CREATE INDEX IF NOT EXISTS idx_audit_action ON audit_event(action);

-- RLS infra: we assume these session variables are set by the app per request
--   SELECT set_config('app.user_id', '<uuid>', false);
--   SELECT set_config('app.role', '<role>', false);
--   SELECT set_config('app.org_id', '<uuid | null>', false);
--   SELECT set_config('app.support_session_expires', '<timestamptz | null>', false);

-- Enable RLS
ALTER TABLE "user" ENABLE ROW LEVEL SECURITY;
ALTER TABLE consult ENABLE ROW LEVEL SECURITY;
ALTER TABLE message ENABLE ROW LEVEL SECURITY;
ALTER TABLE referral ENABLE ROW LEVEL SECURITY;
ALTER TABLE appointment ENABLE ROW LEVEL SECURITY;
ALTER TABLE prescription ENABLE ROW LEVEL SECURITY;
ALTER TABLE pharmacy_claim ENABLE ROW LEVEL SECURITY;
ALTER TABLE lab_order ENABLE ROW LEVEL SECURITY;

-- Patients: can read/write their own consults/messages; read their prescriptions/lab
orders
CREATE POLICY patient_consults ON consult
  USING (patient_id::text = current_setting('app.user_id', true))
  WITH CHECK (patient_id::text = current_setting('app.user_id', true));

CREATE POLICY patient_messages ON message
  USING (consult_id IN (SELECT id FROM consult WHERE patient_id::text = cur-
rent_setting('app.user_id', true)));

CREATE POLICY patient_rx ON prescription
  USING (patient_id::text = current_setting('app.user_id', true));

CREATE POLICY patient_labs ON lab_order
  USING (patient_id::text = current_setting('app.user_id', true));

-- GP: can access consults for which they are gp_id; see their patients' data
CREATE POLICY gp_consults ON consult
  USING (gp_id::text = current_setting('app.user_id', true));

CREATE POLICY gp_messages ON message
  USING (consult_id IN (SELECT id FROM consult WHERE gp_id::text = current_setting('ap
p.user_id', true)));

CREATE POLICY gp_referrals ON referral
  USING (from_gp_id::text = current_setting('app.user_id', true));

CREATE POLICY gp_rx ON prescription
  USING (prescriber_id::text = current_setting('app.user_id', true)
      OR consult_id IN (SELECT id FROM consult WHERE gp_id::text = current_setting('ap
p.user_id', true)));

CREATE POLICY gp_labs ON lab_order
  USING (consult_id IN (SELECT id FROM consult WHERE gp_id::text = current_setting('ap
p.user_id', true)));

-- Specialist: read-only access to consults referred to them or their org; can write
messages
CREATE POLICY specialist_consults ON consult
  USING (id IN (
    SELECT r.consult_id FROM referral r
    WHERE (r.to_specialist_id::text = current_setting('app.user_id', true)
       OR r.to_org_id::text = current_setting('app.org_id', true))
      AND r.status IN ('sent','accepted','completed')
```

```
  ));

CREATE POLICY specialist_messages ON message
  USING (consult_id IN (
    SELECT r.consult_id FROM referral r
    WHERE (r.to_specialist_id::text = current_setting('app.user_id', true)
       OR r.to_org_id::text = current_setting('app.org_id', true))
  ))
  WITH CHECK (consult_id IN (
    SELECT r.consult_id FROM referral r
    WHERE (r.to_specialist_id::text = current_setting('app.user_id', true)
       OR r.to_org_id::text = current_setting('app.org_id', true))
  ));

-- Pharmacy Admin: access prescriptions only via claims for own org; minimal fields
CREATE POLICY pharmacy_claims_org ON pharmacy_claim
  USING (org_id::text = current_setting('app.org_id', true))
  WITH CHECK (org_id::text = current_setting('app.org_id', true));

-- To allow pharm. to verify QR against prescription without PII, gate by qr_enabled
and claim org
CREATE POLICY pharmacy_verify_rx ON prescription
  USING (
    qr_enabled = TRUE
    AND EXISTS (
      SELECT 1 FROM pharmacy_claim c
      WHERE c.prescription_id = prescription.id
        AND c.org_id::text = current_setting('app.org_id', true)
    )
  );

-- Diagnostics Admin: access lab_orders for own org; minimal PII in app responses
CREATE POLICY lab_orders_org ON lab_order
  USING (org_id::text = current_setting('app.org_id', true))
  WITH CHECK (org_id::text = current_setting('app.org_id', true));

-- Support: time-boxed masked read access via support sessions
-- Create a support session table to anchor time-box
CREATE TABLE IF NOT EXISTS support_session (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  support_user_id UUID NOT NULL REFERENCES "user"(id),
  target_user_id  UUID NOT NULL REFERENCES "user"(id),
  expires_at      TIMESTAMPTZ NOT NULL,
  reason          TEXT
);
ALTER TABLE support_session ENABLE ROW LEVEL SECURITY;

CREATE POLICY support_self ON support_session
  USING (support_user_id::text = current_setting('app.user_id', true));

-- Masking is best done via SECURITY BARRIER views; example masked user view
CREATE OR REPLACE VIEW support_user_masked WITH (security_barrier=true) AS
  SELECT id,
         role,
         org_id,
         left(full_name, 1) || '***' AS full_name_masked,
         '+***' || right(phone_e164, 4) AS phone_masked,
         NULL::TEXT AS email_masked,
         dob,
         gender,
         created_at
  FROM "user";
```

```
-- Policy: allow support to select from masked view if within time-box for target user
-- Application should set app.support_session_expires and app.role='support'

-- Audit: ensure write-only
REVOKE ALL ON audit_event FROM PUBLIC;
GRANT INSERT ON audit_event TO PUBLIC; -- app role only in prod
```

1.3 RLS Access Matrix (illustrative)

- Patient: own consult/messages/prescriptions/lab_orders (read; writes limited to messages/intake where applicable)
- GP: consults where gp_id, related messages/referrals/prescriptions/labs
- Specialist: consults via accepted referral (read), can add messages
- Pharmacy Admin: pharmacy_claim rows for their org; prescription readable when linked to claim and qr_enabled
- Diagnostics Admin: lab_order rows for their org; minimal PII returned by API
- Support: masked views only, restricted by support_session.expires_at and justification required in audit_event

1.4 Audit Dictionary

Event (action)	Who (actor_role)	What (subject_table)	When	Why (reason)	Required fields
auth.otp.sent	ops/system	n/a	send time	template/version	request_id, phone_e164
auth.otp.verified	patient	user	verification time	session start	actor_user_id, request_id
consult.created	gp	consult	creation time	patient_care	consult_id, patient_id
message.sent	any clinician/patient	message	send time	care_comm	consult_id, sender_id, type
referral.sent	gp	referral	send time	specialist_needed	referral_id, to_org_id/to_specialist_id
prescription.issued	gp/specialist	prescription	issue time	treatment	prescription_id, patient_id
prescription.pdf.downloaded	gp/specialist	prescription	download time	patient_copy	prescription_id
prescription.qr.disabled	system	prescription	disable time	pdf_downloaded	prescription_id
pharmacy.qr.verified	pharmacy_admin	pharmacy_claim	verify time	dispense	claim_id, org_id
pharmacy.claim.submitted	pharmacy_admin	pharmacy_claim	submit time	reimbursement	claim_id
lab.order.created	gp/specialist	lab_order	creation time	diagnostics	lab_order_id
lab.result.uploaded	diagnostics_admin	lab_order	upload time	results_ready	lab_order_id

Event (action)	Who (actor_role)	What (subject_table)	When	Why (reason)	Required fields
support.session.started	support	support_session	start time	troubleshooting	support_session_id, target_user_id
support.session.ended	support/system	support_session	end time	expiry	support_session_id

Retention
- audit_event: 2 years online warm, optional archive for 7 years (compliance-dependent). PII minimised in context.

---

2) API Stubs (OpenAPI v3)

```yaml
openapi: 3.0.3
info:
  title: Mediconnect API
  version: 1.1.0
servers:
  - url: https://api.mediconnect.example/v1
    description: Production
  - url: https://staging.api.mediconnect.example/v1
    description: Staging
components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
  schemas:
    Problem:
      type: object
      properties:
        type: { type: string, format: uri }
        title: { type: string }
        status: { type: integer }
        detail: { type: string }
        instance: { type: string }
        requestId: { type: string }
      required: [title, status]
    Meta:
      type: object
      properties:
        page: { type: integer }
        page_size: { type: integer }
        total: { type: integer }
    User:
      type: object
      properties:
        id: { type: string, format: uuid }
        role: { type: string, enum: [patient,gp,specialist,pharmacy_admin,dia-
gnostics_admin,support,ops] }
        full_name: { type: string }
        phone_e164: { type: string }
        org_id: { type: string, format: uuid, nullable: true }
    Consult:
      type: object
      properties:
        id: { type: string, format: uuid }
        status: { type: string, enum: [draft,active,completed,cancelled] }
        patient_id: { type: string, format: uuid }
        gp_id: { type: string, format: uuid }
        specialist_id: { type: string, format: uuid, nullable: true }
        chief_complaint: { type: string, nullable: true }
        started_at: { type: string, format: date-time }
        ended_at: { type: string, format: date-time, nullable: true }
      required: [id,status,patient_id,gp_id,started_at]
    Intake:
      type: object
      properties:
        id: { type: string, format: uuid }
        consult_id: { type: string, format: uuid }
        responses: { type: object, additionalProperties: true }
      required: [id,consult_id,responses]
    Referral:
      type: object
```

```
      properties:
        id: { type: string, format: uuid }
        consult_id: { type: string, format: uuid }
        from_gp_id: { type: string, format: uuid }
        to_specialist_id: { type: string, format: uuid, nullable: true }
        to_org_id: { type: string, format: uuid, nullable: true }
        status: { type: string, enum: [proposed,sent,accepted,declined,completed] }
        reason: { type: string, nullable: true }
      required: [id,consult_id,from_gp_id,status]
    Prescription:
      type: object
      properties:
        id: { type: string, format: uuid }
        consult_id: { type: string, format: uuid }
        patient_id: { type: string, format: uuid }
        prescriber_id: { type: string, format: uuid }
        status: { type: string, enum: [draft,issued,revoked] }
        items: { type: array, items: { type: object } }
        qr_enabled: { type: boolean }
        pdf_downloaded_at: { type: string, format: date-time, nullable: true }
      required: [id,consult_id,patient_id,prescriber_id,status,items,qr_enabled]
    PharmacyClaim:
      type: object
      properties:
        id: { type: string, format: uuid }
        prescription_id: { type: string, format: uuid }
        org_id: { type: string, format: uuid }
        status: { type: string, enum: [pending,approved,rejected,reversed] }
        items_filled: { type: array, items: { type: object } }
        qr_verified: { type: boolean }
        verified_at: { type: string, format: date-time, nullable: true }
      required: [id,prescription_id,org_id,status,items_filled,qr_verified]
    LabOrder:
      type: object
      properties:
        id: { type: string, format: uuid }
        consult_id: { type: string, format: uuid }
        patient_id: { type: string, format: uuid }
        org_id: { type: string, format: uuid }
        status: { type: string, enum: [created,sent,sample_collected,in_progress,com-
pleted,cancelled] }
        tests: { type: array, items: { type: object } }
        results_url: { type: string, nullable: true }
      required: [id,consult_id,patient_id,org_id,status,tests]
    NotificationTest:
      type: object
      properties:
        id: { type: string, format: uuid }
        channel: { type: string, enum: [whatsapp,in_app] }
        status: { type: string, enum: [queued,delivered,failed] }
  parameters:
    IdempotencyKey:
      in: header
      name: x-idempotency-key
      required: false
      schema: { type: string }
      description: Provide to make POST idempotent for 24h.
  responses:
    RateLimited:
      description: Too many requests
      headers:
        Retry-After:
          schema: { type: integer }
```

```
      content:
        application/problem+json:
          schema: { $ref: '#/components/schemas/Problem' }
paths:
  /auth/whatsapp/otp:
    post:
      summary: Send OTP to WhatsApp
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                phone_e164: { type: string }
              required: [phone_e164]
      responses:
        '202': { description: OTP queued }
        '400':
          description: Bad request
          content: { application/problem+json: { schema: { $ref: '#/components/schem-
as/Problem' } } }
        '429': { $ref: '#/components/responses/RateLimited' }
        '500':
          description: Server error
          content: { application/problem+json: { schema: { $ref: '#/components/schem-
as/Problem' } } }
  /auth/whatsapp/verify:
    post:
      summary: Verify OTP
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                phone_e164: { type: string }
                code: { type: string }
              required: [phone_e164, code]
      responses:
        '200':
          description: Verified
          content:
            application/json:
              schema:
                type: object
                properties:
                  token: { type: string }
                  user: { $ref: '#/components/schemas/User' }
        '400': { description: Invalid code, content: { application/problem+json: {
schema: { $ref: '#/components/schemas/Problem' } } } }
        '401': { description: Unauthorised, content: { application/problem+json: {
schema: { $ref: '#/components/schemas/Problem' } } } }
        '429': { $ref: '#/components/responses/RateLimited' }
  /consults:
    post:
      summary: Create consult
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
```

```
          application/json:
            schema:
              type: object
              properties:
                patient_id: { type: string, format: uuid }
                chief_complaint: { type: string }
              required: [patient_id]
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/Consult' } } } }
        '400': { description: Bad request, content: { application/problem+json: {
schema: { $ref: '#/components/schemas/Problem' } } } }
        '401': { description: Unauthorised }
        '409': { description: Conflict (idempotency) }
        '429': { $ref: '#/components/responses/RateLimited' }
  /consults/{id}:
    get:
      summary: Get consult by id
      security: [{ bearerAuth: [] }]
      parameters:
        - in: path
          name: id
          required: true
          schema: { type: string, format: uuid }
      responses:
        '200': { description: OK, content: { application/json: { schema: { $ref: '#/
components/schemas/Consult' } } } }
        '404': { description: Not found }
    patch:
      summary: Update consult
      security: [{ bearerAuth: [] }]
      parameters: [ { in: path, name: id, required: true, schema: { type: string,
format: uuid } }, { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                status: { type: string, enum: [draft,active,completed,cancelled] }
                specialist_id: { type: string, format: uuid }
                ended_at: { type: string, format: date-time }
      responses:
        '200': { description: Updated, content: { application/json: { schema: { $ref:
'#/components/schemas/Consult' } } } }
        '400': { description: Bad request }
        '404': { description: Not found }
  /intake:
    post:
      summary: Submit intake
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                consult_id: { type: string, format: uuid }
                responses: { type: object, additionalProperties: true }
              required: [consult_id, responses]
```

```
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/Intake' } } } }
        '400': { description: Bad request }
  /intake/{id}:
    get:
      summary: Get intake
      security: [{ bearerAuth: [] }]
      parameters: [ { in: path, name: id, required: true, schema: { type: string,
format: uuid } } ]
      responses:
        '200': { description: OK, content: { application/json: { schema: { $ref: '#/
components/schemas/Intake' } } } }
        '404': { description: Not found }
  /referrals:
    post:
      summary: Create referral
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                consult_id: { type: string, format: uuid }
                to_specialist_id: { type: string, format: uuid }
                to_org_id: { type: string, format: uuid }
                reason: { type: string }
              anyOf:
                - required: [to_specialist_id]
                - required: [to_org_id]
              required: [consult_id]
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/Referral' } } } }
        '400': { description: Bad request }
  /referrals/{id}:
    get:
      summary: Get referral
      security: [{ bearerAuth: [] }]
      parameters: [ { in: path, name: id, required: true, schema: { type: string,
format: uuid } } ]
      responses:
        '200': { description: OK, content: { application/json: { schema: { $ref: '#/
components/schemas/Referral' } } } }
        '404': { description: Not found }
  /prescriptions:
    post:
      summary: Issue prescription
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                consult_id: { type: string, format: uuid }
                patient_id: { type: string, format: uuid }
                items: { type: array, items: { type: object } }
```

```
              required: [consult_id, patient_id, items]
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/Prescription' } } } }
        '400': { description: Bad request }
  /prescriptions/{id}:
    get:
      summary: Get prescription
      security: [{ bearerAuth: [] }]
      parameters: [ { in: path, name: id, required: true, schema: { type: string,
format: uuid } } ]
      responses:
        '200': { description: OK, content: { application/json: { schema: { $ref: '#/
components/schemas/Prescription' } } } }
        '404': { description: Not found }
  /pharmacy/claims/verify-qr:
    post:
      summary: Verify QR for a prescription (pharmacy)
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                qr_payload: { type: string }
                org_id: { type: string, format: uuid }
              required: [qr_payload, org_id]
      responses:
        '200':
          description: Verified
          headers:
            X-RateLimit-Limit: { schema: { type: integer } }
            X-RateLimit-Remaining: { schema: { type: integer } }
          content:
            application/json:
              schema:
                type: object
                properties:
                  prescription_id: { type: string, format: uuid }
                  qr_valid: { type: boolean }
                  minimal_info: { type: object }
        '400': { description: Bad request }
        '401': { description: Unauthorised }
        '404': { description: Not found }
        '409': { description: QR disabled }
        '429': { $ref: '#/components/responses/RateLimited' }
  /pharmacy/claims:
    post:
      summary: Submit pharmacy claim
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                prescription_id: { type: string, format: uuid }
                org_id: { type: string, format: uuid }
```

```
              items_filled: { type: array, items: { type: object } }
            required: [prescription_id, org_id, items_filled]
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/PharmacyClaim' } } } }
        '400': { description: Bad request }
        '401': { description: Unauthorised }
        '409': { description: Duplicate claim }
  /labs/orders:
    post:
      summary: Create lab order
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                consult_id: { type: string, format: uuid }
                patient_id: { type: string, format: uuid }
                org_id: { type: string, format: uuid }
                tests: { type: array, items: { type: object } }
              required: [consult_id, patient_id, org_id, tests]
      responses:
        '201': { description: Created, content: { application/json: { schema: { $ref:
'#/components/schemas/LabOrder' } } } }
        '400': { description: Bad request }
  /labs/orders/{id}:
    get:
      summary: Get lab order
      security: [{ bearerAuth: [] }]
      parameters: [ { in: path, name: id, required: true, schema: { type: string,
format: uuid } } ]
      responses:
        '200': { description: OK, content: { application/json: { schema: { $ref: '#/
components/schemas/LabOrder' } } } }
        '404': { description: Not found }
  /labs/results:
    post:
      summary: Upload lab results (diagnostics)
      security: [{ bearerAuth: [] }]
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                lab_order_id: { type: string, format: uuid }
                results_url: { type: string }
              required: [lab_order_id, results_url]
      responses:
        '200': { description: Updated, content: { application/json: { schema: { $ref:
'#/components/schemas/LabOrder' } } } }
        '400': { description: Bad request }
        '404': { description: Not found }
  /notifications/test:
    post:
      summary: Send test notification (ops only)
      security: [{ bearerAuth: [] }]
```

```
      parameters: [ { $ref: '#/components/parameters/IdempotencyKey' } ]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                channel: { type: string, enum: [whatsapp,in_app] }
                to: { type: string }
                template: { type: string }
                variables: { type: object, additionalProperties: true }
              required: [channel, to]
      responses:
        '202': { description: Queued, content: { application/json: { schema: { $ref: '
#/components/schemas/NotificationTest' } } } }
        '400': { description: Bad request }
        '403': { description: Forbidden }

# Global error shapes for 4xx/5xx are application/problem+json.
# Rate limits: default 60 rpm per IP; 429 returns Retry-After. Other responses include
X-RateLimit headers.
```

---

3) Notification & Template Matrix

Channels: in-app, WhatsApp (templated).

Event	Chan-nel(s)	Template name	Sample body (place-holders)	Variables	Audience	Throttle
OTP sent	WhatsApp	auth_otp_v1	“Your Medicon-nect code is {{code}}. Expires in {{ttl_minutes}} min.”	code, ttl_minutes	Patient	3 sends / hour per number
Consult created	In-app, WhatsApp	con-sult_created_v1	“Your con-sultation with {{gp_name}} has started.”	gp_name	Patient	Once per consult
Referral sent	WhatsApp	refer-ral_sent_v1	“You were referred to {{spe-cialty}} at {{org_name}}. We’ll notify when ac-cepted.”	specialty, org_name	Patient	Once per referral
Prescrip-tion issued	WhatsApp	rx_issued_v1	“Prescrip-tion ready. QR is valid until PDF is down-loaded. ID: {{rx_short}}.”	rx_short	Patient	Once per prescrip-tion
RX PDF down-loaded	In-app	rx_pdf_downloaded_v1	“You down-loaded your pre-scription PDF. QR is	rx_id	Patient	Once per prescrip-tion

Event	Chan-nel(s)	Template name	Sample body (place-holders)	Variables	Audience	Throttle
			now dis-abled.”			
Pharmacy QR verified	In-app	phar-macy_qr_v erified_v1	“Pharmacy {{org_nam e}} veri-fied your prescrip-tion.”	org_name	Patient	Once per claim
Pharmacy claim sub-mitted	In-app	claim_sub mitted_v1	“Claim from {{org_nam e}} sub-mitted.”	org_name	GP	Once per claim
Lab order created	WhatsApp	lab_order_ created_v1	“Lab order to {{org_nam e}} cre-ated. Tests: {{tests_sh ort}}.”	org_name, tests_short	Patient	Once per order
Lab results ready	WhatsApp	lab_results _ready_v1	“Your lab results are ready. View securely: {{link}}.”	link	Patient	Retry up to 3x on fail-ure
Ops test	WhatsApp	ops_test_v 1	“Test: {{mes-sage}}”	message	Ops	Unlimited (ops only)

Retries
- WhatsApp: exponential backoff 1m, 5m, 15m; stop after 3 attempts or if read receipt delivered.
- In-app: guaranteed delivery; shown on next session.

---

4) Analytics & Observability Plan

Events (name → when → properties)
- auth_otp_sent → after /auth/whatsapp/otp accepted → phone_e164_hash, template, ttl

- auth_otp_verified → on successful verify → user_id, attempt_count, latency_ms
- consult_created → after creation → consult_id, patient_id_hash, gp_id, source
- message_sent → on send → consult_id, sender_role, type, size_bytes
- referral_sent → on create → referral_id, to_org_type, to_org_id
- prescription_issued → on create → prescription_id, items_count, has_qr
- pdf_downloaded → when PDF generated & downloaded → prescription_id, user_role
- qr_disabled → when qr_enabled toggled false → prescription_id, reason
- qr_verified → on pharmacy verify → claim_id, org_id, success
- claim_submitted → on create → claim_id, org_id, items_count
- lab_order_created → on create → lab_order_id, org_id, tests_count
- lab_result_uploaded → on upload → lab_order_id, org_id, file_size
- notification_queued → on enqueue → channel, template, audience_role
- notification_delivered → on receipt → channel, delivery_ms
- notification_failed → on error → channel, error_code

SLIs/SLOs/KPIs mapping
- OTP conversion rate (KPI) = auth_otp_verified/auth_otp_sent (SLO ≥ 95%)
- Consult start latency (SLI) = time from POST /consults to first message_sent (SLO p95 < 2 min)
- Referral acceptance rate (KPI) = accepted/(sent) per org
- RX fulfilment time (KPI) = time from prescription_issued to claim_submitted (p50 < 24h)
- Lab TAT (KPI) = lab_order_created→lab_result_uploaded (p90 < 36h)
- WhatsApp delivery success (SLI) = notification_delivered/notification_queued (SLO ≥ 98%)
- Error budget (Ops) from 5xx rate per endpoint (SLO 99.9% success over 30d)

Dashboards & alerts
- Auth funnel: sent→delivered→verified; alert if conversion < 90% 1h window
- Consult flow: creations/hour, time to first message; alert p95 > 5 min
- Referral: sent vs accepted by org; alert if any partner < 50% weekly
- Prescriptions: issued/day, pdf_downloaded ratio, qr_disabled counts
- Pharmacy: QR verify success rate; alert if < 95% daily
- Labs: orders vs results; alert if p90 TAT > 48h
- Notifications: WA provider errors by code; alert spike > 3σ
- Infra: latency/5xx/429 per endpoint with SLO burn alerts

---

5) ADRs (short form)

1. WhatsApp-only Auth
   - Context: Patient-first onboarding with minimal friction.
   - Options: WhatsApp OTP; SMS OTP; Email link; Passwords.
   - Decision: WhatsApp OTP as primary; SMS fallback disabled in V1.
   - Consequences: High conversion in WA-heavy markets; dependency on WA availability.
   - Status: Accepted.

2. GP-led Referral
   - Context: Clinical governance and continuity.
   - Options: Patient self-referral; GP-only; Mixed.
   - Decision: GP initiates; specialist/organisation accepts.
   - Consequences: Better triage, slower patient-initiated flows.
   - Status: Accepted.

3. QR Disable After PDF Download
   - Context: Mitigate duplicate dispensing.
   - Options: Keep QR active; Disable on PDF download; Time-bound QR.
   - Decision: Disable immediately when PDF downloaded; visible warning.
   - Consequences: Pharmacy must rely on claim linkage; reduces fraud.
   - Status: Accepted.

4. Partner PII Minimisation (Pharmacy)
   - Context: Non-care entities should not see excess PII.
   - Options: Full RX view; Minimal view; Tokenised view.
   - Decision: Minimal view via claim + org scope; no diagnosis or full name.
   - Consequences: Lower support load, stronger privacy.
   - Status: Accepted.

5. Diagnostics Minimal PII
   - Context: Labs need only test set and routing.
   - Decision: Provide initials/ID only; results delivered via secure link.
   - Consequences: Harder manual reconciliation without ID card; better privacy.
   - Status: Accepted.

6. Consult Timing Windows
   - Context: SLA and scheduling.
   - Options: Hard time-box; Soft guidance.
   - Decision: Soft target 20–30 min per consult; auto-complete after 24h inactivity.
   - Consequences: Predictable metrics; possible premature closure edge cases.
   - Status: Accepted.

7. Data Retention Policy
   - Context: Compliance vs. storage.
   - Options: Indefinite; Tiered; Minimal.
   - Decision: Tiered—clinical records 7y, audit 2y warm + archive, notifications 90d.
   - Consequences: Archival service needed.
   - Status: Accepted.

8. Idempotency Keys on Mutating POSTs
   - Context: Mobile retries and WA backoffs.
   - Decision: Require x-idempotency-key on client SDK for POSTs.
   - Consequences: Lower duplicates; store keys 24h.
   - Status: Accepted.

---

6) Open Issues & Decisions Needed

1. Patient Identity Proofing Level
   - Blocker: Strength of KYC needed for RX and lab.
   - Default: Phone+DOB check for V1; upgrade path to NIN/ID later.

2. Specialist Onboarding
   - Blocker: Direct vs via org admin approval.
   - Default: Via org admin with document upload.

3. Pharmacy Claim Settlement Model
   - Blocker: Reimbursement vs simple fulfilment record.
   - Default: V1 only records fulfilment; no financials.

4. Lab Results File Storage
   - Blocker: Vendor S3 vs on-prem.
   - Default: Encrypted S3 with 30-day signed URLs.

5. WhatsApp Provider Choice
   - Blocker: Meta direct vs aggregator.
   - Default: Aggregator with SLA and webhooks.

6. PDF Generation Service
   - Blocker: Server-side vs client.
   - Default: Server-side HTML→PDF using Chromium.

7. Timezone Handling
   - Blocker: Patient vs clinician locale.
   - Default: Store UTC; render per viewer tz.

8. PII Hashing Standard
   - Blocker: Consistent hashing for analytics.
   - Default: SHA-256 with per-environment salt.

9. Support Session Maximum Duration
   - Blocker: Risk vs usability.
   - Default: 30 minutes; single target user; reason mandatory.

10. Rate Limits Per Role
    - Blocker: Different ceilings for pharmacy/lab.
    - Default: 60 rpm default; pharmacy verify 120 rpm; ops test 10 rpm.

---

7) Changelog

- Normalised entity names and enums across design spec and starter docs.
- Added prescription.qr_enabled and prescription.pdf_downloaded_at.
- Completed API stubs with idempotency and problem+json.
- Added RLS policies for all roles, including support time-boxing.
- Defined notification templates and analytics events.

Template → Our PRD mapping

External template heading	Our final section
User Flows	Core Journeys & API (prior PRD)
Data Schema	1) Data Model Pack
API Endpoints	2) API Stubs
Notifications	3) Notification & Template Matrix
Metrics	4) Analytics & Observability
Decisions	5) ADRs
Risks/Issues	6) Open Issues
Release Notes	7) Changelog
Security	8) Security & Privacy Addenda
Localisation	9) i18n Pack
QA	10) Accessibility & Low-Bandwidth QA

---

8) Security & Privacy Addenda

RLS Examples (read vs unmask)

```
-- Read: diagnostics sees lab_order with minimal PII via API serializer; DB side
restricts rows by org.
-- Unmask: Only GP or patient can see full name/phone; expose via JOINs only when
viewer is authorised.

-- Example: read policy already defined (lab_orders_org).
-- Example: unmask policy for patient to see their own PII in user table
CREATE POLICY patient_user_self ON "user"
  USING (id::text = current_setting('app.user_id', true));
```

Support time-boxed session flow
- Ops approves support_session(target_user_id, expires_at, reason).
- App middleware sets app.support_session_expires; logs audit_event support.session.started.
- All access during session writes audit_event with justification.
- On expiry, middleware revokes; log support.session.ended.

Access justification logging
- For any elevated read (role in [‘support’,’ops’,’pharmacy_admin’,’diagnostics_admin’]), include reason and request_id in audit_event.context.

DR/BCP & WhatsApp outage runbook
- Detect: Spike in notification_failed (provider codes); WA webhook outage alerts.

- Degrade: Pause OTP sends; show in-app banner; switch to backup provider if configured.
- Communicate: Status page update within 15 min; inform clinicians via email.
- Recover: Drain retries with jitter; recalc funnels; post-mortem within 48h.

PII Field Map

Field	Table	Stored	Masked in logs	Masked in support view
full_name	user	Yes	Yes	Yes (initial+***)
phone_e164	user	Yes	Yes	Yes (+***last4)
email	user	Optional	Yes	Null
dob	user	Yes	Yes	Shown
gender	user	Yes	Yes	Shown
items_json (rx)	prescription	Yes	Yes	Yes (drug names only)
tests_json	lab_order	Yes	Yes	Yes (test codes only)
results_url	lab_order	Yes	Yes	Hidden

---

9) i18n Pack (EN + Swahili)

```
{
  "en": {
    "consent.title": "Consent to care",
    "consent.body": "By continuing you consent to share data with your care team.",
    "otp.enter": "Enter the 6-digit code sent to WhatsApp",
    "otp.resend": "Resend code",
    "consult.start": "Start consult",
    "consult.end": "End consult",
    "rx.warning.pdf_disables_qr": "Downloading PDF disables QR verification.",
    "referral.sent": "Referral sent",
    "labs.results_ready": "Your lab results are ready"
  },
  "sw": {
    "consent.title": "Idhini ya huduma",
    "consent.body": "Kwa kuendelea unakubali kushiriki data na timu yako ya huduma.",
    "otp.enter": "Weka nambari ya tarakimu 6 iliyotumwa kwa WhatsApp",
    "otp.resend": "Tuma tena nambari",
    "consult.start": "Anza ushauri",
    "consult.end": "Maliza ushauri",
    "rx.warning.pdf_disables_qr": "Kupakua PDF kunalemaza uthibitishaji wa QR.",
    "referral.sent": "Rufaa imetumwa",
    "labs.results_ready": "Matokeo yako ya maabara yako tayari"
  }
}
```

---

10) Accessibility & Low-Bandwidth QA

Accessibility checklist
- Focus order: logical through chat input → send → attachments → actions.
- Labels: aria-labels for buttons; form inputs with programmatic labels.
- Contrast: minimum 4.5:1 for text; 3:1 for large text/icons.
- Tap targets: ≥ 44x44 px; spacing to avoid accidental taps.
- Keyboard: all actions operable via keyboard; visible focus rings.
- Screen reader: announce new messages, errors, and status changes.

Low-bandwidth/degraded modes
- Chat history: load last 50 messages, lazy-load older on scroll.
- Media: image/file placeholders with retry; progressive download.
- Retries: network backoff 1s/2s/5s up to 4 tries; show offline banner.
- PDFs: queue generation; notify when ready; cache last successful copy.
- WhatsApp: if outage, switch to in-app notifications and email for clinicians.
- Payloads: compress JSON; use compact schemas and ETags for GETs.