TEST MANTHAN — PRD MODULE 5

Question Bank Architecture

Document Type: Product Requirements Document (Module 5 of 10)
Product: Test Manthan
Parent Company: PsiGenei EdTech Services LLP
Version: 1.0 — Confirmed
Date: February 16, 2026
Depends on: Modules 1-4 (Confirmed), Architectural Context (Confirmed)

5.1 PURPOSE

This module specifies the complete content infrastructure: the master taxonomy, the question data model, the Excel-to-database harvester pipeline, the syllabus map system, and the exam config structure. This is where the UUID/bucket code architecture (confirmed in Architectural Context) gets its full implementation spec.

5.2 MASTER TAXONOMY

Structure

```
Subject (≈22) → Topic (≈8-15 per subject) → Subtopic (≈3-10 per topic)
```

Estimated total: ~22 subjects × ~10 topics × ~5 subtopics = **~1,100 subtopic nodes**

Node Schema

Every node in the taxonomy (subject, topic, or subtopic) follows this schema:

Field	Type	Description	Example
uuid	UUID v4	Internal identifier. Immutable. Used by algorithm.	a3f7b2c1-8d4e-...
bucket_code	Integer	Human-readable numeric alias. Immutable. 1:1 with UUID.	100078
canonical_name	String	Scientific name of the concept.	"G-Protein Coupled Receptors"
level	Enum	stream / subject / topic / subtopic	subtopic
parent_uuid	UUID (nullable)	Parent node's UUID. Null for subjects.	b4e8c3d2-9f5a-...
description	String (optional)	Brief description for internal reference	"GPCRs, 7-TM receptors, G-protein signaling"
created_at	Timestamp	When this node was created	2026-02-15T...

Bucket Code Convention

Numeric, range-based. The digit count identifies the taxonomy level:

Range	Level	Example
100–999	Stream	100 (Life Sciences)
1000–9999	Subject	1001 (Cell Biology)
10000–99999	Topic	10015 (Cell Signaling)
100000–999999	Subtopic	100078 (GPCR)

Rules:

- Integer values only
- Must be globally unique across the entire taxonomy
- Never reused even if a node is deprecated
- Currently one stream (100 = Life Sciences). Future: Chemical Sciences, Physical Sciences, etc.

Taxonomy Integrity Rules

1. Every topic must have exactly one parent subject
2. Every subtopic must have exactly one parent topic

3. No orphan nodes (every non-subject node has a valid parent_uuid)
4. No circular references
5. UUIDs and bucket codes are write-once, never modified or deleted (nodes can be marked deprecated but never removed — questions may reference them)

Sample Taxonomy Fragment

```
100 - Life Sciences [Stream]
└── 1001 - Cell Biology [Subject]
    ├── 10011 - Cell Structure & Organization [Topic]
    │   ├── 100111 - Prokaryotic Cell Structure [Subtopic]
    │   ├── 100112 - Eukaryotic Cell Organization [Subtopic]
    │   └── 100113 - Organelle Structure & Function [Subtopic]
    ├── 10012 - Cell Membrane & Transport [Topic]
    │   ├── 100121 - Membrane Lipid Bilayer [Subtopic]
    │   ├── 100122 - Membrane Proteins [Subtopic]
    │   ├── 100123 - Passive Transport [Subtopic]
    │   └── 100124 - Active Transport & Pumps [Subtopic]
    ├── 10013 - Cell Signaling [Topic]
    │   ├── 100131 - G-Protein Coupled Receptors [Subtopic]
    │   ├── 100132 - Receptor Tyrosine Kinases [Subtopic]
    │   ├── 100133 - Second Messengers [Subtopic]
    │   ├── 100134 - MAPK Signaling Cascade [Subtopic]
    │   ├── 100135 - Wnt Signaling [Subtopic]
    │   ├── 100136 - Notch Signaling [Subtopic]
    │   └── 100137 - JAK-STAT Pathway [Subtopic]
    └── ...
```

5.3 QUESTION DATA MODEL

Question Schema

Field	Type	Required	Description
`question_id`	UUID v4	Yes	Unique identifier
`question_text`	Text (Markdown + LaTeX)	Yes	Question content. Supports Markdown formatting and LaTeX via `$...$` delimiters.
`question_type`	Enum: `MCQ` / `MSQ` / `NAT`	Yes	Question format
`options`	JSON array (nullable)	MCQ/MSQ	Array of option objects: `[{id: "A", text: "...", has_image: false}, ...]`

Field	Type	Required	Description
correct_answer	JSON	Yes	MCQ: "B", MSQ: ["A","C","D"], NAT: {"min": 4.50, "max": 4.62}
explanation	Text (Markdown + LaTeX)	Yes	Teaching-focused explanation. Why correct is correct AND why wrong options are wrong.
subtopic_uuid	UUID (FK)	Yes	The ONE subtopic this question belongs to. Permanent.
topic_uuid	UUID (FK)	Yes	Denormalized parent topic UUID. Set by harvester. For weakness mode performance.
cognitive_level	Enum: recall / conceptual / application / analytical	Yes	Bloom's-aligned cognitive demand
scope_tier	Enum: 1 / 2 / 3 / 4	Yes	Depth tier (see Architectural Context §4)
source_type	Enum: pyq / practice	Yes	Past Year Question or original practice question
source_exam	String (nullable)	PYQ only	e.g., "CSIR-NET", "GATE-BT"
source_year	Integer (nullable)	PYQ only	e.g., 2022
source_session	String (nullable)	PYQ only	e.g., "June", "February"
has_image	Boolean	Yes	Whether question text contains an image
image_refs	JSON array (nullable)	If has_image	Array of image paths: ["questions/q123-fig1.png"]
option_images	JSON object (nullable)	If options have images	{"A": "options/q123-optA.png", "C": "options/q123-optC.png"}
review_status	Enum: reviewed / pending / flagged	Yes	Quality control status. Only reviewed questions appear in tests.
created_at	Timestamp	Yes	When harvested into the database
updated_at	Timestamp	Yes	Last modification
harvester_batch_id	String	Yes	Which Excel batch this came from (traceability)

Content Formatting Rules

Content	In Excel	In Database	In Frontend
Plain text	Plain text	Stored as-is	Rendered as text
Bold/italic	`**bold**` / `*italic*`	Markdown stored	Parsed → HTML
Match-the-column	Markdown table syntax	Markdown stored	Parsed → formatted HTML table component
Subscript/superscript	`H~2~O` / `x^2^`	Markdown stored	Parsed → `<sub>` / `<sup>`
Chemical formulas	`$H_2O$`, `$CO_2$`	LaTeX stored	KaTeX/MathJax renders
Math equations	`$\frac{d[P]}{dt} = k[S]$`	LaTeX stored	KaTeX/MathJax renders
Greek symbols	`$\alpha$`, `$\beta$`	LaTeX stored	KaTeX/MathJax renders
Images	Filename ref: `IMG:q123-fig1.png`	Image path JSON	`<img>` tag, loaded from storage

Database Indexes & Column Strategy

Column Type Decision

Fields that the selection algorithm filters on are stored as **separate columns, not JSON:**

Field	Storage	Why
`subtopic_uuid`	Separate column (UUID)	Filtered every query, needs composite index
`topic_uuid`	Separate column (UUID)	Filtered by weakness mode, needs its own index
`question_type`	Separate column (ENUM)	Filtered every query, fixed values (MCQ/MSQ/NAT)
`cognitive_level`	Separate column (ENUM)	Filtered every query, fixed values (4 levels)
`scope_tier`	Separate column (INTEGER)	Filtered every query, fixed values (1-4)
`source_type`	Separate column (ENUM)	Filtered every query, fixed values (pyq/practice)
`options`	JSON column	Variable structure — MCQ has 4 options, MSQ has 5, NAT has none
`correct_answer`	JSON column	Different shape per type — `"B"` vs `["A","C"]` vs `{"min":4.5,"max":4.6}`
`image_refs`	JSON column	Variable-length array
`option_images`	JSON column	Variable key-value mapping

Rule: Separate columns for everything you filter on. JSON for everything that varies in shape. This gives the fastest possible queries on the critical path AND flexibility where it's actually needed.

Indexing Strategy: Composite Index on Separate Columns

Three indexes total. No more, no less for MVP.

```sql
```

```
-- INDEX 1: The workhorse. Powers every test creation and live counter query.
-- Composite index: all four filter columns in one index, ordered by selectivity.
-- Partial index: only includes reviewed questions (smaller, faster).
CREATE INDEX idx_question_selection
ON questions (subtopic_uuid, question_type, cognitive_level, scope_tier)
WHERE review_status = 'reviewed';

-- INDEX 2: Source filtering (PYQ vs Practice, source transparency display).
CREATE INDEX idx_question_source
ON questions (source_type, source_exam, source_year);

-- INDEX 3: Weakness mode (topic-level accuracy queries).
-- Denormalized topic_uuid column, set by harvester.
CREATE INDEX idx_question_topic
ON questions (topic_uuid);
```

Why Composite Index (Not Multiple Single Indexes)

A composite index on `(subtopic_uuid, question_type, cognitive_level, scope_tier)` works like a phone book sorted by all four fields together. The database jumps directly to the exact combination in one lookup — no scanning, no combining.

With multiple single-column indexes, the database would use one index to narrow down (say, subtopic), then scan those results for the other filters. Slower.

The column order in the composite index is by selectivity (most selective first):

1. `subtopic_uuid` — narrows to ~10-50 questions per subtopic
2. `question_type` — splits further (3 possible values)
3. `cognitive_level` — narrows more (4 possible values)
4. `scope_tier` — final filter (4 possible values)

The `WHERE review_status = 'reviewed'` makes this a partial index — it only includes questions that students can actually see, making the index smaller and faster.

Why Not GIN Index

GIN (Generalized Inverted Index) is for searching inside JSON/arrays/full-text. Since `question_type`, `cognitive_level`, and `scope_tier` are separate columns (not packed into JSON), GIN is not needed for the primary query path. GIN would only be relevant if these fields were stored as JSON — but they're not (see Column Type Decision above).

Performance at Scale

At 7,000-20,000 questions with these three indexes, all queries respond in **single-digit milliseconds**. These indexes become a concern only at millions of rows. The harvester's insert speed is negligibly affected (a few extra milliseconds per row to update indexes).

When to revisit: If queries slow down at 100,000+ questions, run PostgreSQL's `EXPLAIN ANALYZE` on the slow query to identify bottlenecks and add targeted indexes.

5.4 SYLLABUS MAP STRUCTURE

Purpose

One JSON file per exam. Maps the exam's syllabus onto the master taxonomy using bucket codes. Defines what the student sees in the Test Creation Wizard for that exam.

Schema

```json
```

```
{
 "exam_id": "csir-net-ls",
 "exam_name": "CSIR-NET Life Sciences",
 "version": "1.0",
 "last_updated": "2026-02-15",

 "subjects": [
  {
   "display_name": "Cell Biology",
   "bucket_codes_included": [1001],
   "topics": [
    {
     "display_name": "Cell Signaling",
     "bucket_codes_included": [10013],
     "subtopics": [
      {
       "display_name": "G-Protein Coupled Receptors",
       "bucket_codes_included": [100131]
      },
      {
       "display_name": "Receptor Tyrosine Kinases",
       "bucket_codes_included": [100132]
      },
      {
       "display_name": "Second Messengers & Cascades",
       "bucket_codes_included": [100133, 100134]
      }
     ]
    }
   ]
  }
 ]
}
```

How Merging Works

In the example above, CSIR-NET merges `100133` (Second Messengers) and `100134` (MAPK Cascade) into one display unit called "Second Messengers & Cascades."

- **Student sees:** "Second Messengers & Cascades" as one selectable subtopic
- **System resolves:** bucket codes `100133` + `100134` → two UUIDs
- **Algorithm receives:** two separate UUIDs, distributes across both

How Exclusion Works

Any bucket code not referenced in an exam's syllabus map is excluded. No explicit "exclude" field needed. If

CUET-PG's syllabus map doesn't reference `100137`, that subtopic (JAK-STAT Pathway) simply doesn't appear for CUET-PG students.

How Renaming Works

The `display_name` at any level can differ from the `canonical_name` in the master taxonomy. The student sees `display_name`. The database uses the UUID.

Syllabus Map Validation Rules

Before an exam's syllabus map goes live:

1. Every `bucket_codes_included` value must exist in the master taxonomy
2. No bucket code should appear in two different display groups at the same level (no duplicate references)
3. Hierarchy must be consistent: a subtopic's parent topic must be included, and that topic's parent subject must be included
4. At least one subject with at least one topic must be defined

---

5.5 EXAM CONFIG STRUCTURE

Purpose

One JSON file per exam. Governs test-taking behavior, marking scheme, and difficulty mapping. Separate from the Syllabus Map (which governs content scope).

Schema (Complete)

```json
```

```
{
  "exam_id": "csir-net-ls",
  "exam_name": "CSIR-NET Life Sciences",
  "exam_short_name": "CSIR-NET LS",
  "version": "1.0",

  "permissible_tiers": [2, 3, 4],

  "question_types_available": ["MCQ", "MSQ"],

  "marking_scheme": {
    "MCQ": {
      "correct": 2.0,
      "incorrect": -0.5,
      "unanswered": 0
    },
    "MSQ": {
      "all_correct": 2.0,
      "partial_correct": 0,
      "incorrect": 0,
      "unanswered": 0
    },
    "NAT": {
      "correct": 2.0,
      "incorrect": 0,
      "unanswered": 0
    }
  },

  "time_per_question_minutes": {
    "MCQ": 2.0,
    "MSQ": 3.0,
    "NAT": 3.0
  },

  "question_type_distribution": {
    "MCQ": 0.70,
    "MSQ": 0.30
  },

  "difficulty_mapping": {
    "easy": {
      "tier_cognitive_pairs": [
        {"tier": 2, "cognitive": ["recall", "conceptual"]}
      ]
    },
    "medium": {
      "tier_cognitive_pairs": [
        {"tier": 2, "cognitive": ["application", "analytical"]},
        {"tier": 3, "cognitive": ["recall", "conceptual"]}
      ]
    },
    "hard": {
      "tier_cognitive_pairs": [
        {"tier": 3, "cognitive": ["application", "analytical"]},
        {"tier": 4, "cognitive": ["recall", "conceptual", "application", "analytical"]}
      ]
    }
  },

  "display_config": {
    "card_color": "#005059",
    "card_gradient": "teal",
    "icon": "csir-net-icon"
  }
}
```

(Note: `difficulty_mapping` values are illustrative drafts — awaiting founder validation per Module 4 §4.13 Decision 1)

Exam Config Validation Rules

1. All required fields present and correctly typed
2. `permissible_tiers` contains valid tier values (1-4)
3. `question_types_available` is non-empty, values are `MCQ`/`MSQ`/`NAT`
4. `marking_scheme` covers every type in `question_types_available`
5. `question_type_distribution` values sum to 1.0 (±0.01 tolerance)
6. `difficulty_mapping` — all tiers referenced must be within `permissible_tiers`
7. `difficulty_mapping` — all cognitive levels must be valid enum values

Configs Required for MVP

Exam	Tiers	Types	Config Status
CSIR-NET LS	2,3,4	MCQ, MSQ	Draft
GATE-BT	2,3	MCQ, MSQ, NAT	Draft
GATE-XL	2,3	MCQ, MSQ, NAT	Draft
IIT-JAM BT	1,2	MCQ, MSQ, NAT	Draft

Exam	Tiers	Types	Config Status
GAT-B	1,2	MCQ	Draft
CUET-PG	1,2	MCQ	Draft

5.6 EXCEL TEMPLATE & HARVESTER PIPELINE

Excel Sheet Structure

Each Excel sheet represents one batch of questions, typically organized by subject or topic. The content team fills one row per question.

Column Specification

Column	Header	Required	Format	Example
A	`question_text`	Yes	Markdown + LaTeX	`Which of the following is $\alpha$-helix?`
B	`question_type`	Yes	`MCQ` / `MSQ` / `NAT`	`MCQ`
C	`option_a`	MCQ/MSQ	Markdown + LaTeX	`Parallel $\beta$-sheet`
D	`option_b`	MCQ/MSQ	Markdown + LaTeX	`Antiparallel $\beta$-sheet`
E	`option_c`	MCQ/MSQ	Markdown + LaTeX	`3.6 residues per turn helix`
F	`option_d`	MCQ/MSQ	Markdown + LaTeX	`$\pi$-helix`
G	`option_e`	MSQ (optional)	Markdown + LaTeX	(5th option for MSQ)
H	`correct_answer`	Yes	`B` / `A,C,D` / `4.56`	`C`
I	`nat_range_min`	NAT only	Decimal	`4.50`
J	`nat_range_max`	NAT only	Decimal	`4.62`
K	`explanation`	Yes	Markdown + LaTeX	Detailed teaching explanation
L	`subtopic_bucket_code`	Yes	Numeric (6-digit)	`100131`

Column	Header	Required	Format	Example
M	cognitive_level	Yes	recall / conceptual / application / analytical	conceptual
N	scope_tier	Yes	1 / 2 / 3 / 4	2
O	source_type	Yes	pyq / practice	pyq
P	source_exam	PYQ only	Exam name	CSIR-NET
Q	source_year	PYQ only	Year	2022
R	source_session	PYQ only	Session	June
S	image_question	If applicable	Filename	q123-fig1.png
T	image_option_a	If applicable	Filename	q123-optA.png
U-X	image_option_b through image_option_e	If applicable	Filename	(same pattern)

Harvester Script Specification

Input: Excel file path + image folder path
Output: Database records in the questions table
Language: Python (openpyxl for Excel reading)

Processing per row:

```
1. READ row from Excel
2. VALIDATE all required fields present
3. VALIDATE bucket_code exists in master taxonomy → get UUID
4. VALIDATE cognitive_level and scope_tier are valid enum values
5. VALIDATE question_type and answer format match:
   - MCQ: correct_answer is single letter, options A-D present
   - MSQ: correct_answer is comma-separated letters, options present
   - NAT: nat_range_min and nat_range_max are valid decimals
6. VALIDATE image references: if image filenames provided, verify files exist
7. PACK into question record:
   - Generate question_id (UUID v4)
   - Map bucket_code → subtopic_uuid (lookup taxonomy table)
   - Lookup subtopic's parent → topic_uuid (denormalize onto question record)
   - Pack options into JSON array
   - Pack correct_answer into JSON
   - Pack image refs into JSON
   - Set review_status = 'reviewed' (human review already completed in Excel)
   - Set harvester_batch_id = [batch identifier]
8. DEDUPLICATE: Check if question_text already exists (fuzzy match or hash)
   - If duplicate found: skip, log warning
9. INSERT into database
10. LOG: success/skip/error per row
```

Post-run report:

```
Batch: BIOCHEM-2026-02-15
Total rows: 250
Inserted: 237
Skipped (duplicate): 8
Errors: 5
  - Row 34: Missing subtopic_bucket_code
  - Row 67: Invalid cognitive_level "medium" (expected: recall/conceptual/application/analytical)
  - Row 112: Image file "q112-fig1.png" not found
  - Row 189: NAT question missing nat_range_min
  - Row 201: Bucket code "999999" not found in taxonomy
```

Harvester Properties

Property	Requirement
Idempotent	Re-running on the same sheet does not create duplicates (dedup by question text hash + subtopic_uuid)
Validating	Catches all errors before database write. No partial inserts — either a row passes all validation or it's skipped entirely.

Property	Requirement
Atomic per row	Each row is independent. A failure in row 34 does not block row 35.
Traceable	Every question records which batch it came from (`harvester_batch_id`)
Append-only	The harvester never modifies or deletes existing questions. Updates are handled separately via a manual review process.

Image Handling

1. Content team places images in a designated folder alongside the Excel file
2. Harvester validates image files exist
3. Harvester uploads images to storage (S3 / Supabase Storage) in a structured path: `questions/{question_id}/{filename}`
4. Database stores the storage path, not the local filename
5. Frontend loads images via the storage URL

5.7 QUESTION QUALITY PIPELINE

Review Workflow

Review happens BEFORE import, not after. The Excel sheet is the review tool. No separate admin review interface is needed for MVP.

```
Content team writes questions in Excel
        ↓
Subject expert reviews IN THE EXCEL SHEET:
  - Accuracy of content (correct answer verifiably correct)
  - Quality of explanation (teaching-oriented, not just answer-key)
  - Correct tagging (subtopic bucket code, cognitive level, tier)
  - LaTeX/Markdown formatting (renders correctly)
  - Image clarity (referenced images are legible)
        ↓
Expert marks each row: ✅ Ready / ❌ Needs Fix
(Rows marked ❌ are fixed or removed before import)
        ↓
Only ✅ rows remain in the final Excel file
        ↓
Harvester imports with review_status = 'reviewed'
(Human review already completed — all imported questions are live-ready)
        ↓
Harvester validates TECHNICAL issues automatically:
  - Valid bucket code exists in taxonomy?
  - Valid enum values (cognitive level, tier, question type)?
  - Image files exist in storage?
  - Required fields present?
  - Duplicate check (text hash + subtopic_uuid)
        ↓
Questions that pass technical validation → live in student-facing tests
Questions that fail technical validation → logged in error report, skipped
```

Why review-first: Reviewing in Excel is something the team already knows how to do. Building a separate admin review interface would be extra engineering for MVP. The harvester's job is technical validation (correct data types, valid references) — not scientific review (is the answer correct, is the explanation good). Those are separate concerns handled by separate people at separate stages.

Quality Standards

Every question must meet these criteria before being marked `reviewed`:

1. **Accuracy:** Correct answer is verifiably correct. Wrong options are verifiably wrong.
2. **Clarity:** Question text is unambiguous. A subject expert would not disagree on interpretation.
3. **Explanation quality:** Explains WHY the correct answer is correct. Addresses common misconceptions. Teaching-oriented, not just answer-key style.
4. **Tagging accuracy:** Subtopic, cognitive level, and scope tier are correctly assigned.
5. **Formatting:** LaTeX renders correctly. Markdown parses correctly. Images are clear and properly referenced.

6. **Exam pattern alignment:** Question style matches the type of questions asked in competitive exams at the assigned tier level.

Question Count Targets (MVP)

Exam	Minimum (Launch)	Stretch	Status
CSIR-NET LS	2,500	4,000	In progress
GATE-BT	1,500	2,500	In progress
GATE-XL	1,000	1,500	In progress
IIT-JAM BT	1,000	1,500	In progress
GAT-B	500	800	In progress
CUET-PG	500	800	In progress
Total	**7,000**	**11,100**	

Note: These counts are by exam relevance (questions whose subtopic + tier falls within the exam's syllabus map and config). A single question may be relevant to multiple exams.

5.8 PRE-COMPUTED COUNTS TABLE

Purpose

Powers the live preview counter in the Test Creation Wizard (Module 4 §4.8). Must respond within 500ms to any filter change.

MVP Recommendation: Skip This Table, Use Live Queries

At 7,000-20,000 questions with Index 1 (the composite index), a live `SELECT COUNT(*)` with all filters responds in under 100ms. The pre-computed table is a performance optimization that becomes necessary at 50,000+ questions.

For MVP: Use live COUNT queries directly against the questions table. The composite index handles the performance.

When to build this table: When query response time exceeds 200ms on the live counter, or when the question bank exceeds 50,000 questions.

Structure (For When It's Needed)

An aggregation table that pre-computes question counts per filter combination:

```
sql
```

```
CREATE TABLE question_counts (
  subtopic_uuid   UUID,
  question_type   ENUM('MCQ','MSQ','NAT'),
  cognitive_level ENUM('recall','conceptual','application','analytical'),
  scope_tier      INTEGER,
  source_type     ENUM('pyq','practice'),
  count           INTEGER,
  PRIMARY KEY (subtopic_uuid, question_type, cognitive_level, scope_tier, source_type)
);
```

How It Works

When a student adjusts filters in the wizard, the system sums matching rows:

```
sql

SELECT SUM(count) FROM question_counts
WHERE subtopic_uuid IN ([resolved UUIDs from syllabus map])
  AND question_type IN ([selected types])
  AND cognitive_level IN ([selected levels])
  AND scope_tier IN ([exam's permissible tiers])
  AND source_type IN ([selected sources]);
```

This sums maybe 50-100 small rows instead of scanning the full questions table. The table itself is small (~1,100 subtopics × 3 types × 4 levels × 4 tiers × 2 sources = max ~105,600 rows, most absent).

Refresh Trigger

Recalculated after every harvester run. The harvester finishes importing a batch, then rebuilds this table via a simple GROUP BY query on the questions table — takes seconds.

Test Mode Adjustment

This table does NOT account for user question history (Standard/Revision mode). Those require a lightweight per-user query layered on top:

- Standard mode: `total_count - user_seen_count_for_these_filters`
- Revision mode: `user_seen_count_for_these_filters`
- Weakness mode: same as Standard but filtered to weak topic UUIDs

---

5.9 RESOLUTION FLOW (Complete)

Architecture Diagram

The complete flow from student selection to algorithm input:

```
STUDENT UI
  │ Student selects: CSIR-NET → Cell Biology → Cell Signaling
  │ (These are display_names from the CSIR-NET syllabus map)
  ▼
SYLLABUS MAP RESOLUTION
  │ Lookup: CSIR-NET syllabus map
  │ "Cell Signaling" → bucket codes:
  │   [100131, 100132, 100133, 100134, 100135, 100136, 100137]
  ▼
BUCKET CODE → UUID RESOLUTION
  │ Lookup: taxonomy table
  │ 100131 → UUID-001
  │ 100132 → UUID-002
  │ ... etc.
  ▼
EXAM CONFIG LOADING
  │ Load: csir-net-ls config
  │ Permissible tiers: [2, 3, 4]
  │ Types: [MCQ, MSQ]
  │ Difficulty mapping: (as configured)
  ▼
QUESTION SELECTION ALGORITHM (Module 4)
  │ Receives ONLY:
  │   - Subtopic UUIDs: [UUID-001, UUID-002, ...]
  │   - Permissible tiers: [2, 3, 4]
  │   - Question types: [MCQ, MSQ]
  │   - Cognitive levels: [user-selected]
  │   - Difficulty mapping: (tier×cognitive → easy/med/hard)
  │   - Question count: N
  │   - Test mode: Standard/Revision/Weakness
  │   - User question history (for mode filtering)
  │
  │ Does NOT receive: exam name, bucket codes, display names
  ▼
TEST OBJECT → Test-Taking Interface (Module 6)
```

Detailed Walkthrough: What Actually Happens

A CSIR-NET student named Priya opens the Test Creation Wizard.

STEP 1 — She selects CSIR-NET LS + Standard mode

Behind the scenes (she sees nothing):

- System loads `csir-net-ls` exam config JSON → knows: permissible tiers [2,3,4], question types [MCQ,MSQ], marking scheme, difficulty mapping, time-per-question

- System loads `csir-net-ls` syllabus map JSON → knows which subjects/topics/subtopics to show her in Step 2
- Test mode = Standard → will exclude her previously seen questions

STEP 2 — She selects Cell Biology → Cell Signaling → picks 3 subtopics

What she sees in the UI: "GPCR", "Second Messengers & Cascades", "MAPK Signaling"

Behind the scenes:

```
"GPCR"                       → bucket code 100131 → UUID-001
"Second Messengers & Cascades" → bucket codes [100133, 100134] → [UUID-003, UUID-004]
"MAPK Signaling"             → bucket code 100134 → (already in set from merge above)
```

"Second Messengers & Cascades" is a merged display group containing 100133 + 100134. She also selected "MAPK Signaling" which is 100134. The system deduplicates. Final UUID set: `[UUID-001, UUID-003, UUID-004]`.

The live counter fires instantly (using the composite index):

```sql
SELECT COUNT(*) FROM questions
WHERE subtopic_uuid IN (UUID-001, UUID-003, UUID-004)
  AND scope_tier IN (2, 3, 4)
  AND review_status = 'reviewed';
```

→ Returns 87. Sidebar shows: "Available: 87 questions 🟢 "

STEP 3 — She sets her filters

She selects:

- Question types: MCQ + MSQ ✅
- Cognitive levels: Application + Analytical only (unchecks Recall, Conceptual)
- Difficulty: Easy 20% / Medium 50% / Hard 30%
- Source: PYQ + Practice
- Questions: 15
- Duration: auto-calculated → 15 × 2.5 min avg = ~38 minutes

Counter recalculates with the additional filters:

```sql
```

```
SELECT COUNT(*) FROM questions
WHERE subtopic_uuid IN (UUID-001, UUID-003, UUID-004)
  AND question_type IN ('MCQ', 'MSQ')
  AND cognitive_level IN ('application', 'analytical')
  AND scope_tier IN (2, 3, 4)
  AND source_type IN ('pyq', 'practice')
  AND review_status = 'reviewed';
```

→ Returns 34. Sidebar updates: "Available: 34 questions 🟢 "

SHE CLICKS "CREATE TEST"

The selection algorithm runs (Module 4). It receives ONLY UUIDs and attributes — no exam name, no bucket codes, no display names:

```
1. Input UUIDs: [UUID-001, UUID-003, UUID-004]
   Test mode: Standard

2. Standard mode filter → query her history:
   She's seen 5 questions from these UUIDs before.
   Candidate pool: 34 - 5 = 29 unseen questions.

3. Distribute 15 questions across 3 UUIDs (equal = 5 each):
   UUID-001 (GPCR):              11 available → allocate 5 ✅
   UUID-003 (Second Messengers):  4 available → allocate 4 (short by 1)
   UUID-004 (MAPK):              14 available → allocate 6 (absorbs extra 1)
   Fallback Level 1 triggered: redistribution within selected topics only.

4. Within each allocation, apply difficulty distribution:
   Easy 20% = 3 questions → (Tier 2 + Application per exam config mapping)
   Medium 50% = 8 questions → (Tier 3 + Application, Tier 2 + Analytical)
   Hard 30% = 4 questions → (Tier 3/4 + Analytical)

5. Apply question type distribution (exam config: MCQ 70%, MSQ 30%):
   Target: ~10 MCQ, ~5 MSQ
   Adjusted based on what's actually available in each bucket.

6. Random select within each bucket.

7. Assemble test object:
   - 15 questions, randomized order
   - Exam config attached (marking scheme, time)
   - Test ID generated
   - Saved to database

8. Sidebar shows: "ℹ️ Some adjustments were made based on available questions."
   (Because UUID-003 was short by 1 and redistribution occurred)
```

Priya enters the test-taking interface with her 15-question custom test.

Every step above happens in under 1 second. She clicks "Create Test" and is immediately in the test.

5.10 FOUNDER DECISIONS (Confirmed February 16, 2026)

Decision 1 — Bucket Code Convention: NUMERIC RANGE-BASED

Decision: Bucket codes are numeric integers, with the range determining the taxonomy level:

Range	Level	Capacity	Example
100–999	Stream	900	Life Sciences = 100
1000–9999	Subject	9,000	Cell Biology = 1001
10000–99999	Topic	90,000	Cell Signaling = 10015
100000–999999	Subtopic	900,000	GPCR = 100078

The digit count instantly identifies the level: 3-digit = stream, 4-digit = subject, 5-digit = topic, 6-digit = subtopic. Currently only one stream (Life Sciences = 100). The range system is built for future expansion into Chemical Sciences, Physical Sciences, etc.

Content team uses these numeric bucket codes in Excel sheets. The harvester resolves them to UUIDs.

Decision 2 — Denormalized topic_uuid: YES (Via Harvester)

Decision: Excel sheets store only the subtopic bucket code. The harvester resolves: bucket code → subtopic UUID → parent topic UUID. Both `subtopic_uuid` and `topic_uuid` are written to the questions table in the database.

This means:

- Content team only enters one ID per question (the subtopic bucket code)
- The harvester does the lookup and denormalization automatically
- Weakness mode queries (which operate at topic level per Module 4) can filter directly on `topic_uuid` without joining through the taxonomy table

Decision 3 — Question Deduplication: ONE RECORD

Decision: One record per question, regardless of how many exams it appeared in. If a GATE-2019 question also appeared in a coaching compilation or another exam, it is stored once, tagged with its original source. Multi-exam relevance is handled entirely by the syllabus map + tier system — not by duplicating questions.

Decision 4 — Review Team: 6 PEOPLE

Decision: Review team consists of 6 members: Shabab, Sabiha, Aliya, Aliza, Kulsum, and 1 intern. At ~50 questions per person per day, throughput is ~300 questions/day, meaning 7,000 launch questions require approximately 23 working days of review.

Decision 5 — Excel Sheet Organization: BATCH MODE

Decision: No fixed organization by subject or topic. Sheets are organized by batch — each batch is a working unit produced by the content team. The `harvester_batch_id` tracks provenance. A single batch may contain questions across multiple subjects/topics.

Module 5 complete. All decisions confirmed. Proceeding to Module 6: Test-Taking Interface.