Exam Q&A Generator – End-to-End Technical Documentation

Version: July 2025 > **Author:** Siva Sagar / ChatGPT

---

1 Executive Summary

An interactive web application that ingests study PDFs, automatically generates exam-style quizzes, grades student responses with an LLM, recommends weak topics, and lets you fine-tune and deploy a custom TinyLlama model locally via **Ollama**.

---

2 Technology Stack

Layer	Technology	Purpose
UI	**Streamlit 1.35**	Lightweight web front-end
LLM runtime	**Ollama**	Serve local GGUF models + simple REST
Base model	**TinyLlama-1.1B-Chat-v1.0**	1.1 B parameter chat model
Fine-tuning	**PEFT (LoRA) + bitsandbytes 4-/8-bit**	Parameter-efficient adaptation
Vector DB	**ChromaDB 0.5**	Store paragraph embeddings + similarity search
Embeddings	**sentence-transformers/ all-MiniLM-L6-v2**	Embed PDF text
PDF parsing	**PyPDF2 3.0.1**	Extract raw text
Conversion	**llama.cpp convert_hf_to_gguf.py**	HF ⇒ GGUF (q8_0)
Analytics	**SQLite + Pandas + Matplotlib**	Persist & visualise quiz scores
Infra	macOS 13 / Python 3.11 / Colab GPU (T4)	

---

3 Repository Layout

```
exam-qa-generator/
├── app/
```

```
│   ├── pdf_loader.py       # PDF → clean chunks
│   ├── vector_store.py     # Chroma client helpers
│   ├── qa_generator.py     # Ollama prompt to gen Q-A pairs
│   ├── scoring.py          # Strict grader (LoRA model)
│   ├── recommendation.py   # Topic weakness detection
│   ├── database.py         # SQLite schema
│   └── ui.py               # Streamlit front-end
├── fine-tuning-scripts/
│   ├── convert_jsonl.py    # Raw Q-A → JSONL dataset
│   ├── dataset.jsonl       # ~4 k Constitution Q-A pairs
│   └── fine_tuning.py      # LoRA training loop
└── models/
    ├── tinyllama-merged/   # HF merged weights + tokenizer
    └── tinyllama-qa.gguf   # Quantised q8_0 for Ollama
```

---

4 Application Flows

4.1 Quiz Generation

1. **Upload PDF** → `pdf_loader.load_pdf` → list of paragraphs.
2. Store embeddings in persistent Chroma (`vector_store.add_chunks`).
3. Prompt Ollama with `_SYSTEM_PROMPT` to create `n` JSON Q&A pairs.

4.2 Student Session

1. Streamlit renders each question and captures answers.
2. For each answer call `scoring.grade()` → LoRA model returns score 0-1 + feedback JSON.
3. Results persisted to `SQLite` (`store_results`).
4. Recommended topics computed via a simple threshold on per-topic average.
5. Option to **export a PDF report** using `FPDF` and to view historical analytics.

---

5 Key Source Files (excerpts)

5.1 `qa_generator.py`

```python
from ollama import Client, generate_prompt
_CLIENT = Client(host=os.getenv("OLLAMA_BASE_URL", "http://localhost:11434"))

_SYSTEM_PROMPT = """
You are a helpful tutor creating quiz questions from textbook content…
Return a list: [{"question":…, "answer":…, "topic":…}]"""
```

```python
def generate_qa_pairs(doc_id:str, n:int=10, topic:str|None=None):
    seed = similarity_search(topic or "overview", k=50)
    ctx = "\n".join(random.sample(seed, 8))
    prompt = f"{_SYSTEM_PROMPT}\nContext:\n{ctx[:800]}\nGenerate {n} pairs."
    resp = _CLIENT.generate(model=os.getenv("OLLAMA_MODEL"), prompt=prompt)
    return json.loads(first_json(resp["response"]))
```

5.2 `fine_tuning.py` (LoRA on Colab)

```python
model = AutoModelForCausalLM.from_pretrained(BASE_MODEL, load_in_4bit=True,
            quantization_config=bnb_cfg, device_map="auto")
peft_cfg = LoraConfig(r=8, lora_alpha=16, target_modules=["q_proj","v_proj"],
                      task_type=TaskType.CAUSAL_LM)
model = get_peft_model(model, peft_cfg)
…
trainer.train()
model.save_pretrained("tinyllama-merged")
```

5.3 GGUF Conversion & Deployment

```bash
# convert HF to q8_0 GGUF
python3 llama.cpp/convert_hf_to_gguf.py tinyllama-merged \
       --outfile tinyllama-qa.gguf --outtype q8_0

# Modelfile for Ollama
FROM ./tinyllama-qa.gguf

# create & run
ollama create tinyllama-qa -f Modelfile
ollama run tinyllama-qa
```

---

6 Setup Instructions

6.1 Local MacBook

```bash
brew install ollama chroma llama.cpp
python3 -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
export OLLAMA_MODEL=tinyllama-qa  # after import
streamlit run app/ui.py
```

6.2 Colab Fine-Tuning (\~25 min on T4)

```
!pip install transformers datasets peft bitsandbytes accelerate
!python convert_jsonl.py constitution.pdf dataset.jsonl
!python fine_tuning.py  # produces tinyllama-merged/
!zip -r tinyllama-merged.zip tinyllama-merged
from google.colab import files; files.download("tinyllama-merged.zip")
```

---

7 Performance

Stage	Time (T4)	Notes
PDF ingestion (100 p)	\~7 s	tokenisation + embeddings
Question generation (10)	\~12 s	TinyLlama q8 in Ollama
Grading 10 answers	\~4 s	single batch
LoRA fine-tune (4k ex, 3 epochs)	\~22 min	LR 1e-4, VRAM $\leq$ 11 GB

---

8 Security & Privacy

- **All inference is local** – no external API calls are made after the initial model pull.
- Quiz history is stored in local SQLite only.
- PDFs are deleted from `/tmp` after ingestion.

---

9 Future Work

1. Add **multi-PDF merging** with chapter weight sliders.
2. Vector-based answer grading to reduce hallucination.
3. Docker compose for one-click deployment.
4. WebSocket progress updates during fine-tuning.

---

10 Appendix – Full Requirements

```
streamlit>=1.35.0
pypdf2>=3.0.1
chromadb>=0.5.0
sentence-transformers>=2.7.0
ollama-python>=0.1.6
```

```
python-dotenv>=1.0.1
transformers>=4.40.0
datasets>=3.6.0
peft>=0.15.2
bitsandbytes>=0.43.0
```

End of Document – Happy Studying! 📚