Receptionist Robot App: Frontend Guide

Hello Frontend Team!

This document explains how our Receptionist Robot works and how your mobile app will connect to its "brain" (our backend API). Think of your app as the robot's face, ears, and voice – it shows what the robot says, lets people talk to it, and displays what the robot "sees."

1. Understanding the Robot's Workflow (The Story)

Imagine a visitor walking up to the robot. Here's what happens, step-by-step:

1. **Waiting for a Visitor:** The robot stands by, waiting.
2. **Visitor Arrives & Stays:** If someone stands in front of the robot for a few seconds, looking at it, the robot notices them.
3. **Checking Who They Are:**
    - The robot tries to recognize the person's face.
    - **If Known:** The robot says, "Hello, I think you're [Name]. Is that right?" It waits for the person to say "Yes" or "No." If "No," it tries a few more guesses.
    - **If Unknown:** The robot says, "Welcome! I don't recognize you. Please tell me my name after the beep." It also takes a photo.
4. **Learning a New Name (for Unknown Visitors):**
    - The visitor says their name. Your app will turn their speech into text and send it to the robot.
    - The robot shows the name it heard and the photo it took, asking, "Is this you and your name correct?"
    - If "Yes," the robot remembers them for next time. If "No," it asks them to try again.
5. **Who Do You Want to Meet?:**
    - Once the robot knows who the visitor is, it asks, "Whom would you like to meet? Please say their name."
    - The visitor says a name. Your app sends this to the robot.
    - The robot tries to match the name to a company official. It might show a list of similar names and ask, "Did you mean [Name]?"
6. **Contacting the Official:**
    - Once the official's name is confirmed, the robot sends a message (with the visitor's photo) to that official.
    - The robot then tells the visitor, "Please wait while I inform [Official's Name]."
7. **Waiting for a Reply:**
    - The robot waits for the official to reply.

    - **If Official Replies (Approve/Deny):** The robot tells the visitor the official's response (e.g., "Great news! You're approved!").
    - **If No Reply in 3 Minutes:** The robot tries to contact the official up to 3 times.
    - **After 3 Tries (No Reply):** The robot tells the visitor, "I'm afraid [Official's Name] is busy. Would you like to wait, leave a message, or leave?"
8. **Visitor's Choice (If Official Unreachable):**
    - **Wait:** The robot keeps waiting.
    - **Leave Message:** The robot asks, "Please say your message." Your app will turn their speech into text and send it to the robot, which then forwards it to the official.
    - **Leave:** The robot says goodbye.
9. **Visitor Leaves:** If the visitor walks away at any point, the robot notices they're gone and resets, ready for the next person.

2. How Your App Talks to the Robot (The API Endpoints)

Your app will communicate with the backend using these specific "channels" or "endpoints."

API Base URL: http://<YOUR_BACKEND_IP_ADDRESS>:8000

You can see all these details, including exact data formats, at:
Swagger UI (Interactive Docs): http://<YOUR_BACKEND_IP_ADDRESS>:8000/docs

2.1. GET / (Check if Robot is Awake)

- **What it does:** Just asks the robot, "Are you there?"
- **When to use:** When your app first starts, to make sure the backend is running.
- **What you send:** Nothing.
- **What you get back:** A message like {"message": "Robot is running...", "action": "display_message", "state": "IDLE", ...}.

2.2. POST /process_frame (Show Robot What It Sees)

- **What it does:** You send a single picture (frame) from the camera to the robot. The robot uses this to detect faces and decide what to do next.
- **When to use:** Continuously, from your app's live camera feed (e.g., 5-10 times per second).
- **What you send:** An image file (JPEG or PNG) from the camera. Make sure the "Content-Type" is set correctly (e.g., image/jpeg).
- **What you get back:** A RobotResponse (see below) telling you the robot's new message and what to do next.

2.3. POST /submit_input (Tell Robot What the User Says/Chooses)

- **What it does:** You send text from the user (either typed or from their speech) or their choice from a list of options.
- **When to use:** Whenever the robot asks the user a question and expects an answer (e.g., "What's your name?", "Yes/No", "Wait/Leave Message").
- **What you send:** A JSON message like {"input_text": "the user's answer here"}.
- **What you get back:** A RobotResponse with the robot's next message and action.

2.4. POST /official_response (Simulate Official's Reply)

- **What it does:** This endpoint is for a separate system (like an official's app, or for your testing) to tell the robot if an official approves or denies a visitor.
- **When to use:** When you want to simulate an official responding.
- **What you send:** A JSON message like {"message": "approve"} or {"message": "deny"}.
- **What you get back:** A RobotResponse showing the robot's updated status based on the official's reply.

2.5. GET /status (Get Robot's Current Info)

- **What it does:** Gets the robot's current message and state without sending new camera frames or user input.
- **When to use:** For debugging, or if your app needs to periodically check the robot's status without actively interacting.
- **What you send:** Nothing.
- **What you get back:** A RobotResponse showing the robot's current message, action, and state.

3. How the Visitor Communicates with the Robot

The visitor interacts with the robot primarily through **your mobile app**, which acts as the robot's interface.

3.1. Robot "Talks" to the Visitor (Output from Robot)

- **Spoken Words:** The backend converts its messages into spoken audio. This audio is played through the speakers connected to the backend computer. So, the visitor **hears** the robot speak.
- **Displayed Messages & Interactive Elements:** The backend sends messages and instructions to your app. Your app then **displays** these for the visitor to see. This includes:
    - Simple text messages (e.g., "Welcome to XYZ Company!").
    - Text input prompts (e.g., "Please tell me your name.").
    - Choices presented as buttons (e.g., "Are you John? Yes / No").

    - Captured photos (e.g., for confirming their enrollment).

3.2. Visitor "Talks" to the Robot (Input to Robot)

The visitor provides input through your app in a few ways:

- **Speaking:** The visitor speaks their responses (e.g., their name, whom they want to meet, a message).
    - **Your App's Role:** Your app is crucial here. It uses the device's microphone and **Speech-to-Text (STT)** technology (like built-in phone features or a cloud STT service) to convert the visitor's spoken words into text. This text is then sent to the backend.
- **Typing:** If the robot asks for text input, the visitor can also type their response directly into a text box shown on your app's screen.
- **Tapping Buttons:** If the robot presents options (like "Yes" or "No" buttons), the visitor simply taps their choice on your app's screen.

In all these cases, your app acts as the **bridge**, taking the visitor's actions and turning them into digital messages that the backend robot brain can understand and process.

4. Understanding the RobotResponse (What You Get Back)

Almost every time you send something to the robot, you'll get a RobotResponse back. This is the most important message from the backend to your app.

Here's what it contains:

```
{
  "message": "string",       // The text the robot wants to say or display. Your app should show this prominently.
  "action": "string",        // TELLS YOUR APP WHAT TO DO. This is key!
    // Possible actions:
    //   "display_message": Just show the "message" text.
    //   "request_text_input": Show a typing box for the user.
    //   "request_options_input": Show buttons for the user to choose from.
    //   "no_action": No specific UI change needed, just an internal update.
  "state": "string",         // The robot's current internal "thinking" state (e.g., "IDLE", "CONFIRMING_IDENTITY"). Useful for debugging.
  "options": ["string"],     // (Optional) Only present if "action" is "request_options_input". List of choices for buttons.
  "prompt": "string",        // (Optional) Only present if "action" is "request_text_input". A hint for the typing box.
  "image_base64": "string"   // (Optional) A base64-encoded image. Decode this and
```

display it (e.g., the photo taken during enrollment).
}

Example: If the robot wants the user to type their name:
{"message": "Please tell me your full name now.", "action": "request_text_input", "state": "ENROLL_NAME", "prompt": "Your Name"}
Example: If the robot wants the user to choose "Yes" or "No":
{"message": "Are you John Doe? Right?", "action": "request_options_input", "state": "CONFIRMING_IDENTITY", "options": ["yes", "no"]}

5. Your App's To-Do List (Frontend Tasks)

Here's what your team needs to build in the mobile app:

1. **Camera Integration:**
   - Get permission to use the device camera.
   - Continuously capture live video frames.
   - Convert each frame into a JPEG or PNG image (as bytes).
   - Send these image bytes to the POST /process_frame endpoint.
2. **Speech-to-Text (STT) Integration:**
   - Get permission to use the device microphone.
   - Integrate a mobile-native Speech-to-Text (STT) feature (e.g., using built-in OS features or a cloud STT API) to convert the user's spoken words into text.
   - When the robot's action is request_text_input, activate the microphone and listen for speech.
   - Convert the user's speech into a text string.
3. **Dynamic UI Management:**
   - **Main Message Area:** Create a large text area to display the message from the RobotResponse.
   - **Input Area:**
     - Create a text input box (like a keyboard input).
     - Create a section for dynamic buttons.
   - **Conditional Display:** Write code to show/hide these elements based on the action received from the RobotResponse:
     - If action is display_message: Only show the main message. Hide input box and buttons.
     - If action is request_text_input: Show the text input box (with the prompt as a hint). Hide buttons.
     - If action is request_options_input: Show the buttons (using the options list). Hide the text input box.
   - **Image Display:** Create an image display area. If image_base64 is in the

RobotResponse, decode it and show the image. Hide it otherwise.

4. **User Input Submission:**
    - When the user types something and presses "Send" (or hits Enter), send that text to the POST /submit_input endpoint.
    - When the user taps one of the dynamic buttons, send the text of that button (e.g., "yes", "no", "wait") to the POST /submit_input endpoint.
5. **Error Handling & Feedback:**
    - Show friendly messages if there are network problems or if the robot backend sends an error.
    - Display "Loading..." or "Processing..." indicators when waiting for a backend response, especially after sending a frame or input.
6. **App Lifecycle Management:**
    - Properly release camera and microphone resources when the app is paused, backgrounded, or closed.

6. Important Notes

- **Stay Responsive:** Ensure your app doesn't freeze while sending data or waiting for responses. Use asynchronous operations for network calls.
- **Test Continuously:** Use the Swagger UI (/docs) to understand and test each backend endpoint as you build your app.
- **Communication is Key:** If anything is unclear, or you need more details, please ask the backend team!

Good luck with the implementation!