Design Document & Approach Strategy:

AI Excel Mock Interviewer

1. Overview & Purpose

This document outlines the design and strategic approach for the "Excel Mock Interview Agent," a web application built with Streamlit and powered by the Google Gemini API.

The primary purpose of the application is to provide users with a simulated, interactive interview experience to assess their proficiency in Microsoft Excel. It aims to offer a realistic, structured, and educational tool for job seekers preparing for roles that require Excel skills. The application asks a series of questions with progressively increasing difficulty, evaluates the user's answers, and provides a comprehensive, personalized feedback report.

---

2. System Architecture

The application follows a simple three-tier architecture:

Frontend (Presentation Layer):

- **Technology:** Streamlit.
- **Responsibility:** Renders the user interface, captures user input, and displays questions and feedback. The entire user experience, from the introduction to the final report, is managed here. Custom CSS is injected to enhance the visual design and create a professional, "agent-like" theme.

Backend (Application Logic Layer):

- **Technology:** Python.
- **Responsibility:** Manages the application's state and interview flow. This includes orchestrating the sequence of questions,

handling user navigation (save, skip, next), processing answers, and calculating scores. It's tightly integrated with the Streamlit frontend within the same script.

AI Service Layer (External Dependency):

- **Technology:** Google Gemini API.
- **Responsibility:** Provides the core intelligence. It is used for three distinct tasks:
    - Generating dynamic, context-aware interview questions.
    - Evaluating the technical accuracy and completeness of user answers.
    - Synthesizing a comprehensive final feedback report based on the entire interview session.

Interaction Flow: User -> Streamlit Frontend -> Python Backend Logic -> Google Gemini API -> Python Backend Logic -> Streamlit Frontend -> User

---

3. Core Features & Functionality

The application's core features include a **Progressive Difficulty Interview**, which is a structured 5-question flow from "Basic" to "Advanced" topics. **Dynamic Question Generation** uses the Gemini API to make each interview unique. It includes robust **State Management** to maintain user progress and answers throughout the session. Users have **Flexible User Navigation**, allowing them to save answers, skip questions, or mark them for review. The app provides **AI-Powered Evaluation** for intelligent feedback and concludes with a **Comprehensive Final Report** summarizing performance and offering recommendations. A key feature is the **Robust Fallback System**, which uses pre-defined content if the API fails, ensuring the application remains functional.

---

4. Approach Strategy

The development and design of this application are guided by the following strategic principles:

- **Modular & Function-Oriented Design:** The code is organized into distinct functions, each with a single responsibility (e.g., generate_excel_question, evaluate_answer). This separation of concerns makes the code easier to read, maintain, and debug.
- **API-First with Graceful Degradation:** The primary strategy is to leverage the advanced capabilities of the Gemini LLM for the best user experience. However, a critical design choice was to build a robust fallback mechanism to ensure the application remains functional even in cases of API failure.
- **Prompt Engineering as a Core Component:** The quality of the application is directly tied to the quality of the prompts sent to the Gemini API. The strategy involves role-playing, context setting, demanding structured output, and providing few-shot examples to guide the model.
- **Stateful User Experience:** By heavily utilizing st.session_state, the application remembers every detail of the user's journey to create a coherent and interactive session.
- **User-Centric Flow:** The entire application flow is designed from the user's perspective, with a clear introduction, progress indicators, intuitive navigation, and a valuable, actionable report.

---

5. Component Breakdown (Key Functions)

- **main():** Entry point of the application. Acts as the main controller, directing the user through the different stages.

- **initialize_session_state():** Sets up all the necessary session variables at the start.
- **call_gemini_api():** A centralized wrapper for all API calls, encapsulating request logic and error handling.
- **generate_excel_question():** Constructs a detailed prompt for the AI to generate a question and calls the fallback function if the API call fails.
- **evaluate_answer():** Constructs a prompt to evaluate a user's answer and implements a keyword-based scoring algorithm as its fallback.
- **generate_final_report():** Compiles the entire interview transcript into a single prompt for the AI to generate the final report.
- **display_*() functions:** A set of functions responsible for rendering specific UI components.

---

6. Security Considerations

- **API Key Management:** The current code hardcodes the GEMINI_API_KEY. This is a major security risk. In a production environment, this key must be removed from the source code and managed securely using environment variables or a secrets management tool (like Streamlit's built-in secrets management).
- **Input Sanitization:** User inputs are sent to an external API. For a production system, it would be best practice to sanitize inputs to prevent potential prompt injection attacks.