SDPF SPECIFICATION
Problem Owner: Hamza Abdullah
Problem Statement: "Recruiters on HackerEarth's platform produce different question sets for equivalent roles across different recruiters, against a target of 100% of role assessments producing a system-guided consistent question set, resulting in non-comparable candidate scores and degraded hiring decision quality."
Style: 1 — Technical Specification
Status: DRAFT
Version: 1.0.0
Date: 2026-05-11

S-01 — Style Declaration

Style ID: 1 **Style Name:** Technical Specification

Rationale: The system accepts a fully definable input (job description and required skills), executes deterministic retrieval logic against a fixed FAISS vector index, and returns a fully defined output (ranked assessment question set with similarity scores). All processing logic is specifiable before implementation begins. Requirements are fully known. The system is a single-service REST API built by one team. No compliance framework is the primary design driver. No exploration is required. Style 1 is the correct selection: deterministic system with fully definable I/O.

AI Framing: Expert software engineer implementing a contract-first REST API specification using LangChain, FAISS, and sentence-transformers. No behaviour beyond the contract. Every function annotated with its REQ-ID. No assumptions about intent.

S-02 — Style Context

Domain: Technical talent assessment. The system is a question retrieval service that takes a recruiter's job description and required skill list and returns a ranked set of assessment questions drawn from a structured knowledge base.

System Boundary:

- IN SCOPE: Input validation. Query embedding using sentence-transformers. FAISS index search. Result ranking by cosine similarity. JSON response construction. REST API serving via Flask.
- OUT OF SCOPE: Candidate-facing assessment delivery. Candidate answer evaluation. Hiring decisions. Question authoring. User authentication. Multi-tenancy. Integration with HackerEarth production platform.

Problem Boundary (from Phase 0): The gap is in question selection — the step where a recruiter defines which questions form a role's assessment. The system closes this gap by replacing unguided manual selection with deterministic, similarity-ranked retrieval that produces the same output for the same input regardless of which recruiter submits the request.

S-03 — External Contract

Interface Type: REST API **Data Serialisation Format:** JSON **Version:** v1 **Base Path:** /api/v1 **Protocol:** HTTP/1.1 **Host:** localhost:5000 (case study deployment)

Problem Boundary Reference: The external contract defines the input-output boundary of the question selection step identified in Phase 0. The desired state — consistent, system-guided question sets for identical role inputs — is expressed as a guaranteed output property in S-06 (OG-9).

Endpoints declared in this specification:

Method	Path	Description
POST	/api/v1/retrieve	Retrieve and rank assessment questions for a role
GET	/api/v1/health	System health and readiness status

Normative error response schema for all error conditions:

```json
{ "error": { "code": string, "message": string, "field": string | null } }
```

S-04 — Input Contract

POST /api/v1/retrieve

Request body: JSON object. Header Content-Type: application/json required.

Field	Type	Required	Constraints
job_description	string	YES	Non-null. Non-empty after stripping whitespace. Minimum 10 characters after stripping. Maximum 5,000 characters after stripping. UTF-8.
required_skills	array of string	YES	Non-null. JSON array. Minimum 1 element. Maximum 20 elements. Each element: non-null string, non-empty after stripping, maximum 100 characters after stripping.
top_k	integer	NO	JSON integer type (not float). Minimum 1. Maximum 50. Default when absent: 10.

Validation rules:

- `job_description`: strip leading and trailing whitespace before all length checks. Null value rejected. Non-string type rejected.
- `required_skills`: must be a JSON array. Non-array rejected. Empty array rejected. Each element stripped before length check. Null elements rejected. Duplicate skill strings permitted and kept as-is.
- `top_k`: if present, must be a JSON integer. Float type (e.g. 10.0) rejected. Values below 1 or above 50 rejected. If key absent, system uses 10.

GET /api/v1/health

No request body. No query parameters. No headers required beyond standard HTTP.

S-05 — Processing Rules

PR-01 — Content-Type Enforcement Every POST /api/v1/retrieve request: check Content-

Type header before any other processing. If absent or not application/json, return HTTP 415 with no body. No further processing shall occur.

PR-02 — JSON Parsing After PR-01 passes, parse request body as JSON. If parsing fails, return HTTP 400, code INVALID_JSON, message "Request body is not valid JSON", field null. Stop.

PR-03 — Input Validation After PR-02 passes, validate each field in order:

1. Validate job_description per S-04. On failure: HTTP 422, code INVALID_JOB_DESCRIPTION, message "job_description must be a non-empty string between 10 and 5000 characters", field "job_description". Stop.
2. Validate required_skills per S-04. On failure: HTTP 422, code INVALID_SKILLS, message "required_skills must be a non-empty array of 1 to 20 non-empty strings each at most 100 characters", field "required_skills". Stop.
3. Validate top_k per S-04. On failure: HTTP 422, code INVALID_TOP_K, message "top_k must be an integer between 1 and 50", field "top_k". Stop.
4. If top_k is absent, set top_k = 10.
5. If all validations pass, proceed to PR-04.

PR-04 — Readiness Check Before retrieval, verify both the FAISS index and the sentence-transformers model are loaded and ready. If either is not ready, return HTTP 503, code INDEX_NOT_READY, message "Retrieval index is not ready. Retry after server initialisation completes.", field null. No retrieval shall occur against an unloaded index.

PR-05 — Query Construction Construct the query string: query = strip(job_description) + " " + " ".join([strip(s) for s in required_skills]) Example: job_description="Senior backend engineer", required_skills=["python","sql"] produces: "Senior backend engineer python sql"

PR-06 — Query Embedding Encode the query string from PR-05 using the sentence-transformers model all-MiniLM-L6-v2 loaded at server startup. Call model.encode() with

parameter normalize_embeddings=True. Result is a 384-dimensional float32 numpy array. The model instance shall be loaded once at startup and reused for every request. The model shall not be loaded per-request.

PR-07 — FAISS Index Search Query the FAISS IndexFlatIP index with the normalised query embedding from PR-06. Retrieve the top top_k nearest neighbours. IndexFlatIP with normalised vectors computes cosine similarity. Search returns: indices (integer array) and scores (float array) of length top_k. Map each score to: similarity_score = (score + 1.0) / 2.0 producing a value in [0.0, 1.0]. If the index contains fewer than top_k questions, retrieve all available.

PR-08 — Result Construction For each retrieved index position i (0-indexed), look up the question record from the question knowledge base by the question ID stored at that index position. Construct a result object: rank=(i+1), id, title, skill, difficulty, type, tags, similarity_score rounded to 4 decimal places.

PR-09 — Determinism Given identical job_description, required_skills, and top_k against an identical FAISS index, the system shall return an identical questions array with identical scores, ranks, and ordering on every call. No random seeds, stochastic sampling, or runtime variation shall be introduced at any processing step.

PR-10 — Health Check GET /api/v1/health returns HTTP 200 with JSON body: status ("healthy" if FAISS index and model are both loaded, "degraded" otherwise), version "1.0.0", timestamp in ISO 8601 UTC format.

S-06 — Output Guarantees

POST /api/v1/retrieve — HTTP 200 Success

```
json
```

```
{
  "questions": [
    {
      "rank": integer,
      "id": string,
      "title": string,
      "skill": string,
      "difficulty": string,
      "type": string,
      "tags": array of string,
      "similarity_score": float
    }
  ],
  "metadata": {
    "total_returned": integer,
    "top_k_requested": integer,
    "query_skills": array of string,
    "timestamp": string
  }
}
```

Guaranteed output properties:

- OG-1: `questions` array length equals min(top_k, total questions in FAISS index).
- OG-2: `rank` starts at 1 and increments by 1 with no gaps and no duplicates.
- OG-3: `similarity_score` is a float in [0.0, 1.0] rounded to 4 decimal places.
- OG-4: Questions ordered by similarity_score descending. Rank 1 has the highest similarity_score. If two questions have identical similarity_score to 4 decimal places, order by id ascending as tiebreaker.
- OG-5: `total_returned` equals the length of the `questions` array.

- OG-6: `top_k_requested` equals the resolved top_k value used for retrieval.
- OG-7: `query_skills` contains the required_skills array as submitted, in original order, unmodified.
- OG-8: `timestamp` is ISO 8601 UTC: "YYYY-MM-DDTHH:MM:SS.sssZ".
- OG-9 (Desired State from Phase 0): For identical job_description, required_skills, and top_k inputs against an identical FAISS index, the questions array, all similarity_scores, and all ranks shall be identical across every call from any recruiter or session.

GET /api/v1/health — HTTP 200

```json
{
  "status": "healthy" | "degraded",
  "version": "1.0.0",
  "timestamp": string
}
```

S-07 — Exception Handling

Code	HTTP	Trigger Condition	field
INVALID_JSON	400	Request body cannot be parsed as JSON	null
INVALID_JOB_DESCRIPTION	422	job_description absent, null, not a string, empty after strip, < 10 chars, or > 5000 chars	"job_description"

Code	HTTP	Trigger Condition	field
INVALID_SKILLS	422	required_skills absent, null, not an array, empty, any element null or empty after strip, any element > 100 chars, or array length > 20	"required_skills"
INVALID_TOP_K	422	top_k present and not an integer, or present and outside [1, 50]	"top_k"
INDEX_NOT_READY	503	POST /retrieve called before FAISS index or model has completed loading	null
INTERNAL_ERROR	500	Any unhandled exception during embedding, index search, or result construction	null

EH-4 — CRITICAL error recovery: For INDEX_NOT_READY, log the request and not-ready state. The system shall not attempt partial retrieval against an unloaded index under any condition.

S-08 — Technical Verification Gate

STATUS: UNVERIFIED — All entries must pass verification against the live target environment before this specification may be locked. TVG-3 applies: if any entry fails, update the specification before proceeding. Do not work around a failure.

TVG-01 Tool / Asset: Python interpreter Asserted Value: 3.8 or higher Verification Command: python --version Pass Condition: Output contains "Python 3." followed by

minor version 8 or higher HALT: Do NOT proceed if Python version is below 3.8

TVG-02 Tool / Asset: langchain package Asserted Value: 0.2.0 or higher Verification Command: python -c "import langchain; print(langchain.**version**)" Pass Condition: Output is a version string of 0.2.0 or higher HALT: Do NOT proceed if langchain is not installed or below 0.2.0

TVG-03 Tool / Asset: langchain-community package Asserted Value: 0.2.0 or higher Verification Command: python -c "import langchain_community; print(langchain_community.**version**)" Pass Condition: Output is a version string of 0.2.0 or higher HALT: Do NOT proceed if langchain_community is not installed

TVG-04 Tool / Asset: faiss-cpu package Asserted Value: 1.7.4 or higher Verification Command: python -c "import faiss; print(faiss.**version**)" Pass Condition: Output is a version string of 1.7.4 or higher HALT: Do NOT proceed if faiss is not installed

TVG-05 Tool / Asset: sentence-transformers package Asserted Value: 2.7.0 or higher Verification Command: python -c "import sentence_transformers; print(sentence_transformers.**version**)" Pass Condition: Output is a version string of 2.7.0 or higher HALT: Do NOT proceed if sentence_transformers is not installed

TVG-06 Tool / Asset: sentence-transformers model all-MiniLM-L6-v2 Asserted Value: Model loads and encodes to 384 dimensions Verification Command: python -c "from sentence_transformers import SentenceTransformer; m = SentenceTransformer('all-MiniLM-L6-v2'); v = m.encode('test', normalize_embeddings=True); print(len(v))" Pass Condition: Output is exactly "384" HALT: Do NOT proceed if model does not load or output dimension is not 384

TVG-07 Tool / Asset: flask package Asserted Value: 2.3.0 or higher Verification Command: python -c "import flask; print(flask.**version**)" Pass Condition: Output is a version string of 2.3.0 or higher HALT: Do NOT proceed if flask is not installed

TVG-08 Tool / Asset: numpy package Asserted Value: 1.24.0 or higher Verification Command: python -c "import numpy; print(numpy.**version**)" Pass Condition: Output is a

version string of 1.24.0 or higher HALT: Do NOT proceed if numpy is not installed

TVG-09 Tool / Asset: question_kb.json knowledge base file Asserted Value: File exists at ./question_kb.json and contains at least 50 question records Verification Command: python -c "import json; d=json.load(open('question_kb.json')); print(len(d['questions']))" Pass Condition: Output is an integer of 50 or higher HALT: Do NOT proceed if file is absent or question count is below 50

TVG-10 Tool / Asset: FAISS IndexFlatIP cosine similarity behaviour with normalised vectors Asserted Value: Inner product of two identical normalised vectors equals 1.0 Verification Command: python -c "import faiss, numpy as np; idx=faiss.IndexFlatIP(4); v=np.array([[1,0,0,0]],dtype='float32'); idx.add(v); s,i=idx.search(v,1); print(round(float(s[0][0]),1))" Pass Condition: Output is exactly "1.0" HALT: Do NOT proceed if FAISS inner product behaviour is not as asserted

---

S-09 — Verification Requirements

Test ID	REQ-ID	Description	Pass Condition
TEST-001	R-001	POST /retrieve with valid inputs and top_k=5	HTTP 200, questions array length == 5
TEST-002	R-001	POST /retrieve called twice with identical inputs	Response 1 questions array == Response 2 questions array (all fields, all values)
TEST-003	R-002	POST /retrieve with top_k absent	HTTP 200, questions array length == 10
TEST-004	R-002	POST /retrieve with top_k=1	HTTP 200, questions array length == 1

Test ID	REQ-ID	Description	Pass Condition
TEST-005	R-003	POST /retrieve returns similarity_score values in [0.0, 1.0]	All similarity_score values are floats between 0.0 and 1.0 inclusive
TEST-006	R-003	POST /retrieve returns questions sorted by similarity_score descending	questions[i].similarity_score >= questions[i+1].similarity_score for all i
TEST-007	R-004	POST /retrieve with job_description absent	HTTP 422, error.code == "INVALID_JOB_DESCRIPTION", error.field == "job_description"
TEST-008	R-004	POST /retrieve with job_description of 5 characters	HTTP 422, error.code == "INVALID_JOB_DESCRIPTION"
TEST-009	R-004	POST /retrieve with required_skills as empty array	HTTP 422, error.code == "INVALID_SKILLS", error.field == "required_skills"
TEST-010	R-004	POST /retrieve with top_k=51	HTTP 422, error.code == "INVALID_TOP_K", error.field == "top_k"
TEST-011	R-004	POST /retrieve with top_k=0	HTTP 422, error.code == "INVALID_TOP_K"
TEST-012	R-004	POST /retrieve with top_k=10.0 (float)	HTTP 422, error.code == "INVALID_TOP_K"

Test ID	REQ-ID	Description	Pass Condition
TEST-013	R-004	POST /retrieve with non-JSON body	HTTP 400, error.code == "INVALID_JSON"
TEST-014	R-005	POST /retrieve returns rank starting at 1 with no gaps	questions[0].rank==1, questions[n-1].rank==n
TEST-015	R-005	POST /retrieve returns all required fields for every question	id, title, skill, difficulty, type, tags present and non-null in every question object
TEST-016	R-006	POST /retrieve without Content-Type: application/json	HTTP 415
TEST-017	R-007	GET /api/v1/health when index and model loaded	HTTP 200, status == "healthy", version == "1.0.0"
TEST-018	R-001	POST /retrieve with different required_skills for same job_description	Top-ranked question differs between the two responses

S-10 — Traceability Matrix

REQ-ID	Priority	Requirement Summary	TEST-ID(s)	Implementation Artifact
R-001	[CRITICAL]	System accepts valid job_description and required_skills and returns HTTP 200 with ranked question list	TEST-001, TEST-002, TEST-018	retrieve_endpoint()
R-002	[CRITICAL]	top_k controls number of returned questions; default 10; valid range 1–50	TEST-003, TEST-004	retrieve_endpoint()
R-003	[CRITICAL]	Questions ranked by cosine similarity descending; similarity_score in [0.0,1.0] rounded to 4 decimal places	TEST-005, TEST-006	rank_results()
R-004	[REQUIRED]	System validates all inputs and returns HTTP 422 with structured error for every invalid input condition	TEST-007, TEST-008, TEST-009, TEST-010, TEST-011, TEST-012, TEST-013	validate_input()
R-005	[REQUIRED]	Response includes rank, id, title, skill, difficulty, type, tags for every question; rank starts at 1 with no gaps	TEST-014, TEST-015	build_response()

REQ-ID	Priority	Requirement Summary	TEST-ID(s)	Implementation Artifact
R-006	[REQUIRED]	System returns HTTP 415 when Content-Type is not application/json	TEST-016	retrieve_endpoint()
R-007	[OPTIONAL]	GET /api/v1/health returns HTTP 200 with status, version, timestamp	TEST-017	health_endpoint()

Phase 0 Validation Summary

Component	Value	Status
Current State	Recruiters produce different question sets for equivalent roles with no guided retrieval	Observable and measurable
Desired State	100% of role assessments produce a system-guided consistent question set	Specific and testable
Gap	0% of assessments currently system-guided — gap of 100 percentage points	Quantified
Impact	Non-comparable candidate scores and degraded hiring decision quality	Business terms
Problem Owner	Hamza Abdullah	Identified

Component	Value	Status
T-1 Observable	Question set inconsistency across recruiters is measurable	PASS
T-2 Bounded	Scoped to question selection step for technical role assessments	PASS
T-3 Cause-Free	No cause language in problem statement	PASS
T-4 Solution-Free	No solution language in problem statement	PASS

Pre-Lock Checklist

- [x] All 10 required sections (S-01 through S-10) present in declared order. Problem statement referenced.
- [x] Exactly one style declared in S-01.
- [x] CRITICAL requirements R-001, R-002, R-003 directly address the problem gap.
- [x] Every requirement begins with a priority tag.
- [x] Every requirement written at precision level — single testable condition, one interpretation.
- [x] Exception Handling covers every reachable error condition.
- [ ] TVG: all 10 entries UNVERIFIED. Must be verified against live environment before locking.
- [x] Traceability Matrix contains at least one mapping row per CRITICAL and REQUIRED requirement.
- [x] No unresolved CRITICAL-CRITICAL conflicts.
- [x] Style 1 requires no additional sections beyond S-01 to S-10.

STATUS: DRAFT
Next action: Run all 10 TVG verification commands in the target environment.
Update any entry that fails. Lock only after all 10 pass.