AI Flashcard Generator - Technical Documentation

Tech Stack: Django (Python), HTML/CSS/JS, PyPDF2 (for PDF text extraction), AI (LLM for flashcard generation)

1. System Overview

Purpose

A web application that converts study materials (text or PDFs) into structured flashcards using AI.

Features

✅ Text-to-Flashcard Conversion
✅ PDF/TXT File Upload Support
✅ Responsive & Interactive UI
✅ Drag-and-Drop File Handling
✅ Loading State with Spinner

2. Architecture

Backend (Django)

- views.py
  - Handles form submission (POST requests).
  - Extracts text from PDFs using PyPDF2.
  - Calls generate_flashcards() (AI integration) and splits results into a list.
- forms.py
  - Defines the FlashcardForm with fields:

```
class FlashcardForm(forms.Form):
    subject = forms.CharField()
    input_text = forms.TextField()
    upload_file = forms.FileField()
```

- utils.py
    - Contains helper functions:
        - extract_text_from_pdf(file) → Extracts text from PDFs.
        - generate_flashcards(text, subject) → Calls AI (e.g., OpenAI API) to generate Q&A pairs.

Frontend (HTML/CSS/JS)

- **index.html**
    - Dynamic form with file upload and real-time feedback.
    - Flashcards rendered in a responsive grid.
- **JavaScript**
    - Handles drag-and-drop file uploads.
    - Displays loading spinner during processing.

---

3. Setup Guide

Prerequisites

- Python 3.8+
- Django 4.0+
- PyPDF2 (pip install pypdf2)
- AI API key (if using OpenAI/Gemini)

Installation

1. Clone the repository.
2. Install dependencies:

    ```
    pip install django pypdf2 openai
    ```

3. Configure Django:
    - Add your AI API key in settings.py (if applicable).
    - Run migrations:

    ```
    python manage.py migrate
    ```

4. Start the server:

    ```
    python manage.py runserver
    ```

4. Code Walkthrough

Key Functions

Text Extraction (utils.py)

```
from PyPDF2 import PdfReader


def extract_text_from_pdf(file):
    reader = PdfReader(file)
    text = ""
    for page in reader.pages:
        text += page.extract_text()
    return text
```

Flashcard Generation (utils.py)

```
import openai  # Example using OpenAI


def generate_flashcards(text, subject):
# I first tried to use chatgpt api key but I is paid so I found one more to use
# I install ollama in my system and run it locally then I put api generate in my this function
and generate minimum 15 flashcard
    )


def generate_flashcards(text, subject):
    try:
        # Optimized prompt for 15 flashcards
        prompt = f"""Generate exactly 15 high-quality flashcards about {subject} using this strict
format:
        Q1: [question]
        A1: [concise answer]
        Q2: [question]
```

```
A2: [answer]
...
Q15: [question]
A15: [answer]

Reference text: {text[:1000]}"""

# Try API first
try:
    response = requests.post(
        "http://localhost:11434/api/generate",
        json={
            "model": "llama3:8b",
            "prompt": prompt,
            "stream": False,
            "options": {
                "temperature": 0.7,
                "num_ctx": 2048,
                "num_thread": 6
            }
        },
        timeout=120
    )
    data = response.json()
    if data.get("response"):
        return data["response"]
except requests.exceptions.RequestException:
    pass  # Fall through to CLI method

# Fallback to CLI if API fails
result = subprocess.run(
            ["ollama", "run", "llama3:8b", prompt],
            capture_output=True,
            text=True,
            timeout=180
        )

        if result.stdout:
            return result.stdout
        return "Failed to generate content after multiple attempts"

    except Exception as e:
        return f"System error: {str(e)}"
```

View Logic (views.py**)**

```
def index(request):
    if request.method == "POST":
        form = FlashcardForm(request.POST, request.FILES)
        if form.is_valid():
            text = form.cleaned_data["input_text"]
            file = form.cleaned_data["upload_file"]

            if file:
                if file.name.endswith(".pdf"):
                    text = extract_text_from_pdf(file)
                elif file.name.endswith(".txt"):
                    text = file.read().decode("utf-8")

            flashcards_raw = generate_flashcards(text, form.cleaned_data["subject"])
            flashcards = [card.strip() for card in flashcards_raw.split("\n") if card.strip()]

            return render(request, "index.html", {"form": form, "flashcards": flashcards})
    else:
        form = FlashcardForm()
    return render(request, "index.html", {"form": form})
```

5. Frontend Components

Key UI Elements

1. **File Uploader**
   - Drag-and-drop zone with real-time feedback.
   - Shows selected filename and size.
2. **Flashcard Display**
   - Questions (Q:) and Answers (A:) styled with badges.
   - Hover animations for better interactivity.
3. **Loading State**
   - Dual-ring spinner with progress text.

6.Customization

- Modify Flashcard Prompts
- Edit the prompt in `generate_flashcards()` (e.g., to generate MCQs):

```
prompt = f"Generate 5 multiple-choice questions about {subject} with 4 options each."
```

7. Future Improvements

- **Export Flashcards** → Save as Anki deck (.apkg) or CSV.
- **User Accounts** → Django authentication to save flashcard history.
- **Spaced Repetition** → Algorithm to schedule reviews.

Appendix

- **Dependencies**: requirements.txt

```
django==4.2
pypdf2==3.0.1
openai==0.28
```

- **Repository Structure**:

```
/flashcard_generator
  /templates
  index.html
  views.py
 forms.py
 utils.py
```