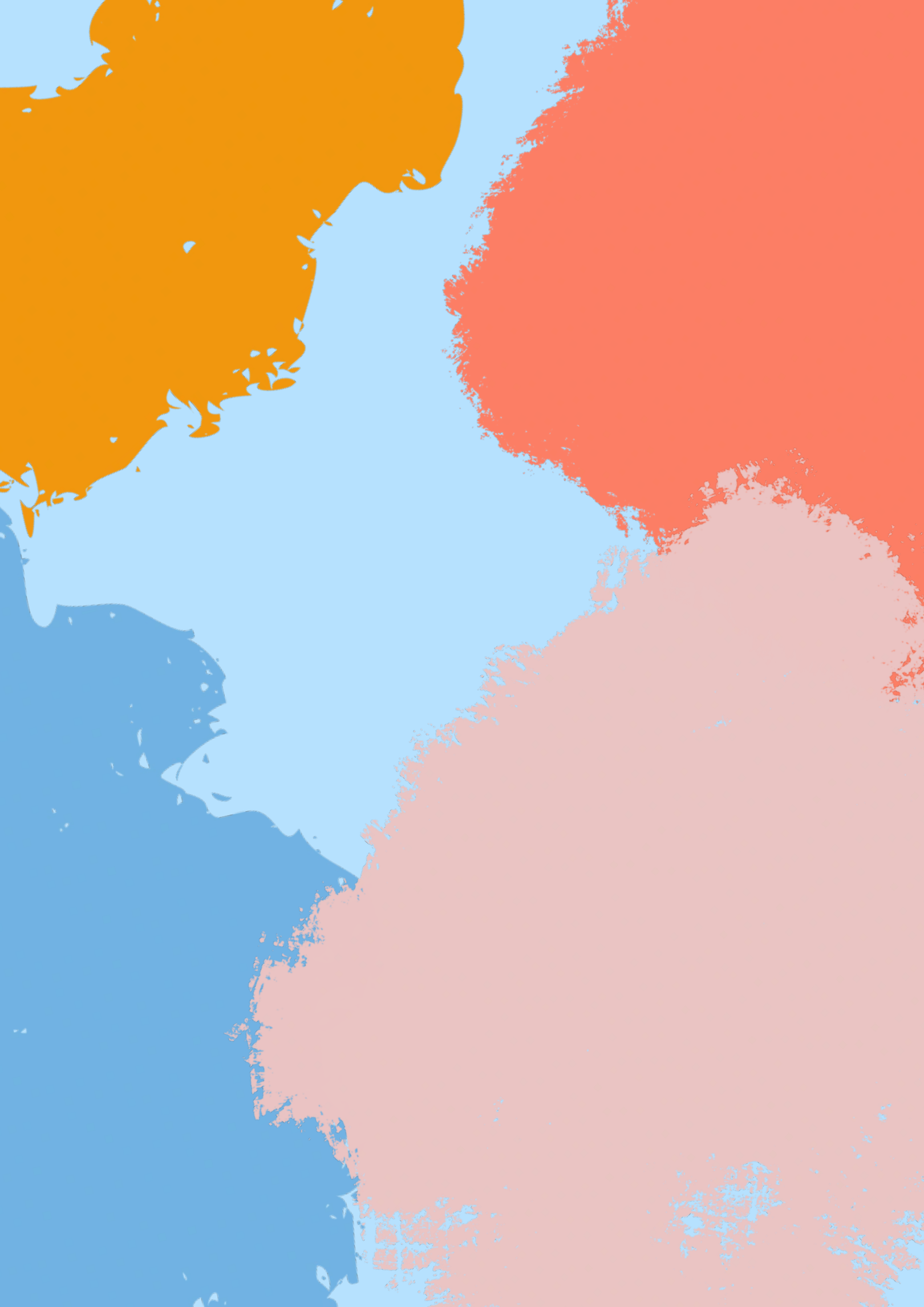


Autogen - Module 4

Human in the loop

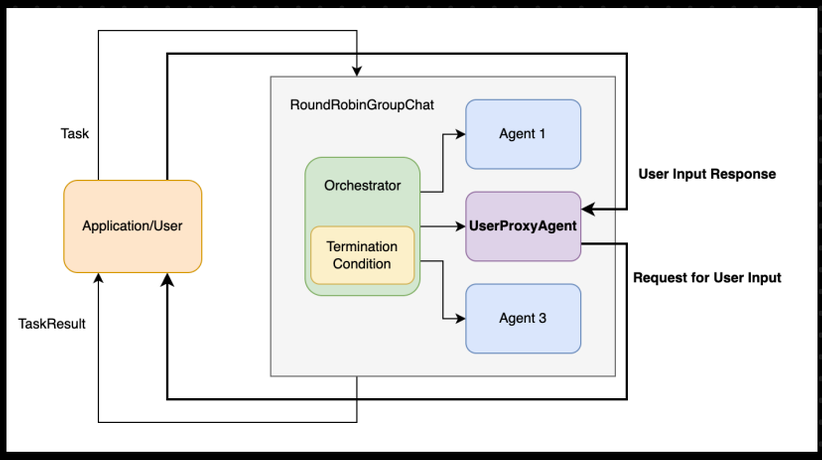
We will focus on how can we interact with our team from our application and provide human feedback to our team.

1. During our team's run: run or run_stream()
   User Proxy Agent

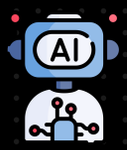
2. Once the run terminate: feedback token for the next run()

Providing feedback during a run

Selector Group Chat



User Proxy Agent

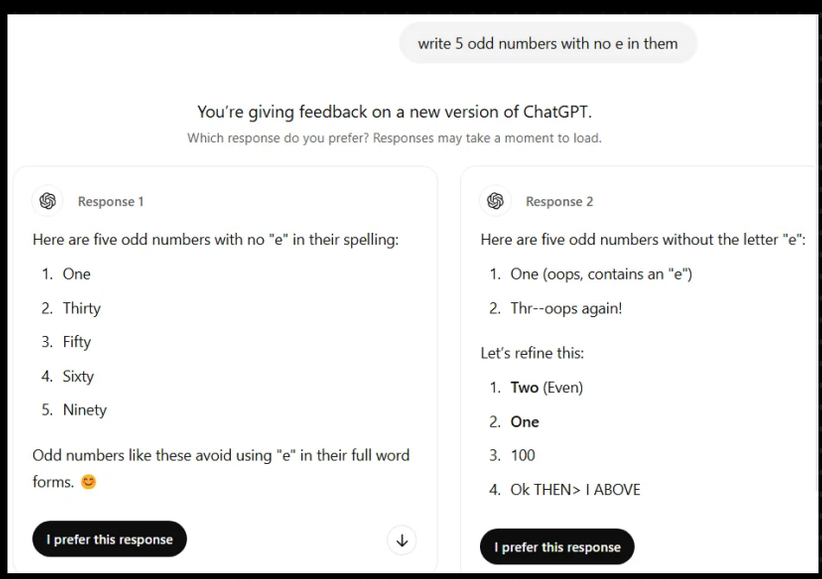


'APPROVE'

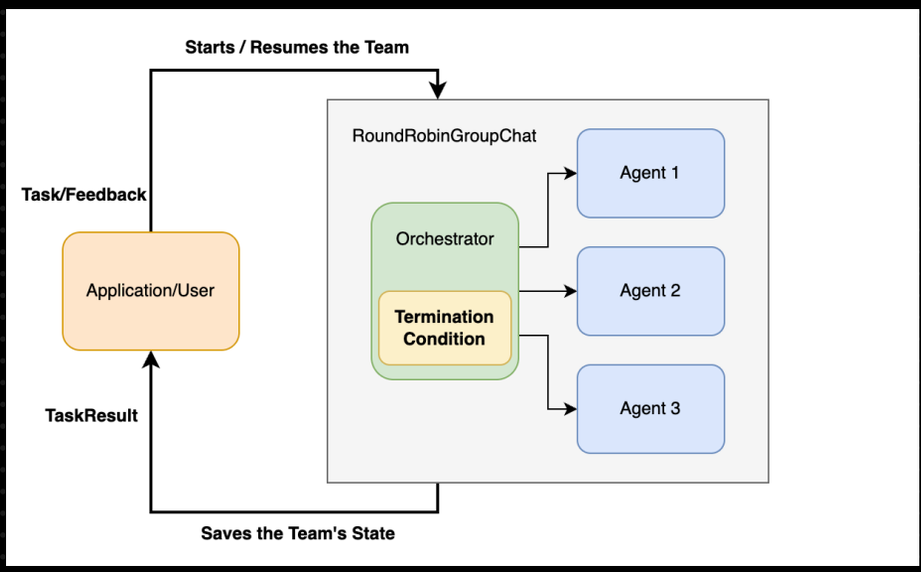
User Proxy agent

The state of the team is unstable & the progress is held up until the user provides a feedback or say errors out.

Because of this blocking approach, it is recommended that we use it only for short interactions.



Providing Feedback to the Next Run



1st run

2nd run

no user proxy agent

There are 2 ways to implement this approach.

3 agents

1. Maximum number of turns ÷ team ⇒ max turns 5 turns
2. Using Termination Condition
   ↳ TERMINATE

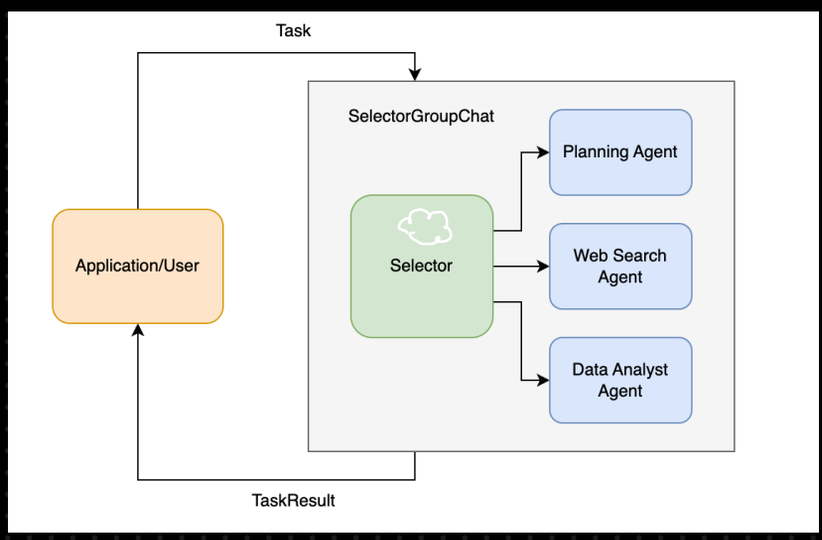
Providing Feedback to the Next Run: Using Max Turns

pause the team for user input / feedback

Providing Feedback with Termination Conditions

Selector Group Chat

Selector GroupChat is a group chat similar to Round robin GroupChat but with a model based next speaker selection Mechanism.



How does a Selector GroupChat Work ?

→ a group chat mechanism

0. team recieves the task via run() or run_stream()
1. team analyzes the current conversion context (Conversation history $ name, description) to select the next speaker

By default, team will not select same speaker (can be changed)

2. team prompts the selected agent to give response.
   → broadcasted to all the other participants.
3. We check on the termination condition. if Yes → end the Conversation
   No → repeat from step 1
4. When conversation ends, team return
   Task Result ← (having conversation history)

When task is finished, the conv. context is kept within team and all participants, so that next task can continue from this context

→ can be reset by calling `.reset()`

By default, AssistantAgent returns tool value as the response.

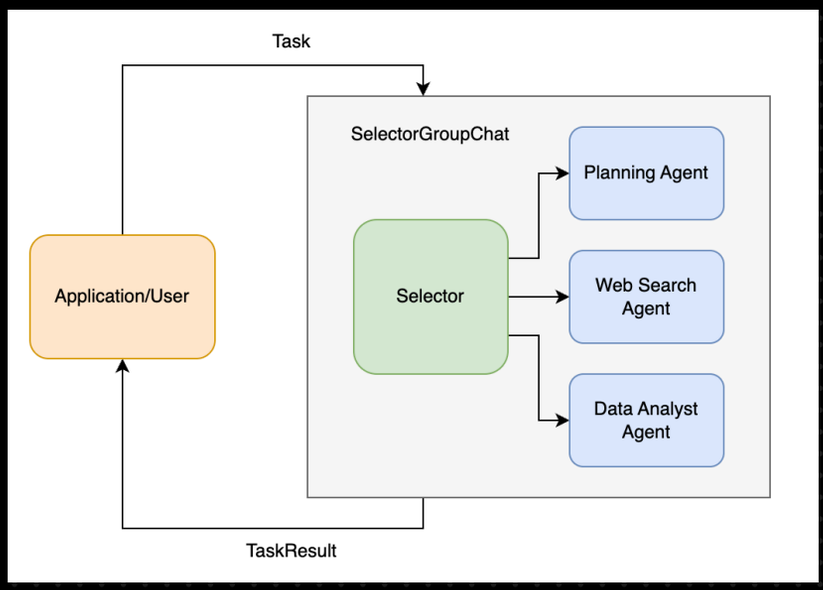
Selector Prompt

uses a model to select the next speaker/agent based on context.

Custom Selector Function

For a better control over the selection process, we can use a Custom Selector function.

We are going to call Planning agent after any specialized agent work



Web Search Agent was last Used

Yes → Planning

No → None

Swarm in Autogen

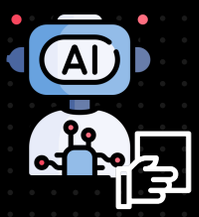
Swarm implements a team where agents can hand off task to other agent based on their capabilities.

→ multi-agent framework designed by openAI

→ enables agent to make local decision about task planning

→ not relying on central orchestrator/authority as in Selector group chat

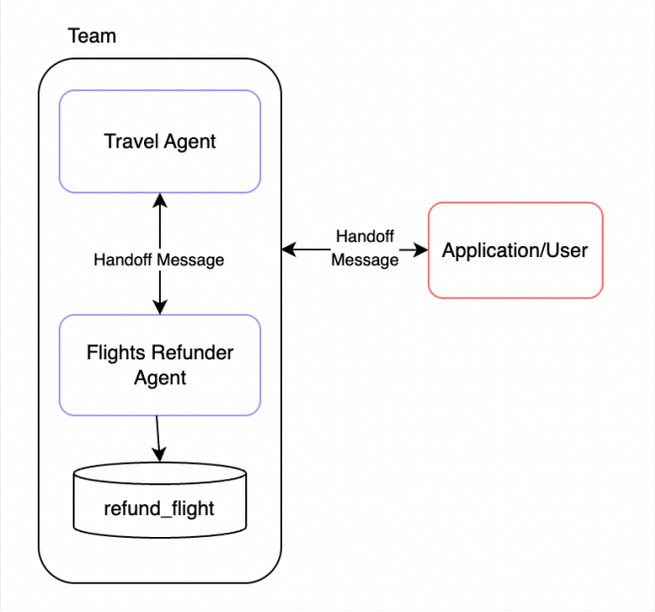
How Does Swarm Work?



A1 A2 A3

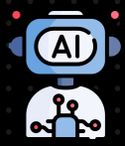
1. Each agent has the ability to generate Handoff Message to signal which other agent it can handoff to.
2. When team starts on a task
   a) first speaker agent operate on the task
   b) make a decision on whether to handoff and to whom
3. When an agent generates a Handoff Message, the recieving agent takes over the task with the same message context.
4. Process continues until a termination condition is met.

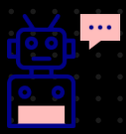
Autogen Swarm - Customer Support Example



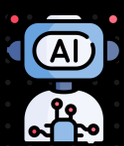
Graphflow in Autogen

1. Round robin
2. Selector Group
3. Swarm





Graphflow provides a way of defining structured execution & precise control on how agents interact to accomplish a task



Writer

Reviewer

DiGraph Builder

utility to build DiGraph which can then be used in Graphflow.

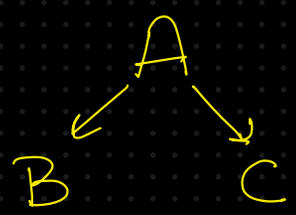
Using digraph we can easily build agent interactions including

1. Sequential Chain

$$A \rightarrow B \rightarrow C$$

3 nodes
2 edges

2. Parallel fans-out

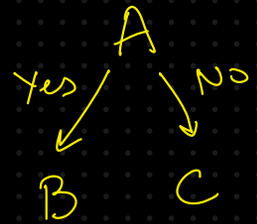


3. Conditional Branching

$$A \rightarrow B \longrightarrow C$$

(Yes) TERMINATE

(No) CONTINUE



4. Cyclic loops with safe exits.

$$A \rightarrow B \rightarrow A \text{ ('loop')},\ B \rightarrow C \text{ ('exit')}$$