**AI-Powered Interview System Documentation**

**Overview**

The system is designed to automate the recruitment process by allowing users to:

- Sign up and sign in using Supabase authentication.
- Create, manage, and apply for job postings.
- Upload and process CVs (PDF format) using Google Gemini for parsing and generating tailored interview questions.
- Conduct live interviews via WebSocket with real-time audio transcription and evaluation.
- Store and manage data (users, jobs, applicants, CVs, interviews) in Supabase.
- Log and evaluate interview responses for sentiment, clarity, confidence, and relevance.

The codebase is modular, with separate files handling specific functionalities such as CV processing, database interactions, interview simulation, and logging.

---

**File Structure and Purpose**

1. **cv.py**: Handles CV processing and interview question generation using Google Gemini.
2. **database.py**: Configures the Supabase client for database and storage operations.
3. **interview_agent.py**: Manages the interview process, including question delivery, audio transcription, and response evaluation.
4. **evaluation_agent.py**: Evaluates candidate responses for sentiment, clarity, confidence, and relevance.
5. **interview_simulator.py**: Simulates a client-side interview experience with audio recording and playback.
6. **logger_agent.py**: Logs interview transcripts, evaluations, and media for storage in Supabase.
7. **models.py**: Defines Pydantic models for data validation (e.g., user signup, job creation).

8. **main.py**: Defines FastAPI routes for user management, job management, CV processing, and live interviews.
9. **auth.py**: Handles user authentication using Supabase JWT tokens.

---

**Detailed File Documentation**

**1. cv.py**

**Purpose**: Processes CV files (PDF or DOCX) to extract structured data and generate tailored interview questions using Google Gemini.

**Key Functions**:

- extract_text_from_pdf(pdf_path): Extracts text from a PDF file using pdfplumber.
- extract_text_from_docx(docx_path): Extracts text from a DOCX file using docx2txt.
- parse_cv_with_gemini(cv_text): Uses Google Gemini to parse CV text into structured JSON with fields: name, email, phone, education, experience, skills.
- process_cv(file_path, file_type="pdf"): Orchestrates CV text extraction and parsing, ensuring valid JSON output.
- generate_interview_questions(cv_data, job_title="software developer"): Generates 5 tailored interview questions based on CV data and job title, focusing on coding skills, data structures, problem-solving, and tools.

**Dependencies**:

- fastapi, google.generativeai, pdfplumber, docx2txt, json, re

**Notes**:

- The Google API key is hardcoded (replace with environment variables in production).
- The system assumes CVs are primarily in PDF format.
- JSON parsing errors are handled with detailed error messages.

---

**2. database.py**

**Purpose**: Initializes the Supabase client for database and storage operations.

**Key Components**:

- **Constants**:
    - SUPABASE_URL: Supabase project URL.
    - SUPABASE_KEY: Anonymous API key for public access.
- **Client**:
    - supabase: A Client instance created using create_client(SUPABASE_URL, SUPABASE_KEY).

**Dependencies**:

- supabase

**Notes**:

- The anon key is hardcoded (use environment variables in production).
- The client is used across other modules for database operations (e.g., users, jobs, cvs, interviews tables).

---

**3. interview_agent.py**

**Purpose**: Manages the live interview process, including question delivery, audio transcription, response evaluation, and logging.

**Key Components**:

- **Class**: InterviewAgent
    - **Attributes**:
        - applicant_id: Unique identifier for the applicant.
        - interview_id: Unique identifier for the interview (generated if not provided).
        - logger: LoggerAgent instance for logging transcripts and evaluations.
        - evaluator: EvaluationAgent instance for analyzing responses.
        - questions: List of interview questions fetched from the cvs table.
        - parsed_info: Processed CV data from the cvs table.

        - job_title: Job title associated with the applicant's job application.
    - **Methods**:
        - __init__(applicant_id, interview_id): Initializes the agent, fetches applicant data, and creates an interview record in Supabase.
        - start_interview(websocket): Runs the interview loop, sending questions, streaming audio, transcribing responses, evaluating answers, and logging results.
        - text_to_speech(text): Generates TTS audio for questions using gTTS.

**Dependencies**:

- uuid, datetime, asyncio, numpy, sounddevice, gtts, whisper_module, evaluation_agent, logger_agent, database

**Notes**:

- Uses WebSocket for real-time communication with the client.
- Assumes the whisper_module (not provided) handles audio transcription.
- Interview status is updated to Completed upon finishing.

---

**4. evaluation_agent.py**

**Purpose**: Analyzes candidate responses during interviews using Google Gemini.

**Key Components**:

- **Class**: EvaluationAgent (inherits from crewai.Agent)
    - **Attributes**:
        - name: Agent name (EvaluationAgent).
        - role: Role description (Interview Evaluator).
        - goal: Objective to evaluate responses for sentiment, clarity, confidence, relevance, and quality.
    - **Methods**:
        - __init__(name): Initializes the agent with its role and goal.

        - analyze_response(transcribed_text, job_title, parsed_info): Analyzes a response and returns a JSON object with:
            - sentiment: Positive/Neutral/Negative.
            - clarity: Score (1–10).
            - confidence: Score (1–10).
            - relevance: Description of relevance to the job.
            - summary: Key points from the response.
            - score: Overall quality score (1–10).

**Dependencies**:

- json, crewai, cv (for Gemini model), re

**Notes**:

- Handles JSON parsing errors gracefully with fallback error messages.
- Relies on the Gemini model from cv.py for response analysis.

---

**5. interview_simulator.py**

**Purpose**: Simulates a client-side interview experience, handling audio recording, TTS playback, and WebSocket communication.

**Key Functions**:

- login_user(): Authenticates the user via the /signin endpoint and returns a JWT token.
- record_audio(duration=30, fs=16000): Records audio from the microphone for 30 seconds and saves it as a WAV file.
- play_tts(tts_path): Plays TTS audio files using playsound.
- start_interview(token, interview_id): Connects to the WebSocket endpoint, receives questions, plays TTS, records responses, and displays evaluations.
- get_tts(token, interview_id): Fetches TTS audio for the current question via the /tts endpoint.

**Dependencies**:

- requests, json, websocket, base64, sounddevice, soundfile, os, time, playsound, pathlib

**Notes**:

- Hardcoded credentials and interview ID (replace with dynamic inputs in production).
- Assumes a local FastAPI server running at http://127.0.0.1:8000.
- Cleans up temporary audio files after use.

---

**6. logger_agent.py**

**Purpose**: Logs interview data (transcripts, evaluations, media) and saves it to Supabase.

**Key Components**:

- **Class**: LoggerAgent
    - **Attributes**:
        - interview_id: Unique identifier for the interview.
        - session_data: Dictionary storing transcripts, evaluations, and media.
    - **Methods**:
        - __init__(interview_id): Initializes the logger with a session data structure.
        - log_transcript(question, answer_text): Logs a question-answer pair with a timestamp.
        - log_evaluation(evaluation): Logs an evaluation result with a timestamp.
        - log_media_file(file_type, file_url): Logs media (e.g., audio files) with a timestamp.
        - generate_interview_summary(): Creates a summary of the interview (transcripts and evaluations).
        - save_to_database(): Updates the interviews table with logged data.

**Dependencies**:

- uuid, datetime, database

**Notes**:

- Ensures all logged data is timestamped for traceability.
- Handles database errors with appropriate logging.

---

**7. models.py**

**Purpose**: Defines Pydantic models for data validation across API endpoints.

**Key Models**:

- UserSignUp: Validates user signup data (email, password, role).
- UserSignIn: Validates user signin data (email, password).
- JobCreate: Validates job creation data (title, brief, requirements, status).
- JobUpdate: Validates job update data (all fields optional).
- ApplicantRequest: Validates applicant data (user_id).
- CVUpdate: Validates CV update data (applicant_id, file_url, processed_data, processing_status).

**Dependencies**:

- pydantic, typing, json

**Notes**:

- Uses Pydantic's strict validation to ensure API inputs are correct.
- Optional fields are supported for flexible updates.

---

**8. main.py**

**Purpose**: Defines the FastAPI application with routes for user management, job management, CV processing, and live interviews.

**Key Endpoints**:

- **User Management**:

    - POST /signup: Registers a new user with Supabase Auth and stores user data.
    - POST /signin: Authenticates a user and returns a JWT token.
    - GET /users: Retrieves all users (requires authentication).
- **Job Management**:
    - POST /jobs: Creates a new job posting.
    - GET /jobs: Retrieves all jobs.
    - GET /jobs/{job_id}: Retrieves a specific job's details.
    - DELETE /jobs/{job_id}: Deletes a job.
    - PUT /jobs/{job_id}: Updates a job's details.
- **Applicant Management**:
    - POST /jobs/{job_id}/applicants: Adds an applicant to a job.
    - DELETE /jobs/{job_id}/applicants/{applicant_id}: Removes an applicant.
    - GET /applicants/{applicant_id}/history: Retrieves an applicant's interview history.
- **CV Management**:
    - POST /applicants/{applicant_id}/cv: Uploads and processes a CV, storing it in Supabase Storage.
    - DELETE /applicants/{applicant_id}/cv/{cv_id}: Deletes a CV.
    - PUT /applicants/{applicant_id}/cv/{cv_id}: Updates CV metadata.
- **Interview Management**:
    - POST /applicants/{applicant_id}/interviews: Schedules a new interview.
    - GET /interviews/{interview_id}: Retrieves interview details.
    - GET /applicants/{applicant_id}/interviews/results: Retrieves interview results.
- **Live Interview**:

- WEBSOCKET /interviews/{interview_id}/live: Handles real-time interview interactions.
- POST /interviews/{interview_id}/tts: Generates TTS audio for interview questions.

**Dependencies**:

- fastapi, json, supabase, models, auth, cv, interview_agent, asyncio, base64, os, gtts

**Notes**:

- Uses Depends(get_current_user) for authentication on protected routes.
- Stores CVs in Supabase Storage and metadata in the cvs table.
- Handles errors with appropriate HTTP status codes and messages.

---

**9. auth.py**

**Purpose**: Handles user authentication using Supabase JWT tokens.

**Key Functions**:

- get_current_user(request): Extracts and validates the JWT token from the Authorization header, returning user data from the users table.

**Dependencies**:

- fastapi, supabase, database

**Notes**:

- Raises 401 errors for invalid or missing tokens.
- The commented-out get_current_admin function suggests planned admin role support.

---

**System Workflow**

1. **User Authentication**:
    - Users sign up or sign in via /signup or /signin endpoints, receiving a JWT token.

2. **Job and Applicant Management**:
    - Admins create jobs via /jobs.
    - Users apply to jobs via /jobs/{job_id}/applicants.
3. **CV Processing**:
    - Users upload CVs via /applicants/{applicant_id}/cv.
    - The CV is processed using cv.py, parsed by Google Gemini, and stored in Supabase.
    - Interview questions are generated and stored with the CV.
4. **Interview Scheduling**:
    - Interviews are scheduled via /applicants/{applicant_id}/interviews.
5. **Live Interview**:
    - The client connects to the WebSocket endpoint /interviews/{interview_id}/live.
    - InterviewAgent delivers questions, generates TTS, transcribes responses (via whisper_module), and evaluates answers using EvaluationAgent.
    - LoggerAgent logs transcripts and evaluations, saving them to Supabase.
6. **Results and History**:
    - Interview results are retrieved via /applicants/{applicant_id}/interviews/results.
    - Applicant history is available via /applicants/{applicant_id}/history.

---

**Database Schema (Inferred from Code)**

The system uses Supabase with the following tables:

- **users**:
    - id (UUID): Unique user ID from Supabase Auth.
    - email (string): User's email.
    - password (string): Plaintext password (should be hashed in production).

    - role (string): User role (user or admin).
- **jobs**:
    - id (UUID): Unique job ID.
    - title (string): Job title.
    - brief (string): Short job description.
    - requirements (string): Job requirements.
    - status (string): Job status (Draft, etc.).
- **applicants**:
    - id (UUID): Unique applicant ID.
    - user_id (UUID): Foreign key to users.id.
    - job_id (UUID): Foreign key to jobs.id.
- **cvs**:
    - id (UUID): Unique CV ID.
    - applicant_id (UUID): Foreign key to applicants.id.
    - file_path (string): Public URL to CV in Supabase Storage.
    - processed_data (JSONB): Parsed CV data (e.g., name, skills).
    - interview_questions (JSONB): List of generated interview questions.
- **interviews**:
    - id (UUID): Unique interview ID.
    - applicant_id (UUID): Foreign key to applicants.id.
    - job_id (UUID): Foreign key to jobs.id.
    - status (string): Interview status (Pending, Completed).
    - transcripts (JSONB): List of question-answer pairs.
    - results (JSONB): Evaluation results.

**Storage**:

- CVs are stored in the cvs bucket in Supabase Storage.