Exam Q&A Generator – End-to-End Code-Oriented Documentation

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

---

1 Project Summary

A complete local AI-based exam preparation system. You upload a PDF, generate quiz questions, grade your answers using a fine-tuned TinyLlama model, and get analytics & recommendations. We'll walk through all major code and model steps below.

---

2 Technology Stack

Layer	Technology	Purpose
UI	**Streamlit 1.35**	Quiz interface and analytics
Backend	**Ollama**	Local LLM inference for Q&A & scoring
Base model	**TinyLlama-1.1B-Chat-v1.0**	Small, chat-tuned model
Finetuning	**PEFT + LoRA + bitsandbytes**	Efficient training
Vector DB	**ChromaDB**	Context chunk similarity search
Embeddings	**MiniLM-L6-v2**	Text chunk vectorizer
PDF Parsing	**PyPDF2**	Text extraction from PDF
Deployment	**GGUF + llama.cpp + Ollama**	Run fine-tuned models locally

---

3 Code Overview

3.1 `qa_generator.py` – Generate Q&A with Ollama

```
from ollama import Client
import os, random, json

SYSTEM_PROMPT = """
You are a tutor. Given textbook content, generate N question-answer-topic
triplets.
Return: [{"question": ..., "answer": ..., "topic": ...}]
```

```
"""

client = Client(host=os.getenv("OLLAMA_BASE_URL", "http://localhost:11434"))

def generate_qa_pairs(context_chunks, n=10):
    context = "\n".join(random.sample(context_chunks, k=8))
    prompt = f"{SYSTEM_PROMPT}\nContext:\n{context}\nGenerate {n} questions."
    response = client.generate(model="tinyllama-qa", prompt=prompt)
    return json.loads(response['response'])
```

3.2 `convert_jsonl.py` – Dataset Conversion

```
import json, argparse

# Load raw Q&A pairs and output JSONL format
parser = argparse.ArgumentParser()
parser.add_argument("--infile")
parser.add_argument("--outfile")
args = parser.parse_args()

data = json.load(open(args.infile))
with open(args.outfile, "w") as out:
    for item in data:
        qa = {"prompt": f"Q: {item['question']}\nA:", "completion":
item['answer']}
        out.write(json.dumps(qa) + "\n")
```

3.3 `fine_tuning.py` – LoRA on Colab (GPU)

```
from transformers import AutoTokenizer, AutoModelForCausalLM, TrainingArguments,
Trainer
from peft import LoraConfig, get_peft_model, TaskType
from datasets import load_dataset
from bitsandbytes import BitsAndBytesConfig

BASE_MODEL = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"

bnb_cfg = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=torch.float16,
)
```

```
model = AutoModelForCausalLM.from_pretrained(BASE_MODEL,
            quantization_config=bnb_cfg, device_map="auto")

peft_cfg = LoraConfig(r=8, lora_alpha=16, lora_dropout=0.05,
            task_type=TaskType.CAUSAL_LM, target_modules=["q_proj", "v_proj"])
model = get_peft_model(model, peft_cfg)

tok = AutoTokenizer.from_pretrained(BASE_MODEL)
tok.pad_token = tok.eos_token
data = load_dataset("json", data_files="dataset.jsonl", split="train")

def tokenize(example):
    return tok(example["prompt"] + example["completion"], truncation=True)

data = data.map(tokenize)

args = TrainingArguments(output_dir="output", per_device_train_batch_size=2,
                         num_train_epochs=3, fp16=True, logging_steps=10)
trainer = Trainer(model=model, args=args, train_dataset=data)
trainer.train()

model.save_pretrained("tinyllama-merged")
tok.save_pretrained("tinyllama-merged")
```

3.4 Convert to GGUF for Ollama

```
python3 llama.cpp/convert_hf_to_gguf.py \
    tinyllama-merged \
    --outfile tinyllama-qa.gguf \
    --outtype q8_0
```

3.5 Ollama Deployment

Create a file `Modelfile`:

```
FROM ./tinyllama-qa.gguf
```

Then run:

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

---

4 End-to-End Workflow Summary

1. 📝Upload PDF → extract text chunks with PyPDF2.
2. 🔍ChromaDB stores embedded chunks for similarity.
3. Q&A generated with TinyLlama via Ollama API.
4. 🧪Student answers questions → scored via LoRA model.
5. 📊Analytics computed from topic-tagged scores.
6. 🔗Fine-tune TinyLlama with Q&A pairs (Colab).
7. 🧱Export LoRA weights → merge → GGUF → Ollama.

---

5 Installation & Execution

Local MacBook (in your repo folder)

```
brew install ollama
python3 -m venv .venv && source .venv/bin/activate
pip install -r requiremrnts.txt
streamlit run app/ui.py
```

Colab Fine-Tuning

```
!pip install transformers datasets peft bitsandbytes accelerate
!python convert_jsonl.py --infile qa_pairs.json --outfile dataset.jsonl
!python fine_tuning.py
```

---

6 Conclusion

You now have a fully integrated system that: - Uses a local LLM (TinyLlama) to generate and grade questions. - Can be fine-tuned with your own data (e.g. Constitution Q&A). - Runs 100% offline after first-time model fetch.

Let me know if you want this turned into a blog, PDF, or GitHub README next! 🔗