# Exam Simulator — User Functionality Specification (Final)

This document describes the end‑user features, workflow, and interface design for the Exam Simulator. Admin/author roles are intentionally excluded. The UI must be fully responsive and mobile‑ready. All scoring, item selection, and reveal gating should be server‑side (temporary Replit server now; Firebase Functions later).

## 1) Purpose

Enable learners to practice certification‑style questions via configurable quizzes and timed exams. Mirror real exam conditions (domain weights, timing) while allowing controlled review and explanations.

## 2) Key Objectives

• Realistic, configurable simulations.  
• Daily practice (Quiz) and full Exam modes.  
• Save/resume at any time.  
• Domain‑weighted scoring based on blueprint.  
• Mobile‑first, accessible UI with clear feedback.

## 3) Functional Overview (User)

Users can:  
1. Create tests by selecting Certification Name, domains, question count, and time.  
2. Choose mode: Quiz (practice) or Exam (simulation).  
3. Configure review: quick review before/after, explanations while taking.  
4. Save progress and resume later.  
5. Submit and review domain‑weighted results with explanations.

## 4) Test Configuration Options

At session start, user must select:  
• Certification Name (required). Dropdown of known certifications with free‑text option.  
• Domains/topics (multi‑select).  
• Number of questions (5–100).  
• Timer (on/off, minutes).  
• Mode (Quiz or Exam).  
• Review preferences: Quick Review Before/After, Explanations While Taking.  
• Blueprint (domain weights). If a certification is chosen that has a default blueprint, prefill it.

## 5) Question Design

Each question includes:  
• Stem (rich text; images allowed).  
• Options (4–5), each with weight: 100%, 80%, 50%, 0%.  
• Explanation (overview + optional option notes).  
• Domain, difficulty, suggested time.  
Types: MCQ (single) and MSQ (multi‑select, labeled clearly).

## 6) Review & Explanations

• Quick Review Before: optional preview (no correctness/weights).  
• Quick Review After: immediate feedback (typically only in Quiz; configurable in Exam).  
• Explanations While Taking: show rationale immediately after answering if allowed for the session.

## 7) Selection & Assembly

• Eligibility: published items within chosen domains.  
• Domain distribution: approx round(weight% × total).  
• Renormalize weights when only a subset of blueprint domains is used.  
• Randomize items and option order; avoid recent repeats if possible.

## 8) Scoring

Per‑item:  
• MCQ = weight(selected).  
• MSQ = average(weight of selected) × coverage, where coverage = (#selected with weight>0) / min(#positive‑weight options, #selected).  
  
Domain mean = average of item scores in that domain.  
Final score (0–100%) = Σ(domainMean × domainWeight) × 100.  
Optional cut score for pass/fail.

## 9) Timer, Save/Resume & States

• Countdown timer (server‑authoritative).  
• Save anytime; store remainingSec and state.  
• Resume restores position, answers, reveal flags, and timer.  
• Auto‑submit when time reaches zero.  
• States: Active → Paused → Active → Submitted (or Expired).

## 10) Results & Review

Post‑submission show:  
• Overall weighted score and pass/cut label.  
• Domain table: Domain | Weight | #Qs | Mean | Contribution.  
• Item review with stem, your selections, per‑option weights, explanation.

## 11) Responsive & Mobile UI

• Mobile‑first with large tap targets (≥44px) and readable text.  
• Sticky bottom bar: Prev | Next | Save | Submit.  
• Progress (Q x / N) and timer chip in header.  
• Collapsible Review drawer for Quick‑Before/After content.  
• Color + icons for correctness (not color‑only).

## 12) Accessibility

• Keyboard navigable (Tab/Enter/Space), ARIA labels, screen‑reader support.  
• Adjustable font sizes; color‑safe palette and high‑contrast option.

## 13) Data Stored per Session (User Scope)

• sessionId, uid, certificationName  
• domains[], blueprint{domain:weight}, questionCount  
• questions[{qid, domain, type}], answers{qid:{selected[], perItemScore, revealed{explanation,feedback}}}  
• timer{enabled, durationMin, endsAt|remainingSec}, review toggles  
• status, index, createdAt, updatedAt, submittedAt  
• (after submit) perDomain scores and overallScorePct

## 14) Edge Cases & Error Handling

• Network loss → local cache & auto‑resume.  
• Timer drift → server time wins.  
• Insufficient questions → realloc shortfall and inform user pre‑exam.  
• Invalid config → fall back to defaults and alert the user.

## 15) Non‑Functional Requirements

• Responsive on desktop/tablet/mobile; load time < 3s per page.  
• Autosave at least every 30s.  
• Secure per‑user access to session data.  
• Availability target 99.5%.

## 16) Acceptance Criteria (User Scope)

✔ Certification Name captured and displayed across flow.  
✔ Domain‑weighted selection & scoring (with renormalization).  
✔ MCQ/MSQ with partial credit logic.  
✔ Save/resume with timer continuity.  
✔ Mobile‑ready interface and accessibility basics.  
✔ Results with domain breakdown and explanations.

# Implementation Notes for Replit

Implement a thin server on Replit with the same API contracts intended for Firebase Functions. Keep all authoritative logic server‑side. Use Replit DB (or KV/SQLite) behind a repository layer to ease migration to Firestore later.

## DB‑Agnostic Data Shapes

• Session: { id, uid, certificationName, mode, domains[], blueprint{domain:weight}, questionCount, timer{enabled, durationMin, endsAt|remainingSec}, review{quickBefore, quickAfter, explanationsWhileTaking}, index, questions[{qid, domain, type}], answers{ qid:{ selected:[ids], perItemScore, revealed:{explanation,feedback} } }, status, createdAt, updatedAt, submittedAt }  
• Question: { id, type:'MCQ'|'MSQ', stem, options[{id,text,weight in {1,.8,.5,0}}], explanation{ overview, optionNotes? }, domain, difficulty, timeSuggestedSec, version, status }  
• Result: { sessionId, certificationName, overallScorePct, perDomain:{ [domain]: { meanScore0to1, weight, contribution0to1, count } } }

## Renormalization Rule

Normalize selected domains so weights sum to 1.0; allocate round(weight\*questionCount); distribute rounding remainder to the largest fractional parts.

## MSQ Scoring Example

Selected A(1.0), B(0.8), C(0.0); two positive options exist.  
avgSelectedWeight = (1.0+0.8+0.0)/3 = 0.6  
coverage = (#selected with weight>0) / min(#positive‑weight options, #selected) = 2/2 = 1.0  
Per‑item score = 0.6 × 1.0 = 0.6 (60%).

## Timer & Pause Policy

Pause stores remainingSec; resume sets a new endsAt from remainingSec. Time‑out triggers auto‑submit.

## Insufficient Questions

Reallocate shortfall proportionally to the other selected domains and display a pre‑exam note.

## Mobile UI Specifics

Sticky bottom bar (Prev | Next | Save | Submit), big tap targets, progress/timer in header, and a collapsible Review drawer.

## Results Screen

Show weighted overall score, pass/cut label, domain table, and full item review (selections, weights, explanations).

# Server API Spec (Replit now → Firebase later)

All JSON. Keep shapes identical on Firebase. Include 'certificationName' everywhere.

## POST /sessions — Create a session

Request:  
{  
 "certificationName": "AWS SAA-C03",  
 "mode": "quiz" | "exam",  
 "domains": ["Security","Networking"],  
 "blueprint": {"Security":0.25,"Networking":0.20,"Compute":0.25,"Storage":0.15,"Governance":0.15},  
 "questionCount": 30,  
 "timer": {"enabled": true, "durationMin": 45},  
 "review": {"quickBefore": false, "quickAfter": false, "explanationsWhileTaking": false}  
}  
  
Response:  
{  
 "sessionId": "sess\_123",  
 "certificationName": "AWS SAA-C03",  
 "summary": { "allocations": [{"domain":"Security","questions":8}, ...], "endsAt": "2025-10-10T15:00:00Z" }  
}

## POST /grade — Grade a single item

Request: { "sessionId":"sess\_123", "qid":"q\_001", "selected":["A","E"] }  
  
Response: { "certificationName":"AWS SAA-C03", "perItemScore":0.6, "feedbackAllowed":true, "explanation":{"overview":"...","optionNotes":{"A":"...","E":"..."}} }

## POST /pause — Pause timer

Request: { "sessionId":"sess\_123" }  
Response: { "certificationName":"AWS SAA-C03", "remainingSec":1320 }

## POST /resume — Resume timer

Request: { "sessionId":"sess\_123" }  
Response: { "certificationName":"AWS SAA-C03", "endsAt":"2025-10-10T15:00:00Z" }

## POST /submit — Submit and score

Request: { "sessionId":"sess\_123" }  
  
Response:  
{  
 "certificationName":"AWS SAA-C03",  
 "overallScorePct":72,  
 "perDomain":[{"domain":"Security","weight":0.25,"count":8,"mean":0.80,"contribution":0.20}],  
 "items":[{"qid":"q\_001","yourSelection":["A","E"],"perItemScore":0.6,"weights":{"A":1,"B":0.8,"C":0,"D":0},"explanation":{"overview":"..."}}]  
}

## GET /sessions/:id — Fetch state

Response: { "certificationName":"AWS SAA-C03", "status":"active|paused|submitted|expired", "index":7, "questions":[{"qid":"q\_001","domain":"Security","type":"MSQ"}], "answers":{"q\_001":{"selected":["A","E"],"perItemScore":0.6}}, "review":{"quickBefore":false,"quickAfter":false,"explanationsWhileTaking":false}, "timer":{"enabled":true,"endsAt":"...","remainingSec":900} }

# Question Fetching & Redaction Model

The client never reads questions directly from the database. The server (Replit now → Firebase later) selects items for a session and returns only a redacted, per-session view. Weights and full explanations are never sent unless the session settings allow explanations for that moment.

## Endpoints for Delivering Question Content

1) GET /sessions/:id/items?from=0&limit=10  
 • Purpose: Return paged, redacted items for the active session.  
 • Response (array of items): [{ qid, type:'MCQ'|'MSQ', domain, stem, options:[{id,text}], exhibits?:[{type:'image'|'pdf',url}] }]  
 • Notes: No weights. No explanations. Use from/limit for pagination.  
  
2) GET /sessions/:id/item/:qid/explanation  
 • Purpose: Return explanation for a single item only if allowed by session policy.  
 • Response: { explanation:{ overview, optionNotes? } }  
 • Notes: If not allowed, return 403 or a response with allowed:false.

## Question Store (Server/Internal)

Collection: questions  
• id (pk), status('published'|'retired'|'pilot'), version(int)  
• type('MCQ'|'MSQ'), stem, options[{id,text,weight in {1,.8,.5,0}}]  
• explanation:{ overview, optionNotes? }  
• domain, objectiveId?, difficulty(1-5), timeSuggestedSec  
• exhibits?:[{type:'image'|'pdf',url}]  
• createdAt, updatedAt  
  
The session stores only: qid, domain, type, and a version snapshot. Server joins session.questions with questions to produce the redacted response, filtering out weights and (unless allowed) explanations.