Microsoft

Generative AI with JavaScript

Streaming generative AI output with the AI Chat Protocol

Agenda

- Streaming GenAI output
- Introducing the AI Chat Protocol
- Usage Sample

Streaming GenAI output

Why use streaming?

Why use streaming?

Streaming is the expectation for GenAI apps

- Reduced latency
- Enhanced user experience

Why use streaming?

Streaming is the expectation for GenAI apps

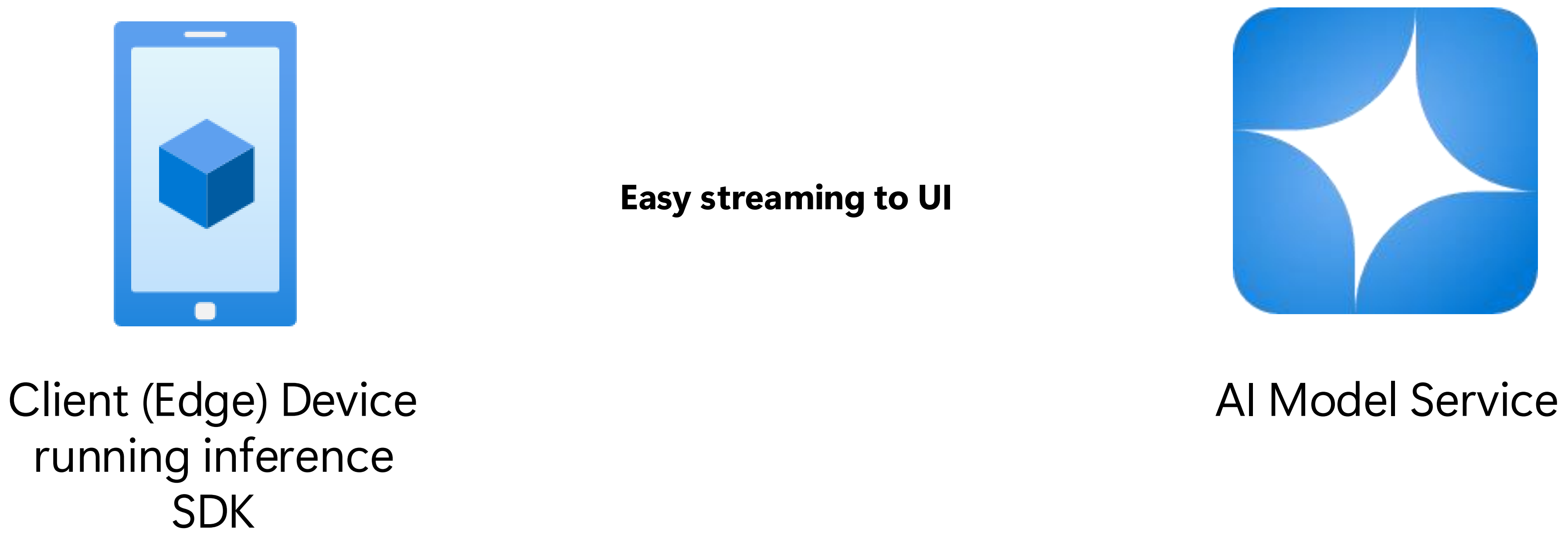
- Reduced latency
- Enhanced user experience

2 options for implementing

- Use a GenAI Inference SDK directly in the browser
- Use an AI inference server to stream to your client

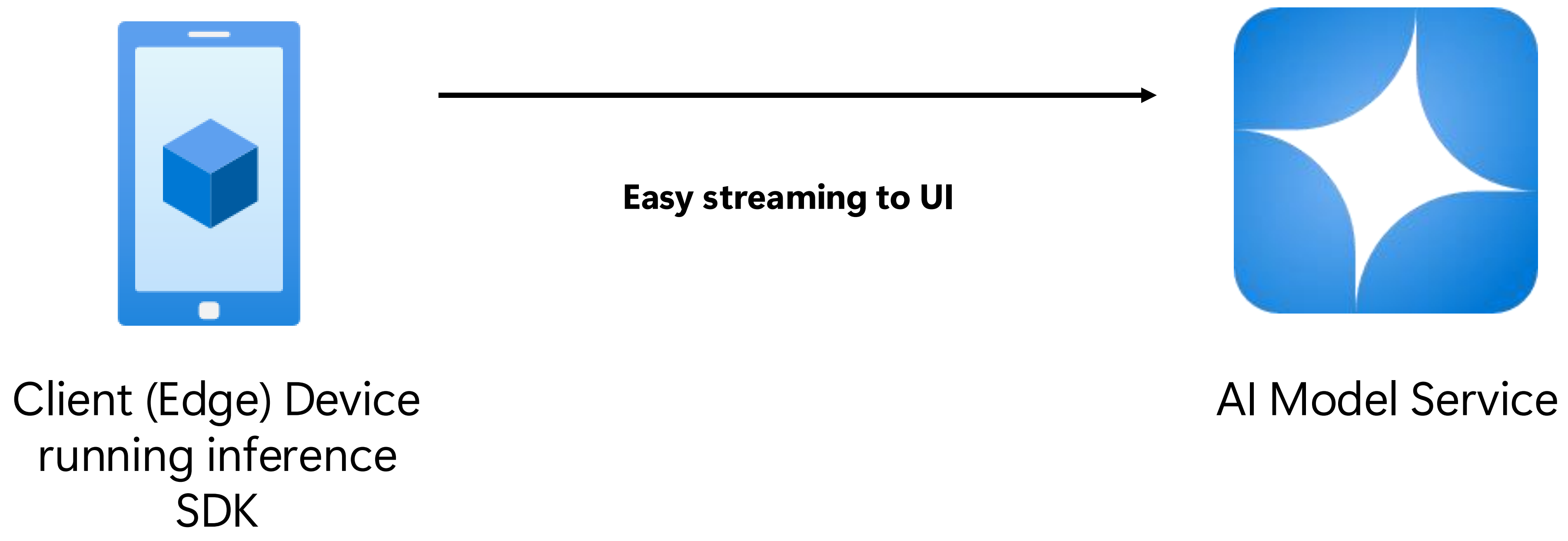
Streaming - Inference in Browser

- The simplest approach to streaming is by using the SDK in the edge device
- For example, using Azure OpenAI SDK for JS in your frontend browser code



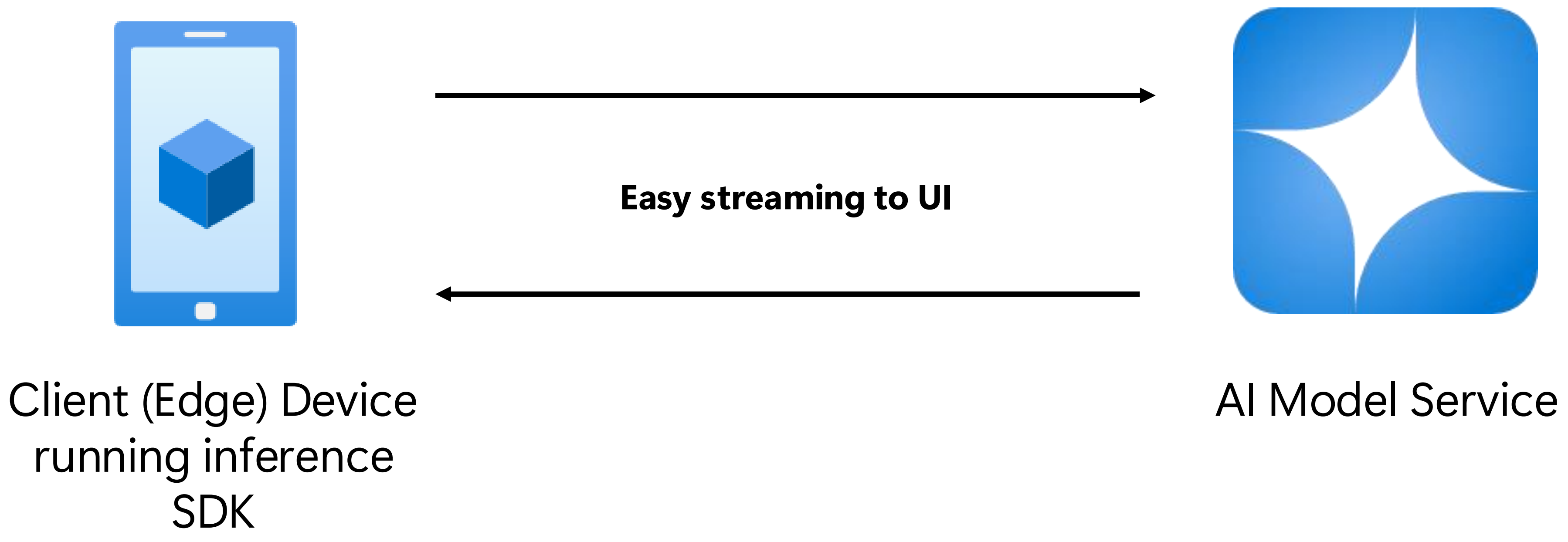
Streaming - Inference in Browser

- The simplest approach to streaming is by using the SDK in the edge device
- For example, using Azure OpenAI SDK for JS in your frontend browser code



Streaming - Inference in Browser

- The simplest approach to streaming is by using the SDK in the edge device
- For example, using Azure OpenAI SDK for JS in your frontend browser code



Browser-side inference is unsafe and not best practice

Browser-side inference is unsafe and not best practice

Security risks

- API key exposure
- No data sanitization of input
- No integration with data compliance

Browser-side inference is unsafe and not best practice

Security risks

- API key exposure
- No data sanitization of input
- No integration with data compliance

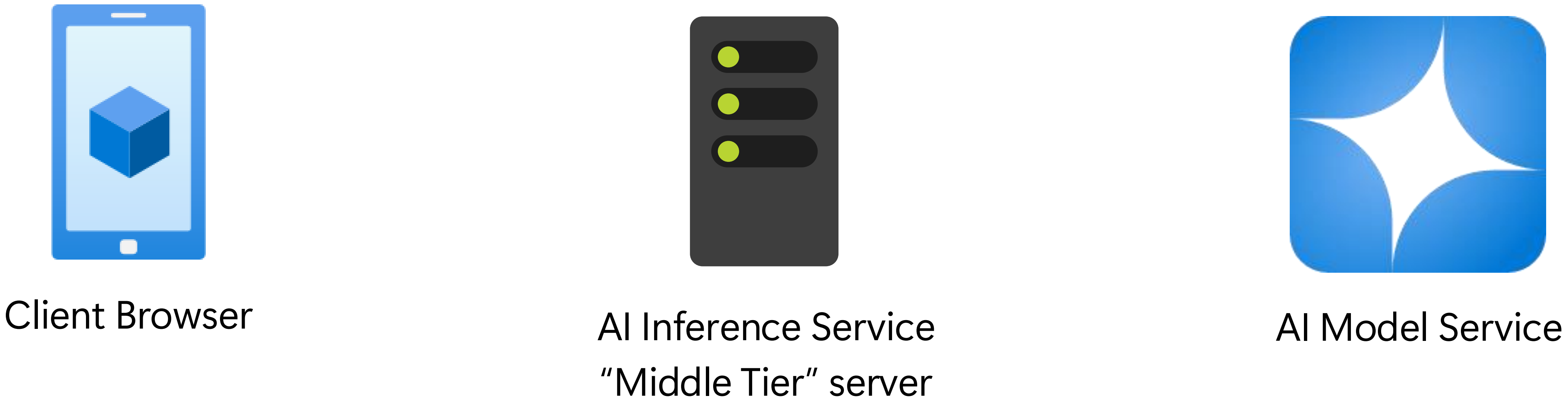
Application limitations

- No rate limit/quota handling
- No caching for performance
- No integration with business logic
- No handling of excess data to AI Model service ($$)

Streaming – AI Inference Service

A server sits in between the user's device and the AI Model Service

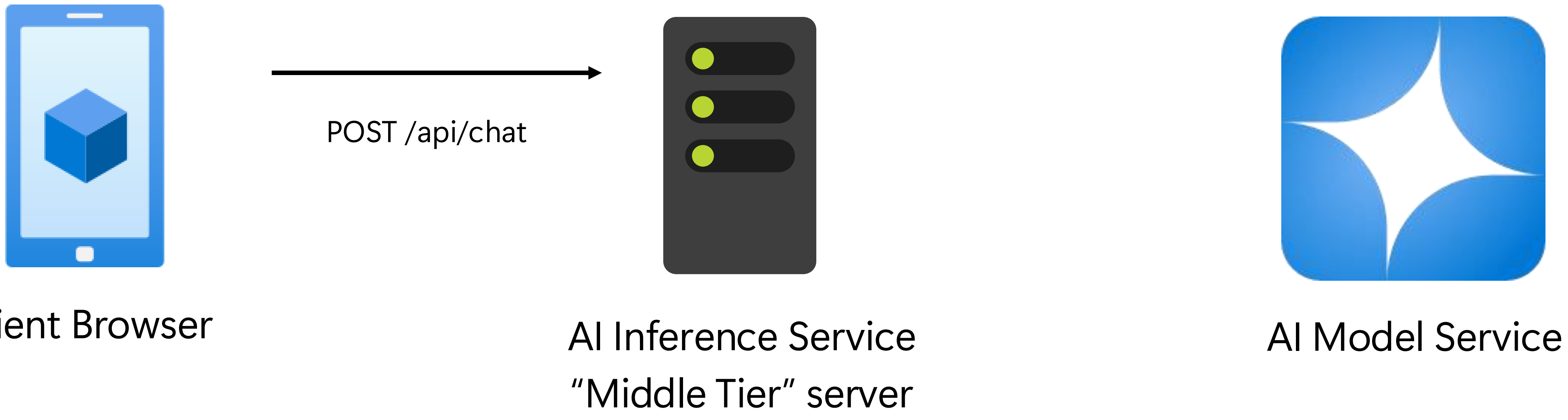
- Clean separation of concerns with AI inference and backend logic
- **Added challenge**: streaming to the client/UI is now tricky



Streaming – AI Inference Service

A server sits in between the user's device and the AI Model Service

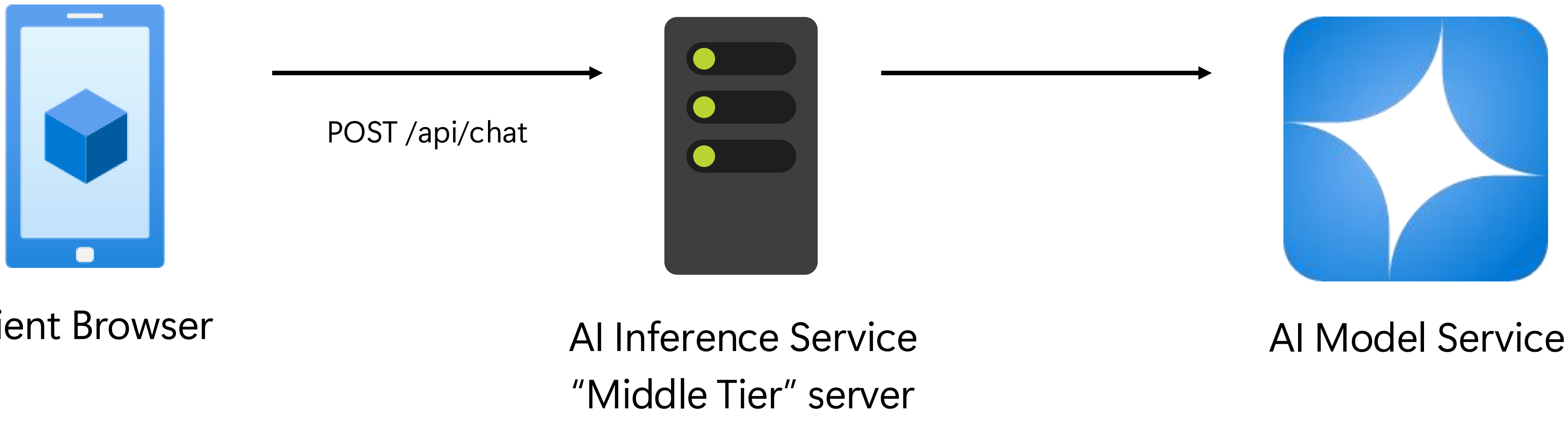
- Clean separation of concerns with AI inference and backend logic
- **Added challenge**: streaming to the client/UI is now tricky



Streaming – AI Inference Service

A server sits in between the user's device and the AI Model Service

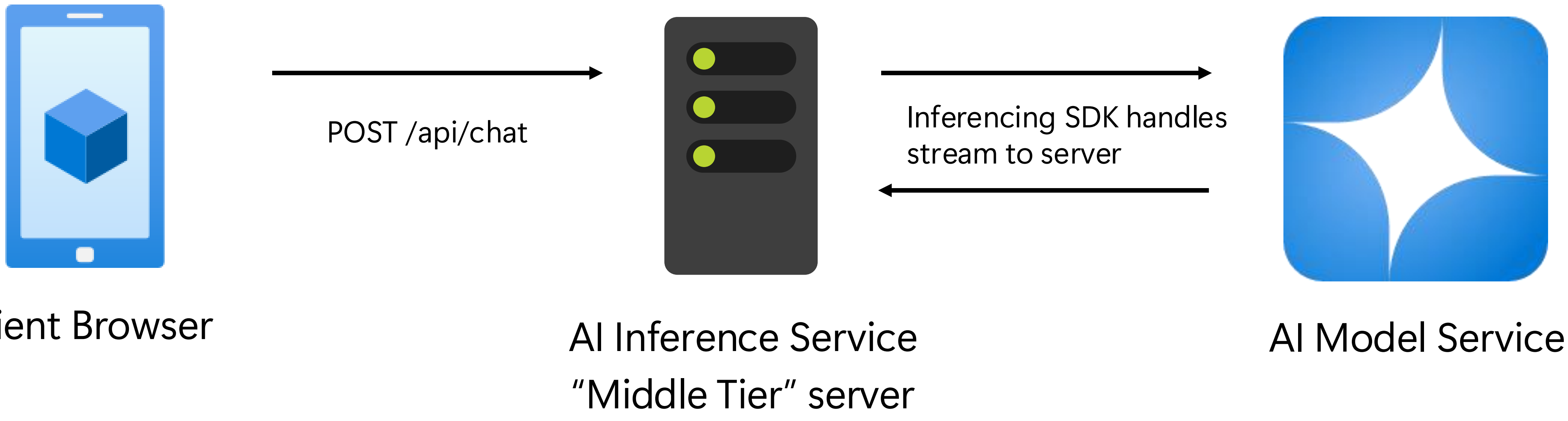
- Clean separation of concerns with AI inference and backend logic
- **Added challenge**: streaming to the client/UI is now tricky



Streaming – AI Inference Service

A server sits in between the user's device and the AI Model Service

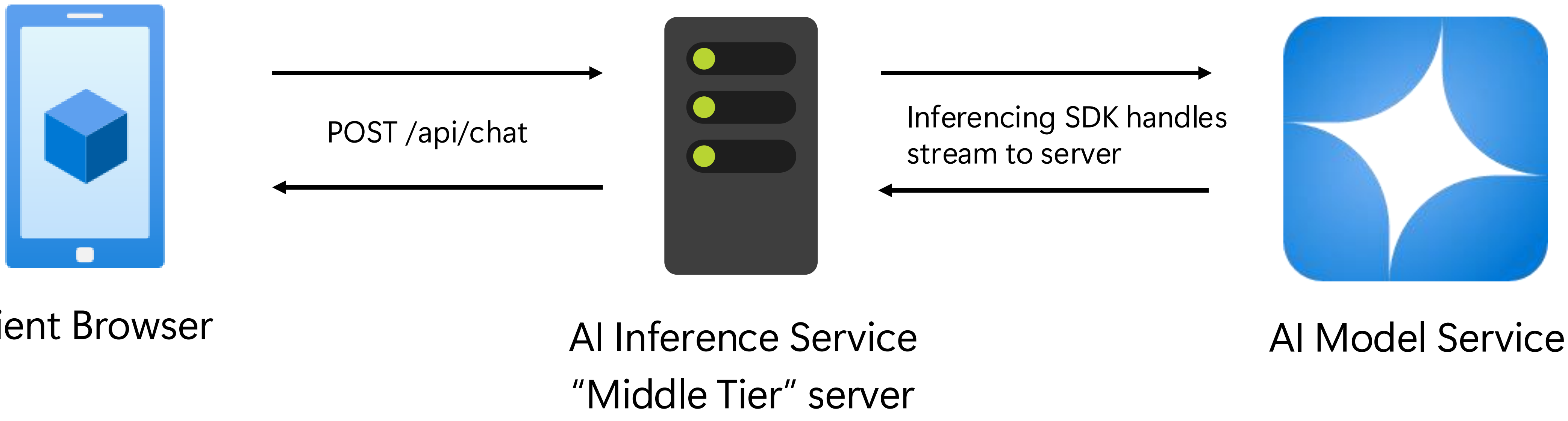
- Clean separation of concerns with AI inference and backend logic
- **Added challenge**: streaming to the client/UI is now tricky


POST /api/chat
Inferencing SDK handles stream to server
Client Browser
AI Inference Service "Middle Tier" server
AI Model Service

Streaming – AI Inference Service

A server sits in between the user's device and the AI Model Service

- Clean separation of concerns with AI inference and backend logic
- **Added challenge**: streaming to the client/UI is now tricky



Streaming is hard!

Streaming is hard!

1. Select a network protocol for streaming (HTTP, WebSockets, SSE…)

Streaming is hard!

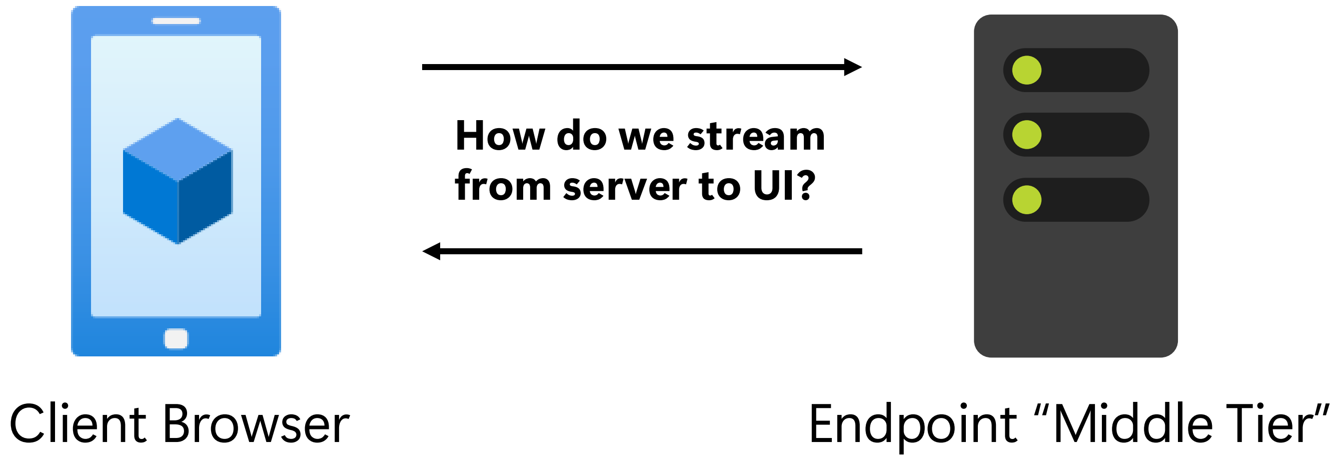
1. Select a network protocol for streaming (HTTP, WebSockets, SSE…)
2. Define an exchange format

Streaming is hard!

1. Select a network protocol for streaming (HTTP, WebSockets, SSE…)
2. Define an exchange format
3. Parse the output (hundreds of lines of browser-side JS to write)

Streaming is hard!

1. Select a network protocol for streaming (HTTP, WebSockets, SSE…)
2. Define an exchange format
3. Parse the output (hundreds of lines of browser-side JS to write)



Introducing the AI Chat Protocol

Streaming to UI with AI Chat Protocol

The **AI Chat Protocol** (JS) is designed to make streaming easy

Streaming to UI with AI Chat Protocol

The **AI Chat Protocol** (JS) is designed to make streaming easy

- 2 Requirements:
  - Server uses API Specification
  - Frontend uses lightweight parsing SDK

Streaming to UI with AI Chat Protocol

The **AI Chat Protocol** (JS) is designed to make streaming easy

- 2 Requirements:
  - Server uses API Specification
  - Frontend uses lightweight parsing SDK
- One client - `AIChatProtocolClient`

Streaming to UI with AI Chat Protocol

The **AI Chat Protocol** (JS) is designed to make streaming easy

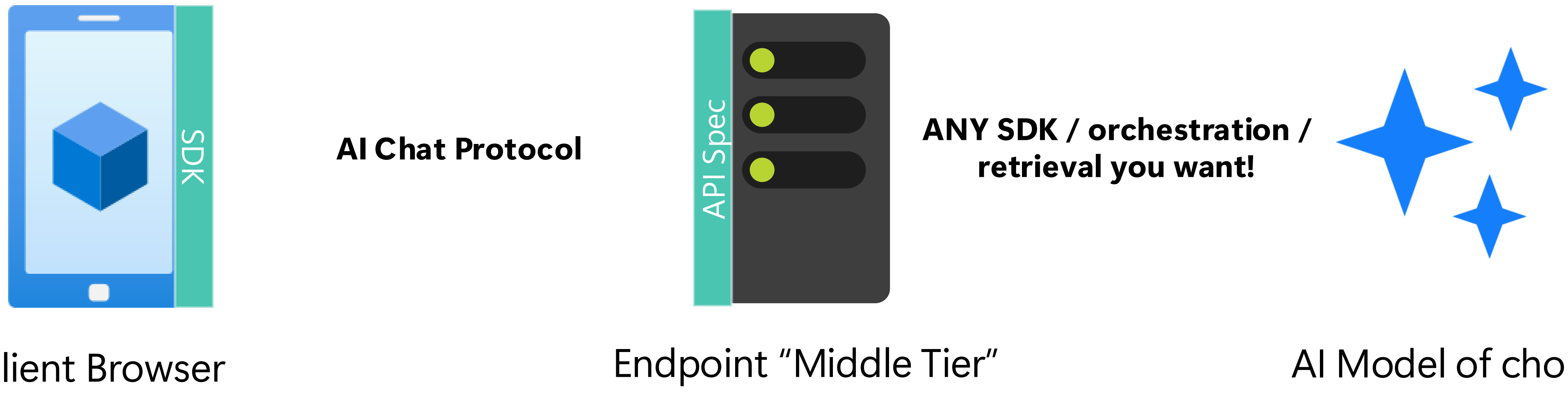
- 2 Requirements:
  - Server uses API Specification
  - Frontend uses lightweight parsing SDK
- One client - `AIChatProtocolClient`
- Two methods: `getCompletion + getStreamedCompletion`
  - Now you have easy streaming along with TypeScript models!
  - Also includes a flexible 'context' options bag for any other data needed

"For our sample, I would roughly estimate it to remove something between 200-400 lines of code, maybe a bit more because of the parser"

Yohan Lasorsa, Cloud Advocate, Microsoft

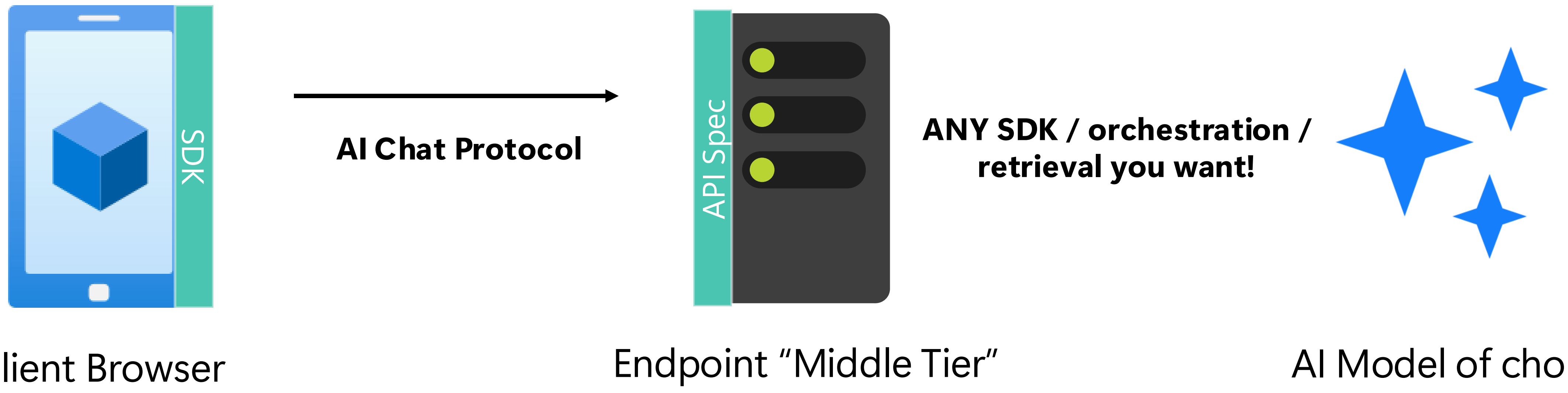
Building with the AI Chat Protocol

- **Server-to-client only** design pattern, not server-to-AI model
- AI backend choices are flexible
  - Any orchestration, RAG option, programming language, and language model



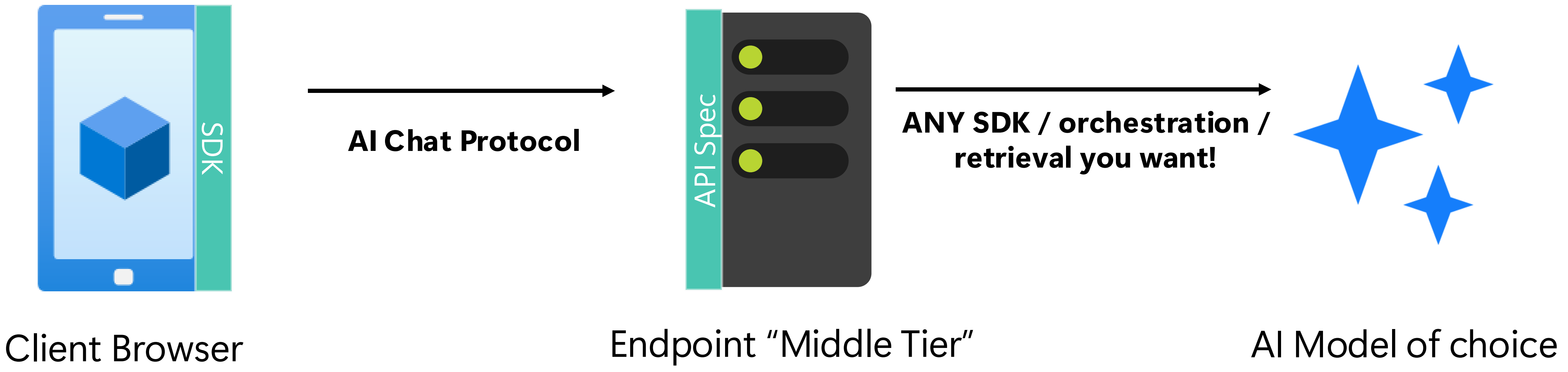
Building with the AI Chat Protocol

- **Server-to-client only** design pattern, not server-to-AI model
- AI backend choices are flexible
  - Any orchestration, RAG option, programming language, and language model



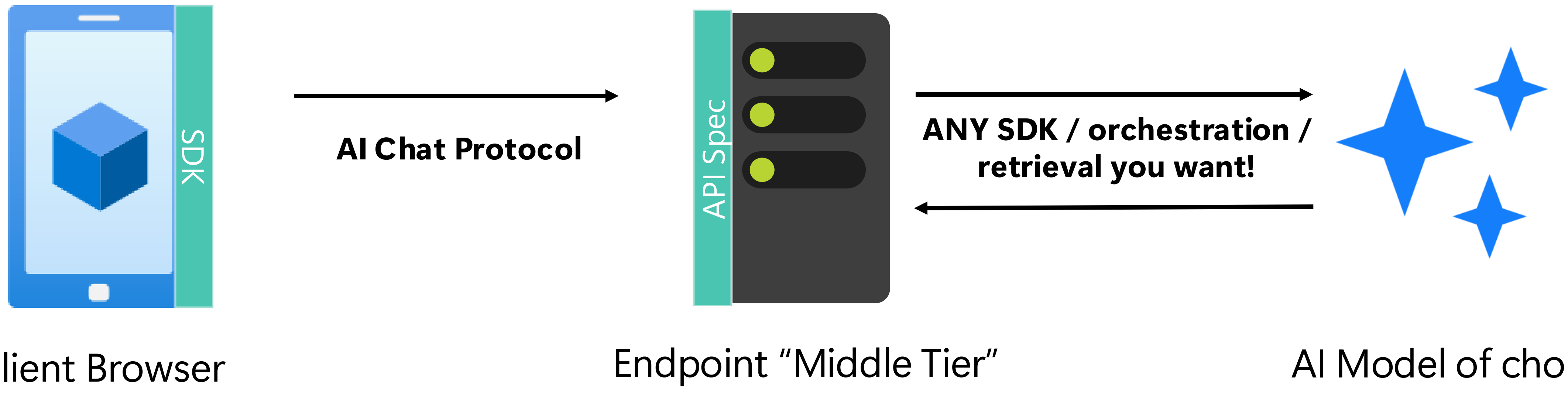
Building with the AI Chat Protocol

- **Server-to-client only** design pattern, not server-to-AI model
- AI backend choices are flexible
  - Any orchestration, RAG option, programming language, and language model



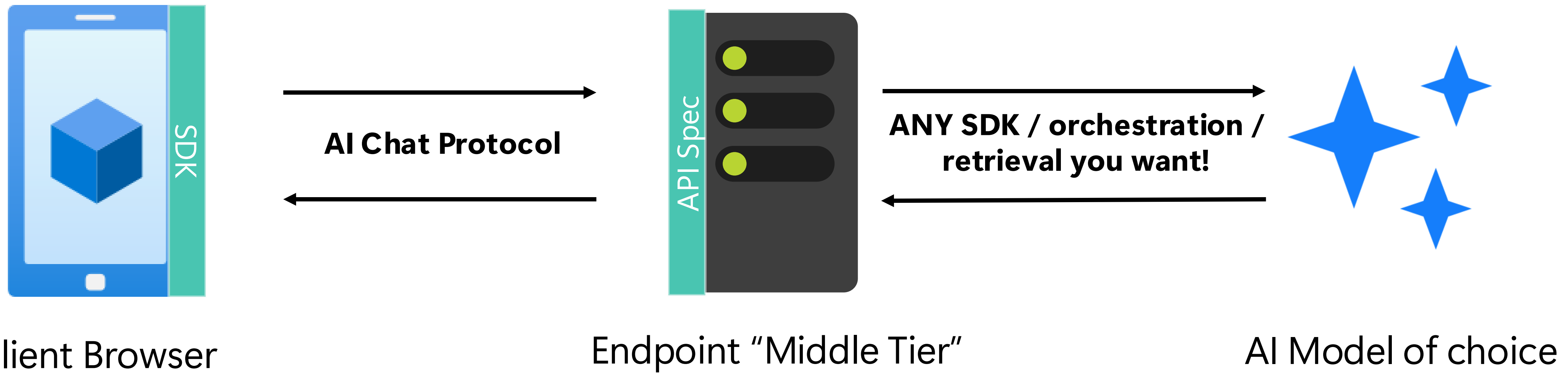
Building with the AI Chat Protocol

- **Server-to-client only** design pattern, not server-to-AI model
- AI backend choices are flexible
    - Any orchestration, RAG option, programming language, and language model

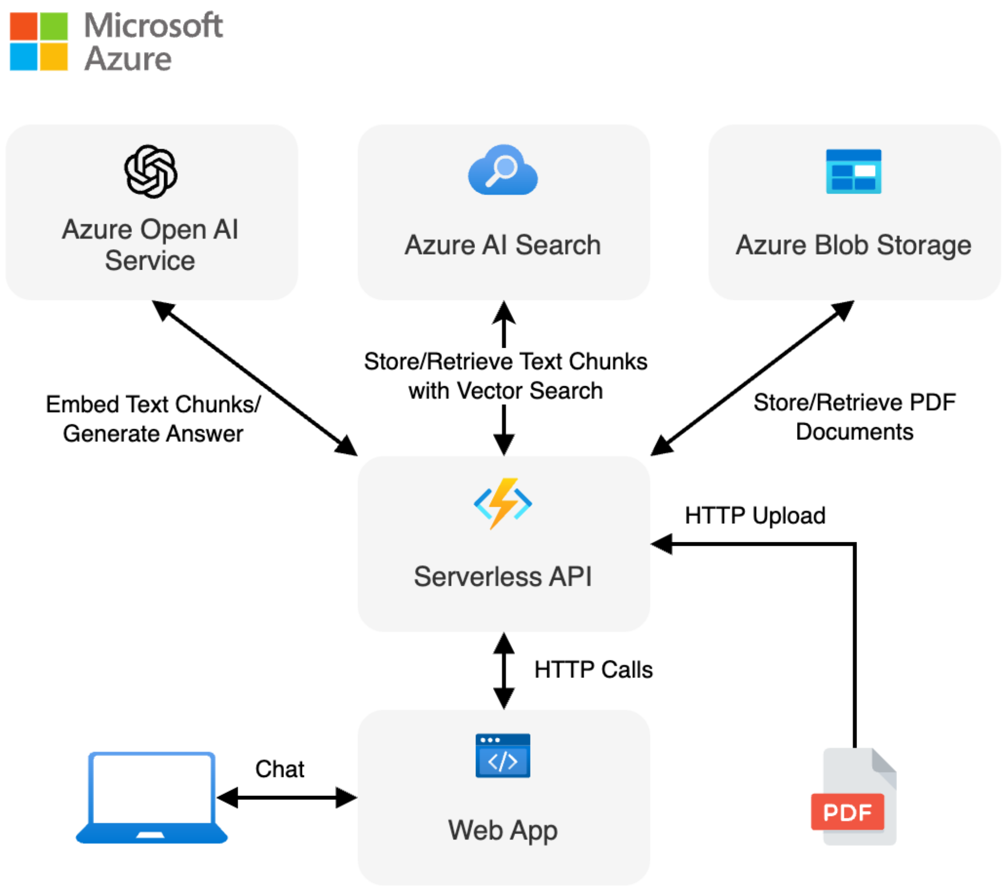


Building with the AI Chat Protocol

- **Server-to-client only** design pattern, not server-to-AI model
- AI backend choices are flexible
  - Any orchestration, RAG option, programming language, and language model

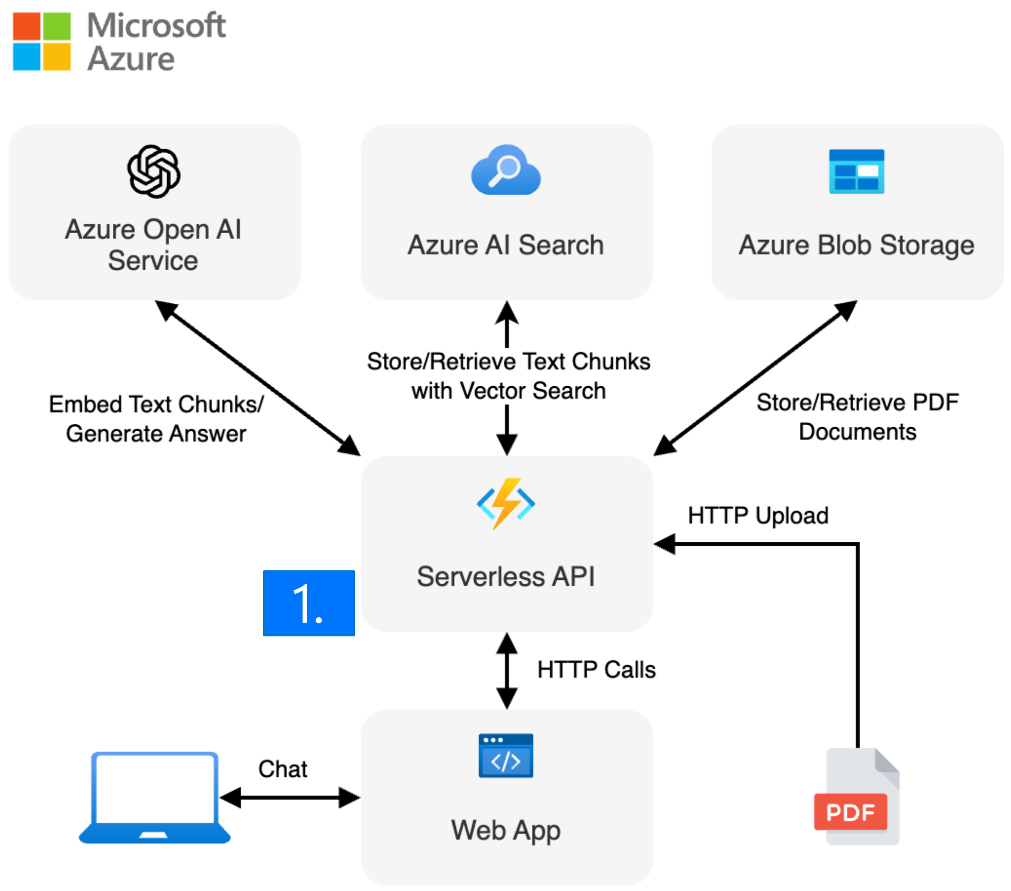


Adding the AI Chat Protocol



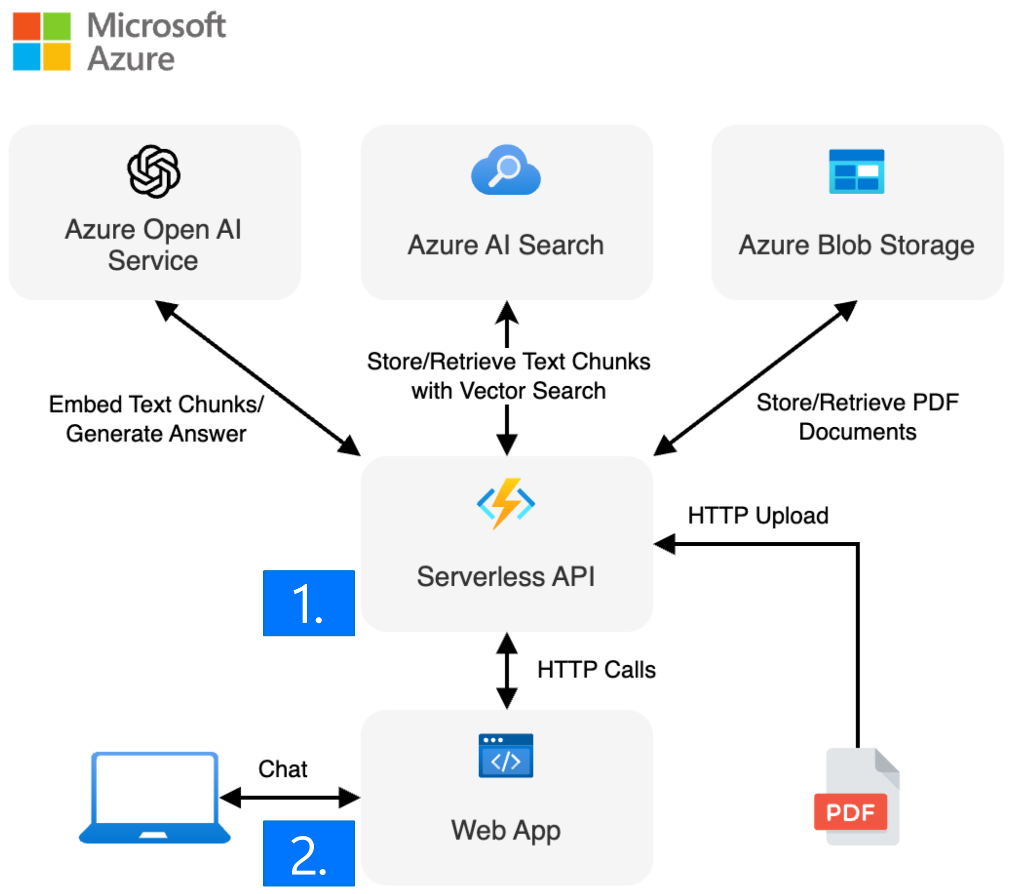
Adding the AI Chat Protocol

1. Ensure service API conforms to Chat Protocol



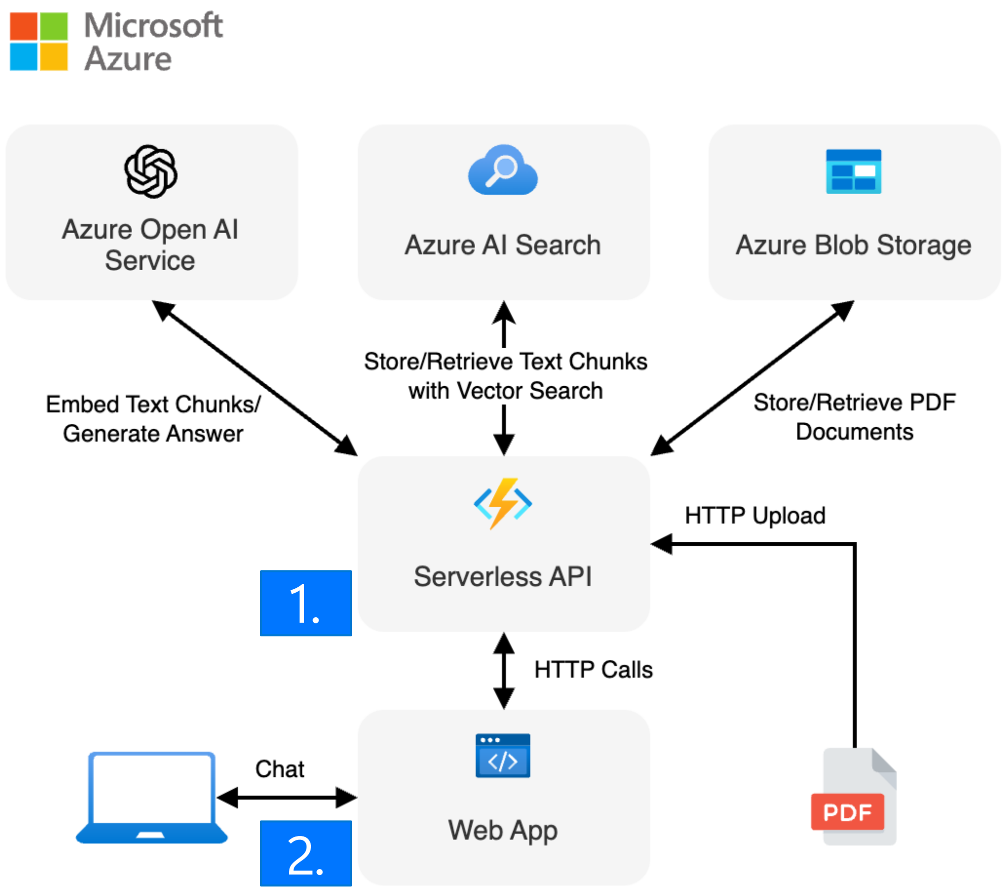
Adding the AI Chat Protocol

1. Ensure service API conforms to Chat Protocol
2. Incorporate SDK into frontend code



Adding the AI Chat Protocol

1. Ensure service API conforms to Chat Protocol
2. Incorporate SDK into frontend code
3. Enjoy streamed output to the client 🙂

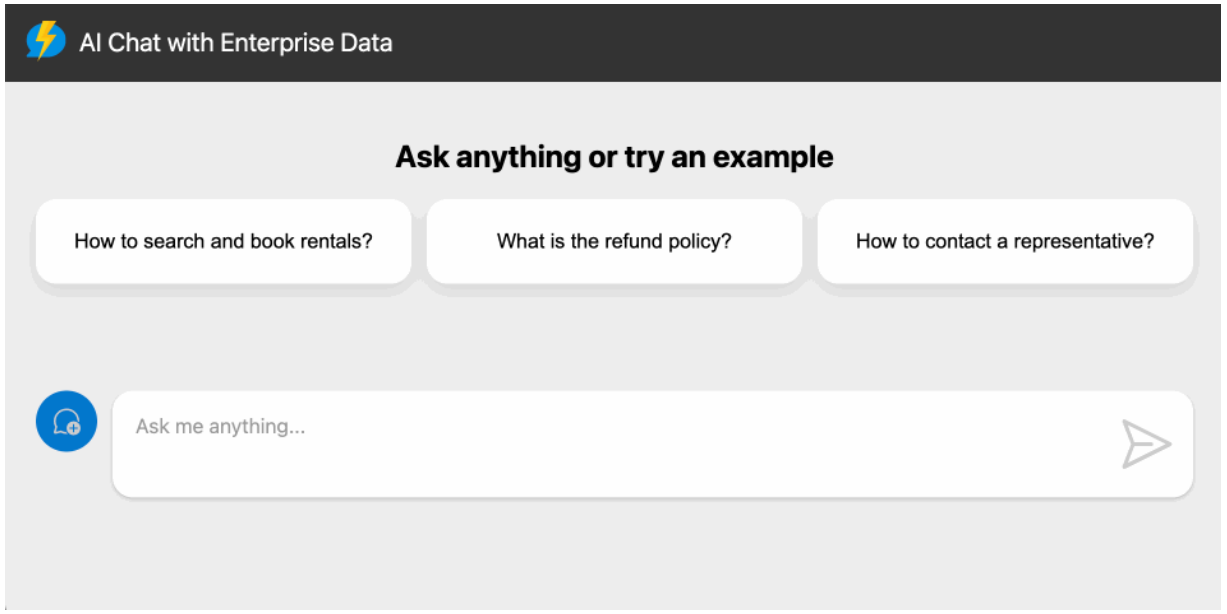


Example integration

Sample Code

Serverless RAG with Langchain.js Sample

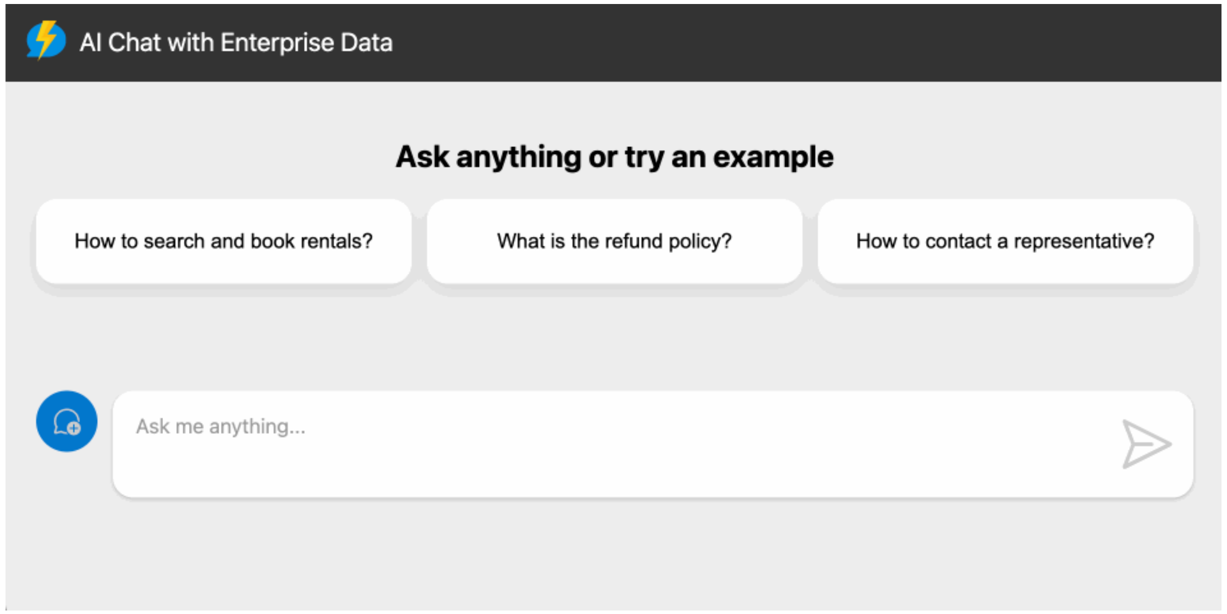
Sample uses the AI Chat Protocol for streaming to UI


AI Chat with Enterprise Data
Ask anything or try an example
How to search and book rentals?
What is the refund policy?
How to contact a representative?
Ask me anything...

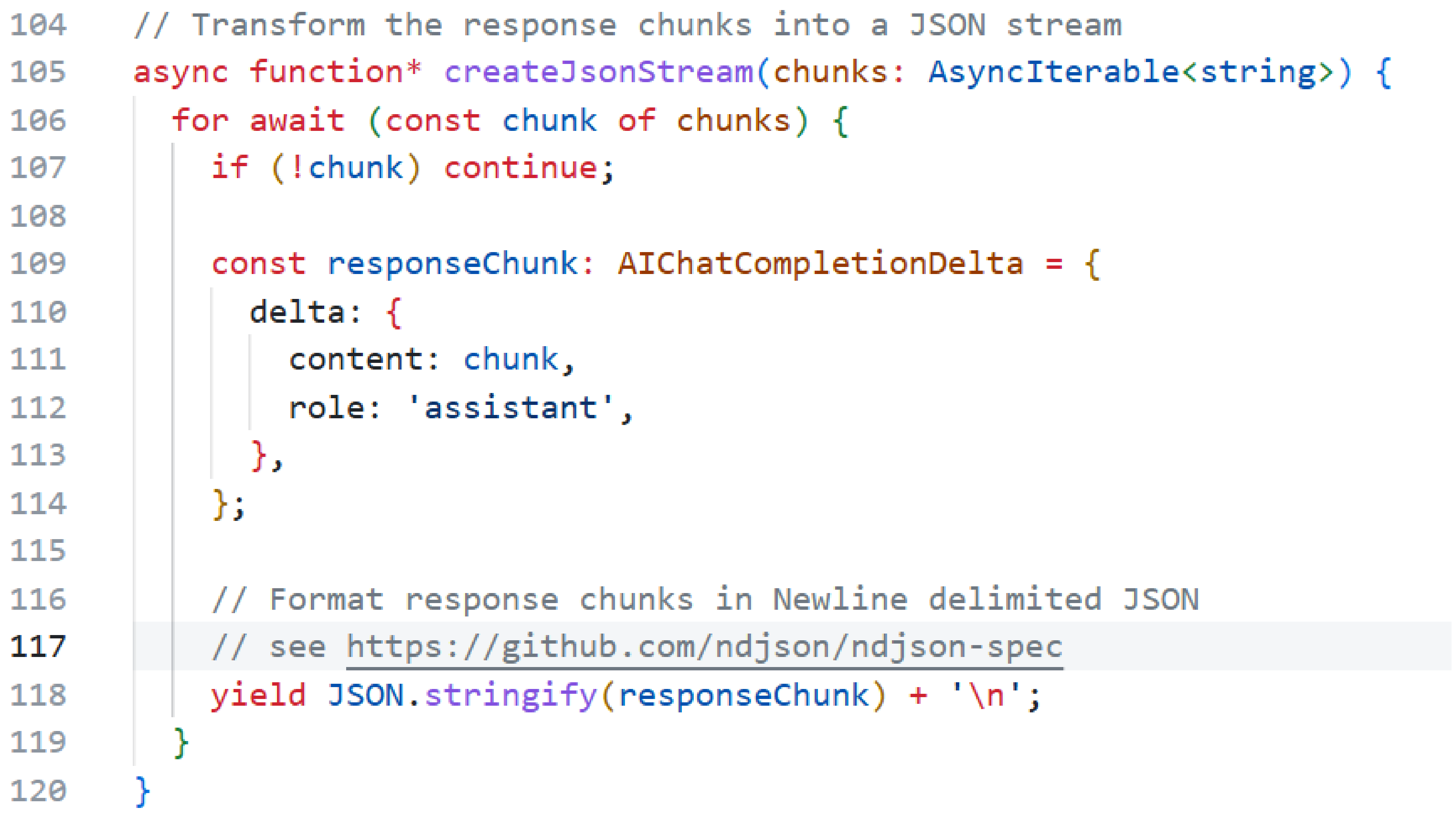
Serverless RAG with Langchain.js Sample

Sample uses the AI Chat Protocol for streaming to UI

- 2 core pieces:
  - RAG Service API – Conforms to AI Chat Protocol API Spec
  - Web App – Uses AI Chat Protocol library


AI Chat with Enterprise Data
Ask anything or try an example
How to search and book rentals?
What is the refund policy?
How to contact a representative?
Ask me anything...

RAG Service API – ndjson Stream



```typescript
// Transform the response chunks into a JSON stream
async function* createJsonStream(chunks: AsyncIterable<string>) {
  for await (const chunk of chunks) {
    if (!chunk) continue;

    const responseChunk: AIChatCompletionDelta = {
      delta: {
        content: chunk,
        role: 'assistant',
      },
    };

    // Format response chunks in Newline delimited JSON
    // see https://github.com/ndjson/ndjson-spec
    yield JSON.stringify(responseChunk) + '\n';
  }
}
```

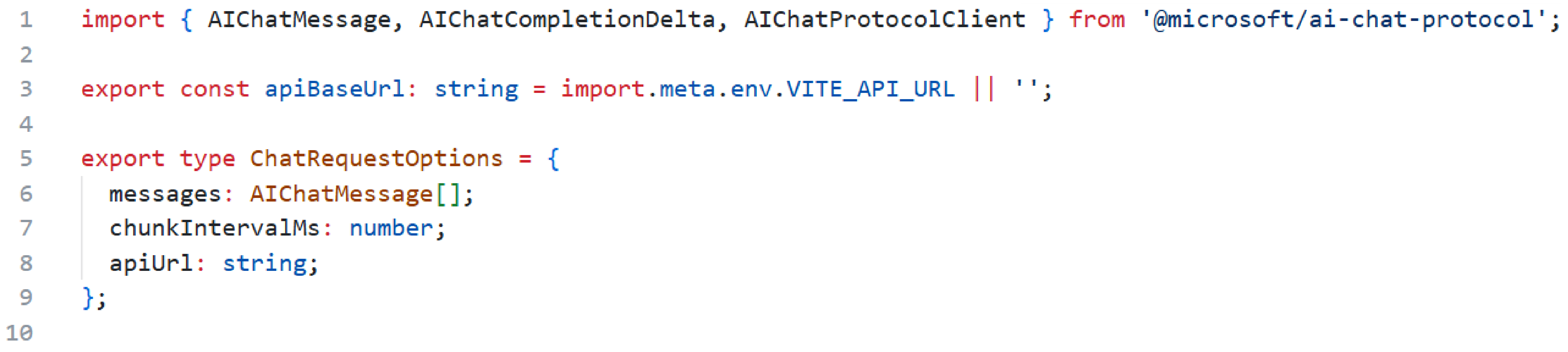
RAG Service API – AI Chat Protocol Spec

```
    const responseStream = await ragChain.stream({
      input: question,
      context: await retriever.invoke(question),
    });
    const jsonStream = Readable.from(createJsonStream(responseStream));

    return data(jsonStream, {
      'Content-Type': 'application/x-ndjson',
      'Transfer-Encoding': 'chunked',
    });
  } catch (_error: unknown) {
    const error = _error as Error;
    context.error(`Error when processing chat-post request: ${error.message}`);

    return serviceUnavailable('Service temporarily unavailable. Please try again later.');
  }
}
```

Web App – AI Chat Protocol Library



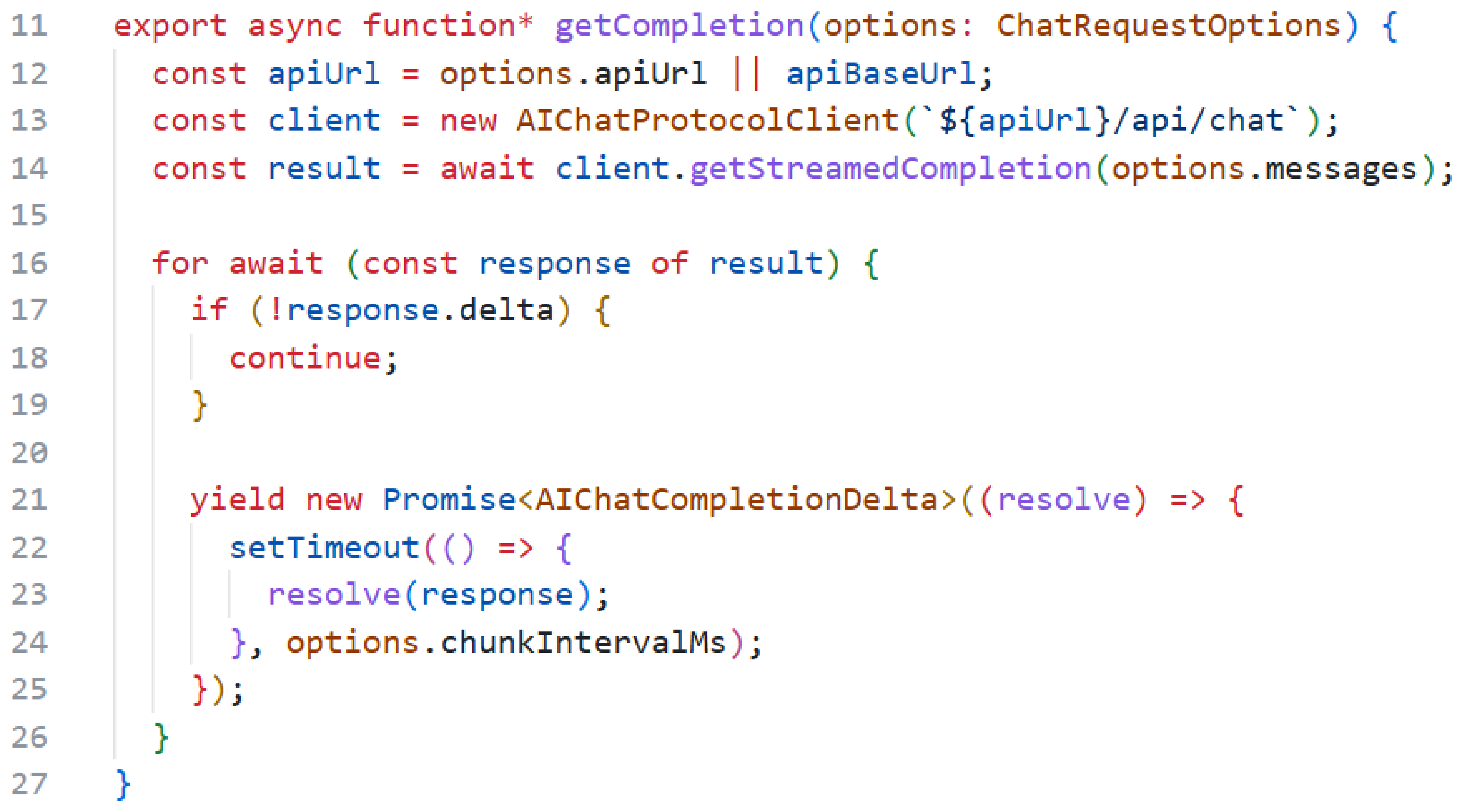
```typescript
import { AIChatMessage, AIChatCompletionDelta, AIChatProtocolClient } from '@microsoft/ai-chat-protocol';

export const apiBaseUrl: string = import.meta.env.VITE_API_URL || '';

export type ChatRequestOptions = {
  messages: AIChatMessage[];
  chunkIntervalMs: number;
  apiUrl: string;
};

```

Web App – AI Chat Protocol Library

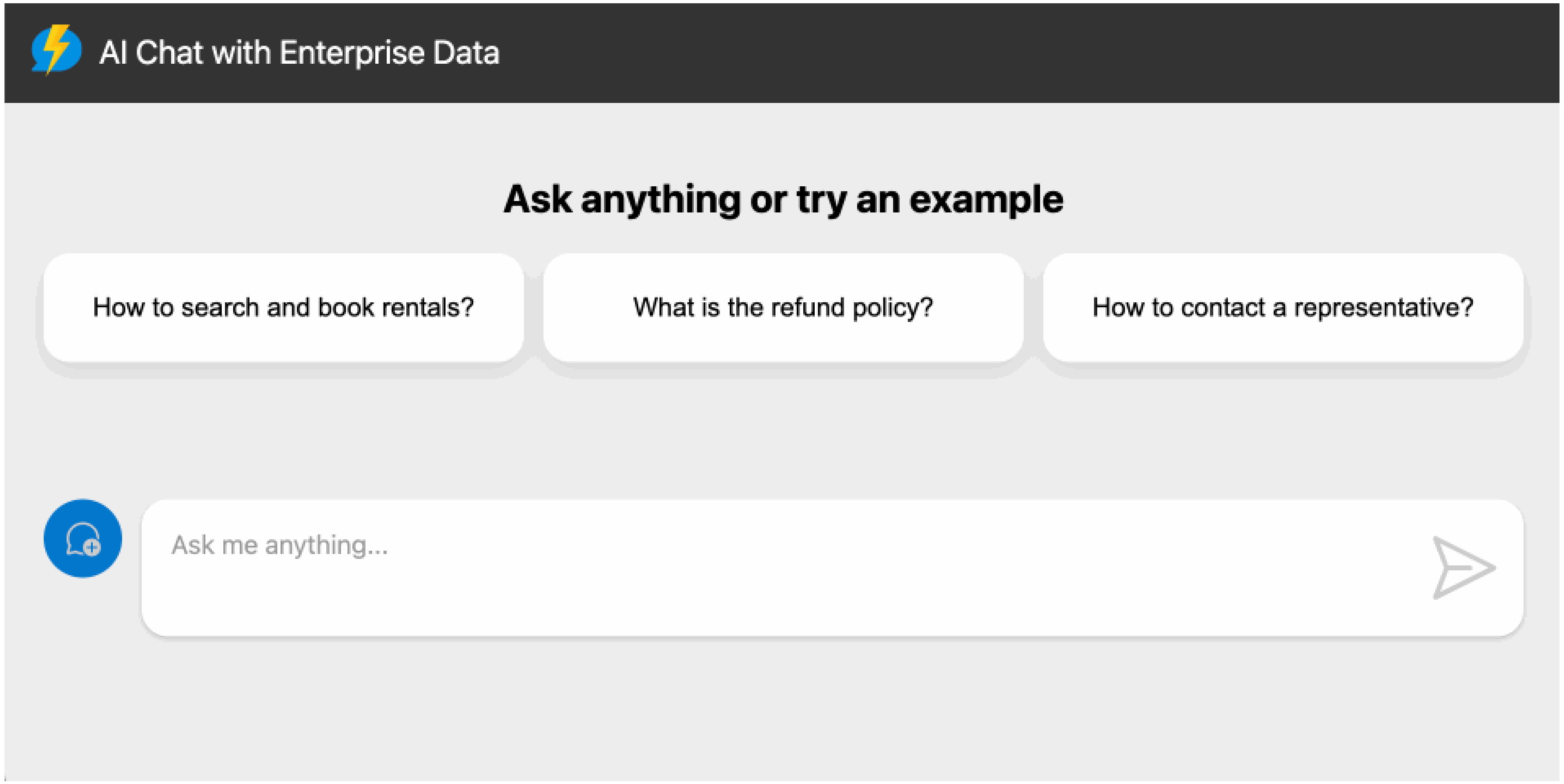


```
export async function* getCompletion(options: ChatRequestOptions) {
  const apiUrl = options.apiUrl || apiBaseUrl;
  const client = new AIChatProtocolClient(`${apiUrl}/api/chat`);
  const result = await client.getStreamedCompletion(options.messages);

  for await (const response of result) {
    if (!response.delta) {
      continue;
    }

    yield new Promise<AIChatCompletionDelta>((resolve) => {
      setTimeout(() => {
        resolve(response);
      }, options.chunkIntervalMs);
    });
  }
}
```

Easy streaming with context!


AI Chat with Enterprise Data

Ask anything or try an example

How to search and book rentals?

What is the refund policy?

How to contact a representative?

Ask me anything...

Microsoft

Resources

• AI Chat Protocol GitHub	aka.ms/aichat
• AI Chat Protocol API Spec	aka.ms/chatprotocol
• Serverless AI Chat sample	aka.ms/ai/js/chat

Microsoft