**Step-by-Step Project Workflow (Conceptual)**

* **[Design] Select LLM + rubric**: Choose model and rubric weights/criteria per subject/topic.
* **[Prompting] Prepare prompt**:
  + Role as expert examiner
  + Provide Ideal Answer
  + CoT steps: extract concepts → compare → allocate points → final score + justification
  + Output in strict structure: {"Score":"X/10","Justification":"...","Key\_Concepts\_Covered":[...]}
* **[Data] Author question + ideal answer**: Stored in MSSQL and retrieved by services (QuestionService, AnswerService).
* **[Preprocessing] Extract key concepts**: From the ideal answer (RAG/CoT) and persist concepts for consistent marking.
* **[Runtime] Student submits answer**: Persist to DB.
* **[Grading] Run workflow**:
  + Load question + ideal + rubric
  + Ensure concepts available (extract if missing)
  + Compare student answer vs key concepts
  + Allocate rubric points; produce final response JSON
  + Persist results and return
* **[Ops] Monitor bias and audit**: Store detailed justification for review and fairness checks.

**API Endpoints and Call Order For a New User**

There are two supported paths: a quick “LLM-only” flow (no DB) and the full DB-backed grading workflow.

**A) Quick Start (No Database Required): LLM-only**

Best for trying the AI examiner quickly with ad-hoc payloads; uses

src/api/routers/llm\_api.py.

1. **(Optional) Check LLM provider**
   * GET /llm/health or POST /llm/provider/test
   * Why: Validate the provider/API key is reachable and model configured.
2. **(Optional) Explore examples**
   * GET /llm/examples/rubric
   * GET /llm/examples/ideal-answer
   * GET /llm/examples/student-answer
   * Why: See the expected schema for payloads.
3. **(Optional) Extract concepts from ideal**
   * POST /llm/analyze/concepts
   * Body: IdealAnswer object
   * Why: Understand what the LLM sees as key concepts.
4. **(Optional) Semantic similarity (no grading)**
   * POST /llm/analyze/similarity
   * Body: IdealAnswer, StudentAnswer
   * Why: Inspect concept coverage and similarity details.
5. **Grade a single answer**
   * POST /llm/grade
   * Body: GradingRequest with ideal\_answer, student\_answer (include rubric)
   * Why: Runs CoT grading entirely in-memory and returns structured result:
     + result.percentage, Key\_Concepts\_Covered, Justification, etc.
6. **Batch grade**
   * POST /llm/grade/batch
   * Body: BatchGradingRequest
   * Why: Efficiently evaluate many answers in one request.

**B) Full Workflow (Backed by MSSQL)**

Best for production where questions/answers/concepts/grades are persisted; uses

src/api/routers/question\_api.py,

src/api/routers/answer\_api.py,

src/api/routers/grade\_api.py.

1. **(One-time) Confirm DB connectivity**
   * Start the app and check root GET / or the interactive docs /docs.
   * Why:

src/api/API.py initializes DB and injects services into routers at startup.

1. **Retrieve the question (with ideal answer and marks)**
   * GET /question/{question\_id}
   * Why: Ensures the ideal answer and rubric exist before extraction/grading.
2. **Extract and persist key concepts for the question**
   * POST /question/extract-concepts/{question\_id}
   * Why: Creates/updates Question\_KeyConcept rows used by the grader for consistent marking and auditing.
3. **(Optional) Verify concepts saved**
   * GET /question/concepts/{question\_id}
   * Why: Inspect concepts, total\_max\_points, and composition.
4. **Submit student’s narrative answer**
   * POST /answer/submit
   * Body: SubmitAnswerRequest with student\_id, question\_id, answer\_text, language
   * Why: Persists the student’s response used by the workflow.
5. **Run the complete grading workflow**
   * POST /grade/workflow
   * Body: GradingWorkflowRequest with question\_id, student\_id
   * Why: Executes end-to-end:
     + Loads ideal answer/rubric
     + Extracts/saves concepts (if not already done)
     + Retrieves student’s answer
     + Grades via LLM + rubric
     + Saves and returns the result as the required JSON:
       - {"Score":"X/10","Justification":"...","Key\_Concepts\_Covered":[...]}
   * For multiple students/questions: POST /grade/batch/workflow
6. **(Optional) Review stored data**
   * Answers:
     + GET /answer/student/{student\_id}
     + GET /answer/student/{student\_id}/question/{question\_id}
     + GET /answer/all
   * Ideal answers:
     + GET /answer/ideal-answers
     + GET /answer/ideal-answers/{question\_id}
   * Why: Auditing and result validation.

**Example Minimal Payloads**

* **Submit answer** (POST /answer/submit):

json

{  
 "student\_id": 123,  
 "question\_id": 45,  
 "answer\_text": "Student narrative answer...",  
 "language": "en"  
}

* **Run workflow** (POST /grade/workflow):

json

{  
 "question\_id": 45,  
 "student\_id": 123  
}

* **LLM grade (no DB)** (POST /llm/grade):

json

{  
 "ideal\_answer": {  
 "question\_id": "Q45",  
 "content": "Ideal narrative answer...",  
 "subject": "Subject",  
 "rubric": {  
 "subject": "Subject",  
 "topic": "Topic",  
 "criteria": [  
 {"name": "Main Idea", "description": "...", "max\_points": 4, "weight": 1.0},  
 {"name": "Evidence", "description": "...", "max\_points": 4, "weight": 1.0},  
 {"name": "Clarity", "description": "...", "max\_points": 2, "weight": 1.0}  
 ],  
 "total\_max\_points": 10,  
 "passing\_threshold": 6  
 }  
 },  
 "student\_answer": {  
 "student\_id": "S123",  
 "question\_id": "Q45",  
 "content": "Student narrative answer..."  
 }  
}

**Which Endpoint First-to-Last (Why)**

* **LLM-only trial (fastest)**
  + First: GET /llm/health → validate provider.
  + Then: POST /llm/grade → get a full structured grade without DB.
  + Why: No setup; ideal for demos or sandboxing.
* **Production workflow (persistent, auditable)**
  + First: GET /question/{question\_id} → ensure question and ideal answer exist.
  + Then: POST /question/extract-concepts/{question\_id} → persist concepts for consistent marking.
  + Then: POST /answer/submit → store the student narrative.
  + Then: POST /grade/workflow → full grading and result persistence.
  + Optional: GET /answer/student/{student\_id}/question/{question\_id} → review saved result.
  + Why: Aligns with rubric-driven grading, auditing, and data integrity requirements.

**Recommended Actions**

* **[Configure LLM provider]** Ensure API key/model in your config so llm\_service passes GET /llm/health.
* **[Seed questions/ideals]** Make sure the DB has questions and ideal answers before concept extraction and workflow.
* **[Test both paths]** Validate the LLM-only path for correctness and the DB workflow for persistence and auditing.

**Task Status**

* **Completed**: Mapped all available endpoints, provided step-by-step project workflow, and defined the exact first-to-last endpoint call order for both quick LLM-only usage and full DB-backed grading.