TitleWize

**Strictly private and confidential**
**Monday, 17 March 2025**

**Short Full-Stack Web App Assignment:**
**Content Summary from PDF**

**Objective:**
Build a full-stack web application that accepts a PDF file upload, extracts its text, and returns a concise summary using the Gemini LLM API. This project will assess your ability to design and implement a solution with Django (backend), PostgreSQL (database), and HTML/CSS with Bootstrap (frontend).

**Functional Requirements:**
1. Input Options: PDF Upload
Provide a form for users to upload a PDF file.
Validate that the file is a PDF.
Extract text content from the PDF.

2. Content Summarization
Integrate with the Gemini LLM API to generate a summary from the extracted text.
Gracefully handle API errors or incomplete responses.

3. User Interface
Create a responsive UI using HTML, CSS, and Bootstrap.
Display input forms, loading indicators, and summary results.
Provide clear feedback for invalid inputs, unsupported file types, or processing/API errors.

4. Backend API
Develop Django views (or Django REST Framework endpoints) to:
Handle PDF uploads and validate file types.
Extract text from PDFs.
Call the Gemini LLM API for summarization.
Return summarized content with appropriate HTTP status codes.
Implement robust error handling (e.g., 400 for bad requests, 500 for server errors).

5. Database Integration
Use PostgreSQL to cache PDF content and/or store summarization results.
Design models to record file uploads, extracted text, and summaries.

6. Optional Enhancements

Allow users to choose the summary length (e.g., short, medium, long).
Implement user authentication for history tracking.
Create an admin dashboard for managing PDFs, cached content, and API logs.

**Technical Requirements:**

Frontend:

Build the UI with HTML and CSS using Bootstrap for layout and responsiveness.
Ensure the design is clean, user-friendly, and mobile-responsive.

Backend:

Use Django as your framework (Django REST Framework is optional).
Write clean, modular, and well-documented code.

Database:

Utilize PostgreSQL for storing cached data and summarization results.
Support necessary migrations and smooth database operations.

LLM API Integration:

Integrate the Gemini LLM API for content summarization.
Include instructions in your README for obtaining and setting up API keys.

**Evaluation Criteria**

1. Functionality:
Correct implementation of PDF upload, validation, and text extraction.
Successful integration with the Gemini LLM API.
Robust error handling and input validation.

2. Code Quality:
Clean, modular, and well-documented code.
Clear separation between frontend and backend logic.
Proper utilization of Django and PostgreSQL.

3. User Experience:
Responsive, intuitive UI with clear feedback mechanisms (errors, loading states).

4. Documentation:
Comprehensive README detailing the solution, setup, deployment, design decisions, and API key instructions.

5. Creativity & Initiative:
Additional features or enhancements that improve user experience or demonstrate advanced technical skills.

**Submission Guidelines**

Submit your solution via a public repository (e.g., GitHub, GitLab) with a clear commit history.
Include a README with all necessary instructions to run the application locally.