React Native 项目手动实现 AG-UI 协议的基础模板代码

下面是一个**完整可运行的最小模板**，帮助你在 React Native（Expo 或 Bare Workflow 均可）中接入 AG-UI 协议的后端，实现：

- 实时流式接收 Agent 消息（打字机效果）
- 实时状态同步（STATE_DELTA）
- 工具调用进度显示
- Human-in-the-Loop 确认（REQUEST_HUMAN_INPUT）

1. 安装依赖

Bash

```
npm install react-native-sse eventsource   # SSE          react-native-sse
#       expo install expo-task-manager expo-background-fetch

npm install jsonpatch   #     JSON Patch  STATE_DELTA
```

2. 核心代码：AGUIClient.ts（可直接复制）

tsx

```
// src/utils/AGUIClient.ts
import EventSource from 'react-native-sse';
import { applyPatch } from 'jsonpatch';

type Message = {
  role: 'user' | 'assistant' | 'system';
  content: string;
};

type AgentState = Record<string, any>;  //

type AGUIEvent =
  | { type: 'TEXT_MESSAGE_CONTENT'; delta: string }
  | { type: 'MESSAGE_START'; messageId: string }
  | { type: 'MESSAGE_END'; messageId: string }
  | { type: 'STATE_DELTA'; patch: any[] }  // JSON Patch
  | { type: 'TOOL_CALL_START'; name: string; args: any }
  | { type: 'TOOL_CALL_END'; name: string; result: any }
  | { type: 'REQUEST_HUMAN_INPUT'; prompt: string; options?: string[] }
  | { type: 'THREAD_START' | 'THREAD_END' };

type Listener = {
  onMessageDelta: (delta: string) => void;
  onStateUpdate: (newState: AgentState) => void;
  onToolStart: (name: string, args: any) => void;
  onToolEnd: (name: string, result: any) => void;
```

```
  onHumanInputRequest: (prompt: string, options?: string[]) => Promise<any>;
  onError: (error: string) => void;
};

export class AGUIClient {
  private es: EventSource | null = null;
  private currentState: AgentState = {};
  private listeners: Partial<Listener> = {};

  constructor(private endpoint: string) {}

  setListeners(listeners: Partial<Listener>) {
    this.listeners = listeners;
  }

  //
  async sendMessage(userMessage: string, initialState: AgentState = {}) {
    this.currentState = initialState;

    const response = await fetch(this.endpoint, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        messages: [{ role: 'user', content: userMessage }],
        state: initialState,  //
      }),
    });

    if (!response.ok) {
      this.listeners.onError?.('       ');
      return;
    }

    //         SSE
    this.es = new EventSource(response.body!, { headers: response.headers });

    this.es.addEventListener('message', (event: any) => {
      if (event.data === '[DONE]') {
        this.es?.close();
        return;
      }

      try {
        const data: AGUIEvent = JSON.parse(event.data);
        this.handleEvent(data);
      } catch (e) {
```

```
        console.error('          ', e);
      }
    });

    this.es.addEventListener('error', (err) => {
      this.listeners.onError?.('       ');
      this.es?.close();
    });
  }

  private handleEvent(event: AGUIEvent) {
    switch (event.type) {
      case 'TEXT_MESSAGE_CONTENT':
        this.listeners.onMessageDelta?.(event.delta);
        break;

      case 'STATE_DELTA':
        this.currentState = applyPatch(this.currentState, event.patch).doc;
        this.listeners.onStateUpdate?.(this.currentState);
        break;

      case 'TOOL_CALL_START':
        this.listeners.onToolStart?.(event.name, event.args);
        break;

      case 'TOOL_CALL_END':
        this.listeners.onToolEnd?.(event.name, event.result);
        break;

      case 'REQUEST_HUMAN_INPUT':
        //               Promise
        this.listeners.onHumanInputRequest?.(event.prompt,
event.options).then((response) => {
          //                        POST /input
          fetch(this.endpoint + '/input', {
            method: 'POST',
            body: JSON.stringify({ response }),
          });
        });
        break;
    }
  }

  close() {
    this.es?.close();
```

```
  }
}
```

3. 使用示例：聊天页面（ChatScreen.tsx）

tsx

```
// src/screens/ChatScreen.tsx
import React, { useState, useRef } from 'react';
import { View, TextInput, Button, Text, ScrollView, Alert } from 'react-
native';
import { AGUIClient } from '../utils/AGUIClient';

export default function ChatScreen() {
  const [input, setInput] = useState('');
  const [messages, setMessages] = useState<string[]>([]);
  const [currentAssistantMsg, setCurrentAssistantMsg] = useState('');
  const [agentState, setAgentState] = useState<any>({});
  const [isLoading, setIsLoading] = useState(false);
  const scrollViewRef = useRef<ScrollView>(null);

  const client = useRef(new AGUIClient('https://your-
backend.com/api/copilotkit')).current;

  React.useEffect(() => {
    client.setListeners({
      onMessageDelta: (delta) => {
        setCurrentAssistantMsg(prev => prev + delta);
      },
      onStateUpdate: (newState) => {
        setAgentState(newState);
        //      newState.itinerary?.map(...)
      },
      onToolStart: (name) => {
        setMessages(prev => [...prev, `🔧            : ${name}`]);
      },
      onToolEnd: (name, result) => {
        setMessages(prev => [...prev, `✅ ${name}     :
${JSON.stringify(result).slice(0,100)}`]);
      },
      onHumanInputRequest: async (prompt) => {
        return new Promise((resolve) => {
          Alert.alert('       ', prompt, [
            { text: '    ', style: 'cancel', onPress: () => resolve({
approved: false }) },
            { text: '    ', onPress: () => resolve({ approved: true }) },
          ]);
```

```
        });
      },
      onError: (err) => Alert.alert('    ', err),
    });
  }, []);

  const send = () => {
    if (!input.trim() || isLoading) return;

    setMessages(prev => [...prev, `    : ${input}`]);
    setMessages(prev => [...prev, '    : ']);
    setCurrentAssistantMsg('');
    setIsLoading(true);

    client.sendMessage(input, { itinerary: [] }).then(() => {
      setIsLoading(false);
      //
      if (currentAssistantMsg) {
        setMessages(prev => {
          const newMsgs = [...prev];
          newMsgs[newMsgs.length - 1] += currentAssistantMsg;
          return newMsgs;
        });
        setCurrentAssistantMsg('');
      }
      setInput('');
    });
  };

  return (
    <View style={{ flex: 1, padding: 10 }}>
      <ScrollView ref={scrollViewRef} onContentSizeChange={() =>
scrollViewRef.current?.scrollToEnd()}>
        {messages.map((msg, i) => <Text key={i} style={{ marginVertical: 4
}}>{msg}</Text>)}
        {currentAssistantMsg ? <Text>    : {currentAssistantMsg}</Text> :
null}
        <Text style={{ marginTop: 20, color: 'green' }}>
                : {JSON.stringify(agentState, null, 2)}
        </Text>
      </ScrollView>

      <View style={{ flexDirection: 'row', marginTop: 10 }}>
        <TextInput
          style={{ flex: 1, borderWidth: 1, borderRadius: 8, padding: 10 }}
          value={input}
```

```
          onChangeText={setInput}
          placeholder="       ..."
        />
        <Button title="    " onPress={send} disabled={isLoading} />
      </View>
    </View>
  );
}
```

后续扩展建议

- 用 Gifted Chat 或 react-native-chat-library 美化聊天界面。
- 为 STATE_DELTA 定义强类型接口。
- 支持图片/文件上传（后端需支持）。
- 添加加载动画、重连机制。

这个模板已经可以直接跑起来连接你的 AG-UI 后端（LangGraph、CopilotKit Runtime 等）。把 endpoint 换成你的实际地址，就能看到实时流式输出和状态更新了！

需要我帮你加多 Agent 协调、生成式 UI 渲染，或者适配 Expo？随时说！🚀