Voice-Driven Banking via Large Acoustic Models (LAMs)

SET UP Environment variable :

This project uses a `.env` file to manage all secret keys and environment-specific configurations.

In the `Backend/` directory, find the `.env.example` file. Make a copy of this file and rename it to `.env`. Fill in the values in the `.env` file as described below.Environment Variables (.env)

- **`FIRESTORE_PROJECT_ID`**
  - **Description:** The unique Project ID for your Firebase project.
  - **Where to find it:** In your Firebase project settings (click the gear icon ⚙️).
  - **Used in:** `services/firestore_db.py`
- **`FIRESTORE_SERVICE_ACCOUNT_PATH`**
  - **Description:** The local path to the JSON key file for your Firebase service account.
  - **Where to find it:** Generate a new private key in Firebase Settings -> Service accounts. Place the downloaded JSON file in a `config` folder.
  - **Example:** `./config/your-key-file.json`
  - **Used in:** `services/firestore_db.py`
- **`GCP_PROJECT_ID`**
  - **Description:** The Project ID for your Google Cloud Platform project, used for the paid-tier Gemini API.
  - **Where to find it:** From the dashboard of your Google Cloud Console.
  - **Used in:** `services/llm_gemini.py` (Vertex AI version)
- **`GCP_LOCATION`**
  - **Description:** The Google Cloud region where your Vertex AI models are deployed.
  - **Example:** `us-central1`
  - **Used in:** `services/llm_gemini.py` (Vertex AI version)
- **`HUGGING_FACE_TOKEN`**
  - **Description:** Your Hugging Face access token, required to download gated models like Whisper, Llama 3, and Mistral.
  - **Where to find it:** Generate a new token with "read" permissions in your Hugging Face account settings.
  - **Used in:** Any service that loads a gated model from Hugging Face (`stt_whisper.py`, `tts_hf.py`).
- **`SENDER_EMAIL`**
  - **Description:** The Gmail address used to send OTP emails for the money transfer feature.
  - **Example:** `your.email@gmail.com`

    - **Used in:** `services/email_service.py`
- **`EMAIL_APP_PASSWORD`**
    - **Description:** A 16-digit Google App Password for your `SENDER_EMAIL`. This is NOT your regular password.
    - **Where to find it:** Generated from your Google Account's security settings (requires 2-Step Verification).
    - **Used in:** `services/email_service.py`
- **`APP_ID`**
    - **Description:** An identifier used to separate data from different environments (e.g., "dev", "prod") within the same Firestore database.
    - **Example:** `default-app-id`
    - **Used in:** `services/firestore_db.py`, `services/firestore_session.py`

Detailed Configuration Guide

This guide provides a step-by-step process for setting up all the external services and credentials required to run the application.

A. Detailed Firestore Setup & Data Population

This guide provides a step-by-step process to create a new Firebase project, configure the necessary credentials, and populate it with the required mock data to run the application.

1. **Setting Up Firebase/Firestore:**
    - **Create a Firebase Project:**
        - Go to the Firebase Console.
        - Click "Add project" and give your project a name (e.g., "voice-banking-assistant").
        - Follow the on-screen instructions to create the project.
    - **Create a Firestore Database:**
        - From your project's main dashboard, go to the "Firestore Database" section in the left-hand menu.
        - Click "Create database".
        - When prompted, select "Start in test mode". This will allow your application to read and write to the database during development. You can secure this later with security rules.
        - Choose a server location (e.g., `us-central`) and click "Enable".
    - **Generate a Service Account Key:**
        - This key is a JSON file that allows your backend to securely authenticate with your Firebase project.
        - Click the gear icon ⚙️ next to "Project Overview" in the top-left and select "Project settings".

        - Go to the "Service accounts" tab.
        - Click the "Generate new private key" button. A warning will appear; click "Generate key" to confirm.
        - A `.json` file will be downloaded to your computer.
        - In your `Backend` project folder, create a new folder named `config`.
        - Move the downloaded `.json` file into this `config` folder.
        - **Important:** Add the path to this key file to your `.gitignore` file to avoid accidentally committing it to your repository.
2. **Configuring the `.env` File:**
    - Now, you'll use the information from your new Firebase project to configure your application.
    - In the `Backend` directory, make a copy of `.env.example` and rename it to `.env`.
    - Open the `.env` file and fill in the following values:
        - `FIRESTORE_PROJECT_ID`: You can find this in your Firebase Project settings. It's the unique ID for your project (e.g., `voice-banking-fd409`).
        - `FIRESTORE_SERVICE_ACCOUNT_PATH`: This is the path to the JSON key file you just downloaded.
            - **Example:** `./config/voice-banking-fd409-firebase-adminsdk-...json`
3. **Running the Population Script:**
    - This script will use your credentials to connect to your new database and create the necessary collections and documents.
    - Make sure your virtual environment is activated.
    - From your `Backend` directory, run the script:
      python data_populate.py
    - You will see log messages in your terminal confirming the creation of a test user, accounts, and transactions.
    - You can then go to your Firebase Console and view the `artifacts` collection to see the newly created data.
    - This script will automatically:
        - Create a test user named "Vickey kumar".
        - Add a "Savings" and a "Current" account for this user.
        - Add several random transactions to the "Savings" account.
    - This mock data corresponds to the hardcoded `user_id` used in `main.py`, allowing you to test the `check_balance` and `list_transactions` features immediately.

B. Google Cloud / Vertex AI (for Gemini Pro)

To get production-level access to the Gemini Pro model without the low free-tier limits, you must use a Google Cloud project with billing enabled.

1. **Set Up Your Google Cloud Project:**
    - Go to the Google Cloud Console.
    - Create a new project or select an existing one.
    - Enable Billing for that project.
    - Go to the API Library and search for "Vertex AI API". Click `Enable`.
2. **Authenticate Your Local Machine:**
    - Install the Google Cloud CLI (gcloud) if you don't have it.
    - Run the following command and follow the browser pop-up to log in to your Google account:
      gcloud auth login
    - Run this command to set up Application Default Credentials, which your Python script will automatically find:
      gcloud auth application-default login
3. **Update Your `.env` File:**

Add your Google Cloud Project ID and desired location to the `.env` file:
GCP_PROJECT_ID="your-gcp-project-id-here"

- GCP_LOCATION="us-central1"

C. Hugging Face (for Local Models)

A Hugging Face token is required to download "gated" models like Whisper and Mistral.

Steps to obtain the token and accept model terms:

- **Accept Model Terms:**
    - Visit the model pages and accept their respective license terms.
    - For example, accept the terms for Mistral 7B Instruct.
- **Accept Terms for TTS and STT Models:**
    - Accept the terms for the Text-to-Speech (TTS) model.
    - Accept the terms for the Speech-to-Text (STT) Whisper model.

D. Generate an App Password for Gmail:

- Go to your Google Account Security settings.
- Under "How you sign in to Google," make sure "2-Step Verification" is **On**. If it's off, you'll need to enable it first.
- Once 2-Step Verification is enabled, you'll see "App passwords." Click on it.
- You may be asked to re-enter your Google password.
- From the "Select app" dropdown, choose "Mail."
- From the "Select device" dropdown, choose "Other (Custom name)" and give it a name like "Voice Banking App."
- Click "Generate." A 16-character password will be displayed in a yellow bar. Copy this password immediately, as you won't be able to see it again.
- **Used in:** services/email_service.py

Final Enviroment variable look like this :

1. Firestore Database

- `FIRESTORE_PROJECT_ID`: Your unique Firestore project identifier.
- `FIRESTORE_SERVICE_ACCOUNT_PATH`: The file path to your service account key (e.g., `./config/your-service-account-key.json`).

2. Google Cloud Platform (for Vertex AI - Gemini Pro)

- `GCP_PROJECT_ID`: Your Google Cloud project identifier.
- `GCP_LOCATION`: The geographical region for your GCP services (e.g., `us-central1`).

3. Hugging Face (for Gated Models)

- `HUGGING_FACE_TOKEN`: Your personal Hugging Face access token, required for models like Whisper, Mistral, and Llama (e.g., `hf_YourHuggingFaceTokenHere`).

4. Email Service (for OTP)

- `SENDER_EMAIL`: The email address used to send One-Time Passwords (e.g., `your.email@gmail.com`).
- `EMAIL_APP_PASSWORD`: Your 16-digit application-specific password for the sender email.

5. Application ID (for Firestore Multi-Tenancy)

- `APP_ID`: The identifier for your application, used to support multi-tenancy within Firestore (e.g., `default-app-id`).