TEST MANTHAN — PRD MODULE 4

Test Creation Engine (Detailed Technical Spec)

Document Type: Product Requirements Document (Module 4 of 10)
Product: Test Manthan
Parent Company: PsiGenei EdTech Services LLP (Brand: PsiGenei)
Version: 1.0 — Draft for Founder Review
Date: February 15, 2026
Depends on: Module 1 (Confirmed), Module 2 (Confirmed), Module 3 (Confirmed)

4.1 PURPOSE OF THIS MODULE

Module 3 defined WHAT the Test Creation Wizard does from the user's perspective. Module 4 defines HOW it works under the hood. This is the engineering blueprint for:

- The question selection algorithm (the brain of the product)
- Difficulty derivation from Tier + Cognitive Level
- Distribution logic across topics and question types
- Fallback waterfall when ideal distribution is impossible
- Test mode implementation (Standard / Revision / Weakness)
- The live preview counter system
- Template save/load mechanics

This module is written for the engineering team. Every decision here has a direct impact on what the student experiences.

4.2 SYSTEM OVERVIEW: HOW A TEST GETS CREATED

Here is the complete flow from the moment a user clicks "Create Test" to the moment questions appear in the test-taking interface:

User clicks "Create Test" in the wizard

↓

[1] COLLECT PARAMETERS

Gather all user selections:

- Exam ID → loads Exam Config JSON
- Test Mode (Standard / Revision / Weakness)
- Subject ID
- Topic IDs (one or more)
- Subtopic IDs (one or more, or "all" within selected topics)
- Question Source (PYQ / Practice / Both)
- Question Types (MCQ / MSQ / NAT — subset of what exam config allows)
- Cognitive Levels (Recall / Conceptual / Application / Analytical)
- Difficulty Distribution (Easy % / Medium % / Hard %)
- Question Count (N)
- Duration (auto-calculated or manual)

↓

[2] RESOLVE SYLLABUS MAP

Using the Exam's Syllabus Map JSON:

- Map selected Topic IDs → master taxonomy Topic IDs
- Map selected Subtopic IDs → master taxonomy Subtopic IDs
- Handle merged subtopics: if the exam merges subtopics 4201+4202 into "Classical Genetics," resolve to both underlying IDs
- Handle exclusions: if a subtopic is excluded from this exam's syllabus map, it's already invisible in the wizard — no resolution needed here

↓

[3] RESOLVE PERMISSIBLE TIERS

From the Exam Config JSON:

- Get the list of permissible Scope Tiers for this exam
- Example: CSIR-NET → [Tier 2, Tier 3, Tier 4]
- Example: CUET-PG → [Tier 1, Tier 2]

↓

[4] MAP DIFFICULTY DISTRIBUTION TO TIER + COGNITIVE LEVEL

Using the Exam Config's difficulty mapping rules:

- Convert user's Easy/Medium/Hard percentages into specific Tier + Cognitive Level combinations
- (Detailed in Section 4.4)

↓

[5] BUILD THE QUERY

Construct database query with ALL resolved filters:

- Subtopic IDs (from resolved syllabus map)
- Scope Tiers (from exam config)
- Question Types (user-selected, validated against exam config)
- Cognitive Levels (user-selected)
- Source (PYQ / Practice / Both)
- Test Mode filter (from User Question History — detailed in 4.5)

```
    ↓
[6] EXECUTE QUERY → GET CANDIDATE POOL
    Returns all questions matching the filters.
    This is the "candidate pool" from which we'll select.
    ↓
[7] APPLY DISTRIBUTION LOGIC
    Distribute N questions across:
    - Topics (equal distribution, with fallback)
    - Difficulty buckets (Easy/Medium/Hard per user preference)
    - Question types (per exam config distribution rules)
    - (Detailed in Section 4.6)
    ↓
[8] APPLY FALLBACK LOGIC (if needed)
    If ideal distribution is impossible:
    - Redistribute from over-represented buckets
    - Reduce count if critically low
    - Warn user if any significant deviation
    - (Detailed in Section 4.7)
    ↓
[9] RANDOM SELECT WITHIN EACH BUCKET
    For each distribution bucket, randomly select
    the required number of questions from candidates.
    ↓
[10] ASSEMBLE TEST
    - Compile selected questions into a test object
    - Randomize question order (default)
    - Attach exam config (marking scheme, time rules)
    - Generate test ID
    - Save test metadata to database
    - Return test object to test-taking interface
    ↓
User enters the test-taking interface with their custom test.
```

4.3 THE EXAM CONFIG JSON (Structure)

Each exam has a JSON config file that governs how Test Manthan behaves for that exam. This config is loaded the moment a student selects an exam in Step 1 of the wizard.

Config Structure

```
json
```

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

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

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

  "marking_scheme": {
    "MCQ": {
      "correct": 2.0,
      "incorrect": -0.5,
      "unanswered": 0
    },
    "MSQ": {
      "correct": 2.0,
      "incorrect": 0,
      "partial": 0,
      "unanswered": 0
    }
  },

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

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

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

```
    }
  },

  "syllabus_map_ref": "csir-net-ls-syllabus-map.json",

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

Exam Configs Required for MVP

Exam	ID	Tiers	Question Types	Notes
CSIR-NET LS	`csir-net-ls`	2, 3, 4	MCQ, MSQ	Part A/B/C distinction handled via tier + cognitive level
GATE-BT	`gate-bt`	2, 3	MCQ, MSQ, NAT	NAT is a key differentiator for GATE
GATE-XL	`gate-xl`	2, 3	MCQ, MSQ, NAT	Similar structure to GATE-BT, different syllabus map
IIT-JAM BT	`iit-jam-bt`	1, 2	MCQ, MSQ, NAT	MSQ and NAT are the "hard stuff" for JAM
GAT-B	`gat-b`	1, 2	MCQ	Simpler exam format
CUET-PG	`cuet-pg`	1, 2	MCQ	Simpler exam format

Config Validation Rules

Before an exam goes live on the platform, its config must pass validation:

- All fields present and correctly typed
- `permissible_tiers` contains at least one valid tier (1-4)
- `question_types_available` is non-empty and contains only valid types
- `marking_scheme` covers every type in `question_types_available`
- `question_type_distribution` values sum to 1.0
- `difficulty_mapping` tiers are all within `permissible_tiers`
- Referenced `syllabus_map_ref` file exists and is valid

4.4 DIFFICULTY DERIVATION: TIER + COGNITIVE LEVEL → EASY/MEDIUM/HARD

The Problem

Students think in "Easy / Medium / Hard." But difficulty is subjective and exam-relative. A question that's "Hard" for a CUET student might be "Easy" for a CSIR-NET student. We cannot tag each question with a difficulty per exam — that doesn't scale.

The Solution

Difficulty is **computed, not tagged.** It's derived from two independent, objective attributes that ARE tagged on every question:

Scope Tier (1-4): How deep into the subject matter does this question go?

- Tier 1: UG Foundation — basic concepts, definitions, straightforward recall
- Tier 2: UG Advanced — deeper conceptual understanding, multi-step reasoning
- Tier 3: PG Level — postgraduate depth, experimental design, cross-topic integration
- Tier 4: PG Advanced — highest complexity, CSIR-NET Part C level, research-level thinking

Cognitive Level (4 levels): What type of thinking does this question require?

- Recall: Remember facts, definitions, terms
- Conceptual: Understand mechanisms, processes, relationships
- Application: Apply knowledge to new scenarios, experiments, problems
- Analytical: Analyze data, evaluate hypotheses, synthesize across topics

The Mapping

Each exam config defines how Tier + Cognitive Level combinations map to Easy / Medium / Hard FOR THAT EXAM.

Example: CSIR-NET LS

	Recall	**Conceptual**	**Application**	**Analytical**
Tier 2	Easy	Easy	Medium	Medium
Tier 3	Easy	Medium	Medium	Hard
Tier 4	Medium	Medium	Hard	Hard

Example: IIT-JAM BT

	Recall	Conceptual	Application	Analytical
Tier 1	Easy	Easy	Medium	Hard
Tier 2	Medium	Medium	Hard	Hard

Example: CUET-PG

	Recall	Conceptual	Application	Analytical
Tier 1	Easy	Easy	Medium	Medium
Tier 2	Medium	Hard	Hard	Hard

How the Difficulty Slider Works

When a student sets the difficulty slider to, say, Easy 30% / Medium 50% / Hard 20%:

1. The system looks up the exam config's `difficulty_mapping`
2. For "Easy 30%": It identifies all Tier + Cognitive Level combinations that map to "Easy" for this exam
3. For "Medium 50%": Same for "Medium"
4. For "Hard 20%": Same for "Hard"
5. These combinations become additional filters on the candidate pool
6. The selection algorithm tries to pull 30% of N questions from the "Easy" bucket, 50% from "Medium," 20% from "Hard"

Why This Is Better Than Static Difficulty Tags

- **Scales automatically:** Add a new exam → define its difficulty mapping → done. No re-tagging of questions.
- **Objectively grounded:** Tier and Cognitive Level are properties of the question's content, not subjective opinions about "how hard" it is.
- **Exam-relative:** The same question is correctly calibrated as "Easy" for an advanced exam and "Hard" for a basic exam, without any manual override.
- **Two-dimensional:** A Tier 4 + Recall question (remembering an obscure fact) is different from a Tier 2 + Analytical question (analyzing a basic experiment). Both might be "Medium" for CSIR-NET, but for different reasons. The system captures this nuance.

4.5 TEST MODE IMPLEMENTATION

Standard Mode

Objective: Serve questions the user has NOT seen before.

Query modification:

```
WHERE question_id NOT IN (
  SELECT question_id FROM user_question_history
  WHERE user_id = [current_user]
)
```

Edge case — Pool exhaustion: If the filtered candidate pool after excluding seen questions has fewer questions than requested:

- If remaining ≥ 5: Allow test creation with reduced count. Show warning: "Only [X] unseen questions match your filters. Creating test with [X] questions."
- If remaining < 5: Block test creation. Show message: "You've attempted nearly all questions matching these filters. Try Revision mode to re-practice, or broaden your topic/subtopic selection."

Revision Mode

Objective: Serve questions the user HAS seen before, for re-practice.

Query modification:

```
WHERE question_id IN (
  SELECT question_id FROM user_question_history
  WHERE user_id = [current_user]
)
```

Additional sort preference: Prioritize questions the user got WRONG in previous attempts (most recent attempt). This makes Revision mode naturally focus on errors.

Edge case — No history: If user has no question history (new user), Revision mode is disabled in the wizard with tooltip: "Take your first test to unlock Revision mode."

Weakness Mode

Objective: Serve questions from topics/subtopics where the user's historical accuracy is below threshold.

Implementation (two-step):

Step 1: Identify weak areas

```
SELECT subtopic_id,
    COUNT(CASE WHEN is_correct THEN 1 END) as correct,
    COUNT(*) as total,
    (correct / total) as accuracy
FROM user_question_history
WHERE user_id = [current_user]
GROUP BY subtopic_id
HAVING accuracy < 0.60
```

Step 2: Filter questions to those weak subtopics

```
WHERE subtopic_id IN ([weak_subtopic_ids])
```

Weakness mode ignores the user's topic/subtopic selection in Step 2 of the wizard. Instead, it auto-selects weak subtopics within the selected subject. The wizard shows which subtopics are being targeted: "Weakness mode detected low accuracy in: Cell Signaling (38%), Apoptosis (45%), MAPK Cascade (52%)"

Weakness mode can include both seen and unseen questions. The priority is targeting the weak area, not whether the question is fresh.

Edge case — No weak areas: If all subtopics have accuracy ≥ 60% (or user has insufficient history to calculate): "No critical weaknesses detected! Your accuracy is above 60% across all tested topics. Try Standard mode for continued practice."

Edge case — Weak areas outside selected subject: Weakness mode operates within the subject selected in Step 2. If the user selects "Cell Biology" but their weaknesses are in "Genetics," the system only shows Cell Biology weaknesses. If there are none in Cell Biology, show: "No weaknesses detected in Cell Biology. Your accuracy is strong here. Try selecting a different subject or use Standard mode."

Weakness Threshold

The 60% accuracy threshold is a starting default. It should be configurable (admin-level setting, not user-facing) so it can be tuned based on real data post-launch. If most users are below 60% in most topics, the threshold is too high and Weakness mode becomes indistinguishable from Standard mode.

4.6 DISTRIBUTION LOGIC

When the system has a candidate pool of matching questions, it needs to select exactly N questions distributed across three dimensions:

1. **Topics** (equal distribution across selected topics)
2. **Difficulty** (matching the user's Easy/Medium/Hard slider)
3. **Question Types** (matching the exam config's type distribution)

Priority Order

These three dimensions can conflict. The priority order for resolution is:

Topic distribution > Difficulty distribution > Question type distribution

Rationale:

- Topic balance is the user's primary intent (they selected specific topics for a reason)
- Difficulty balance is their secondary preference (they want a mix, but it's a preference not a requirement)
- Question type balance is the least critical (they'd rather get the right topics at the right difficulty than a perfect MCQ/MSQ/NAT ratio)

Topic Distribution Algorithm

Input: User selected K topics. Requested N questions total.

Step 1: Calculate ideal per-topic count

```
ideal_per_topic = floor(N / K)
remainder = N mod K
```

Step 2: Check availability per topic For each topic, count available questions in the candidate pool (after all filters including test mode).

Step 3: Allocate

```
for each topic (sorted by available questions, ascending):
    allocated = min(ideal_per_topic, available_for_this_topic)
    if topic is in the first 'remainder' topics:
        allocated = min(allocated + 1, available_for_this_topic)
    shortfall += (ideal_per_topic - allocated)  // if any

// redistribute shortfall to topics with surplus
for each topic with available > allocated (sorted by surplus, descending):
    give_extra = min(shortfall, available - allocated)
    allocated += give_extra
    shortfall -= give_extra
```

Result: Each topic gets as close to equal as the pool allows, with surplus topics absorbing the shortfall from thin topics.

Difficulty Distribution Algorithm

Input: User's slider values (e.g., Easy 30%, Medium 50%, Hard 20%). Total N questions.

Step 1: Calculate ideal per-difficulty count

```
easy_target = round(N * easy_pct)
medium_target = round(N * medium_pct)
hard_target = N - easy_target - medium_target  // ensures sum = N
```

Step 2: Within each topic's allocation, distribute by difficulty For each topic's allocated questions, attempt to maintain the same Easy/Medium/Hard ratio.

Step 3: If a difficulty bucket is short within a topic Pull from adjacent difficulty: if Hard is short, pull from Medium. If Medium is short, split between Easy and Hard.

Question Type Distribution Algorithm

Input: Exam config's `question_type_distribution` (e.g., MCQ 70%, MSQ 30%). Total N questions.

This is applied AFTER topic and difficulty allocation as a soft constraint:

- Attempt to maintain the ratio across the full test
- If a specific type is unavailable for certain topics/difficulties, redistribute to available types
- This is the lowest priority — never block test creation because the MCQ/MSQ ratio isn't perfect

---

4.7 FALLBACK WATERFALL

When the ideal distribution is impossible, the system follows this waterfall, step by step. Each step is only triggered if the previous step couldn't fully resolve the shortfall.

LEVEL 0: IDEAL STATE
All N questions distributed perfectly across topics, difficulty, and types.
→ Proceed to test creation. No warnings.

↓ (if not possible)

LEVEL 1: REDISTRIBUTE ACROSS TOPICS
Some topics don't have enough questions.
Over-represent topics with deeper banks.
→ Proceed. Sidebar note: "Topic distribution adjusted —
[Topic X] had fewer questions than requested."

↓ (if still short)

LEVEL 2: RELAX DIFFICULTY DISTRIBUTION
Can't match Easy/Medium/Hard ratio.
Pull from adjacent difficulty buckets.
→ Proceed. Sidebar note: "Difficulty distribution adjusted —
not enough Hard questions for selected topics."

↓ (if still short)

LEVEL 3: RELAX QUESTION TYPE DISTRIBUTION
Can't match MCQ/MSQ/NAT ratio.
Fill with whatever types are available.
→ Proceed. Sidebar note: "Question type mix adjusted —
fewer NAT questions available than ideal."

↓ (if still short)

LEVEL 4: REDUCE QUESTION COUNT
Total available (after all relaxations) < N.
→ If available ≥ 5: Create test with reduced count.
Show warning: "Only [X] questions match your filters.
Creating test with [X] questions. Broaden your filters
for more questions."
→ If available < 5: Block test creation.
Show message: "Not enough questions for a meaningful test.
Please select more topics or adjust your filters."
"Create Test" button disabled.

Fallback Messages in the Live Preview Sidebar

The sidebar should reflect fallback state in real-time as the user adjusts filters:

State	Sidebar Display
Everything ideal	"✅ Perfect — [N] questions ready across [K] topics"
Topic imbalance	"⚠️ [Topic X] has fewer questions — distribution adjusted"
Difficulty adjusted	"⚠️ Difficulty mix adjusted — limited Hard questions in your selection"
Type adjusted	"ℹ️ Question type mix adjusted for availability"
Reduced count	"⚠️ Only [X] questions available. Test will have [X] questions instead of [N]."
Critically low	"❌ Not enough questions. Select more topics or adjust filters."

These messages are NOT errors — they're transparent communication about what the system is doing. Students should trust that the system is giving them the best possible test from the available pool.

4.8 LIVE PREVIEW COUNTER: HOW IT WORKS

The "Available Questions" counter in the sidebar must update in real-time as the user changes any filter in any step. This is one of the product's key trust signals.

When It Updates

User Action	Counter Recalculates Based On
Selects exam (Step 1)	Exam's syllabus map + permissible tiers
Selects test mode (Step 1)	+ User question history filter
Selects subject (Step 2)	+ Subject filter
Selects/deselects topics (Step 2)	+ Topic filter (live, per toggle)
Selects/deselects subtopics (Step 2)	+ Subtopic filter
Toggles PYQ/Practice (Step 3)	+ Source filter
Toggles question types (Step 3)	+ Question type filter
Toggles cognitive levels (Step 3)	+ Cognitive level filter
Changes difficulty slider (Step 3)	+ Difficulty bucket filter (via tier+cognitive mapping)

Performance Requirement

The counter must update within **500ms** of any filter change. This means:

Option A: Pre-computed counts (Recommended for MVP)

- Maintain a counts table that stores question counts per combination of: subtopic_id × question_type × cognitive_level × scope_tier × source
- When filters change, sum the relevant rows from the counts table (fast aggregation, no full-table scan)
- Counts table is refreshed whenever the question bank is updated (via harvester script)
- For test mode adjustments (Standard/Revision), subtract the user's seen-question count per relevant dimensions

Option B: Live database queries

- Direct COUNT queries with all filters applied
- Simpler to implement but slower at scale
- Acceptable for MVP if the question bank is under 20,000 questions and indexed properly
- May need to move to Option A as the bank grows

What the Counter Shows

Single number: "Available: 247 questions"

Color coding:

- 🔴 Red (< 10): "Too few questions. Broaden your filters."
- 🟡 Yellow (10-50): "Good for quick practice"
- 🟢 Green (50-200): "Great range available"
- 🟢 Green (200+): "Plenty of questions"

This number represents the total pool BEFORE distribution logic. It tells the student "there are 247 questions that match your filters." The actual test may pull 15-30 from this pool. The number builds confidence that the bank has depth.

4.9 TEMPLATE SAVE / LOAD

Save Template

Trigger: User clicks "Save as Template" link in the sidebar (Pro/Elite only).

What gets saved:

```
json
```

```
{
  "template_id": "auto-generated UUID",
  "user_id": "user-123",
  "template_name": "CSIR-NET Cell Signaling Hard",  // user can edit
  "created_at": "2026-02-15T14:30:00Z",
  "config": {
    "exam_id": "csir-net-ls",
    "test_mode": "standard",
    "subject_id": "sub-005",
    "topic_ids": ["top-041", "top-042"],
    "subtopic_ids": ["stop-201", "stop-205", "stop-208"],
    "question_source": "both",
    "question_types": ["MCQ", "MSQ"],
    "cognitive_levels": ["application", "analytical"],
    "difficulty_distribution": {"easy": 0.10, "medium": 0.40, "hard": 0.50},
    "question_count": 20,
    "duration_minutes": 40
  }
}
```

What does NOT get saved:

- Specific questions (each test from a template gets a fresh random selection)
- User's answer history
- Available question count (recalculated on load)

Load Template

Where: Accessible from Test History → Saved Templates tab, or from a "My Templates" shortcut on the dashboard.

Flow:

1. User clicks "Use Template" on a saved template
2. Test Creation Wizard opens with all fields pre-filled from the template config
3. The live preview sidebar recalculates the available question count (may differ from when template was saved, if questions have been added to the bank)
4. User can modify any field before creating the test
5. "Create Test" works exactly as normal

Template Limits

- Free: No templates
- Pro: 5 saved templates

- Elite: 20 saved templates
- At limit: "You've reached your template limit. Delete an existing template to save a new one."

4.10 AUTO-GENERATED TEST NAME

Format

```
[Exam Short Name] • [Subject or Topic Names] • [Mode] Test • [Date]
```

Rules

Condition	Name Format	Example
1 topic selected	`[Exam] • [Topic] • [Mode] Test • [Date]`	"CSIR-NET • Cell Signaling • Standard Test • 15 Feb"
2-3 topics selected	`[Exam] • [Topic1], [Topic2] • [Mode] Test • [Date]`	"GATE-BT • Genetics, Immunology • Standard Test • 15 Feb"
4+ topics selected	`[Exam] • [Subject] (X topics) • [Mode] Test • [Date]`	"IIT-JAM • Cell Biology (5 topics) • Weakness Test • 15 Feb"
All topics in subject selected	`[Exam] • [Subject] (All topics) • [Mode] Test • [Date]`	"CUET-PG • Microbiology (All topics) • Revision Test • 15 Feb"

- Date format: DD Mon (e.g., "15 Feb")
- User can edit the name before or after creation
- Name is used in test history and analytics for identification

---

4.11 DURATION CALCULATION

Auto-Calculate (Default)

```
duration_minutes = Σ (count_per_type × time_per_question_for_type)
```

Where `time_per_question_for_type` comes from the exam config's `time_per_question` field.

Example: CSIR-NET, 15 questions (10 MCQ + 5 MSQ):

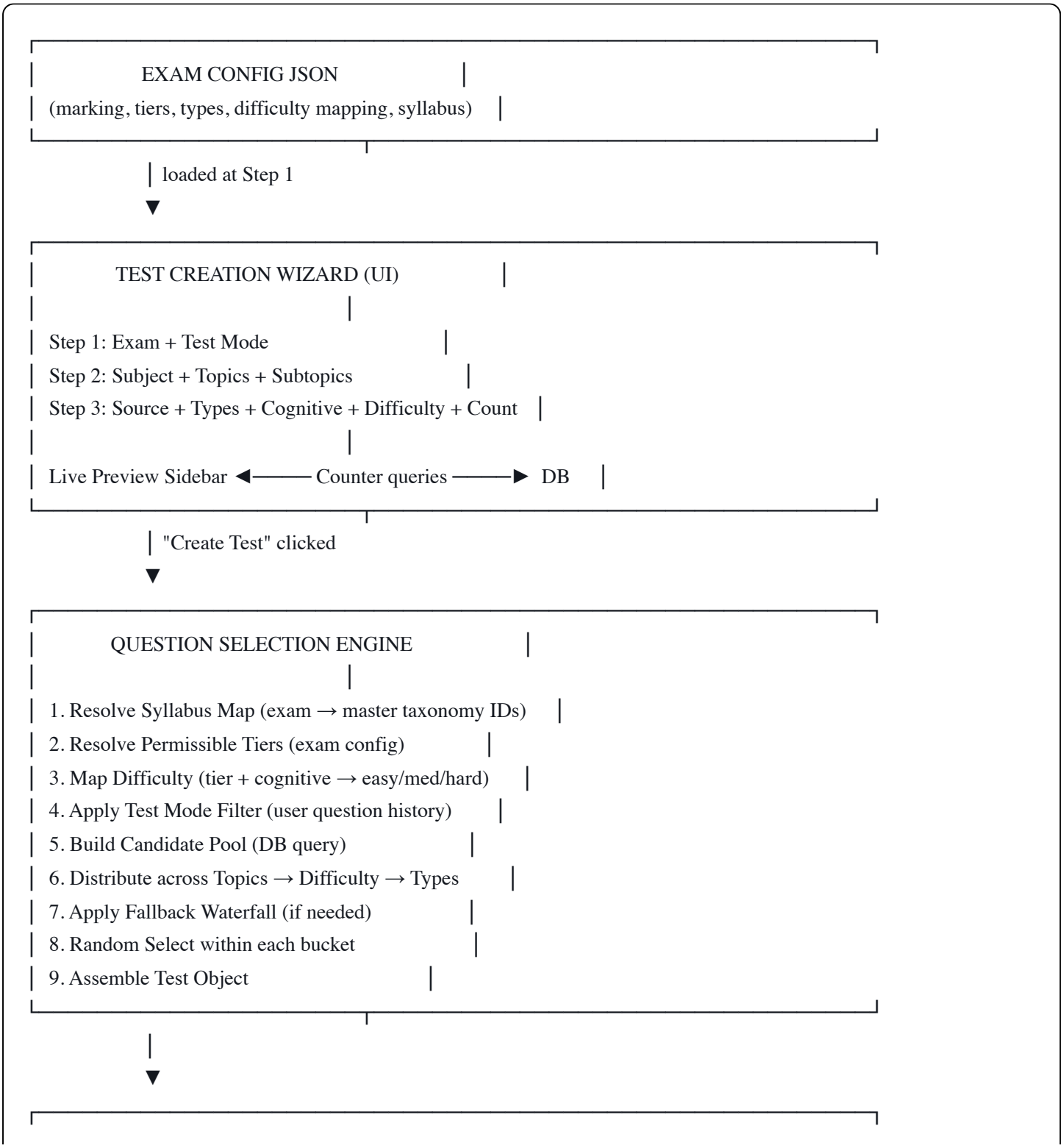
```
= (10 × 2.0 min) + (5 × 3.0 min) = 35 minutes
```

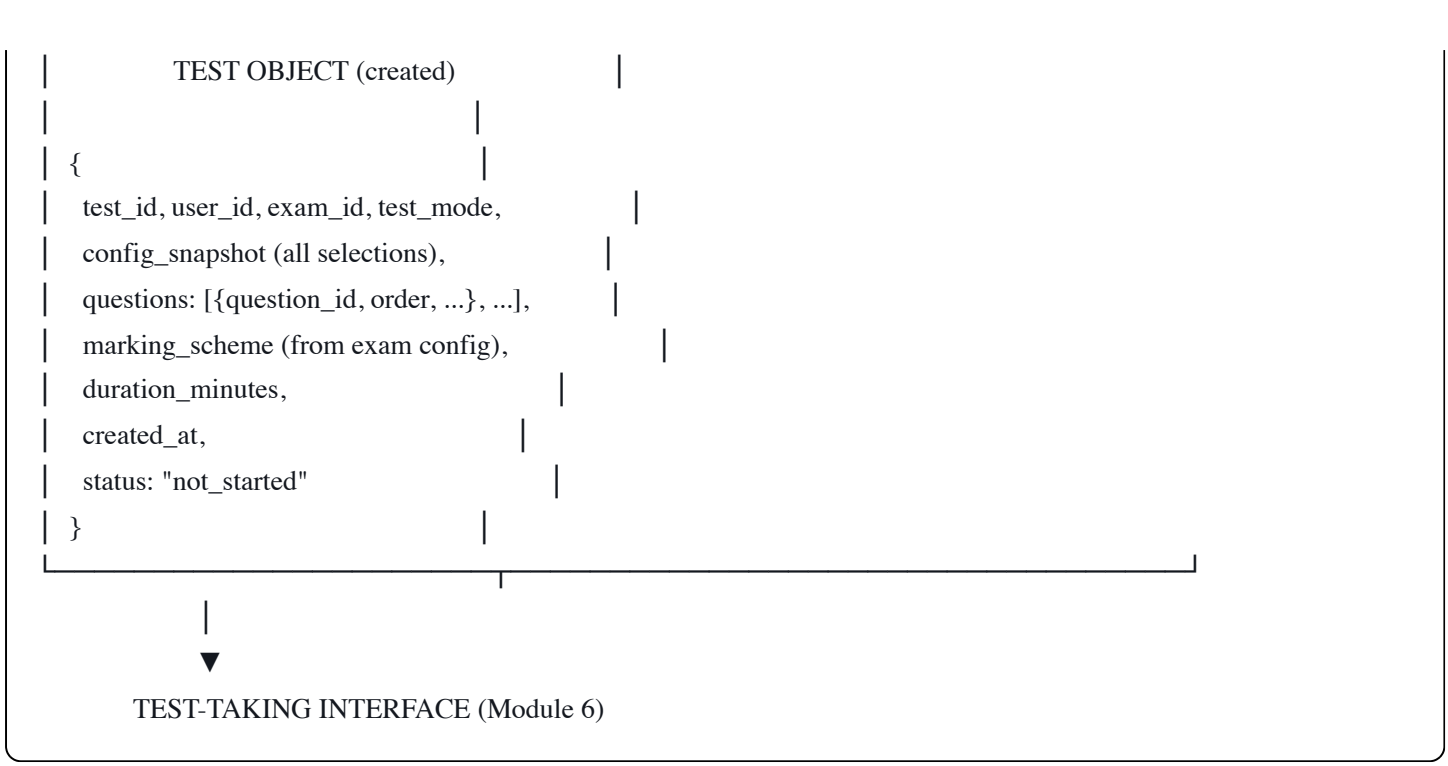
Display: "Estimated duration: 35 minutes (based on CSIR-NET timing)"

Manual Override

- User can set any duration from 5 minutes to 180 minutes
- No validation against auto-calculated time (student may want to practice under time pressure or with extra time)
- If manual duration is set, display: "Custom duration: X minutes"

4.12 COMPLETE DATA FLOW DIAGRAM





4.13 QUESTIONS FOR FOUNDER REVIEW

1. **Difficulty Mapping per Exam:** I've proposed specific Tier × Cognitive Level → Difficulty matrices for CSIR-NET, IIT-JAM, and CUET-PG (Section 4.4). These are illustrative — the actual mappings need your domain expertise to validate. Can you review the mapping tables and confirm or adjust them? This directly affects what students experience when they set the difficulty slider.
2. **Weakness Mode — Subtopic vs. Topic Level:** I've implemented Weakness mode at the subtopic level (accuracy calculated per subtopic). An alternative is topic level (coarser but requires less data per user). Which is correct? Subtopic-level is more precise but requires more test history before it has enough data to be meaningful.
3. **Weakness Threshold (60%):** Is 60% the right starting threshold? For competitive exam students, even 70% accuracy might indicate a weakness worth targeting. Should this be higher, or is 60% a good conservative start?
4. **Question Order in Test:** I've defaulted to randomized order. Should there be an option to order by difficulty (Easy → Hard, simulating exam progression)? Or is randomized always the right choice for practice tests?
5. **Test Mode in Free Tier:** Should free users have access to all 3 test modes, or only Standard? Revision and Weakness are powerful features — gating them to paid tiers could be a strong upgrade driver. On the other hand, giving free users Weakness mode could create a very compelling "aha moment."
6. **Fallback Transparency Level:** When the system adjusts topic distribution, difficulty, or type mix due to insufficient questions — how much detail should the student see? Options: (a) Detailed per-adjustment messages as I've specced, (b) A single summary: "Some adjustments were made to your test based on available questions", (c) No message — just serve the best test silently. I recommend (a) for trust, but it might feel overwhelming.

End of Module 4. Awaiting founder review before proceeding to Module 5: Question Bank Architecture.