High-Level Design: AI Agent for Alzheimer's Patients

This document outlines the high-level design of the AI agent, focusing on its modular architecture, inter-module communication, and the role of the memory module in storing user data.

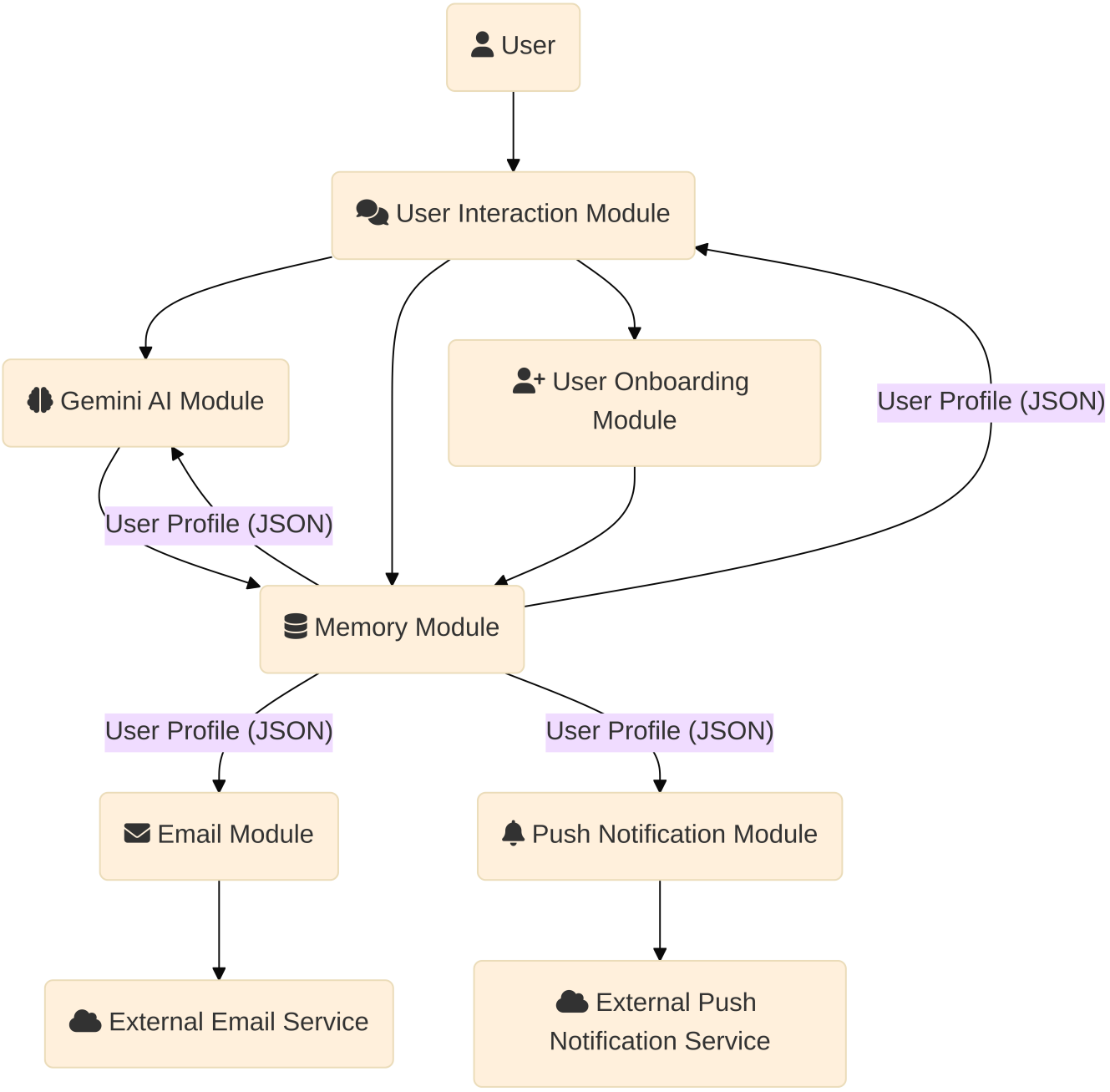
1. Core Modules

The AI agent is composed of several distinct modules, each responsible for a specific set of functionalities. This modular approach enhances maintainability, scalability, and the ability to integrate new features like push notifications.

- **User Interaction Module:** Handles all direct communication with the user, including input reception and output display.
- **Gemini AI Module:** Processes natural language input, generates empathetic responses, and infers user intent using the Google Gemini API.
- **Memory Module:** Manages the persistence and retrieval of user-specific data, acting as the agent's long-term memory.
- **Email Module:** Facilitates sending email notifications to predefined contacts.
- **Push Notification Module (New):** Responsible for sending timely push notifications to the user for reminders.
- **User Onboarding Module:** Guides new users through the initial setup process, collecting essential personal and routine information.

2. High-Level Architecture and Data Flow

The following diagram illustrates the primary modules and their interactions:



Data Flow Overview:

1. **User Input:** The `User Interaction Module` receives input from the user.
2. **Processing:** User input is sent to the `Gemini AI Module` for natural language understanding and response generation. The `Gemini AI Module` also queries the `Memory Module` for contextual user data.
3. **Memory Access:** The `Memory Module` is central to the system, providing user profile data to other modules as needed (e.g., routines for reminders, contact emails).

4. **Onboarding:** For new users, the `User Onboarding Module` collects initial data and stores it via the `Memory Module`.
5. **External Communication:** Based on user requests or scheduled events, the `Email Module` and `Push Notification Module` interact with external services to send communications. These modules retrieve necessary recipient/timing information from the `Memory Module`.

3. Module Functionality and Memory Storage

3.1. User Interaction Module

- **Responsibility:** Acts as the primary interface between the user and the AI agent. It handles command-line input/output but can be extended to support other interfaces (e.g., web, mobile app).
- **Interaction with Memory:** Retrieves the user's name and other basic profile information from the `Memory Module` for personalized greetings and conversational context.

3.2. Gemini AI Module

- **Responsibility:** The brain of the agent, leveraging Google's Gemini API for advanced natural language processing, understanding, and generation. It interprets user queries, infers intent, and crafts empathetic responses.
- **Interaction with Memory:** Before generating a response, this module fetches relevant user data (e.g., name, routines, medications, contacts) from the `Memory Module`. This data is then injected into the prompt sent to the Gemini API, providing the AI with crucial context for personalized and accurate interactions.

3.3. Memory Module

- **Responsibility:** The core component for persistent data storage. It manages the loading and saving of the user's profile, ensuring that all critical information is retained across sessions.

- **Memory Storage (JSON Data):** The `Memory Module` stores user data in a structured JSON format within a file (e.g., `user_profile.json`). This file acts as the agent's long-term memory. The JSON structure allows for flexible and hierarchical storage of various data points:
    - **`save_user_profile(profile)`:** This function takes a Python dictionary representing the user's profile and serializes it into a JSON string, which is then written to the `user_profile.json` file. This ensures data persistence.
    - **`load_user_profile()`:** This function reads the `user_profile.json` file, deserializes the JSON string back into a Python dictionary, and returns it. This makes the user's data available to other modules.
    - **`get_contact_email(contact_name)`:** This function specifically queries the loaded user profile dictionary to retrieve the email address associated with a given contact name.

3.4. Email Module

- **Responsibility:** Handles the sending of email communications. This is primarily used for sending messages to emergency contacts or other predefined individuals.
- **Interaction with Memory:** Retrieves recipient email addresses from the `Memory Module` (via `get_contact_email`) before composing and sending an email.

3.5. Push Notification Module (New)

- **Responsibility:** This new module will be responsible for proactively sending push notifications to the user. This is crucial for timely reminders for tasks like meals and medication.
- **Interaction with Memory:** This module will regularly access the `Memory Module` to retrieve the user's `routines` (breakfast, lunch, dinner times) and `medications` (medication names and times). It will then schedule notifications to be sent approximately 30 minutes prior to these scheduled times. This module will likely require a background process or scheduler to trigger notifications at the appropriate times.

3.6. User Onboarding Module

- **Responsibility:** Guides new users through an interactive process to collect their initial profile information, including name, preferences, contacts, and daily routines.
- **Interaction with Memory:** Once collected, the `User Onboarding Module` passes the structured user data to the `Memory Module` for initial saving into `user_profile.json`.