1. Content Development (Human/Expert Side)
a. Blueprint & Coverage

Align all questions to the exam blueprint (domains, competencies, learning objectives).

Example:

USMLE Step 1: Biochem, Physiology, Pathology, Micro, Pharm, Behavioral.

PLAB: Clinical scenarios in Medicine, Surgery, Pediatrics, OBGYN, Psychiatry.

Engineering FE Exam: Circuits, Electronics, Materials, Thermo, Probability, Ethics.

This prevents overemphasis on one topic while neglecting others.

b. Item Writing

SMEs (physicians, engineers, lawyers, etc.) write scenario-based questions.

Each question includes:

Stem (realistic scenario)

Lead-in (specific question, e.g. "Which of the following is the next best step?")

Options (1 best answer, 3–4 plausible distractors)

Explanation (long-form, with diagrams/tables)

UWorld differentiator → The explanation is like a mini textbook page, often longer than the question.

c. Review & Psychometrics

Questions are reviewed by:

Peer SMEs (accuracy, realism)

Editors (language, clarity)

Psychometricians (stats: difficulty, discrimination index)

Pilot testing (collect answer patterns → see if the right people get the right answer).

Revise or retire low-performing items.

🔷 2. Technical Infrastructure (Engineering Side)

This is the backend brain of a Qbank.

a. Data Model (Database Schema)

A simplified relational schema might look like:

Tables:

Questions

id (PK)

stem (text)

lead_in (text)

difficulty (float or categorical)

topic_id (FK → Topics)

created_at, updated_at

Options

id (PK)

question_id (FK → Questions)

option_text

is_correct (boolean)

Explanations

id (PK)

question_id (FK → Questions)

content (HTML/Markdown with images, tables)

Topics

id

name (e.g. “Cardiology”, “Op-Amps”)

blueprint_code (aligns to exam spec)

UserResponses

user_id

question_id

option_id

is_correct

time_taken

UserAnalytics

user_id

topic_id

accuracy

percentile_rank

study_recommendation

b. Core Features

Tagging/Metadata System
Each question tagged by: subject, difficulty, Bloom’s level, exam category.
→ Enables custom quizzes (e.g., “50 cardiology questions, medium difficulty”).

Adaptive Engine
Uses Item Response Theory (IRT) or Bayesian Knowledge Tracing:

If you miss a cardiology question, system serves another at similar difficulty.

If you master it, difficulty ratchets up.

This mimics personalized tutoring.

Analytics Layer

Performance dashboards → accuracy by subject, time per question, comparison to peers.

Heatmaps → highlight weak areas.

Percentile rank across population (UWorld does this very well).

Frontend Experience

Test mode (simulate real exam)

Tutor mode (see explanation immediately)

Highlighting, note-taking, flashcards integration

Mobile + Web sync

c. Content Management Workflow

CMS (Content Management System) for SMEs:

Web-based editor to draft MCQs.

Rich text + LaTeX for equations.

Image/media upload (diagrams, histology slides, circuits).

Version control (Git-like for questions).

Review Dashboard:

SMEs → Editors → Psychometricians.

Track status: Draft → In Review → Approved → Live.

d. Continuous Updating

Guideline Monitoring: e.g., new AHA CPR guidelines, new DSM edition, new IEEE standard.

Editors push updates.

Questions get flagged as “outdated” and retired or revised.

🔷 3. Example Workflow (How UWorld Might Build One Question)

SME Drafts Question: A cardiologist writes a vignette about chest pain.

Peer Review: Two other cardiologists review for accuracy.

Editorial Check: Language is tightened, references added.

Pilot Testing: 10,000 users answer it → data shows 65% get it correct, good discrimination.

Live: Added to Qbank with full explanation + illustrations.

Analytics: System records how many users get it right, which distractors are most picked.

Update: If new ACC guidelines change treatment, the question is revised.

✅ In short:

Content side = SMEs + psychometricians + editors.

Tech side = a relational database with tagging, adaptive algorithms, analytics dashboards, and CMS workflow.

Moat = high-quality explanations + continuous updating + psychometric calibration.
High-Level System Map

Core planes

Content plane (authoring + versioning)

Delivery plane (quiz gen + rendering + timing)

Telemetry plane (events + stream processing)

Analytics/ML plane (IRT calibration + adaptivity + recs)

Governance plane (security, PII, audit, RBAC)

Suggested tech (battle-tested choices)

Primary DB: PostgreSQL (with pg_partman for time partitions, pgvector for semantic retrieval)

Object storage: S3 (images, rich explanations, assets) + CDN

Search: OpenSearch or Algolia (full-text, filters)

Caching: Redis (hot questions, user session, rate limits)

Streams: Kafka (answer events, quiz starts/ends, item telemetry)

Warehouse: BigQuery/Snowflake/Redshift (analytics, IRT jobs)

ETL/Orchestration: dbt + Airflow (or Dagster)

Model serving: FastAPI + BentoML/MLflow (IRT, recommender, difficulty)

Observability: OpenTelemetry → Tempo/Jaeger + Prometheus/Grafana + Loki

Feature store (optional): Feast (user proficiency, topic priors)

1) Content Plane: Authoring, Versioning, Publishing
1.1 Canonical Data Model (normalized + append-only versions)
```
-- Topic taxonomy (multi-level tree)
CREATE TABLE topics (
  id BIGSERIAL PRIMARY KEY,
  parent_id BIGINT REFERENCES topics(id),
  name TEXT NOT NULL,
  blueprint_code TEXT,          -- aligns to official exam blueprint
  path LTREE                    -- optional for hierarchical queries
);

-- Core identity of a question (immutable identity)
CREATE TABLE questions (
  id BIGSERIAL PRIMARY KEY,
  external_ref TEXT UNIQUE,        -- human-friendly code (e.g., USMLE-IM-12345)
  created_by UUID NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  is_deleted BOOLEAN NOT NULL DEFAULT FALSE
);

-- Versioned payload (append-only; never UPDATE in place)
CREATE TABLE question_versions (
  id BIGSERIAL PRIMARY KEY,
  question_id BIGINT NOT NULL REFERENCES questions(id),
  version INT NOT NULL,                              -- 1..N
  state TEXT NOT NULL CHECK (state IN
('draft','review','approved','published','retired')),
  stem_md TEXT NOT NULL,                             -- Markdown/HTML
  lead_in TEXT NOT NULL,
  rationale_md TEXT NOT NULL,                        -- long explanation
  difficulty_label TEXT,                             -- 'easy','medium','hard' (editorial)
  bloom_level SMALLINT,                              -- taxonomy level
  topic_id BIGINT REFERENCES topics(id),
  tags JSONB NOT NULL DEFAULT '{}'::jsonb,           -- {exam:"USMLE",
sub:"cardio", ...}
  assets JSONB NOT NULL DEFAULT '[]'::jsonb,         -- S3 keys, image refs
  references JSONB NOT NULL DEFAULT '[]'::jsonb,     -- textbooks/guidelines
  created_by UUID NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE(question_id, version)
);

-- Options are versioned with the question_version
CREATE TABLE question_options (
  id BIGSERIAL PRIMARY KEY,
  question_version_id BIGINT NOT NULL REFERENCES question_versions(id) ON
DELETE CASCADE,
  option_label CHAR(1) NOT NULL,                     -- 'A','B','C','D','E'
  option_text_md TEXT NOT NULL,
  is_correct BOOLEAN NOT NULL
);

-- Publication table points to exactly one published version per tenant/exam
CREATE TABLE question_publications (
  id BIGSERIAL PRIMARY KEY,
  question_id BIGINT NOT NULL REFERENCES questions(id),
  live_version INT NOT NULL,                         -- must exist in question_versions
  exam_code TEXT NOT NULL,                           -- 'USMLE-S1-2025-Q2'
  tenant_id UUID NOT NULL,
  published_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE (question_id, tenant_id, exam_code)
);

-- “Change requests” + review workflow
CREATE TABLE question_reviews (
  id BIGSERIAL PRIMARY KEY,
  question_version_id BIGINT NOT NULL REFERENCES question_versions(id),
  reviewer_id UUID NOT NULL,
  status TEXT NOT NULL CHECK (status IN
('pending','changes_requested','approved','rejected')),
  comments_md TEXT,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Why this shape works

Immutable versions → reproducibility for audits, psychometrics, and citations.

Publication pointer → easy rollback/blue-green deploys of content by switching live_version.

Options tied to version → option text and correctness are frozen with the version.

Tags JSONB → flexible, index with GIN for fast filtering.

Indexes you’ll want

```
CREATE INDEX idx_qv_topic ON question_versions(topic_id);
CREATE INDEX idx_qv_state ON question_versions(state);
CREATE INDEX idx_qv_tags_gin ON question_versions USING GIN (tags
jsonb_path_ops);
CREATE INDEX idx_topics_path_gist ON topics USING GIST (path);      -- if using
ltree
```

1.2 Assets & LaTeX

Store media in S3: s3://qbanks/questions/{question_id}/v{version}/

Pre-render LaTeX → SVG/PNG on publish (build workers), store rendered assets alongside raw.

Use Content-Security-Policy and HTML sanitization if explanations allow limited HTML.

1.3 Search/Retrieval

Push searchable fields (stem, lead_in, topic, tags) to OpenSearch with a denormalized doc:

```
{"qid":123,"version":4,"topic_path":"Medicine.Cardiology.Ischemia", "tags":{...},
"difficulty":"medium","text":"..."}
```

For semantic “similar items” or study recs, embed (mpnet, e5, etc.) explanation text into pgvector:

```
ALTER TABLE question_versions ADD COLUMN emb vector(768);
-- cosine ivfflat index
CREATE INDEX idx_qv_emb ON question_versions USING ivfflat (emb
vector_cosine) WITH (lists = 100);
```

2) Delivery Plane: Quiz Generation & Session State
2.1 REST/GraphQL contracts

Create quiz

```
POST /v1/quizzes
{
  "user_id": "...",
  "tenant_id": "...",
  "filters": {
    "topics": ["Cardiology.Ischemia"],
    "difficulty": ["medium","hard"],
    "num_questions": 40,
    "mode": "tutor" | "exam",
    "exam_code": "USMLE-S1-2025-Q2"
  },
  "adaptive": true
}
→ 201 { "quiz_id": "...", "question_ids": [ ... ], "expires_at": "...", "mode":"exam" }
```

Fetch next question (adaptive or fixed)

```
GET /v1/quizzes/{quiz_id}/next
```

→ { "question_id": 12345, "version": 4, "payload": { stem_md, lead_in, options[], assets[] } }

Submit answer

```
POST /v1/quizzes/{quiz_id}/answers
{
  "question_id": 12345,
  "selected": "C",
  "time_taken_ms": 74210,
  "client_latency_ms": 83
}
```

→ 200 { "correct": false, "correct_option": "D", "explanation": { rationale_md, tables[], images[] }, "difficulty": 0.82 }

2.2 Delivery DB (sessions + responses)

```
CREATE TABLE quiz_sessions (
  id UUID PRIMARY KEY,
  user_id UUID NOT NULL,
  tenant_id UUID NOT NULL,
  mode TEXT NOT NULL CHECK (mode IN ('tutor','exam')),
  adaptive BOOLEAN NOT NULL DEFAULT FALSE,
  exam_code TEXT,
  started_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  expires_at TIMESTAMPTZ,
  sealed_at TIMESTAMPTZ
);

CREATE TABLE quiz_items (
  id BIGSERIAL PRIMARY KEY,
  quiz_id UUID NOT NULL REFERENCES quiz_sessions(id) ON DELETE CASCADE,
  question_id BIGINT NOT NULL,
  version INT NOT NULL,
  position INT NOT NULL,
  served_at TIMESTAMPTZ NOT NULL DEFAULT now(),
  UNIQUE(quiz_id, position)
);

CREATE TABLE user_responses (
  id BIGSERIAL PRIMARY KEY,
  quiz_id UUID NOT NULL REFERENCES quiz_sessions(id) ON DELETE CASCADE,
  user_id UUID NOT NULL,
  question_id BIGINT NOT NULL,
  version INT NOT NULL,
  option_label CHAR(1) NOT NULL,
  is_correct BOOLEAN NOT NULL,
  time_taken_ms INT,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

-- Fast lookups
CREATE INDEX idx_resp_user ON user_responses(user_id);
CREATE INDEX idx_resp_question ON user_responses(question_id, version);
```

Caching

Redis keys quiz:{quiz_id}:cursor (next position), quiz:{quiz_id}:served (set), user:{id}:weak_topics (hot features).

Concurrency

For timed exams, maintain server-authoritative timer, return deltas to client; prevent double submissions with idempotency keys.

3) Telemetry Plane: Events & Streaming

Everything the learner does → event.

Event envelope (Kafka topic: events.qbank)

```
{
  "event_id": "uuid",
  "event_type": "quiz_started|question_served|answer_submitted|
explanation_viewed|note_added|flagged",
  "user_id": "uuid",
  "tenant_id": "uuid",
  "quiz_id": "uuid",
  "question_id": 12345,
  "version": 4,
  "timestamp": "2025-08-15T20:12:31.123Z",
  "client": { "device":"ios", "app_version":"3.2.1", "latency_ms":83 },
  "payload": { "selected":"C", "is_correct": false, "time_taken_ms":74210 }
}
```

Stream processing

Flink/Kafka Streams job computes rolling stats: item p-value (proportion correct), option distractor attraction, median time, DIF signals (differential item functioning by cohort).

Write aggregates into ClickHouse (fast OLAP) or backfill to warehouse hourly.

4) Analytics & ML Plane
4.1 Item Calibration (IRT/CTT)

CTT:

p (difficulty proxy) = % correct

r (discrimination) = point-biserial correlation(correctness, total score)

IRT (2PL/3PL):

Parameters per item: a (discrimination), b (difficulty), c (guessing)

Batch fit nightly: use py-irt, mirt (R), or custom Stan/Turing model over warehouse data

Store fitted params:

```
CREATE TABLE item_calibration (
  question_id BIGINT PRIMARY KEY,
  version INT NOT NULL,
  model TEXT NOT NULL,                   -- '2PL','3PL'
  a FLOAT, b FLOAT, c FLOAT,
  se_a FLOAT, se_b FLOAT, se_c FLOAT,
  n_respondents INT,
  fitted_at TIMESTAMPTZ NOT NULL DEFAULT now()
);
```

Why per-version: wording changes shift parameters; tie psychometrics to the content you served.

4.2 Adaptive Engine (service)

Input: user ability θ estimate (per topic and global), item bank with IRT params, constraints (blueprint coverage).

Select next item maximizing information at current θ (Fisher information), respecting blueprint quotas and exposure control (Sympson-Hetter).

Update θ online via EAP/MLE after each response; cap updates in exam mode to avoid instability.

Feature store (optional) keeps:

user_id, topic_id, theta_mean, theta_var, last_update

4.3 Recommender (post-quiz)

Rank weak subtopics by low θ / low accuracy / high time-to-correct.

Retrieve similar items via pgvector (embedding of stems/explanations).

Blend: score = 0.6 * need(topic) + 0.3 * sim_embedding + 0.1 * novelty.

5) Governance & Multi-Tenancy
5.1 Tenant partitioning

Row-level: every row carries tenant_id.

RLS policies in Postgres for app roles:

```
ALTER TABLE questions ENABLE ROW LEVEL SECURITY;
CREATE POLICY tenant_isolation ON questions
  USING (tenant_id = current_setting('app.tenant_id')::uuid);
```

Alternative: Schema-per-tenant for enterprise deals; or physical DB sharding when >100M rows.

5.2 Roles

Roles: Author, Editor, Psychometrician, Publisher, Auditor, Student, InstitutionAdmin.

Fine-grained permissions table or policy engine (Oso/Casbin).

5.3 PII / Compliance

PII (name, email) in separate user store; reference only user_id in learning data.

Right-to-erasure → hard delete in PII store; pseudonymize historical events (keep stats).

Audit trail of content changes (append-only versions + reviews).

COPPA/GDPR/FERPA depending on market.

6) Performance, Scaling, and Cost

Hot path (serving questions) must be cache-friendly:

GET /next pulls from Redis prefetch queue; hydrate from Postgres on cache miss.

Write path (answers) is append-only to Postgres + emit Kafka; batch ETL to warehouse.

Indexes: keep them lean on user_responses; rely on partitioning by created_at for pruning.

Media via CDN; pre-generate responsive images; lazy-load explanations.

Throughput targets (healthy margins)

P95 GET /next < 80 ms from cache, < 200 ms from DB

P99 POST /answers < 120 ms (excluding network)

Stream latency < 5 s to aggregations

7) Editorial CMS & Workflows (Engineering details)

State machine for question_versions.state with webhooks:

draft → review → approved → published → retired

Branching: allow new version from any prior version for quick hotfix; on publish, build workers:

render LaTeX + figures

push assets to S3/CDN

snapshot to search index

optionally run lint rules (option balance, absolute qualifiers, forbidden clues)

A/B content rollout: question_publications may include cohort_id for experiments.

8) Search & Assembly Logic (fast filters + quota satisfaction)
Query pattern (fixed quiz generation)

```
WITH pool AS (
  SELECT qv.id AS qv_id, qv.question_id, qv.version, qv.topic_id, qv.tags,
         ic.a, ic.b, ic.c
  FROM question_publications qp
  JOIN question_versions qv
    ON qv.question_id = qp.question_id AND qv.version = qp.live_version
  LEFT JOIN item_calibration ic
    ON ic.question_id = qp.question_id AND ic.version = qp.live_version
  WHERE qp.exam_code = 'USMLE-S1-2025-Q2'
    AND qp.tenant_id = :tenant
    AND qv.state = 'published'
    AND qv.topic_id IN (:topic_ids)
    AND (qv.tags->>'difficulty') IN ('medium','hard')
)
SELECT *
FROM pool
ORDER BY random()
LIMIT 40;
```

For adaptive, fetch a candidate set constrained by blueprint/topic, then pick by max information at user θ.

9) Reporting & Institutional Dashboards

Star schema in warehouse:

fact_responses(user_id, question_id, version, topic_id, is_correct, time_taken_ms, ts, exam_code, tenant_id)

dim_users, dim_topics, dim_questions

Materialize:

Cohort accuracy/time curves

DIF: per cohort/locale/timezone

Item health: exposure, drift, discrimination decay

dbt models produce clean marts → Looker/Mode dashboards.

10) Observability & Quality Gates

SLOs: availability on quiz endpoints; latency P95; error rate < 0.1%

Synthetic canaries: auto-take quizzes hourly in staging/prod

Content linting: static analyzers for item-writing rules (e.g., length balance, negation in lead-in, duplicate distractors)

Event schema contracts: Protobuf/JSON-Schema with CI checks; reject unknown fields at stream edge (or route to DLQ)

11) Security Posture

Token scopes per role; short-lived JWTs; step-up auth for publish actions

Row-level encryption (pgcrypto) for any optional PII in analytics snapshots

Secrets in KMS; CI/CD with OIDC to cloud; image signing for worker fleets

Rate limiting per IP/user for scraping; watermark images; option order shuffling per session

12) Migration Strategy / Lifecycle

Content: append-only versions → no risky migrations there.

Schema: use online migrations (gh-ost/pg-osc) for big tables.

Blue-green publish: move question_publications.live_version in a transaction; warm caches; invalidate by qid.

13) Minimal Viable Build

step 1–3: Postgres schema, authoring UI (Next.js), basic search, S3 asset pipeline

step 4–6: Quiz sessions, answer capture, Redis cache, Kafka events

step 7–9: Analytics ETL to warehouse, first dashboards (CTT stats) step 10–12: IRT nightly job + adaptive engine v1, recommendations v1

step 13–16: Institution reporting, RBAC/RLS hardening, observability & SLOs

14) Common Pitfalls (and fixes)

Item drift after edits: tie psychometrics to version, re-calibrate after significant changes.

Cold-start adaptivity: start with editorial difficulty + topic coverage, quickly blend in IRT as $N \geq 200$ responses.

Overexposed items (leaks): exposure control + rotating forms, distractor edits, watermarking, legal TOS.

PII sprawl: keep PII in a separate service; only user_id in learning tables.

Slow markdown/LaTeX rendering**: pre-render at publish; CDN all assets.

Deliverables:

The SQL DDL above (content, delivery, calibration)

API contracts for quiz lifecycle

Event schema for streaming

Indexing & caching plan

Nightly IRT job spec + storage table

RBAC/RLS examples