Document Q&A Web App using Streamlit + Gemini Flash 1.5 + ChromaDB

1. Objective

The purpose of this project is to build a simple and interactive web application that allows users to:

- Upload documents in PDF, DOCX, or TXT format.
- Ask questions about the uploaded content.
- Get intelligent, accurate answers using Google's Gemini Flash 2.0 large language model.
- Store and retrieve document data using ChromaDB for efficient querying.

This application is ideal for students, researchers, HR departments, and small businesses needing to extract knowledge from documents.

---

2. Tools & Technologies

Technology	Description
Streamlit	Python-based open-source framework to build web apps easily
Gemini Flash 2.0	A lightweight generative model by Google for fast content generation
ChromaDB	An open-source embedding database for storing and retrieving vectorized content
PyPDF2	Python library to extract text from PDF files
python-docx	Library for reading text from Microsoft Word files
google-generativeai	SDK to connect and interact with Gemini API

---

3. Features

- Intuitive drag-and-drop interface for file uploads.
- Supports large files up to 200MB.
- Text extraction from PDF, DOCX, and TXT files.
- Vector-based search from ChromaDB.
- Natural language Q&A using Gemini Flash 2.0.
- Accurate and context-aware answers.

---

4. SDLC (Software Development Life Cycle)

Phase 1: Planning

- **Goal**: Build a QA app that can understand and respond to questions about uploaded documents.
- **Scope**: Single document upload, real-time QA, Streamlit UI.

Phase 2: Requirement Analysis

Functional Requirements:

- Upload different document formats.
- Allow user input for questions.
- Return precise answers.

Non-Functional Requirements:

- Real-time performance.
- Error handling for unsupported file types.
- Secure API key handling.

Phase 3: Design

- **UI Design**:
    - File uploader widget
    - Text input box for questions
    - Display area for answers
- **Component Design**:
    - read_document() to extract text
    - ChromaDB to store embeddings
    - Gemini API for generating answers

Phase 4: Development

- Created using Python in app.py
- Integrated Google Generative AI SDK
- Created embeddings with ChromaDB
- Developed real-time QA pipeline

Phase 5: Testing

- Validated text extraction from PDFs, DOCX, TXT
- Simulated various types of user queries
- Handled errors like missing files, empty input

Phase 6: Deployment

- Can be deployed locally or using Streamlit Cloud, Hugging Face Spaces, or Docker.

Phase 7: Maintenance

- Regular API updates for Gemini Flash.
- Enhancements to support more file formats.
- Add multilingual support and better error messages.

---

5. Project Folder Structure

```
Document_QA_App/
|
├── app.py              # Main application file
├── requirements.txt    # Project dependencies
└── .env                # Environment file for storing secrets like API keys
```

---

6. How It Works

1. User uploads a document.
2. The application extracts text from the file.
3. Text is chunked and converted into embeddings.
4. Embeddings are stored in ChromaDB.
5. User asks a question.
6. ChromaDB retrieves related chunks.
7. Gemini Flash 2.0 uses that context to generate an answer.
8. Answer is displayed on the UI.

---

7. How to Set the API Key

```
import os
import google.generativeai as genai

os.environ["GOOGLE_API_KEY"] = "your_api_key_here"
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])
```

You can also store the key in a .env file and load it using the dotenv package.

---

8. How to Run

1. Install dependencies:

```
pip install -r requirements.txt
```

2. Run the app:

```
streamlit run app.py
```

---

9. Sample Use Case

- **Uploaded File**: Resume.pdf
- **User Question**: "List the projects mentioned in this document"
- **Gemini Response**:

"Developed Smart Waste Bin using IoT and a Personal Fitness Tracker using Machine Learning."

---

10. requirements.txt

```
streamlit
PyPDF2
python-docx
chromadb
google-generativeai
```

---

11. Future Enhancements

- Allow multiple document uploads.
- Export Q&A history.
- Add support for Excel, PPTX formats.
- Voice-based question input.
- UI improvements using Streamlit components.
- Save session history using SQLite or Firebase.