Refactoring Plan

Multi Agent Feature

위치

- 구현 위치: `./src/components/orchestrators`

개념

- 여러 Assistant를 그룹으로 묶을 수 있습니다. (기존 `AssistantManager.tsx` 확장)
- 그룹을 선택하면 "Group 모드"가 활성화됩니다.
- Group 모드에서는 아래와 같은 흐름으로 동작합니다:
    - Assistant가 `MultiAgentOrchestrator`라는 커스텀 Assistant로 전환됩니다.
    - Orchestrator는 6가지 도구 함수를 사용할 수 있습니다:
        1. **promptToUser(prompt: string)**
            - 사용자에게 추가 정보를 요청합니다.
            - `useChatContext()`의 `addMessage`로 메시지를 추가하고, 재귀 호출을 종료합니다.
        2. **switchAssistant(instruction: string)**
            - 다른 Assistant로 전환하고, 해당 Assistant에게 instruction을 전달합니다.
            - `useChatContext()`의 `submit`을 사용합니다.
        3. **setPlan(items: string[])**
            - 체크리스트 형태의 계획을 설정합니다.
            - 모든 항목이 완료되지 않은 상태에서 다시 호출되면, 아직 완료되지 않았음을 알립니다.
        4. **checkDone(index: number)**
            - 계획의 특정 항목을 완료로 표시합니다.
        5. **clearPlan()**
            - 계획을 취소하고 제거합니다.
        6. **reportResult(result_in_detail: string)**
            - 작업 결과를 사용자에게 정리해서 전달하고, 재귀를 종료합니다.
            - `addMessage`를 사용합니다.
- 활성화 시, `useAssistantContext`로 Assistant를 `MultiAgentOrchestrator`로 설정합니다. (setCurrentAssistant를 호출하면 Assistant에 설정된 localService 및 MCP를 자동 연결함)
- 사용자의 요청에 대해 위 6가지 도구로 대응합니다.
- `useChatContext`의 messages를 모니터링합니다.
    - 마지막 메시지가 tool_call이 아니고, `MultiAgentOrchestrator`가 보낸 메시지가 아니라면, 자동으로 `submit()`을 호출해 응답을 생성합니다.
- 이 과정은 재귀적으로 동작하며, 아래 함수가 호출되면 재귀가 종료됩니다:
    - `promptToUser()`
    - `reportResult()`

UI/UX

- Unlike the existing viewless component (`ToolCaller.tsx`), provide a simple view to check whether it is activated.
- This view is an expandable container, placed near the input box of `Chat.tsx`.
  - Ensure that the children of `Chat.tsx` are rendered naturally.

Considerations

- You must manage the state of which Assistant created each message.
  - All messages must include Assistant information (at least the name), and this should also be reflected in the prompt when making API calls.

Recursion Safety

- To prevent infinite recursion of the Orchestrator, set a maximum recursion depth or limit the number of iterations.
- Add logic to safely terminate recursion in exceptional cases (e.g., if an Assistant does not respond or the plan never completes).

Message Structure Clarification

- The following structure is recommended for each message: (Modify the one already defined in `./src/lib/ai-service.ts` as needed)

```
export interface StreamableMessage {
  id: string;
  content: string;
  assistantName: string; // Name of the Assistant that created the
message
  role: "user" | "assistant" | "system" | "tool";
  thinking?: string;
  isStreaming?: boolean;
  attachments?: { name: string; content: string }[];
  tool_calls?: {
    id: string;
    type: "function";
    function: { name: string; arguments: string };
  }[];
  tool_use?: { id: string; name: string; input: Record<string,
unknown> };
  function_call?: { name: string; arguments: Record<string, unknown>
};
  tool_call_id?: string;
}
```

- 메시지 생성 시 assistantName을 반드시 포함하고, 필요한 경우 isToolCall 등 추가 정보를 명확히 표기합니다.