Project Plan: AI-Based Virtual Interview Assistant

1. Proposed Technical Architecture

This architecture is designed for scalability and leverages our choice of GCP, integrating the specified technologies.

- **Frontend:** Next.js (Our current stack). Will handle the user interface, real-time communication (mic, camera), and interaction with the backend. We will use a library like `MediaPipe.js` for in-browser facial analysis.
- **Backend:** FastAPI (Our current stack). Will serve as the central hub, managing user authentication, interview logic, communication with AI services, and database operations.
- **Database:** PostgreSQL (Hosted on GCP Cloud SQL). This relational database will store user data, interview roles, questions, session history, and feedback reports.
- **Deployment:** Docker / Docker Compose (Our current stack). The entire application will be containerized and deployed to **GCP Cloud Run** for scalable, serverless execution.
- **Core AI Services:**
  - **Avatar Interaction: HeyGen Streaming Avatar API**. We will use their SDK to stream the avatar to the frontend and send text prompts for the avatar to speak in real-time.
  - **Language Models: GCP Vertex AI (Gemini Models)**. For dynamic question generation, response analysis, scoring, and feedback generation.
  - **Speech-to-Text: Google Cloud Speech-to-Text API**. To transcribe user's spoken answers in real-time.

---

2. Phased Development Roadmap

We will break the project into four distinct phases.

Phase 0: Foundation & Setup (Current State + Next Steps)

This phase expands on our existing codebase to prepare it for feature development.

- **What:**
  - Integrate PostgreSQL with the FastAPI backend using an ORM (Object-Relational Mapper) like SQLAlchemy.
  - Implement a robust User Authentication system (e.g., JWT-based). This includes signup, login, and protected API endpoints.
  - Define the core database schemas for `Users`, `InterviewRoles`, and `InterviewSessions`.
  - Set up the basic project structure in Next.js for different pages: Login, Dashboard, Interview Room.
- **Why:** To establish a secure and scalable foundation. Without user accounts and a database, we cannot save progress or provide personalized experiences.

Phase 1: The Core Interview Loop (Minimum Viable Product)

The goal of this phase is to have a user complete a full, albeit basic, interview.

- **What:**
  - **Role & Difficulty Selection:** Create a UI where users can select one of the 10 specified roles (e.g., "Engineering - Python Developer") and a difficulty level (e.g., "Junior").
  - **Static Question Engine:** For the MVP, questions will be pulled from our PostgreSQL database for the selected role.
  - **Text-Based Interview:** Implement a chat-like interface for the user to type their answers.
  - **Static Avatar Display:** The avatar will be present visually but will not be interactive yet. The questions will appear as text on the screen.
  - **AI-Powered Content Analysis:** The backend will send the user's text answer to the Gemini API with a prompt designed to evaluate its correctness, relevance, and clarity.
  - **Post-Interview Report v1:** A simple summary page will be generated at the end, displaying the questions, the user's answers, and the AI-generated score/feedback for each.
- **Why:** This phase focuses on validating the most critical part of the application: the question-answering and AI evaluation loop, without the added complexity of real-time streaming.

Phase 2: Introducing Voice & Avatar Interactivity

This phase brings the assistant to life, focusing on a more immersive and natural user experience.

- **What:**
  - **HeyGen Streaming Integration:** Integrate the HeyGen Streaming Avatar SDK into the Next.js frontend. The backend will now send the question text to the frontend, which then instructs the avatar to speak it in real-time.
  - **Speech-to-Text Integration:** Implement voice input. The frontend will capture microphone audio, stream it to the backend, which will then use the Google Speech-to-Text API to get a transcription.
  - **Real-time UI Updates:** The UI will show the live transcription as the user speaks.
  - **Non-Intrusive Feedback UI:** Design and implement the side/bottom panel for displaying optional, real-time feedback snippets after a user finishes answering a question.
  - **Detailed Dashboard v1:** Enhance the post-interview report into a more detailed dashboard, visualizing the scores from Phase 1.
- **Why:** To deliver the core promise of an interactive, voice-driven mock interview with an AI avatar, which is a major user value proposition.

Phase 3: Advanced Voice & Facial Analysis

This phase adds the deep analytical capabilities that will set this tool apart.

- **What:**
  - **Facial Emotion Recognition:** Integrate `MediaPipe.js` directly into the Next.js frontend. The user's webcam feed will be processed *in the browser* to detect facial landmarks and expressions (e.g., confidence, engagement). This data (not the video itself) will be sent to the backend.
  - **Voice Tone & Pace Analysis:** Enhance the backend to analyze the detailed output from the Speech-to-Text API (which includes word timings). This will allow us to calculate the user's speaking pace, hesitation, and usage of filler words.

    - **Complex Scoring Model:** Upgrade the AI evaluation logic. The backend will now combine the content analysis (from Phase 1) with the new facial expression and voice analysis data to generate a multi-dimensional score.
    - **Dashboard v2:** Greatly enhance the feedback dashboard with new visualizations for:
        - Confidence levels over time (from facial analysis).
        - Speaking pace and filler word count per answer.
        - A timeline view of the interview with markers for key events.
- **Why:** To provide users with comprehensive, actionable feedback not just on *what* they said, but *how* they said it, which is crucial for real-world interview success.

Phase 4: Scaling, Intelligence & Polish

This phase focuses on making the platform smarter, more robust, and ready for a wider audience.

- **What:**
    - **Dynamic Question Generation (RAG):** Implement a Retrieval-Augmented Generation system. We will build a high-quality vector database of interview questions. The AI will dynamically select and adapt questions from this database based on the user's previous answers and performance, making each interview unique.
    - **User History & Progress Tracking:** Develop a user dashboard where individuals can view their past interviews, compare scores, and track their improvement over time.
    - **Cloud Deployment & Optimization:** Finalize the deployment configuration on GCP Cloud Run, ensuring auto-scaling, proper security, and performance monitoring.
    - **Extensive Role Library:** Expand the database with more questions and roles to cater to all three target audiences (students, professionals, and companies).
- **Why:** To ensure the platform is scalable, provides long-term value to users, and delivers a consistently high-quality, intelligent experience.
