USMLE API – Contract & Project Snapshot

"Living" Engineering Handbook

HelpUS

February 1, 2026

Contents

List of Figures

List of Tables

About this document

This document is the **anchor** for the USMLE practice platform project. Its goal is to allow the team to restart work in a new chat or a new day by consulting a single source of truth.

How to use

- Keep this document updated as the system evolves.
- When resuming work: open `main.tex` and use the table of contents.
- For code changes: follow the mandatory workflow rules in chapter 1.

Scope

This handbook captures:

- Collaboration rules and development workflow;
- Stack and architecture decisions;
- API contract and routes;
- Data model (tables, enums, integrity rules);
- UI/UX blueprint for the Review screen;
- Infrastructure and deployment notes;
- Current status and recommended next steps.

Living document

This is a living document; it is expected to change frequently. Use LaTeX cross-references (e.g., chapter 6) to keep the structure stable as content grows.

Chapter 1

Rules of engagement (Mandatory)

This chapter defines **mandatory rules of engagement** for all work conducted under this handbook.

These rules apply whenever this document is provided, referenced, or treated as the source of truth for the project, **independently of conversation, session, time, or tooling**.

They are designed to prevent context loss, regressions, unsafe edits, and non-reproducible changes.

If a rule conflicts with convenience, speed, or assumptions, **the rule takes precedence**.

1.1 Scope and applicability

The rules in this chapter apply to:

- Human collaborators working on the project;
- Any AI assistant involved in analysis, design, or implementation;
- Any future session where this handbook is reused or reintroduced.

Rules explicitly marked as **Assistant-only** are mandatory for the AI assistant. Rules marked as **General** apply to all collaborators.

1.2 Assistant-only mandatory rules

The following rules are **explicit instructions to the assistant** and must be followed whenever this handbook governs the interaction.

Failure to follow these rules is considered a process violation.

1.2.1 Before any code or document change (Assistant-only)

Before proposing or applying any modification to a file, the assistant must ask:

> **"Paste the current full contents of file X."**

Only after the user pastes the **entire file**, the assistant may respond with the **entire updated file**, ready for direct copy/paste.

No partial delivery The assistant must not deliver diffs, snippets, or partial patches as the primary output. The default and mandatory format is the **complete file**.

No assumptions The assistant must not assume file contents, project state, folder structure, or previous changes that were not explicitly pasted or documented in this handbook.

1.2.2 Explicit change notification and pause (Assistant-only)

Whenever a modification is required, the assistant must:

- Explicitly state that a change is being proposed;
- Provide exactly one updated file per step (unless otherwise approved);
- Deliver the **entire file** as output;
- Ask the user to apply the change and report results;
- **Pause** and avoid proposing further changes until confirmation.

Parallel or speculative changes are forbidden.

1.2.3 One step at a time (Assistant-only)

The assistant must enforce incremental execution:

- One change;
- One file;
- One verification step;
- User confirmation;
- Only then proceed.

If the user has not confirmed the previous step, the assistant must wait.

1.3 General collaboration rules

The following rules apply to all collaborators, human or AI.

1.3.1 Incremental and testable changes

All changes must be small, reviewable, and testable.

- No multi-file “big bang” updates;
- No large unverified refactors;
- Each change should have a clear verification path.

1.3.2 Testing and verification

After each change, at least one verification action must occur:

- Local build or execution;
- Relevant user flow validation;
- Or explicit error output for diagnosis.

Where available, automated tests and linting should be preferred.

1.4 Quality and integrity gates

The following quality gates are mandatory:

- Preserve backward compatibility unless a breaking change is explicit;
- Prefer idempotent APIs and database constraints for integrity;
- Never bypass data integrity rules for convenience;
- Avoid introducing new dependencies unless explicitly justified.

1.5 Documentation-first discipline

This handbook is the **source of truth**.

When a new behavior, feature, or rule is introduced:

- Documentation must be updated first or in the same step;

- Implementation must align with documented contracts;
- Divergence between code and documentation is considered a defect.

This rule applies across conversations and over time.

1.6 Persistence across conversations

Whenever this handbook (or an excerpt containing this chapter) is provided in a new conversation, the assistant must treat these rules as **active and binding**, regardless of prior context availability.

The absence of chat history does not invalidate these rules.

Chapter 2

Stack and architecture

2.1 Current stack

- **Framework:** Next.js (App Router) [1]
- **Auth:** NextAuth v4.x (confirmed in production: 4.24.13) [2]
- **Database:** PostgreSQL [3]
- **ORM:** Prisma (schema already exists in the repository) [4]
- **Validation:** Zod [6]
- **Hosting/Deploy:** Vercel [5]

2.2 Database access

All SQL is executed through a transaction helper (see section 2.2.1).

2.2.1 `withTx` helper

- All queries are performed via `client.query`.
- Always inside a transaction.
- Avoid mixing Prisma client operations with raw SQL in the same request flow.

2.3 High-level architecture

At MVP level, the backend implements the Session lifecycle:

1. Create session (`in_progress`);

2. Generate session items (idempotent);
3. Record attempts (idempotent per item);
4. Submit session (`submitted`);
5. Review submitted sessions only;
6. Aggregate stats for submitted sessions.

Chapter 3

Repository layout (snapshot)

3.1 Backend routes (App Router)

```
src/
|-- app/
|   |-- api/
|   |   |-- auth/
|   |   |   `-- [...nextauth]/
|   |   |       `-- route.ts
|   |   |
|   |   |-- sessions/
|   |   |   |-- route.ts
|   |   |   `-- [sessionId]/
|   |   |       |-- items/route.ts
|   |   |       |-- submit/route.ts
|   |   |       `-- review/route.ts
|   |   |
|   |   |-- session-items/
|   |   |   `-- [sessionItemId]/
|   |   |       `-- question/route.ts
|   |   |
|   |   |-- sessions/
|   |   |   `-- [sessionId]/
|   |   |       `-- items/
|   |   |           `-- [sessionItemId]/
|   |   |               `-- attempt/route.ts
|   |   |
|   |   |-- me/
```

```
|   |   |   `-- stats/route.ts
|   |   |
|   |   |-- health/route.ts
|   |   |-- debug/headers/route.ts
|   |   `-- dev/seed-minimal/route.ts
|   |
|   `-- session/
|       `-- [sessionId]/
|           |-- page.tsx
|           `-- review/page.tsx
|
`-- lib/
    |-- db.ts
    |-- auth.ts
    `-- apiClient.ts
```

3.2 Client helper

The UI uses a shared HTTP helper:

`src/lib/apiClient.ts`

Chapter 4

Authentication contract

4.1 Development override header

`x-user-id: <UUID>`

When this header is present:

- NextAuth is fully bypassed.
- The value is used as `user_id`.

4.2 Browser/production

When section 4.1 is not present:

- Use NextAuth v4 session via `getServerSession(authOptions)`.
- Derive `user_id` deterministically from `session.user.email`.

4.3 Deterministic user id

Rule:

1. If `x-user-id` exists, use it.
2. Else, take `session.user.email`.
3. Generate a deterministic UUID from email.

This guarantees the same email maps to the same `user_id` across environments.

Chapter 5

API contract

This chapter defines the public API contract of the system. It specifies endpoints, request and response formats, and behavioral rules.

All endpoints described here must comply with the data integrity rules defined in chapter 6.

5.1 Design principles

The API is designed according to the following principles:

- Stateless HTTP semantics;
- Idempotent write operations whenever applicable;
- Database as the system of record;
- Clear separation between session lifecycle and user actions;
- Predictable error behavior.

5.2 Sessions

5.2.1 Create session

`POST /api/sessions`

Creates a new study session with initial status `in_progress`.

Request body

```
{
  "exam": "step1",
```

```
  "mode": "practice" | "timed_block" | "exam_sim"
}
```

Response (example)

```
{
  "session_id": "...",
  "user_id": "...",
  "exam": "step1",
  "mode": "practice",
  "language": "en",
  "timed": false,
  "time_limit_seconds": null,
  "status": "in_progress",
  "started_at": "...",
  "submitted_at": null
}
```

5.2.2 List sessions

`GET /api/sessions`

Returns all sessions belonging to the authenticated user.

Only metadata is returned; session items are not included.

5.2.3 Generate session items

`POST /api/sessions/:sessionId/items`

Generates the ordered list of session items.

Rule (Idempotency) If items already exist for the session, the endpoint must return the existing items and must not recreate them.

This behavior is enforced by application logic and supported by the data model described in chapter 6.

5.2.4 Submit session

`POST /api/sessions/:sessionId/submit`

Finalizes a session.

Effects

- `status` transitions from `in_progress` to `submitted`;
- `submitted_at` is set.

Once submitted, a session becomes immutable.

5.2.5 Review session

`GET /api/sessions/:sessionId/review`

Returns the full review of a submitted session, including:

- questions;
- user attempts;
- correctness information.

Rule The session must be in status `submitted`. Otherwise the API returns:

```
{ "error": "Session must be submitted to review" }
```

5.3 Session items

5.3.1 Get question

`GET /api/session-items/:sessionItemId/question`

Returns the question stem and its answer choices.

Rule The correct answer must not be revealed at this stage.

5.3.2 Attempt question

`POST /api/sessions/:sessionId/items/:sessionItemId/attempt`

Records the user's attempt for a single session item.

Rule (Uniqueness) At most one attempt is allowed per session item.

This invariant is enforced by the database constraint documented in section 6.4.1.

Rule (Idempotency) Repeated requests with the same payload must not create duplicate attempts.

5.4 User statistics

5.4.1 Get statistics

`GET /api/me/stats?range=30`

Returns aggregated performance metrics for the authenticated user.

Rules

- Only sessions with status `submitted` are considered;
- `range` is expressed in days (1–365);
- Default range is 30 days.

5.5 Utility and development endpoints

5.5.1 Health check

`GET /api/health`

Simple liveness probe for infrastructure monitoring.

5.5.2 Debug headers

`GET /api/debug/headers`

Echoes request headers to assist in authentication debugging, especially during development.

5.5.3 Development seed

`POST /api/dev/seed-minimal`

Seeds minimal development data.

Warning This endpoint must never be enabled in production.

5.6 Endpoint summary

Table 5.1: Primary API endpoints

Area	Method / Path	Purpose
Sessions	POST /api/sessions	Create session
Sessions	POST /api/sessions/:id/items	Generate items (idempotent)
Sessions	POST /api/sessions/:id/submit	Submit session
Sessions	GET /api/sessions/:id/review	Review submitted session
Items	GET /api/session-items/:id/question	Retrieve question
Attempts	POST /api/sessions/:sid/items/:iid/attempt	Record attempt
Stats	GET /api/me/stats?range=N	Aggregate statistics

Chapter 6

Data model

This chapter defines the canonical PostgreSQL data model for the USMLE platform. It is the single source of truth for table structure, relationships, and integrity rules.

All API behavior described in chapter 5 must remain consistent with this model.

6.1 Scope and principles

The data model is designed with the following principles:

- Explicit integrity constraints at the database level;
- Idempotent writes enforced by schema rules;
- Clear separation between session lifecycle and user actions;
- Forward compatibility with future analytics and content expansion.

6.2 Enums

Table 6.1: PostgreSQL enums

Enum	Values
`attempt_result`	`correct`, `wrong`, `skipped`
`session_status`	`in_progress`, `submitted`
`session_mode`	`practice`, `timed_block`, `exam_sim`

6.3 Tables

6.3.1 `sessions`

Represents a single study session created by a user.

Table 6.2: Table `sessions`

Column	Type	Notes
`session_id`	uuid	Primary key
`user_id`	uuid	Deterministic user identifier
`exam`	text	e.g. `step1`, `step2ck`
`mode`	session_mode	Session behavior profile
`language`	text	Default: `en`
`timed`	boolean	Derived from mode
`time_limit_seconds`	int	Nullable; set for timed modes
`status`	session_status	Lifecycle state
`started_at`	timestamptz	Creation timestamp
`submitted_at`	timestamptz	Nullable; set on submission

Primary relationships

- One `sessions` row owns many `session_items`;
- One `sessions` row owns many `attempts`.

6.3.2 `session_items`

Represents the ordered list of questions presented within a session.

Table 6.3: Table `session_items`

Column	Type	Notes
`session_item_id`	uuid	Primary key
`session_id`	uuid	FK → `sessions.session_id`
`position`	int	1-based order inside the session
`question_version_id`	uuid	Snapshot of the question version shown

Notes

- `position` is immutable after generation;
- `question_version_id` ensures historical consistency even if questions change.

6.3.3 `attempts`

Stores the user's interaction with a single session item.

Table 6.4: Table `attempts`

Column	Type	Notes
`attempt_id`	uuid	Primary key
`user_id`	uuid	Redundant but intentional (query efficiency)
`session_id`	uuid	FK → `sessions.session_id`
`session_item_id`	uuid	**UNIQUE**; FK → `session_items.session_item_id`
`question_version_id`	uuid	Copied from session item
`selected_choice_id`	uuid	Nullable (skipped questions)
`result`	attempt_result	Normalized outcome
`is_correct`	boolean	Nullable; denormalized helper
`time_spent_seconds`	int	Nullable
`confidence`	smallint	Nullable (future UX)
`flagged_for_review`	boolean	Default false
`answered_at`	timestamptz	Timestamp of final attempt

6.4 Integrity and idempotency

6.4.1 Uniqueness rule for attempts

- `attempts.session_item_id` is **UNIQUE**.

This guarantees:

- At most one attempt per session item;
- Natural idempotency for the attempt endpoint;
- Race-condition safety without application locks.

6.4.2 Referential integrity

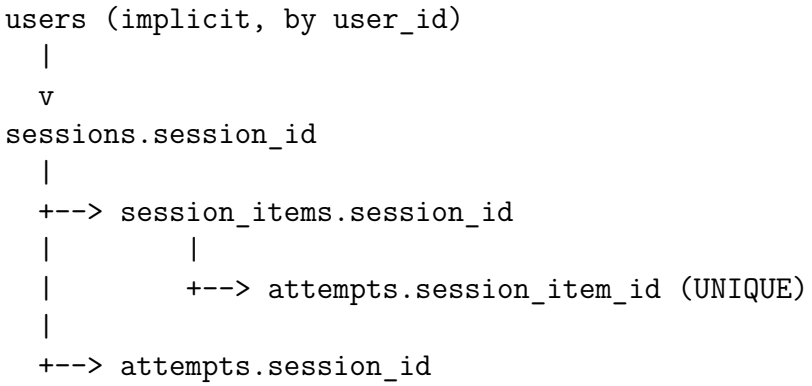
All foreign keys enforce:

- `ON DELETE RESTRICT`;
- `ON UPDATE NO ACTION`.

This prevents accidental loss of historical data.

6.5 Relationship diagram (textual)

Figure 6.1: Relationship overview (logical)



6.6 Model evolution log

This section documents structural changes to the data model over time.

2026-01-27 — Initial MVP schema

- Introduced `sessions`, `session_items`, and `attempts`;
- Added enum-based normalization for status and result fields.

2026-01-29 — Idempotency hardening

- Added `UNIQUE(session_item_id)` constraint to `attempts`;
- Ensured safe retry semantics for the attempt endpoint.

Future changes

All future schema changes must:

- Be appended to this section with date and rationale;
- Preserve backward compatibility for submitted sessions;
- Be reflected in API behavior described in chapter 5.

Chapter 7

UI/UX Blueprint: Session Review

This chapter defines the **Review screen** as the primary learning surface of the platform.

The question player (pre-submit) is optimized for exam simulation and cognitive load control; the review experience is optimized for reflection, explanation, and knowledge consolidation.

7.1 Design goals

The review interface is designed to:

- Transform every answered question into a structured learning event;
- Make correct and incorrect reasoning immediately distinguishable;
- Provide layered educational explanations without overwhelming the learner;
- Support long-form reading and review comfort;
- Enable future enrichment with external educational resources.

The review experience prioritizes clarity, semantic signaling, and pedagogical depth over exam realism.

7.2 High-level layout overview

The layout is designed mobile-first, while remaining fully functional on desktop devices.

7.3 Semantic states of a question

Each question progresses through well-defined visual and semantic states.

Figure 7.1: Review page layout (conceptual wireframe)

```
[ Sticky header: session info | timer | controls ]
-------------------------------------------------
[ Question navigator: 1..N with semantic status ]
-------------------------------------------------
[ Question stem (read-only, adjustable font) ]
-------------------------------------------------
[ Answer choices with semantic highlighting ]
-------------------------------------------------
[ Educational explanation blocks ]
  - Educational Objective
  - Correct answer explanation
  - Incorrect option explanations (accordion)
  - Key Concept / Bottom Line
  - Exam Tip (optional)
-------------------------------------------------
[ References & external learning resources ]
-------------------------------------------------
[ Prev / Next navigation ]
```

7.3.1 Unanswered state (player)

- Neutral styling for all answer choices;
- No correctness indicators;
- Focus on exam-like interaction and decision making.

This state is handled exclusively by the session player and not by the review interface.

7.3.2 Answered state (immediate review)

- Correct answer highlighted using positive semantic color (e.g., green);
- Incorrect answers highlighted using negative semantic color (e.g., red);
- The learner's selected choice explicitly indicated;
- Explanations become visible but structured.

This state provides immediate pedagogical feedback while the context is fresh.

7.3.3 Post-session review state

- Identical semantic signaling as the answered state;
- Expanded educational blocks enabled by default;

- References and external resources fully accessible;
- Navigation optimized for review rather than speed.

The post-session state is the primary surface for deep learning.

7.4 Answer choice presentation

Answer choices are presented as interactive cards rather than plain radio buttons.

Design principles

- Each choice remains readable as a standalone statement;
- Semantic color is applied only after answering;
- Visual emphasis distinguishes correctness without relying on color alone;
- Explanations are spatially associated with their respective choices.

Incorrect choices are not hidden or collapsed by default, reinforcing the importance of understanding distractors.

7.5 Educational explanation blocks

The review interface organizes explanations into structured blocks.

7.5.1 Educational Objective

A concise statement describing the core learning goal of the question.

This block answers the question: *"What was this question trying to test?"*

7.5.2 Correct answer explanation

A detailed explanation justifying why the correct choice is correct, grounded in clinical reasoning or scientific evidence.

7.5.3 Incorrect option explanations

Each incorrect choice includes a dedicated explanation, typically presented as an accordion to reduce visual overload.

7.5.4 Key Concept / Bottom Line

A distilled takeaway summarizing the most important principle the learner should retain.

7.5.5 Exam Tip (optional)

Short, exam-oriented guidance highlighting common traps, heuristics, or high-yield reminders.

7.6 Reader and interaction controls

To support long reading sessions and accessibility, the review interface includes optional controls:

- Font size increase and decrease;
- Language selection (where content is available);
- Flagging questions for later review;
- Lightweight engagement signals (e.g., like or bookmark).

These controls influence presentation only and never modify historical session data.

7.7 References and external learning resources

At the end of the review content, references and optional external learning resources are presented.

Characteristics

- References are clickable and open externally;
- External resources are linked, not embedded or hosted;
- Resources may include text or audio formats;
- Presentation is clearly separated from core explanations.

This design enables integration with high-quality third-party educational material while preserving system ownership of learning data.

7.8 Navigation behavior

Navigation within review mode differs from the player:

- Navigation is non-linear via the question navigator;
- Previous and next controls remain available;
- No actions can modify attempts or results.

Review navigation is optimized for reflection rather than exam pacing.

7.9 Relationship to other chapters

This chapter operationalizes concepts defined in:

- Content and question model (chapter 11);
- Session and attempt semantics (chapter 5);
- External integration constraints (Appendix G).

Any change to the review experience must remain consistent with these foundations.

7.10 Conclusion

By treating the review interface as the primary learning surface, the platform shifts focus from correctness alone to understanding and retention.

This blueprint establishes a scalable, pedagogically sound foundation for advanced educational features while preserving architectural clarity.

Chapter 8

Infrastructure and deployment

8.1 Vercel deployment

The project is deployed on Vercel with automatic deployments triggered by GitHub pushes.

8.1.1 Deploy trigger policy

- **Enabled:** GitHub integration (push to `main`)
- **Disabled:** external *Deploy Hooks* (to avoid duplicate deployments)

Chapter 9

Current status and roadmap

9.1 Current status

- Backend validated locally and in production.
- Session lifecycle working: create → items → attempt → submit → review.

Chapter 10

System Analysis and Design Process

This chapter documents the analytical process that led to the current system architecture, data model, and API design.

Its purpose is not to describe *what* the system does, but to record *how and why* design decisions were made from a systems analysis perspective.

This chapter serves as a permanent design record.

10.1 Problem context

The system operates in the domain of medical education, specifically USMLE exam preparation. This domain imposes constraints that go beyond standard quiz or assessment platforms.

Key contextual factors include:

- The need for reproducibility of learning sessions;
- The requirement to review past answers under the exact conditions in which they were originally presented;
- The importance of tracking user performance over time;
- The necessity of auditability for educational correctness.

From an analytical standpoint, the core problem is not simply to present questions, but to model *a learning process that unfolds over time.*

10.2 Requirements analysis

10.2.1 Functional requirements

From the domain analysis, the following functional requirements were identified:

- Create and manage study sessions;
- Generate a fixed set of questions per session;
- Record exactly one definitive attempt per question per session;
- Allow session submission and post-submission review;
- Aggregate performance statistics across sessions.

10.2.2 Non-functional requirements

Equally important were non-functional requirements, which directly shaped the data model:

- Idempotency of write operations;
- Resistance to race conditions;
- Historical consistency even if questions change later;
- Predictable behavior under retries and network failures;
- Clear separation between transient UI state and persistent truth.

10.3 Analysis of modeling alternatives

Several alternative designs were considered and explicitly rejected.

10.3.1 Attempt-only model

A naive design would store only attempts, without explicit session items.

This approach was rejected because:

- It makes question ordering implicit and fragile;
- It complicates review and replay of sessions;
- It couples question selection too tightly to attempts.

10.3.2 Multiple attempts per question

Allowing multiple attempts per question was considered.

This approach was rejected because:

- It does not reflect real exam conditions;
- It complicates scoring and analytics;
- It weakens the educational signal of a single decisive answer.

10.3.3 Application-level idempotency only

Relying solely on application logic to prevent duplicate attempts was rejected.

This approach was rejected because:

- It is vulnerable to concurrency issues;
- It cannot guarantee correctness under retries;
- It shifts responsibility away from the database, which is the system of record.

10.4 Key architectural decisions

10.4.1 Explicit session items

The introduction of the `session_items` table was a deliberate decision to materialize the structure of a session.

This provides:

- Stable ordering of questions;
- A clear unit of work for attempts;
- A durable reference point for review.

10.4.2 Database-enforced idempotency

The `UNIQUE(session_item_id)` constraint in the `attempts` table enforces a core invariant at the database level.

This ensures:

- Exactly one attempt per session item;
- Safe retries without duplicate data;
- Simplified application logic.

10.4.3 Propagation of `question_version_id`

The explicit propagation of `question_version_id` across tables ensures historical accuracy.

This guarantees that:

- Reviews reflect the exact content originally shown;
- Future edits to questions do not retroactively affect past sessions;
- Analytics remain interpretable over time.

10.5 Risk analysis and mitigation

The following risks were identified and addressed through design:

- **Race conditions:** mitigated via database constraints;
- **Partial writes:** mitigated via transactional boundaries;
- **Schema drift:** mitigated via explicit evolution logging (see section 6.6);
- **Inconsistent analytics:** mitigated by considering only submitted sessions.

10.6 Analyst workflow

The analyst workflow followed an iterative cycle:

1. Domain understanding and requirement elicitation;
2. Hypothesis of a candidate model;
3. Validation against edge cases and failure modes;
4. Refinement of constraints and relationships;
5. Documentation of decisions and rationale.

Documentation is treated as a first-class artifact, not an afterthought.

10.7 Relationship to other chapters

This chapter provides the analytical foundation for:

- **API behavior** described in chapter 5;
- **Data integrity rules** defined in chapter 6;
- **Future evolution** documented in the project roadmap.

Any future architectural change should be reflected here before being implemented in code.

10.8 Conclusion

By explicitly recording the analytical process, the system becomes easier to evolve, audit, and reason about.

This chapter ensures that future contributors understand not only *what* the system does, but *why* it was designed this way.

Chapter 11

Content and Question Model

This chapter defines the conceptual and logical model for educational content used in the USMLE platform.

It focuses on how questions, answer choices, explanations, and educational references are structured to support learning, review, and long-term consistency.

This chapter complements the structural database model defined in chapter 6 and the analytical rationale described in chapter 10.

11.1 Educational principles

The content model is designed according to the following educational principles:

- Learning is driven by explanation, not only correctness;
- Each answer option must be pedagogically meaningful;
- Learners benefit from multiple layers of explanation;
- References must be explicit, verifiable, and optional;
- Content must remain historically stable once presented to a user;
- The system must support iterative improvement over time.

11.2 Core content entities

From a conceptual standpoint, the educational domain is composed of the following entities:

- Question;
- Question version;

- Answer choice;
- Explanation;
- Educational explanation block;
- Bibliographic or external reference.

Each entity serves a distinct pedagogical and technical role.

11.3 Question and versioning model

Questions are treated as versioned artifacts.

Rationale Medical knowledge evolves, and questions may be refined for clarity, correctness, or pedagogical effectiveness. However, once a question is shown to a user, its content must remain immutable for that session.

Model

- A logical `question` represents a conceptual assessment item;
- One or more `question_versions` represent concrete realizations;
- Sessions always reference a specific `question_version`.

This design ensures historical accuracy and analytical integrity.

11.4 Answer choices

Each question version contains multiple answer choices.

Design rules

- Exactly one choice is marked as correct;
- Incorrect choices must be realistic distractors;
- Choices are ordered and labeled consistently;
- Choices are never evaluated independently of explanations.

Answer choices derive their educational value from the explanations associated with them.

11.5 Explanations per choice

Explanations are defined at the level of individual answer choices.

Rationale Explaining only the correct answer is insufficient for medical education. Learners must understand why each alternative is correct or incorrect.

Model For each answer choice:

- A dedicated explanation text is provided;
- The explanation is immutable for a given question version;
- The explanation may reference evidence or clinical reasoning.

This structure enables granular review and precise feedback.

11.6 Educational explanation blocks

In addition to per-choice explanations, question versions may include structured educational explanation blocks.

11.6.1 Educational Objective

A concise statement describing the primary learning goal of the question.

This block answers: *"What was this question designed to test?"*

11.6.2 Key Concept / Bottom Line

A distilled summary of the most important principle the learner should retain after review.

11.6.3 Exam Tip (optional)

Short, exam-oriented guidance highlighting common traps, heuristics, or high-yield reminders.

Educational blocks are optional but recommended for high-impact questions.

11.7 Multiple explanation layers

The model supports multiple explanation layers for different learning needs:

- Concise explanations for rapid review;

- In-depth explanations for deeper study;
- Optional enrichment content linked externally.

Layering improves retention without increasing cognitive overload.

11.8 Bibliographic and external references

Each question version or explanation block may be linked to references.

Reference types

- Standard USMLE textbooks;
- Peer-reviewed articles;
- Clinical guidelines;
- Authoritative online resources;
- External educational media (e.g., podcasts, lectures).

Constraints

- References are linked, not ingested or hosted;
- External resources are optional and supplementary;
- References are shown only during review.

These constraints preserve ownership and legal clarity.

11.9 Relationship to attempts and sessions

The content model integrates with the session model as follows:

- `session_items` reference a specific `question_version`;
- `attempts` store only the selected choice and result;
- Explanations and educational blocks are resolved dynamically during review.

This separation keeps attempts lightweight and content evolvable.

11.10 Content lifecycle

The lifecycle of educational content includes:

1. Authoring of a new question version;
2. Editorial validation and approval;
3. Activation for session generation;
4. Presentation to learners;
5. Long-term archival for historical sessions.

Once a question version is used in a submitted session, it must never be modified.

11.11 Future extensions

The content model is designed to support future enhancements, including:

- Topic and system tagging;
- Difficulty calibration based on empirical data;
- Media-rich explanations (figures, diagrams, audio);
- Research-informed instructional design.

All extensions must preserve backward compatibility.

11.12 Relationship to other chapters

This chapter provides the conceptual foundation for:

- Review behavior (chapter 7);
- Data integrity guarantees (chapter 6);
- Educational rationale (chapter 10);
- External reference constraints (Appendix G).

Any content model change must be reflected here before implementation.

11.13 Conclusion

By modeling educational explanations as structured, layered, and immutable entities, the platform aligns technical rigor with learning effectiveness.

This chapter ensures that content remains evolvable, auditable, and pedagogically sound as the system grows.

Chapter 12

Analytics and Learning Metrics Model

This chapter defines the analytical model used to measure learning outcomes, user performance, and system effectiveness.

It focuses on how raw interaction data is transformed into meaningful, explainable metrics that support feedback, personalization, and long-term learning evaluation.

This chapter builds upon the data structures defined in chapter 6, the content concepts described in chapter 11, the review experience defined in chapter 7, and the analytical rationale presented in chapter 10.

12.1 Analytical objectives

The analytics layer is designed to answer three fundamental questions:

1. How is the learner performing?
2. How is the learner evolving over time?
3. How effective is the content in promoting learning?

Metrics are collected to support pedagogical insight and system improvement, not surveillance or behavioral manipulation.

12.2 Sources of analytical data

All learning metrics are derived from operational interaction data.

Primary sources include:

- `attempts`: correctness, timing, confidence (when available);

- `session_items`: ordering, exposure, and navigation;
- `sessions`: context, mode, lifecycle, and submission state;
- `question_versions`: content identity and versioning.

Additional non-evaluative signals may be derived from review interactions. No analytical data is manually entered.

12.3 Granularity levels

Analytics are computed at multiple levels of granularity:

- **Attempt-level**: single-question interaction metrics;
- **Session-level**: aggregated performance per session;
- **Content-level**: question, choice, and topic performance;
- **User-level**: longitudinal learning trends.

This structure supports both micro-level feedback and macro-level insight.

12.4 Core learning metrics

12.4.1 Accuracy

Accuracy is defined as the proportion of correct attempts over total attempts.

Notes

- Computed only from submitted sessions;
- Skipped questions may be excluded where pedagogically appropriate;
- Accuracy is always contextualized by difficulty and exposure.

12.4.2 Time-on-task

Time-on-task measures the duration spent on each question.

Purpose

- Identify rushed or overly slow reasoning;
- Detect hesitation or cognitive overload;
- Support time-management feedback.

12.4.3 Confidence vs. correctness

When confidence capture is enabled, confidence is compared against correctness.

This enables identification of:

- Overconfidence (high confidence, incorrect);
- Underconfidence (low confidence, correct);
- Calibration quality over time.

12.5 Review interaction signals

The review experience generates additional non-evaluative signals.

Examples include:

- Time spent reviewing a question;
- Expansion of explanation blocks;
- Use of font-size or accessibility controls;
- Question flagging or bookmarking;
- Lightweight engagement signals (e.g., likes).

These signals are used to infer engagement and learning effort, never correctness.

12.6 Session-level analytics

At the session level, analytics include:

- Overall score and accuracy;
- Accuracy distribution across questions;
- Average and median time per question;
- Completion and navigation patterns.

Session analytics are immutable once the session is submitted.

12.7 Longitudinal learning analysis

Learning is modeled as a time series rather than isolated outcomes.

Tracked dimensions

- Accuracy trends over time;
- Repeated exposure to similar concepts;
- Reduction in time-on-task;
- Improvement in confidence calibration;
- Stability of performance under time pressure.

Longitudinal analysis enables detection of durable learning rather than short-term gains.

12.8 Content effectiveness metrics

Content analytics focus on the quality and pedagogical impact of questions.

Metrics include:

- Difficulty index (percentage correct);
- Discrimination potential (variance of outcomes);
- Average time-to-answer;
- Common distractor selection patterns;
- Review engagement indicators per question.

These metrics inform editorial review and content refinement.

12.9 Constraints and safeguards

The analytics model enforces strict safeguards:

- No metric alters historical attempt or session data;
- Analytics never mutate operational tables;
- Derived metrics are deterministically reproducible;
- Only submitted sessions are included in reporting;
- External resources consumption does not affect scoring.

These constraints preserve trust, auditability, and interpretability.

12.10 Operationalization

Analytics may be implemented using:

- SQL aggregation queries;
- Materialized or cached views;
- Periodic batch jobs;
- On-demand API computation.

Implementation choices must balance performance, freshness, and cost.

12.11 Future extensions

The analytics model is designed to evolve toward:

- Topic- and system-level mastery estimation;
- Adaptive question selection;
- Personalized study recommendations;
- Predictive readiness and risk indicators.

All future analytics must remain explainable and ethically grounded.

12.12 Relationship to other chapters

This chapter operationalizes:

- Learning interactions defined in chapter 5;
- Data integrity guarantees in chapter 6;
- Educational structure in chapter 11;
- Review behavior in chapter 7;
- Analytical rationale in chapter 10.

Any analytical extension must be documented here prior to implementation.

12.13 Conclusion

By treating analytics as a first-class, explainable model, the platform ensures that learning outcomes are measurable, interpretable, and actionable.

This chapter completes the transformation from raw interaction data into educational insight while preserving user trust and system integrity.

Chapter 13

Editorial Workflow and Governance

This chapter defines the editorial workflow and governance model for educational content within the USMLE platform.

Its goal is to ensure medical accuracy, pedagogical quality, traceability of changes, and long-term consistency across questions, explanations, educational blocks, and references.

This chapter establishes organizational and policy-level rules rather than technical implementation details.

13.1 Editorial objectives

The editorial governance model is designed to achieve the following objectives:

- Ensure medical accuracy and educational effectiveness;
- Preserve learner trust through content stability and transparency;
- Provide full traceability for all substantive content changes;
- Support collaborative, multi-role content development;
- Align content evolution with analytical and learning insights;
- Govern the use of external educational references responsibly.

Editorial governance treats educational content as a regulated, versioned asset rather than static text.

13.2 Roles and responsibilities

The editorial workflow distinguishes clearly defined roles.

13.2.1 Content author

Responsible for:

- Drafting question stems and answer choices;
- Writing explanations for each answer choice;
- Defining educational blocks (e.g., Educational Objective, Key Concept);
- Proposing bibliographic or external references.

13.2.2 Medical reviewer

Responsible for:

- Validating clinical and scientific accuracy;
- Verifying alignment with current medical guidelines;
- Identifying unsafe, outdated, or misleading content.

13.2.3 Editorial reviewer

Responsible for:

- Ensuring pedagogical clarity and coherence;
- Reviewing explanation structure and depth;
- Enforcing formatting, tone, and consistency standards;
- Assessing cognitive load and clarity in review presentation.

13.2.4 Platform administrator

Responsible for:

- Approving publication and activation of content versions;
- Managing content lifecycle states (draft, active, archived);
- Enforcing governance policies and exception handling.

13.3 Editorial lifecycle

Educational content follows a controlled and auditable lifecycle:

1. Draft creation by the content author;
2. Medical review and correction;
3. Editorial review and pedagogical refinement;
4. Approval and version activation;
5. Exposure to learners via sessions;
6. Archival after use in submitted sessions.

Once a question version is used in a submitted session, it becomes immutable and may never be altered or replaced retroactively.

13.4 Versioning and change control

All substantive content changes require creation of a new version.

Examples of version-triggering changes

- Modification of question stem or clinical scenario;
- Change in the correct answer;
- Alteration of explanation logic or reasoning;
- Addition, removal, or reinterpretation of educational blocks;
- Update or replacement of referenced guidelines or sources.

Minor editorial corrections (e.g., typographical fixes) may follow simplified processes but must remain fully auditable.

Version identifiers are never reused or reassigned.

13.5 Educational blocks governance

Structured educational blocks (e.g., Educational Objective, Key Concept, Exam Tip) are governed as first-class content elements.

Governance rules

- Educational blocks must be consistent with the question's intent;
- Blocks must not introduce contradictions or new unstated assumptions;
- Changes to blocks follow the same versioning rules as explanations;
- Blocks may evolve in future versions without affecting historical sessions.

This ensures pedagogical clarity without compromising historical accuracy.

13.6 Traceability and audit

For each content version, the system must be able to determine:

- Authorship and contributor roles;
- Review and approval history;
- Publication and activation timestamps;
- Associated references and external resources.

This traceability supports quality assurance, editorial accountability, and external audit requirements.

13.7 Integration with analytics

Editorial decisions are informed by learning analytics, not driven by them.

Examples include:

- Identifying questions with poor discrimination;
- Detecting consistently misleading distractors;
- Observing excessive review time or confusion indicators;
- Recognizing outdated or unclear explanations.

Analytics may recommend review or revision, but content changes always require explicit human approval.

13.8 External references and resources

The editorial workflow governs the use of external educational references.

Principles

- External resources are supplementary, not authoritative;
- Content is linked, not ingested or mirrored;
- References must be reputable and educationally appropriate;
- External links must not substitute core explanations.

All external references must comply with Appendix G.

13.9 Governance constraints

The following constraints are strictly enforced:

- Historical sessions must never be altered;
- Analytics cannot retroactively change learner outcomes;
- Deactivated questions remain accessible for review;
- Emergency corrections must still preserve versioning rules;
- Convenience must never override governance principles.

Governance rules take precedence over operational shortcuts.

13.10 Exception handling

Exceptional situations (e.g., discovery of a critical medical error) are handled via a controlled process:

- Immediate deactivation of affected content versions;
- Creation and review of a corrected new version;
- Explicit documentation of the incident and resolution.

Previously delivered content remains visible to learners to preserve transparency and trust.

13.11 Relationship to other chapters

This chapter governs how content defined in chapter 11 is created, how analytics in chapter 12 inform decisions, and how integrity guarantees in chapter 6 are preserved.

It also supports the review experience described in chapter 7 and the integration boundaries defined in Appendix G.

Editorial governance provides the organizational backbone of the platform.

13.12 Conclusion

By formalizing editorial workflow and governance, the platform ensures that educational quality, learner trust, and accountability scale alongside technical growth.

This chapter completes the transformation of the platform from a software system into a governed, auditable educational product.

Chapter 14

Security, Privacy, and Compliance

This chapter defines the security, privacy, and compliance principles that govern the USMLE platform.

Its purpose is to ensure that technical design, data handling, and operational processes protect user data, preserve trust, and comply with applicable legal and ethical standards.

This chapter builds upon the structural guarantees defined in chapter 6 and the governance rules described in chapter 13.

14.1 Security objectives

The security model is designed to achieve the following objectives:

- Protect user identity and personal data;
- Prevent unauthorized access to educational records;
- Ensure integrity and immutability of learning history;
- Reduce attack surface and limit blast radius;
- Support auditability, transparency, and incident response.

Security is treated as a system-wide property, not as an afterthought.

14.2 Threat model

The platform considers the following threat categories:

- Unauthorized access to sessions, attempts, or review data;
- Leakage or inference of personally identifiable information (PII);

- Tampering with historical learning or analytics data;
- Abuse of development, debug, or administrative endpoints;
- Re-identification risks via aggregated analytics.

System design decisions aim to mitigate these threats by construction rather than by reactive controls.

14.3 Authentication and authorization

14.3.1 Authentication

Authentication is handled via:

- Secure session-based authentication in production environments;
- Explicit header-based authentication strictly limited to development and testing contexts.

Authentication mechanisms are defined in detail in chapter 5.

14.3.2 Authorization

Authorization rules include:

- Users may access only their own sessions, attempts, and review data;
- Editorial and administrative actions are role-restricted;
- Development and diagnostic endpoints are disabled in production;
- External integrations operate under least-privilege access.

Authorization is enforced consistently at the API boundary.

14.4 Data minimization and separation

The platform follows strict data minimization principles:

- Only data required for learning and analytics is stored;
- Authentication and identity data are not duplicated in learning tables;
- Analytical data is derived, not manually entered;
- Review interaction signals are non-evaluative and optional.

User identity is represented by deterministic internal identifiers, as defined in chapter 6.

14.5 Integrity and immutability guarantees

Key integrity guarantees include:

- Database-level constraints preventing duplicate attempts;
- Immutability of submitted sessions and historical results;
- Versioned content and explanations to preserve accuracy over time;
- Append-only evolution of analytical interpretations.

These guarantees prevent both accidental and malicious data corruption.

14.6 Privacy considerations

Learning data is treated as sensitive personal information.

Privacy principles include:

- No sharing of individual performance data without explicit consent;
- Aggregation and anonymization for analytics and reporting;
- Clear separation between operational data and analytical views;
- External educational resources do not receive learner data.

Analytics and engagement signals are designed to support learning, not surveillance or behavioral profiling.

14.7 External resources and integrations

The platform may link to external educational resources during review.

Security and privacy constraints

- External resources are linked, not embedded or hosted;
- No learner identity or performance data is shared externally;
- External links open outside the authenticated system context;
- Resource usage does not affect scoring or analytics outcomes.

All integrations comply with Appendix G.

14.8 Compliance and regulatory alignment

Although the platform is not a clinical system, it aligns with best practices from:

- General data protection regulations (e.g., GDPR principles);
- Educational data protection and privacy standards;
- Industry security and risk management best practices.

Compliance is achieved through transparent design, documentation, and auditability rather than opaque enforcement mechanisms.

14.9 Logging, monitoring, and audit

Operational logging supports:

- Detection of anomalous access patterns;
- Investigation of security or integrity incidents;
- Verification of editorial and administrative actions;
- Monitoring of external integration health.

Logs must never expose sensitive content, learner answers, or personal data.

14.10 Environment separation

Strict separation is enforced between:

- Development environments;
- Testing and staging environments;
- Production environments.

Test data, debug endpoints, and non-production credentials must never be promoted to production.

14.11 Incident response

In the event of a security, privacy, or data integrity incident, the response process includes:

1. Incident identification and containment;
2. Impact assessment and scope determination;
3. Remediation and corrective actions;
4. Documentation, notification, and governance review.

Incident handling prioritizes transparency, proportionality, and user trust.

14.12 Relationship to other chapters

This chapter enforces and protects:

- API boundaries defined in chapter 5;
- Data integrity guarantees in chapter 6;
- Editorial controls in chapter 13;
- Analytics constraints in chapter 12;
- Integration boundaries in Appendix G.

Security and privacy considerations apply across all system layers.

14.13 Conclusion

By embedding security, privacy, and compliance principles directly into system design, the platform ensures that educational value is delivered without compromising learner trust, data integrity, or ethical responsibility.

This chapter completes the governance, security, and risk framework of the platform.

Chapter 15

Deployment, Operations, and Reliability

This chapter defines how the system is deployed, operated, monitored, and maintained in production environments.

Its goal is to ensure that the platform remains reliable, observable, and recoverable as it evolves in functionality and scale.

This chapter complements the security guarantees described in chapter 14 and the data integrity principles defined in chapter 6.

15.1 Deployment objectives

The deployment and operations model is designed to achieve the following objectives:

- Fast, repeatable, and auditable deployments;
- Minimal downtime with safe and rapid rollbacks;
- Strict separation of environments and credentials;
- Predictable behavior under load and peak usage;
- Rapid detection, diagnosis, and recovery from failures.

Operational simplicity is treated as a first-class reliability feature.

15.2 Environment model

The platform operates across multiple isolated environments:

- **Development**: local testing, experimentation, and debugging;

- **Staging**: pre-production validation and integration testing;
- **Production**: live, user-facing environment.

Each environment enforces:

- Separate databases and storage;
- Separate credentials, secrets, and API keys;
- Explicit configuration boundaries.

Cross-environment data sharing is strictly prohibited.

15.3 Deployment strategy

Deployments follow an automated and controlled pipeline.

15.3.1 Continuous deployment

The system supports continuous deployment with the following properties:

- Builds triggered exclusively by version-controlled changes;
- Automated validation, linting, and build checks;
- Deterministic artifact generation;
- Environment-specific configuration injection at deploy time.

Deployment pipelines are observable and auditable.

15.3.2 Backward compatibility

All deployments must preserve:

- Compatibility with existing API contracts;
- Integrity and immutability of submitted sessions;
- Interpretability of historical analytics and review data.

Breaking changes require explicit versioning and migration planning.

15.4 Database operations

The database is the authoritative system of record.

Operational rules

- Schema changes are applied via controlled migrations;
- Migrations must be backward-compatible whenever feasible;
- Destructive operations are forbidden in production;
- Referential integrity and constraints are never disabled;
- Analytics-related schema changes must preserve reproducibility.

All database operations respect the constraints defined in chapter 6.

15.5 Observability

System observability is achieved through:

- Structured and contextual application logs;
- Request-level tracing and correlation identifiers;
- Key performance indicators (latency, error rate, throughput);
- Health check and readiness endpoints.

Observability data is used for diagnosis, reliability, and capacity planning, never for learner surveillance.

15.6 Reliability principles

Reliability is treated as an architectural constraint.

Key principles include:

- Stateless application instances;
- Idempotent write operations for critical endpoints;
- Explicit handling of retries and duplicate requests;
- Clear, documented failure modes;
- Graceful degradation of non-critical features.

Core invariants are enforced at the database level whenever possible.

15.7 Failure handling and recovery

The system anticipates and plans for failure scenarios.

15.7.1 Application failures

- Automatic restart and replacement of failed instances;
- Clear and consistent error responses to clients;
- Transactional guarantees preventing partial writes.

15.7.2 Database failures

- Automated, regular backups with retention policies;
- Point-in-time recovery capabilities;
- Tested and documented restoration procedures.

Recovery procedures prioritize data integrity and correctness over availability.

15.8 Operational safeguards

Operational safeguards include:

- Rate limiting and throttling of critical endpoints;
- Strict access control for administrative and editorial operations;
- Feature flags for experimental or staged functionality;
- Explicit kill-switches for unsafe or degraded components;
- Isolation of external integration failures.

These safeguards reduce the blast radius of both faults and misconfigurations.

15.9 Change management

Operational and architectural changes follow a controlled lifecycle:

1. Proposal and risk assessment;
2. Validation in development and staging environments;

3. Gradual, monitored rollout to production;
4. Post-deployment verification and review.

All significant incidents, rollbacks, and corrective actions are documented and reviewed.

15.10 Relationship to other chapters

This chapter operationalizes:

- Security and privacy guarantees from chapter 14;
- Data integrity rules from chapter 6;
- Analytics reliability constraints from chapter 12;
- Editorial governance controls from chapter 13;
- Integration resilience defined in Appendix G.

Deployment and operations are the final enforcement layer of all system decisions.

15.11 Conclusion

By formalizing deployment, operations, and reliability practices, the platform ensures that technical correctness, educational integrity, and learner trust persist beyond individual releases.

This chapter completes the system lifecycle from architectural intent to sustained production operation.

Chapter 16

Roadmap, Technical Debt, and Future Research

This chapter documents the forward-looking evolution of the platform. It explicitly distinguishes between planned features, acknowledged technical debt, and open research questions.

The objective is to ensure intentional growth rather than accidental complexity.

This chapter should be treated as a living strategic document.

16.1 Purpose of the roadmap

The roadmap exists to:

- Align technical development with educational goals;
- Make pedagogical intent explicit in product decisions;
- Prevent uncontrolled accumulation of technical debt;
- Guide prioritization as the system scales.

All roadmap items must respect the architectural constraints defined in chapter 6 and the governance rules in chapter 13.

16.2 Short-term roadmap

Short-term initiatives focus on strengthening the learning experience on top of the existing MVP without altering core architecture.

16.2.1 Advanced review experience

- Deployment of an advanced review layout with semantic color-coding (correct vs. incorrect);
- Structured educational blocks (Educational Objective, Key Concept, Exam Tip);
- Clear visual association between answer choices and explanations;
- Support for long-form reading and mobile-first ergonomics.

16.2.2 Reader and interaction controls

- Font size adjustment during review;
- Question flagging for later review;
- Lightweight engagement signals (e.g., like or bookmark);
- Persistent session timer visibility in timed modes.

16.2.3 Operational hardening

- Rate limiting for critical endpoints;
- Improved observability and alerting;
- Safer deployment and rollback checks.

16.3 Mid-term roadmap

Mid-term initiatives aim to deepen personalization and educational insight while maintaining deterministic behavior.

16.3.1 Educational enrichment

- Multiple explanation layers (concise vs. in-depth);
- Optional per-question commenting and annotation mechanisms;
- Expanded reference navigation within review.

16.3.2 Analytics expansion

- Topic- and system-level mastery tracking;
- Review-time and engagement signal analytics;
- Confidence calibration and trend visualization.

16.3.3 Content management

- Question tagging and taxonomy support;
- Difficulty calibration based on empirical performance data;
- Editorial workflow tooling for content iteration.

16.4 Long-term roadmap

Long-term initiatives explore higher-order learning and ecosystem integration.

16.4.1 Personalized learning paths

- Adaptive study recommendations based on performance and review behavior;
- Identification of persistent weak concepts across sessions;
- Predictive readiness indicators.

16.4.2 External educational resource integration

- Integration of external educational resources as review-linked companions;
- Support for text and audio references (e.g., podcast-style resources);
- Clear separation between internal content and external materials.

All external resources are linked, not hosted, in accordance with Appendix G.

16.4.3 Research-driven assessment

- Research-informed assessment methodologies;
- Explainable analytics for learners;
- Ethical evaluation of personalization boundaries.

Long-term features may influence architectural assumptions and require formal design review.

16.5 Technical debt

Technical debt is explicitly acknowledged rather than hidden.

16.5.1 Known areas of debt

- Limited schema support for advanced analytics in early phases;
- Absence of dedicated editorial user interfaces;
- Simplified authorization and role management;
- Manual governance enforcement in initial workflows.

16.5.2 Debt management principles

- Debt must be documented when incurred;
- Debt must be reviewed at regular intervals;
- Debt must be addressed before architectural constraints are violated.

No technical debt may compromise data integrity or historical accuracy.

16.6 Research directions

Some questions extend beyond straightforward engineering and require research-oriented exploration.

16.6.1 Learning science

- Impact of explanation depth on long-term retention;
- Relationship between engagement signals and mastery;
- Optimal balance between feedback immediacy and reflection.

16.6.2 Assessment theory

- Question discrimination and difficulty metrics;
- Bias detection in assessment items;
- Adaptive difficulty modeling.

16.6.3 System design research

- Transparency vs. cognitive overload trade-offs;
- Explainability of analytics to learners;
- Ethical limits of personalization.

Research outcomes may inform future roadmap revisions.

16.7 Decision review cadence

Strategic decisions are revisited periodically:

- Roadmap items are reviewed quarterly;
- Technical debt is reassessed before major releases;
- Research insights are evaluated for practical adoption.

This cadence ensures deliberate, evidence-informed evolution.

16.8 Relationship to other chapters

This chapter synthesizes insights from:

- System analysis (chapter 10);
- Content and review models (chapter 11, chapter 7);
- Governance and operations (chapter 13, chapter 15);
- External integration constraints (Appendix G).

It provides the strategic context for future development.

16.9 Conclusion

By explicitly documenting the roadmap, technical debt, and research directions, the platform avoids reactive development and preserves architectural clarity.

This chapter completes the handbook as a living document connecting past decisions, present capabilities, and future intent.

Appendix A

Glossary

This glossary defines key terms used throughout the handbook. All terms are used with the meanings specified here unless explicitly stated otherwise.

The glossary serves as a shared vocabulary for engineering, editorial, and analytical stakeholders.

A

API contract Formal definition of endpoints, request/response formats, and behavioral rules governing interactions between clients and the system. See chapter 5.

Attempt A single, definitive user interaction with a session item, representing the final selected answer (or skip). Attempts are immutable once recorded. See chapter 6.

C

Confidence A self-reported measure indicating how confident a learner is in their answer. Used for calibration and learning analytics. See chapter 12.

Content version A specific immutable realization of a question, its answer choices, and explanations. Once used in a submitted session, it must never change. See chapter 11.

D

Data model The canonical definition of database tables, relationships, and constraints. Serves as the system of record for integrity guarantees. See chapter 6.

Deterministic user identifier A stable UUID derived from user identity, ensuring consistent linkage of data without storing raw credentials. See chapter 5.

E

Editorial governance The set of rules, roles, and processes controlling how educational content is created, reviewed, approved, and evolved. See chapter 13.

Explanation A pedagogical justification explaining why a specific answer choice is correct or incorrect. Defined per answer choice. See chapter 11.

I

Idempotency The property by which repeated execution of the same operation produces the same result without unintended side effects. Enforced primarily at the database level. See chapter 6.

Integrity constraint A database-enforced rule ensuring validity and consistency of stored data, such as uniqueness or referential integrity. See chapter 6.

L

Learning analytics Derived metrics that transform raw interaction data into insights about learner performance, progress, and content effectiveness. See chapter 12.

R

Review The post-submission phase in which learners can inspect questions, attempts, correctness, explanations, and references. See chapter 5.

Roadmap A forward-looking plan describing intended system evolution, known technical debt, and future research directions. See chapter 16.

S

Session A bounded study context representing a coherent set of questions presented to a learner under defined conditions. See chapter 5.

Session item An ordered element within a session representing the presentation of a specific question version. See chapter 6.

Submitted session A session whose lifecycle is complete and whose data is immutable. Only submitted sessions are used for analytics. See chapter 5.

T

Technical debt A conscious trade-off where a simpler or incomplete solution is chosen with the intent of future improvement. Tracked explicitly in the roadmap. See chapter 16.

U

Uniqueness rule A constraint enforcing that a specific data relationship can occur only once, such as one attempt per session item. See section 6.4.1.

V

Versioning The practice of creating immutable snapshots of content or schema elements to preserve historical accuracy while allowing evolution. See chapter 11.

W

Workflow A structured sequence of actions and approvals governing system behavior or content lifecycle. See chapter 13.

Appendix B

Architectural Decision Records (ADR)

This appendix documents significant architectural decisions made during the design and evolution of the platform.

Each record captures the context, decision, rationale, and consequences, in order to preserve institutional knowledge and support future change.

ADR-001 — Session-based learning model

Status

Accepted

Context

The platform needed a way to group learning interactions into coherent units that reflect real exam conditions and support review, analytics, and reproducibility.

Decision

Introduce an explicit `sessions` entity as the primary learning context.

Rationale

- Sessions model real exam blocks;
- They provide a natural boundary for analytics;
- They simplify lifecycle management (start, submit, review).

Consequences

- All attempts are contextualized by a session;
- Sessions become immutable after submission;
- API design follows session lifecycle semantics.

Related chapters: chapter 5, chapter 6

ADR-002 — Explicit session items

Status

Accepted

Context

Question ordering and exposure needed to be stable and auditable.

Decision

Materialize session structure via a dedicated `session_items` table.

Rationale

- Preserves ordering independent of attempts;
- Supports accurate review;
- Decouples question selection from user interaction.

Consequences

- Session generation becomes idempotent;
- Review logic becomes simpler and more reliable.

Related chapters: chapter 6, chapter 10

ADR-003 — Database-enforced idempotency

Status

Accepted

Context

The attempt endpoint must tolerate retries and concurrent requests without creating inconsistent data.

Decision

Enforce idempotency using a database-level `UNIQUE(session_item_id)` constraint.

Rationale

- Databases are the strongest consistency boundary;
- Application-level guards are insufficient under concurrency;
- Guarantees correctness even during failures.

Consequences

- Simplified API logic;
- Natural retry safety;
- Clear invariants for analytics.

Related chapters: chapter 6, chapter 5

ADR-004 — Versioned educational content

Status

Accepted

Context

Questions and explanations evolve over time, but historical sessions must remain accurate.

Decision

Treat questions as versioned artifacts and reference specific versions in sessions.

Rationale

- Preserves historical accuracy;
- Allows continuous improvement of content;
- Enables reliable analytics.

Consequences

- Content becomes immutable once used;
- Editorial workflow must create new versions for changes.

Related chapters: chapter 11, chapter 13

ADR-005 — Analytics derived from operational data

Status

Accepted

Context

The system needed analytics without compromising data integrity or introducing manual data paths.

Decision

Derive all analytics exclusively from operational tables.

Rationale

- Prevents data divergence;
- Ensures reproducibility;
- Simplifies auditing.

Consequences

- Analytics remain read-only;
- Historical data can be recomputed deterministically.

Related chapters: chapter 12, chapter 6

ADR-006 — Documentation as a first-class artifact

Status

Accepted

Context

The system is complex and expected to evolve with multiple contributors.

Decision

Treat documentation as part of the system, not as an afterthought.

Rationale

- Reduces onboarding time;
- Preserves rationale behind decisions;
- Prevents architectural drift.

Consequences

- Changes require documentation updates;
- Chapters serve as authoritative references.

Related chapters: All chapters in this handbook

ADR template for future decisions

```
ADR-XXX - <Title>

Status: Proposed | Accepted | Deprecated | Superseded

Context:
  <What problem are we solving?>

Decision:
  <What was decided?>

Rationale:
```

```
  <Why was this decision made?>

Consequences:
  <What are the trade-offs and impacts?>

Related chapters:
  <References>
```

Appendix C

Error Codes and API Failure Semantics

This appendix defines the error model used by the API. It standardizes error responses, status codes, and failure semantics to ensure predictable client behavior and ease of debugging.

Errors are treated as first-class API responses, not as exceptional edge cases.

C.1 Error design principles

The API follows these error-handling principles:

- Errors must be explicit and machine-readable;
- HTTP status codes convey the error class;
- Error messages are stable and not free-form;
- Clients must be able to react programmatically;
- Internal details are never leaked.

C.2 Standard error response format

All error responses follow a common JSON structure:

```
{
  "error": {
    "code": "STRING_CODE",
    "message": "Human-readable description"
  }
}
```

Optional fields (when applicable):

```
{
  "error": {
    "code": "STRING_CODE",
    "message": "Description",
    "details": { ... }
  }
}
```

Clients must rely on `code`, not on `message`.

C.3 HTTP status code semantics

Table C.1: HTTP status code usage

Status	Meaning
400	Invalid request or malformed input
401	Authentication required or failed
403	Authenticated but not authorized
404	Resource not found
409	Conflict with current system state
422	Semantically invalid request
429	Rate limit exceeded
500	Internal server error
503	Temporary service unavailability

C.4 Canonical error codes

C.4.1 Authentication and authorization

Table C.2: Authentication errors

Code	HTTP	Description
AUTH_REQUIRED	401	Authentication is required
AUTH_INVALID	401	Invalid or expired credentials
AUTH_FORBIDDEN	403	Insufficient permissions

Table C.3: Session-related errors

Code	HTTP	Description
SESSION_NOT_FOUND	404	Session does not exist
SESSION_ALREADY_SUBMITTED	409	Session is already submitted
SESSION_NOT_SUBMITTED	409	Session must be submitted for this operation
SESSION_IMMUTABLE	409	Session can no longer be modified

C.4.2 Session lifecycle errors

C.4.3 Session item and attempt errors

Table C.4: Attempt-related errors

Code	HTTP	Description
SESSION_ITEM_NOT_FOUND	404	Session item does not exist
ATTEMPT_ALREADY_EXISTS	409	Attempt already recorded
ATTEMPT_INVALID_STATE	409	Attempt not allowed in current state
ATTEMPT_PAYLOAD_INVALID	422	Invalid attempt data

C.4.4 Validation errors

Table C.5: Validation errors

Code	HTTP	Description
VALIDATION_FAILED	422	Request failed schema validation
INVALID_PARAMETER	400	Invalid query or path parameter
MISSING_FIELD	400	Required field missing

C.5 Idempotency and conflict semantics

Certain conflicts are expected and non-fatal.

Example: duplicate attempt submission If a client retries an attempt submission, the API may return:

```
HTTP 409
{
  "error": {
    "code": "ATTEMPT_ALREADY_EXISTS",
```

```
    "message": "Attempt already recorded for this session item"
  }
}
```

Clients should treat this response as success-equivalent.

This behavior aligns with the database constraint defined in section 6.4.1.

C.6 Security-related errors

Security-related errors must:

- Avoid revealing sensitive information;
- Use generic messages when appropriate;
- Be logged internally for audit purposes.

Example:

```
HTTP 403
{
  "error": {
    "code": "AUTH_FORBIDDEN",
    "message": "Access denied"
  }
}
```

C.7 Internal server errors

Unexpected failures return:

```
HTTP 500
{
  "error": {
    "code": "INTERNAL_ERROR",
    "message": "An unexpected error occurred"
  }
}
```

Internal details are logged but never returned to clients.

C.8 Client behavior guidelines

Clients interacting with the API should:

- Branch logic based on error `code`;
- Retry safely on idempotent conflicts;
- Avoid retrying non-idempotent validation errors;
- Display user-friendly messages mapped from error codes.

Client-side behavior must align with the semantics defined here.

C.9 Relationship to other chapters

This appendix formalizes failure behavior for:

- API endpoints defined in chapter 5;
- Data integrity constraints in chapter 6;
- Security policies in chapter 14;
- Operational safeguards in chapter 15.

C.10 Conclusion

By standardizing error codes and failure semantics, the API becomes easier to consume, debug, and evolve.

This appendix ensures that failures are predictable, explainable, and safe.

Appendix D

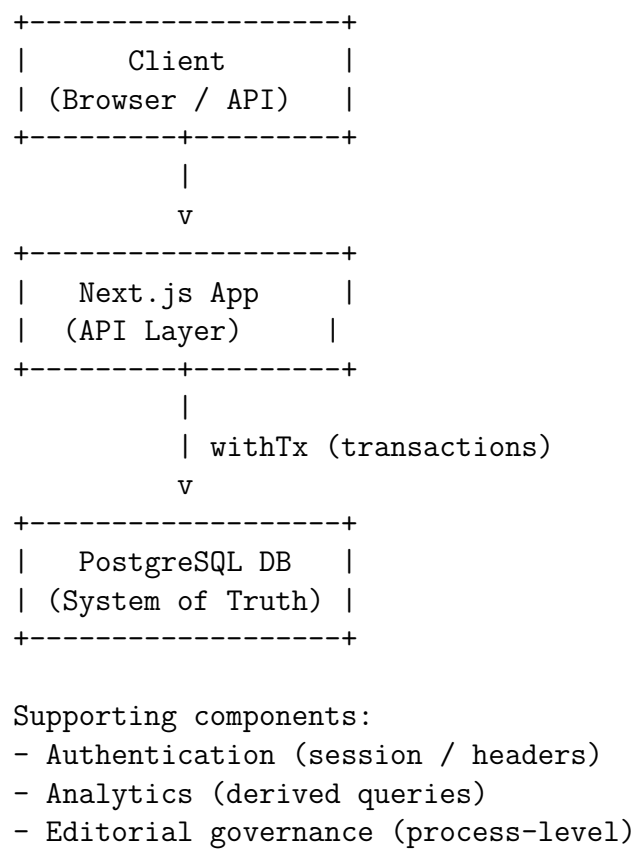
System Diagrams

This appendix provides visual representations of the system architecture, data relationships, and core functional flows.

The diagrams are conceptual and intended to support understanding, onboarding, and reasoning about the system.

D.1 System architecture overview

Figure D.1: High-level system architecture

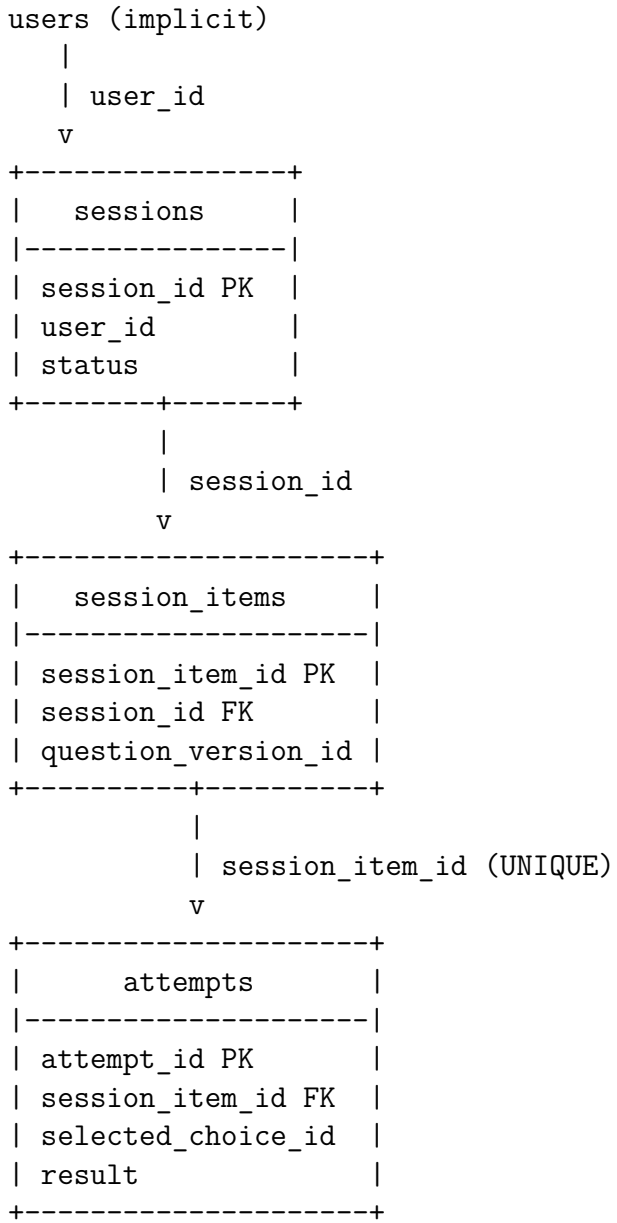


This architecture emphasizes:

- Stateless application logic;
- Database-centered consistency;
- Clear separation of concerns.

D.2 Data model relationships

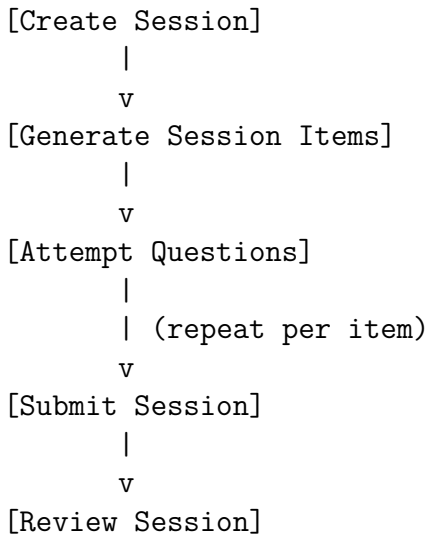
Figure D.2: Logical data model relationships



This diagram reflects the canonical structure defined in chapter 6.

D.3 Session lifecycle flow

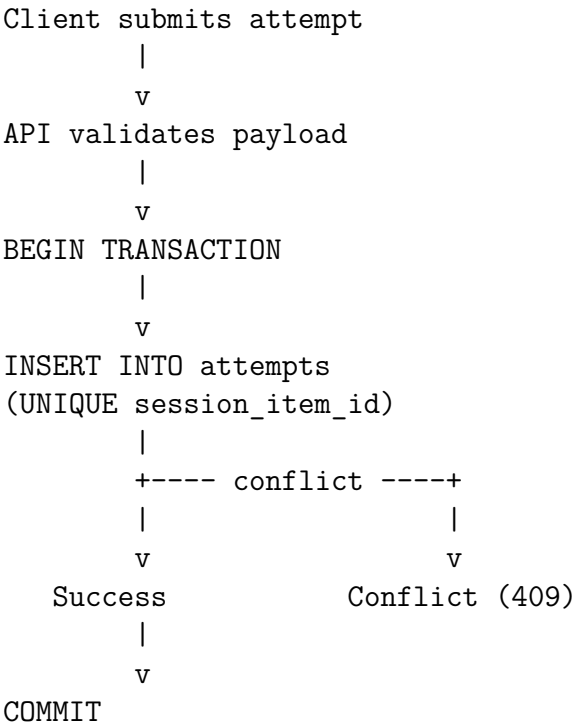
Figure D.3: Session lifecycle flow



Once a session is submitted, it becomes immutable.

D.4 Attempt flow and idempotency

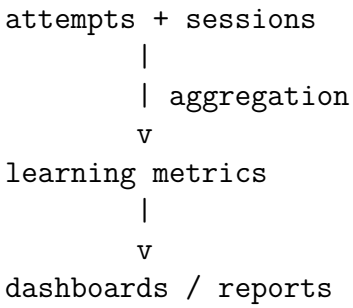
Figure D.4: Attempt recording and idempotency



This flow enforces the invariant described in section 6.4.1.

D.5 Analytics derivation flow

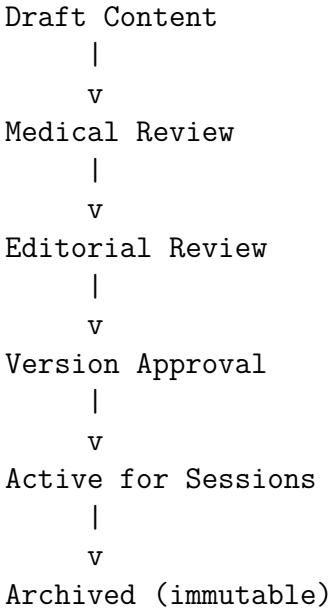
Figure D.5: Analytics derivation



Analytics never mutate operational data.

D.6 Editorial content flow

Figure D.6: Editorial content lifecycle



This flow aligns with governance rules in chapter 13.

D.7 Relationship to other chapters

These diagrams visually support:

- API behavior (chapter 5);
- Data integrity rules (chapter 6);

- Analytical models (chapter 12);
- Editorial governance (chapter 13);
- Operational reliability (chapter 15).

They serve as a conceptual map of the system.

D.8 Conclusion

By providing visual representations of architecture, data, and flows, this appendix complements the textual documentation and enhances system understandability.

These diagrams should evolve alongside the system but always reflect canonical design decisions.

Appendix E

Operational Checklists

This appendix provides practical, actionable checklists for operating the platform safely and reliably.

These checklists are intended to be used during routine operations, deployments, incident response, and governance reviews.

E.1 Pre-deployment checklist

Use this checklist before deploying any change to production.

- All changes are version-controlled and reviewed;
- Database migrations are backward-compatible;
- API changes preserve existing contracts (chapter 5);
- No destructive schema operations are included;
- Feature flags are configured correctly;
- Rollback plan is documented.

Deployment must not proceed unless all items are satisfied.

E.2 Deployment checklist

Perform during the deployment window.

- Correct environment selected (production vs. staging);
- Secrets and environment variables verified;
- Build artifacts validated;

- Health checks passing post-deploy;
- No elevated error rates observed;
- Logs monitored for anomalies.

Any unexpected behavior triggers immediate rollback.

E.3 Post-deployment checklist

Verify system stability after deployment.

- Core API endpoints responding successfully;
- Session creation and submission tested;
- Attempt idempotency validated;
- Analytics queries functioning;
- No integrity constraint violations detected.

Post-deployment validation confirms production readiness.

E.4 Database change checklist

Apply before any schema or migration change.

- Migration reviewed and tested in staging;
- Data integrity constraints preserved (chapter 6);
- No data loss scenarios identified;
- Backups completed successfully;
- Rollback or mitigation strategy defined.

Database changes are the highest-risk operations.

E.5 Incident response checklist

Use during security, availability, or data integrity incidents.

1. Identify and contain the incident;
2. Assess scope and impacted components;
3. Preserve logs and forensic data;
4. Apply corrective actions;
5. Verify system integrity;
6. Document incident and lessons learned.

Transparency and data integrity take precedence over speed.

E.6 Security review checklist

Perform periodically or after major changes.

- Authentication flows reviewed;
- Authorization boundaries verified;
- Development endpoints disabled in production;
- Secrets rotated if necessary;
- Logs reviewed for suspicious activity.

Security posture must be reviewed continuously.

E.7 Editorial governance checklist

Apply when reviewing or publishing content.

- Content reviewed by medical and editorial roles;
- Versioning rules respected;
- References verified and current;
- Historical content remains immutable;
- Analytics signals considered.

Editorial actions must follow governance rules (chapter 13).

E.8 Analytics validation checklist

Apply when introducing or modifying metrics.

- Metrics derived only from submitted sessions;
- No mutation of operational data;
- Aggregations reproducible;
- Results explainable to stakeholders;
- Performance impact assessed.

Analytics must remain trustworthy and interpretable.

E.9 Operational readiness review

Use this checklist before major launches or scale events.

- System load tested;
- Observability dashboards configured;
- Alerting thresholds validated;
- Incident response roles assigned;
- Communication plan prepared.

Readiness reviews prevent avoidable failures.

E.10 Relationship to other chapters

These checklists operationalize principles defined in:

- Deployment and reliability (chapter 15);
- Security and compliance (chapter 14);
- Data integrity (chapter 6);
- API behavior (chapter 5);
- Editorial governance (chapter 13).

They translate documentation into action.

E.11 Conclusion

By providing clear operational checklists, the platform reduces risk, standardizes execution, and supports reliable growth.

This appendix ensures that best practices are consistently applied across environments and over time.

Appendix F

Technical Glossary and Acronyms

This appendix defines technical terms and acronyms used throughout the handbook.

It complements the general glossary (Appendix A) by focusing on engineering, architecture, operations, and analytics terminology.

All acronyms are listed alphabetically.

A

ADR (Architectural Decision Record) A structured document capturing an important architectural decision, including context, rationale, and consequences. See Appendix B.

API (Application Programming Interface) A contract defining how clients interact with the system via HTTP endpoints. See chapter 5.

Atomicity A property of transactions ensuring that operations either complete fully or have no effect. See chapter 6.

B

Backend The server-side components responsible for business logic, data access, and API responses.

Blast radius The extent of impact caused by a failure or incident within the system. Discussed in chapter 15.

C

CI/CD (Continuous Integration / Continuous Deployment) Automated processes that build, test, and deploy code changes. See chapter 15.

Client Any consumer of the API, including browsers, mobile apps, or automated systems.

Conflict (HTTP 409) An HTTP status indicating a request conflicts with the current state of the system. Formalized in Appendix C.

D

DDL (Data Definition Language) SQL commands used to define or modify database schema.

Derived data Data computed from primary sources rather than directly stored. See chapter 12.

E

ERD (Entity Relationship Diagram) A visual representation of entities and their relationships. Conceptually represented in Appendix D.

Environment An isolated deployment context (development, staging, production). See chapter 15.

F

FK (Foreign Key) A database constraint linking a column to a primary key in another table. Defined in chapter 6.

Frontend The client-side application responsible for user interaction.

H

HTTP (Hypertext Transfer Protocol) The protocol used for communication between clients and the API.

Health check An endpoint used to verify system availability. See chapter 5.

I

Idempotency The property that repeated execution of the same operation yields the same result. Enforced by database constraints. See chapter 6.

Immutable A property indicating that data cannot be modified after creation. Applied to submitted sessions and content versions.

L

Latency The time taken to process a request.

Lifecycle The sequence of states an entity passes through. See session lifecycle in chapter 5.

M

Migration A controlled change to database schema or data. Discussed in chapter 15.

MVP (Minimum Viable Product) The initial functional version of the system with core features only.

O

Observability The ability to understand system behavior through logs, metrics, and traces. See chapter 15.

Operational data Primary data generated by system usage (sessions, attempts).

P

PII (Personally Identifiable Information) Any data that can identify an individual. Handled according to chapter 14.

PostgreSQL The relational database used as the system of record.

R

Rollback The act of reverting a system to a previous stable state.

Retry A repeated attempt to perform a failed operation. Safe retries rely on idempotency.

S

Schema The structural definition of database tables and constraints. See chapter 6.

Stateless A property of services that do not retain client state between requests.

T

Transaction A group of database operations executed atomically.

Technical debt A conscious trade-off that prioritizes speed over completeness. Tracked in chapter 16.

U

UUID (Universally Unique Identifier) A globally unique identifier used as primary keys.

V

Versioning The practice of maintaining immutable snapshots to allow evolution without breaking history. See chapter 11.

Z

Zero-downtime deployment A deployment strategy that avoids service interruption. Discussed in chapter 15.

Appendix G

External Integrations

This appendix documents how the platform integrates with external systems and services.

It defines integration principles, supported integration types, security constraints, and operational expectations.

External integrations are treated as extensions of the system boundary and must comply with all core architectural guarantees.

G.1 Integration principles

All external integrations must adhere to the following principles:

- The platform remains the system of record;
- External systems never mutate internal core data;
- Integrations are explicitly authenticated and authorized;
- Failures in external systems must not corrupt internal state;
- Integrations must be observable and auditable.

No integration may bypass API contracts or database constraints.

G.2 Types of external integrations

G.2.1 Authentication providers

External authentication providers may be used to establish user identity.

Constraints

- External providers supply identity only;
- Authorization decisions remain internal;
- Deterministic user identifiers are derived internally.

Authentication behavior is governed by chapter 5 and chapter 14.

G.2.2 Analytics and monitoring services

Third-party analytics or monitoring tools may be integrated for operational insight.

Constraints

- No personally identifiable learning data is exported;
- Metrics are aggregated and anonymized;
- Raw attempt or session data remains internal.

These constraints preserve privacy guarantees.

G.2.3 Content and reference providers

External sources may be referenced to support educational explanations.

Examples

- Medical textbooks;
- Clinical guidelines;
- Peer-reviewed publications.

External content is linked, not ingested, unless explicitly governed by editorial policy.

G.2.4 Payment and subscription systems

If applicable, external billing or subscription services may be integrated.

Constraints

- Financial data is isolated from learning data;
- Subscription status influences access control only;
- No billing system may modify educational records.

G.3 Integration boundaries

All integrations interact with the platform exclusively through:

- Public API endpoints;
- Explicit webhook receivers (if defined);
- Read-only data export mechanisms.

Direct database access by external systems is prohibited.

G.4 Failure and resilience model

External integrations are treated as unreliable by default.

Failure handling

- Timeouts and retries are bounded;
- External failures never block core workflows;
- Partial integration failures degrade gracefully.

All failure semantics must align with Appendix C.

G.5 Security considerations

External integrations must comply with platform security requirements:

- Least-privilege access;
- Credential rotation and revocation;
- Secure transport (TLS);
- Audit logging of integration activity.

Security requirements are governed by chapter 14.

G.6 Data privacy and compliance

Privacy guarantees extend across integration boundaries.

- Learning data is never shared without explicit consent;
- Aggregated exports must prevent re-identification;
- External systems must not store sensitive internal identifiers.

Integrations must respect all compliance constraints.

G.7 Operational management

Operational practices for integrations include:

- Environment-specific configuration;
- Feature flags for enabling/disabling integrations;
- Monitoring of integration health and latency;
- Rapid deactivation mechanisms.

Integrations must not introduce single points of failure.

G.8 Future integration scenarios

The integration model supports future scenarios such as:

- Institutional learning management systems (LMS);
- Research data exports (anonymized);
- Adaptive learning engines;
- External assessment tools.

All future integrations must be evaluated against the principles defined in this appendix.

G.9 Relationship to other chapters

This appendix enforces and extends:

- API contracts (chapter 5);
- Security and privacy guarantees (chapter 14);
- Operational reliability (chapter 15);
- Error semantics (Appendix C).

External integrations are first-class architectural concerns.

G.10 Conclusion

By clearly defining integration boundaries and constraints, the platform can interoperate with external systems without compromising integrity, privacy, or reliability.

This appendix ensures that extensibility remains intentional and governed.

Appendix H

Compliance Audit Checklist

This appendix provides a structured checklist for compliance audits. It is intended to support internal reviews, external assessments, and regulatory due diligence.

The checklist focuses on evidence-based verification rather than policy intent.

H.1 Audit scope definition

Before initiating an audit, confirm the scope:

- Systems and environments included (production, staging);
- Time period under review;
- Applicable regulations or standards;
- Data categories involved (learning data, metadata).

Audit scope must be documented and approved.

H.2 Data protection and privacy

Verify compliance with data protection principles.

- Personal data is minimized and purpose-limited;
- Learning data is treated as sensitive information;
- No unnecessary PII is stored or exported;
- Aggregated analytics prevent re-identification;
- Data access is logged and auditable.

Reference: chapter 14, chapter 6.

H.3 Authentication and authorization

Verify identity and access controls.

- Authentication mechanisms are enforced in production;
- Development authentication paths are disabled in production;
- Authorization rules restrict users to their own data;
- Editorial and administrative roles are segregated;
- Access reviews are performed periodically.

Reference: chapter 5, chapter 14.

H.4 Data integrity and immutability

Verify integrity safeguards at the database level.

- Primary and foreign key constraints enforced;
- Uniqueness constraints prevent duplicate attempts;
- Submitted sessions are immutable;
- Content versions remain historically stable;
- No direct database manipulation bypasses constraints.

Reference: chapter 6.

H.5 Change management

Verify controlled handling of changes.

- All changes tracked via version control;
- Database migrations reviewed and approved;
- Backward compatibility maintained;
- Architectural decisions documented (ADR);
- Emergency changes documented retroactively.

Reference: Appendix B, chapter 15.

H.6 Logging, monitoring, and audit trails

Verify observability and traceability.

- Access and operational events logged;
- Logs protected from tampering;
- Sensitive data excluded from logs;
- Audit trails retained per policy;
- Monitoring alerts configured for anomalies.

Reference: chapter 15.

H.7 Incident response

Verify preparedness for security and data incidents.

- Incident response procedures documented;
- Roles and responsibilities defined;
- Incident logs and reports available;
- Post-incident reviews performed;
- Corrective actions tracked to completion.

Reference: chapter 14.

H.8 External integrations

Verify compliance of external system interactions.

- External integrations documented;
- Least-privilege access enforced;
- No direct database access by external systems;
- Integration failures do not corrupt internal data;
- Data sharing complies with privacy rules.

Reference: Appendix G.

H.9 Analytics and reporting

Verify analytical practices.

- Analytics derived only from submitted sessions;
- Metrics are reproducible and explainable;
- No mutation of operational data;
- Analytical outputs reviewed for bias or misuse;
- Access to analytics is role-restricted.

Reference: chapter 12.

H.10 Documentation completeness

Verify that documentation is current and complete.

- Architecture and data models documented;
- API contracts and error semantics defined;
- Governance and editorial workflows documented;
- Operational procedures available;
- Glossaries and ADRs maintained.

Reference: Entire handbook.

H.11 Audit findings and remediation

Document audit results.

- Findings categorized by severity;
- Evidence collected and archived;
- Remediation actions assigned;
- Deadlines and owners defined;
- Follow-up verification scheduled.

Audit reports must be retained for accountability.

H.12 Conclusion

By using this compliance audit checklist, the platform demonstrates proactive risk management, regulatory awareness, and operational maturity.

This appendix enables audits to be systematic, repeatable, and evidence-based.

Appendix I

Data Retention and Deletion Policy

This appendix defines the policies governing data retention, archival, and deletion within the platform.

Its purpose is to balance educational value, analytical integrity, legal compliance, and user privacy.

This policy applies to all environments unless explicitly stated otherwise.

I.1 Policy objectives

The data retention policy is designed to:

- Preserve the integrity of learning history;
- Support longitudinal analytics and research;
- Minimize unnecessary storage of personal data;
- Enable compliance with applicable data protection principles;
- Ensure safe and auditable deletion processes.

Retention decisions are intentional and documented.

I.2 Data classification

All data handled by the platform falls into the following categories:

I.2.1 Learning data

Includes:

- Sessions;

- Session items;
- Attempts;
- Derived learning metrics.

Learning data is considered high-value and sensitive.

I.2.2 Content data

Includes:

- Questions and versions;
- Answer choices;
- Explanations;
- Bibliographic references.

Content data is treated as institutional knowledge.

I.2.3 Operational metadata

Includes:

- Logs;
- Audit trails;
- Deployment and monitoring data.

Operational metadata supports security and reliability.

I.3 Retention periods

I.3.1 Learning data

- Retained for the lifetime of the platform by default;
- Required to support longitudinal learning analysis;
- Subject to anonymization where appropriate.

Submitted sessions are never deleted automatically.

I.3.2 Content data

- Retained indefinitely;
- Versioned and immutable once published;
- Archived but never removed to preserve historical accuracy.

I.3.3 Operational metadata

- Retained for a limited period (e.g., 30–180 days);
- Retention period defined by operational and security needs;
- Automatically purged after expiration.

Retention durations may vary by environment.

I.4 Deletion principles

Data deletion follows strict principles:

- Deletion is intentional, not implicit;
- Deletion operations are auditable;
- Referential integrity must not be violated;
- Historical and analytical consistency is preserved.

Hard deletion is avoided whenever possible.

I.5 User-initiated data requests

Users may request actions related to their data, including:

- Access to personal learning data;
- Correction of factual errors in metadata;
- Deletion or anonymization of personal identifiers.

Handling strategy

- Identity is verified prior to processing requests;
- Learning records are anonymized rather than deleted;
- Educational and analytical integrity is preserved.

This approach balances user rights and system integrity.

I.6 Anonymization and pseudonymization

When deletion of learning data is not feasible, anonymization is applied.

- Direct identifiers are removed or replaced;
- Deterministic user identifiers are decoupled;
- Re-identification is prevented.

Anonymized data may still be used for aggregated analytics.

I.7 Deletion workflow

All deletion or anonymization actions follow a controlled workflow:

1. Request intake and validation;
2. Impact assessment on data integrity;
3. Approval by authorized roles;
4. Execution via controlled procedures;
5. Verification and audit logging.

Ad-hoc deletion is prohibited.

I.8 Backups and archival data

Backups are subject to separate retention rules.

- Backups are retained for disaster recovery only;
- Backup retention periods are finite;

- Deleted data may persist in backups until expiration;
- Restored backups must respect current policies.

Backup handling prioritizes system recoverability.

I.9 Compliance considerations

This policy aligns with principles from:

- General data protection regulations (e.g., data minimization);
- Educational data protection best practices;
- Internal governance and audit requirements.

Compliance is achieved through transparency and documented processes.

I.10 Audit and verification

Retention and deletion practices are periodically audited.

- Retention schedules reviewed annually;
- Deletion logs verified;
- Anonymization effectiveness assessed;
- Policy adherence documented.

Audit findings are addressed via corrective actions.

I.11 Relationship to other chapters

This policy enforces and complements:

- Data model integrity (chapter 6);
- Security and privacy guarantees (chapter 14);
- Operational safeguards (chapter 15);
- Compliance audits (Appendix H).

Data retention is a cross-cutting concern.

I.12 Conclusion

By explicitly defining data retention and deletion policies, the platform demonstrates responsible data stewardship, regulatory awareness, and long-term operational maturity.

This appendix completes the handbook's coverage of the full data lifecycle.

Appendix J

Legal Disclaimers and Terms Alignment

This appendix documents how the technical design and operational behavior of the platform align with legal disclaimers, terms of service, and user-facing policies.

Its purpose is to ensure consistency between what the system does, what is documented, and what is legally communicated.

This appendix does not replace legal documents; it aligns engineering reality with them.

J.1 Purpose and scope

This appendix establishes a bridge between:

- System behavior as implemented;
- Public-facing legal documents;
- Internal governance and compliance practices.

It applies to all environments and user interactions.

J.2 Educational disclaimer

The platform provides educational content only.

- Content is intended for learning and exam preparation;
- Content does not constitute medical advice;
- No clinical decisions should be made based on platform output.

This disclaimer aligns with:

- Content governance rules;
- Editorial review processes;
- Versioned and referenced explanations.

J.3 No guarantee of outcomes

The platform does not guarantee:

- Exam results;
- Professional certification;
- Academic or clinical performance.

Analytics and performance metrics are informational and probabilistic.
This aligns with the analytics model described in chapter 12.

J.4 Limitation of liability

From a system design perspective:

- The platform minimizes risk through validation and controls;
- Failures are handled predictably (Appendix C);
- Data integrity is enforced at multiple layers.

Legal limitation of liability clauses must reflect these technical realities, without overstating guarantees.

J.5 Data usage and ownership

User data

- Users retain rights over personal data;
- The platform processes data for educational purposes only;
- Learning data may be anonymized for analytics.

Platform data

- Content, structure, and analytics models are proprietary;
- Versioned content constitutes institutional knowledge.

This section aligns with Appendix I.

J.6 Consent and user responsibility

Users are responsible for:

- Understanding platform limitations;
- Using content appropriately;
- Complying with applicable laws and regulations.

Consent mechanisms must be explicit and auditable.

J.7 Privacy policy alignment

Privacy-related disclosures must accurately reflect:

- What data is collected;
- Why it is collected;
- How long it is retained;
- How it is protected.

Technical enforcement is described in:

- chapter 14;
- Appendix I.

No undocumented data processing is permitted.

J.8 Terms of service alignment

Terms of Service must align with actual system behavior:

- Session immutability after submission;
- Attempt limits enforced by the database;
- Content versioning and historical accuracy;
- Account access and suspension mechanisms.

Any divergence requires either:

- System changes; or
- Legal document updates.

J.9 Jurisdiction and regulatory posture

The platform is designed to be adaptable across jurisdictions.

- Core principles are jurisdiction-agnostic;
- Local requirements may introduce additional constraints;
- Compliance adaptations are documented explicitly.

Jurisdiction-specific rules must not undermine core guarantees.

J.10 Auditability and evidence

The system provides evidence to support legal claims:

- Logs and audit trails;
- Versioned documentation;
- Architectural decision records (Appendix B);
- Compliance checklists (Appendix H).

This supports legal defensibility.

J.11 Change management and legal review

Any change affecting:

- Data handling;
- User rights;
- Content interpretation;
- Liability exposure

must trigger a coordinated review involving:

- Engineering;
- Product;
- Editorial;
- Legal.

Unilateral changes are prohibited.

J.12 Relationship to other chapters

This appendix aligns legal positioning with:

- Security and privacy guarantees (chapter 14);
- Data retention policies (Appendix I);
- Compliance audits (Appendix H);
- Operational practices (Appendix E).

Legal accuracy depends on technical truth.

J.13 Conclusion

By aligning legal disclaimers and terms with actual system behavior, the platform avoids misrepresentation, reduces legal risk, and strengthens trust with users and partners.

This appendix completes the handbook's integration of engineering, governance, and legal accountability.

Bibliography

[1] *Next.js Documentation*. Vercel. URL: https://nextjs.org/docs (visited on 01/30/2026).

[2] *NextAuth.js Documentation*. NextAuth.js. URL: https://next-auth.js.org/ (visited on 01/30/2026).

[3] *PostgreSQL Documentation*. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/ (visited on 01/30/2026).

[4] *Prisma Documentation*. Prisma. URL: https://www.prisma.io/docs (visited on 01/30/2026).

[5] *Vercel Documentation*. Vercel. URL: https://vercel.com/docs (visited on 01/30/2026).

[6] *Zod Documentation*. Zod. URL: https://zod.dev/ (visited on 01/30/2026).