CIVIL SERVICES EXAM PREP · AI-ASSISTED

Answering the Question Every UPSC Aspirant Actually Asks: Why This Option?

UPSC-Setu turns 15 years of GS Paper I Previous Year Questions into a structured, searchable knowledge base — using vector search and AI-generated, option-level explanations to show not just the right answer, but the examiner's reasoning behind every option.

1,353	2011–2025	88
PYQ-derived questions covered	years of GS Paper I covered	taxonomy topic codes across 7 subjects

The Problem

Most aspirants treat PYQs as a scored practice bank: attempt, check the key, move on. But UPSC Prelims questions reuse thematic logic — a wrong option (a “distractor”) in one year's paper often becomes the correct premise in a later year. Tracking that repetition across 15 years of papers by hand means notebooks, scattered PDFs, and hundreds of manual hours.

Under the Hood: How the AI Actually Works

UPSC-Setu isn't a single AI feature bolted onto a question bank — it's three distinct AI subsystems working over a shared vector database. Here's what each one actually does.

1. Semantic Search — Finding Questions That “Mean the Same Thing”

Every question and every option is converted into a vector embedding — a list of 1,536 numbers (via OpenAI's text-embedding-3-small model) that represents the statement's meaning as a point in mathematical space. Two questions about, say, the Finance Commission end up as points that sit close together in that space, even if they don't share a single common word.

> **Vector embedding:** *a way of turning text into numbers so a computer can measure how similar two ideas are, not just whether they use the same words.*
>
> **Cosine similarity:** *the specific formula (measuring the angle between two vectors) used to score how close two embeddings are — output as a score from 0 to 1.*
>
> **pgvector:** *a plug-in for PostgreSQL that lets the database store and search these number-lists efficiently, so “find similar questions” runs as a fast database query instead of a slow custom program.*

This is what powers the “Similar Past Questions” panel on every question page — it runs two searches in parallel: a cosine-similarity search over embeddings (catches conceptual overlap) and a shared-topic-code search over the 88-topic taxonomy (catches structural overlap), then merges the results.

2. Theme Deduplication — One Note Per Concept, Not One Per Question

When a new question is ingested, its underlying concepts (e.g. “Finance Commission composition”) are embedded and compared against existing theme nodes already in the database. If the similarity score clears a set threshold, the system links the question to the existing theme instead of generating a duplicate note; if not, a new theme node is created. This retrieve-then-decide step — check what already exists before generating something new — is the same core pattern used in Retrieval-Augmented Generation (RAG) systems, applied here to prevent note sprawl rather than to answer a query.

> **RAG (Retrieval-Augmented Generation):** *an AI pattern where the system first retrieves relevant existing information, then uses that as grounding context for what it generates next — instead of the model generating purely from memory.*

Each theme also accumulates “occurrences” across years — e.g. asked as a true statement in 2018, then flipped into a false distractor in 2021 — with a generated one-line relation summary explaining how the trap changed.

3. Content Generation — Explaining Every Option, Not Just the Answer

A separate offline batch pipeline (run ahead of time, not live per user) sends each question — its statement, all four options, the correct answer, and its official source citation — to GPT with a structured prompt, and asks it to produce an explanation for each option individually: why it's right, or specifically why it's wrong and what trap it represents. A rule-based classification engine (keyword heuristics, with a GPT-4o fallback for ambiguous cases) tags each question by subject and topic code before this stage runs, so the explanation prompt is already grounded in the right context.

4. Trap Detection — Naming the Examiner's Tricks

Distractor options are catalogued against a fixed taxonomy of trap types — extreme qualifiers (“all”, “only”, “none”), numerical swaps, reversed assertions, and institutional misattribution (swapping one constitutional body for a similar one). Tagging traps this way lets the product show a student the pattern, not just this one wrong answer — the same trap type reappears across different years and subjects.

Data Pipeline: From Raw PDF to Structured Databank

Fifteen years of PDF question papers are converted into vector-indexed database rows through a five-stage, resumable ETL pipeline (each stage writes its own intermediate file, so a failed run resumes from the last completed stage instead of restarting).

> **ETL:** *Extract, Transform, Load — the standard pattern for turning messy raw source files into clean, structured database records.*

01

Parse

Raw PDF pages are converted to cached text (pages.json), including a custom two-column parser that resolves side-by-side options like “(a) 1 only (b) 1 and 2”.

02 **Extract**
Cached text is split into individual question-and-answer sections.

03 **Adapt**
Sections are normalized into a consistent schema, one structured record per year.

04 **Load**
Normalized records are inserted into Postgres as question and option rows.

05 **Enrich & Link**
GPT generates explanations, OpenAI generates vector embeddings, and the similarity graph is computed and linked across the whole database.

Practice Test Engine

Custom tests replicate the real UPSC Prelims format rather than a generic quiz: timing runs at 1.2 minutes per question (the official Prelims pace) and pauses automatically if a student navigates away or closes the tab. Scoring follows the exact UPSC marking scheme, and 10% of every generated test is deliberately pulled from the uncategorized general-studies pool to mirror real exam variance rather than only testing well-tagged questions.

```
Correct answer:      +2.00 marks
Incorrect answer:    −0.667 marks  (1/3rd negative marking)
Unattempted:           0.00 marks
```

Monetization Infrastructure

The freemium paywall is enforced at the database layer, not just in the UI — free users get unlimited access to the question bank and explanations, but a hard lifetime cap (5 saves) on bookmarked notes and theme unlocks. The first 500 subscribers lock in a lifetime discounted rate, claimed through an atomic counter so concurrent signups can't oversell the offer:

```
UPDATE founding_counter
SET claimed = claimed + 1
WHERE claimed < 500
RETURNING claimed;
```

Atomic update: *a database operation guaranteed to complete as a single, uninterruptible step — so two people signing up at the exact same millisecond can't both claim the 500th spot.*

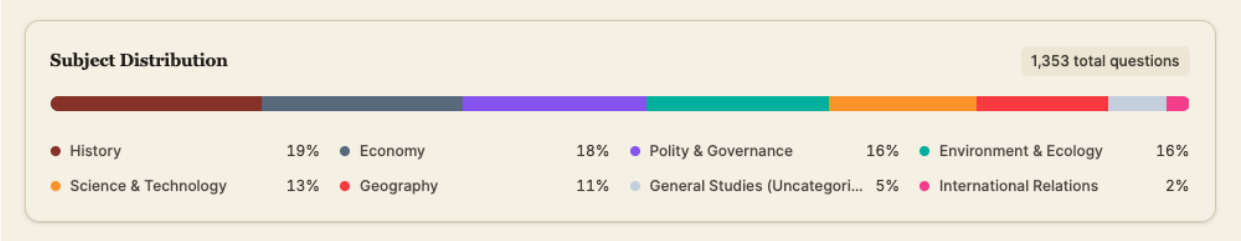
Razorpay webhook events are deduplicated by a unique event ID and ordered using a last-event timestamp, so out-of-order delivery (a known risk with payment webhooks) can't corrupt a user's subscription state. Cancelled subscribers keep permanent read access to notes and unlocks they already made — only new unlocks are blocked.

Technical Stack

Frontend	Next.js 15 (App Router) — React Server Components for fast initial loads, Server Actions for form handling without separate API endpoints
Styling	Tailwind CSS v4, with a distinct “Examiner Mode” (light) and “Student Mode” (dark) theme
Database	PostgreSQL via Supabase, with the pgvector extension for embedding search; Drizzle ORM for type-safe queries
AI / LLM	OpenAI GPT (explanations, classification) and text-embedding-3-small (1,536-dimension embeddings)
Payments	Razorpay Subscriptions, with signature-verified, idempotent webhook processing
Tooling	pnpm workspaces + Turborepo monorepo, with Vitest for parallel test runs across packages

Coverage

The databank spans UPSC Prelims GS Paper I from 2011 to 2025, with 100% syllabus and subject coverage, tagged against an 88-code taxonomy across 7 subjects.



Subject distribution across 1,353 PYQ-derived questions.

Product	UPSC-Setu — AI-assisted UPSC Prelims GS Paper I prep platform
Built by	Setu Labs
Coverage	PYQs from 2011–2025 · 100% syllabus & subject coverage · 88-topic taxonomy
Core AI	OpenAI GPT + text-embedding-3-small, pgvector semantic search
Website	upsc-setu.com