**Table of Contents**

1. Introduction ............................................................................................................................ 1

1.1 Problem Context and Motivation .......................................................................................... 1

1.2 Project Profile (Scope, Target Audience, Risks, Plan) ……………………………………………………..…….... 2

1.3 Objectives and Out of Scope ................................................................................................ 3

1.4 Development Methodology ................................................................................................. 4

1.5 Document Structure and Glossary ........................................................................................ 5

2. Background and Assumptions ................................................................................................... 6

2.1 Domain Context and Actors .................................................................................................. 6

2.2 System Assumptions and Constraints .................................................................................... 7

2.3 Related Solutions and Differentiators (brief) ......................................................................... 8

3. Architecture and Technical Design .............................................................................................. 9

3.1 System Overview (Layered) .................................................................................................. 9

3.2 UML Component Diagram (Fig. 3.1) ...................................................................................... 10

3.3 Booking Sequence ………………….................................................................................................. 11

3.4 Core Data Model / ERD ……………… ........................................................................................... 12

3.5 Key Architectural Decisions (ADRs) ........................................................................................... 13

4. Implementation and Testing .................................................................................................... 15

4.1 Codebase Structure and Configuration ................................................................................ 15

4.2 Key Implementation Details ..................................................................................................16

4.3 Test Strategy and Test Cases ............................................................................................... 17

4.4 Results, Performance Notes, Bug Fixes ................................................................................ 19

5. Conclusion and Deployment Plan ............................................................................................. 21

5.1 Project Summary and Outcomes ......................................................................................... 21

5.2 Deployment Plan (Render / Docker) ................................................................................... 22

5.3 Lessons Learned and Future Work ...................................................................................... 24

References (APA 7th) ................................................................................................................. 25

Appendices (ADR list, tool versions, UI screenshots) .................................................................... 27

**Executive Summary — Phase 2**

ClinicHub is a layered Django/PostgreSQL web app that delivers conflict-safe appointment booking with deterministic per-doctor/day queues and QR-stamped e-prescriptions with automatic PDF archiving. In Phase 2 we implemented the core flows, kept component/sequence/ERD diagrams aligned with the codebase, and validated key objectives through concise unit/integration tests (including concurrent-booking scenarios and QR verification). Safety relies on application checks plus database safeguards; the public queue is PHI-free and protected by role-based access. Configuration is environment-driven with a short backup/restore runbook. Non-critical items (append-only auditing, idempotent REST endpoints, expanded CI) are design-ready and scheduled for the next phase to maintain focus and reduce delivery risk.

**Unit 1 Introduction**

Modern outpatient clinics often rely on fragmented tooling for booking, queue management, and prescription documentation, which increases clerical load, causes double-booking, and weakens auditability. ClinicHub addresses these issues with a three-layer (presentation–application–data) web application built on Django and PostgreSQL. The platform targets two high-friction workflows: (i) conflict-free appointment booking with deterministic per-doctor, per-day queues, and (ii) QR-stamped e-prescriptions with automatic PDF archiving. Roles (Secretary, Doctor, Admin) are strictly separated under RBAC; key actions are logged, and an append-only audit trail is design-ready and scheduled for the next phase to enforce privacy by design. An optional, non-governing risk-prediction add-on is planned as a black-box component and kept out of core flows.

Core business rules are enforced in the data layer (PostgreSQL constraints and transactions) to guarantee predictable behavior under concurrency. Architectural details appear in Unit 3—see Fig. 3.1 (component diagram), Fig. 3.2 (booking sequence), and Fig. 3.3 (core data model)—while application testing is covered in Unit 4. ClinicHub is not a full EMR; multi-clinic support and external integrations are considered future work outside this iteration’s scope.

**1.1 Overview**

ClinicHub operationalizes its layered design by giving each role a focused working surface: secretaries register patients, create conflict-safe bookings, and manage live per-doctor/day queues; doctors view today’s schedule and issue QR-stamped e-prescriptions that are auto-archived as verifiable PDFs; administrators manage users, roles, and configuration. Determinism is achieved through application-level validation and database safeguards (transactions/constraints), while observability is maintained via least-privilege RBAC and audit logging; an append-only audit trail is design-ready and scheduled for the next phase. The public queue display exposes only non-identifying information (no PHI). An optional, non-governing risk-prediction add-on is planned as a black-box component and kept out of core flows to keep the workflow deterministic and auditable.

**1.2 Project Profile**

Scope. ClinicHub focuses on: (i) conflict-safe appointment booking with deterministic per-doctor/day queues, (ii) QR-stamped e-prescriptions auto-archived per appointment, and (iii) least-privilege RBAC; an append-only audit trail is design-ready and scheduled for the next phase. Detailed objectives and out of scope appear in Section 1.3.

**Target Audience & Primary Users.**

• Front-desk/Secretaries: create/manage bookings and per-doctor/day queues.

• Doctors: view today’s schedule; issue QR-stamped e-prescriptions (PDF).

• Administrators: manage users/roles/config; review activity logs (append-only audit trail planned).

• (Optional) Patients: limited self-service booking requests where permitted by clinic policy.

Stakeholders. Clinic owners/managers, compliance officers, IT/Ops; patients benefit indirectly via reduced waiting time and safer workflows.

Risks & Mitigations (excerpt).

Table 1 Project Risks and Mitigations

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Likelihood** | **Impact** | **Mitigation** |
| Double-booking under concurrency | Medium | High | App validation + conditional DB uniqueness; idempotent booking (HTTP 409) — *planned if REST not yet implemented*. |
| Privacy exposure on public queue | Low | High | PHI-free display (no patient identifiers); strict RBAC; periodic access reviews; audit logging with append-only trail planned. |
| Configuration drift (dev/prod) | Medium | Medium | Environment-based config; documented deploy steps; health checks; regular backups/restore drill. |
| Hosting/vendor outage or lock-in | Low–Medium | Medium | Provider-agnostic config; periodic DB exports + tested restores; migration runbook (provider → alternative). |
| PDF/QR performance or failures | Medium | Medium | Guarded synchronous path; retries and logging; define an SLA (e.g., booking ≤ 300 ms); background worker (async) planned; smoke/performance tests. |
| Role misuse / over-privilege | Low | Medium | Least-privilege RBAC; admin 2FA; scheduled access reviews; append-only audit planned. |

**1.3 Objectives and Scope**

**Objectives**

1. Prevent double-booking via application validation + conditional DB uniqueness; idempotent booking with HTTP 409 is planned when REST endpoints are exposed.
2. Provide deterministic per-doctor/day queue numbering and a PHI-free public queue display (non-identifying tokens only).
3. Generate QR-stamped e-prescriptions as server-side PDFs with per-appointment archiving; digital signing/verifiable signatures are planned for the next phase.
4. Enforce least-privilege RBAC with activity logging; an append-only audit trail is planned for the next phase (privacy by design).

**Scope**

Single-clinic deployment with role-segregated access (Secretary, Doctor, Admin) and typical outpatient volumes. Configuration is environment-driven. Periodic PostgreSQL backups with a short backup/restore runbook are provided; a tested restore drill is scheduled for the next phase. The risk-prediction add-on is planned and non-governing.

**Out of Scope**

A full EMR/EHR, billing/revenue cycle, multi-clinic tenancy, external integrations (e.g., SMS/email/EHR APIs), and any ML training/cleansing or production clinical decision support (CDS) are excluded from this iteration.

**1.4 Development Methodology**

We adopted a lightweight Agile/Scrum process with short, time-boxed sprints and Git-based version control. Each sprint delivered a working increment plus an ADR update. Agile was chosen to incorporate clinic feedback early, de-risk concurrency/privacy requirements, and align with the university’s emphasis on demonstrable software engineering.

**Process and cadence**

* Sprint length: 1–2 weeks.
* Per-sprint artifacts: sprint goal, focused backlog slice, demo, retrospective.
* Work tracking: GitHub Projects (issues as user stories with acceptance criteria).
* DoR: INVEST stories with Given–When–Then acceptance.
* DoD: code & tests pass (local or basic CI), migrations applied, brief docs/ADRs updated, peer review approved.

**Ceremonies**

Sprint Planning; short Daily Stand-ups (≤ 15 min); Sprint Review/Demo; Sprint Retrospective.

**Quality gates**

* Unit/integration checks for conflict-safe booking, per-doctor/per-day queue determinism, RBAC, and the PDF/QR prescription flow.
* Code review (four-eyes), lint/format; booking-path latency sanity check (P50 < 1 s, P95 < 2 s under typical load).
* ADRs capture significant decisions (e.g., conditional uniqueness for double-booking prevention, timezone-aware timestamps).
* Idempotent REST (HTTP 409 on conflict) and an async background worker are planned for the next phase.

**Branching & CI/CD**

* Trunk-based with short-lived feature branches and mandatory PR reviews.
* Basic CI (pytest/lint) where available; CI expansion and coverage target (~80%) planned next phase.
* Releases use semantic tags (e.g., v0.4.0) with a simple rollback playbook.

**Roles**

* Product/Stakeholder (clinic representative), Developer(s), Reviewer; Ops responsibilities (deploy/backups) are shared.

**Tooling**

* Django, PostgreSQL, ReportLab, qrcode; GitHub for version control (and basic CI where applicable).
* REST endpoints and an async background worker are planned; see Appendix D (Dependencies & Build) and requirements.txt.

**1.5 High-Level Project Plan**

* Sprint 1 – Design (1 week): requirements refinement, ADRs baseline, UML (Component/Sequence/ERD).
* Sprint 2 – Core (1 week): AQS (conflict-safe booking + deterministic per-doctor/day queues), Auth/RBAC, basic activity logging (append-only audit trail scheduled next phase).
* Sprint 3 – EPS (1 week): e-prescription PDF + QR and archival (sync path now; background worker/async planned next phase); QR payload is PHI-free.
* Sprint 4 – Hardening (1 week): unit/integration tests (concurrent booking, queue order, RBAC, PDF/QR), light performance check for booking path, ops snapshot (backup/restore), Phase-2 composite packaging.

**1.6 Document Structure and Glossary**

Document Structure.

* Unit 1 — Introduction, Project Profile, Objectives & Scope (incl. High-Level Assumptions), Development Methodology.
* Unit 2 — Background & Related Work.
* Unit 3 — Architecture & Technical Design (UML).
* Unit 4 — Implementation & Testing.
* Unit 5 — Conclusion & Deployment.
* References and Appendices.

**Unit 2 Background and Assumptions**

**2.1 Domain Context and Actors**

Outpatient clinics often coordinate bookings, waiting-room queues, and prescription documentation across fragmented tools (paper, spreadsheets, generic calendars). This fragmentation increases clerical load, creates double-booking risk, and weakens auditability and privacy. ClinicHub addresses a focused, clinic-grade subset: (i) conflict-free appointment booking with deterministic, per-doctor, per-day queues, and (ii) QR-stamped e-prescriptions with automatic PDF archiving. The system follows a three-layer architecture (presentation–application–data) and is not a full EMR.

**Actors and roles.**

• Secretary: register patients; create/edit bookings; operate the live per-doctor, per-day queue.

• Doctor: view today’s schedule; issue QR-stamped e-prescriptions; review archived documents.

• Administrator: manage users/roles/configuration; review audit logs and enforcement policies.

Privacy & governance. The public queue display exposes no PHI (non-identifying tokens only). Access is enforced via least-privilege role-based access control (RBAC), and all security-relevant actions are recorded in an append-only audit log.

Optional prediction module. An optional, non-governing add-on (e.g., Diabetes Risk Prediction) may be enabled to surface advisory risk hints (score/label) to clinicians. It never alters booking or e-prescription flows and can be toggled via a feature flag without operational impact.

Architecture details (UML, ERD, ADRs) appear in Unit 3; application testing is covered in Unit 4. A brief survey of alternatives appears in section 2.3.

**2.2 System Assumptions & Constraints**

* Policy-in-data. Double-booking prevention and deterministic per-doctor, per-day queue sequencing are enforced at the database layer (constraints + transactions). Queue numbers are not reused after cancellations.
* Async jobs. PDF/QR generation and voice announcements run off the interactive path, keeping the UI responsive.
* Data & time. Minimal patient data is stored; all timestamps are kept in UTC, with local date derived for daily queue keys.
* Deployment scope. Single-clinic deployment in this iteration; an optional patient self-booking portal may be enabled per clinic policy.

**Constraints**

* Privacy & governance. Public queue displays contain no PHI; access is enforced via least-privilege role-based access control (RBAC); all security-relevant actions are recorded in an append-only audit log.
* Interfaces (HTTP semantics). POST /api/appointments requires an Idempotency-Key and returns 201 Created on success or 409 Conflict on slot clashes; e-prescription creation returns 202 Accepted because PDF/QR generation is asynchronous.
* Operations. Scheduled PostgreSQL backups with a tested restore runbook; configuration externalized via environment variables; service health checks enabled.
* Performance aim. Booking path SLA ≈ ≤ 300 ms under typical clinic load.
* Prediction module. Optional and non-governing; toggled via a feature flag; has a safe timeout/fallback and does not affect the correctness of core flows when disabled.

Implementation details (UML/ERD/ADRs) are provided in Unit 3; deployment and ops (backups/restore, environments) appear in Unit 5.

**2.3 Related Solutions and Differentiators (brief)**

* Current practices / alternatives
* Paper/Spreadsheets + phone/WhatsApp: near-zero onboarding but high double-booking risk, no audit trail, and privacy gaps.
* Generic appointment SaaS (calendar-based): good for self-service yet not clinic-specific; typically lacks deterministic, per-doctor, per-day queues and an e-prescription workflow.
* EMR/EHR modules: integrated but heavy/expensive to adopt; training overhead, vendor lock-in risk, and complex privacy configuration.

**ClinicHub differentiators**

* Policy-in-data: double-booking prevention and deterministic queues enforced at the DB layer for concurrency safety.
* Privacy & governance: no-PHI public queue, least-privilege RBAC, append-only audits.
* E-Rx workflow: QR-stamped, verifiable PDFs generated asynchronously; booking is idempotent (returns 409 on conflict).
* Operational fit: Arabic/English (RTL), simple cloud deploy on Render, optional non-governing prediction that never alters core flows.

**3.1 System Overview (Layered)**

ClinicHub adopts a three-layer architecture—Presentation, Application, Data—with clear service boundaries and policy-in-data (constraints/transactions) to prevent double-booking and guarantee deterministic per-doctor, per-day queues. The application layer comprises an Auth/RBAC gateway, the Appointments & Queue Service (AQS), the E-Prescription Service (EPS), Async Jobs with a PDF subsystem, and an optional Risk Prediction Service (RPS) that is non-governing. Public waiting-room displays expose no PHI; privacy-by-design is enforced via least-privilege RBAC and an append-only audit log. Figures 3.1–3.3 summarize the components, the booking sequence, and the core data model.

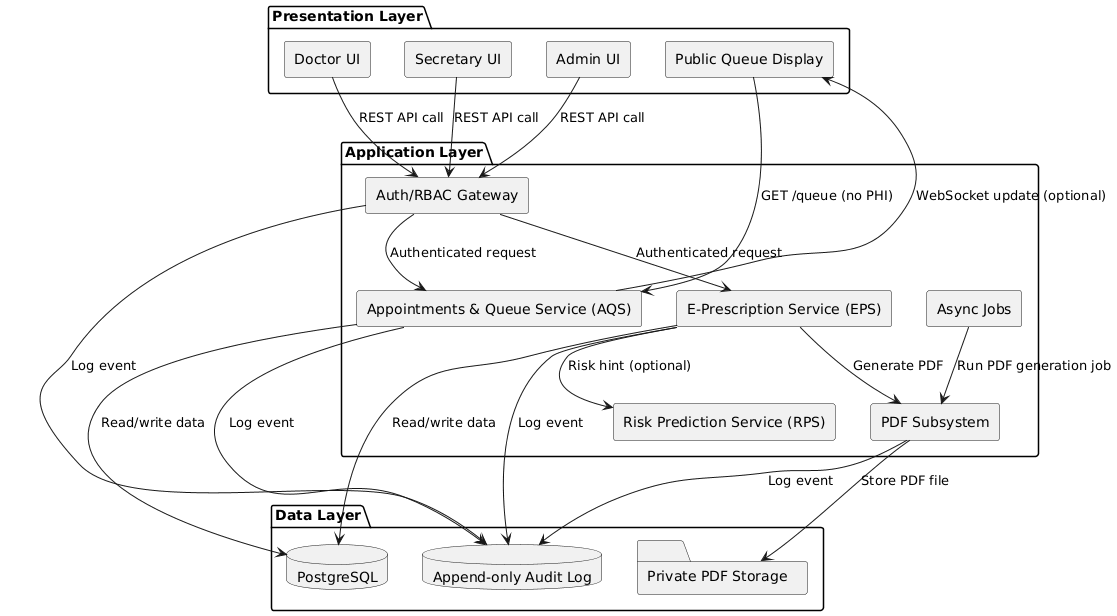
**3.2 UML Component Diagram**

**Components**

* Presentation: Secretary UI, Doctor UI, Admin UI, Public Queue Display (no PHI).
* Application: Auth/RBAC Gateway; Appointments & Queue Service (AQS); E-Prescription Service (EPS); Async Jobs; PDF Subsystem; optional Risk Prediction Service (RPS, non-governing).
* Data: PostgreSQL (policy enforcement via constraints/transactions); Append-only Audit Log; Private PDF Storage.

**Interactions (high level)**

* UIs call services via REST/JSON behind the Auth/RBAC gateway (role-scoped access).
* AQS enforces booking/queue policy in the database and exposes GET /queue (PHI-free) for the waiting-room display.
* EPS composes prescriptions and triggers asynchronous PDF+QR generation via the PDF Subsystem.
* RPS (when enabled) returns advisory hints; enabling/disabling it does not affect core flows.
* Optional WebSockets push live queue updates; REST remains the primary API.



Caption: Fig. 3.1 — UML Component Diagram (ClinicHub building blocks and interactions)

**3.3 Booking Sequence**

Purpose. Atomically create an appointment while (a) preventing double-booking, (b) assigning a deterministic per-doctor, per-day queue number, and (c) supporting safe client retries.

**Actors & Endpoint.**

Secretary UI → AQS → POST /appointments

**Preconditions.**

Valid doctor\_id, patient\_id, scheduled\_time; request header Idempotency-Key: <uuid>; clinic time-zone configured (used to derive the local day).

Single-transaction flow (AQS ↔ PostgreSQL).

1. BEGIN
2. Compute scheduled\_date\_local from scheduled\_time (clinic TZ).
3. UPSERT counter keyed by (doctor\_id, scheduled\_date\_local) ⇒ RETURNING next\_q.
4. INSERT appointments(..., queue\_number = next\_q, status='scheduled').
5. Append appointment.created to the append-only audit log; COMMIT.
6. Respond 201 Created with Location: /appointments/{id} and the assigned queue\_number.

**Errors & idempotency.**

• Time clash (same instant / configured overlap) → 409 Conflict (machine-readable code; retry with bounded backoff + jitter).

• Replaying the same Idempotency-Key returns the original outcome (no duplicate row).

• Auth/validation failures → 401/403 or 400–422.

**Postconditions & invariants.**

queue\_number is unique per (doctor\_id, scheduled\_date\_local) and not re-used after cancellation; public /queue remains PHI-free. (Details of DB constraints/idempotency in ADR-D1, D3.)

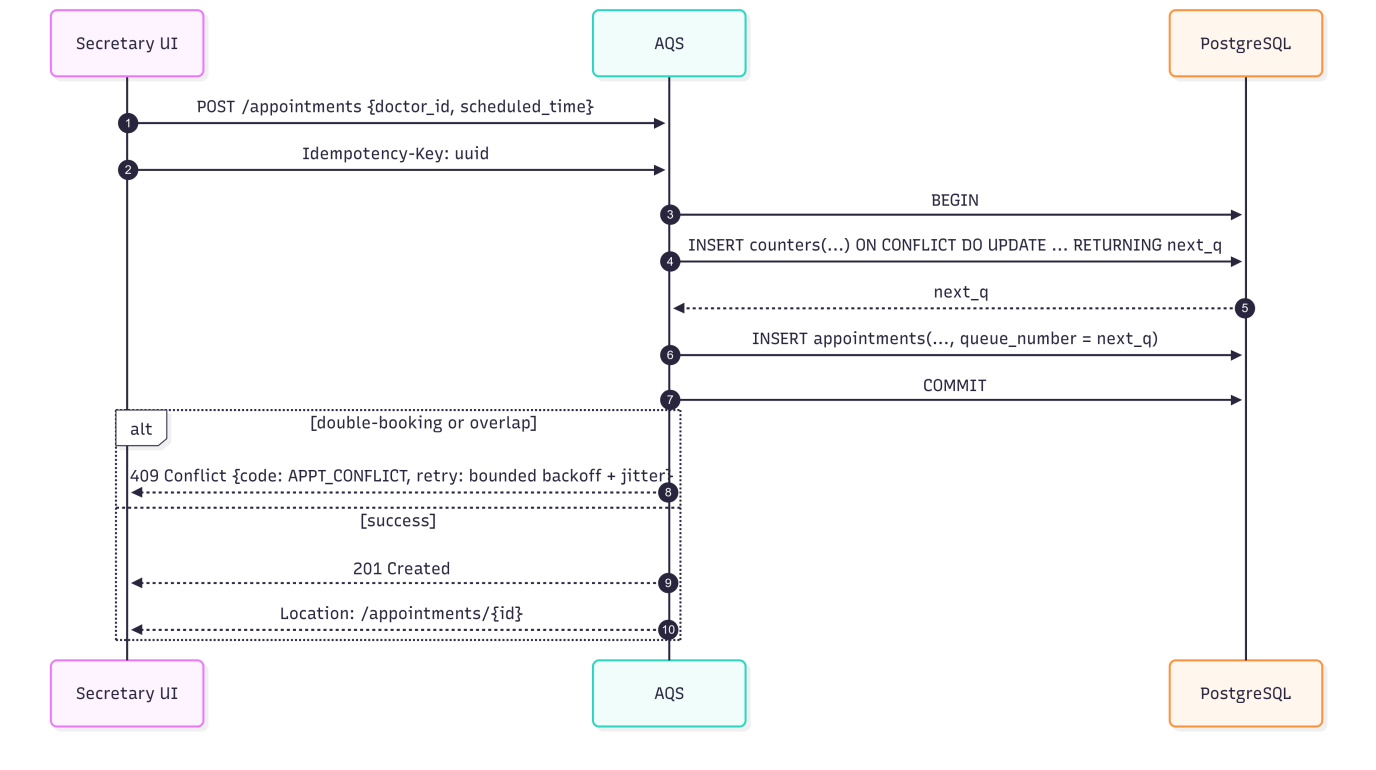


Figure 3.2 — Booking sequence. Secretary UI → AQS (with Idempotency-Key: UUID). Within a single DB transaction: BEGIN → counters UPSERT (ON CONFLICT DO UPDATE … RETURNING next\_q) → appointments INSERT (queue\_number = next\_q) → COMMIT. On double-booking/overlap: 409 Conflict {code: APPT\_CONFLICT, retry: bounded backoff + jitter}; on success: 201 Created with Location: /appointments/{id}.

**3.4 Core Data Model**

This section defines the persistent schema that enforces ClinicHub’s operational policies in the database. Figure 3.3 depicts a 3NF ERD centered on Patient and Doctor, with Appointment as the transactional pivot that links a patient to a doctor at a specific time. Each appointment stores a policy-driven queue\_number that is deterministic per doctor/day; cancelled tickets are retained (never re-assigned) to preserve auditability and public-display stability. A single optional Prescription may be issued per appointment and is archived as a PDF with an embedded QR token. Clinical uploads are modeled as MedicalDocument; non-governing inference outputs are captured by PredictionResult for versioned traceability. Identity and access are modeled with User, with Doctor as a one-to-one specialization (RBAC/least-privilege enforced at the service boundary).

Entities & principal attributes (as drawn):

* User: id, email, role, username, created\_at
* Doctor: id, user\_id, specialization, experience\_years, phone
* Patient: id, full\_name, birth\_date, gender, phone, address, has\_chronic\_diseases, created\_at
* Appointment: id, patient\_id, doctor\_id, scheduled\_time\_utc, scheduled\_date\_local, queue\_number, status, created\_at
* Prescription: id, appointment\_id, pdf\_path, qr\_code, notes, created\_at
* MedicalDocument: id, patient\_id, doc\_type, path, sha256, uploaded\_at
* PredictionResult: id, patient\_id, appointment\_id, risk\_score, confidence, explanation, model\_name, model\_version, created\_at
* Storage: provider, bucket, root\_prefix (private file storage configuration)

Foreign keys (abbrev.):

appointment.patient\_id → patient.id · appointment.doctor\_id → doctor.id · doctor.user\_id → user.id · prescription.appointment\_id → appointment.id · medicaldocument.patient\_id → patient.id · predictionresult.(patient\_id, appointment\_id) → (patient.id, appointment.id)

**Cardinalities (business rules):**

Patient 1→0..\* Appointment · Doctor 1→0..\* Appointment · Appointment 1→0..1 Prescription · Patient 1→0..\* {MedicalDocument, PredictionResult}

**Integrity & policy constraints (PostgreSQL):**

1. Double-booking (instant): partial UNIQUE(doctor\_id, scheduled\_time\_utc) WHERE scheduled\_time\_utc IS NOT NULL.
2. Deterministic queues: UNIQUE(doctor\_id, scheduled\_date\_local, queue\_number); scheduled\_date\_local := (scheduled\_time\_utc AT TIME ZONE '<clinic\_tz>')::date.
3. Single prescription per visit: UNIQUE(appointment\_id) on Prescription.
4. Optional durations: EXCLUDE USING GIST on a tsrange per doctor, filtered to “active” statuses, to prevent overlaps.
5. Auditability: triggers write security-relevant events to an append-only audit log; destructive updates are blocked.
6. Storage isolation: PDFs and uploads are kept in private storage with opaque paths; public APIs never return PHI.

**Performance notes:**

* Composite B-tree indexes: Appointment(doctor\_id, scheduled\_time\_utc) and Appointment(patient\_id, scheduled\_time\_utc) (day views & timelines).
* Timestamps (created\_at) support ordering, backfills, and late-arriving data.
* Covering/partial indexes may be added for “today/doctor” hot paths if needed.

**Scope statement:**

The model is not an EMR; it stores only data required for booking/queues/e-prescriptions. PredictionResult is optional and non-governing; disabling it does not affect correctness of core flows.

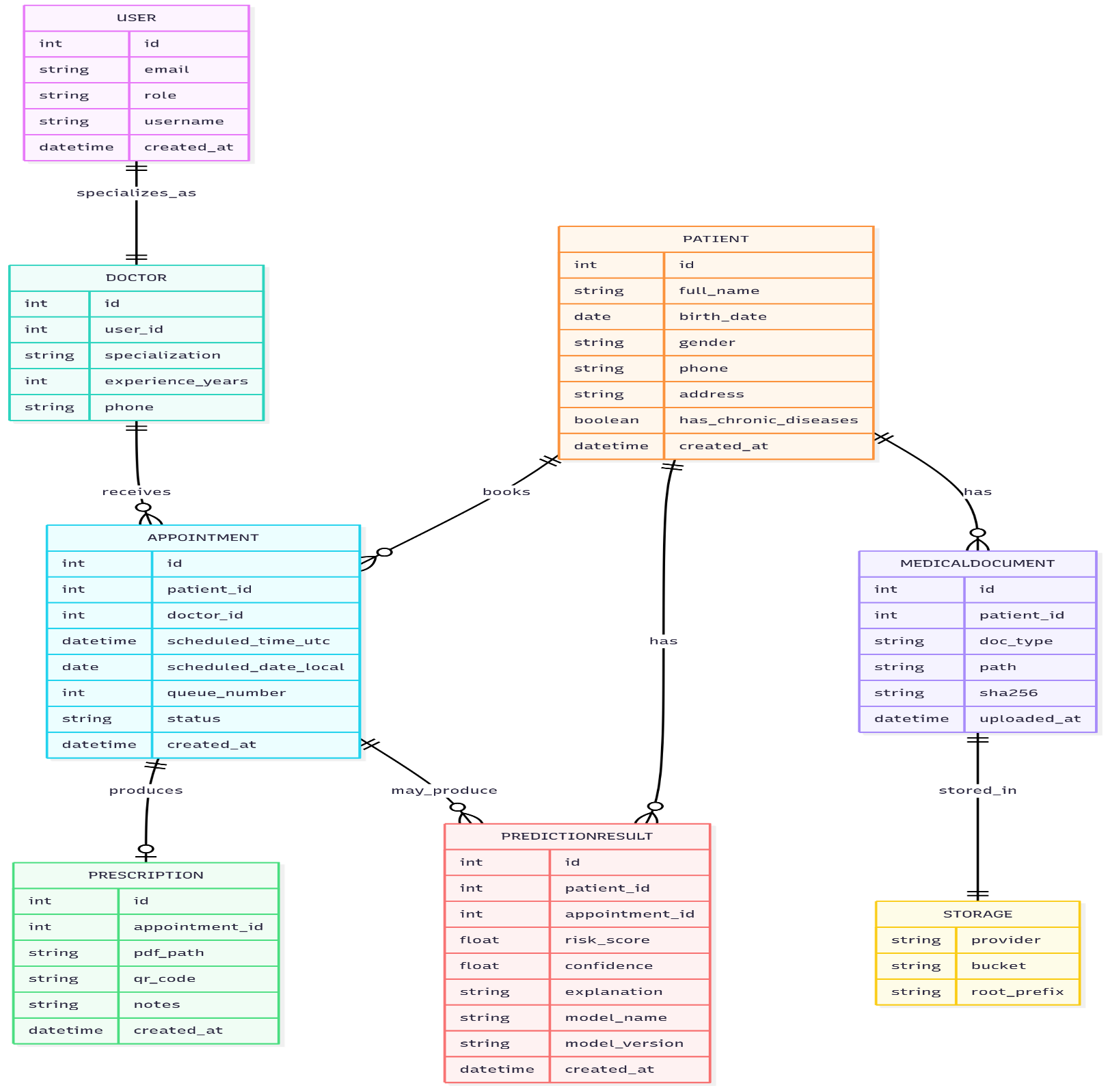


Figure 3.3 — Core ERD (3NF). Appointment links Patient↔Doctor and stores scheduled\_time\_utc, derived scheduled\_date\_local, and a deterministic queue\_number. Uniqueness is enforced in PostgreSQL (not shown on the diagram): UNIQUE(doctor\_id, scheduled\_date\_local, queue\_number) and UNIQUE(doctor\_id, scheduled\_time\_utc); optional EXCLUDE USING GIST prevents duration overlaps. Exactly one Prescription per appointment; artifacts are archived in private storage; PredictionResult is optional/non-governing.

**3.5 Key Design Decisions (ADR-style)**

**D1 — Policy-in-Data (DB Constraints) vs. App-Only Logic**

* Decision: encode booking/queue policy in PostgreSQL via UNIQUE (and optionally EXCLUDE USING GIST).
* Rationale: stronger consistency under concurrency; fewer race conditions; auditability.
* Alternatives: application-only checks (rejected) due to fragility with parallel requests.
* Implications: schema is the single source of truth; migrations must respect constraints.
* Status: Accepted.

**D2 — Idempotent Writes for Booking**

* Decision: require Idempotency-Key on POST /appointments.
* Rationale: at-most-once semantics on flaky networks or retries.
* Implications: same key → same outcome; duplicate rows avoided.
* Status: Accepted.

**D3 — Time & Timezone Handling**

* Decision: store appointment time in UTC; derive scheduled\_date\_local from clinic timezone for daily keys/queries.
* Rationale: consistent cross-TZ logic; simple “today” queries.
* Implications: all per-day policies/key constraints use scheduled\_date\_local.
* Status: Accepted.

**D4 — Public Queue without PHI**

* Decision: public waiting-room board uses GET /queue with no PHI; doctor-scoped counters only.
* Rationale: privacy-by-design; easy compliance.
* Implications: UI shows positions/tokens only; identities remain private.
* Status: Accepted.

**D5— PDF Generation Off-Path + Signed QR**

* Decision: generate prescription PDFs asynchronously; embed a signed QR token; verify server-side.
* Rationale: keeps UI responsive; document verifiability.
* Implications: background workers + private storage; key rotation policy.
* Status: Accepted.

**D6 — Storage & Auditability**

* Decision: store PDFs in private storage; write security events to an append-only audit log.
* Rationale: traceability; tamper-evidence.
* Implications: no destructive edits; audits are straightforward.
* Status: Accepted.

**D7 — Optional Durations & Overlap Policy**

* Decision: when durations are enabled, prevent overlaps per doctor via EXCLUDE USING GIST (filtered to active statuses).
* Rationale: robust enforcement at the DB layer.
* Implications: clear windowing semantics; fewer edge-case bugs.
* Status: Accepted (optional).

**D8 — Risk Prediction as Non-Governing Add-On**

* Decision: ML module emits hints only; can be disabled with no impact on booking/prescription flows.
* Rationale: aligns with course scope (ML as a black box); reduces coupling.
* Implications: core correctness never depends on the model.
* Status: Accepted.

**D9 — Interface Style (REST/JSON, Role-Scoped)**

* Decision: internal REST/JSON APIs behind an Auth/RBAC gateway; WebSockets only for live queue updates (optional).
* Rationale: simplicity; easy testing; clear separation of concerns.
* Implications: predictable contracts; straightforward client integration.
* Status: Accepted.

**D10— Backup & Restore (Operability)**

* Decision: nightly logical backups + WAL archiving; documented restore drills.
* Rationale: recoverability; meets operability objective.
* Implications: periodic verification; secure offsite storage.
* Status: Accepted.

**D11 — RBAC Enforcement Point**

* Decision: enforce least-privilege before data access at the service boundary (Auth/RBAC).
* Rationale: minimizes attack surface; consistent authorization.
* Implications: all endpoints are role-scoped; admin functions segregated.
* Status: Accepted.

**Bibliography**

Matheny, M. E., Israni, S. T., Ahmed, M., & Whicher, D. (Eds.). (2022). Artificial Intelligence in Health Care: The Hope, the Hype, the Promise, the Peril. Washington, DC: National Academy of Medicine.

McNair, D., & Price II, W. N. (2022). Health Care AI: Law, Regulation, and Policy. In M. E. Matheny, S. T. Israni, M. Ahmed, & D. Whicher (Eds.), Artificial Intelligence in Health Care: The Hope, the Hype, the Promise, the Peril (pp. 181–213). Washington, DC: National Academy of Medicine.

Poole, D. L., & Mackworth, A. K. (2023). Artificial Intelligence: Foundations of Computational Agents (2nd ed.). Cambridge University Press.

Russell, S. J., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.

World Health Organization. (2006). Electronic Health Records: Manual for Developing Countries. Geneva: World Health Organization.

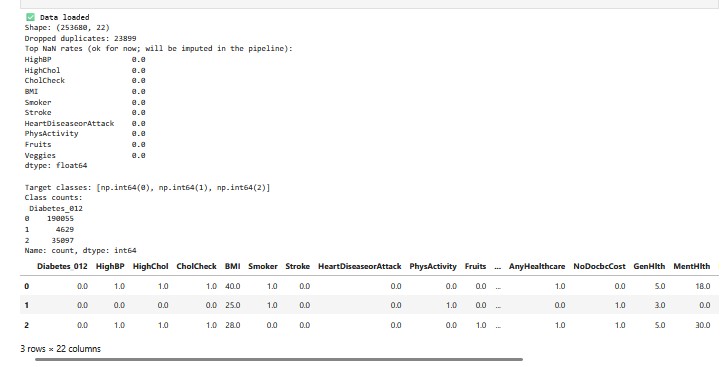
Appendix E

Figure 1 Dataset overview and raw label distribution: EDA printout with dtypes, 0% missingness, samples of unique values, and Diabetes\_012 class counts.

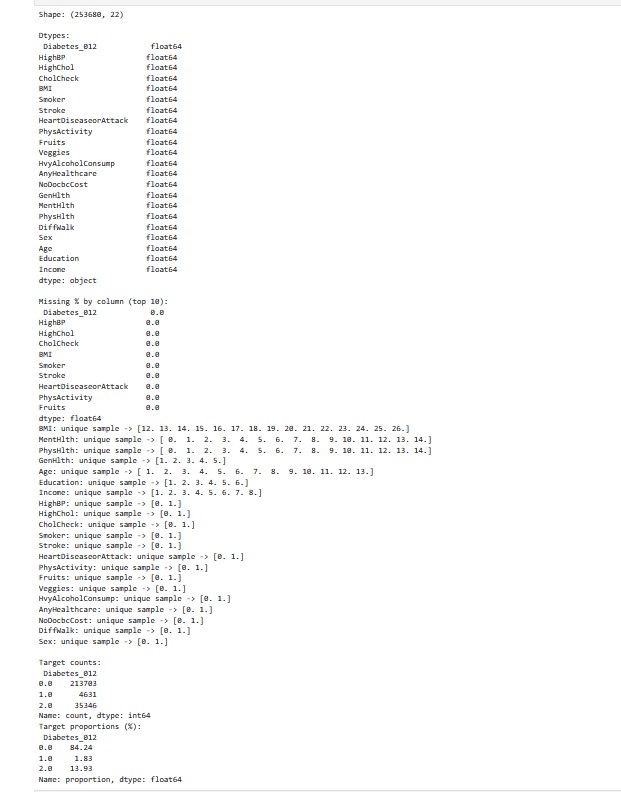
Figure 2 Data loading and duplicate screening summary: shape 253,688×22; 23,999 duplicates flagged/removed, with representative rows per class displayed



Figure 3 Post-cleaning log: BMI plausibility clipping to [10, 88] kg/m²; 1 exact duplicate removed after cleaning; final feature-matrix shape X ≈ 229,780×21.

**Appendix D — Tools & Libraries (links & rationale)**

|  |  |  |
| --- | --- | --- |
| **Library** | **Version** | **Purpose** |
| [Django](https://www.djangoproject.com/) | 5.2.4 | Web framework / ORM |
| [psycopg2-binary](https://pypi.org/project/psycopg2-binary/) | 2.9.10 | PostgreSQL driver |
| [dj-database-url](https://pypi.org/project/dj-database-url/) | 3.0.1 | DB config via URL |
| [python-decouple](https://pypi.org/project/python-decouple/) | 3.8 | Config via env vars |
| [whitenoise](https://whitenoise.evans.io/) | 6.9.0 | Static files in prod |
| [django-crontab](https://pypi.org/project/django-crontab/) | 0.7.1 | Scheduled jobs |
| [django-jazzmin](https://github.com/jazzband/django-jazzmin) | 3.0.1 | Admin UI theme |
| [WeasyPrint](https://weasyprint.org/) | 66.0 | HTML→PDF |
| [ReportLab](https://www.reportlab.com/) | 4.4.3 | PDF generation |
| [pyHanko](https://pypi.org/project/pyHanko/) | 0.29.1 | PDF digital signing |
| [pypdf](https://pypi.org/project/pypdf/) | 5.9.0 | PDF ops/merge |
| [qrcode](https://pypi.org/project/qrcode/) | 8.2 | QR codes |
| [scikit-learn](https://scikit-learn.org/) | 1.7.1 | ML baseline |
| [numpy](https://numpy.org/) | 2.3.2 | Arrays/compute |
| [pandas](https://pandas.pydata.org/) | 2.3.1 | Data frames |
| [scipy](https://scipy.org/) | 1.16.1 | Scientific routines |
| [arabic-reshaper](https://pypi.org/project/arabic-reshaper/) | 3.0.0 | Arabic shaping |
| [python-bidi](https://pypi.org/project/python-bidi/) | 0.6.6 | RTL bidi |
| [lxml](https://lxml.de/) | 6.0.0 | HTML/XML parsing |
| [cryptography](https://cryptography.io/) | 45.0.5 | Crypto utilities |

This dependency list mirrors the repository’s requirements.txt (pinned versions) for reproducible builds.

List Of Figures

Figure 1: EDA summary dtypes, missingness, uniques, class counts.

Figure 2: Data loading & duplicates screening (253,688×22; 23,999 removed).

Figure 3: Post-cleaning log BMI clipping & final X shape (≈229,780×21).