



i am working for a hackathon-->

That's awesome! What's the theme of the hackathon? Are you working solo or with a team? Let me know if you need any help with brainstorming ideas, tech stack, or implementation! 🚀



Problem Statement 4: Empowering Elderly Care with multi agent AI system

Challenge Overview:

As the global population ages, ensuring the well-being of elderly individuals living independently presents a major challenge.

The goal of this hackathon is to develop a multi-agentic AI system that assists elderly individuals by providing real-time monitoring, reminders, and safety alerts, while promoting health management and social engagement. The agents will work together to create a collaborative support system, involving caregivers, healthcare providers, and family members to ensure optimal care and peace of mind.

The system should monitor health, detect unusual behavior, and provide alerts in case of emergencies. It should also provide reminders to manage daily routines, such as medication schedules, appointments, and daily activities.

Current Process:

Healthcare Provider: Monitors health data (e.g., heart rate, blood pressure, glucose levels) through wearable devices and gets alerted if abnormal values are detected.

Safety Monitoring System: Uses sensors or wearables to track movement, activity, and falls. If an elderly person falls or exhibits unusual behavior (e.g., remaining stationary for too long), the system triggers an alert.

Daily activity Reminder System: Sends reminders in form of voice notes to the elderly individual for medication intake, scheduled appointments or daily activities.

Expected Technical Output: Multiagent framework

That's an interesting and impactful problem statement! Since you need a **multi-agent AI system**, you'll want to break it down into multiple intelligent agents that work together efficiently. Here's a structured approach:

1. System Architecture

Your system will consist of different AI agents working in sync:

- **Health Monitoring Agent** 🏠: Collects and analyzes health data (heart rate, BP, glucose levels) from wearables and alerts caregivers if anomalies are detected.

- **Safety Monitoring Agent** 🚨: Uses motion sensors, wearables, or cameras to detect falls or inactivity and sends alerts.
 - **Reminder Agent** 🔔: Sends voice or text-based reminders for medication, appointments, and routine tasks.
 - **Communication Agent** 📞: Connects elderly individuals with caregivers/family members for emergency or regular check-ins.
 - **Data Analytics & Predictive Agent** 📊: Analyzes trends in health and behavior to predict potential risks (e.g., detecting early signs of cognitive decline).
-

2. Multi-Agent System Framework

Use **Multi-Agent Systems (MAS)** like:

- **JADE (Java Agent Development Framework)** for agent communication.
- **Python-based MAS frameworks** (e.g., SPADE or Mesa).
- **ROS (Robot Operating System)** if IoT integration is needed.

Each agent should have: ☒ **Autonomy**: Can operate independently.

☒ **Reactivity**: Responds to changes in real-time (e.g., fall detection).

☒ **Proactivity**: Predicts risks and prevents issues.

☒ **Communication**: Shares data with other agents for coordinated action.

3. Technologies You Can Use

- **Sensors & Wearables**: Fitbit, Apple Watch, Oura Ring for health tracking.
 - **AI Models**:
 - ML/DL for **anomaly detection** (e.g., detecting abnormal heart rate).
 - NLP for **voice-based reminders & chatbot interactions**.
 - Computer Vision for **fall detection** (OpenCV, YOLO).
 - **Cloud & Edge Computing**:
 - AWS IoT / Google Firebase for real-time data.
 - Edge devices (Raspberry Pi, Jetson Nano) for local processing.
 - **Communication & Alerts**:
 - Twilio API (SMS, calls).
 - WhatsApp bots for family updates.
 - WebRTC for video calls.
-

4. Implementation Strategy

1. **Develop individual agents** separately.
2. **Integrate agent communication** using an MAS framework.

3. **Deploy** in a simulated environment (Unity, Unreal Engine) or real-world testing.

Would you like help with a specific part (e.g., ML models, communication protocols, or agent architecture)? 🚀



i just have to make a mutli agent AI system

Got it! You need to build a **multi-agent AI system (MAS)**. Here's a streamlined approach to building one:

1. Choose a Multi-Agent Framework

Since you're making a **multi-agent AI system**, you need a framework to manage the agents. Some good choices:

- **JADE (Java Agent Development Framework)** – Best for complex MAS, supports agent communication.
- **SPADE (Python)** – Python-based, easy to integrate with AI/ML.
- **Mesa (Python)** – Good for simulating agent interactions.
- **ROS (Robot Operating System)** – If IoT or robotics is involved.

Recommendation: If you're using **Python**, go with **SPADE** since it allows easy ML/AI integration.

2. Define Your Agents

You'll need multiple AI agents working together. Example:

- ✓ **Health Monitoring Agent** → Reads wearable data, detects anomalies.
- ✓ **Safety Agent** → Detects falls, inactivity, and emergencies.
- ✓ **Reminder Agent** → Sends voice/text reminders for meds, appointments.
- ✓ **Communication Agent** → Alerts caregivers & family via SMS/WhatsApp.

Each agent should:

- ♦ Have autonomy (work independently).
- ♦ Communicate with other agents.
- ♦ Be reactive and proactive.

3. Implement Agent Communication

Agents must **talk to each other** via **message passing**. If using SPADE:

```
python

`from spade.agent import Agent
from spade.behaviour import CyclicBehaviour
from spade.message import Message
```

```
class HealthAgent(Agent):
    class MonitorHealth(CyclicBehaviour):
        async def run(self):
            health_data = {"heart_rate": 120} # Simulated data
            if health_data["heart_rate"] > 100:
                msg = Message(to="safety_agent@localhost") # Send alert
                msg.set_metadata("performative", "inform")
                msg.body = "High heart rate detected!"
                await self.send(msg)

    async def setup(self):
        self.add_behaviour(self.MonitorHealth())

health_agent = HealthAgent("health_agent@localhost", "password")
health_agent.start()
```

Other agents (e.g., **SafetyAgent**) would receive this message and take action.

4. Deploy & Test

- Run agents on a **local network** or **cloud**.
- Simulate sensor inputs.
- Use **Firebase** or **Twilio API** for real-time alerts.

This should get your **multi-agent AI system** up and running! 🚀 Need help with a specific part (like deployment or AI integration)?



 Uploaded image

 Uploaded image

 Uploaded image

i have three dataset-->

daily_remainder-

Device-ID/User-ID	Timestamp	Reminder Type	Scheduled Time	Reminder Sent (Yes/No)	Acknowledged (Yes/No)
D1000	1/2/2025 11:25	Exercise	13:00:00	No	No
D1001	1/3/2025 2:52	Hydration	13:00:00	Yes	Yes
D1002	1/8/2025 13:50	Appointment	13:30:00	No	No
D1003	1/5/2025 5:16	Exercise	8:00:00	No	No
D1004	1/1/2025 4:20	Medication	11:30:00	No	No
D1005	1/20/2025 10:39	Hydration	14:30:00	No	No
D1006	1/25/2025 10:05	Medication	15:30:00	No	No
D1007	1/2/2025 21:13	Appointment	22:00:00	Yes	No
D1008	1/21/2025 15:25	Medication	12:30:00	No	No
D1009	1/15/2025 2:47	Exercise	20:30:00	Yes	Yes
D1010	1/24/2025 13:58	Appointment	20:00:00	Yes	Yes
D1011	1/10/2025 6:39	Exercise	12:30:00	Yes	Yes
D1012	1/25/2025 1:18	Appointment	10:30:00	No	No
D1013	1/6/2025 12:23	Appointment	20:30:00	No	No
D1014	1/26/2025 14:14	Hydration	13:00:00	No	No
D1015	1/7/2025 9:45	Hydration	13:00:00	No	No

D1016	1/8/2025 19:04	Appointment	7:30:00	Yes	Yes
D1017	1/21/2025 18:14	Appointment	19:30:00	Yes	Yes
D1018	1/18/2025 22:50	Medication	16:00:00	No	No
D1019	1/30/2025 20:49	Medication	13:00:00	Yes	Yes
D1020	1/29/2025 15:14	Exercise	7:00:00	Yes	No
D1021	1/12/2025 11:38	Medication	20:00:00	Yes	No

health_monitoring-

Device-ID/User-ID	Timestamp	Heart Rate	Heart Rate Below/Above Threshold (Yes/No)	Blood Pressure	Blood Pressure Below/Above Threshold (Yes/No)	Glucose Levels	Glucose Levels Below/Above Threshold (Yes/No)	Oxygen Saturation (SpO ₂ %)	SpO ₂ Below Threshold (Yes/No)	Alert Triggered (Yes/No)	Caregiver Notified (Yes/No)
-------------------	-----------	------------	---	----------------	---	----------------	---	--	---	--------------------------	-----------------------------

D1000	1/22/2025 20:42	116	Yes	136/79 mmHg	Yes	141	Yes	98	No	Yes	Yes
D1001	1/16/2025 12:22	119	Yes	105/77 mmHg	No	146	Yes	93	No	Yes	Yes
D1002	1/10/2025 9:26	97	No	120/87 mmHg	Yes	133	No	97	No	Yes	Yes
D1003	1/10/2025 9:53	113	Yes	138/65 mmHg	Yes	82	No	98	No	Yes	Yes
D1004	1/3/2025 15:50	88	No	108/69 mmHg	No	146	Yes	97	No	Yes	Yes
D1005	1/5/2025 8:29	119	Yes	114/65 mmHg	No	133	No	91	Yes	Yes	Yes
D1006	1/30/2025 3:01	109	Yes	137/61 mmHg	Yes	143	Yes	98	No	Yes	Yes
D1007	1/10/2025 16:51	114	Yes	119/61 mmHg	No	107	No	99	No	Yes	Yes
D1008	1/20/2025 12:22	101	Yes	117/61 mmHg	No	134	No	97	No	Yes	Yes
D1009	1/23/2025 16:54	88	No	101/76 mmHg	No	133	No	94	No	No	No
D1010	1/29/2025 17:52	61	No	110/61 mmHg	No	72	Yes	98	No	Yes	Yes
D1011	1/23/2025 15:03	89	No	121/70 mmHg	No	129	No	91	Yes	Yes	Yes
D1012	1/11/2025 10:32	105	Yes	111/74 mmHg	No	145	Yes	93	No	Yes	Yes
D1013	1/14/2025 19:53	69	No	127/80 mmHg	No	128	No	95	No	No	No
D1014	1/13/2025 10:53	63	No	114/82 mmHg	No	140	No	91	Yes	Yes	Yes
D1015	1/14/2025 5:57	63	No	136/68 mmHg	Yes	74	Yes	99	No	Yes	Yes
D1016	1/24/2025 20:01	74	No	103/67 mmHg	No	147	Yes	90	Yes	Yes	Yes
D1017	1/17/2025 7:26	64	No	107/89 mmHg	Yes	102	No	91	Yes	Yes	Yes
D1018	1/6/2025 3:14	115	Yes	125/77 mmHg	No	103	No	99	No	Yes	Yes
D1019	1/5/2025 14:00	118	Yes	134/62 mmHg	Yes	104	No	95	No	Yes	Yes
D1020	1/20/2025 15:30	69	No	136/69 mmHg	Yes	147	Yes	95	No	Yes	Yes
D1021	1/8/2025 9:47	117	Yes	109/90 mmHg	Yes	135	No	96	No	Yes	Yes
D1022	1/9/2025 3:16	117	Yes	121/67 mmHg	No	148	Yes	95	No	Yes	Yes
D1023	1/8/2025 9:00	102	Yes	122/87 mmHg	Yes	95	No	90	Yes	Yes	Yes

safety_monitoring--

Device-ID/User-ID	Timestamp	Movement Activity	Fall Detected (Yes/No)	Impact Force Level	Post-Fall Inactivity Duration (Seconds)	Location	Alert Triggered (Yes/No)	Caregiver Notified (Yes/No)
-------------------	-----------	-------------------	------------------------	--------------------	---	----------	--------------------------	-----------------------------

D1000	1/7/2025 16:04	No Movement	No	0	Kitchen	No	No
D1001	1/20/2025 15:45	Lying	No	0	Living Room	No	No
D1002	1/2/2025 2:42	No Movement	No	0	Bedroom	No	No
D1003	1/1/2025 22:36	Lying	No	0	Kitchen	No	No
D1004	1/3/2025 16:30	No Movement	No	0	Bedroom	No	No
D1005	1/19/2025 12:13	Sitting	No	0	Bedroom	No	No
D1006	1/4/2025 10:58	Lying	No	0	Bathroom	No	No
D1007	1/21/2025 7:31	Lying	No	0	Bedroom	No	No

D1008	1/10/2025 23:56	Sitting	No	-	0	Kitchen	No	No
D1009	1/4/2025 16:16	Sitting	No	-	0	Kitchen	No	No
D1010	1/13/2025 7:05	Walking	No	-	0	Bedroom	No	No
D1011	1/9/2025 6:32	Sitting	No	-	0	Bathroom	No	No
D1012	1/1/2025 22:34	No Movement	No	-	0	Kitchen	No	No
D1013	1/17/2025 10:52	Walking	No	-	0	Living Room	No	No
D1014	1/29/2025 1:59	Walking	No	-	0	Bedroom	No	No
D1015	1/23/2025 2:14	Sitting	No	-	0	Bedroom	No	No
D1016	1/13/2025 10:48	Walking	No	-	0	Living Room	No	No
D1017	1/4/2025 12:16	Sitting	No	-	0	Bathroom	No	No
D1018	1/6/2025 8:18	No Movement	No	-	0	Living Room	No	No
D1019	1/11/2025 21:01	Sitting	No	-	0	Living Room	No	No
D1020	1/1/2025 16:18	Walking	No	-	0	Bathroom	No	No
D1021	1/20/2025 17:34	Walking	No	-	0	Living Room	No	No
D1022	1/19/2025 19:46	No Movement	Yes	Medium	463	Bathroom	Yes	Yes
D1023	1/1/2025 20:45	Lying	No	-	0	Kitchen	No	No

It looks like you've uploaded multiple files named `image.png`. How can I assist you with them? Do you want me to process or analyze them in any specific way?



i have three dataset-->

daily_remainder-

Device-ID/User-ID	Timestamp	Reminder Type	Scheduled Time	Reminder Sent (Yes/No)	Acknowledged (Yes/No)
D1000	1/2/2025 11:25	Exercise	13:00:00	No	No
D1001	1/3/2025 2:52	Hydration	13:00:00	Yes	Yes
D1002	1/8/2025 13:50	Appointment	13:30:00	No	No
D1003	1/5/2025 5:16	Exercise	8:00:00	No	No
D1004	1/1/2025 4:20	Medication	11:30:00	No	No
D1005	1/20/2025 10:39	Hydration	14:30:00	No	No
D1006	1/25/2025 10:05	Medication	15:30:00	No	No
D1007	1/2/2025 21:13	Appointment	22:00:00	Yes	No
D1008	1/21/2025 15:25	Medication	12:30:00	No	No
D1009	1/15/2025 2:47	Exercise	20:30:00	Yes	Yes
D1010	1/24/2025 13:58	Appointment	20:00:00	Yes	Yes
D1011	1/10/2025 6:39	Exercise	12:30:00	Yes	Yes
D1012	1/25/2025 1:18	Appointment	10:30:00	No	No
D1013	1/6/2025 12:23	Appointment	20:30:00	No	No
D1014	1/26/2025 14:14	Hydration	13:00:00	No	No
D1015	1/7/2025 9:45	Hydration	13:00:00	No	No
D1016	1/8/2025 19:04	Appointment	7:30:00	Yes	Yes
D1017	1/21/2025 18:14	Appointment	19:30:00	Yes	Yes
D1018	1/18/2025 22:50	Medication	16:00:00	No	No
D1019	1/30/2025 20:49	Medication	13:00:00	Yes	Yes
D1020	1/29/2025 15:14	Exercise	7:00:00	Yes	No
D1021	1/12/2025 11:38	Medication	20:00:00	Yes	No

health_monitoring-

Device-ID/User-ID	Timestamp	Heart Rate	Heart Rate Below/Above Threshold (Yes/No)	Blood Pressure	Blood Pressure Below/Above Threshold (Yes/No)	Glucose Levels	Glucose Levels Below/Above Threshold (Yes/No)	Oxygen Saturation (SpO ₂ %)	SpO ₂ Below Threshold (Yes/No)	Alert Triggered (Yes/No)	Caregiver Notified (Yes/No)
D1000	1/22/2025 20:42	116	Yes	136/79 mmHg	Yes	141	Yes	98	No	Yes	Yes
D1001	1/16/2025 12:22	119	Yes	105/77 mmHg	No	146	Yes	93	No	Yes	Yes
D1002	1/10/2025 9:26	97	No	120/87 mmHg	Yes	133	No	97	No	Yes	Yes
D1003	1/10/2025 9:53	113	Yes	138/65 mmHg	Yes	82	No	98	No	Yes	Yes
D1004	1/3/2025 15:50	88	No	108/69 mmHg	No	146	Yes	97	No	Yes	Yes
D1005	1/5/2025 8:29	119	Yes	114/65 mmHg	No	133	No	91	Yes	Yes	Yes
D1006	1/30/2025 3:01	109	Yes	137/61 mmHg	Yes	143	Yes	98	No	Yes	Yes
D1007	1/10/2025 16:51	114	Yes	119/61 mmHg	No	107	No	99	No	Yes	Yes
D1008	1/20/2025 12:22	101	Yes	117/61 mmHg	No	134	No	97	No	Yes	Yes
D1009	1/23/2025 16:54	88	No	101/76 mmHg	No	133	No	94	No	No	No
D1010	1/29/2025 17:52	61	No	110/61 mmHg	No	72	Yes	98	No	Yes	Yes
D1011	1/23/2025 15:03	89	No	121/70 mmHg	No	129	No	91	Yes	Yes	Yes
D1012	1/11/2025 10:32	105	Yes	111/74 mmHg	No	145	Yes	93	No	Yes	Yes
D1013	1/14/2025 19:53	69	No	127/80 mmHg	No	128	No	95	No	No	No
D1014	1/13/2025 10:53	63	No	114/82 mmHg	No	140	No	91	Yes	Yes	Yes
D1015	1/14/2025 5:57	63	No	136/68 mmHg	Yes	74	Yes	99	No	Yes	Yes
D1016	1/24/2025 20:01	74	No	103/67 mmHg	No	147	Yes	90	Yes	Yes	Yes
D1017	1/17/2025 7:26	64	No	107/89 mmHg	Yes	102	No	91	Yes	Yes	Yes
D1018	1/6/2025 3:14	115	Yes	125/77 mmHg	No	103	No	99	No	Yes	Yes
D1019	1/5/2025 14:00	118	Yes	134/62 mmHg	Yes	104	No	95	No	Yes	Yes
D1020	1/20/2025 15:30	69	No	136/69 mmHg	Yes	147	Yes	95	No	Yes	Yes
D1021	1/8/2025 9:47	117	Yes	109/90 mmHg	Yes	135	No	96	No	Yes	Yes
D1022	1/9/2025 3:16	117	Yes	121/67 mmHg	No	148	Yes	95	No	Yes	Yes
D1023	1/8/2025 9:00	102	Yes	122/87 mmHg	Yes	95	No	90	Yes	Yes	Yes

safety_monitoring--

Device-ID/User-ID	Timestamp	Movement Activity	Fall Detected (Yes/No)	Impact Force Level	Post-Fall Inactivity Duration (Seconds)	Location	Alert Triggered (Yes/No)	Caregiver Notified (Yes/No)
D1000	1/7/2025 16:04	No Movement	No	0	Kitchen	No	No	
D1001	1/20/2025 15:45	Lying	No	0	Living Room	No	No	
D1002	1/2/2025 2:42	No Movement	No	0	Bedroom	No	No	
D1003	1/1/2025 22:36	Lying	No	0	Kitchen	No	No	
D1004	1/3/2025 16:30	No Movement	No	0	Bedroom	No	No	
D1005	1/19/2025 12:13	Sitting	No	0	Bedroom	No	No	
D1006	1/4/2025 10:58	Lying	No	0	Bathroom	No	No	
D1007	1/21/2025 7:31	Lying	No	0	Bedroom	No	No	
D1008	1/10/2025 23:56	Sitting	No	0	Kitchen	No	No	
D1009	1/4/2025 16:16	Sitting	No	0	Kitchen	No	No	
D1010	1/13/2025 7:05	Walking	No	0	Bedroom	No	No	
D1011	1/9/2025 6:32	Sitting	No	0	Bathroom	No	No	
D1012	1/1/2025 22:34	No Movement	No	0	Kitchen	No	No	
D1013	1/17/2025 10:52	Walking	No	0	Living Room	No	No	
D1014	1/29/2025 1:59	Walking	No	0	Bedroom	No	No	
D1015	1/23/2025 2:14	Sitting	No	0	Bedroom	No	No	

D1016	1/13/2025 10:48	Walking	No	-	0	Living Room	No	No
D1017	1/4/2025 12:16	Sitting	No	-	0	Bathroom	No	No
D1018	1/6/2025 8:18	No Movement	No	-	0	Living Room	No	No
D1019	1/11/2025 21:01	Sitting	No	-	0	Living Room	No	No
D1020	1/1/2025 16:18	Walking	No	-	0	Bathroom	No	No
D1021	1/20/2025 17:34	Walking	No	-	0	Living Room	No	No
D1022	1/19/2025 19:46	No Movement	Yes	Medium	463	Bathroom	Yes	Yes
D1023	1/1/2025 20:45	Lying	No	-	0	Kitchen	No	No

with these i have to make models

What do you need to do with these datasets? Are you looking for insights, data cleaning, merging them, or something else? Let me know your goal so I can help effectively! 😊



agents that you told me use these datasets and give how i will make MAS with this

To create a **Multi-Agent System (MAS)** using your three datasets (**daily reminders, health monitoring, and safety monitoring**), we need to design agents that handle specific tasks. Below is a structured plan outlining the agents, their roles, and how they will interact.

Step 1: Define Agents and Their Responsibilities

Each agent will have a specific role based on your datasets.

1. Reminder Agent

- **Purpose:** Manages reminders for users and ensures they acknowledge them.
- **Inputs:** `daily_reminder` dataset.
- **Tasks:**
 - Sends reminders at the scheduled time.
 - Checks if the reminder was acknowledged.
 - Escalates unacknowledged reminders.

2. Health Monitoring Agent

- **Purpose:** Tracks health metrics and detects anomalies.
- **Inputs:** `health_monitoring` dataset.
- **Tasks:**
 - Monitors heart rate, blood pressure, glucose, and SpO₂.
 - Flags abnormal values and triggers alerts.
 - Notifies caregivers if needed.

3. Safety Monitoring Agent

- **Purpose:** Detects unusual inactivity or fall events.
- **Inputs:** `safety_monitoring` dataset.
- **Tasks:**
 - Identifies falls or prolonged inactivity.
 - Determines location-based risks.
 - Notifies caregivers if required.

4. Caregiver Agent

- **Purpose:** Receives alerts from other agents and takes action.
- **Inputs:** Alerts from Health, Safety, and Reminder Agents.
- **Tasks:**
 - Sends emergency notifications.
 - Escalates unacknowledged reminders.
 - Can request additional health or safety checks.

5. User Interaction Agent

- **Purpose:** Interfaces with the user for feedback and acknowledgment.
- **Inputs:** Responses from users.
- **Tasks:**
 - Collects responses to reminders.
 - Acknowledges safety and health alerts.
 - Updates other agents based on user feedback.

Step 2: Design Agent Interactions

The agents communicate to maintain an efficient monitoring system.

- **Reminder Agent → User Interaction Agent:** Checks if reminders are acknowledged.
- **Health Monitoring Agent → Caregiver Agent:** Sends alerts if abnormal health values are detected.
- **Safety Monitoring Agent → Caregiver Agent:** Notifies if a fall or inactivity is detected.
- **Caregiver Agent → User Interaction Agent:** Confirms if emergency actions are needed.

Step 3: Implementing the MAS

1. Technology Stack

- **Programming Language:** Python
- **Frameworks:**
 - **JADE (Java)** – for agent-based modeling (optional).
 - **Spade (Python)** – for agent communication.
 - **Flask/FastAPI** – for API-based agent communication.
 - **RabbitMQ/Kafka** – for message passing.
 - **SQLite/MySQL** – for storing user data.

2. Implementation Steps

Step 1: Create Agents in Python

Each agent can be an independent script running as a microservice.

Example: **Reminder Agent**

```
python

`import datetime

class ReminderAgent:
    def __init__(self, user_id, reminder_type, scheduled_time, sent, acknowledged):
        self.user_id = user_id
        self.reminder_type = reminder_type
        self.scheduled_time = scheduled_time
        self.sent = sent
        self.acknowledged = acknowledged

    def check_reminder(self, current_time):
        if not self.sent and current_time >= self.scheduled_time:
            print(f"Sending {self.reminder_type} reminder to {self.user_id}")
            self.sent = True

    def acknowledge(self):
        self.acknowledged = True
        print(f"Reminder {self.reminder_type} acknowledged by {self.user_id}")

# Example Usage
reminder = ReminderAgent("D1000", "Exercise", datetime.time(13, 0), False, False)
reminder.check_reminder(datetime.datetime.now().time())`
```

Step 2: Implement Communication Between Agents

Use **RabbitMQ/Kafka** for real-time communication.

- **Health Agent publishes alerts** → Caregiver Agent subscribes.
- **Safety Agent publishes fall detection** → Caregiver Agent subscribes.
- **Reminder Agent updates User Agent** → Tracks acknowledgment.

Step 3: Build a User Dashboard

Use **Flask/Django** to display:

- Pending reminders.

- Health alerts.
- Safety warnings.

Final Thoughts

1. **Use Spade or JADE** if you want real agent-based behavior.
2. **Use Flask/FastAPI** if you prefer API-based agent interactions.
3. **Use Kafka/RabbitMQ** if real-time messaging is required.

Would you like code for a specific agent? 🚀



i will use spade .
give in detail steps on how to implement it for complete my problem statement

Great choice! **Spade** is a Python-based **multi-agent system (MAS) framework** that allows agents to communicate asynchronously using **XMPP (Extensible Messaging and Presence Protocol)**. Below is a **step-by-step guide** to implement your **Multi-Agent System (MAS)** using **Spade** for your problem statement.

📌 Step 1: Setup and Installation

Before we start coding, we need to install the required libraries.

1 Install Spade

```
bash
`pip install spade`
```

2 Install Other Required Packages

```
bash
`pip install aioxmpp pandas`
```

- **Spade**: The main MAS framework.
- **aioxmpp**: Handles XMPP communication.
- **pandas**: Used for dataset handling.

📌 Step 2: Setting Up an XMPP Server

Spade agents communicate over the **XMPP protocol**, so we need an **XMPP server**.

Option 1: Use a Free Public XMPP Server

You can create accounts on a free XMPP server:

- [Gajim](#) - A chat client to test XMPP.
- [XMPP Hosting](#) - Free XMPP accounts.

Option 2: Install a Local XMPP Server

Use **Prosody** (for Linux) or **Openfire** (Windows/macOS).

Step 3: Define Agent Roles

We will implement **five agents**:

1. **ReminderAgent** – Manages daily reminders.
2. **HealthMonitoringAgent** – Tracks health metrics.
3. **SafetyMonitoringAgent** – Detects falls and inactivity.
4. **CaregiverAgent** – Receives alerts and escalates if needed.
5. **UserInteractionAgent** – Interfaces with users.

Step 4: Implement Agents Using Spade

1 Reminder Agent (Sends Reminders to Users)

This agent reads from the `daily_reminders` dataset and sends reminders.

```
python

`import` spade
import pandas as pd
import datetime

class ReminderAgent(spade.agent.Agent):
    class ReminderBehaviour(spade.behaviour.PeriodicBehaviour):
        async def run(self):
            print("[ReminderAgent] Checking reminders...")
            reminders = pd.read_csv("daily_reminders.csv")
            now = datetime.datetime.now().strftime("%H:%M")

            for _, row in reminders.iterrows():
                if row["time"] == now and row["sent"] == "no":
                    print(f"Sending reminder: {row['reminder_type']} to {row['user_id']}")
                    # Mark reminder as sent
                    reminders.loc[reminders["user_id"] == row["user_id"], "sent"] = "yes"
                    reminders.to_csv("daily_reminders.csv", index=False)

            async def setup(self):
                print("[ReminderAgent] Starting...")
                self.add_behaviour(self.ReminderBehaviour(period=60)) # Checks every minute

if __name__ == "__main__":
```

```
jid = "reminder_agent@yourxmppserver.com"
password = "password"
agent = ReminderAgent(jid, password)
agent.start()
```

2 Health Monitoring Agent (Detects Abnormal Health Data)

Reads from the `health_monitoring` dataset and flags anomalies.

```
python

`import spade
import pandas as pd

class HealthMonitoringAgent(spade.agent.Agent):
    class MonitorHealth(spade.behaviour.PeriodicBehaviour):
        async def run(self):
            print("[HealthMonitoringAgent] Monitoring health data...")
            health_data = pd.read_csv("health_monitoring.csv")

            for _, row in health_data.iterrows():
                if row["heart_rate"] > 100 or row["blood_pressure"] > 140:
                    print(f"ALERT: {row['user_id']} has abnormal vitals!")
                    # Here, send a message to CaregiverAgent

            async def setup(self):
                print("[HealthMonitoringAgent] Starting...")
                self.add_behaviour(self.MonitorHealth(period=60)) # Check every 60 seconds

if __name__ == "__main__":
    jid = "health_agent@yourxmppserver.com"
    password = "password"
    agent = HealthMonitoringAgent(jid, password)
    agent.start()
```

3 Safety Monitoring Agent (Detects Falls/Inactivity)

Detects inactivity and sends alerts.

```
python

`import spade
import pandas as pd

class SafetyMonitoringAgent(spade.agent.Agent):
    class MonitorSafety(spade.behaviour.PeriodicBehaviour):
        async def run(self):
            print("[SafetyMonitoringAgent] Checking safety data...")
            safety_data = pd.read_csv("safety_monitoring.csv")

            for _, row in safety_data.iterrows():
                if row["fall_detected"] == "yes":
                    print(f"ALERT: Fall detected for {row['user_id']}!")
                    # Notify CaregiverAgent

            async def setup(self):
                print("[SafetyMonitoringAgent] Starting...")
                self.add_behaviour(self.MonitorSafety(period=60))

if __name__ == "__main__":
    jid = "safety_agent@yourxmppserver.com"
    password = "password"
```

```
agent = SafetyMonitoringAgent(jid, password)
agent.start()
```

4 Caregiver Agent (Receives Alerts and Takes Action)

Listens for alerts from health and safety agents.

```
python

`import spade

class CaregiverAgent(spade.agent.Agent):
    class ReceiveAlerts(spade.behaviour.CyclicBehaviour):
        async def run(self):
            msg = await self.receive(timeout=10)
            if msg:
                print(f"[CaregiverAgent] Received alert: {msg.body}")

        async def setup(self):
            print("[CaregiverAgent] Starting...")
            self.add_behaviour(self.ReceiveAlerts())

if __name__ == "__main__":
    jid = "caregiver_agent@yourxmppserver.com"
    password = "password"
    agent = CaregiverAgent(jid, password)
    agent.start()
```

5 User Interaction Agent (Handles User Responses)

Handles acknowledgments for reminders.

```
python

`import spade

class UserInteractionAgent(spade.agent.Agent):
    class HandleUserResponse(spade.behaviour.CyclicBehaviour):
        async def run(self):
            msg = await self.receive(timeout=10)
            if msg:
                print(f"[UserInteractionAgent] Received user response: {msg.body}")

        async def setup(self):
            print("[UserInteractionAgent] Starting...")
            self.add_behaviour(self.HandleUserResponse())

if __name__ == "__main__":
    jid = "user_agent@yourxmppserver.com"
    password = "password"
    agent = UserInteractionAgent(jid, password)
    agent.start()
```

Step 5: Running the Multi-Agent System

1 Start XMPP Server

If using a local server like Prosody:

```
bash

`prosodyctl start
```

If using a cloud XMPP server, ensure your agent accounts are registered.

2 Start Each Agent

Run each agent script in separate terminals:

```
bash

`python reminder_agent.py
python health_monitoring_agent.py
python safety_monitoring_agent.py
python caregiver_agent.py
python user_interaction_agent.py
```

3 Test Communication

- Add test data in `daily_reminders.csv`, `health_monitoring.csv`, and `safety_monitoring.csv`.
- Observe agent logs as they send reminders, detect anomalies, and trigger alerts.

Conclusion

♦ What We Built:

- ✓ A Spade-based Multi-Agent System (MAS)
- ✓ Agents that **communicate over XMPP**
- ✓ **Periodic monitoring of reminders, health, and safety**
- ✓ **Caregiver agent handles alerts**
- ✓ **User agent acknowledges alerts**

Would you like me to help with adding **GUI or API integration**? 