AI Question Paper Generator

Technical Documentation

System Name

AI-Powered Question Paper Generator

Stack Overview

- **Backend Framework:** FastAPI
- **LLM:** LLaMA 3 (via Groq API)
- **Cache Layer:** Redis
- **Document Output:** PDF (fpdf), Word (python-docx)
- **Storage:** Local filesystem (with timestamps)

Architecture Overview

```
                    Teacher UI (Frontend)
                            ⬇️
                      FastAPI Backend
           ┏-------------------------------┓
           |   /generate-paper (POST)         |
           |   /download/pdf/{timestamp}      |
           |   /topics, /subjects, /clear-cache|
           ┗-------------------------------┛
                 |      |          |
                 |      |          └───> Groq (LLaMA 3)
                 |      |
      Redis Cache     PDF/Word Generator
          (cache + pool)     (fpdf/docx)

Knowledge Base (JSON)
- Subjects → Grades → Topics
- Objectives and Facts
```

Core Algorithms

1. Question Generation Loop

```
for difficulty, count in difficulty_distribution.items():
    for _ in range(count):
        topic = next(topic_cycle)
        type = next(type_cycle)

        if cached:
            use_cached()
        else:
            kb_data = get_topic_knowledge(...)
            question = generate_from_llm(...)
            cache_question(...)

        if not duplicate:
            add_to_pool(...)
            paper.append(question)
```

2. LLM Prompt Template

```
Generate a {type} question.
Subject: {subject}  Grade: {grade}
Topic: {topic}  Difficulty: {difficulty}
Objectives: {kb_objectives}
Facts: {kb_facts}
```

3. Document Export

- Word: Headings + numbered questions
- PDF: Auto pagination, wrapped text

Components Explained

FastAPI

- All routes and orchestration logic

Generator (Groq)

- Uses LLaMA 3 model via `chat/completions`
- Prompt injected dynamically with context

Redis

- `used_questions:{user_id}`: Prevents duplication
- `question_cache:{topic}:{difficulty}:{type}`: Caches generated questions

Knowledge Base (KB)

- Stored in JSON: `/data/sample_kb.json`
- Contains topics, objectives, and facts
- Used to guide LLM outputs

Document Generator

- `fpdf` for PDFs (paginated)
- `python-docx` for Word
- Files named like `question_paper_YYYYMMDDHHMMSS.pdf`

Scalability

- Redis hosted on Redis Cloud for scaling
- Stateless backend for Docker deployment
- LLM (Groq) supports high-throughput

Suggested Project Structure

```
app/
├── main.py
├── routes.py
├── models/
│   └── schema.py
├── services/
│   └── generator.py
│   └── knowledge_base.py
├── utils/
│   └── caching.py
│   └── document_generator.py
├── data/
│   └── sample_kb.json
├── generated_papers/
```

Output Format Example

```json
{
  "paper": {
    "Question 1": "What is the by-product of photosynthesis?",
    "Question 2": "Define osmosis with an example."
  },
  "download_links": {
    "pdf": "/download/pdf/20250714122000",
    "word": "/download/word/20250714122000"
  }
}
```
