USMLE API – Contract & Project Snapshot

"Living" Engineering Handbook

HelpUS

February 27, 2026

Contents

About this document 1

1 Rules of engagement (Mandatory) 2
- 1.1 Scope and applicability . . . 2
- 1.2 Assistant-only mandatory rules . . . 2
  - 1.2.1 Before any code or document change (Assistant-only) . . . 3
  - 1.2.2 Explicit change notification and pause (Assistant-only) . . . 3
  - 1.2.3 One step at a time (Assistant-only) . . . 3
- 1.3 General collaboration rules . . . 4
  - 1.3.1 Incremental and testable changes . . . 4
  - 1.3.2 Testing and verification . . . 4
- 1.4 Quality and integrity gates . . . 4
- 1.5 Documentation-first discipline . . . 4
- 1.6 Persistence across conversations . . . 5

2 Stack and architecture 6
- 2.1 Current stack . . . 6
- 2.2 Runtime and deployment model . . . 6
- 2.3 Database access and transactions . . . 7
  - 2.3.1 `withTx` helper . . . 7
- 2.4 High-level system architecture . . . 7
- 2.5 Administrative and ingestion endpoints . . . 7

3 Repository layout (snapshot) 9
- 3.1 Backend routes (App Router) . . . 9
- 3.2 Client helper . . . 10

4 Authentication contract 11
- 4.1 Development override header . . . 11
- 4.2 Browser/production . . . 11
- 4.3 Deterministic user id . . . 11

5 API contract **12**
5.1 Design principles . . . 12
5.2 Sessions . . . 12
5.2.1 Create session . . . 12
5.2.2 List sessions . . . 13
5.2.3 Generate session items . . . 13
5.2.4 Submit session . . . 14
5.2.5 Review session . . . 14
5.3 Session items . . . 15
5.3.1 Get question . . . 15
5.3.2 Attempt question . . . 15
5.4 User statistics . . . 15
5.4.1 Get statistics . . . 15
5.5 Administrative and development endpoints . . . 16
5.5.1 Development seed (import-only) . . . 16
5.6 Utility endpoints . . . 17
5.6.1 Health check . . . 17
5.6.2 Debug headers . . . 17
5.7 Endpoint summary . . . 17

6 Data model **19**
6.1 Scope and architectural principles . . . 19
6.2 PostgreSQL Enums . . . 20
6.3 Content Engine (Question System) . . . 20
6.3.1 `questions` . . . 20
6.3.2 `question_versions` . . . 21
6.3.3 `question_choices` . . . 21
6.3.4 `tags` and `question_version_tags` . . . 22
6.4 Study Engine . . . 22
6.4.1 `sessions` . . . 22
6.4.2 `session_items` . . . 22
6.4.3 `attempts` . . . 22
6.5 Spaced Repetition Layer . . . 23
6.5.1 `user_question_state` . . . 23
6.5.2 `user_reviews` . . . 23
6.6 Billing & Subscription Layer . . . 24
6.7 Usage & Entitlements . . . 24
6.7.1 `usage_counters` . . . 24
6.7.2 `usage_grants` . . . 24

6.7.3 `user_entitlements` . . . 24
6.8 Promotions . . . 24
6.8.1 `promotions` . . . 24
6.8.2 `promotion_redemptions` . . . 25
6.9 Feedback System . . . 25
6.9.1 `question_feedback` . . . 25
6.10 Integrity Guarantees . . . 25
6.10.1 Idempotency rules . . . 25
6.10.2 ON DELETE policies . . . 25
6.11 Model evolution log . . . 25

7 UI/UX Blueprint: Session Review 27
7.1 Design goals . . . 27
7.2 High-level layout overview . . . 27
7.3 Semantic states of a question . . . 27
7.3.1 Unanswered state (player) . . . 28
7.3.2 Answered state (immediate review) . . . 28
7.3.3 Post-session review state . . . 28
7.4 Answer choice presentation . . . 29
7.5 Educational explanation blocks . . . 29
7.5.1 Educational Objective . . . 29
7.5.2 Correct answer explanation . . . 29
7.5.3 Incorrect option explanations . . . 29
7.5.4 Key Concept / Bottom Line . . . 30
7.5.5 Exam Tip (optional) . . . 30
7.6 Reader and interaction controls . . . 30
7.7 References and external learning resources . . . 30
7.8 Navigation behavior . . . 31
7.9 Relationship to other chapters . . . 31
7.10 Conclusion . . . 31

8 Infrastructure and deployment 32
8.1 Deployment model overview . . . 32
8.2 Vercel deployment . . . 32
8.2.1 Deploy trigger policy . . . 32
8.3 Environment variables . . . 32
8.3.1 Required variables . . . 33
8.4 Database connectivity . . . 33
8.5 Administrative endpoints and security . . . 33

8.6 Operational boundaries . . . 33

9 Current status and roadmap **35**

9.1 Current status . . . 35

10 System Analysis and Design Process **36**

10.1 Problem context . . . 36
10.2 Requirements analysis . . . 36
10.2.1 Functional requirements . . . 36
10.2.2 Non-functional requirements . . . 37
10.3 Analysis of modeling alternatives . . . 37
10.3.1 Attempt-only model . . . 37
10.3.2 Multiple attempts per question . . . 38
10.3.3 Application-level idempotency only . . . 38
10.4 Key architectural decisions . . . 38
10.4.1 Explicit session items . . . 38
10.4.2 Database-enforced idempotency . . . 38
10.4.3 Propagation of `question_version_id` . . . 39
10.5 Risk analysis and mitigation . . . 39
10.6 Analyst workflow . . . 39
10.7 Relationship to other chapters . . . 40
10.8 Conclusion . . . 40

11 Content and Question Model **41**

11.1 Educational principles . . . 41
11.2 Core content entities . . . 41
11.3 Question and versioning model . . . 42
11.4 Answer choices . . . 42
11.5 Explanations per choice . . . 43
11.6 Educational explanation blocks . . . 43
11.6.1 Educational Objective . . . 43
11.6.2 Key Concept / Bottom Line . . . 43
11.6.3 Exam Tip (optional) . . . 43
11.7 Multiple explanation layers . . . 43
11.8 Bibliographic and external references . . . 44
11.9 Relationship to attempts and sessions . . . 44
11.10 Content lifecycle . . . 45
11.11 Future extensions . . . 45
11.12 Relationship to other chapters . . . 45
11.13 Conclusion . . . 46

12 Analytics and Learning Metrics Model **47**
12.1 Analytical objectives . . . 47
12.2 Sources of analytical data . . . 47
12.3 Granularity levels . . . 48
12.4 Core learning metrics . . . 48
12.4.1 Accuracy . . . 48
12.4.2 Time-on-task . . . 48
12.4.3 Confidence vs. correctness . . . 49
12.5 Review interaction signals . . . 49
12.6 Session-level analytics . . . 50
12.7 Longitudinal learning analysis . . . 50
12.8 Content effectiveness metrics . . . 50
12.9 Constraints and safeguards . . . 51
12.10 Operationalization . . . 51
12.11 Future extensions . . . 51
12.12 Relationship to other chapters . . . 52
12.13 Conclusion . . . 52

13 Editorial Workflow and Governance **53**
13.1 Editorial objectives . . . 53
13.2 Roles and responsibilities . . . 53
13.2.1 Content author . . . 54
13.2.2 Medical reviewer . . . 54
13.2.3 Editorial reviewer . . . 54
13.2.4 Platform administrator . . . 54
13.3 Editorial lifecycle . . . 55
13.4 Versioning and change control . . . 55
13.5 Educational blocks governance . . . 55
13.6 Traceability and audit . . . 56
13.7 Integration with analytics . . . 56
13.8 External references and resources . . . 57
13.9 Governance constraints . . . 57
13.10 Exception handling . . . 57
13.11 Relationship to other chapters . . . 58
13.12 Conclusion . . . 58

14 Security, Privacy, and Compliance **59**
14.1 Security objectives . . . 59
14.2 Threat model . . . 59

14.3 Authentication and authorization . . . 60
14.3.1 Authentication . . . 60
14.3.2 Authorization . . . 60
14.4 Data minimization and separation . . . 61
14.5 Integrity and immutability guarantees . . . 61
14.6 Privacy considerations . . . 61
14.7 External resources and integrations . . . 62
14.8 Compliance and regulatory alignment . . . 62
14.9 Logging, monitoring, and audit . . . 62
14.10 Environment separation . . . 63
14.11 Incident response . . . 63
14.12 Relationship to other chapters . . . 63
14.13 Conclusion . . . 64

15 Deployment, Operations, and Reliability 65
15.1 Deployment objectives . . . 65
15.2 Environment model . . . 65
15.3 Deployment strategy . . . 66
15.3.1 Continuous deployment . . . 66
15.3.2 Backward compatibility . . . 66
15.4 Database operations . . . 67
15.5 Observability . . . 67
15.6 Reliability principles . . . 67
15.7 Failure handling and recovery . . . 68
15.7.1 Application failures . . . 68
15.7.2 Database failures . . . 68
15.8 Operational safeguards . . . 68
15.9 Change management . . . 68
15.10 Relationship to other chapters . . . 69
15.11 Conclusion . . . 69

16 Roadmap, Technical Debt, and Future Research 70
16.1 Purpose of the roadmap . . . 70
16.2 Short-term roadmap . . . 70
16.2.1 Advanced review experience . . . 71
16.2.2 Reader and interaction controls . . . 71
16.2.3 Operational hardening . . . 71
16.3 Mid-term roadmap . . . 71
16.3.1 Educational enrichment . . . 71

16.3.2 Analytics expansion . . . 72
16.3.3 Content management . . . 72
16.4 Long-term roadmap . . . 72
16.4.1 Personalized learning paths . . . 72
16.4.2 External educational resource integration . . . 72
16.4.3 Research-driven assessment . . . 72
16.5 Technical debt . . . 73
16.5.1 Known areas of debt . . . 73
16.5.2 Debt management principles . . . 73
16.6 Research directions . . . 73
16.6.1 Learning science . . . 73
16.6.2 Assessment theory . . . 73
16.6.3 System design research . . . 74
16.7 Decision review cadence . . . 74
16.8 Relationship to other chapters . . . 74
16.9 Conclusion . . . 74

A Glossary **75**

B Architectural Decision Records (ADR) **78**

C Error Codes and API Failure Semantics **84**
C.1 Error design principles . . . 84
C.2 Standard error response format . . . 84
C.3 HTTP status code semantics . . . 85
C.4 Canonical error codes . . . 87
C.4.1 Authentication and authorization . . . 87
C.4.2 Session lifecycle errors . . . 87
C.4.3 Session item and attempt errors . . . 87
C.4.4 Validation errors . . . 87
C.5 Idempotency and conflict semantics . . . 87
C.6 Security-related errors . . . 87
C.7 Internal server errors . . . 88
C.8 Client behavior guidelines . . . 88
C.9 Relationship to other chapters . . . 89
C.10 Conclusion . . . 89

D System Diagrams **90**
D.1 System architecture overview . . . 90
D.2 Data model relationships . . . 91

D.3 Session lifecycle flow . . . 92
D.4 Attempt flow and idempotency . . . 92
D.5 Analytics derivation flow . . . 92
D.6 Editorial content flow . . . 92
D.7 Relationship to other chapters . . . 94
D.8 Conclusion . . . 94

E Operational Checklists **95**
E.1 Pre-deployment checklist . . . 95
E.2 Deployment checklist . . . 95
E.3 Post-deployment checklist . . . 96
E.4 Database change checklist . . . 96
E.5 Incident response checklist . . . 97
E.6 Security review checklist . . . 97
E.7 Editorial governance checklist . . . 97
E.8 Analytics validation checklist . . . 98
E.9 Operational readiness review . . . 98
E.10 Relationship to other chapters . . . 98
E.11 Conclusion . . . 99

F Technical Glossary and Acronyms **100**

G External Integrations **104**
G.1 Integration principles . . . 104
G.2 Types of external integrations . . . 104
G.2.1 Authentication providers . . . 104
G.2.2 Analytics and monitoring services . . . 105
G.2.3 Content and reference providers . . . 105
G.2.4 Payment and subscription systems . . . 105
G.3 Integration boundaries . . . 106
G.4 Failure and resilience model . . . 106
G.5 Security considerations . . . 106
G.6 Data privacy and compliance . . . 107
G.7 Operational management . . . 107
G.8 Future integration scenarios . . . 107
G.9 Relationship to other chapters . . . 108
G.10 Conclusion . . . 108

H Compliance Audit Checklist **109**
H.1 Audit scope definition . . . 109
H.2 Data protection and privacy . . . 109
H.3 Authentication and authorization . . . 110
H.4 Data integrity and immutability . . . 110
H.5 Change management . . . 110
H.6 Logging, monitoring, and audit trails . . . 111
H.7 Incident response . . . 111
H.8 External integrations . . . 111
H.9 Analytics and reporting . . . 112
H.10 Documentation completeness . . . 112
H.11 Audit findings and remediation . . . 112
H.12 Conclusion . . . 113

I Data Retention and Deletion Policy **114**
I.1 Policy objectives . . . 114
I.2 Data classification . . . 114
I.2.1 Learning data . . . 114
I.2.2 Content data . . . 115
I.2.3 Operational metadata . . . 115
I.3 Retention periods . . . 115
I.3.1 Learning data . . . 115
I.3.2 Content data . . . 116
I.3.3 Operational metadata . . . 116
I.4 Deletion principles . . . 116
I.5 User-initiated data requests . . . 116
I.6 Anonymization and pseudonymization . . . 117
I.7 Deletion workflow . . . 117
I.8 Backups and archival data . . . 117
I.9 Compliance considerations . . . 118
I.10 Audit and verification . . . 118
I.11 Relationship to other chapters . . . 118
I.12 Conclusion . . . 119

J Legal Disclaimers and Terms Alignment **120**
J.1 Purpose and scope . . . 120
J.2 Educational disclaimer . . . 120
J.3 No guarantee of outcomes . . . 121
J.4 Limitation of liability . . . 121

J.5 Data usage and ownership . . . 121
J.6 Consent and user responsibility . . . 122
J.7 Privacy policy alignment . . . 122
J.8 Terms of service alignment . . . 123
J.9 Jurisdiction and regulatory posture . . . 123
J.10 Auditability and evidence . . . 123
J.11 Change management and legal review . . . 124
J.12 Relationship to other chapters . . . 124
J.13 Conclusion . . . 124

K Project Journal and Execution Tracker **125**
K.1 Current status (single source of truth) . . . 125
K.2 Daily log (append-only) . . . 126
K.3 Execution plan (next phase) . . . 134
K.4 Backlog (prioritized) . . . 135
K.5 Reference pointers . . . 136

L Project Journal (Rolling Log) **137**
L.1 Rolling summary (current state) . . . 137
L.2 Daily log (rolling, append-only) . . . 138
L.3 Artifacts and pointers . . . 139

List of Figures

7.1 Review page layout (conceptual wireframe) 28

D.1 High-level system architecture 90
D.2 Logical data model relationships 91
D.3 Session lifecycle flow 92
D.4 Attempt recording and idempotency 93
D.5 Analytics derivation 93
D.6 Editorial content lifecycle 94

List of Tables

5.1 Primary API endpoints 18

6.1 Database enums 20

C.1 HTTP status code usage 85
C.2 Authentication and authorization errors 86
C.3 Session-related errors 86
C.4 Attempt-related errors 86
C.5 Validation errors 86

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]
- **Authentication:** NextAuth v4.x (confirmed in production: 4.24.13) [2]
- **Runtime / Hosting:** Vercel (serverless execution for frontend and API routes) [6]
- **Database:** PostgreSQL (managed service) [3]
- **Database provider:** Railway (Postgres-only; no application runtime) [5]
- **ORM:** Prisma (schema maintained in-repository; migrations explicit) [4]
- **Validation:** Zod (request and payload validation at API boundaries) [7]

2.2 Runtime and deployment model

The system follows a **split-runtime architecture**:

- **Application runtime (Frontend + API):** Executed on **Vercel**, including all Next.js App Router pages and API routes (e.g., `/api/*`).
- **Persistence layer:** PostgreSQL hosted on **Railway**, used strictly as a managed database service.

Railway is **not** used to host application code or API endpoints. All HTTP-accessible endpoints (including administrative import routes) are exposed exclusively via the Vercel deployment.

2.3 Database access and transactions

All database interaction occurs through explicit SQL queries executed inside controlled transactions.

2.3.1 `withTx` helper

- All SQL statements are executed via `client.query`.
- Each request operates inside a single explicit transaction.
- No implicit database logic (triggers, stored procedures) is relied upon.
- Mixing Prisma client operations with raw SQL within the same request flow is explicitly avoided.

The `DATABASE_URL` environment variable is injected into the Vercel runtime and points to the Railway-managed PostgreSQL instance.

2.4 High-level system architecture

At MVP level, the backend is responsible for enforcing the complete **session lifecycle** and associated invariants:

1. Create session (`in_progress`);
2. Generate session items (idempotent per session);
3. Record attempts (idempotent per session item);
4. Submit session (`submitted`);
5. Allow review of submitted sessions only;
6. Aggregate learning and performance statistics for submitted sessions.

2.5 Administrative and ingestion endpoints

In addition to learner-facing flows, the backend exposes restricted administrative endpoints for controlled data ingestion.

- Administrative routes (e.g., `POST /api/dev/seed-minimal`) run in the same Vercel environment as user-facing APIs.
- Access is protected via explicit header-based authentication (e.g., `x-admin-key`).

- The backend acts strictly as an **import and validation layer**: it does not generate educational content in code.

This separation enforces a clear architectural boundary between external content authoring pipelines and the application runtime.

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 mandatory 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 and documented error behavior;
- No educational content generation inside application code.

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
}
```

Rules

- A session is always created in status `in_progress`;
- The authenticated user becomes the immutable owner of the session.

5.2.2 List sessions

`GET /api/sessions`

Returns all sessions belonging to the authenticated user.

Rules

- Only session metadata is returned;
- Session items and attempts are not included in this response.

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 uniqueness and integrity rules defined in chapter 6.

Failure conditions

- The session does not exist;
- The session does not belong to the authenticated user;
- The session is already submitted.

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 to the current timestamp.

Rules

- A submitted session becomes immutable;
- No new attempts may be recorded after submission.

5.2.5 Review session

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

Returns the full review of a submitted session, including:

- questions;
- user attempts;
- correctness and explanation data (choice-level).

Precondition The session MUST be in status `submitted`.

Failure response (example)

```
{ "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.

Rules

- The correct answer MUST NOT be revealed;
- Explanations MUST NOT be included 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 **??**.

Rule (Idempotency) Repeated requests with the same payload MUST NOT create duplicate attempts.

If an attempt already exists, the API MUST return the existing attempt.

Additional rules

- The session MUST be in status `in_progress`;
- The session item MUST belong to the specified session;
- The authenticated user MUST own the session.

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.

Design note The aggregation logic relies exclusively on immutable attempt data and is safe against partial or in-progress sessions.

5.5 Administrative and development endpoints

5.5.1 Development seed (import-only)

`POST /api/dev/seed-minimal`

Imports externally authored questions into the system. This endpoint operates in **import-only mode** and MUST NOT generate or modify educational content in code.

Authentication Requests MUST include the header:

`x-admin-key: <ADMIN_SEED_KEY>`

Request body (schema overview)

```
{
  "questions": [
    {
      "stem": "...",
      "difficulty": "easy" | "medium" | "hard",
      "explanation_short": "...",
      "explanation_long": "...",
      "bibliography": { ... },     // optional (JSON)
      "prompt": "...",             // optional
      "choices": [
        { "label": "A", "text": "...", "correct": false, "explanation": "..." },
        { "label": "B", "text": "...", "correct": true,  "explanation": "..." }
      ]
    }
  ],
  "chunkSize": 10,
  "requireExactlyTen": true
}
```

Behavioral rules

- Each question MUST include 4 or 5 answer choices;
- Exactly one choice MUST be marked as correct;
- Each choice MUST include a non-empty explanation;
- In pilot mode, exactly 10 questions are required;
- All inserts are executed transactionally.

Responses

- **201 Created**: import completed successfully;
- **400 Bad Request**: schema validation or pilot constraint failure;
- **403 Forbidden**: missing or invalid admin key.

Security note This endpoint is intended for controlled administrative use only and MUST NOT be exposed to untrusted clients.

5.6 Utility endpoints

5.6.1 Health check

`GET /api/health`

Simple liveness probe for infrastructure monitoring.

5.6.2 Debug headers

`GET /api/debug/headers`

Echoes request headers to assist in authentication debugging during development.

5.7 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
Admin	POST /api/dev/seed-minimal	Import questions (admin only)

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, constraints, and integrity guarantees.

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

6.1 Scope and architectural principles

The data model is designed with:

- Strict referential integrity (database-enforced FK constraints);
- Idempotent write guarantees via schema-level uniqueness;
- Explicit separation between:
  - Content (questions + versions),
  - Study sessions,
  - User learning state,
  - Billing / entitlements,
  - Editorial tagging.
- Historical immutability of submitted sessions;
- Forward compatibility for analytics, AI ranking, and adaptive systems.

Table 6.1: Database enums

Enum	Values
attempt_result	correct, wrong, skipped
session_status	in_progress, submitted, abandoned
session_mode	practice, timed_block, exam_sim
exam_type	step1, step2ck
difficulty_level	easy, medium, hard
question_status	draft, published, archived
billing_provider	stripe
billing_event_status	received, processed, error
invoice_status	paid, open, void, uncollectible
payment_status	succeeded, requires_payment_method, failed
plan_period	none, monthly, yearly
subscription_status	active, trialing, past_due, canceled, incomplete, paused
promotion_type	trial_days, plan_override, usage_bonus
entitlement_source	free_default, subscription, promo, admin_grant
feedback_type	typo, wrong_key, unclear, too_hard, other
feedback_status	open, triaged, fixed
reset_period	daily, monthly, never
tag_type	discipline, system, topic, subtopic, keyword

6.2 PostgreSQL Enums

6.3 Content Engine (Question System)

6.3.1 questions

Canonical identity of a question.

- Primary key: question_id
- canonical_code UNIQUE
- status (draft/published/archived)
- source (text, indexed)
- created_at / updated_at

Indexes:

- idx_questions_source
- idx_questions_status
- UNIQUE(canonical_code)

6.3.2 `question_versions`

Versioned content snapshot.

- FK → `questions`
- UNIQUE(`question_id`, `version`)
- exam (enum)
- difficulty (enum)
- stem (text)
- explanation_short
- explanation_long
- prompt (nullable)
- bibliography (jsonb)
- is_active (boolean)

Indexes:

- `idx_qv_exam_lang`
- `idx_qv_difficulty`
- `idx_qv_question_id`

6.3.3 `question_choices`

- FK → `question_versions`
- label (A–E)
- choice_text
- is_correct
- explanation
- UNIQUE(`question_version_id`, `label`)

6.3.4 `tags` and `question_version_tags`

Editorial classification layer.

- tag_type enum
- UNIQUE(`tag_type`, `name`)
- Many-to-many via `question_version_tags`

6.4 Study Engine

6.4.1 `sessions`

User study container.

- FK → `users_profile`
- exam
- mode
- status
- settings_json (jsonb snapshot)
- started_at / submitted_at

6.4.2 `session_items`

Ordered list of presented questions.

- FK → `sessions` (CASCADE)
- FK → `question_versions`
- UNIQUE(`session_id`, `position`)

6.4.3 `attempts`

One attempt per session item.

- UNIQUE(`session_item_id`)
- FK → `sessions`
- FK → `question_versions`

- FK → `question_choices` (nullable)
- result enum
- is_correct (denormalized helper)

Indexes:

- `idx_attempts_user_time`
- `idx_attempts_session`
- `idx_attempts_qv`

6.5 Spaced Repetition Layer

6.5.1 `user_question_state`

Aggregated exposure metrics per user/question.

- PK(`user_id`, `question_id`)
- times_seen
- times_correct
- last_result
- bookmarked

6.5.2 `user_reviews`

SM-2 style scheduling.

- next_due_at
- interval_days
- ease_factor
- status

Index:

- `idx_reviews_user_due`

6.6 Billing & Subscription Layer

Core entities:

- plans
- plan_limits
- subscriptions
- invoices
- payments
- billing_customers
- billing_events

All billing events are idempotent via:

- UNIQUE(`provider_event_id`)
- UNIQUE(`provider_payment_intent_id`)
- UNIQUE(`provider_subscription_id`)

6.7 Usage & Entitlements

6.7.1 `usage_counters`

Tracks periodic usage (daily/monthly).

UNIQUE(`user_id`, `metric`, `period_start`, `period_end`)

6.7.2 `usage_grants`

Tracks bonus or promotional usage.

6.7.3 `user_entitlements`

Single row per user defining active plan access.

6.8 Promotions

6.8.1 `promotions`

Promotion metadata.

6.8.2 promotion_redemptions

UNIQUE(promotion_id, user_id)

6.9 Feedback System

6.9.1 question_feedback

User-generated quality reports.

Indexed by:

- question_version_id
- created_at

6.10 Integrity Guarantees

6.10.1 Idempotency rules

- One attempt per session item
- Unique canonical question code
- Unique tag names per type
- Unique billing provider IDs
- Unique promotion redemption per user

6.10.2 ON DELETE policies

- CASCADE for session-bound data
- NO ACTION for historical question references
- SET NULL for authorship metadata

No triggers are defined. All guarantees rely on constraints and indexes.

6.11 Model evolution log

2026-01-27 — Initial MVP schema

Core study engine introduced.

2026-01-29 — Idempotency hardening

UNIQUE(`session_item_id`) added.

2026-02-11 — Full production schema snapshot

Integrated billing, promotions, tagging, spaced repetition, and question source tracking.

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 Deployment model overview

The system adopts a **decoupled deployment model**, separating application execution from data persistence:

- **Application runtime:** Vercel (Next.js frontend and API routes).
- **Database service:** Railway (managed PostgreSQL instance only).

No application code or HTTP endpoints are deployed on Railway. All user-facing and administrative APIs are served exclusively from the Vercel environment.

8.2 Vercel deployment

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

8.2.1 Deploy trigger policy

- **Enabled:** GitHub integration (push to `main` branch).
- **Disabled:** external deploy hooks, to avoid duplicate or out-of-band deployments.

Each successful deployment produces an immutable build artifact and a unique deployment URL.

8.3 Environment variables

All runtime configuration is provided through environment variables defined at the Vercel project level.

8.3.1 Required variables

- `DATABASE_URL`: PostgreSQL connection string pointing to the Railway-managed database.
- `ADMIN_SEED_KEY`: Secret key required to authorize administrative import operations.

Environment variables must be defined explicitly for the `production` environment. Any change to these variables requires a redeployment to take effect.

8.4 Database connectivity

The application connects to PostgreSQL using the connection string provided by `DATABASE_URL`. All database access is mediated by the transaction helper (`withTx`) and executed using explicit SQL queries.

The Railway service is used strictly as a managed database provider and does not expose application endpoints or background workers.

8.5 Administrative endpoints and security

Administrative and development-only endpoints (e.g., `POST /api/dev/seed-minimal`) are deployed together with the main application and share the same runtime environment.

- Access to administrative endpoints is protected via a mandatory request header (`x-admin-key`).
- The header value must match `ADMIN_SEED_KEY`.
- Unauthorized requests are rejected with HTTP 403.

This model ensures that sensitive import operations are:

- Explicitly authenticated;
- Executed only in controlled environments;
- Auditable via application logs.

8.6 Operational boundaries

The deployment architecture enforces the following constraints:

- No educational content is generated during deployment.

- No database schema changes occur implicitly at runtime.
- Content ingestion is performed exclusively via authenticated API calls.

These boundaries are intentional and support reproducibility, auditability, and operational safety.

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 **??**);
- **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 specifies 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.

Analytics are strictly read-only and never alter operational data.

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 exist to support pedagogical insight, learner feedback, and system improvement — not surveillance, behavioral manipulation, or opaque scoring.

12.2 Sources of analytical data

All learning metrics are derived exclusively from operational interaction data.

Primary sources include:

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

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

Additional non-evaluative signals may be derived from review interactions.

Invariant No analytical data is manually entered or edited. All metrics are reproducible from canonical operational tables.

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 trajectories.

This layered structure supports both micro-level feedback and macro-level insight without conflating individual interactions with long-term learning signals.

12.4 Core learning metrics

12.4.1 Accuracy

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

Rules

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

Accuracy is never interpreted in isolation.

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.

Time metrics are interpreted relative to question type, difficulty, and session mode.

12.4.3 Confidence vs. correctness

When confidence capture is enabled, confidence is analyzed in relation to correctness.

This enables identification of:

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

Confidence signals are never used as grading inputs.

12.5 Review interaction signals

The review experience generates additional non-evaluative engagement signals.

Examples include:

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

Design rule Review signals inform engagement and effort, never correctness or scoring.

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.

Invariant Session analytics are immutable once the session is submitted and reviewed.

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 related concepts;
- Reduction in time-on-task;
- Improvement in confidence calibration;
- Stability of performance under time pressure.

Longitudinal analysis emphasizes durable learning over short-term gains.

12.8 Content effectiveness metrics

Content analytics evaluate pedagogical quality rather than user ability.

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, calibration, and content refinement.

12.9 Constraints and safeguards

The analytics model enforces strict safeguards:

- No metric mutates historical attempt or session data;
- Analytics never write to operational tables;
- Derived metrics are deterministically reproducible;
- Only submitted sessions are included in reporting;
- External resource usage does not affect scoring.

These safeguards preserve trust, auditability, and interpretability.

12.10 Operationalization

Analytics may be implemented using:

- SQL aggregation queries;
- Cached or materialized views;
- Periodic batch jobs;
- On-demand API computation.

Implementation choices must balance performance, freshness, cost, and transparency.

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.

Constraint All future analytics MUST remain explainable, auditable, 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, pedagogical integrity, and system consistency.

Chapter 13

Editorial Workflow and Governance

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

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

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

Editorial governance is authoritative over technical convenience.

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.

Educational content is treated 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 structured 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.

Medical review is mandatory prior to publication.

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 review readability.

13.2.4 Platform administrator

Responsible for:

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

The platform administrator acts as the final gatekeeper.

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 or deactivation after use in submitted sessions.

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

This invariant is supported by the data model defined in chapter 6.

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, replacement, or reinterpretation of referenced guidelines.

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

Rule 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 align with the question's intent;
- Blocks MUST NOT introduce contradictions or 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, activation, and deactivation 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 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.

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

This rule preserves pedagogical responsibility and ethical integrity.

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, review, and approval of a corrected new version;
- Explicit documentation of the incident and its 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;
- 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 learner 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.

Security, privacy, and compliance are treated as first-class system properties.

14.1 Security objectives

The security model is designed to achieve the following objectives:

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

Security is a system-wide concern and not an afterthought or optional layer.

14.2 Threat model

The platform explicitly 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 or exposure of development, debug, or administrative endpoints;
- Re-identification risks via aggregated or poorly scoped 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 and guarantees are defined in detail in chapter 5.
Credentials, tokens, and secrets are never persisted in learning or analytics tables.

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 and never delegated to client-side logic.

14.4 Data minimization and separation

The platform follows strict data minimization and separation 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 optional and non-evaluative.

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

No direct personal identifiers are required for learning analytics.

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 corruption and malicious tampering.

They are enforced structurally, not by convention.

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 never receive learner data.

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

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 and principles from:

- General data protection regulations (e.g., GDPR principles);
- Educational data protection and learner 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.

Constraint Logs must never expose sensitive content, learner answers, or personal data. Auditability is prioritized over verbosity.

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.

This separation limits blast radius and supports safe experimentation.

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 learner 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 uniformly across all system layers.

14.13 Conclusion

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

This chapter completes the security, privacy, 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 **??**.

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 canonical error model used by the API. It standardizes error responses, HTTP status codes, and failure semantics to ensure predictable client behavior, safe retries, 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 `code` values are stable and contractually defined;
- Clients MUST be able to react programmatically;
- Internal implementation 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": { ... }
  }
}
```

Client rule Clients MUST rely on the `code` field for logic and control flow. The `message` field is intended for diagnostics and user-facing mapping only.

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

Design note Status codes classify failures at a high level. Precise semantics are defined by the error `code`.

Table C.2: Authentication and authorization errors

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

Table C.3: Session-related errors

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

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 for this session item
ATTEMPT_INVALID_STATE	409	Attempt not allowed in the current session state
ATTEMPT_PAYLOAD_INVALID	422	Invalid attempt payload

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 from request

C.4 Canonical error codes

C.4.1 Authentication and authorization

C.4.2 Session lifecycle errors

C.4.3 Session item and attempt errors

C.4.4 Validation errors

C.5 Idempotency and conflict semantics

Certain conflicts are expected, safe, and non-fatal.

Example: duplicate attempt submission If a client retries an attempt submission for the same session item, the API MAY return:

```
HTTP 409
{
  "error": {
    "code": "ATTEMPT_ALREADY_EXISTS",
    "message": "Attempt already recorded for this session item"
  }
}
```

Client interpretation Clients MUST treat this response as success-equivalent and MUST NOT retry.

Design rationale This behavior aligns with the database uniqueness constraint defined in **??**, which guarantees one attempt per session item.

C.6 Security-related errors

Security-related errors MUST:

- Avoid revealing sensitive information;
- Use generic messages when appropriate;
- Be logged internally for audit and monitoring 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"
  }
}
```

Rule Internal error details MUST be logged server-side and MUST NEVER be returned to clients.

C.8 Client behavior guidelines

Clients interacting with the API SHOULD:

- Branch logic based on the error `code`;
- Retry safely on idempotent conflicts only;
- Avoid retrying validation or authorization failures;
- Map error codes to stable, user-friendly messages.

Client behavior MUST align with the semantics defined in this appendix.

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.

Failures are predictable, explainable, and safe by design.

Appendix D

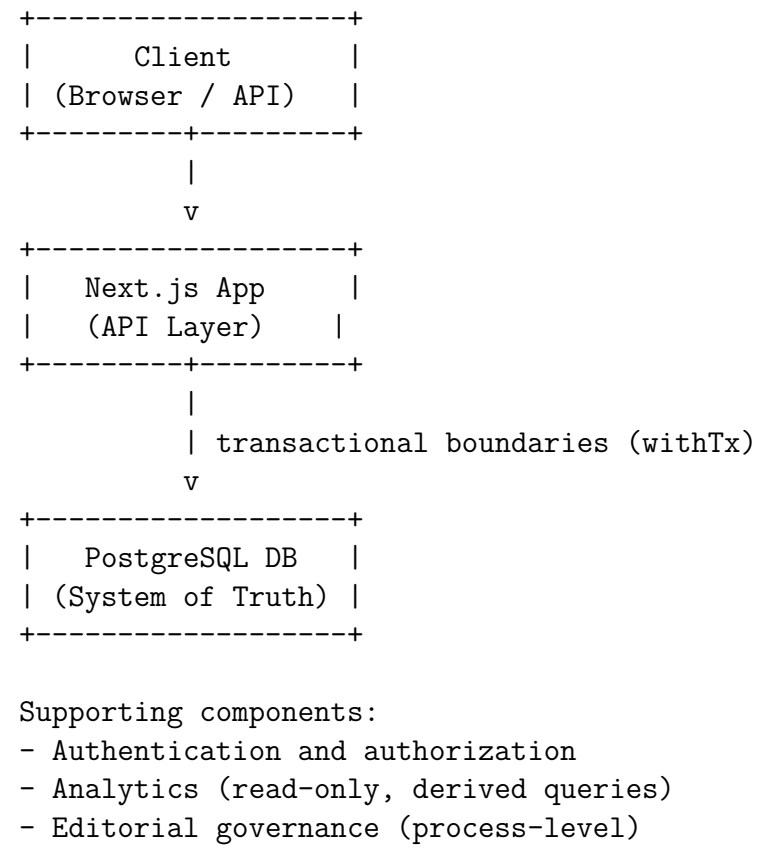
System Diagrams

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 architectural reasoning. They do not replace the canonical definitions in the data model or API contract.

D.1 System architecture overview

Figure D.1: High-level system architecture

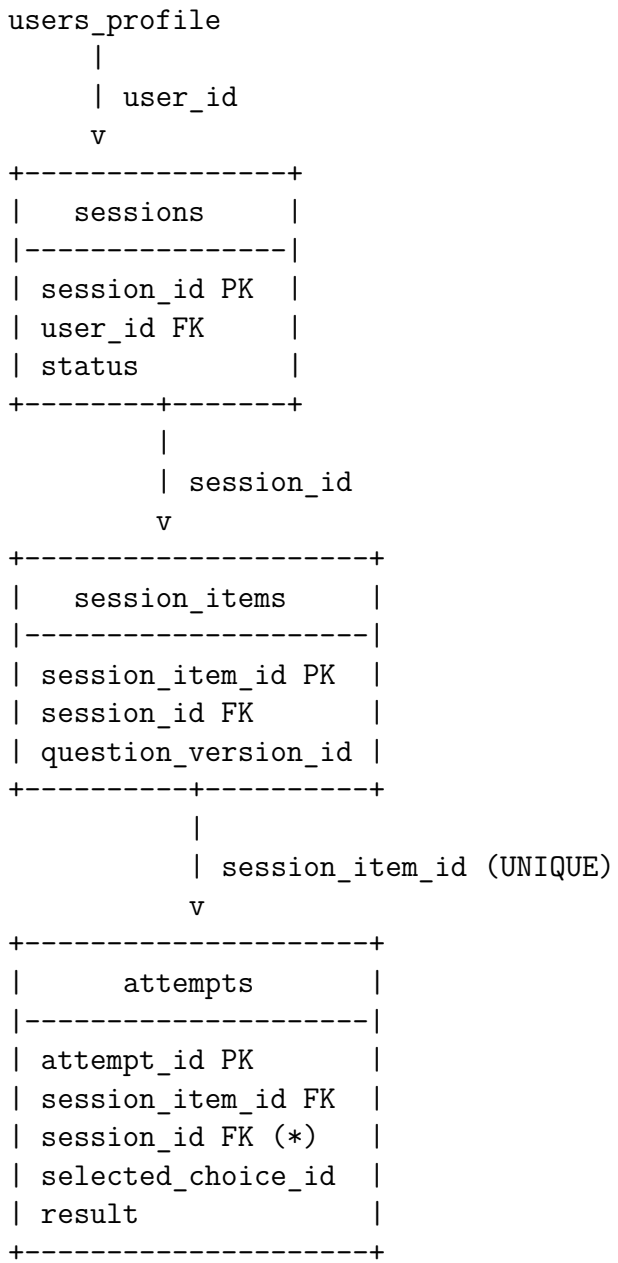


This architecture emphasizes:

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

D.2 Data model relationships

Figure D.2: Logical data model relationships



Design note The `attempts.session_id` foreign key is redundant by design and exists for query efficiency. Consistency between `attempts.session_id` and `session_items.session_id` is enforced at the API layer.

This diagram reflects the canonical structure defined in chapter 6.

D.3 Session lifecycle flow

Figure D.3: Session lifecycle flow

```
[Create Session]
       |
       v
[Generate Session Items]
       |
       v
[Attempt Questions]
       |
       | (repeat per item)
       v
[Submit Session]
       |
       v
[Review Session]
```

Invariant Once a session is submitted, it becomes immutable. This rule is enforced by the API contract and supported by the data model.

D.4 Attempt flow and idempotency

Semantic note A uniqueness conflict does not represent a failure. Clients must treat it as success-equivalent.

This invariant is formally defined in **??**.

D.5 Analytics derivation flow

Invariant Analytics never mutate operational data and never depend on in-progress sessions.

D.6 Editorial content flow

This flow aligns with governance rules defined in chapter 13.

Figure D.4: Attempt recording and idempotency

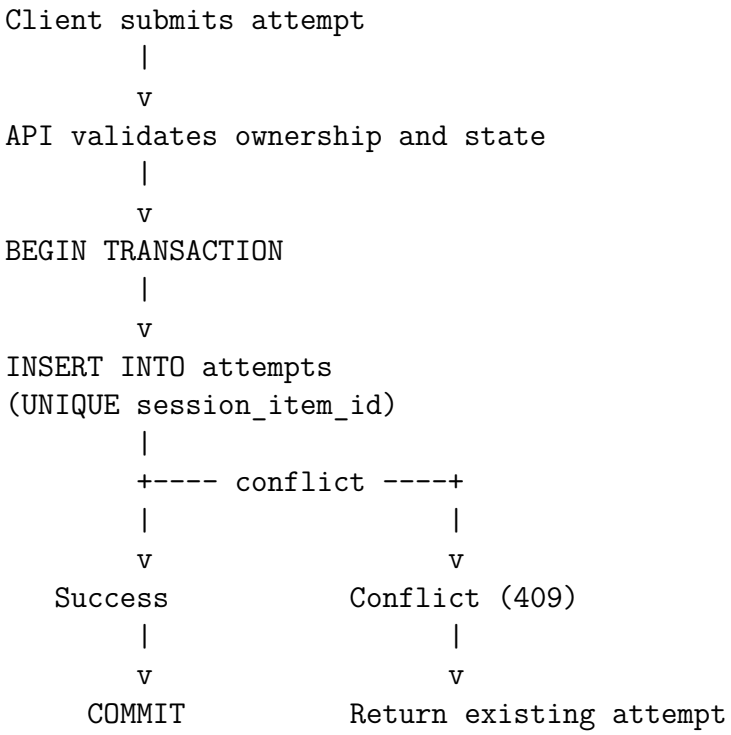


Figure D.5: Analytics derivation

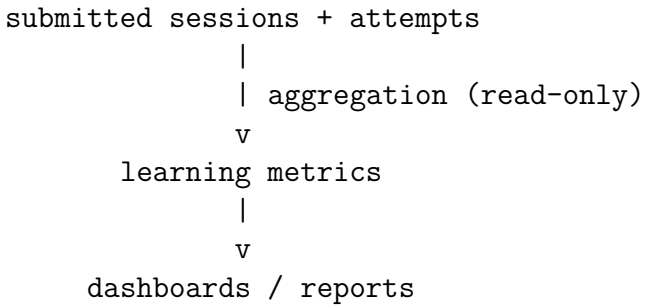
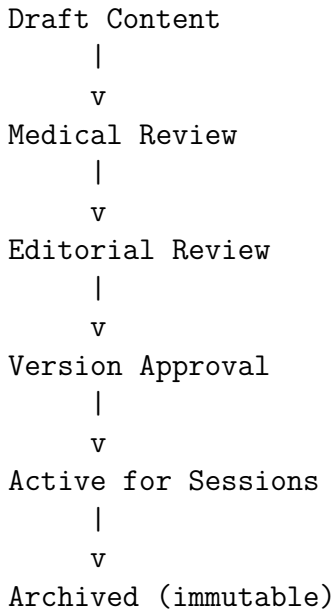


Figure D.6: Editorial content lifecycle



D.7 Relationship to other chapters

These diagrams visually support:

- API behavior (chapter 5);
- Data integrity rules (chapter 6);
- Analytics and learning metrics (chapter 12);
- Editorial governance (chapter 13);
- Operational reliability (chapter 15).

They serve as a conceptual map of the system, not as an executable specification.

D.8 Conclusion

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

Diagrams should evolve alongside the system, but must always reflect canonical design decisions documented elsewhere in this handbook.

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.

Appendix K

Project Journal and Execution Tracker

This appendix is the living execution log of the USMLE platform project. It serves as the single operational memory of decisions, implementations, and planned work discussed and executed incrementally.

This document is append-only and updated chronologically.

K.1 Current status (single source of truth)

- **Last updated:** 2026-02-02 12:15 (BRT)
- **Current focus:** Execution and hardening of the external open-access ingestion pipeline. The objective is to ingest a pilot batch of **10 Step 1–like questions** sourced from open-access medical literature, validated through strict quality gates (full-text availability and extraction), and imported into Railway PostgreSQL for end-to-end validation of Review UX v2.
- **Code state:** Backend review endpoint expanded to support full review rendering (`choices[]` + per-choice `explanation`; plus `bibliography/prompt`). Frontend ReviewPage compiles cleanly and renders baseline review. Review UX v2 update to render all choices + explanations is prepared and awaiting runtime validation after ingestion. **Import endpoint confirmed in backend:** `POST /api/dev/seed-minimal` (transactional, no ORM/Prisma required, protected by `x-admin-key`).
- **Database state:** PostgreSQL schema live (Railway), fully introspected and documented; integrity enforced via PK, FK, UNIQUE, ENUMs; no triggers. Confirmed relevant fields/constraints for Review UX and ingestion: `question_choices.explanation (text)`, `question_versions.bibliography`

(jsonb), question_versions.prompt (text), attempts.session_item_id FK ON DELETE CASCADE, uq_attempt_per_item UNIQUE index on attempts.session_item_id.

- **Documentation state:** Core handbook chapters aligned and consistent. Review UX and content model semantics are stable. Pending: formal handbook section documenting the external ingestion pipeline, its governance, and quality gates.

Top blockers

- Convert the enriched/ready batch format (seed_pilot_10_ready.json) into the exact payload schema expected by /api/dev/seed-minimal.
- Execute a controlled TRUNCATE reset of questions-related tables before importing the pilot into Railway.

Top risks

- Legal/licensing risk if external sources are ingested without explicit open-access verification and full-text availability checks.
- Educational quality risk if external material is not sufficiently transformed into original Step 1–like vignettes and **choice-level explanations**.
- Contract drift risk: if the batch JSON schema used by scripts is not frozen against the backend seed endpoint schema, ingestion may silently break.

K.2 Daily log (append-only)

2026-01-30 18:00–22:00 (BRT)

Context: Migration of an existing technical project into a formal, auditable engineering handbook.

- **Done:**
  - Initial review of the Living Engineering Handbook structure;
  - Alignment of project scope and intent with USMLE platform goals;
  - Decision to treat the handbook as the canonical system reference.
- **Decisions:**
  - (D1) Documentation is the system of truth, not the code.
  - (D2) Database invariants must be explicit and reviewable.

- **Planned next:**
  - Full database introspection and validation against schema.

2026-01-31 09:00–13:00 (BRT)

Context: Deep inspection of the PostgreSQL schema in Railway.

- **Done:**
  - Full table listing, column types, defaults, and nullability;
  - Enumeration of all ENUM types and values;
  - Extraction of all primary keys, foreign keys, and unique indexes;
  - Verification of ON DELETE behaviors (CASCADE, SET NULL);
  - Confirmation that no triggers or stored procedures exist.
- **Decisions:**
  - (D3) No hidden DB logic: all semantics live in schema + API.
  - (D4) Session integrity relies on API validation, not DB triggers (for now).
- **Planned next:**
  - Update Data Model chapter to reflect the real schema.

2026-01-31 13:00–17:30 (BRT)

Context: Consolidation of core handbook chapters.

- **Done:**
  - Data Model chapter finalized (chapter 6);
  - API Contract chapter aligned (chapter 5);
  - Error Codes appendix finalized (Appendix C);
  - System Diagrams appendix completed (Appendix D).
- **Decisions:**
  - (D5) Diagrams are conceptual, not implementation-dependent.
  - (D6) Idempotency and immutability are enforced by design.

2026-01-31 17:30–21:30 (BRT)

Context: Governance, analytics, and security consolidation.

- **Done:**
  - Analytics model finalized (chapter 12);
  - Editorial governance finalized (chapter 13);
  - Security, privacy, and compliance finalized (chapter 14).
- **Decisions:**
  - (D7) Analytics must be explainable and non-manipulative.
  - (D8) Editorial changes are always versioned and auditable.
  - (D9) Security and privacy are design properties, not features.

2026-01-31 21:30–23:55 (BRT)

Context: Review UX upgrade + payload enrichment planning, based on real-world study apps and a reference content experience.

- **Done:**
  - Defined the target review experience based on real app references;
  - Confirmed the objective: explain *why each alternative is correct or incorrect*;
  - Adopted visual semantics (color cues) to guide learning in the review UI;
  - Confirmed database support for choice-level explanations: `question_choices.explanation (text)`;
  - Confirmed additional enrichment fields: `question_versions.bibliography (jsonb)` and `question_versions.prompt (text)`;
  - Confirmed referential integrity detail used by the review system: FK on `attempts.session_item_id` with `ON DELETE CASCADE`;
  - Confirmed idempotency index: `uq_attempt_per_item` UNIQUE on `attempts.session_item`
- **Decisions:**
  - (D10) Review UX must operate at **choice-level granularity**.
  - (D11) Educational blocks are first-class learning artifacts (Educational Objective, Bottom Line, Comment, Exam Tip, References).
  - (D12) External learning resources (e.g., podcasts) are linked, never embedded, and never receive user data.

  - (D13) Review endpoint/state semantics: attempting to review a non-submitted session is a state conflict (prefer HTTP 409).

- **Implementation notes:**
  - Backend review endpoint updated to return: session metadata + items + full `choices[]` per `question_version_id` including `explanation` per choice, plus `bibliography` and `prompt`.
  - Frontend ReviewPage compiles cleanly, is stable, and already applies visual semantics (color cues). Baseline flow is working end-to-end (answer → submit → review).

- **Planned next:**
  - Update chapter 7 with the post-answer/review screen specification (UX v2);
  - Update chapter 5 to formalize the enriched review payload (choices with explanations, references, podcasts/resources);
  - Define the learning resources layer: integrate `divineinterventionpodcasts.com` as a link-only recommendation surface (topic/tag mapping; privacy-safe).

2026-02-01 20:30–21:30 (BRT)

Context: Frontend Review UX v2 implementation work.

- **Done:**
  - Confirmed the current ReviewPage file is stable and compiling without warnings;
  - Identified a learning gap: review still showed correct answer review but not incorrect alternatives (missing "why wrong" coverage);
  - Prepared a ReviewPage UX v2 update to render **all choices** and show an explanation section per choice ("Why correct" / "Why wrong") using `question_choices.explanation`;
  - Decided to default the UI to show explanations for **all** alternatives, to ensure full learning coverage (not only correct + selected).

- **Decisions:**
  - (D14) Review learning completeness requirement: all alternatives should be reviewable with explicit rationale (why correct/why wrong).
  - (D15) If explanations are missing for wrong alternatives, the issue is content/seed, not the review UI (UI should still render placeholders safely).

- **Planned next:**
  - Validate runtime behavior of the "all choices + explanation" UI update;
  - If wrong-choice explanations are missing, update the content creation/seed pipeline to populate `question_choices.explanation` for all options;
  - After UI is validated, update chapter 7 and chapter 5 with the finalized contract.

2026-02-01 21:30–23:30 (BRT)

Context: Content strategy reset for scalable Step 1–like long-form questions, and redefinition of the seed/import workflow.

- **Done:**
  - Revalidated the requirement that questions must be long, reasoning-heavy, and structurally similar to real Step 1 clinical vignettes.
  - Confirmed that interactive/manual creation inside the app is not viable at scale (target is thousands of questions).
  - Reviewed the existing seed route draft and identified a misalignment: it still supported in-code generation and embedded templates (not scalable).
  - Agreed to start small with a high-quality pilot batch (=10 questions), validate end-to-end UX, then scale.
- **Decisions:**
  - (D16) Adopt a **hybrid model** for content:
    - Open datasets are used only as *structural inspiration*;
    - All final text (stem, choices, explanations) must be deeply rewritten, educationally optimized, and original;
    - Explanations are fully new and authored at **choice-level** (why correct / why wrong for every option).
  - (D17) The platform will not "generate questions in code" as a primary strategy. The backend is an ingestion system; authoring is external/editorial.
  - (D18) Seed/import must be **batch-first** and non-interactive, supporting thousands of questions without manual UI interaction.
  - (D19) Any script-based acquisition/curation pipeline must prioritize permissive licensing and compliance: no proprietary NBME/USMLE/UWorld content, and no near-copy transformations.
- **Implementation notes:**

  - The canonical import format remains the JSON schema already described (stem, difficulty, short/long explanation, optional bibliography/prompt, and choices with correct flag + per-choice explanation).
  - The seed route should evolve to **import-only** mode and accept validated JSON batches (chunked uploads), avoiding embedded question templates.
  - The pilot dataset (10 questions) will be used to validate:
    * Database write correctness (questions, question_versions, choices, explanations);
    * Review UX v2 completeness (all choices + why correct/why wrong);
    * Content fit (Step 1-like depth and length).

- **Planned next:**
  - Implement an external ingestion script (PowerShell or Python) that:
    * fetches candidates from open sources,
    * normalizes format,
    * outputs JSON batches compatible with the import endpoint.
  - Define the authoritative list of acceptable open datasets and their licensing.
  - Execute the TRUNCATE reset (questions-related tables) before importing the pilot 10.
  - Validate runtime behavior of Review UX v2 with the imported pilot batch.

2026-02-02 08:30–09:30 (BRT)

Context: Strategic decision and execution alignment for scalable, Step 1–like content sourcing via external open-access datasets.

- **Done:**
  - Explicitly abandoned any approach involving manual or in-code question creation or templating.
  - Reaffirmed the platform role as a **content ingestion and review system**, not an authoring environment.
  - Selected external open-access medical literature as the primary upstream source for question material.
  - Defined the immediate execution target: ingest a pilot batch of **10 Step 1–like questions** using a real ingestion pipeline.

- **Decisions:**

  - (D20) Questions are never authored or templated inside application code; all educational content enters the system via ingestion.
  - (D21) Only open-access, permissively licensed medical sources are eligible for content acquisition.
  - (D22) External materials are used solely as factual and structural input; all final question text and explanations must be original and rewritten.
  - (D23) The ingestion pipeline operates end-to-end in real time: fetch → filter → transform → validate → persist.
  - (D24) A 10-question pilot batch is mandatory as a validation gate before any scale-up of content ingestion.

- **Implementation notes:**
  - Target source category includes open-access clinical articles and case reports suitable for Step 1–level reasoning.
  - The transformation stage must generate long-form vignettes, exactly five alternatives, and per-choice explanations (why correct / why wrong).
  - Bibliography references must be link-only and point to the original open-access source.

- **Planned next:**
  - Define and freeze the authoritative list of acceptable open-access sources and licensing constraints.
  - Define the canonical ingestion JSON schema for externally sourced questions.
  - Implement the external ingestion pipeline (script-based).
  - Execute a controlled TRUNCATE of question-related tables.
  - Ingest and validate the 10-question pilot batch end-to-end.

2026-02-02 09:30–12:15 (BRT)

Context: End-to-end execution of the open-access ingestion pipeline, including metadata enrichment, full-text verification, automatic replacement, and generation of Step 1–like stems and choice-level explanations.

- **Done:**
  - Confirmed the ingestion endpoint present in backend: `POST /api/dev/seed-minimal` (protected by `x-admin-key`).

- Executed the pilot batch generation pipeline (scripts):
  - Generated a 10-item pilot batch JSON (`pilot_batch_10.json`) and validated with `VALIDATION_PROFILE=dryrun`.
  - Ran `enrich_metadata.py` to enrich items based on `external_refs.pmcid` and produce: `seed_pilot_10_enriched.json`.
  - Encountered 403 Forbidden when fetching PMC XML via NCBI web endpoint; adopted Europe PMC and NCBI E-utilities as the retrieval strategy.
  - Implemented full-text retrieval with contact email support (`CONTACT_EMAIL=helpus.ecom`
  - Ran extraction and quality classification; detected one item with insufficient text (`body_words=56`, `quality=too_little_text`).
  - Ran `audit_fulltext_coverage.py` over the 10 items: initial summary `OK_FULLTEXT=9`, `TOO_LITTLE_TEXT=1`.
  - Ran `replace_bad_item.py` to automatically replace the failing item with a new candidate meeting quality thresholds; reran audit to confirm **OK_FULLTEXT=10/10**.
  - Ran `build_step1_stems.py` to generate long-form Step 1 style vignettes: `seed_pilot_10_step1stems.json`.
  - Ran `build_choices_and_explanations.py` to generate exactly 5 choices and per-choice explanations (why correct/why wrong) for all questions: `seed_pilot_10_ready.json`.
- Confirmed that the pilot batch is now **educationally complete** at choice-level, and **full-text verified** for all 10 items.

- **Decisions:**
  - (D25) The authoritative import endpoint for the pilot is `/api/dev/seed-minimal`; no new admin import route will be created now.
  - (D26) Full-text availability is a hard gate: items failing extraction or minimum content threshold must be replaced automatically.
  - (D27) Retrieval strategy uses Europe PMC and NCBI E-utilities with an explicit contact email; direct PMC XML fetch may be blocked and must not be relied upon.
  - (D28) The pipeline must be deterministic and script-driven; no manual editing of individual items is allowed during pilot execution.
  - (D29) The pilot batch artifacts are frozen as intermediate canonical outputs: `enriched` → `step1stems` → `ready`.

  - (D30) Import is only allowed after a schema-conversion step aligns the `ready` structure to the seed endpoint contract.

- **Implementation notes:**
  - The current `ready` batch is richer than the seed-minimal payload schema (nested `question` object, `external_refs`, `batch_meta`), requiring a converter before import.
  - Step 1 stems are generated from verified full text; abstract-only sources or short-body content are excluded.
  - Discipline tagging used during generation is heuristic and will be refined later.

- **Planned next:**
  - Implement a converter script to map `seed_pilot_10_ready.json` to the exact `/api/dev/seed-minimal` payload schema.
  - Perform controlled TRUNCATE of question-related tables in Railway.
  - Import the pilot via `/api/dev/seed-minimal` using `x-admin-key`.
  - Validate Review UX v2 runtime with real data (all choices + explanations).

K.3 Execution plan (next phase)

1. **Phase 1 — Review UX v2 (Frontend)**
   - Render all alternatives with correctness highlighting;
   - Show per-choice explanation blocks (why correct, why wrong);
   - Ensure missing explanations degrade gracefully (placeholder text);
   - Add structured learning blocks (Educational Objective, Bottom Line, Exam Tip, References).
2. **Phase 2 — Contract hardening (Docs + API)**
   - Formalize enriched review payload in chapter 5;
   - Update chapter 7 with post-answer/review screens (UX v2);
   - Keep Data Model chapter aligned with confirmed fields and constraints;
   - Add explicit error semantics mapping to Appendix C.
3. **Phase 3 — Learning resources integration**
   - Map podcasts/resources to tags/topics (initial heuristic mapping);

- Display contextual recommendations during review (link-only);
- Preserve privacy and explainability (no tracking / no leakage).

4. **Phase 4 — External content ingestion (pilot)**
   - Select and validate open-access medical content sources;
   - Fetch and extract Step 1–relevant clinical material;
   - Transform external material into original questions with choice-level explanations;
   - Validate schema, licensing, and educational completeness;
   - Convert batch schema to seed endpoint contract;
   - Ingest the 10-question pilot batch into Railway and validate Review UX v2.

K.4 Backlog (prioritized)

P0 – Must do

- Review UX v2 implementation (choices + per-choice explanations UI);
- Ensure content/seed pipeline populates wrong-choice explanations;
- Formal API contract update for enriched review payload;
- Convert `seed_pilot_10_ready.json` to `/api/dev/seed-minimal` payload schema;
- Execute TRUNCATE reset (questions-related tables);
- Ingest the 10-question pilot batch from open-access sources into Railway;
- Validate Review UX v2 with real, externally sourced content.

P1 – Should do

- Content model update for educational blocks;
- Podcast/resource integration spec (topic/tag mapping + UI slot);
- Formalize ingestion governance and licensing verification in the handbook;
- Define topic coverage targets for Step 1 content ingestion.

P2 – Nice to have

- Analytics-driven resource prioritization;
- ER diagrams auto-generated from schema.

K.5 Reference pointers

- Data model: chapter 6
- API contract: chapter 5
- Error semantics: Appendix C
- System diagrams: Appendix D
- Analytics model: chapter 12
- Editorial governance: chapter 13
- Security, privacy, compliance: chapter 14

Appendix L

Project Journal (Rolling Log)

This appendix is the **rolling execution log** of the USMLE platform project. It is designed for **fast compilation** and day-to-day operational tracking.

Historical reference: the full legacy execution record (frozen) lives in Appendix K.

L.1 Rolling summary (current state)

- **Last updated:** 2026-02-03 21:40 (BRT)
- **Runtime architecture (confirmed):** Next.js API routes running on **Vercel**; PostgreSQL hosted on **Railway (DB-only)**.
- **Current focus:** import the 10-question pilot batch via the Vercel API and validate Review UX v2 runtime rendering.
- **Review UX:** frontend supports baseline review; UX v2 renders all choices with per-choice explanations (awaiting validation with real data).
- **Backend:** review payload already includes `choices[]` with per-choice `explanation`, plus `bibliography/prompt`.
- **Import endpoint (confirmed):** `POST /api/dev/seed-minimal` (Vercel-hosted, admin-key protected).
- **Ingestion pipeline status:** 10/10 items passed full-text coverage gates; outputs frozen as: `seed_pilot_10_enriched.json` → `seed_pilot_10_step1stems.json` → `seed_pilot_10_ready.json`.

Immediate next actions

1. Implement/execute a converter: `seed_pilot_10_ready.json` → payload schema expected by `/api/dev/seed-minimal`.

2. Controlled **TRUNCATE reset** of question-related tables in the Railway PostgreSQL database.

3. Import pilot batch via the Vercel API using `x-admin-key`.

4. Validate Review UX v2 end-to-end in the UI (all choices + explanations).

Current blockers

- Schema alignment: `ready` batch is richer than the seed-minimal input contract and must be mapped deterministically.
- Operational safety: TRUNCATE must be executed with a known-safe order and confirmation queries.

Current risks

- Contract drift between ingestion JSON and backend endpoint expectations.
- Licensing drift if OA verification is not enforced as a hard gate for future scaling.

L.2 Daily log (rolling, append-only)

2026-02-02–03 (BRT)

Context: Architecture alignment, external ingestion hardening, and pilot import preparation.

- **Done:**
  - Confirmed deployment model: Vercel hosts all API routes; Railway provides PostgreSQL only.
  - Confirmed backend seed endpoint: `POST /api/dev/seed-minimal`.
  - Enriched metadata for pilot batch and validated PMCID extraction.
  - Implemented OA full-text retrieval strategy using Europe PMC + NCBI E-utilities (contact email configured).
  - Audited full-text extraction coverage across 10 items; replaced 1 failing item automatically; re-audited to reach **10/10 OK_FULLTEXT**.
  - Generated Step 1-style long-form stems and finalized choice-level explanations for all items.
  - Produced final import candidate artifact: `seed_pilot_10_ready.json`.

  - Aligned Overleaf documentation (architecture, infra, API contract) with the actual runtime and ingestion pipeline.

- **Decisions:**
  - (Z1-D1) Rolling log maintained for performance; historical journal frozen in Appendix K.
  - (Z1-D2) Full-text availability is a hard gate; failing items are replaced automatically.
  - (Z1-D3) Import will use `/api/dev/seed-minimal`; no additional admin import route will be added.
  - (Z1-D4) Vercel is the single execution environment; Railway is used strictly as a managed database.

- **Planned next:**
  - Build the schema converter `ready` → `seed-minimal payload`.
  - TRUNCATE + import into Railway PostgreSQL via Vercel API.
  - Validate Review UX v2 runtime rendering with real imported data.

L.3 Artifacts and pointers

- Frozen historical journal: Appendix K
- Pilot artifacts:
  - `seed_pilot_10_enriched.json`
  - `seed_pilot_10_step1stems.json`
  - `seed_pilot_10_ready.json`
- Backend import endpoint: `POST /api/dev/seed-minimal`
- Review endpoint: `GET /api/sessions/[sessionId]/review`

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] *Railway Documentation*. Managed PostgreSQL hosting platform (database-only usage in this project). Railway Corp. URL: https://docs.railway.app (visited on 02/03/2026).

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

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